Incorporating Topological Constraints within Interactive Segmentation and Contour Completion via Discrete Calculus*

Jia Xu   Maxwell D. Collins   Vikas Singh
University of Wisconsin-Madison
http://pages.cs.wisc.edu/~jiaxu/projects/euler-seg/

Abstract

We study the problem of interactive segmentation and contour completion for multiple objects. The form of constraints our model incorporates are those coming from user scribbles (interior or exterior constraints) as well as information regarding the topology of the 2-D space after partitioning (number of closed contours desired). We discuss how concepts from discrete calculus and a simple identity using the Euler characteristic of a planar graph can be utilized to derive a practical algorithm for this problem. We also present specialized branch and bound methods for the case of single contour completion under such constraints. On an extensive dataset of $\sim$ 1000 *images, our experiments suggest that a small amount of side knowledge can give strong improvements over fully unsupervised contour completion methods. We show that by interpreting user indications topologically, user effort is substantially reduced.*

1. Introduction

This paper is focused on developing optimization models for the problem of multiple contour completion/segmentation subject to side constraints. The type of constraints our algorithm incorporates are **(a)** those relating to inside (or outside) seed indications given via user scribbles; **(b)** global constraints on the topology, i.e., information which reflects the number of unique closed contours a user is looking for. Given the output from a boundary detector (e.g., **P**robability of **B**oundary or Pb [25]), we obtain a large set of *weighted* locally-based contours (or edgelets) as shown in Fig. 1. The objective then is to find k closed "legal" contour cycles with desirable properties (e.g., curvilinear continuity, strong edge gradient, small curvature), where legal solutions are those that satisfy the side constraints, shown in Fig. 1. The basic primitives in our construction are contour fragments, *not* pixels. The motivation for this choice is similar to most works on contour detection for image segmentation – by moving from predominantly region-based terms to a function that utilizes strength of edges, we seek to partly mitigate the dependence of the final segmentation on the *homogeneity* of the regions alone and the number of seeds. Additionally, in at least some circumstances, one expects benefits in terms of running time by utilizing a few hundred edges instead of a million pixels in the image. Our high level goal is the design of practical contour completion algorithms that take advice – which in a sense parallels a powerful suite of methods that have recently demonstrated how global knowledge can be incorporated within popular region-based image segmentation methods [26].

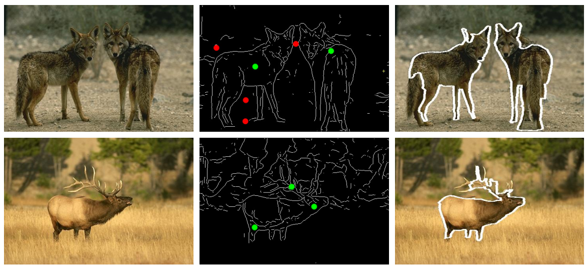
Figure 1: Left to right: input images, edgelets or contours with seed indications, and final contour. Foreground is marked in green; background is marked in red; boundrary is marked in white. Best viewed in color.

Related Work. The study of methods for detection of salient edges and object boundaries from images has a long history in computer vision [37]. The associated body of literature is vast – methods range from performing edge detection at the level of local patches [32], to taking the continuity of edge contours into account [37, 29], to incorporating high-level cues [36] such as those derived from shape and/or appearance [25]. While the appropriateness of a specific contour detector is governed by the downstream application, developments in recent years have given a number of powerful methods that yield high quality boundary detection on a large variety of images and perform well on established benchmarks [25]. Broadly, this class of methods uses local measurements to estimate the likelihood of a boundary at a pixel location. To do this, the conventional approach was to identify discontinuities in the brightness channel,

*This is a extended-CVPR version, dated May 1, 2013, with detailed optimization schemes and more experimental results.

where as newer methods exploit significantly more information. For instance, [27] suggests a logistic regression on brightness, color, and texture, and [9, 24] learns a classifier by operating on a large number of features derived from image patches or filter responses at multiple orientations. Contemporary to this line of research, there are also a variety of existing algorithms that integrate (or group) local edge information into a globally salient contour. Since one expects the global contour to be smooth, the well known Snakes formulation introduced an objective function based on first and second derivative of the curve. Others have proposed utilizing the ratio of two line integrals [18], incorporating curvature [31, 10], joining pre-extracted line segments [40, 35], and using CRFs to ensure the continuity of contours [30]. Note that despite similarities, contour detection on its own is not the same as image segmentation. In fact, even when formalized under *contour completion*, an algorithm may not always produce a closed contour. Nonetheless, from most "edge-based" methods one can obtain a partition of the image into object and background regions. Without getting into the merits of edges versus regions, one can view edge-based contours as a viable alternative to "region-based" image segmentation methods in many applications.

The success of the above developments notwithstanding, the applicability of these methods has been somewhat limited by their inability to successfully discriminate between contours of different classes of objects. To address this limitation, there has been a noticeable shift recently towards the incorporation of additional information within the contour completion process. In particular, several groups have presented frameworks that leverage category specific (or semantic) information into the process of obtaining closed object boundaries. Specific examples of this line of work include semantic contours [16], the hierarchical ultrametric contour map [2], and particle filtering based object detection via edges [23]. The basic idea here is to achieve a balance between bottom up edge/boundary detection and top-down supervision, for simultaneous image segmentation and recognition. While semantic knowledge based contour completion is quite powerful, its performance invariably depends on the richness of the underlying training corpus. Indeed, if the shape epitomes do not reflect the object of interest accurately enough (significant pose variations), if there is clutter/occlusion, or when a novel class is not well represented in the training data, the results may be unsatisfactory. In these circumstances, it seems natural to endow the contour completion models with the capability to leverage some form of user supervision (foreground and background seeds) [15]. Further, knowledge provided in the form of the *number* of closed contours a user requires, can be a powerful form of user guidance as well. Notice that the adoption of Grabcut type methods suggests that a nominal amount of "interactive scribbles" is readily available in many applications, and may significantly improve the quality of solutions. While there are many mechanisms which incorporate such constraints in region based segmentation, only a few methods take such information explicitly into account for edge-based contour completion. In this work, we leverage a discrete calculus based toolset to incorporate such topological and seed indications type supervision within a practical contour completion algorithm.

The primary contributions of the this paper are: **(i)** We present a unified optimization model for multiple contour completion/segmentation which incorporates topological constraints as well as inclusion/exclusion of foreground and background seeds. The topological knowledge is included by using the *Euler characteristic* of the edgelet graph where as inclusion/exclusion constraints utilize concepts from discrete calculus. **(ii)** For an extensive dataset, we provide strong evidence that with a small amount of user interaction, one can obtain high quality segmentations based on edge contours information alone. We give an easy to use implementation, as well as user scribble data corresponding to varying levels of interaction on this large (~ 1000) set of images.

2. Preliminaries

The tools of discrete calculus provide a powerful formalism to represent the topological information in an image [14, 20, 7]. We use conventions of discrete calculus to describe our problem of finding multiple contour closures. In this section, we introduce the idea of *cell complexes* which are the fundamental building blocks of our construction. The following text also introduces the necessary notations, which will be used thoughout the rest of the text.

2.1. Discrete Calculus

The domain of an image is decomposed into a set of *cells*. If the decomposition is such that (*i*) the interiors of the cells are disjoint and (*ii*) the boundary between any two p-dimensional cells is a $(p-1)$-dimensional cell then we have a *cell complex*. As an example, consider a planar graph $G = \langle V, E, F \rangle$ with vertices V, edges E, and faces F. Such a graph has *incidence* relationship between each face and its bounding edges, and between each edge and its endpoint vertices. Similarly, each vertex is incident on two or more edges and each edge is incident on two faces. Notice that the interior of a pair of faces is disjoint, and the boundary between any two faces gives an edge, where the dimension is reduced by one. As a consequence, we get a 2D cell complex for a planar graph, and also a set of incidence relationships among simplices of different dimensions.

A cell complex may be *oriented* such that we can describe directions on each cell relative to its orientation, see Fig. 2(a). Each type of cell has a corresponding pair of possible orientations: a vertex (0-cell) is either a source or

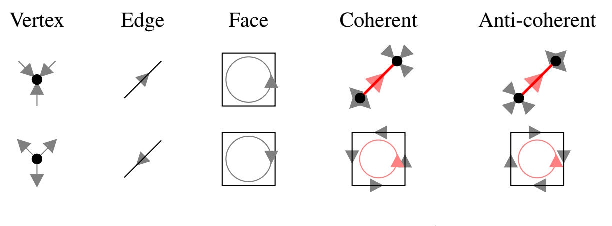

Figure 2: Visualization of the orientations on cells of different dimensionalities (a). In (b) we show in the left column p-cells with all of their boundary $(p-1)$-cells coherently oriented, and all boundary cells anti-coherently oriented in the right column.

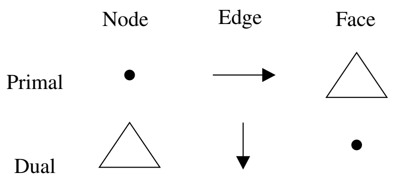

Figure 3: Duality relationships between 2D cell complices.

a sink while an edge (1-cell) may be directed toward either endpoint. Further, each cell *induces* a corresponding orientation on incident cells; for example, a directed edge has a source endpoint vertex at one end and sink at the other. The orientations of a cell and a member of its boundary are *coherent* if the induced orientations agree, an example is shown in Fig. 2(b).

We may represent the two-dimensional image as an oriented complex. All faces are given the same orientation, while edges and vertices are given arbitrary orientations. After enumerating its constituent vertices, edges and faces, a selection of some subset of faces is specified with an indicator vector $\mathbf{x} \in \{0,1\}^{|F|}$. $x_i = 1$ denotes the candidate face $F_i \in F$ is in the foreground, and $x_i = 0$ otherwise. Similarly, we represent the edge and vertex configuration of G by indicator vectors $\mathbf{y} \in \{0,1\}^{|E|}$ and $\mathbf{z} \in \{0,1\}^{|V|}$ respectively. We require that the indicator vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ on each level of cell consistently describe a segmentation. The key relationship is consistency between the labels on the *incident* cells. These relationships can be expressed algebraically using the notion of a dimension-appropriate *incidence matrix*. The edge-face incidence matrix (also called the boundary operator) $C_1 \in \{-1,0,1\}^{|E|\times|F|}$ is defined by

$$C_{1;ij} = \begin{cases} 1 & \text{if edge } i \text{ is incident to face } j \text{ and coherently oriented;} \\ -1 & \text{if edge } i \text{ is incident to face } j \text{ and anti-coherently oriented;} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Here, $C_{1;ij}$ refers to entry (i,j) in C_1. Similarly, by discarding orientation information, we can define the edge-face *corresponding matrix* $C_2 \in \{0,1\}^{|E|\times|F|}$ which labels which edges are incident to which face. It can be calculated as the element-wise absolute value of C_1, such that $C_{2;ij} = |C_{1;ij}|$. The node-edge incident matrix $A_1 \in \{-1,0,1\}^{|V|\times|E|}$ is defined analogously to (1), where $A_{1;ij} = 1$ iff node i is incident to edge j. As with C_2, we define the node-edge corresponding matrix $A_2 = |A_1| \in \{0,1\}^{|V|\times|E|}$. We further use a node-edge *degree matrix* $A_3 \in \mathbb{R}^{|V|\times|E|}$, where $A_{3;ij} = A_{2;ij}/d_i$ where d_i denotes the degree of node i.

Discrete calculus describes the notion of *duality* between cell complices. In a p-complex, each q-cell will have a corresponding dual $(p-q)$-cell (say, $q \le p$). For any given cell complex, we can construct its dual in a way that preserves incidence relationships between cells, see Fig. 3. Using these concepts, in the following sections, we will formalize the required constraints within a contour completion objective function.

3. Problem Formulation

As described in Section 2.1, our model works with selections of the cells constituting the foreground. Since the notion of foreground for a face is self-evident, we will describe the labeling of vertices and edges, starting from a face labeling $\mathbf{x}$. We enforce the following condition:

Condition 1. *A p-cell is in the foreground if and only if it is incident to a $(p+1)$-cell in the foreground.*

This condition ensures that each connected component of the foreground is itself a cell complex, a property we will use shortly.

First, we introduce an auxiliary indicator variable $\mathbf{w} \in \{0,1\}^{|E|}$ which selects the *boundary edges*. These edges are those which are incident to both a foreground and a background face. W.l.o.g., consider edge 1 incident to faces 1 and 2 respectively, then $w_1 = |x_1 - x_2| = \mathbb{I}(x_1 \neq x_2)$. Taken together, the full set of boundary edges precisely represent the contour of the selected foreground. We can now use the *boundary operator* from Section 2.1 to derive the identity

$$\mathbf{w} = |C_1\mathbf{x}| \tag{2}$$

Observe that each edge is incident to exactly two faces, and we specified that all faces have identical orientation. It follows that an edge must be coherent with one face and anti-coherent with the other. Therefore, for all *internal edges* (non-boundary edges in the foreground) the C_1 operator when multiplied with $\mathbf{x}$, cancels the contribution from these two faces, leaving non-zero values only for the boundary edges. The internal edges (which are incident to foreground faces on both sides) can still be computed in a different manner. The vector $C_2\mathbf{x}$ will count the inside edges twice and the boundary edges once, as we discard orientation (and thus sign information). In the preceding, w.l.o.g. $(C_2\mathbf{x})_1 = x_1 + x_2$. Thus, Condition 1 will be satisfied if the following identity holds:

$$2\mathbf{y} = \mathbf{w} + C_2\mathbf{x} \tag{3}$$

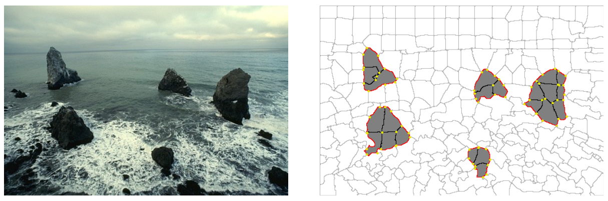

Figure 4: A superpixel-based segmentation with the foreground subgraph consistent under condition 1. Selected faces are shaded, foreground edges are bold and foreground vertices highlighted in yellow. Internal edges $\mathbf{y}_i \neq \mathbf{w}_i = 0$ are bold/black, boundary edges $\mathbf{y}_i = \mathbf{w}_i = 1$ are red.

We use the matrices A_2, A_3 for a pair of linear inequalities which are equivalent to Condition 1 *for vertices*. Observe that the vector $A_2\mathbf{y}$ will be the number of foreground edges incident to each foreground vertex (or node), where $(A_2\mathbf{y})_i$ is the number of foreground edges incident to vertex (or node) i. Similarly, when scaled by the degree d_i of vertex i, $(A_3\mathbf{y})_i \in [0, 1]$ will be the *proportion* of edges incident to i which are in foreground. Enforcing condition 1 is equivalent to:

$$A_3\mathbf{y} \leq \mathbf{z} \leq A_2\mathbf{y} \tag{4}$$

Since $\mathbf{z}_i \in \{0, 1\}$, the condition, $\mathbf{z}_i \geq (A_3\mathbf{y})_i$, will be true only for $\mathbf{z}_i = 1$ if any edge incident to i is in foreground. Conversely, if no edge incident to i is selected in the solution, then $(A_2\mathbf{y})_i = (A_3\mathbf{y})_i = 0$ and (4) is satisfied only for $\mathbf{z}_i = 0$.

The expressions introduced above allow the identification of whether a user provided seed falls "inside" or "outside" the contour completion given by $\mathbf{w}$, and will serve as constraints for our multiple contour completion model. Fig. 4 shows an illustrative example for an image, where the input to the contour completion are edgelets (or edgels) obtained from boundaries of a globalPb derived superpixels.

Euler Characteristic. Our final requirement is to be able to specify the *number* of closed contours desired. The existing literature on region based image segmentation provides some ideas on how this can be accomplished for random field based models – in the form of so-called connectedness constraints. TopologyCuts is an extension of graphcuts and utilizes certain levelset ideas to preserve topology [41]. The DijkstraGC [38] finds a segmentation where two manually indicated seed points are connected via the foreground where as Nowozin [28] makes use of a LP relaxation. Very recently, [8] proposed selectively perturbing the energy function to ensure topological properties. Here, we show how a much simpler form can capture the desired topological properties, as described next.

For any graph we can define the *Euler characteristic* as

$$\chi = |V| - |E| + |F|, \tag{5}$$

where $\chi = 2$ for any planar embedding of a graph. If we explicitly constrain that the Euler characteristic of an induced subgraph created by selecting any given foreground is exactly two, this will give a foreground region that is connected and simple in a geometric sense. For multiple connected regions, we can use the generalized form of this formula for arbitrary planar graphs:

$$|F| + |V| - |E| = n + 1 \tag{6}$$

where n is the number of connected components.)

Lemma 3.1. *Let* $\mathbf{x}, \mathbf{y}, \mathbf{z}$ *denote indicator vectors for the selection of faces, edges, and vertices for planar graph* G. *The selected subgraph will satisfy* (6) *if*

$$\sum_i x_i + \sum_k z_k - \sum_j y_j = n \tag{7}$$

Proof. (Sketch) The left-hand side of this formula counts each relevant quantity for the Euler characteristic of the selected subgraph, but it neglects to count the "outside" face. Subtract one from the RHS and derive the equality. □

This will not count the extra outside faces corresponding to any "holes". This was not a problem in our experiments, but can be explicitly avoided by requiring the background be connected using the spanning tree constraints of [33]. Using (7) as a constraint in our model will guarantee that we recover n *simply connected* foregrounds.

3.1. Optimization Model

Before we introduce the contour completion model, we briefly describe the procedure for deriving the components of the graph from an image. This process follows existing algorithms for contour and boundary detection. First, we run the globalPb detector on an image which provides the probability of boundary for each image pixel. Next, we generate a set of superpixels from the image using the globalPb output in conjunction with TurboPixels (which uses local information and compactness). Each superpixel corresponds to a face, and the boundary of the superpixel corresponds to edges in the graph (these are the basic primitives of the closed contours we will derive). If two edges are connected, we introduce a node in the graph. With this construction, the problem of finding multiple contour closures reduces to finding multiple cycles in the graph. To select the cycles for the strongest contours, we want to weight the edges appropriately. For this purpose, we calculate two types of weight measures following [21]. The first, denoted by $\mathbf{N}$, measures the "goodness" of edges. The better edge i is, the smaller N_i will be. The second, denoted by $\mathbf{D}$, is the count of all the pixels on the superpixel boundary. We use an objective function which is the ratio of these quantities, $\frac{\mathbf{N}(\mathbf{w})}{\mathbf{D}(\mathbf{w})}$. This ends up being the portion of contour w.r.t arclength which does not lie on a true image edge. Minimizing this quantity has been shown to provide a contour that has strong edge support in the image.

Finally, the user indictations are represented in terms of indicator vectors $\mathbf{x}_0, \mathbf{x}_1$, where $\mathbf{x}_{0;i} = 1$ if face i contains a background seed. With the basic components (or constraints) in hand, we now have the main optimization model.

$$\begin{aligned}\min_{\mathbf{w},\mathbf{x},\mathbf{y},\mathbf{z}} \quad & \frac{\mathbf{N}^T\mathbf{w}}{\mathbf{D}^T\mathbf{w}}, \\ \text{s.t.} \quad & \mathbf{w} = |C_1\mathbf{x}|, \quad 2\mathbf{y} = \mathbf{w} + C_2\mathbf{x}, && \text{(8a,b)} \\ & A_3\mathbf{y} \le \mathbf{z} \le A_2\mathbf{y}, \quad 1^T\mathbf{x} + 1^T\mathbf{z} - 1^T\mathbf{y} = n, && \text{(8c,d)} \\ & \mathbf{x}_1 \le \mathbf{x} \le 1 - \mathbf{x}_0, \quad \mathbf{w},\mathbf{x},\mathbf{y},\mathbf{z} \in \{0,1\}. && \text{(8e,f)}\end{aligned}$$

3.2. Optimizing Ratio Objective

Since the objective in (8) of the main paper is in ratio form, we transform it into a linear function with a free variable, t. Our linear ratio cost objective function is solved by minimizing $f(t,\mathbf{u}) = (\mathbf{N} - t\mathbf{D})^T\mathbf{u}$, over admissible $\mathbf{u}$ for a sequence of chosen values of t. Here, $\mathbf{u}$ denotes the concatenated vector of all indicator variables in the model. Assume $\mathbf{D} \ge 0$ and $\mathbf{D}^T\mathbf{u} \ne 0$. For an initial finite bounding interval $[t_l, t_u]$, let t_0 be the initial value. Let $\bar{\mathbf{u}} = \arg\min_{\mathbf{u}} f(t_0,\mathbf{u})$, the procedure proceeds as follows:

- $f(t_0,\bar{\mathbf{u}}) = 0$: $\mathbf{N}^T\bar{\mathbf{u}}/\mathbf{D}^T\bar{\mathbf{u}} = t_0$, stop with solution t_0
- $f(t_0,\bar{\mathbf{u}}) < 0$: $\mathbf{N}^T\bar{\mathbf{u}}/\mathbf{D}^T\bar{\mathbf{u}} < t_0$, $t_u \leftarrow \mathbf{N}^T\bar{\mathbf{u}}/\mathbf{D}^T\bar{\mathbf{u}}$
- $f(t_0,\bar{\mathbf{u}}) > 0$: $\mathbf{N}^T\bar{\mathbf{u}}/\mathbf{D}^T\bar{\mathbf{u}} > t_0$, $t_l \leftarrow t_0$

Each iteration is easily solved in a few seconds using the CPLEX IP solver on a standard workstation.

3.3. Spanning Tree Constraint

In Section 3 of the main paper, we stated that the $\sum_i x_i$ term in (7) will not count the "outside" faces. In addition, it does not count faces introduced due to holes in the selected foreground. If we denote the number of such holes by H, then the actual number of faces of the foreground subgraph is $\sum_i x_i + H + 1$. Modifying the expression to match (6), this takes the form

$$\begin{aligned}\left(\sum_i x_i + H + 1\right) + \sum_k z_k - \sum_j y_j &= C + 1 \\ \sum_i x_i + \sum_k z_k - \sum_j y_j &= C - H\end{aligned} \tag{9}$$

where C is the number of connected components. There remains a small ambiguity in the constraint, such that introducing a new connected component along with a new hole will maintain the equality. This can be easily eliminated either via a choice of objective which favors minimum arc length (and thus will avoid holes), or by *explicitly* restricting $H = 0$.

If the background faces form one connected component, there are no holes inside the selected foreground cycles, This is acchievable with the constraints of [33] to require the existence of a spanning tree on the dual graph of unselected faces.

Denote $\bar{\mathbf{x}} = 1 - \mathbf{x}$ as the faces we did not select in our solution. We introduce auxiliary variables for a face-simultaneous-selection matrix S, and variable T indicating which dual edges are in the spanning tree.

$$S_{ij} = \begin{cases} 1 & \text{if } \bar{\mathbf{x}}_i = \bar{\mathbf{x}}_j = 1, \text{ faces } i \text{ and } j \text{ adjacent;} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

The spanning tree is constructed from S using the following constraints.

First, if $S_{ij} = 0$, this cannot be an edge in the tree.

$$T_{ij} \le S_{ij} \quad \forall\, i,j \tag{11}$$

Second, if $\bar{\mathbf{x}}_i = 1$, there must be at least one edge incident on face i

$$\sum_{j\sim i} T_{ij} \ge \bar{\mathbf{x}}_i \quad \forall i \tag{12}$$

All the background faces should form a tree. A graph is a tree only if it has one fewer edge than faces

$$\sum_{i\sim j} T_{ij} = \sum_i \bar{\mathbf{x}}_i - 1 \tag{13}$$

Finally, one eliminates cycles by ensuring all *subsets* of faces are no more connected than a tree

$$\sum_{i\sim j; i,j\in S} T_{ij} \le \sum_{i\in S} \bar{\mathbf{x}}_i - 1 \quad \forall\, S \subset F \tag{14}$$

this is enforced for *all* subsets S of faces. If a feasible T exists, then the background must be connected and there are no holes in the foreground. We do not use (14) in our solver, and present it for theoretical completeness. Instead, we rely in practice on the tendency of our model's *objective* to prefer short, simple boundaries which do not introduce unnecessary holes.

4. Beyond Superpixel-derived Edgelets

Recall that the model in Section 3.1 constructs a cell complex using a superpixel decomposition of the image domain. While fast algorithms for finding this decomposition are available [22], it is known that superpixels are not robust for all types of images. Occlusion or weak boundaries give cases where the set of superpixel boundary primitives (the input to our optimization) do *not* include some valid edgelets (ones which have not been picked up by either the contour detector or superpixel method). The natural solution to this is to *supplement* the basic set of edgelet primitives with additional contour pieces that bridge the 'gaps' and allow a more accurate contour closure even in the presence of very weak signal variations. Next, we present such an extension to find completions using a base set of disconnected edgelets. But introducing completions between all

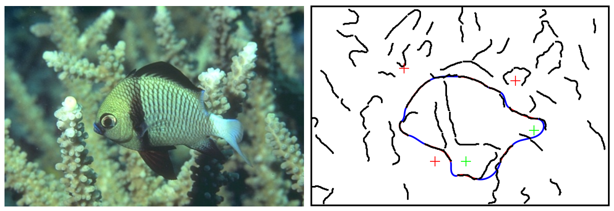
Figure 5: Branch-and-bound result on a BSD image.

pairs of edgelets is prohibitive and leads to a problem with a large number of variables (especially for multiple contours). The following model, while applicable to the multiple contour setting, is most effective for finding *a single contour* which encloses a simply connected foreground region.

Euler Spirals. A key subcomponent of this problem is how to join two edgelets which will follow each other on the contour. This is the problem solved by [19] which proposes to use segments of the *Euler spiral*. This spiral can be shown to be the curve $\mathcal{C}$ with minimal *total curvature* $\mathrm{TC}_2 = \int_{\mathcal{C}} \kappa(s)^2 \; ds$ where $\kappa(s)$ is the curvature at a given point on the curve parameterized by arc-length. For any pair of points along with tangents we can construct a segment of an euler spiral which connects these points with consistent tangents. They show that these completions satisfy the conditions given by [17] for a "pleasing" curve (invariance to similarity transformations, symmetry, extensibility, smoothness, roundness).

We parameterize the spiral by the turning angle as in [39]. To form a completion, we consider the Euler spiral under a similarity transformation determined by the position and Frenet frame $(\mathbf{P}_0, \mathbf{T}_0, \mathbf{N}_0)$ at the spiral's *inflection point*, and a scaling factor a. The transformed spiral is

$$Q(\theta) = \begin{cases} \mathbf{P}_0 + aC(\theta)\mathbf{T}_0 + aS(\theta)\mathbf{N}_0 & \theta \geq 0 \\ \mathbf{P}_0 - aC(-\theta)\mathbf{T}0 - aS(-\theta)\mathbf{N}_0 & \theta < 0 \end{cases}$$

where S and C are the *Fresnel integrals*. A choice of interval $[\theta_1, \theta_2]$ selects a given segment. [39] gives a set of equations to determine these free variables, given segment endpoints $\mathbf{P}_1, \mathbf{P}_2$ and their tangents $\mathbf{T}_1, \mathbf{T}_2$. We solve these equations using a modified Newton's method. The most expensive step, the computation of the Fresnel integrals, is sped up considerably using [12], but augmented with precomputed tables. We can compute an average completion in 30μs, versus 1ms for [19] on the same machine, making it an attractive option to calculate a large number of completions, quickly, within the core contour completion engine.

Euler-Spirals for One Contour Completion. We are given a set of image edgelets derived from an edge detector as before, as well as user-provided foreground and background seeds. The core objective considered by the algorithm is an *alternating path* p which consists of a sequence of edgelets joined by Euler Spiral segments. The goal is to find a closed contour that minimizes an objective function that increases with the addition of each contour segment.

Our solution strategy is to iteratively build upon the current *partial path*, until we get a cycle that encloses a feasible region. To do this, we adopt a specialized branch and bound procedure. Here, each node v of the branch-and-bound tree corresponds to some alternating path p. If p is a cycle, then v is a leaf node and thus a candidate solution. In this case, we check p is checked for feasiblity w.r.t. the seed constraints. If p is *not* a cycle, we may construct the children of this node by considering each image edglet in sequence and calculating the euler completion, on the fly. The path for the a child is then p plus the current completion and edgelet appended to the end. Children are discarded if they give rise to a self-intersecting partial path; therefore, entire subtrees can be discarded directly. Any partial path with objective worse than the best candidate solution found so far may be ignored. Otherwise, we descend the tree to each child in turn, ordered by the cost of their partial contour.

This algorithm implicitly solves a model of the form in (8), with a linear objective function on $\mathbf{w}$ and smoothness constraints on the solution contour. We can construct a planar graph for this model using the *extensibility* property of Euler spirals and splitting any two intersecting segments.

4.1. Branch-and-Bound Method

We give details on the solver which solves our model without explicitly constructing the full cell complex.

Construction. The solver is given a set of image *edgels* E and seeds. The branch-and-bound algorithm considers partial solutions p to a contour completion problem. p is an alternating path which consists of a sequence of edgels joined by Euler Spiral segments. The *children* of a branch-and-bound node are simply those contours which extend p by a single completion and edgel

$$\begin{aligned} \mathrm{children}(p) = &\{\mathrm{concatenate}(p, \mathcal{C}, e) \mid \\ &\mathcal{C} \text{ joins } \mathrm{tail}(p) \text{ and } e \; \forall e \in E\} \end{aligned}$$

Cost Function. We seek a closed p which minimizes some integral cost over the contour, for instance the elastica energy

$$C(p) = \int_p \alpha\kappa_p(s)^2 + \beta \; ds \qquad (15)$$

Note that $\beta > 0$ suggests using completions based on general elastica [17], though our experiments suggest using negligible $\beta \ll \alpha$. Note that any cost of this form will satisfy $C(q) \leq C(p)$ for any $q \in \mathrm{children}(p)$.

Overview. In order to allow a flexible node visit order, we use a *priority queue* over partial solutions. This provides an `enqueue` operation which places an contour in the queue, and a `dequeue` operation which removes the queue element with minimum cost and removes it. If every time we `dequeue` a contour we `enqueue` its children this will iterate over all possible contours in order of increasing

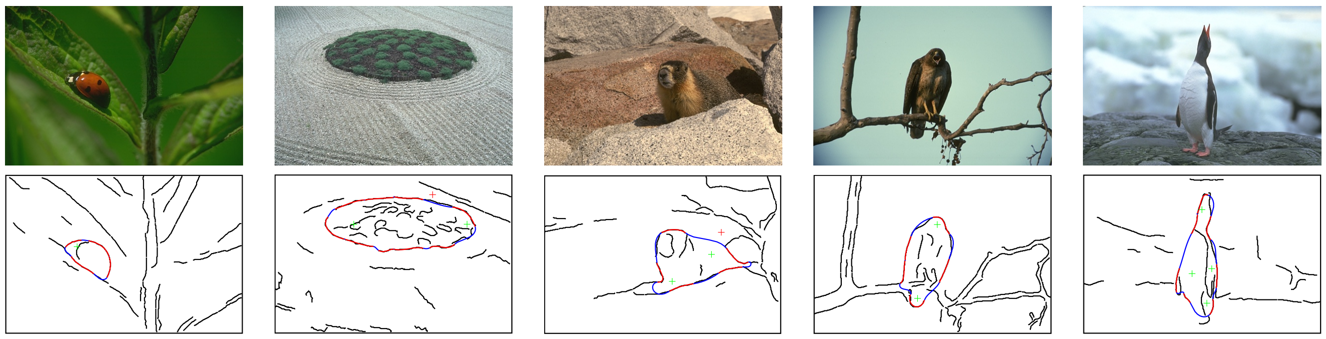

Figure 6: Additional results for the branch-and-bound algorithm for images from the Berkeley Segmentation Dataset. The input edgels are shown in black, euler completions in blue. The red segments are input edgels, smoothed for accurate derivative estimation.

cost. As soon as we find a feasible closed contour this is the solution.

4.2. Pseudocode

```
function expandnode($p$, $E$, $seeds$) {
    if $p$ is self-intersecting then
        return $\emptyset$
    end

    if $p$ is closed then
        if $p$ is feasible for $seeds$ then
            return $p$ as the solution
        end
    else
        for $q \in$ children($p$) do enqueue($q$)
    end

    return $\emptyset$
}

function contour($E, seeds$) {
    ; Enqueue root nodes
    for $e \in E$ do enqueue($e$)

    ; Main Loop
    while queue non-empty do
        $p \leftarrow$ dequeue()
        $p \leftarrow$ expandnode($p, E, seeds$)
        if $p \neq \emptyset$ then
            return $p$
        end
    end while

    return $\emptyset$
}
```

The contour function will return the minimum-cost contour for a given set of edgels and seed points. Note that this is a highly parallelizable algorithm: a set of threads can all be running the main loop independently with their only interaction through the node queue.

There is a natural extension of this method to multiple contours. Each node handles multiple partial contours, and each child of that node appends an edgel to one of the incomplete contours. However, the branching factor of this tree may become large.

5. Experiments

5.1. Dataset and Experiments Setting

We first provide evaluations of the model from Section 3 on images from the Weizmann Horse Database (WHD) [5], the Weizmann Segmentation Database (WSD) [1], and the Berkeley Segmentation Data Set (BSDS500) [3]. We then continue to experiments with a *robot user* on the ISEG dataset. These experiments will show that the combination of interaction with a contour-based method can achieve high levels of accuracy with a minimum of user effort.

We compare our approach (which we refer as EulerSeg) with three other contour grouping methods: (*i*) Ratio Region Cut (RRC) from [34], (*ii*) Superpixel Closure (SC) from [21], and an adaptive grouping method (EJ) [11]. We note that these are unsupervised whereas our algorithm incorporates user interaction, but SC and EJ produce multiple segmentations of which we select the most favorable. We compute the F-measure by the region overlapping and report quantitative results in Fig. 10.

The cell complex is generated from superpixels via [22] and the same number of superpixels as SC in all our experiment. We typically indicate $1 \sim 2$ interior seeds for the sought objects, but in the presence of ≥ 2 objects, we may need $3-7$ points including both interior and exterior seeds. The indicated seeds are shown in the images: green marks are foreground and red marks are background.

RRC was run using the default parameters $\lambda = 0, \alpha = 1$. That method has an additional parameter to indicate an arbitrary number of objects. However, it frequently fails to get a second boundary even when the image includes 2 objects. For SC, we use their reported best parameters with the number of superpixels set to 200 and $T_e = 0.05$. That algorithm generates $K = 10$ possible solutions, here we report results for the best one.

5.2. Qualitative Evaluation on Contour Completion

WHD Results: WHD consists of 328 side-view images of horses, with exactly one horse in each image. Fig. 7 shows both RRC and SC select large regions of ground between the horses' legs due to their large-region bias. As the examples show, our objective function minimizes gaps in the closure and leverages user seeds to handle slender objects better and outperforms both with ≤ 5 seeds.

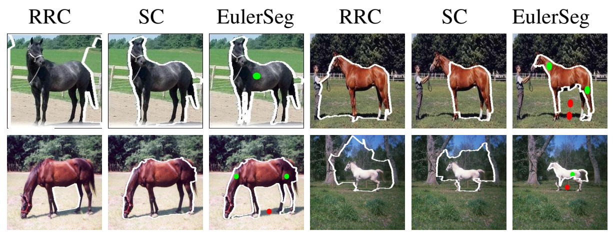

Figure 7: Sample results from WHD. **Best viewed in color.**

WSD Results: WSD contains 200 images and is divided into 2 subsets of images with one or two foreground objects. As shown in Fig. 8, our algorithm is comparable to RRC and SC when there is one object with only one seed. However, when the image contains 2 objects, our Euler characteristic constraint fires in and we correctly segment both objects of interest, while RRC and SC either selects one of the objects or segments one large region which includes both.

BSDS500 Results: Compared with WSD and WHD, images in this dataset are more complicated. We note that in some images of BSDS500, there are no salient objects or closed contours (e.g., images of sky or street). In these cases our algorithm cannot find a meaningful closed contour, but where one is present our model performs at least as well as any of the compared methods. However, another challenging class of images in BSD are those that depict a large number of foreground objects, here our algorithm significantly improves upon previous results with a small amount of user guideline and the topological constraint. An example of this can be seen in the bottom row of Fig. 9, where RRC and SC fail whereas our method is able to find the correct solution easily.

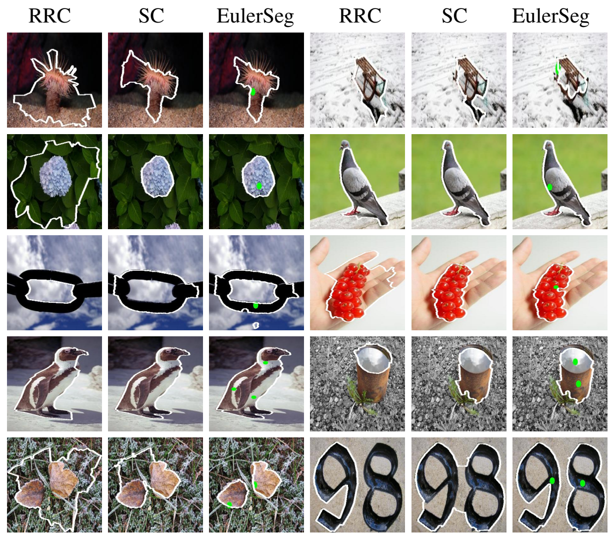

Figure 8: Sample results from WSD. **Best viewed in color.**

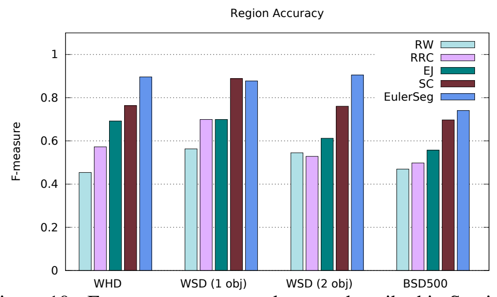

Figure 10: F-measure scores on datasets described in Section 5.

5.3. Quantitative Evaluation on Contour Completion

For a region A from an algorithm and a region B from the ground truth, we define the precision as the ratio of true points on A:

$$P = \frac{|\text{Matched}(A, B)|}{|A|} \tag{16}$$

and recall as the proportion of detected points on B:

$$R = \frac{|\text{Matched}(B, A)|}{|B|} \tag{17}$$

where $|\text{Matched}(A, B)|$ is the intersected pixels of the segmented region and ground truth. We define our F-Measure as

$$F = \frac{2PR}{P + R} \tag{18}$$

The average performance of the four algorithms (RRC, EJ, SC, and ours) is shown in Fig. fig:accuracy. In the BSD500 truth, as the images are parsed into a few number of regions (≥ 5), we use our seed points to extract a binary ground truth, with any regions marked with a foreground seed placed in the foreground. We also compare here to a basic supervised method from the region/graph-based setting, Random Walker (RW) [13] using the same seeds. Fig. 10 shows EulerSeg (our algorithm) performs comparably with SC on the WSD with one object while on the WHD , WSD with 2 objects and BSD500, our algorithm performs significantly better than the four baseline algorithms.

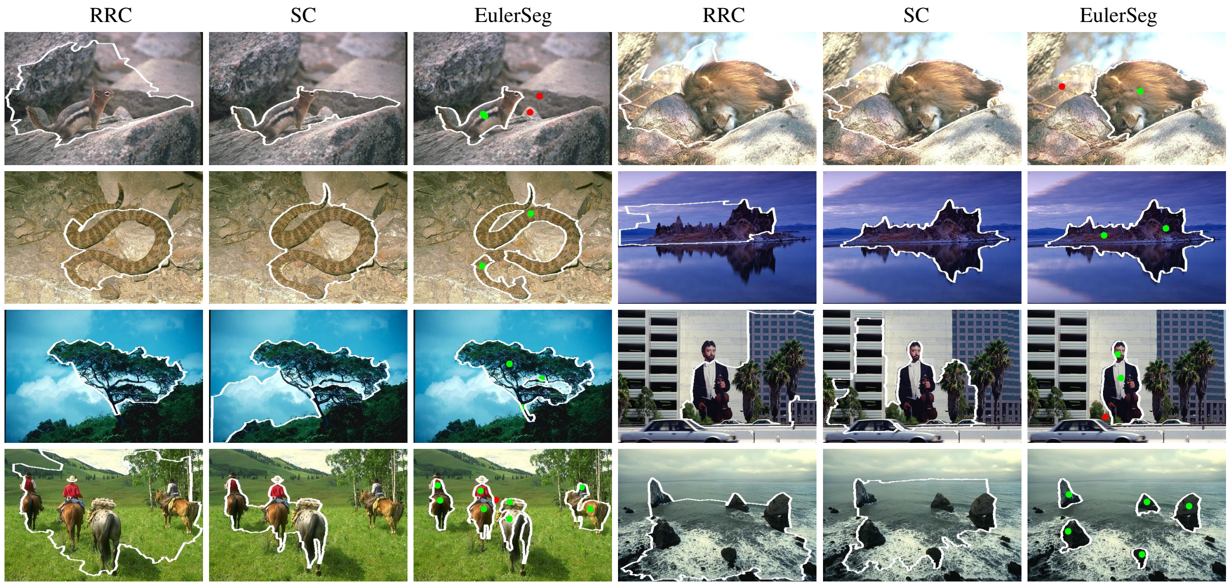

Figure 9: Sample results from BSDS500. **Best viewed in color.**

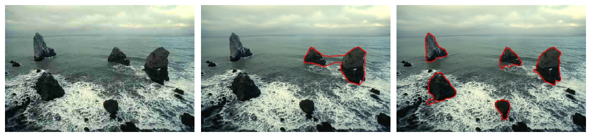

Figure 11: Example of multiple closures

5.4. Multiple Closures

As mentioned in [34], the authors attempt to solve multiple contour closures by removing all the edges associated with the detected one and repeating their single-detection method. However, this approach is problematic as shown in Fig. 11. If the single-detection method select two closures at the very beginning (shown in the middle of Fig. 11) and removes all the edges related to these two, It is not possible to get these two closures back in subsequent step of their algorithm. However, our algorithm can select all five closures in one shot as shown on the right of Fig. 11.

5.5. Results on Interactive Segmentation

ISEG Results: We compare our algorithm with the state-of-art interactive segmentation methods on the ISEG dataset[15]. These include Boykov & Jolly (BJ) with no shape constraints [6], shortest paths method (SP) [4], Random Walker (RW) [13], and Geodesic Star Convexity sequential system (GSCseq) [15]. We measure the effects of user interactions using a robot user setting. All the algorithms are set up with the default setting using the robot engine from [15]. The question we ask is how much user interaction is required to get a region F-measure score of 0.95 for the ISEG dataset (restricted to cases where all algorithms can achieve F=0.95 within 20 strokes). Table 1 demonstrates that EulerSeg requires the fewest stokes to reach a reasonable segmentation. On the other hand, as ISEG already provides a good initialization, which benefits the rest methods for building up an appearance model, the extra effort needed for a good segmentation is reduced. It is important to note that seeds in EulerSeg act as a pure geometric role and enable segmentation with fewer stroked pixels. These results are shown in Fig. 12.

Table 1: Average interaction efforts required to reach an F=0.95

Method	BJ	RW	SP	GSCseq	EulerSeg
Avg. Effort	5.51	6.48	4.54	2.30	2.06

ISEG Results without Initialization: As ISEG already provides a good initialization, which benefits the other methods for building up an appearance model, the extra effort needed for a good segmentation is reduced. It is important to note that seeds in EulerSeg act as a pure geometric role and enable segmentation with fewer stroked pixels. When starting with no initialization, EulerSeg is still able to segment the object(s). Here we provide additional results on our algorithm without initialization (which we refer as EulerSeg-0). In EulerSeg-0, we start our segmentation without any seeds, then we use the robot engine to add seed points iteratively.

Fig. 12 shows the first segmentation found by the robot user which has a region F-Measure of at least 0.95. The varying number of strokes seen between different algorithms on the same image shows the amount of additional input necessary to achieve this level of accuracy. Fig. 12 along with Table 1 in the main paper demonstrates that by interpreting the seeds topologically, the user interactions needed to get high-accuracy segmentation can be significantly reduced.

Running Time The preprocessing to generate superpixels is the primary computational cost, and is the only resolution-dependent component of our method. The total number of variables in our ILP typically is about 2000 (with residuals); on a 3GHz i7 CPU, each iteration of the linear ratio objective solver takes $< 1s$. Given superpixels, our implementation creates a segmentation usually within 15 iterations, though for some exceptionally textured images or those with a large number of components our algorithm may take more than 1 minute to solve.

6. Discussion

We present a framework based on discrete calculus which unifies the contour completion and segmentation settings. This is augmented with a Euler characteristic constraint which allows us to specify the topology of the segmented foreground. Our model easily accommodates user indications and multiple foreground regions. Two solvers specialized toward different aspects of the problem are derived, one based on an ILP over superpixels and the other a branch-and-bound using completions with spirals to join edgelets. We demonstrate our model finds salient contours across a large dataset, showing significant improvement over similar methods.

Acknowledgments: This work is funded via grants NIH R01 AG040396 and NSF RI 1116584. Partial support was provided by UW-ICTR and Wisconsin ADRC. Collins was supported by a CIBM fellowship (NLM 5T15LM007359).

References

[1] S. Alpert, M. Galun, R. Basri, and A. Brandt. Image segmentation by probabilistic bottom-up aggregation and cue integration. In *CVPR*, 2007. 7

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *CVPR*, 2009. 2

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011. 7

[4] X. Bai and G. Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *IJCV*, 82(2):113–132, 2009. 9

[5] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *ECCV*, 2002. 7

[6] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, 2001. 9

[7] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *CVPR*, 2010. 2

[8] C. Chen, D. Freedman, and C. Lampert. Enforcing topological constraints in random field image segmentation. In *CVPR*, 2011. 4

[9] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006. 2

[10] N. Y. El-Zehiry and L. Grady. Fast global optimization of curvature. In *CVPR*, 2010. 2

[11] F. J. Estrada and A. D. Jepson. Robust boundary detetion with adaptive grouping. In *POCV*, 2006. 7

[12] O. L. Fleckner. A method for the computation of the fresnel integrals and related functions. *Mathematics of Computation*, 22(103):635–640, 1968. 6

[13] L. Grady. Random walks for image segmentation. *PAMI*, 28(11):1768–1783, 2006. 8, 9

[14] L. Grady and J. R. Polimeni. *Discrete Calculus: Applied Analysis on Graphs for Computational Science*. Springer, 2010. 2

[15] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *CVPR*, 2010. 2, 9, 12

[16] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 2

[17] B. Horn. The curve of least energy. *ACM Trans. Math. Soft.*, 9(4):441–460, 1983. 6

[18] I. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *PAMI*, 23(10):1075–1088, 2001. 2

[19] B. B. Kimia, I. Frankel, and A.-M. Popescu. Euler spiral for shape completion. *IJCV*, 54(1-3):159–182, 2003. 6

[20] V. A. Kovalevsky. Finite topology as applied to image analysis. *Computer Vision, Graphics, Image Processing*, 46(2):141–161, 1989. 2

[21] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Optimal contour closure by superpixel grouping. In *ECCV*, 2010. 4, 7

[22] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *PAMI*, 31(12):2290–2297, 2009. 5, 7

[23] C. Lu, L. Latecki, N. Adluru, X. Yang, and H. Ling. Shape guided contour grouping with particle filters. In *ICCV*, 2009. 2

[24] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, et al. Discriminative sparse image models for class-specific edge detection and image interpretation. In *ECCV*, 2008. 2

[25] M. Maire, P. Arbeláez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *CVPR*, 2008. 1

[26] S. Maji, N. Vishnoi, and J. Malik. Biased normalized cuts. In *CVPR*, 2011. 1

[27] D. R. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, 2004. 2

[28] S. Nowozin and C. Lampert. Global interactions in random field models: A potential function ensuring connectedness. *SIAM J. Imag. Sci.*, 3(4):1048–1074, 2010. 4

[29] P. Parent and S. Zucker. Trace inference, curvature consistency, and curve detection. *PAMI*, 11(8):823–839, 1989. 1

[30] X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. In *ICCV*, 2005. 2

[31] T. Schoenemann and D. Cremers. Introducing curvature into globally optimal image segmentation: Minimum ratio cycles on product graphs. In *ICCV*, 2007. 2

[32] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *ICCV*, 2005. 1

[33] M. Singh and L. C. Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *STOC*, 2007. 4, 5

[34] J. S. Stahl and S. Wang. Edge grouping combining boundary and region information. *TIP*, 16(10):2590–2606, 2007. 7, 9

[35] J. S. Stahl and S. Wang. Globally optimal grouping for symmetric closed boundaries by combining boundary and region information. *PAMI*, 30(3):395–411, 2008. 2

[36] Z. Tu, X. Chen, A. Yuille, and S. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *IJCV*, 63(2):113–140, 2005. 1

[37] S. Ullman and A. Shaashua. Structural saliency: The detection of globally salient structures using a locally connected network. Technical report, MIT, 1988. 1

[38] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *CVPR*, 2008. 4

[39] D. J. Walton and D. S. Meek. G1 interpolation with a single cornu spiral segment. *Journal of Computational and Applied Mathematics*, 223(1):86–96, 2009. 6

[40] S. Wang, T. Kubota, J. M. Siskind, and J. Wang. Salient closed boundary extraction with ratio contour. *PAMI*, 27(4):546–561, 2005. 2

[41] Y. Zeng, D. Samaras, W. Chen, et al. Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation. *CVIU*, 112:81–90, 2008. 4

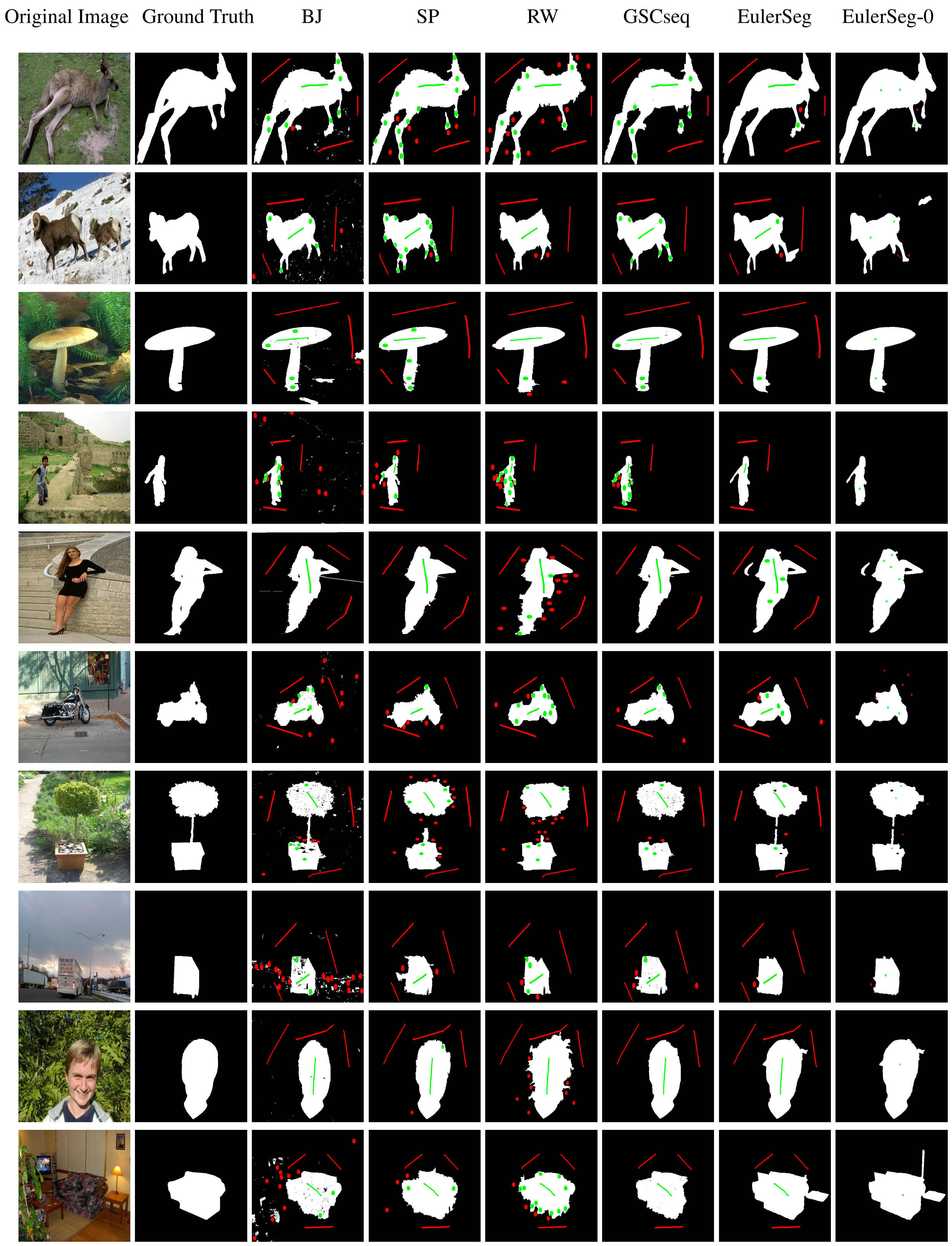

Figure 12: Sample results from ISEG. Red strokes are background seeds while green strokes are foreground seeds. Strokes for column 3-7 are the default setting in the robot engine [15] with brush radius equal to 8 pixels, while strokes feed in EulerSeg-0 are simple point seeds, whose radius is one pixel. We marked seeds for EulerSeg-0 as crosses just for noticeability. **Best viewed in color**.