10-2007

Edge Grouping Combining Boundary and Region Information

Joachim S. Stahl

Song Wang
University of South Carolina - Columbia, songwang@cse.sc.edu

Publication Info

Published in *IEEE Transactions on Image Processing*, Volume 16, Issue 10, 2007, pages 2590-2606.

Edge Grouping Combining Boundary and Region Information

Joachim S. Stahl, *Student Member, IEEE*, and Song Wang, *Member, IEEE*

***Abstract*—This paper introduces a new edge-grouping method to detect perceptually salient structures in noisy images. Specifically, we define a new grouping cost function in a ratio form, where the numerator measures the boundary proximity of the resulting structure and the denominator measures the area of the resulting structure. This area term introduces a preference towards detecting larger-size structures and, therefore, makes the resulting edge grouping more robust to image noise. To find the optimal edge grouping with the minimum grouping cost, we develop a special graph model with two different kinds of edges and then reduce the grouping problem to finding a special kind of cycle in this graph with a minimum cost in ratio form. This optimal cycle-finding problem can be solved in polynomial time by a previously developed graph algorithm. We implement this edge-grouping method, test it on both synthetic data and real images, and compare its performance against several available edge-grouping and edge-linking methods. Furthermore, we discuss several extensions of the proposed method, including the incorporation of the well-known grouping cues of continuity and intensity homogeneity, introducing a factor to balance the contributions from the boundary and region information, and the prevention of detecting self-intersecting boundaries.**

***Index Terms*—Boundary detection, edge grouping, edge linking, graph models, perceptual organization.**

I. Introduction

GROUPING (or *perceptual organization*) is an important problem in computer vision and image processing that seeks to identify perceptually salient structures in noisy images. This is usually achieved by first constructing a set of tokens from the input image and then grouping a subset of these tokens into some salient structures. The grouping process is usually designed to minimize a predefined *grouping cost (function)* that negatively measures the perceptual saliency of the resulting structure based on psychological vision rules, such as the Gestalt laws [1]. As an important step in mid-level computer vision, grouping can provide useful input to many high-level computer-vision applications such as object recognition or content-based image retrieval.

The challenge in grouping comes from both the definition of the grouping cost and the development of an algorithm for finding the optimal grouping with the minimum grouping cost. In this paper, we develop a new grouping method within the *edge grouping* framework, where the tokens are a set of line segments detected from the input image. The goal is then to identify a subset of these line segments and group them into complete boundaries of perceptually salient structures. Being able to more conveniently encode the well known Gestalt laws [1], edge grouping has been studied for decades with a long line of available edge-grouping methods, e.g., [2]–[14].

Most of the previous edge-grouping methods only consider boundary information in their grouping cost. For example, in almost all the previous edge-grouping methods, boundary *proximity* is always considered to make the resulting boundary contain gaps as short as possible when connecting the line segments into a boundary. However, many constructed line segments in fact come from image noise (or image texture) and, therefore, considering only boundary information usually makes the grouping very sensitive to image noise. Some boundary properties such as *continuity*, which requires the resulting boundary to be smooth, and *convexity*, which requires the resulting boundary to be convex, may partially solve this problem by only detecting smooth and convex structures. However, the incorporation of these properties may limit the applicability of the grouping methods since many salient structures in real applications are not always smooth or convex.

To address this problem, in this paper, we develop a new edge-grouping method that combines boundary and region information. In its baseline form, it combines two boundary properties of proximity and *closure* and one region property of the enclosed region area. Specifically, the grouping cost for a resulting boundary is defined as the ratio between the total gap length along the boundary and the region area enclosed by the boundary. The closure is set as a hard constraint by requiring the detected boundary to be closed. This way, we introduce a preference to detect larger-size structures and, therefore, make the grouping more robust to image noise. From this baseline method, we also discuss extensions for finding a better balance between the boundary proximity and region area, and incorporating other types of boundary and region information that may be desirable in certain applications.

To locate the closed boundary that minimizes this ratio-form grouping cost, we first develop a new graph model where line segments and in between gaps are modelled by two different kinds of edges. We represent each line segment or in between gap by two edges, so that the boundary and region information can be encoded into two edge-weight functions, respectively. Based on this graph model, we reduce the edge-grouping problem to finding a special kind of cycle with a minimum

Manuscript received August 14, 2006; revised June 8, 2007. This work was supported in part by the National Science Foundation under Grant NSF-EIA-0312861 and in part by the Air Force Office of Scientific Research under Grant FA9550-07-1-0250. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gaudenz Danuser.

The authors are with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208 USA (e-mail: stahlj@engr.sc.edu, songwang@engr.sc.edu).

Digital Object Identifier 10.1109/TIP.2007.904463

ratio-form cost. This cycle-finding problem can be solved in polynomial time by a globally optimal graph algorithm. We implement this edge-grouping method and test it on both synthetic data and real images, and compare its performance against several available edge-grouping and edge-linking methods.

The remainder of this paper is organized as follows. Section II briefly discusses the major related work. Section III formulates the problem by introducing a new grouping cost that combines boundary and region information. Section IV presents the details of the graph model that is used for solving the formulated edge-grouping problem. Section V presents the experimental results on synthetic data and real images. Section VI discusses several extensions to the proposed method. Section VII is the conclusion.

II. Related Work

As detailed later in Section III, from a set of line (curve) segments detected from an image, *edge grouping* aims to identify the ones along the desired salient structure boundaries and connect them to completely recover such boundaries. Therefore, the output of edge grouping usually consists of one or several complete boundaries. There has been a long line of research on edge grouping with many grouping costs and grouping algorithms developed in past decades [2]–[14]. However, most of the available edge-grouping methods only consider the boundary information, such as the Gestalt laws of closure, proximity and continuity.

In [11], Shashua and Ullman use a local parallel network to model the line segments and define a grouping cost by combining the boundary proximity and continuity. An iterative update algorithm is developed to search for an optimal boundary that may not be closed. Alter and Basri [2] further conduct an extensive analysis of this parallel-network method and point out the problems when applying this method iteratively to detect the second most salient boundary and the problems due to discretization. Recent work on edge grouping includes Elder and Zucker [5], Williams and Thornber [14], and the ratio-contour method [13] where boundary closure, continuity and proximity are combined in the grouping cost. Among these three methods, the ratio-contour method has been shown to have a better performance by being more robust to image noise. Our experiments in Section V further show that the method developed in this paper usually performs better than the ratio-contour method.

In [12], we develop a convex edge-grouping method that combines both boundary and region information. The grouping cost is also defined in a ratio form with a combined measure of proximity and region intensity homogeneity in the numerator and the region area as the denominator. However, the graph modeling and algorithm used in [12] can only detect convex boundaries. Without the convex constraint, we believe it is an NP-hard problem to minimize the grouping cost in [12]. In [15], we pair up line segments into a new grouping token in order to detect symmetric boundaries by combining boundary and region information. However, the assumption of symmetry does not hold for all structures in the real world. Note that our previous work on ratio contour, convex grouping and symmetric grouping also use a ratio-form grouping cost and finally reduces the problem to the graph problem of finding the minimum ratio alternate cycle, which is similar to the one derived in this paper, as shown at the end of Section IV. However, both the grouping cost and graph-modeling process developed in this paper are different from the ones developed in these previous work since the grouping tasks are different.

Closely related to edge grouping is *edge linking*, a category of methods that also connect the detected line/curve segments (or the detected edges from an edge detector) [16]–[21]. However, different from edge grouping, edge linking aims at enhancing the edge-detection result by connecting short edges to form longer edges. This is different from edge grouping investigated in this paper, which aims to detect one or several salient closed boundaries. Some edge-linking methods [18], [21] connect all the detected edges to segment the image into many regions. However, further postprocessing, such as region splitting/merging, is usually needed to identify a salient structure from these regions because many noise and texture segments are also linked. In fact, the edge-linking methods proposed in [18], [21] are mainly applied to segment range images without heavy noise and texture. Edge grouping and edge linking may have different applications. In Section V-B, we will further show the differences between edge grouping and edge linking by showing comparison experiments on several real images. Note that, in some literature [18], edge linking is called edge grouping. In this paper, we distinguish them to clarify the contribution of the proposed method.

While combining boundary and region information has not been extensively explored in previous edge-grouping methods, except for our recent work on convex and symmetric edge grouping [12], [15], [22], it has been investigated in several *pixel-grouping* methods. Different from edge grouping, pixel grouping treats each image pixel as a token and the goal is usually to find a boundary that partitions the image into foreground and background regions or two subregions without labelling the foreground and background. Comparing edge grouping with pixel grouping, there are both advantages and disadvantages. For example, edge grouping simplifies the quantization of many important boundary properties, such as continuity, convexity, and symmetry, which may be difficult to consider in pixel grouping. In addition, the number of line segments detected in an image is usually much smaller than the number of pixels in an image. This may make the edge grouping to take less CPU time when conducting globally optimal grouping. However, edge grouping may fail when the line segments can not be well detected from the input image.

In [23], Cox *et al.* present a ratio-region method to detect a closed boundary that partitions an image into foreground and background regions. The grouping cost is of a ratio form, with the numerator measuring the boundary property and the denominator measuring the foreground region area. It uses an optimization algorithm of repeating the minimum-cut max-flow algorithm to find the graph partitioning that minimizes the grouping cost. In [24], Jermyn and Ishikawa further extend the ratio-region method so that different boundary and region properties can be considered in the grouping cost. In this method, the problem is modelled in a directed graph and finally reduced to a graph problem of finding the minimum ratio cycle in the constructed directed graph. This graph problem

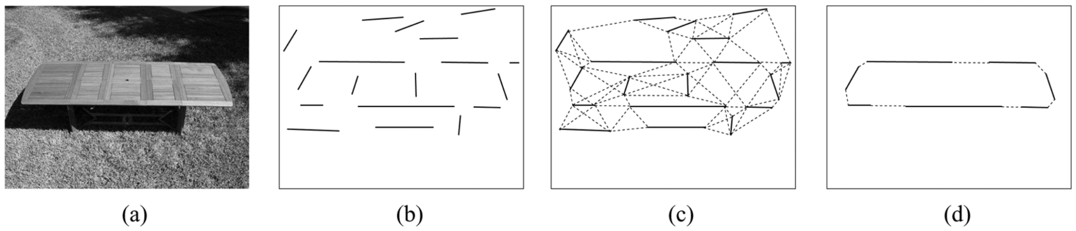


Fig. 1. Typical steps in applying an edge-grouping method to a real image to detect salient boundaries. (a) Input image, (b) the detected segments, (c) constructing gap-filling segments (dashed lines), and (d) the detected closed boundary that traverses some detected and gap-filling segments alternately.

shares some similarity with the one formulated in this paper, but differs in a) its graph is *directed* and the graph in this paper is an *undirected solid-dashed* graph, and b) it searches over all simple cycles while we only consider *alternate* cycles. In addition, some extensions, such as the proximity exponentiation introduced in Section VI-C, cannot be applied in pixel grouping.

In [25], Shi and Malik develop a normalized-cut method that seeks to partition an image into two balanced regions, without the labelling of the foreground and background. Given the NP-completeness of this problem, spectral graph theory is applied to achieve an approximate solution. In [26], Sumengen and Manjunath present a graph-partitioning active contour approach to iteratively search for a locally optimal boundary that minimizes the grouping cost used in normalized cut. In [27], Chan and Vese define a grouping cost that requires the resulting boundary to be both smooth and enclose a region with homogeneous intensity. In [28], Wang and Oliensis define a comprehensive grouping cost that measures the complexity of both foreground and background regions. These complicated grouping costs, however, usually lead to NP-hard or even nonpolynomial-time problems and only local optimal solutions can be found by gradient-descending approaches, such as active contours. By using α-partitioning and region filtering, Liou*et al.* [29] develop a parallel technique to group pixels into regions with certain intensity variations.

III. Problem Formulation

As illustrated in Fig. 1, applying an edge-grouping method to a real image to detect perceptually salient structure boundaries usually involves three steps. The first step is to construct a set of line segments by a) running an edge detector, such as the Canny detector [30], on the input image to detect a set of edges, and b) approximating the detected edges with a set of straight line segments, as shown in Fig. 1(b). These straight line segments are usually referred to as *detected (line) segments*. Since these detected segments are disconnected from each other, in order to construct a closed boundary we need to fill the gaps between them. So, in the second step, we fill the gaps between the detected segments by connecting all possible pairs of endpoints of different detected segments. These connections are referred to as *gap-filling (line) segments*, as denoted by dashed lines in Fig. 1(c).[1] This way, a boundary is defined as a cycle that traverses a set of detected and gap-filling segments *alternately*. The third step is to develop an algorithm to find such a boundary that minimizes a selected grouping cost, as shown in Fig. 1(d).

[1]Note that not all possible gap-filling segments are shown in Fig. 1(c), in order to keep it readable.

To combine the boundary and region information, in this paper, we introduce a ratio-form grouping cost for a closed boundary $\mathcal{B}$ that traverses some detected and gap-filling segments alternately as

$$\phi(\mathcal{B}) = \frac{|\mathcal{B}_G|}{\int\int_{R(\mathcal{B})} dxdy} \tag{1}$$

where the numerator $|\mathcal{B}_G|$ is the total length of the gap-filling segments along the boundary $\mathcal{B}$ and the length of a detected/gap-filling segment is measured by the 2-D Euclidean distance between its two endpoints. This numerator reflects the proximity of the boundary: the smaller the total gap length $|\mathcal{B}_G|$, the higher the proximity and the saliency of the boundary $\mathcal{B}$. $R(\mathcal{B})$ is the region enclosed by the boundary $\mathcal{B}$ and the denominator $\int\int_{R(\mathcal{B})} dxdy$ is the region area, which sets a preference to detect larger structures. Such a preference makes the grouping more robust against image noise. In Section IV, we develop a graph model to find a closed boundary that minimizes the grouping cost (1).

IV. Graph Modeling and Algorithm

We begin by constructing a graph $G = (V, E)$, with a set of vertices $V = \{u_1, u_2, \ldots, u_n\}$ and a set of edges $E = \{e_1, e_2, \ldots, e_m\}$. Particularly, we construct a pair of edges e^+ and e^- for each line segment. We call the constructed pair of edges to be *solid* edges, if the corresponding line segment is a detected one, and *dashed* edges, if the corresponding line segment is a gap-filling one. This way, we actually construct two vertices, $u_i^{(1)}$ and $u_i^{(2)}$, for each line-segment endpoint. Fig. 2 shows an example, where for the detected line segment P_1P_2 shown in Fig. 2(a), we construct two solid edges, e_{12}^+ and e_{12}^-, shown by solid lines in Fig. 2(b). For the gap-filling segment P_2P_3, in Fig. 2(a), we construct two dashed edges, e_{23}^+ and e_{23}^-, shown by dashed lines in Fig. 2(b), and for each line-segment endpoint P_i, $i = 1, 2, 3$, we construct two vertices, $u_i^{(1)}$ and $u_i^{(2)}$, $i = 1, 2, 3$. We will show that this construction of edges in pairs facilitates the quantization of the region area enclosed by the boundary $\mathcal{B}$.

One problem in this graph construction is to determine the edge connection relations, since each line segment is represented by a pair of edges. For example, in Fig. 2(a) and (c), the detected segment P_1P_2 is connected to the gap-filling segment P_2P_3 at P_2. In the constructed graph we need to decide whether we are going to link e_{12}^+ to e_{23}^+ and e_{12}^- to e_{23}^-, or link e_{12}^+ to e_{23}^-

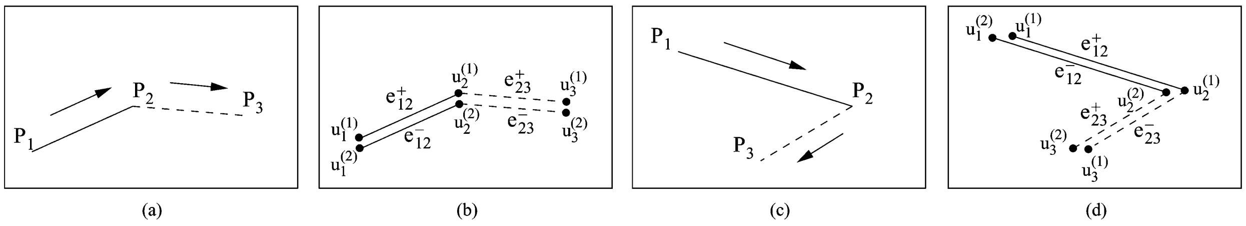


Fig. 2. Illustration of the graph construction. Edges corresponding to two segments with (a), (b) the same direction; (c), (d) opposite direction.

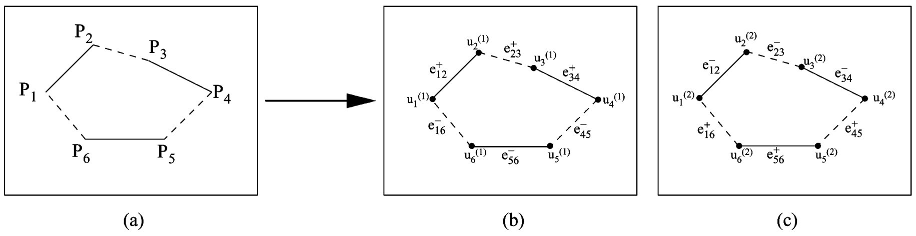


Fig. 3. (a) Boundary with three detected segments and three gap-filling segments. (b), (c) Two "mirror" cycles that correspond to the boundary shown in (a).

and e_{12}^- to e_{23}^+. In this paper, we solve this problem by implicitly associating each of the two edges in the graph, corresponding to the same line segment, with a different direction. Particularly, e^+ indicates that the direction along the corresponding line segment is from the left endpoint to the right endpoint (LR), and e^- indicates that the direction along the corresponding line segment is from the right endpoint to the left endpoint (RL). For any line segment, the left endpoint is the one with the smaller x-coordinate and the right endpoint is the one with the larger x-coordinate. For example, for the line segment P_1P_2 in Fig. 2(a) and (c), P_1 is the left endpoint and P_2 is the right endpoint. This way, we can uniquely determine the edge-connection relation by requiring consistency in direction between the two neighboring line segments. Fig. 2(b) and (d) show the edge connections constructed from the line segments P_1P_2 and P_2P_3 shown in Fig. 2(a) and (c), respectively. If the x-coordinates of the two endpoints are equal, we can decide by the y-coordinates of the two endpoints in a similar fashion. In this case, we define the point with the smaller y-coordinate as the "left" endpoint, and the one with the larger y-coordinate as the "right" endpoint. Note that the constructed graph is still an *undirected* one and we only use this direction information to define the edge-weight functions as discussed later.

In this constructed graph G, a closed boundary $\mathcal{B}$ that traverses some detected and gap-filling segments alternately is modeled by two cycles that traverse the corresponding solid and dashed edges alternately. An example is shown in Fig. 3, where the boundary $P_1P_2 \ldots P_6$ is modeled by the two cycles shown in Fig. 3(b) and (c). We can see that these two cycles are the "mirrors" of each other, i.e., for a pair of edges e^+ and e^- constructed for the same line segment, if one of them is contained in one cycle, the other must be contained in the other cycle. For convenience, we call the graph G to be a *solid-dashed (SD)* graph, because no two solid edges are neighboring to each other, and we refer to a cycle that traverses solid and dashed edges alternately as an *alternate* cycle. This way, the problem of finding the boundary $\mathcal{B}$ that minimizes the grouping cost $\phi(\mathcal{B})$ given in (1) can be reduced to the problem of finding an optimal alternate cycle $\mathcal{C}$ in the constructed SD graph G if we can quantify the grouping cost $\phi(\mathcal{B})$ by some edge weights in G.

We define two edge-weight functions, the *first (edge) weight* $w_1(e)$ and the *second (edge) weight* $w_2(e)$, for each edge $e \in E$. Given any line segment P_1P_2, we set the first weight for the corresponding two edges (see the equation at the bottom of the page), where $|P_1P_2|$ is the length of the line segment P_1P_2 (the 2-D Euclidean distance between P_1 and P_2). For both solid and dashed edges, their second weights are defined as a signed area associated to the corresponding line segment. Specifically, as shown in Fig. 4(a), let the bottom-left pixel in the input image be the origin, the horizontal direction be the direction of the x-axis, and the vertical direction be direction of the y-axis. The area associated to a line segment P_1P_2 is defined as the area of the region bounded by this line segment and its projection in the x-axis. The sign of this area is defined to be positive for the edge corresponding to a line segment that bears a LR direction and negative otherwise. For the example shown in Fig. 4(a), we have $w_2\left(e_{12}^+\right) = -w_2\left(e_{12}^-\right) = \texttt{area}(P_1P_2P_2^xP_1^x) > 0$, where P_1^x and P_2^x are the projections of P_1 and P_2 onto the x-axis. This definition allows us to calculate the total area within a boundary

$$w_1(e_{12}^+) = w_1(e_{12}^-) = \begin{cases} 0, & \text{if } P_1P_2 \text{ is a detected segment} \\ |P_1P_2|, & \text{if } P_1P_2 \text{ is a gap-filling segment} \end{cases}$$

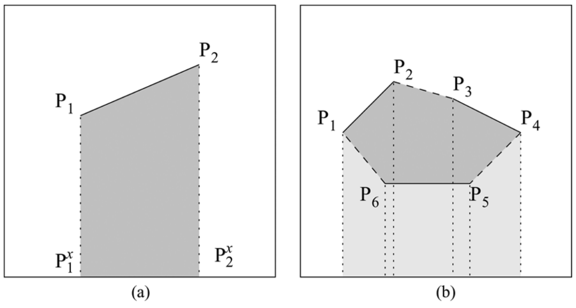


Fig. 4. Illustration of defining the second weight for an edge. (a) Area associated to a line segment. (b) The region area enclosed by a closed boundary is equal to the sum of the signed areas associated to the line segments along this boundary.

by simply summing up the signed areas associated to each of its line segments. An example is shown in Fig. 4(b), where the area of the polygon $P_1 \ldots P_6$ is equal to the summation of positive areas associated to P_1P_2, P_2P_3, P_3P_4, and negative areas associated to P_4P_5, P_5P_6, and P_6P_1.

As discussed above, a closed boundary $\mathcal{B}$ corresponds to two alternate "mirror" cycles $\mathcal{C}^+$ and $\mathcal{C}^-$ in G, as shown in Fig. 3(b) and (c). Since the edges in $\mathcal{C}^+$ and $\mathcal{C}^-$ are constructed in pairs, we have

$$W_1(\mathcal{C}^+) = \sum_{e\in\mathcal{C}^+} w_1(e) = W_1(\mathcal{C}^-) = \sum_{e\in\mathcal{C}^-} w_1(e) > 0$$
$$\text{and } W_2(\mathcal{C}^+) = \sum_{e\in\mathcal{C}^+} w_2(e) = -W_2(\mathcal{C}^-) = -\sum_{e\in\mathcal{C}^-} w_2(e).$$

Without loss of generality, let the cycle $\mathcal{C}^+$ be the one with the positive total second weight, i.e., $W_2(\mathcal{C}^+) = -W_2(\mathcal{C}^-) > 0$. It is easy to verify that $W_1(\mathcal{C}^+)$ is equal to the numerator of $\phi(\mathcal{B})$ and $W_2(\mathcal{C}^+)$ is equal to the total area of the enclosed region, i.e., the denominator of $\phi(\mathcal{B})$. Since for every cycle $\mathcal{C}^+$ there exists a "mirror" cycle $\mathcal{C}^-$ in G, it is easy to see that the cycle $\mathcal{C}$ that minimizes

$$\varphi(\mathcal{C}) = \frac{W_2(\mathcal{C})}{W_1(\mathcal{C})} \tag{2}$$

is a $\mathcal{C}^-$ version (i.e., $W_2(\mathcal{C}) < 0$) that corresponds to the boundary $\mathcal{B}$ that minimizes $\phi(\mathcal{B})$, i.e., $\phi(\mathcal{B}) = -1/\varphi(\mathcal{C})$. This way, we only need to find an alternate $\mathcal{C} \in G$ that minimizes the cycle ratio $\varphi(\mathcal{C})$. This problem can be solved in polynomial time by the minimum-ratio-alternate cycle algorithm presented in [13].

Particularly, the minimum-ratio-alternate cycle algorithm finds the desired optimal cycle by three polynomial-time reductions [13]: a) redistribute the weights of the solid edges in the graph to the adjacent dashed edges, so that for each solid edge both of its weights are zero, b) reduce the problem of finding the minimum ratio cycle to the problem of detecting an alternate cycle with negative total weights, and c) reduce the problem of detecting a negative-weight alternate cycle to the problem of finding the minimum-weight perfect matching, which can be solved in polynomial time [31]. According to [13], the time complexity of the minimum-ratio-alternate cycle algorithm is $O\left(|V|^{3/4}|E|\right)$ with $|V|$ and $|E|$ being the number of vertices and edges in the graph, respectively.

V. Experiments

We implement the above graph model and algorithm in C++ and evaluate the proposed edge-grouping method on a set of synthetic data and real images.[2] The synthetic data was directly generated as a set of detected line segments. For the real images, we construct the detected segments by edge detection and line approximation. Particularly, we use the Canny edge detector from the Matlab image processing toolbox, and the line approximation package developed by Kovesi [32]. For the Matlab Canny edge detector, we use its default settings, and for the line approximation package, we set the minimum edge length to be processed to 30 pixels, and the maximum deviation between an edge and its fitted line segment to 2 pixels. All CPU times reported in this paper come from a 2.33-GHz 64-bit Xeon Linux workstation with 8 GB of memory.

A. Experiments on Synthetic Data

To evaluate the proposed edge-grouping method quantitatively, we construct a set of synthetic data with known desired salient structure boundaries, or *ground truth*. For constructing a synthetic data sample, we pick one of the ten polygonal boundaries shown in Fig. 5 as the ground truth and placed it inside a square region of size 128×128. Then we remove a certain percentage of segments along this ground-truth boundary at random locations to construct some gaps and the remaining segments are then included as detected line segments. The gap percentage along the ground-truth boundary (i.e., the percentage of the ground-truth boundary that is removed to construct gaps) is chosen from the set $\{0\%, 5\%, 10\%, 20\%, 30\%, 40\%, 50\%\}$. We then construct a set of additional detected line segments to simulate the image noise. Specifically, these noise segments are placed at random locations (inside the 128×128 square region), in random directions, and with a length selected randomly between 3 and 7 pixels (all properties uniformly distributed). The number of added noise segments is chosen from the set $\{0, 10, 20, 40, 80\}$. An example is shown in Fig. 6, where the ground truth is chosen to be the fourth boundary in Fig. 5. Fig. 6(c) shows a constructed synthetic data sample by removing 30% of the ground-truth boundary's perimeter and then adding 40 noise segments shown in Fig. 6(b). To remove p percent of the ground-truth boundary, we uniformly partition the boundary into 20 line segments, and then removing p percent of these line segments randomly.

We measure the accuracy of an edge-grouping result using Jaccard's similarity coefficient, $|R_D \cap R_G|/|R_D \cup R_G|$, where R_D and R_G are the region bounded by the optimal boundary detected by the proposed edge-grouping method and the one bounded by the ground-truth boundary, respectively, and $|R|$ is the area of R. Based on this accuracy measure, we also compare the performance of the proposed method against the ratio-contour method [13], an edge-grouping method that has been shown to perform more favorably than several other state-of-the-art

[2]The software developed in this work can be downloaded from http://www.cse.sc.edu/~ songwang/document/RRC.tgz.

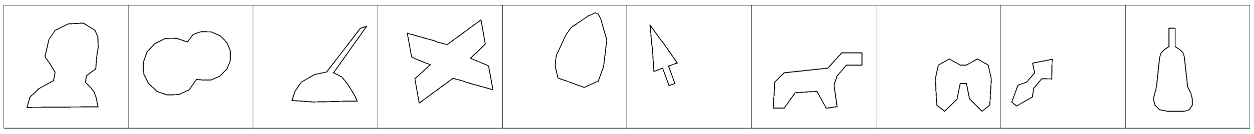


Fig. 5. Ten polygonal closed boundaries used for the construction of synthetic data.

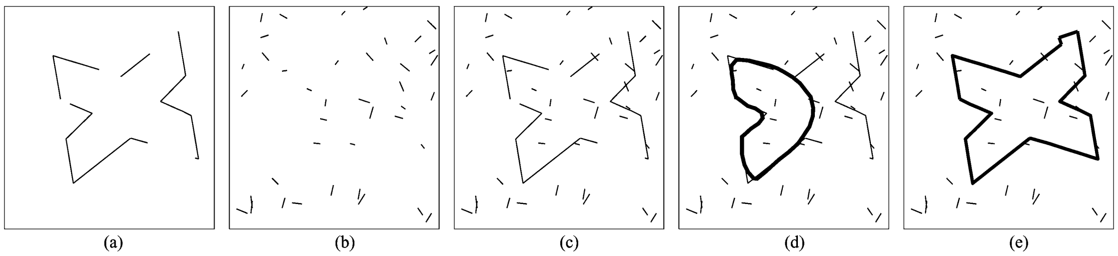


Fig. 6. Illustration of the synthetic data construction and the grouping results. (a) Detected segments constructed from the ground-truth boundary. (b) Additional noise segments. (c) A constructed synthetic data sample by combining the segments shown in (a) and (b). (d) Optimal boundary detected from (c) by applying the ratio-contour method developed in [13]. (e) Optimal boundary detected from (c) by applying the proposed edge-grouping method.

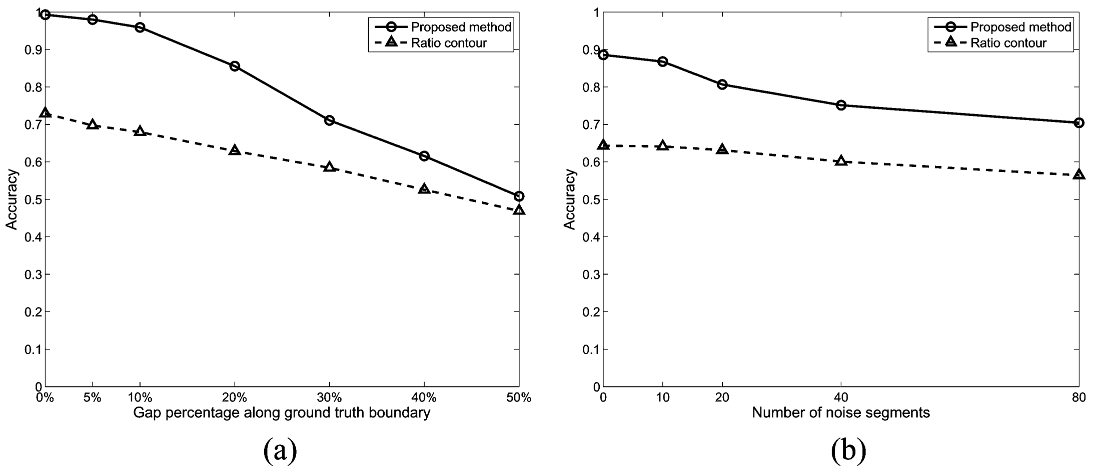


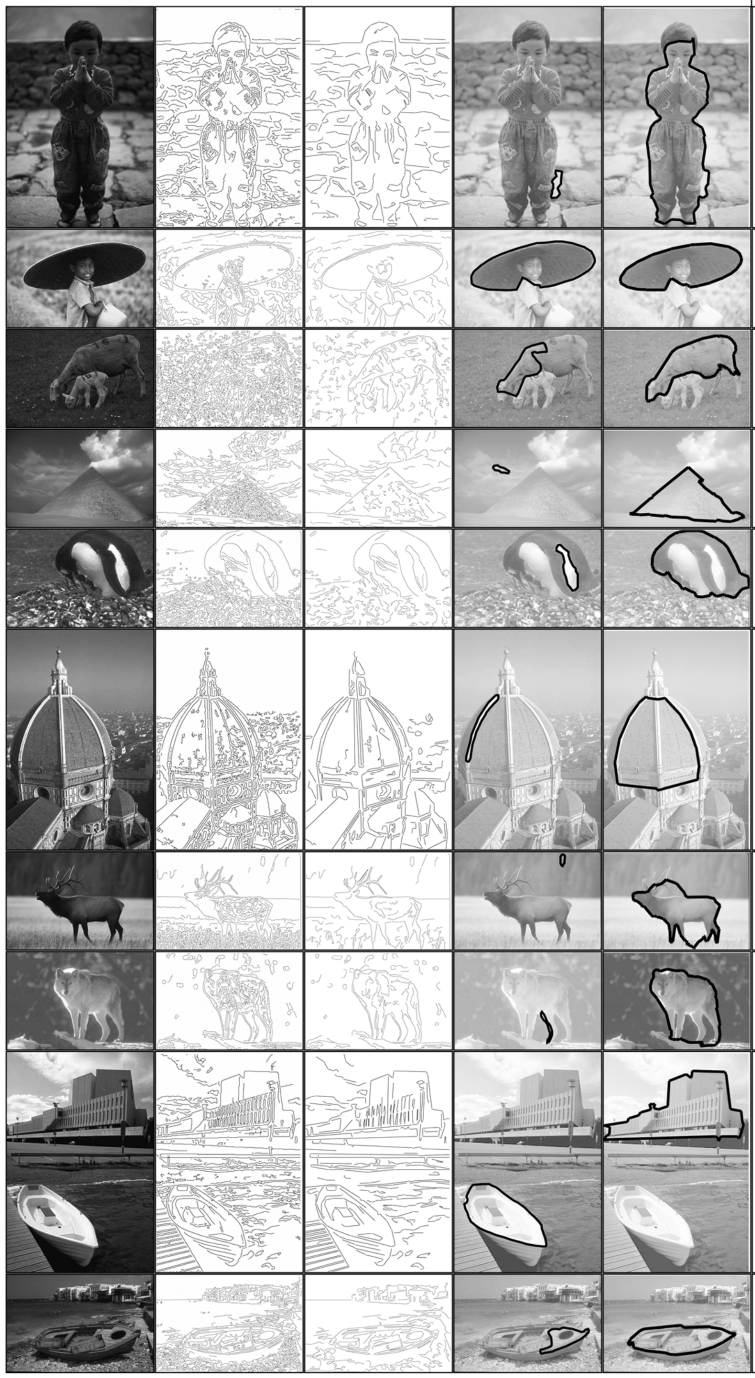
Fig. 7. Performance of the proposed method and the ratio-contour method on the 3 500 synthetic data samples.

edge-grouping methods. By considering only boundary properties of proximity and continuity, the grouping cost used in the ratio-contour method is

$$\phi_r(\mathcal{B}) = \frac{|\mathcal{B}_G| + \lambda \cdot \int_{\mathcal{B}} \kappa^2(t)dt}{\int_{\mathcal{B}} dt}$$

where $\kappa(t)$ is the curvature and t is the parameter of the arc-length parameterized boundary $\mathcal{B}$. This grouping cost is also of a ratio form. However, its denominator is the boundary perimeter, which helps improve the grouping robustness against image noise by avoiding producing overly short boundaries. Note that in this ratio-contour method, the gap-filling tokens are not constructed as straight line segments, but instead approximated as Bezier curves, which connect the detected segments with continuous tangent directions and, therefore, allow the measuring of curvature along the resulting boundary. In our experiments, we set the parameter λ at its default value of 10 [13]. Fig. 6(d) and (e) shows the grouping results of running the ratio-contour method and the proposed method on the segments shown in Fig. 6(c), respectively. These results represent a grouping accuracy of 0.51 and 0.99, respectively.

As mentioned above, we have ten different choices of the ground-truth boundaries, seven different choices of the gap percentage along the ground-truth boundary, and five different choices of the number of additional noise segments. For each possible combination of these choices, we also run the random sampling for noise segments ten times to achieve ten different sets of noise segments. Therefore, in total, we construct $10 \times 10 \times 7 \times 5 = 3500$ synthetic data samples. We run the edge-grouping methods on each of them and evaluate the grouping accuracy by comparing the detected optimal boundary with the ground truth. Fig. 7 shows the performance curves of the proposed edge-grouping method and the ratio-contour method developed in [13]. The performance in Fig. 7(a) is shown in terms of the gap percentage along the ground-truth boundary and each point in this figure indicates the average accuracy over $10 \times 10 \times 5 = 500$ data samples with the same gap percentage along the ground-truth boundary. The performance in Fig. 7(b) is shown in terms of the number of the additional noise segments and each point in this figure indicates the average accuracy over $10 \times 10 \times 7 = 700$ data samples with the same number of additional noise segments. These curves



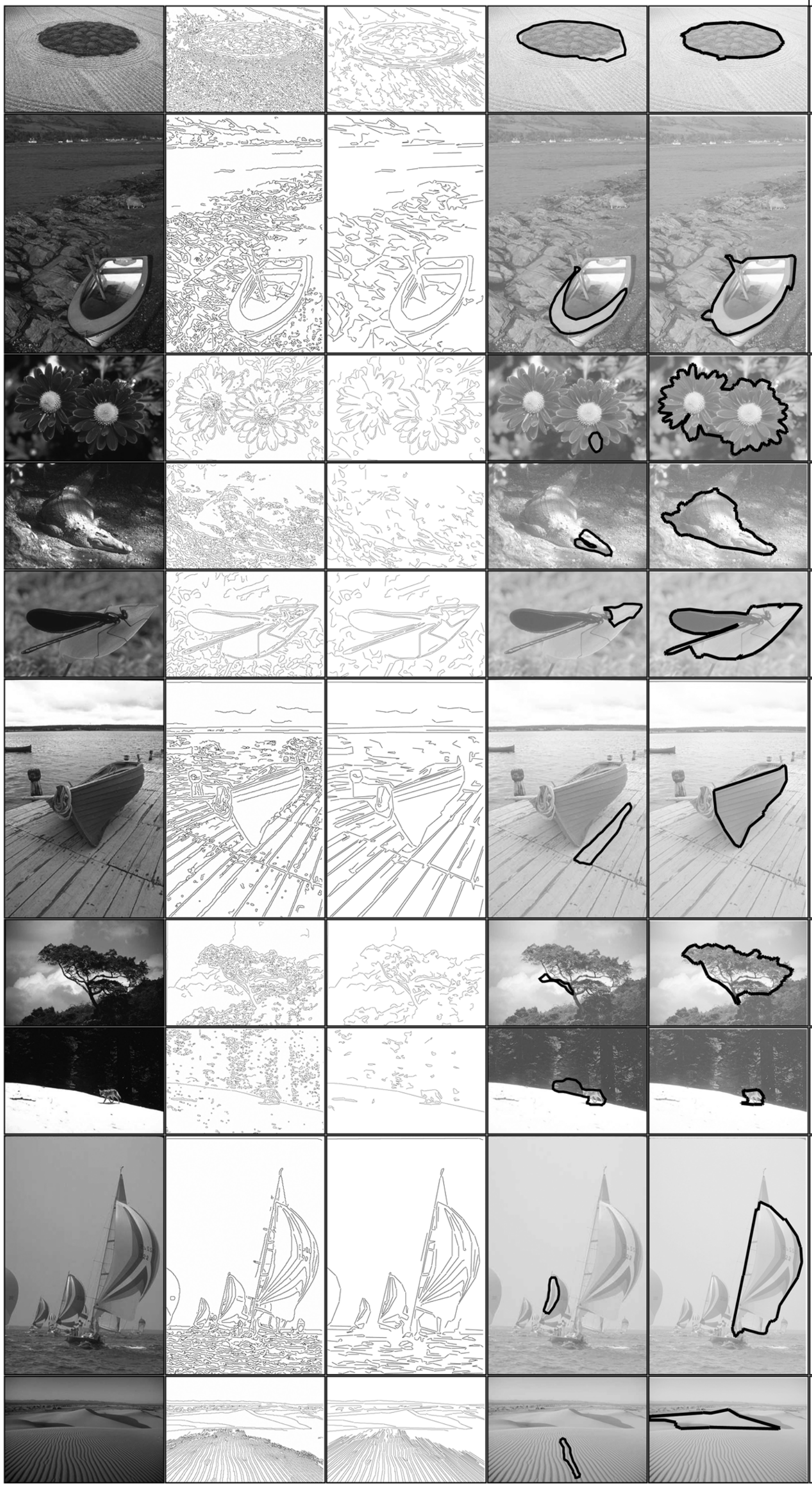
	Number of Segments	CPU time Ratio Contour	CPU time Prop. Method
(a)	562	772.24s	27.45s
(b)	382	589.07s	2.14s
(c)	784	1070.80s	79.58s
(d)	525	594.96s	14.06s
(e)	596	616.58s	24.13s
(f)	482	830.69s	14.13s
(g)	451	469.23s	5.82s
(h)	406	440.50s	7.68s
(i)	502	1187.60s	15.44s
(j)	656	902.93s	37.02s

Fig. 8. Edge-grouping results in ten real images. (From left to right) Column 1: the input image; column 2: the Canny detection result; column 3: the detected segments resulting from line approximation; column 4: the optimal boundary detected by the ratio-contour method; column 5: the optimal boundary detected by the proposed method.; column 6: the number of detected segments in column 3; column 7: the CPU time (in seconds) taken by the ratio-contour method; column 8: the CPU time (in seconds) taken by the proposed method.

clearly show that the inclusion of the region-area information in the proposed method leads to better grouping accuracy than the ratio-contour method. The main reason is that the ratio-contour method explicitly incorporates the continuity property to improve its robustness against noise, but many boundaries in the real world are not necessarily smooth everywhere. For example, as shown in Fig. 6(d), the continuity preference may prevent the ratio-contour method from correctly detecting the desired ground-truth boundary completely.

B. Experiments on Real Images

We also test the proposed edge-grouping method on a set of real images selected from the Berkeley segmentation dataset [33]. All real images have a size of either 481×321 or



	Number of Segments	CPU time Ratio Contour	CPU time Prop. Method
(a)	1208	1233.39s	297.43s
(b)	679	1014.90s	48.86s
(c)	594	707.24s	24.08s
(d)	654	679.80s	34.34s
(e)	390	612.76s	5.29s
(f)	413	979.02s	17.14s
(g)	557	463.23s	59.83s
(h)	235	261.47s	1.97s
(i)	390	545.78s	22.30s
(j)	457	1451.54s	35.02s

Fig. 9. Edge-grouping results on another ten real images. Each column depicts the same information as in Fig. 8.

321×481. In order to reduce the number of the constructed gap-filling segments, which can run in the order of n^2 where n is the number of detected segments, we do not construct gap-filling segments that are highly unlikely to belong to the optimal boundary. Specifically, in our experiments, we do not construct a gap-filling segment between the two segment endpoints if the distance between these two endpoints is larger than 50 pixels.

The results on 20 test images are shown in Figs. 8 and 9. We can see that, for most of them, the proposed method detects the salient boundaries better than the ratio-contour method, e.g., Figs. 8(a), (c)–(h), and (j) and 9(b)–(j). There are also several

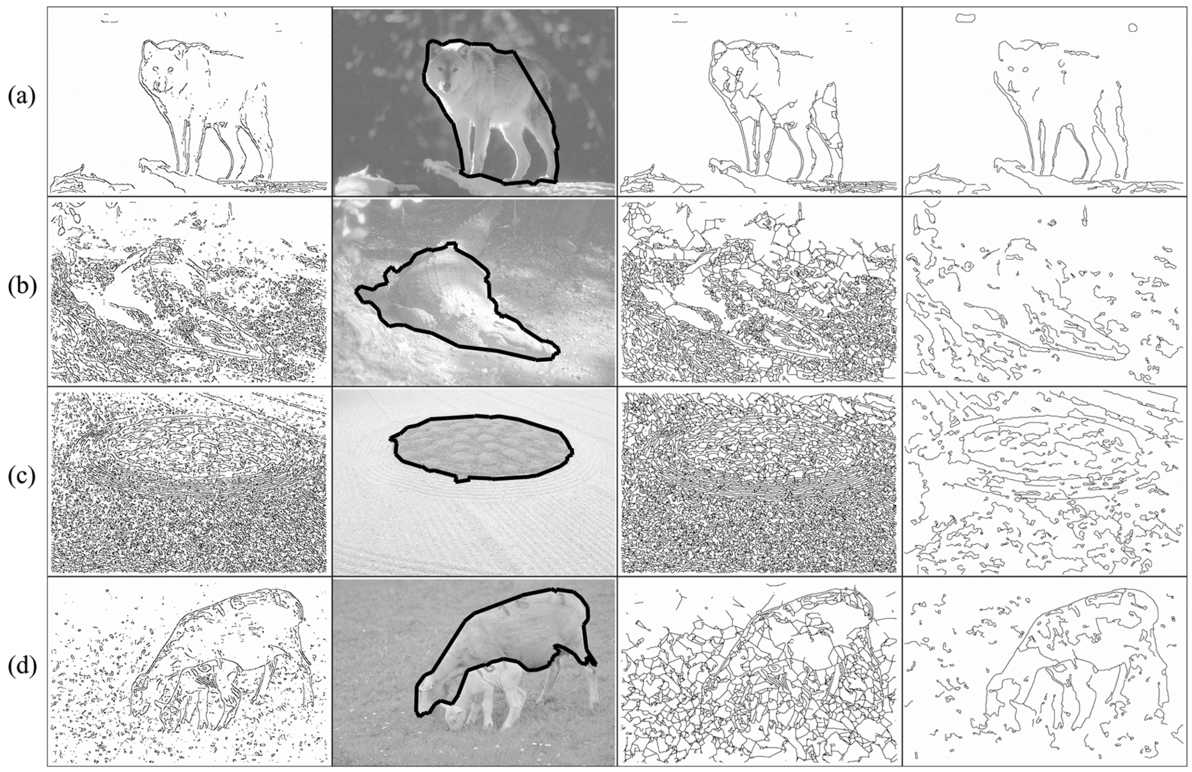


Fig. 10. Illustration of the differences between edge grouping and edge linking. (a)–(d) Experimental results on the real images shown in Fig. 8(h) and (c) and Fig. 9(a) and (d), respectively. Column 1: Canny detection result. Column 2: Optimal boundaries detected by the proposed method. Column 3: Edge-linking result from the Ghita–Whelan method. Column 4: Edge-linking result from the Farag–Delp method.

cases where both methods produce similar results, as shown in Fig. 8(b), and cases where two methods produce different yet both acceptable results, as shown in Fig. 8(i). Note that, the incorporation of the region-area term in the proposed method does not mean that the proposed method always produces a boundary that encloses a larger area than the one produced by the ratio-contour method. For example, Fig. 9(h) shows a case where the proposed method detects a smaller-size structure than the ratio-contour method. In this case, the proximity term may play a more dominating role than the region-area term, yet it may not dominate the continuity term used in the ratio-contour method.

C. Comparisons to Two Edge-Linking Methods

In this section, we conduct experiments to illustrate the differences between edge grouping and edge linking. We compare against the Farag–Delp method [16], based on sequential search techniques, and the Ghita–Whelan method [20], based on a fast search algorithm by considering only local information around the segment endpoints. Both of them are typical edge-linking methods with source codes publicly available. For the Ghita–Whelan method we used the implementation available as a component of the NeatVision package (http://www.neatvision.com), which was created by the authors. Specifically, we leave the Canny edge detector at its default values and the window size is set at 11×11. For the Farag–Delp method, we use the original M-SEL implementation provided by the authors with the default settings. Fig. 10 shows the results of these two edge-linking methods and the proposed edge-grouping method on four real images. Note that, in this experiment the proposed edge-grouping method is operated on the same edge-detection results as used in the Ghita–Whelan method for a fair comparison. These results clearly illustrate the differences between edge linking and edge grouping: edge linking aims at enhancing the edge detection by connecting the edges to form longer edges while edge grouping aims at detecting a salient closed boundary. Note that, by integrating both boundary and region information, the proposed edge-grouping method may detect a boundary that is not fully detected by the edge-linking methods. For example, in Fig. 10(a), the top-right gap on the back of the wolf is filled by the proposed method but is not filled by either of the edge-linking methods. Similar results can be observed in Fig. 10(b) and (d).

D. Multiple Boundary Detection

So far, we present the proposed method in the context of only detecting one boundary that minimizes the grouping cost (1). In practice, there may be more than one salient structures in an image and we can repeat the proposed method to detect multiple boundaries. The basic principle has been widely used in many previous edge-grouping methods, including the ratio-contour method. Given an image, e.g., the one shown in Fig. 11(a), we first process it with the proposed method to detect the optimal boundary as introduced above. Then, we remove from the graph G all the edges associated to line segments that belong to the detected optimal boundary, i.e., for each line segment P_iP_j in the boundary, we remove both edges e_{ij}^+ and e_{ij}^- from G. We then run the minimum-ratio-alternate-cycle algorithm again on

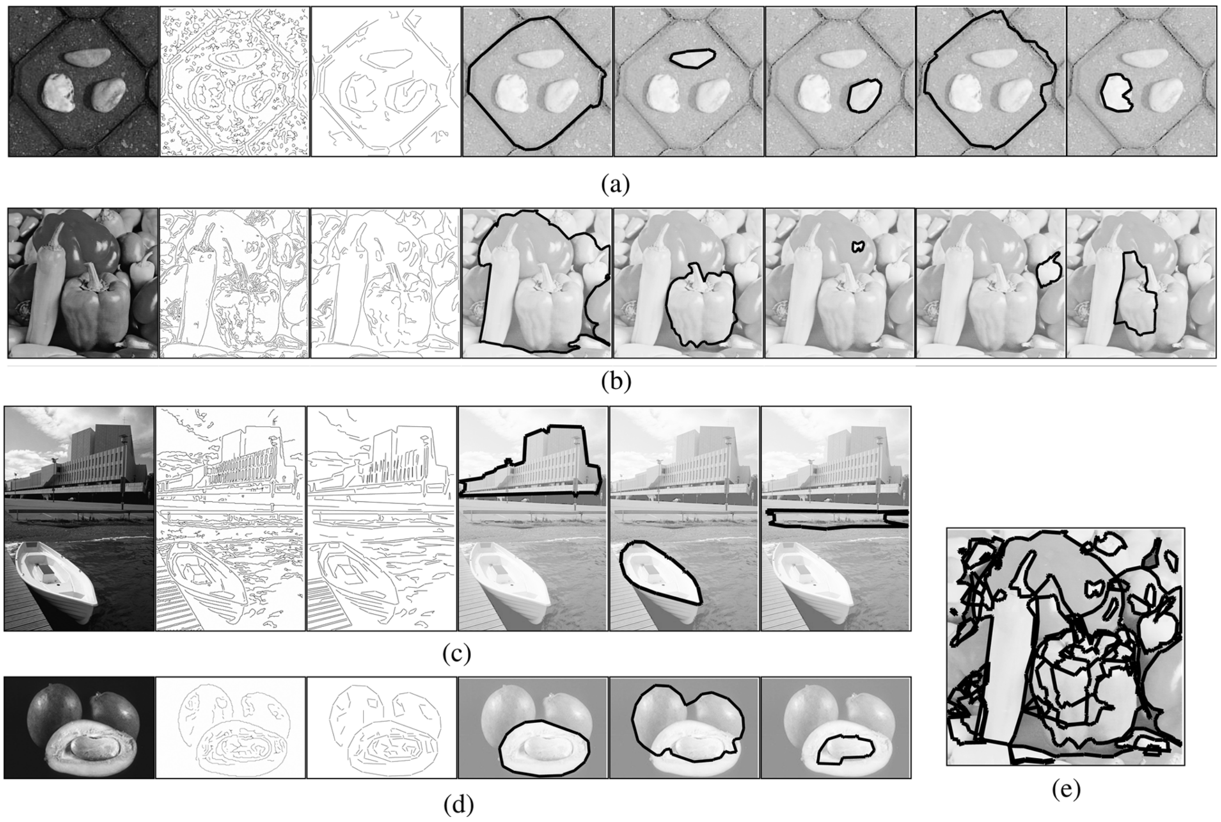


Fig. 11. Examples of detecting multiple boundaries from real images by repeating the proposed edge-grouping method. From (a)–(d), each row shows, from left to right, the input image, the Canny detection result, the detected segments, and the optimal boundaries detected in the first several iterations. (e) All 39 boundaries iteratively detected from the "peppers" image shown in (b). The CPU times for obtaining the results in (a)–(e) are 3.23, 37.57, 59.82, 1.86, and 64.39 s, respectively.

G, to detect the second optimal closed boundary. This process can be repeated k times to detect the kth optimal boundary. It is easy to show that the grouping cost for the detected boundaries increases monotonically in this iteration process.

Fig. 11 shows the results of running the proposed method multiple times on several real images. We can see that the proposed method identifies different boundaries in different iterations. The first and fourth boundaries in Fig. 11(a) are very similar and they bound the same structure of the floor tile. This comes from the fact that two very close parallel edges are detected along each side of this floor tile, as shown in the second and third columns in Fig. 11(a). Therefore, the detected segments along the first boundary are in fact different from the ones along the fourth boundary. In practice, we can compare the optimal boundaries detected in different iterations and remove the redundant ones.

Since the proposed method detects only closed boundaries, the repeating of the proposed method may not handle well the cases where two salient structures share part of the boundaries. Fig. 11(d) clearly illustrates this problem: the detection of the bottom fruit prevents the detection of the top two fruits because part of the boundaries of the top two fruits are removed after the detection of the bottom fruit. Similar results are shown in Fig. 11(b). Therefore, the proposed method is mainly suitable for detecting disjoint salient structures. Note that, since the edge-grouping method is designed to detect one or a small number of salient structure boundaries from an image, a good edge-grouping method should be able to detect the most salient boundaries from an image in the first several iterations by filtering out the undesired noise and texture segments. It may not be suitable to repeat the proposed method a large number of iterations trying to connect all the detected segments, which include texture and noise segments, as shown in Fig. 11(e). This is different from the edge-linking methods [18], [21].

VI. Extensions

In this section, we introduce four extensions to the proposed method. The first two can be useful to exploit possible prior knowledge about the desired salient structures, by adding the properties of continuity and intensity homogeneity to the grouping cost. The third extension seeks to adjust the balance between proximity and region-area terms in the grouping cost, since we find that, in certain cases, the region-area term might have undesirable dominance in grouping. The fourth extension is to ensure that detected salient boundaries are always simple without containing any self intersections.

A. Adding Continuity

In the previous section, we show that, in general, the use of the region-area property in the proposed method leads to more favorable grouping than the use of continuity alone in the ratio-contour method. However, there are certain cases where

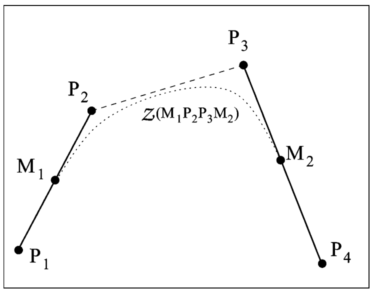


Fig. 12. Illustration of using Bezier curve to approximate the gap-filling and detected segments.

the desired salient structure in an image is *a priori* known to be smooth. To consider continuity, we modify the grouping cost to

$$\phi_c(\mathcal{B}) = \frac{|\mathcal{B}_G| + \lambda \cdot \int_{\mathcal{B}} \kappa^2(t)dt}{\int\int_{R(\mathcal{B})} dxdy} \tag{3}$$

where $\kappa^2(t)$ is the squared curvature along the arc-length parameterized boundary $\mathcal{B}$, and as in the ratio-contour method [13], λ is a regularization factor that balances the proximity and continuity. In our experiments, we consistently set λ to be 10. The additional curvature term in this extension makes the resulting edge grouping biased to detect smoother boundaries.

However, it is difficult to directly measure the boundary curvature in our formulation since the boundary $\mathcal{B}$ is a polygon consisting of a set of straight line segments. We address this problem by interpolating the polygon by smooth cubic splines. Particularly, given a gap-filling segment P_2P_3 that connects detected segments P_1P_2 and P_3P_4, we measure its curvature κ over $\mathcal{Z}(M_1P_2P_3M_2)$, the Bezier curve with control points M_1, P_2, P_3 and M_2, as shown in Fig. 12, where M_1 and M_2 are the midpoints of P_1P_2 and P_3P_4, respectively. We can then calculate the curvature along this Bezier curve and use it to measure the continuity. In the graph modeling, we only need to modify the definition of the first edge weight to incorporate this curvature term. Specifically, for the dashed edges e_{23}^+ and e_{23}^- corresponding to the gap-filling segment P_2P_3 shown in Fig. 12, we define their first edge weight as

$$w_1\left(e_{23}^+\right) = w_1\left(e_{23}^-\right) = |P_2P_3| + \lambda \cdot \int_{\mathcal{Z}(M_1P_2P_3M_2)} \kappa^2(t)dt.$$

With this modification, the graph G remains essentially the same and we can still apply the same graph algorithm to detect the optimal boundary that minimizes this modified grouping cost (3).

Note that this type of Bezier-curve interpolation may not reflect the boundary continuity accurately when the angles $\angle P_1P_2P_3$ and $\angle P_2P_3P_4$ become too small in Fig. 12. However, when any one of these two angles becomes too small, this gap-filling segment P_2P_3 is not likely to be included in a smooth boundary. Therefore, in practice, we do not construct a gap-filling segment P_2P_3 between detected segments P_1P_2 and P_3P_4, if either of the angles $\angle P_1P_2P_3$ and $\angle P_2P_3P_4$ is smaller than a given threshold. In our experiments we set this threshold to be $\pi/2$. Fig. 13 demonstrates several examples of applying this extended edge-grouping method. For comparison, we also include the grouping results from the proposed method without the extension of adding continuity. These results show that this extension may produce more favorable grouping results when the desired structure boundary is smooth.

B. Incorporating Intensity Homogeneity

In many real images, the intensity inside a desired salient boundary shows good homogeneity, while the intensity across the boundary varies abruptly. In this section, we extend the proposed method to incorporate such an intensity-homogeneity property to help detect the desired salient boundaries. Particularly, we modify the grouping cost (1) to

$$\phi_h(\mathcal{B}) = \frac{|\mathcal{B}_G|}{\int\int_{R(\mathcal{B})} \sigma_\epsilon^T(x,y)dxdy} \tag{4}$$

with

$$\sigma_\epsilon^T(x,y) = \begin{cases} 1, & \text{if } |I(x,y) - T| < \epsilon \\ 0, & \text{if } |I(x,y) - T| > \epsilon. \end{cases}$$

Here, $I(x,y)$ is the image intensity of the pixel at (x,y). T is a specified pixel intensity for the desired structure enclosed by the detected boundary. T can be either user specified or automatically selected by histogram analysis. $\epsilon \geq 0$ is the expected pixel-intensity variation within the region enclosed by the detected boundary. In essence, this new grouping cost only counts the pixels with an intensity in the range $[T-\epsilon, T+\epsilon]$ in calculating the enclosed region area, and, therefore, favors in detecting a boundary that encloses a region with intensity as close to T as possible. The smaller the value of ϵ, the more homogeneous the region enclosed by the detected boundary.

With this new grouping cost, the graph G remains the same. The only difference is to slightly modify the definition of the second edge weight w_2 to count only the pixels with an intensity in $[T-\epsilon, T+\epsilon]$ in calculating the enclosed region area. Therefore, we can still apply the same graph algorithm to detect the optimal boundary that minimizes this new grouping cost. Fig. 14 shows the experimental results on several real images by applying this extended edge-grouping method. We can see that the proposed extension of incorporating intensity homogeneity can improve the grouping results when we have some *a priori* knowledge on the intensity of the desired structure. For example, in Fig. 14(e), we set a smaller value for T and detect the bird while the original edge-grouping method without this extension detects part of the tree, which shows larger intensity.

We can also replace $\sigma_\epsilon^T(x,y)$ by $\left(1-\sigma_\epsilon^T(x,y)\right)$ in the grouping cost (4) to detect a boundary that encloses a region with intensity not close to T. Fig. 15 shows the experimental results on several real images by using this extension. We can see that this extension may also help improve the grouping performance when it is *a priori* known that the desired salient structure does not show certain intensity. Note that, the proposed edge-grouping method can only produce a single closed boundary in one iteration. Therefore, we cannot detect both eyes in Fig. 15(d) in the same iteration. Instead, we may need

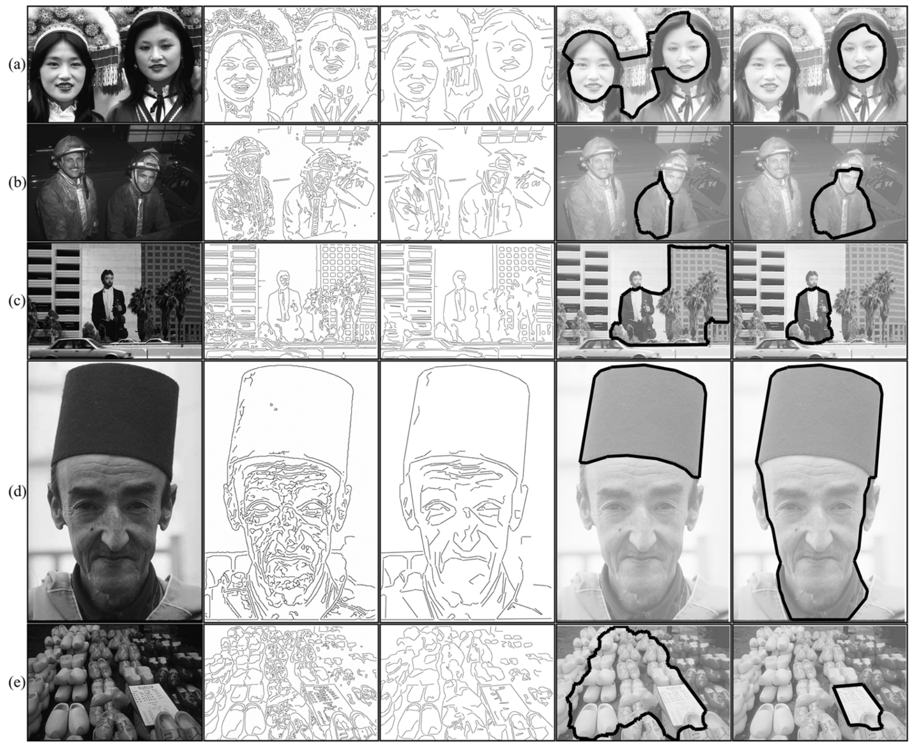


Fig. 13. Sample grouping results of the proposed method with and without the extension of adding the boundary continuity. Each row shows, from left to right, the input image, the Canny detection result, the detected segments, the optimal boundary detected by the proposed method without adding continuity, and the optimal boundary detected by the proposed method extended with continuity. The CPU times for processing images (a)–(e) are 9.93, 13.03, 45.30, 4.26, and 60.08 s, respectively (without the extension of continuity), and 8.60, 10.91, 41.26, 3.61, and 43.14 s, respectively (with the extension of continuity).

to apply the multiple boundary detection strategy introduced in Section V-D to detect them sequentially.

Note that, in this section, we do not use the intensity variance of the enclosed region to measure the region homogeneity. The major reason is that, without knowing the mean intensity of the desired region, it is difficult to define a local weight for each individual segment so that the region intensity variance can be represented by summing up the weights of the involved segments. Therefore, such a grouping problem can not be solved by the graph model and algorithm described in Section IV. In fact, if we use the region intensity variance to measure the region homogeneity, there may not exist a polynomial-time algorithm to solve the resulting grouping problem.

C. Proximity Exponentiation

The grouping cost (1) is simply a ratio between the total gap length along the boundary and enclosed-region area. While we have shown that this grouping cost usually leads to good grouping results in many images, there are cases where the region-area term dominates the grouping cost prompting the proposed method to detect an overly large region that does not align well with any salient structure boundaries. This is mainly due to the fact that region area is quadratic with respect to the boundary perimeter, and, therefore, the total gap length along the boundary. This problem has been found in previous pixel-grouping methods that seek to combine boundary and region information [23].

To address this problem, we introduce a new measurement of proximity by exponentiating the gap length to some power. This would reduce the order of magnitude difference between the proximity and region-area terms. A simple idea is to choose an exponent $\alpha \in [1,2]$ and directly modify the grouping cost to

$$\phi_\alpha(\mathcal{B}) = \frac{|\mathcal{B}_G|^\alpha}{\int\int_{R(\mathcal{B})} dxdy}.$$

However, similar to the case discussed at the end of Section VI-B, the numerator $|\mathcal{B}_G|^\alpha$ is not of an additive form in terms of any weight of the included segments when $\alpha \neq 1$. Therefore, the resulting grouping problem can not be solved by the graph model and algorithm described in Section IV. In fact, there may not exist a polynomial-time algorithm to solve this grouping problem. In this paper, we instead propose to modify the grouping cost to

$$\phi_\alpha(\mathcal{B}) = \frac{\sum\limits_{\Gamma \in \mathcal{B}_G} |\Gamma|^\alpha}{\int\int_{R(\mathcal{B})} dxdy},$$

where Γ is a gap-filling segments along $\mathcal{B}$. To encode this grouping cost into the graph G, we only need to modify the first edge weight of the dashed edges, e.g., e_{12}^+ and e_{12}^- corresponding to the gap-filling segment P_1P_2, to

$$w_1\left(e_{12}^+\right) = w_1\left(e_{12}^-\right) = |P_1P_2|^\alpha.$$

This does not change the graph structure of G and, therefore, we can still apply the same graph model and algorithm in Section IV to detect the optimal boundary that minimizes this new grouping cost.

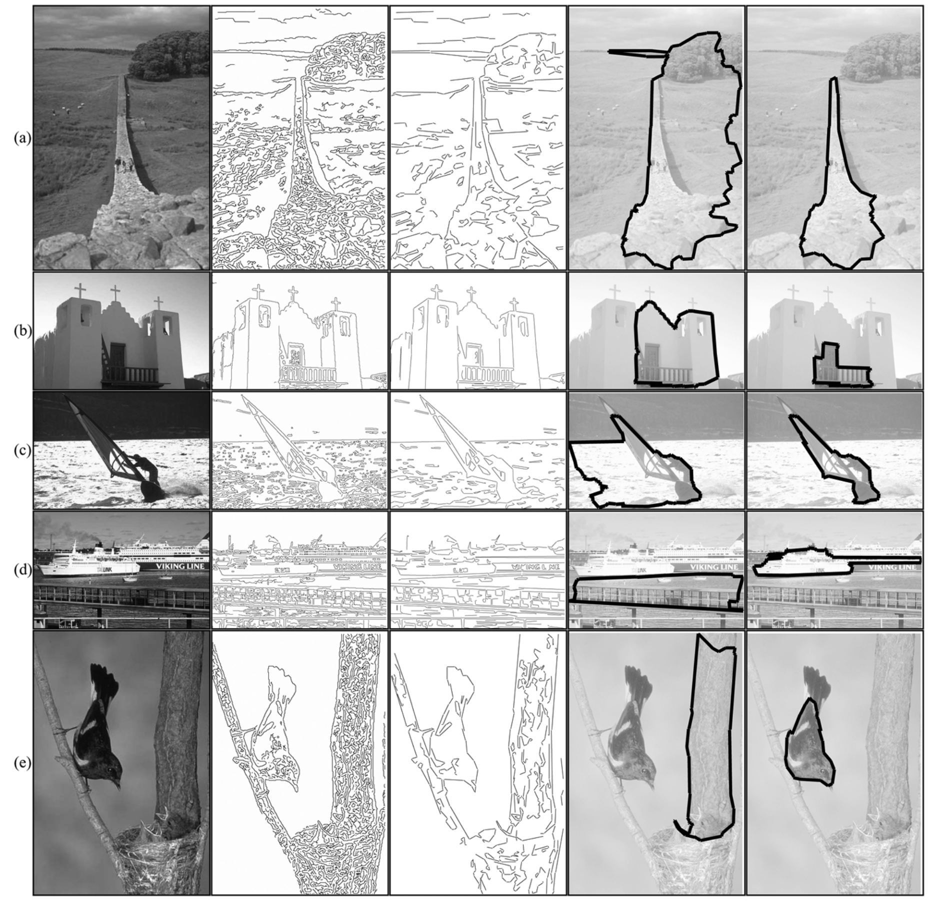


Fig. 14. Edge-grouping results with the extension of incorporating intensity homogeneity. Each row shows, from left to right, the input image, the Canny detection result, the detected segments, the optimal boundary detected by the proposed method without any extension, and the optimal boundary detected by the proposed method with the extension of adding intensity homogeneity. ϵ is set to 50 for all images. T is set to 165, 50, 70, 230, and 50 for images (a)–(e), respectively. The CPU times for processing images (a)–(e) are 82.56, 1.39, 17.27, 29.26, and 22.24 s, respectively (without the extension of intensity homogeneity); and 43.13, 0.53, 9.52, 10.36, and 12.82 s, respectively (with the extension of intensity homogeneity).

Fig. 16 shows an experiment where different α's are used in the proposed edge-grouping method with the extension of proximity exponentiation. We can clearly see that the increase of the proximity exponentiation factor α can reduce the dominance of the region area in the resulting edge grouping. A clear observation is that with the increase of α, we usually detect boundaries with smaller enclosed region areas. Fig. 17 shows more experimental results in applying this extended method on several real images. The grouping results are compared to the proposed edge-grouping method without this extension (or equivalently, with $\alpha = 1$). We can see that setting $\alpha = 1.5$ may lead to more favorable grouping results. This improvement is more noticeable on the image shown in Fig. 17(a), where the grouping result without the extension does not detect any line segments along the boundary of the bird, while setting $\alpha = 1.5$ allows the proposed method to detect the whole bird's boundary accurately.

D. Detecting Simple Boundaries

Just like many previous edge-grouping methods, the proposed edge-grouping method has no guarantee to detect only *simple* boundaries without self intersections. The major reason lies in that the involved gap-filling segments may intersect with other gap-filling or detected segments. The boundaries with self intersections are not desirable since they do not represent the boundaries of any real structures. However, it should be noted first that, in using the proposed method, the nonsimple boundaries do not happen very frequently, since the presence of a self intersection would produce a boundary that encloses multiple subregions with opposite-sign region areas. In this case, the total enclosed area is relatively small and, therefore, such a boundary is not likely to be detected in using the proposed method. For example, the nonsimple boundary shown in Fig. 18(a) encloses two subregions with similar area but different area signs. The total area enclosed by this boundary is in fact close to zero. However, we still need to solve this self-intersection problem when it occurs. In this section, we present a strategy to force the proposed edge-grouping method to produce only *simple* boundaries.

The basic idea of this strategy is that, when a detected boundary contains a self intersection, we try to avoid this self intersection by not allowing the involved intersecting line segments to be included simultaneously in the detected boundary. For example, if the detected boundary $P_1 \ldots P_{16}$ traverses two intersecting line segments P_4P_5 and $P_{11}P_{12}$ as shown in Fig. 18(a), we consider two cases. Case 1: Remove segment P_4P_5 from the input set of segments and repeat the proposed

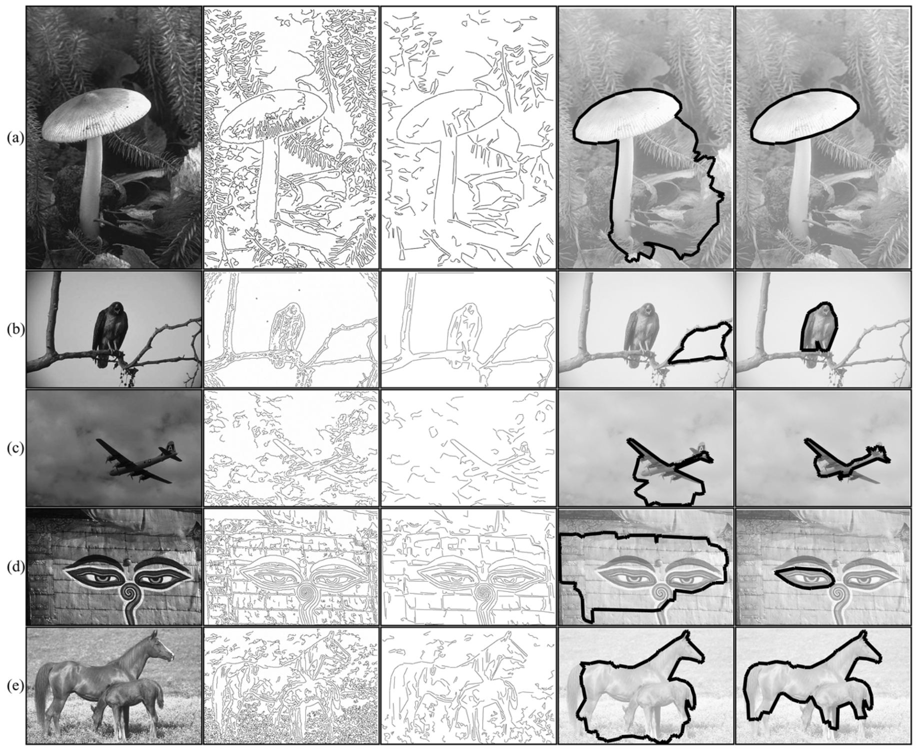


Fig. 15. Each row depicts the same information as in Fig. 14 except that the rightmost column shows the optimal boundary detected by the proposed method with the extension of replacing $\sigma_{\epsilon}^{T}(x,y)$ by $(1-\sigma_{\epsilon}^{T}(x,y))$ in the grouping cost (4). ϵ is set to 50 for all images. T is set to 50, 200, 150, 150, and 220 for images (a)–(e), respectively. The CPU times for processing images (a)–(e) are 42.10, 1.42, 2.52, 59.55, and 77.18 s, respectively (without the extension of intensity homogeneity), and 16.01, 0.65, 0.40, 54.22, and 36.06 s, respectively (with the extension of intensity homogeneity).

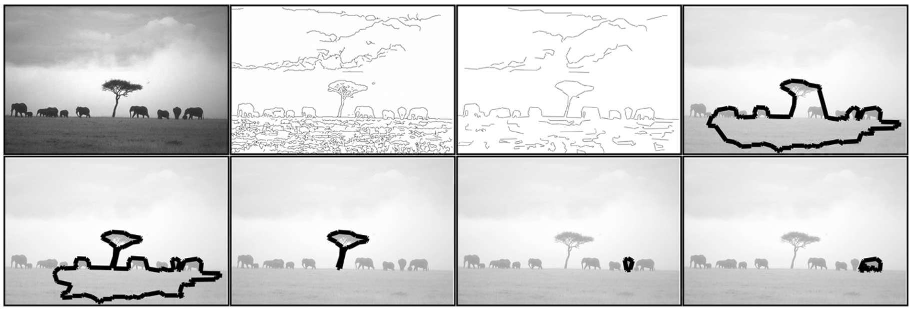


Fig. 16. Illustration of the effect of α in the proposed grouping method with the extension of proximity exponentiation. The first row, from left to right, shows the input image, the Canny detection result, the detected segments, and the optimal boundary detected by the proposed method without proximity exponentiation. The second row, from left to right, shows the optimal boundaries detected by the proposed method with proximity exponentiation factor α =1.1, 1.2, 1.5, 1.7, respectively.

method to obtain a boundary $\mathcal{B}_1$ as shown in Fig. 18(b). Case 2: Remove segment $P_{11}P_{12}$ from the input set of segments and repeat the proposed method to obtain a boundary $\mathcal{B}_2$ as shown in Fig. 18(c). If both $\mathcal{B}_1$ and $\mathcal{B}_2$ are simple, the one with smaller grouping cost $\phi(\cdot)$ is then the final optimal simple boundary. If any one of them is nonsimple, we may continue considering two more cases by further removing one more involved segment.

We can see that the direct implementation of this strategy in fact generates a binary tree where the root node represents an edge grouping on all segments, each nonroot node represents an edge grouping by removing some segments, and each leaf node represents an edge grouping where the detected boundary is simple. It is also easy to see that the grouping cost keeps increasing from a parent node to its children. In practice, we can use a *branch-and-bound* technique [34] to improve the algorithm efficiency: we always process the node with the smallest grouping cost from all the available ones and for any node with a grouping cost larger than a known leaf node (a known detected simple boundary), we are not going to proceed to its children. It is well known that this branch-and-bound strategy finds the optimal solution but may have an exponential time-complexity in the worst case. However, as mentioned above, the consideration of the region-area information makes the proposed edge-grouping method biased to produce simple boundaries. Therefore, even if a nonsimple boundary is detected, we expect that the branch-and-bound tree would have small depth and the optimal simple boundary can be found in a very small number of iterations. In our about 4 500 experiments on detecting the first optimal boundaries (on different real images, with/without various extensions), we only come across one case of detecting

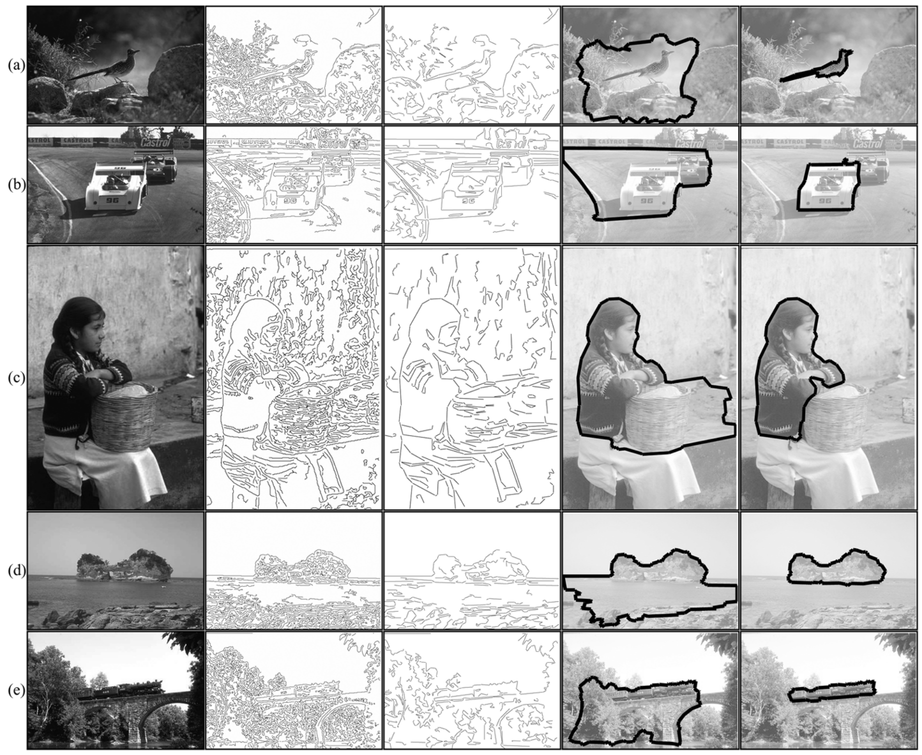


Fig. 17. Edge-grouping results of the proposed method with proximity exponentiation. Each row shows, from left to right, the input image, the Canny detection result, the detected segments, the optimal boundary detected by the proposed method without the proximity exponentiation and optimal boundary detected by the proposed method with a proximity exponentiation factor $\alpha = 1.5$. The CPU times for processing images (a)–(e) are 33.21, 23.55, 42.90, 21.06, and 59.75 s, respectively (without the proximity exponentiation), and 15.78, 17.69, 38.54, 17.22, and 55.49 s, respectively (with the proximity exponentiation factor $\alpha = 1.5$).

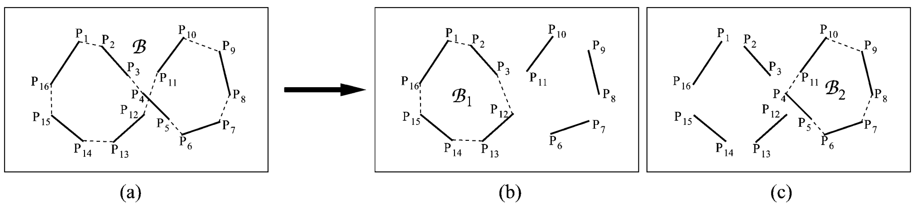


Fig. 18. Illustration of the strategy for detecting only simple boundaries.

a nonsimple boundary, which is shown in Fig. 19. In this case, the optimal simple boundary shown in Fig. 19(d) is achieved after one of the intersecting segments shown in Fig. 19(c) [the zoomed version is shown in Fig. 19(e)] is removed. When repeating the proposed method to detect multiple boundaries as introduced in Section V-D, we may find more cases of self-intersecting boundaries. Note that, in a multiple boundary detection, this branch-and-bound strategy is guaranteed to detect a simple boundary in each iteration, but boundaries detected in different iterations may intersect each other.

E. A Note on the CPU Time

Note that, when considering some of the previous extensions, the proposed method may take less CPU time, as shown in the captions of Figs. 13–15 and Fig. 17. The major reason is that the running time of the proposed method is dominated by the step of finding the desired minimum ratio alternate cycle in the graph G. The actual running time for this step is not only decided by the number of vertices and edges in the graph G, but also the edge weights in G. Particularly, this step consists of several rounds of the minimum weight perfect matching (MWPM) algorithm [13]. Both the needed rounds of MWPM and the running time for each round of MWPM depend on the actual edge weights in G. For example, for the image shown in Fig. 14(d), seven rounds of MWPM (in 1.49, 9.29, 7.86, 3.16, 3.06, 2.60, and 1.57 s, respectively) are taken without the extension of intensity homogeneity and six rounds of MWPM (in 2.04, 6.02, 1.42, 0.18, 0.11, and 0.11 s, respectively) are taken with the extension of intensity homogeneity. While the extensions with intensity homogeneity and proximity exponentiation do not change the number of vertices and edges in G, the extension with continuity reduces the number edges by imposing the additional angle threshold $\pi/2$ in constructing the dashed edges. This is an additional reason for the extension with continuity to take less CPU time. For example, for the image shown in Fig. 13(a), the constructed graph G contains 18 336 dashed edges without the extension of continuity and only 13 696 dash edges with the extension of continuity.

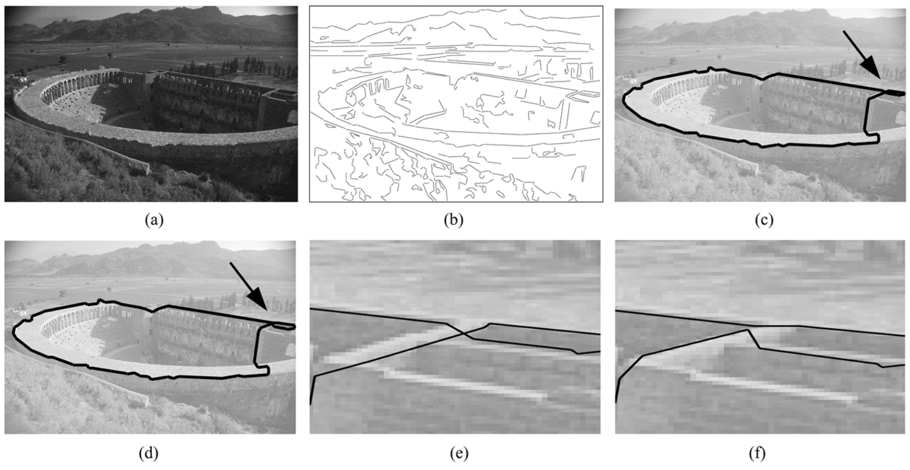


Fig. 19. Example of applying the branch-and-bound strategy to detect simple boundaries. (a) Input image. (b) Detected segments. (c) Optimal boundary detected by the proposed method with the continuity extension (Section VI-A), with $\lambda = 1$. It contains a self intersection pointed by an arrow. (d) Simple boundary detected by the branch-and-bound strategy. (e), (f) Zoomed version of the subregions pointed by arrows in (c) and (d), respectively. The CPU times for processing this image are 21.82 s to detect the boundary in (c) and 54.98 s to detect the boundary in (d).

VII. Conclusion

In this paper, we presented a new edge-grouping method that can detect perceptually salient closed boundaries from an image by combining the boundary and region information. In its baseline form, the boundary proximity and region area are combined into a ratio-form grouping cost function. We then develop a graph model to reduce this edge-grouping problem to a graph problem that can be solved in polynomial time in a globally optimal fashion. We tested this edge-grouping method on a large set of synthetic data and many real images, both with comparisons to the ratio-contour method, which does not consider region information. We showed that the inclusion of region-area information makes the proposed method more robust against image noise and improves the performance in general. We also introduced several useful extensions to the proposed method.

References

[1] G. Kanizsa, *Organization in Vision*. New York: Praeger, 1979.

[2] T. Alter and R. Basri, "Extracting salient contours from images: An analysis of the saliency network," *Int. J. Comput. Vis.*, vol. 27, no. 1, pp. 51–69, 1998.

[3] A. Amir and M. Lindenbaum, "A generic grouping algorithm and its quantitative analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 2, pp. 168–185, Feb. 1998.

[4] J. H. Elder, A. Krupnik, and L. A. Johnston, "Contour grouping with prior models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 6, pp. 661–674, Jun. 2003.

[5] J. H. Elder and S. W. Zucker, "Computing contour closure," in *Proc. Eur. Conf. Computer Vision*, 1996, pp. 399–412.

[6] G. Guy and G. Medioni, "Inferring global perceptual contours from local features," *Int. J. Comput. Vis.*, vol. 20, no. 1, pp. 113–133, 1996.

[7] D. Huttenlocher and P. Wayner, "Finding convex edge groupings in an image," *Int. J. Comput. Vis.*, vol. 8, no. 1, pp. 7–29, 1992.

[8] D. Jacobs, "Robust and efficient detection of convex groups," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 1, pp. 23–27, Jan. 1996.

[9] S. Mahamud, L. R. Williams, K. K. Thornber, and K. Xu, "Segmentation of multiple salient closed contours from real images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 4, pp. 433–444, Apr. 2003.

[10] S. Sarkar and K. Boyer, "Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1996, pp. 478–483.

[11] A. Shashua and S. Ullman, "Structural saliency: The detection of globally salient structures using a locally connected network," in *Proc. IEEE Int. Conf. Computer Vision*, 1988, pp. 321–327.

[12] J. S. Stahl and S. Wang, "Convex grouping combining boundary and region information," in *Proc. IEEE Int. Conf. Computer Vision*, 2005, vol. 2, pp. 946–953.

[13] S. Wang, T. Kubota, J. Siskind, and J. Wang, "Salient closed boundary extraction with ratio contour," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 4, pp. 546–561, Apr. 2005.

[14] L. Williams and K. K. Thornber, "A comparison measures for detecting natural shapes in cluttered background," *Int. J. Comput. Vis.*, vol. 34, no. 2/3, pp. 81–96, 2000.

[15] J. S. Stahl and S. Wang, "Globally optimal grouping for symmetric boundaries," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006, vol. 1, pp. 1030–1037.

[16] A. Farag and E. Delp, "Edge linking by sequential search," *Pattern Recognit.*, vol. 28, no. 5, pp. 611–633, 1995.

[17] E. Saber, A. Tekalp, and G. Bozdagi, "Fusion of color and edge information for improved segmentation and edge linking," *Image Vis. Comput.*, vol. 15, no. 10, pp. 769–780, 1997.

[18] X. Jiang, "An adaptive contour closure algorithm and its experimental evaluation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1252–1265, Nov. 2000.

[19] R. Gonzalez and R. Woods, *Digital Image Processing*. Upper Saddle River, NJ: Prentice-Hall, 2002.

[20] O. Ghita and P. Whelan, "Computational approach for edge linking, journal of electronic imaging," *J. Electron. Imag.*, vol. 11, no. 4, pp. 479–485, 2002.

[21] A. Sappa, "Unsupervised contour closure algorithm for range image edge-based segmentation," *IEEE Trans. Image Process.*, vol. 215, no. 2, pp. 377–384, Feb. 2006.

[22] S. Wang, J. S. Stahl, A. Bailey, and M. Dropps, "Global detection of salient convex boundaries," *Int. J. Comput. Vis.*, vol. 71, no. 3, pp. 337–359, 2007.

[23] I. Cox, S. B. Rao, and Y. Zhong, "Ratio regions: A technique for image segmentation," in *Proc. Int. Conf. Pattern Recognition*, 1996, pp. 557–564.

[24] I. H. Jermyn and H. Ishikawa, "Globally optimal regions and boundaries as minimum ratio cycles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 10, pp. 1075–1088, Oct. 2001.

[25] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[26] B. Sumengen and B. S. Manjunath, "Graph partitioning active contours (GPAC) for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 509–521, Apr. 2006.

[27] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 266–277, Feb. 2001.

[28] H. Wang and J. Oliensis, "Salient contour detection using a global contour discontinuity measurement," in *Proc. IEEE Workshop Perceptual Organization in Computer Vision*, 2006, pp. 190–190.

[29] S.-P. Liou, A. H. Chiu, and R. C. Jain, "A parallel technique for signal-level perceptual organization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 4, pp. 317–325, Apr. 1991.

[30] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Jun. 1986.

[31] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin*, Network Flows: Theory, Algorithms, & Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[32] P. D. Kovesi, *Matlab Functions for Computer Vision and Image Analysis*.

[33] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Int. Conf. Computer Vision*, Jul. 2001, vol. 2, pp. 416–423.

[34] R. Horst and H. Tuy*, Global Optimization: Deterministic Approaches*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.

Joachim S. Stahl (S'03) received the B.S. degree in computer science from the Augusta State University, Augusta, GA, in 2003, and the M.E. degree in computer science and engineering from the University of South Carolina, Columbia, in 2005, where he is currently pursuing the Ph.D. degree in the Department of Computer Science and Engineering.

His research interests are in the fields of computer vision and image processing, including the areas of perceptual organization, segmentation, and image understanding.

Song Wang (M'03) received the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, in 2002.

From 1998 to 2002, he was a Research Assistant in the Image Formation and Processing Group, Beckman Institute, University of Illinois at Urbana-Champaign. Since August 2002, he has been an Assistant Professor in the Department of Computer Science and Engineering, University of South Carolina, Columbia. His research interests include computer vision, medical image processing, and machine learning.