Beyond Connecting the Dots: A Polynomial-time Algorithm for Segmentation and Boundary Estimation with Imprecise User Input

Thomas Windheuser, Thomas Schoenemann and Daniel Cremers
Department of Computer Science
University of Bonn

Abstract

We propose a polynomial-time algorithm for segmentation and (open) boundary estimation which takes into account a series of user-specified attraction points. In contrast to existing algorithms which impose that the segmenting boundary passes through *these points, our algorithm allows an imprecision in the user input. An energy minimization approach imposes that the segmenting boundary optimally passes along high-contrast edges in such a way that at least one point along the computed boundary is as close as possible to any given attraction point. In this sense, the user input can be seen as a soft constraint. We prove that the resulting optimization problem is NP-hard. We prove that in the case that the user attraction points are ordered, then optimal solutions can be computed in polynomial time using a shortest path formulation in an appropriately constructed four-dimensional graph spanned by the image pixels, a set of tangent angles and the user attraction points. Experimental results on a variety of images demonstrate that good quality segmentations can be obtained with a few imprecise user clicks.*

1. Introduction

1.1. Interactive Image Segmentation

Despite numerous advances on purely low-level image segmentation, there is a consensus among researchers that purely low-level segmentation schemes are often of little practical use. Without a user specifying what objects in a given scene he is interested in, a computer has little chance of distinguishing a good from a bad segmentation. Most of the commonly used segmentation schemes therefore include some kind of user input biasing the respective low-level algorithm in some way toward the desired solution. Among the more popular examples are interactive graph cut algorithms [2, 9] or the Random Walker [7, 11] which allow the user to click points which are then labeled object or background. Similar interactive segmentation schemes using regional user labels have been proposed in the context of level set methods [4] or convex relaxation schemes [12].

Input image & attraction points | Computed boundary line

Figure 1. Segmentation with soft user input: Given an input image and a user-specified set of attraction points, the proposed polynomial-time algorithm determines a boundary which optimally passes through areas of strong intensity gradient while preserving a minimal distance to each of the attraction points.

In this paper we focus on a different user input where the user specifies points which are likely to be *on* the desired boundary. In contrast to the above region-based approaches, such boundary-based methods can also be used to identify *open* boundaries. To date researchers have proposed algorithms which allow to find segmenting boundaries that pass through the user-specified points [3, 5]. The main limitation of such approaches and the reason why they have recently been replaced by the mentioned region-labeling approaches is that respective user points need to be placed on the boundary *very precisely*. Any imprecision in the user labeling of the boundary will invariably degrade the segmentation result. To alleviate this, researchers have proposed to first move the user points to areas of high intensity gradient before computing the segmentation [8]. Such strategies are clearly suboptimal in the sense that the optimal placement of the user labels cannot be solved independently from the segmentation problem.

1.2. From Hard to Soft User Input

In this work we revisit the problem of interactive image segmentation and consider a user input which is by definition *imprecise*. Instead of imposing the segmenting curve to

pass through the user-specified points or heuristically moving the user points before computing a segmentation, we formulate the interactive segmentation as a problem of energy minimization where the distance of each user-point to the closest point on the segmenting contour is penalized. We refer to this user input as *attraction points*, because each point induces an attraction on the closest boundary point.

First we show that the resulting optimization problem is NP-hard. Secondly, we consider the simpler case that the user-specified points are *ordered*. We show that in that case optimal segmentations can be computed in polynomial time as minimal paths in the three-dimensional graph spanned by the image pixels and the user-specified attraction points. Lastly we propose a generalization of this approach to impose optimal curvature regularity of the computed segmentations by computing shortest paths through respective four-dimensional graphs spanned by the attraction points, the image pixels and the local tangent angle.

Figure 1 provides an example of the proposed method: Given an input image and a user-specified list of attraction points, the algorithm determines a contour in the image, which optimally passes through areas of high intensity gradient while maintaining a minimal distance to the attraction points.

2. Curve Estimation with Imprecise User Input

Given an image and a set of attraction points from the user, we want to fit a curve into the image that follows the edges of the image, has a small curvature and simultaneously passes through the vicinity of the given points.

Let $I : \Omega \to \mathbb{R}$ be the image defined on some domain $\Omega \subset \mathbb{R}^2$, let $\mathbf{X}_1, \dots, \mathbf{X}_K \in \Omega$ be a set of K attraction points, and let $\mathcal{C}$ be the set of all closed curves in Ω. Then we search for a curve $\mathbf{C} \in \mathcal{C}, \mathbf{C} : [0, \mathcal{L}(\mathbf{C})] \to \Omega$ (parameterized by arc-length) of length $\mathcal{L}(\mathbf{C})$ that minimizes an energy function $E : \mathcal{C} \to \mathbb{R}$ consisting of three terms: The first is an attraction to image edges, realized by some edge detector $g : \Omega \to \mathbb{R}^+$ which assigns low cost to high image gradients. The second term penalizes the curvature of the curve and hence favors smooth curves.

The third term is devoted to the attraction points. Unlike previous works, where the curve must pass directly through the input points as a hard constraint, we allow the curve to pass in some distance to the input points. We define the distance as the minimal L_2 -distance of the curve $\mathbf{C} \in \mathcal{C}$ to a point $\mathbf{X}_i, i \in \{1, \dots, K\}$:

$$d(\mathbf{C}, \mathbf{X}_i) = \min_{s \in [0, \mathcal{L}(\mathbf{C})]} \|\mathbf{X}_i - \mathbf{C}(s)\|. \tag{1}$$

Because long distances are penalized in the energy function, the curve is pulled to each of the attraction points as if a rubber band connected the curve and the respective point. Adding weighting factors $\alpha, \lambda \in \mathbb{R}_0^+$ and denoting the curvature at $\mathbf{C}(s)$ as $\kappa_{\mathbf{C}}(s)$ the energy function is defined as

$$E(\mathbf{C}) = \alpha \sum_{i=1}^{K} d(\mathbf{C}, \mathbf{X}_i) + \int_0^{\mathcal{L}(\mathbf{C})} \left[g(\mathbf{C}(s)) + \lambda |\kappa_{\mathbf{C}}(s)|^2 \right] ds. \tag{2}$$

Unfortunately, in its current form the problem cannot be solved efficiently.

Proposition 1. *The discrete optimization problem corresponding to functional (2) is NP-hard.*

Proof. For the specific case of $\lambda = 0$ and $g(\mathbf{y}) = 1$ and $\alpha \to \infty$ (i.e. α sufficiently large), the problem amounts to finding the shortest path connecting all attractor points. This problem is known as the Euclidean Traveling Salesman Problem and was shown to be NP-hard [6]. □

Fortunately it turns out that the problem becomes solvable if we impose the set of attraction points to be *ordered*.

Before we formalize the order constraint, we first note that minimizing (2) can be rewritten as minimizing an equivalent functional over the contour $\mathbf{C}$ and real numbers $s_1, \dots, s_K \in [0, \mathcal{L}(\mathbf{C})]$:

$$E(\mathbf{C}, s_1, \dots, s_K) = \tag{3}$$
$$\alpha \sum_{i=1}^{K} \|\mathbf{C}(s_i) - \mathbf{X}_i\| + \int_0^{\mathcal{L}(\mathbf{C})} \left[g(\mathbf{C}(s)) + \lambda |\kappa_{\mathbf{C}}(s)|^2 \right] ds$$

The order is now imposed by enforcing the constraints

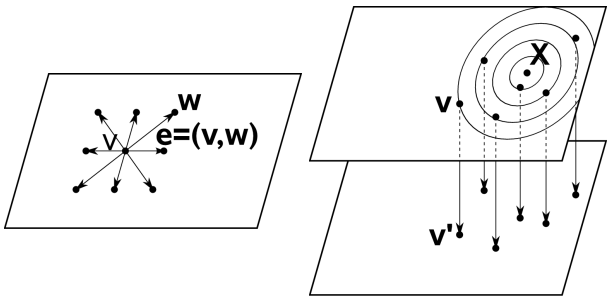
$$s_i < s_j, \qquad \forall i < j \,. \tag{4}$$

Proposition 2. *The minimum of function (3) subject to constraint (4) can be found in polynomial time.*

Proof: In Sections 3, 4 and 5 we will show that this problem reduces indeed to a search for a shortest path in a 4D graph. With Dijkstra's algorithm this path can be found in polynomial time. □

3. The Discrete Problem: The Search for a Shortest Path in a Layered Graph

We will globally solve the arising problem in a discrete setting, i.e. we only consider polygonal curves that are composed out of a given set of line segments. In the discretization we embed the image I and the user points $\mathbf{X}_1, \dots, \mathbf{X}_K$ into a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We add two special nodes $\mathbf{r}$, $\mathbf{t}$ to this graph and add weights $w : \mathcal{E} \to \mathbb{R}^+$ onto the edges. To solve problem (3) subject to the constraint (4) we compute a shortest path from the root node $\mathbf{r}$ to target node $\mathbf{t}$



graph structure within a layer. graph structure between two layers.

Figure 2. The proposed algorithm determines shortest paths through a 3D graph made of several copies of the image plane. The graph includes two types of edges: **(left)** Edges within each copy of the image plane favoring paths along strong gradient edges. Weights are set according to the edge detector function g. **(right)** Edges between copies of the image favor transitions in the vicinity of the attraction points. Edge weights are set according to the distance between input point $\mathbf{X}$ and pixel $\mathbf{v}$. The attraction point defines circles on the plane where the transition edges have equal costs.

in the weighted graph $(\mathcal{G}, w)$. The shortest path corresponds to the minimum curve we are looking for.

To illustrate the principle of the graph construction, for the present we make two simplifications. First, we simplify the energy function by removing the curvature term. We will deal with the curvature term in Section 4. Second, we search only for open curves, starting and ending at two arbitrary points in the image. We will close the curves in Section 5. The minimization problem now is

$$\min_{\mathbf{C}\in\mathcal{C}, s_1,\ldots,s_K\in[0,\mathcal{L}(\mathbf{C})]} E(\mathbf{C}, s_1, \ldots, s_K) = \alpha \sum_{i=1}^{K} \|\mathbf{C}(s_i) - \mathbf{X}_i\| + \int_0^{\mathcal{L}(\mathbf{C})} g(\mathbf{C}(s))ds \quad \text{s.t.} \quad s_i < s_j, \quad \forall i < j\,. \tag{5}$$

We embed the image and input points into a 3D graph. The graph consists of $K+1$ two-dimensional layers each representing a copy of the image. Inside the layers the graph mirrors the pixels, their neighborhood and the costs of the edge detector g. Between two layers there are transition edges which indicate distance costs of a path and an input point.

In each layer the pixels of the image correspond one-to-one to nodes. In order to add edges between nodes, we first have to define a connectivity between pixels. A pixel is connected to all pixels in a certain neighborhood. For example we choose the popular 8- and 16-neighborhood in our experiments. If two pixels $\mathbf{p}$ and $\mathbf{q}$ are neighbors, we write $(\mathbf{p}, \mathbf{q}) \in \mathcal{N}$ and add a directed edge $\mathbf{e} = (\mathbf{v_p}, \mathbf{v_q})$ between its corresponding nodes $\mathbf{v_p}$ and $\mathbf{v_q}$. This is illustrated on the left in Figure 2. Because the curve is represented by a path through this graph we define the weight of $\mathbf{e} = (\mathbf{v_p}, \mathbf{v_q})$ as $w((\mathbf{v_p}, \mathbf{v_q})) := \frac{1}{2}\|\mathbf{p} - \mathbf{q}\|(g(\mathbf{p}) + g(\mathbf{q}))$.

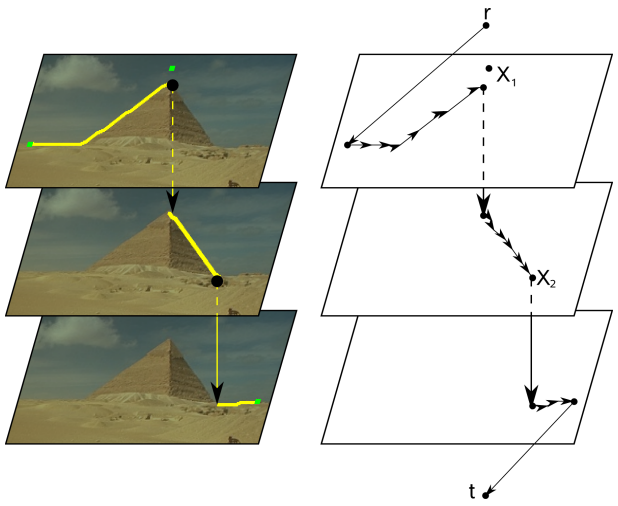


Figure 3. The Optimal Curve as a shortest path in a layered graph. For each attraction point a copy of the image is added to the graph. The part of the shortest path that belongs to the copy gives the part of the optimal contour between the respective attraction point and the next one. In this example start and end points are fixed.

For each input point $\mathbf{X}_i$ we add an additional layer as a copy of the image to the graph. Thus we have $K+1$ layers and each pixel $\mathbf{p}$ corresponds to $K+1$ nodes $\mathbf{v}_{\mathbf{p},1}, \ldots, \mathbf{v}_{\mathbf{p},K+1}$. Directed connections between layers i and $i+1$ are made. So for each node $\mathbf{v}_{\mathbf{p},i}, 1 \le i \le K$ there is an edge $\mathbf{e} = (\mathbf{v}_{\mathbf{p},i}, \mathbf{v}_{\mathbf{p},i+1})$ with weight $w(\mathbf{e}) = \|\mathbf{p} - \mathbf{X}_i\|$ in the graph. This defines circle-shaped iso lines around $\mathbf{X}_i$ of equal weight on the transition edges. This is illustrated in Figure 2 on the right.

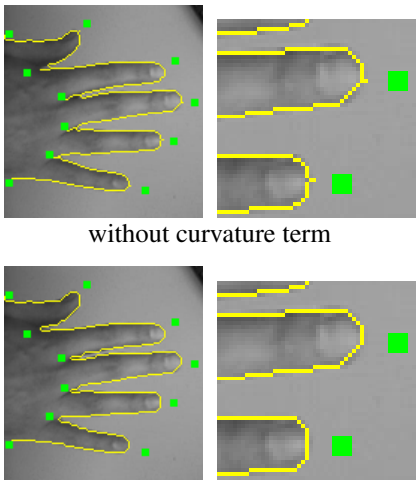
A path from $\mathbf{v}_{\mathbf{p},i}$ to $\mathbf{v}_{\mathbf{q},i+1}$ crosses the border between the layers i and $i+1$ exactly once. Furthermore a shortest path chooses the transition between layers exactly at the point where the distance from the path to input point $\mathbf{X}_i$ is minimum.

Edges of zero weight are added from the start node $\mathbf{r}$ to each node in layer 1 and from each node in layer $K+1$ to target node $\mathbf{t}$. If $\mathcal{P}$ is the set of pixels the set of nodes can be specified by

$$\mathcal{V} = \{\mathbf{r}, \mathbf{t}\} \cup \{\mathbf{v}_{\mathbf{p},i} \mid \mathbf{p} \in \mathcal{P} \wedge 1 \le i \le K+1\}\,, \tag{6}$$

and the set of edges by

$$\begin{aligned} \mathcal{E} = \; & \{(\mathbf{r}, \mathbf{v}_{\mathbf{p},1}) \mid \mathbf{p} \in \mathcal{P}\} \\ & \cup \{(\mathbf{v}_{\mathbf{p},K+1}, \mathbf{t}) \mid \mathbf{p} \in \mathcal{P}\} \\ & \cup \{(\mathbf{v}_{\mathbf{p},i}, \mathbf{v}_{\mathbf{q},i}) \mid (\mathbf{p}, \mathbf{q}) \in \mathcal{N} \wedge 1 \le i \le K+1\} \\ & \cup \{(\mathbf{v}_{\mathbf{p},i}, \mathbf{v}_{\mathbf{p},i+1}) \mid \mathbf{p} \in \mathcal{P} \wedge 1 \le i \le K\}\,. \end{aligned} \tag{7}$$



without curvature term

with curvature term

Figure 4. Curvature regularity keeps the curve smooth.

It is easy to see that a path from $\mathbf{r}$ to $\mathbf{t}$ passes all layers and makes exactly K transitions. Furthermore a shortest path from $\mathbf{r}$ to $\mathbf{t}$ directly corresponds to a minimum solution of (5).

4. Including Curvature Regularity

The presentation so far assumed that there is no curvature term. However, as Figure 4 demonstrates curvature terms are important to get smooth curves. Without curvature the curves get peaked to reduce the distance to the input points.

To include dependences on curvature, the graph has to be altered. The problem is that the weights are defined on edges, but so far edges corresponded to line segments. And straight line segments always have curvature 0.

To accurately reflect the curvature of the contour, one needs to consider *pairs* of line segments as proposed in [1, 10]. The arising graph is again composed of layers. Now a layer contains a node for each line segment. Edges (within a layer) then correspond to pairs of line segments. Hence, their weights can now reflect curvature cost.

Formally, the node set of the graph is now given as

$$\mathcal{V} = \{\mathbf{r}, \mathbf{t}\} \cup \{\mathbf{v}_{\mathbf{p},\mathbf{q},i} \,|\, (\mathbf{p},\mathbf{q}) \in \mathcal{N} \wedge 1 \leq i \leq K+1\}$$

Edges that represent a part of the contour now connect a pair $(\mathbf{p},\mathbf{q}) \in \mathcal{N}$ of neighboring pixels with an adjacent pair $(\mathbf{q},\mathbf{s}) \in \mathcal{N}$. Again, such an edge is present in all layers i. The set of all edges is now

$$\begin{aligned}\mathcal{E} = \quad & \{(\mathbf{r}, \mathbf{v}_{\mathbf{p},\mathbf{q},1}) \mid (\mathbf{p},\mathbf{q}) \in \mathcal{N}\} \\ & \cup \{(\mathbf{v}_{\mathbf{p},\mathbf{q},K+1}, \mathbf{t}) \mid (\mathbf{p},\mathbf{q}) \in \mathcal{N}\} \\ & \cup \{(\mathbf{v}_{\mathbf{p},\mathbf{q},i}, \mathbf{v}_{\mathbf{q},\mathbf{s},i}) \mid (\mathbf{p},\mathbf{q}) \in \mathcal{N} \wedge (\mathbf{q},\mathbf{s}) \in \mathcal{N} \\ & \qquad\qquad \wedge 1 \leq i \leq K+1\} \\ & \cup \{(\mathbf{v}_{\mathbf{p},\mathbf{q},i}, \mathbf{v}_{\mathbf{p},\mathbf{q},i+1}) \mid (\mathbf{p},\mathbf{q}) \in \mathcal{N} \wedge 1 \leq i \leq K\}.\end{aligned} \tag{8}$$

The edge weights on inter-layer edges do not change, but we have to add the curvature term to the intra-layer edge weights. We set the weight of edge $(\mathbf{v}_{\mathbf{p},\mathbf{q},i}, \mathbf{v}_{\mathbf{q},\mathbf{s},i})$ to

$$w(\mathbf{v}_{\mathbf{p},\mathbf{q},i}, \mathbf{v}_{\mathbf{q},\mathbf{s},i}) := \tfrac{1}{2}\|\mathbf{p}-\mathbf{q}\|(g(\mathbf{p})+g(\mathbf{q}))+\lambda|\kappa(p,q,s)|^2,$$

where $\kappa(p,q,s)$ is the curvature between line segments (p,q) and (q,s).

The arising graphs can get quite big. To reduce the memory consumption we revert to an implicit graph representation. This is possible since in Dijkstra's method one evolves the distance labels of nodes. The edges of the graph, together with their weights, can hence be computed on-the-fly whenever they are needed.

5. Handling Closed Curves

The described formalism is easily extended to handle the case of closed contours: closed contours can be represented as open contours where the start point equals the end point:

$$\mathbf{C}(0) = \mathbf{C}(\mathcal{L}(\mathbf{C}))\,.$$

To ensure this constraint, the above procedure needs to be modified. Here we assume that we know the start and end point and call it $\mathbf{p} = \mathbf{C}(0) \in \mathcal{P}$. The arising optimization problem can be handled by removing some edges from the above described graph. These are edges that either leave the node $\mathbf{r}$ or enter the node $\mathbf{t}$.

The only edges that now leave node $\mathbf{r}$ end in nodes of the form $\mathbf{v}_{\mathbf{q},\mathbf{p},1}, \mathbf{q} \in \mathcal{P}, (\mathbf{q},\mathbf{p}) \in \mathcal{N}$. Likewise, the only edges that enter $\mathbf{t}$ start in nodes of the form $\mathbf{v}_{\mathbf{q},\mathbf{p},K+1}, \mathbf{q} \in \mathcal{P}, (\mathbf{q},\mathbf{p}) \in \mathcal{N}$. Using this construction it is indeed assured that the curve starts and ends in $\mathbf{p}$.

6. Experiments

We now demonstrate a number of interesting properties of the proposed method on images from the well-known Berkeley Image Database. For better visibility of segmentation results, the image intensities have been slightly reduced.

In all experiments we use the edge detector function

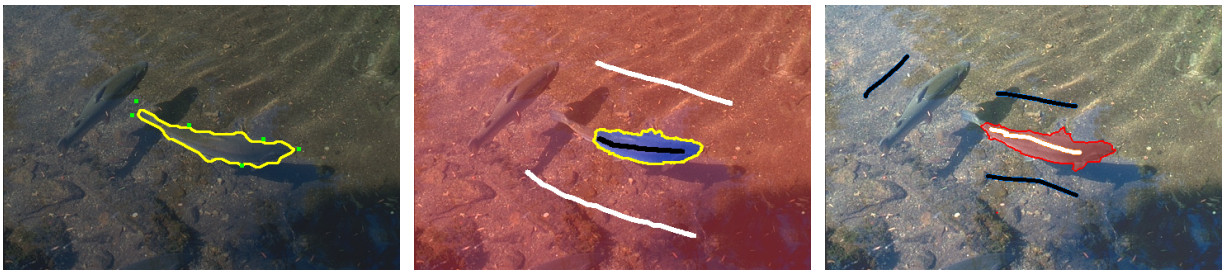
$$g(\mathbf{x}) = \frac{1}{1+|\nabla R(\mathbf{x})|+|\nabla G(\mathbf{x})|+|\nabla B(\mathbf{x})|},$$

where R, G, B denote the color channels of the images.

Figure 7 demonstrates that the attraction points need not lie on the boundary, since they only impose a soft constraint to bias the curve in a certain direction.

Finally Figure 8 provides a comparison of popular methods for interactive segmentation showing that we can obtain comparable results to graph cuts [2] and TVSeg [12] with very few user clicks.

Figure 7. The proposed method reliably extracts objects despite the imprecision of the user input.



proposed shortest path approach | graph cut [2] | TV Seg [12]

Figure 8. Even with very few input points the proposed method outperforms region-based methods.

The improvement over region-based approaches becomes more apparent if objects are not characterized by their color histograms. Figure 5 shows that with very few clicks we can compute high-quality segmentations. In contrast to segmentation approaches like graph cuts the proposed algorithm supports both open and closed curves.

Lastly, Figure 6 demonstrates how important the handling of *imprecise* input points is: If one connects the points in a strict manner the results are far from satisfactory.

7. Conclusion

We proposed an efficient algorithm to optimally integrate imprecise user input in tasks of boundary estimation and image segmentation. The key idea is to introduce a functional that contains a distance term penalizing the minimal distance of each of the input points to the curve. We showed that the general problem is NP-hard, but that when an order of the points is given, the functional can be optimized globally in polynomial time. This is achieved by computing the shortest path in a layered graph, where each layer represents a part of the optimal contour.

Experimental results on numerous real-world images demonstrate that this soft user input substantially outperforms hard user input. With very few user clicks, we can generate segmentation results which are competitive with state-of-the-art segmentation schemes like graph cuts or TVSeg.

References

[1] A. Amini, T. Weymouth, and R. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 12(9):855 – 867, Sept. 1990. 4

[2] Y. Boykov and M.-P. Jolly. Interactive organ segmentation using graph cuts. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 276–286, 2000. 1, 4, 5, 6

[3] L. Cohen and R. Kimmel. Global minimum for active contour models: A minimal path approach. *Int. Jour. of Comp. Vision*, 24(1):57–78, Aug. 1997. 1

[4] D. Cremers, O. Fluck, M. Rousson, and S. Aharon. A probabilistic level set formulation for interactive organ segmentation. In *Proc. of the SPIE Medical Imaging*, San Diego, USA, February 2007. 1

[5] A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. d. A. Lotufo. User-steered image segmentation paradigms: Live wire and live lane. *Graphical Models and Image Processing*, 60(4):233–260, July 1998. 1

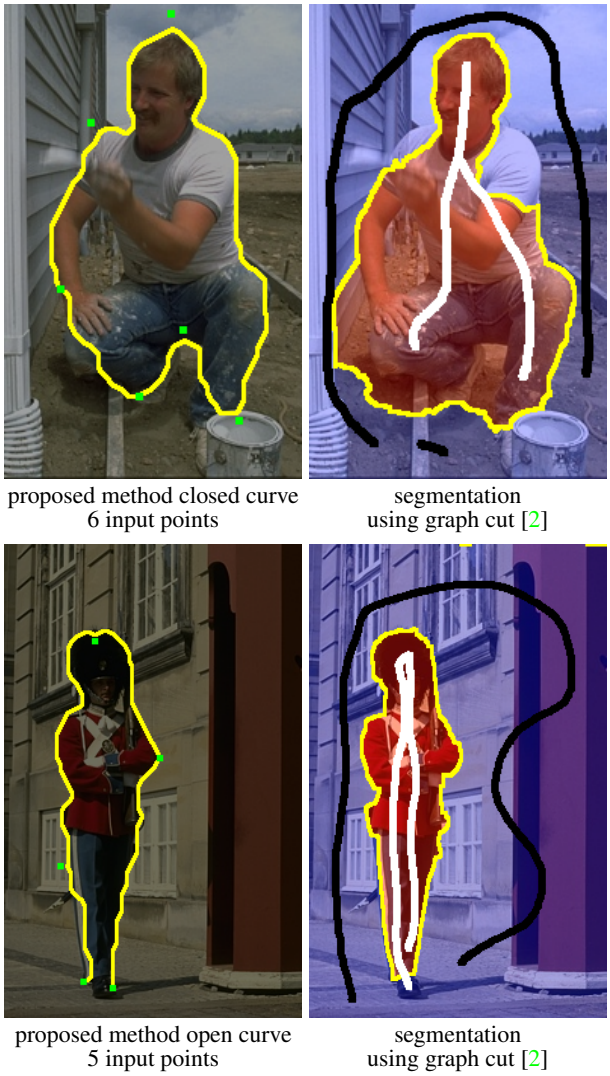


Figure 5. Extracting open and closed curves in complex scenes with very few input points. In our experiments, the graph cut method provided less reliable results with more user input. Note that the results on the left required only six and five input points respectively.

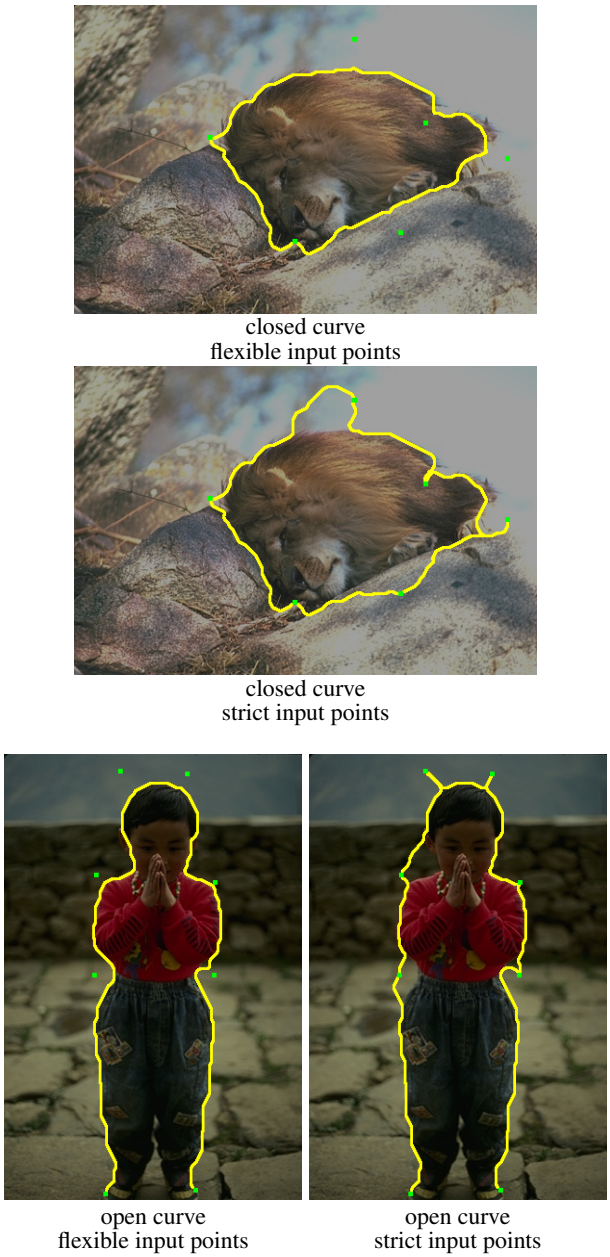


Figure 6. Strict versus flexible input points. When the input points are not precisely on the desired contour the proposed method performs much better than the strict variant where the contour has to go directly through all input points.

[6] M. Garey, R. Graham, and D. Johnson. Some NP-complete geometric problems. In *ACM Symposium on the Theory of Computing*, pages 10–22, 1976. 2

[7] L. Grady. Random walks for image segmentation. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 28(11):1768–1783, Nov. 2006. 1

[8] E. Mortensen and W. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384, Sept. 1998. 1

[9] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004. 1

[10] T. Schoenemann and D. Cremers. Globally optimal image segmentation with an elastic shape prior. In *IEEE Int. Conf. on Computer Vision*, Rio de Janeiro, Brasil, October 2007. 4

[11] A. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *IEEE Int. Conf. on Comp. Vision*, Rio de Janeiro, Brazil, Oct. 2007. 1

[12] M. Unger, T. Pock, D. Cremers, and H. Bischof. TVSeg - Interactive Total Variation based image segmentation. In *British Machine Vision Conference (BMVC)*, Leeds, UK, Sept. 2008. 1, 4, 5