Object Segmentation Using Graph Cuts Based Active Contours*

Ning Xu
Beckman Institute & ECE Dept.
University of Illinois
Urbana, IL, USA
ningxu@vision.ai.uiuc.edu

Ravi Bansal
Department of Psychiatry
Columbia University
New York, NY, USA
bansalr@childpsych.columbia.edu

Narendra Ahuja
Beckman Institute & ECE Dept.
University of Illinois
Urbana, IL, USA
ahuja@vision.ai.uiuc.edu

Abstract

In this paper we present a graph cuts based active contours (GCBAC) approach to object segmentation problems. Our method is a combination of active contours and the optimization tool of graph cuts and differs fundamentally from traditional active contours in that it uses graph cuts to iteratively deform the contour. Consequently, it has the following advantages. (1). It has the ability to jump over local minima and provide a more global result. (2). Graph cuts guarantee continuity and lead to smooth contours free of self-crossing and uneven spacing problems. Therefore, the internal force which is commonly used in traditional energy functions to control the smoothness is no longer needed, and hence the number of parameters is greatly reduced. (3). Our approach easily extends to the segmentation of three and higher dimensional objects. In addition, the algorithm is suitable for interactive correction and is shown to always converge. Experimental results and analyses are provided.

1 Introduction

A natural and popular formulation of object segmentation is in terms of energy minimization of certain objective functions. For example, the frame work of active contours [13, 6] minimizes the contour energy E defined as the sum of external energy and internal energy. The external energy pulls the contours towards desired image features while the internal energy helps achieve smooth boundaries. An early implementation of active contours, called Snakes, is based on deforming an initial contour at a number of control points selected along a given initial contour. The deformation is directed towards the object boundary by minimizing the energy E so that its local minimum occurs at the boundary of the object. This implementation has several disadvantages. First, the Snakes approach the nearest local minimum of the initial contour and are therefore prone to finding a local minimum which in general does not coincide with the object contour. This leads to sensitivity to initialization, e.g. when there are a large number of local minima near the initial contour due to image noise or background clutter. Second, the discretization of the contours into a number of control points may cause problems with uneven spacing and self-crossing while the contours are deforming, and make it difficult to extend the approach to segment 3D objects. Finally, automatic selection of various parameters such as the weights in the energy function is still an open problem.

Many approaches have been proposed to improve the robustness and stability of Snakes. An inflation force is defined on active contours so that the model behaves like a balloon [6]. The active contours then are stopped by a strong edge but move past a spurious edge which is weak relative to the ambient inflation force. The GVF Snakes method introduces an external force called gradient vector flow (GVF) which is computed in terms of a diffusion of the gradient vectors of the image [22]. These two Snakes methods change their external forces to achieve a large capture range but they still find a local minimum. Geodesic active contours [5] find a geodesic curve in a Riemannian space derived from image content. Implicit active shape models embed contours as the zero level set of a higher dimensional function and then solve a partial differential equation of motion [16, 15, 18, 8, 17]. These two approaches eliminate the problems with uneven control points and self-crossing and can easily be extended to higher dimensional applications. However, they still have the problem of local minima. Dynamic programming is used to extract the globally optimal contour within a certain region [2, 11]. However, these methods do not scale properly from extracting contours to extracting surfaces and still have the self-crossing and uneven spacing problems.

In contrast to the framework of active contours, graph cuts approaches are applied as global optimization methods for computer vision problems such as image segmenta-

*This work is partly supported by National Science Foundation under grant ECS-0225523, and is partly completed at Siemens Corporate Research, Inc.

tion [21, 10, 14]. The image is represented using an adjacency graph. Each vertex of the graph represents an image pixel, while the edge weight between two vertices represents the similarity between two corresponding pixels. Usually, the cost function to be minimized is the summation of the weights of the edges that are cut. The exact solution can be found in polynomial time. However, this solution has a bias towards cuts with short boundaries, resulting in small regions. The normalized cut approach [19] is aimed at reducing this bias by introducing a cost function called disassociation, but it also introduces another bias towards similar weight partition [20]. The minimum mean cut [20] normalizes the cost function by the length of the cut. However, the algorithm is slow for planar graphs and NP-hard for non-planar graphs. Interactive graph cuts approach [3] uses $s-t$ minimum cut as an optimization method with the user identifying the object and background regions interactively.

In this paper, we present an approach called graph cuts based active contours (GCBAC). A key assumption of our approach is that the desired segmentation contour is a global minimum within its $apriori$ known size (width) contour neighborhood (CN, which is defined as a belt-shaped neighborhood region around a contour). The size of the CN may be specified by the user for a given segment, image or a class of images according to the characteristics of local minima. Thus, the objective of our approach is to find the closest contour that is a global minimum within its CN, given an initial contour. The GCBAC algorithm iteratively replaces a contour with a global minimum within the CN of the contour until the objective is achieved. At each step of the iterative deformation, the CN is obtained by dilating the current contour with the $apriori$ known size. At the same time, an inner boundary and an outer boundary of the CN are obtained. The image within the CN is represented by an (pixel) adjacency graph, and the problem of finding the global minimum contour within this CN is formulated as a multi-source multi-sink $s-t$ minimum cut problem on this graph, by treating the pixels on the inner boundary as multiple sources and the pixels on the outer boundary as multiple sinks. Since the resulting contour should be long enough to separate the two boundaries, our approach overcomes the well-known shortcoming of minimum cut, which is prone to yield a short boundary.

Moreover, by using graph cuts at each step of contour deformation, our approach has the following advantages compared to traditional active contours. (1). It has the ability to jump over local minima and provide a more global result. (2). Graph cuts guarantee continuity and lead to smooth contours free of self-crossing and uneven spacing problems. Therefore the internal force which is commonly used in traditional energy functions to control the smoothness is no longer needed, and hence the number of parameters is greatly reduced. (3). Our approach easily extends to the segmentation of three dimensional(3D) and higher dimensional objects. In addition, the proposed approach is suitable for interactive correction and the algorithm is guaranteed to converge after a finite number of contour updates unless it oscillates between equal capacity solutions.

In the next section, we describe our proposed approach in detail. Section 3 presents experimental results and comparison with other results. Section 4 presents concluding remarks.

2 Our Approach

2.1 Related graph theory

In this section, we present a graph-theoretic description of $s-t$ minimum cut which is the basis of our graph cuts based active contours approach. Let a flow network $G=(V,E)$ be a connected graph with vertex set V and edge set E. Each edge $(u,v)\in E$ has a nonnegative capacity $c(u,v)\geq 0$. If $(u,v)\notin E$, we assume that $c(u,v)=0$. Two vertices in V are distinguished: a source s and a sink t. A cut (S,T) of the flow network G is a partition of V into S and $T=V-S$ such that $s\in S$ and $t\in T$. The capacity of a cut is defined as the summation of the capacities of the edges across the cut, i.e. $c(S,T)=\sum_{u\in S,v\in T}c(u,v)$. The $s-t$ minimum cut problem is to find a cut in G that separates s and t with the smallest capacity. This problem is very closely related to the max-flow problem in graph theory. A flow in G is a real-valued function $f: V\times V\rightarrow R$ that satisfies the following properties [7]:

1. for all $u,v\in V$, $f(u,v)\leq c(u,v)$;
2. for all $u,v\in V$, $f(u,v)=-f(v,u)$;
3. for all $u\in V-\{s,t\}$, $\sum_{v\in V}f(u,v)=0$.

The value of a flow f from s is defined as $|f|=\sum_{v\in V}f(s,v)$. In the maximum-flow problem, we are given a flow network G with a source s and a sink t, and we wish to find a flow with the maximum value from s to t. There is an important correspondence between flows and cuts in networks, as we can see in the max-flow min-cut theorem as follows:

Theorem 1 (Ford-Fulkerson Theorem [9]) *The maximum flow from a vertex s to vertex t, $|f|$, is equal to the value of the capacity $c(s,t)$ of the minimum cut separating s and t.*

With this theorem, the $s-t$ minimum cut problem can be solved by using existing max-flow algorithms. Several polynomial-time algorithms for the maximum flow problem are described in [1]. Experimental comparison of several different max-flow min-cut algorithms can be seen in [4].

The minimum cut of interest in this paper is required to separate multiple source nodes from multiple sink nodes.

A simple operation on a graph of interest G in this regard is *node identification* which identifies a set of nodes $\{v_1, v_2, ..., v_n\}$ as a single new node v, deleting self loops, if any, and merging parallel edges with cumulative capacity, as shown in Fig. 1. In terms of this operation, we have the following Theorem for the multi-source multi-sink $s-t$ minimum cut problem:

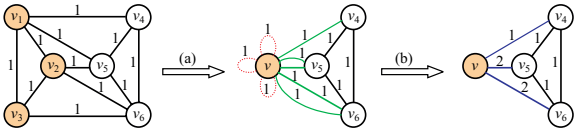


Figure 1. Node identification. (a) v_1, v_2, v_3 are merged into a new node v. (b) Self loops are deleted and parallel edges are replaced by a single edge.

Theorem 2 (Multi-source Multi-sink min-cut) *The minimum cut of graph G which separates a source set $\{s_1, s_2, \ldots, s_n\}$ and a sink set $\{t_1, t_2, \ldots, t_m\}$ is exactly the $s-t$ minimum cut of the graph that results after identifying $s_1, s_2, \ldots, s_n$ as a new source s and identifying $t_1, t_2, \ldots, t_m$ as a new sink t.*

Proof. There is a one to one mapping between a cut (S, T) in the original graph G that separates $\{s_1, s_2, \ldots, s_n\}$ from $\{t_1, t_2, \ldots, t_m\}$ and $s-t$ cut (S', T') in the graph G' that results after identifying the source set as s and the sink set as t, where $S = S' - s + \{s_1, s_2, \ldots, s_n\}$ and $T = T' - t + \{t_1, t_2, \ldots, t_m\}$. The capacities of the two corresponding cuts are the same because the process of *node identification* only deletes self loops which are not on the cuts. So if a cut (S', T') is an $s-t$ minimum cut in G', its corresponding cut (S, T) is a minimum cut in G that separates the source set and the sink set.

With the help of this theorem, we can use $s-t$ minimum cut algorithms to solve the multi-source multi-sink minimum cut problem by simply identifying the multiple sources as a single source and multiple sinks as a single sink, respectively.

2.2 Graph cuts based active contours

2.2.1 Overview

The graph cuts theory discussed above provides us with a method to compute the globally optimal partition of an image after we transform it into an edge capacitated graph $G(V, E)$. One such transformation is as follows. Each pixel within the image is mapped to a vertex $v \in V$. If two pixels are adjacent, there exists an edge $(u, v) \in E$ between the corresponding vertices u and v. The edge weight $c(u, v)$ is assigned according to some measure of similarity between the two pixels: the higher the edge weight, the more similar they are.

Each contour that partitions the image into two parts S and T corresponds to a cut (S, T) on the graph. The cut corresponding to a desired object contour is in general not the global minimum among all possible cuts on the graph (for example, the contour of another, smaller object might correspond to a cut with smaller capacity). However, as explained in section 1, the desired object contour is a global minimum within its contour neighborhood (CN). Since there might be many this kind of global minima in the image, an initial contour is required to distinguish them, and the objective of our approach is to find the closest contour that is a global minimum within its contour neighborhood. Given an initial contour, our algorithm consists of the following steps:

1. Represent the image as an adjacency graph G.
2. Dilate current boundary into its CN with an inner boundary and an outer boundary (Fig. 2).
3. Identify all the vertices corresponding to the inner boundary as a single source s and identify all the vertices corresponding to the outer boundary as a single sink t.
4. Compute the $s-t$ minimum cut to obtain a new boundary that better separates the inner boundary from the outer boundary.
5. Return to step 2 until the algorithm converges.

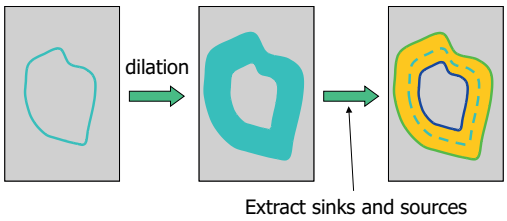
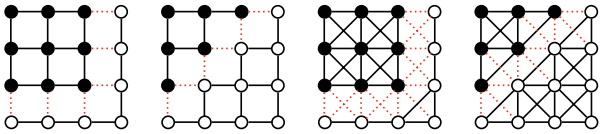


Figure 2. Using dilation to obtain the CN of a contour and the inner and outer boundaries of the CN. The inner and outer boundaries are treated as the sources and sinks, respectively, in the corresponding graph.

2.2.2 Connectivity and edge weights

We represent the image as an 8-connectivity graph $G(V, E)$, which means each vertex $v \in V$ in G, corresponding to a pixel p, has edges connecting it to its 8 neighboring vertices, which correspond to the 8 neighboring pixels of p. Fig. 3 presents a simple example of a homogenous area that shows why 8-connectivity is better than 4-connectivity. Fig. 3(a) and Fig. 3(b) are 4-connectivity graphs while Fig. 3(c) and Fig. 3(d) are 8-connectivity graphs, each edge having capacity 1. The dotted edges on each graph represent the edges on a cut. The contour corresponding to the cuts in Fig. 3(b) and Fig. 3(d) is better than the contour corresponding to the cuts in Fig. 3(a) and Fig. 3(c) because the former has a shorter length. The two cuts in Fig. 3(a) and

Fig. 3(b) have the same capacity 6. It means that if we use 4-connectivity, the two resulting cuts are indistinguishable. However, with 8-connectivity, the two cuts in Fig. 3(c) and Fig. 3(d) have different capacities, 15 and 11, respectively. This means that the cut with the shorter length has less capacity as is desired for homogeneous areas.



(a) (b) (c) (d)

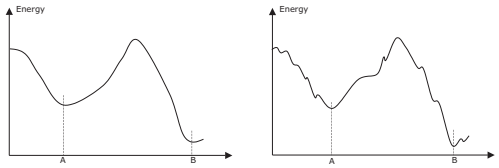
Figure 3. 4-connectivity and 8-connectivity graphs for a homogeneous area. The cuts are represented using red dotted edges. The cuts have the same capacity in (a) and (b) but different capacities in (c) and (d).

Besides selecting of the type of connectivity, edge weights assignment is also very important. In our implementation, we use $c(i,j) = (g(i,j) + g(j,i))^6$, where $g(i,j) = \exp(-\text{grad}_{ij}(i)/\max_k(\text{grad}_{ij}(k)))$, and $\text{grad}_{ij}(k)$ is the image pixel intensity gradient at location k in the direction of $i \rightarrow j$. This weight assignment method leads the active contours to high gradient edges and considers direction of the gradients.

2.2.3 Dilation

The dilation process in step 1 has several objectives. First, the dilation process generates a CN of the current contour, and makes our algorithm capable of jumping over local minima within this CN. Second, if the CN is a homogeneous area, the globally optimal contour should be the convex hull of the inner boundary, since it will cut the least number of edges in the corresponding graph. This property is helpful when the background is simple and the initial contour is much larger than the real object contour. Third, the dilation process generates an inner boundary that corresponds to the multiple sources in the corresponding graph. These multiple sources are identified as a single source, which is always contained in the S part of the resulting $s-t$ minimum cut. Although the minimum cut is prone to yield a small region, the use of *node identification* helps us to avoid this shortcoming as the resulting boundary in each step should be bigger than the inner boundary of the previous one.

The dilation process in our approach consists of a few single binary dilation steps, whose structuring element is a 3×3 matrix with all entries set to 1 in the 2D case or a $3 \times 3 \times 3$ tensor of 1 in the 3D case. Other structuring elements can also be used. The number of single dilations in each step (also referred to as step size in this paper), determines the size of the CN. The step size is a very important parameter and is selected based on two factors: the size of the object to be segmented and the amount of noise in the data. Large size objects may have large step size. For noisy images, a large step size is required to make the active contour break away from the many local minima near the object contour. However, if the step size is too large, the active contours may skip the real object contour. Fig. 4 shows the tradeoff in selecting step size. Fig. 4(a) shows clean data and Fig. 4(b) shows noisy data. Point A is the desired minimum point, and point B is another local minimum, having smaller energy than point A. For the energy function in Fig. 4(a), if we select a small step size, the active contours will find point A for a large range of initializations. However, for the energy function in Fig. 4(b), the active contours will probably get stuck at one of those local minima. On the contrary, if we select a very big step size for both cases, point B will be found, which is not desired. In our implementation, we select the step size manually according to the size of the object we want to segment and the noisiness of the data.



(a) (b)

Figure 4. Step size selection. The energy function in (a) is much smoother than the energy function in (b). Point A is the desired minimum while point B is not desired even though it has a smaller energy value. Step size should be selected so as to avoid local minima but not miss the desired solution A in favor of a global solution B.

2.2.4 Convergence

The proposed approach is guaranteed to converge by the following theorem:

Theorem 3 (Convergence Theorem) *Within a finite data set, the graph cuts based active contour will either converge or oscillate between several results with the same capacity after finite iterations.*

Proof. Let C_i be the capacity of the result R_i in the ith iteration, then $C_{i+1} \leq C_i, i = 1, 2, \cdots$, because in each iteration, the result is globally optimal within the area of interest. Finite data yields a finite number of different results $R^u, 0 < u < N$. Since the dilation process and the edge weights are well defined, we should have $R_{i+k} = R_{j+k}$, for $k = 1, 2, \cdots$, if $R_i = R_j$. If the algorithm does not converge after N iterations at least one result R^u will appear twice, following which the sequence between the two

R^u will repeat. Also, since $C_{i+1} \leq C_i, i = 1, 2, \cdots$, the capacity of each R_i within this sequence will be the same.

So if a boundary reoccurs, the algorithm terminates.

2.2.5 Interactive correction

If the final boundary is not satisfactory, the user can correct it interactively. The user can click a point that the boundary must pass through and the algorithm updates the source set and the sink set accordingly. Three situations are distinguished based on the user clicked point:

1. The input point lies between the inner boundary and the outer boundary.
2. The input point lies inside the inner boundary.
3. The input point lies outside the outer boundary.

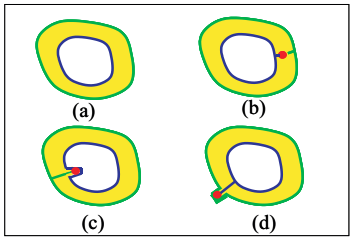


Figure 5. Updating sources and sinks to incorporate an input interactively supplied by the user. The yellow region indicates contour neighborhood (CN), the blue curve indicates the sources and the green curve indicates sinks.

In each case, the algorithm updates the sources and sinks so that the resulting boundary is forced to pass through the input point (Fig. 5). In each iteration thereafter, this input point lies in the dilated CN and is treated as an input point of situation 1. Therefore, the final boundary should also pass through this input point.

2.3 Comparison with traditional active contours

2.3.1 Cost function

The proposed GCBAC approach has a much simpler energy function than the traditional active contours. A cost function often used by the traditional active contours is $E = E_{internal} + E_{external} = \alpha E_{tension} + \beta E_{rigidity} + E_{ext}$, where α and β are weighting parameters that control the curve's tension and rigidity, respectively. The internal energy is designed to hold the curve together and to keep it from bending excessively. This allows control of the object shape, but requires careful adjustment of the weights for different kinds of object boundaries, or even at different stages of deformation. In our approach, no internal energy is explicitly needed. The graph cuts guarantee the continuity of the resulting contour. The absence of internal energy makes it very easy to extend our algorithm to segment 3D and even higher-dimensional objects. It is difficult to analogously generalize the traditional active contours to higher dimensions.

The external energy in our approach is derived from the image and represented by edge weights on the corresponding graph. Through the assignment of edge weights and selecting different connectivities (for example, constructing an edge between each pair of vertices), the cut based methods are able to take more image information into account than captured in the external energy in traditional active contours approaches.

2.3.2 Final contour

The final contour resulting from our approach is globally optimal within its CN, whereas the traditional active contours provide a locally optimal contour. Since minimum cut is computed on a graph where each vertex corresponds to a pixel in the corresponding image, the resulting contour does not have any problem due to uneven spacing that accompanies the use of control points for representing the contour. Further, the final contour provided by the traditional active contours algorithm might have the problem of self-crossing (Fig. 6). This situation will not occur in our approach as the minimum cut will partition the graph into two parts instead of three.

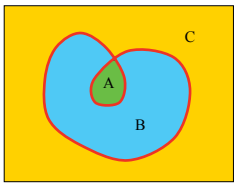


Figure 6. Self-crossing problem. The red contour self-crosses and segments the image into three parts: A, B and C.

3 Experimental Results

In this section, we present our experimental results and compare them with the results of GVF Snakes. The results demonstrating different aspects are presented in different subsections.

3.1 Insensitivity to initialization

Fig. 7 shows a synthetic image containing a gray, diamond-shaped object constructed to demonstrate the insensitivity of the algorithm to the initial contour. Different rows correspond to the use of different initial contours. The leftmost image in each row shows the initial contour and the rightmost image shows the final contour. The middle two images depict the intermediate stages of GCBAC algorithm. Fig. 8 shows an application to lung nodule segmentation. The images are extracted from a CT volume of

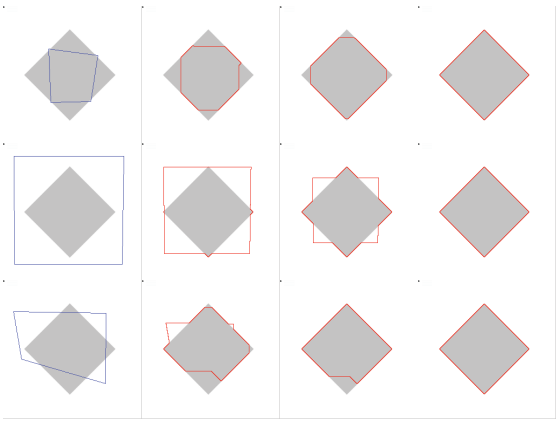


Figure 7. Experimental results on synthetic images show that the proposed approach is insensitive to initial contours. The leftmost image in each row shows the initial contour and the rightmost image shows the final contour. The intervening images show how the active contours deform.

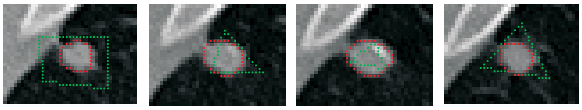


Figure 8. GCBAC used for lung nodule segmentation. The green polygon in each image shows the initial contour and the red contour shows the final contour.

a patient's lung. Four different initializations are shown in green, and the corresponding final contours are shown in red.

However, if an initial boundary is far from the object boundary, inside or outside, it is still difficult for our approach to find the real object boundary.

3.2 Discontinuities and noise

Fig. 9 demonstrates that the GCBAC algorithm is able to handle large edge gaps on object boundary as well as noise in the data. The left image in the first row of Fig. 9 shows a synthetic image of a triangle-shaped object with large gaps in the boundary. The left image in the second row shows an image with scattered noise points. The left image in the last row is corrupted by the additive Gaussian noise. The rightmost image of each row shows the final contour. The middle two images depict the intermediate stages.

Fig. 10 shows an application to lung nodule segmentation. The nodules have discontinuities on their boundaries. Initializations are shown in green, and the corresponding final contours are shown in red.

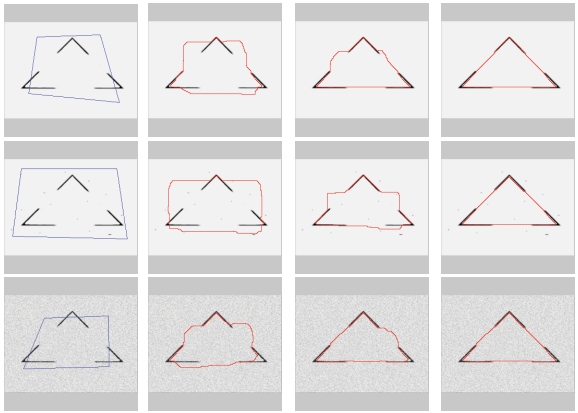


Figure 9. Experiments with an image containing edge discontinuities and noise. The leftmost image in each row shows the initial contour and the rightmost image shows the final contour. The intervening images show how the active contours deform.

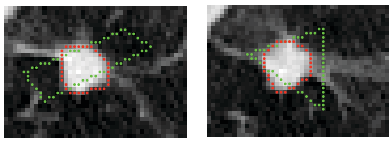


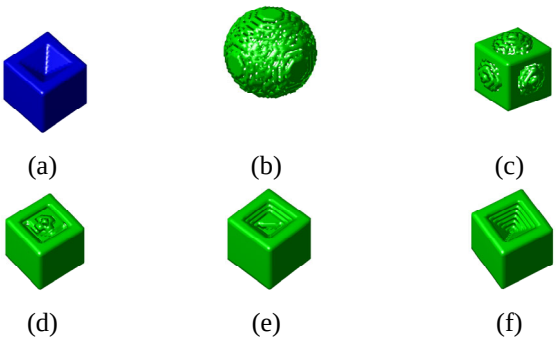
Figure 10. Application to lung nodule segmentation. The green polygon in each image shows the initial contour and the red contour shows the final contour. Both nodules have discontinuities on their boundaries.

3.3 Higher dimensions

Fig. 11 shows the application of GCBAC to segment 3D objects. A synthetic object is shown in Fig. 11(a). The initial boundary is a sphere as shown in Fig. 11(b). Fig. 11(c) through (e) show intermediate results. The final resulting surface is shown in Fig. 11(f).

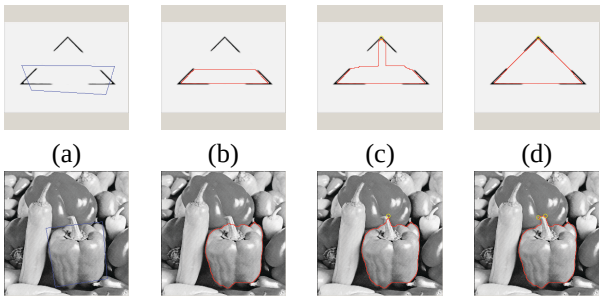
3.4 Interactive corrections

The first row of Fig. 12 shows the interactive correction procedure applied to a triangle with gaps in its boundary and an initial contour of Fig. 12(a). The resulting contour without correction is shown in Fig. 12(b). However, this is not the desired result. By accepting a user clicked point, marked with yellow circle in Fig. 12(c), our algorithm continues and obtains a satisfactory result shown in Fig. 12(d). Fig. 12(e)-(h) show our results on a real image. Fig. 12(e) is the original image with an initial boundary. Fig. 12(f) shows the resulting boundary without correction. Since the resulting pepper contour is not satisfactory, the user clicks two points one after another, marked as yellow circles, to guide the active contours to the desired results. Fig. 12(g)



(a) (b) (c) (d) (e) (f)

Figure 11. Experimental results for a synthetic 3D object. (a): 3D object. (b): Initial surface. (c)-(e): Intermediate resulting surfaces. (f): Final result.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 12. Interactive corrections. The leftmost image in each row shows the initial contour. The yellow circles are the correction points clicked by the user to guide the deformation of the active contours. The rightmost image shows the final results.

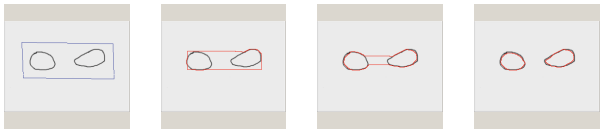
shows the result after first correction and Fig. 12(h) shows the final result after both corrections.

3.5 Changes in contour topology

Our algorithm is capable of changing the topology of the initial contour during deformation. A simple example is shown in Fig. 13. Changing topology is desired in some instances but not in others [12].

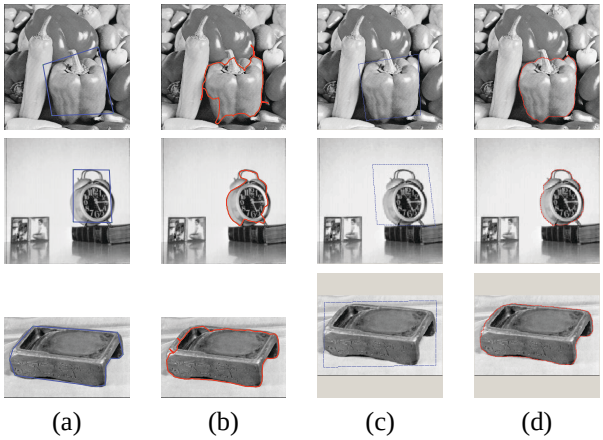
3.6 Comparisons with GVF snakes

We compare our results with those obtained using the traditional active contours, specifically, the GVF Snakes, which have a large capture range and can move into boundary concavities. We use the GVF implementation available at http://iacl.ece.jhu.edu/projects/gvf/. We select the parameters of the GVF Snakes as follows: $\mu = 0.1$, Iteration count$= 200$ (for computing the gradient vector field),



(a) (b) (c) (d)

Figure 13. A simple example of topology changes in the active contour during deformation. (a) Original image and initial contour. (b), (c) Intermediate results. (d) Final contour with a different topology.



(a) (b) (c) (d)

Figure 14. Comparison with GVF Snakes for three real images. Column (a) Initial boundary for GVF Snakes. (b) Final result from GVF Snakes. (c) Initial boundary for GCBAC. (d) Final result from GCBAC. Note that the initial contour for GVF in the third row is very accurate.

$\alpha = 0.05$, $\beta = 0$, $\gamma = 1$, $\kappa = 0.6$, Dmin $= 2$, and Dmax $= 4$. Fig. 14 shows the results of our approach and GVF Snakes approach for the same real images. Note that the initial contours provided for GVF Snakes are more accurate than these for GCBAC.

3.7 Running time analysis

We implement the excess scaling preflow-push algorithm as described in [1] to solve the $s - t$ minimum cut problem. Without using sophisticated data structures, the algorithm achieves the running time of $O(nm + n^2 \log U)$, where n is the number of nodes, m is the number of edges, and U is the largest edge weight. However, the simple topology of the graph used in GCBAC makes the algorithm run much faster in practice. By obtaining the best polynomial fit to observed data, the minimum cut algorithm used in GCBAC has running time of $O(n^{1.2})$. In Fig. 15, the running time is shown as a function of the number n of nodes.

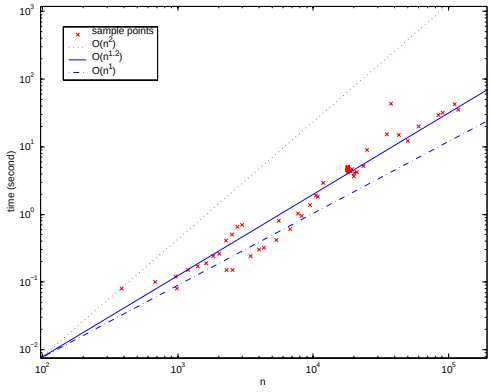


Figure 15. The observed running time of the minimum cut algorithm used in our approach is $O(n^{1.2})$. The performance levels of $O(n)$ and $O(n^2)$ are shown, respectively, as references.

4 Conclusions and Discussion

We have presented a graph cuts based active contours (GCBAC) approach to object segmentation. First, we transform a multi-source multi-sink minimum cut problem into a single $s-t$ minimum cut problem. Then, the problem of finding desired segmentation contour is formulated as that of finding the closest contour that is a global minimum within its contour neighborhood (CN), given an initial contour.

A drawback of our algorithm is that we must construct the graph with appropriate pixel connectivity and edge weights. An example of the need for appropriate topology selection is illustrated by the fact that the 8-connectivity graph may result in smoother curves than the 4-connectivity graph, but the 8-connectivity graph still has limitations in homogeneous image areas. It will be interesting to devise a strategy for constructing the graph such that in homogeneous areas a cut of shorter length has a strictly smaller capacity. Other future problems include: how to incorporate texture information into our algorithm so that both color and texture information are considered simultaneously, how to extend our algorithm to segment multiple objects, and how to determine when our algorithm should allow changes in topology.

References

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

[2] A. Amini et al. Using dynamic programming for solving variational problems in vision. *IEEE Trans. on PAMI*, 12(9), September 1990.

[3] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. *The 8th ICCV*, 1:105–112, July 2001.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *3rd. Intnl. Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition(EMMCVPR)*, pages 359–374, Sept. 2001.

[5] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *Proc. 5th IEEE Int. Conf. on Computer Vision (ICCV)*, pages 694–699, 1995.

[6] L. D. Cohen. On active contour models and balloons. *CVGIP:Image Understanding*, 53(2):211–218, March 1991.

[7] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw–Hill Companies, 1990.

[8] O. Faugeras and R. Keriven. Variational principles, surface evolution, pde's, level set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–344, March 1998.

[9] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[10] D. Forsyth and J. Ponce. *Computer Vision: A Morden Approach*. Prentice-Hall, 2002.

[11] D. Geiger, A. Gupta, L. A. Costa, and J. Vlontzos. Dynamic programming for detecting, tracking, and matching deformable contours. *IEEE Trans. on PAMI*, 17(3), March 1995.

[12] X. Han, C. Xu, and J. Prince. A topology preserving deformable model using level sets. *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition*, 2:765 –770, Dec. 2001.

[13] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331, 1988.

[14] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *ECCV (3)*, pages 65–81, 2002.

[15] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(2):158–175, 1995.

[16] S. Osher and J. Sethian. Fronts propagating with curvature dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[17] N. Paragios, O. Mellina-Gottardo, and V. Ramesh. Gradient vector flow fast geodesic active contours. *The 8th ICCV*, 1:67–73, July 2001.

[18] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 2 edition, 1999.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE CVPR*, June 1997.

[20] S. Wang and J. Siskind. Image segmentation with minimum mean cut. *The 8th ICCV*, 1:517–524, July 2001.

[21] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Trans. on PAMI*, 15(11):1101–1113, 1993.

[22] C. Xu and J. Prince. Snakes, shapes, and gradient vector flow. *IEEE Trans. on Image Processing*, 7(3):359–369, March 1998.