Globally Optimal Grouping for Symmetric Closed Boundaries By Combining Boundary and Region Information

Joachim S. Stahl, Song Wang[1]

Abstract

Many natural and man-made structures have a boundary that shows a certain level of bilateral symmetry, a property that plays an important role in both human and computer vision. In this paper, we present a new grouping method for detecting closed boundaries with symmetry. We first construct a new type of grouping token in the form of symmetric trapezoids by pairing line segments detected from the image. A closed boundary can then be achieved by connecting some trapezoids with a sequence of gap-filling quadrilaterals. For such a closed boundary, we define a unified grouping cost function in a ratio form: the numerator reflects the boundary information of proximity and symmetry and the denominator reflects the region information of the enclosed area. The introduction of the region-area information in the denominator is able to avoid a bias toward shorter boundaries. We then develop a new graph model to represent the grouping tokens. In this new graph model, the grouping cost function can be encoded by carefully designed edge weights and the desired optimal boundary corresponds to a special cycle with a minimum ratio-form cost. We finally show that such a cycle can be found in polynomial time using a previous graph algorithm. We implement this symmetry-grouping method and test it on a set of synthetic data and real images. The performance is compared to two previous grouping methods that do not consider symmetry in their grouping cost functions.

Keywords – Perceptual organization, edge grouping, boundary detection, boundary symmetry, edge detection, graph models.

1 Introduction

The boundaries of many structures of interest encountered in the real world show a certain level of (bilateral) symmetry [4, 5]. For example, most objects (or their components) that are machine fabricated have a revolved surface that is perfectly symmetric over a straight axis. Many natural objects, such as leaves and animals, also have a boundary with a certain level of symmetry, where the symmetry axes may

[1]Corresponding author. Department of Computer Science and Engineering, University of South Carolina, Columbia SC 29208. Email: songwang@cse.sc.edu; Tel: 1-803-777-2487; Fax: 1-803-777-3767.

not be perfectly straight, as shown in Fig. 1(c). In computer vision, symmetry has been shown to be an important property in both boundary *interpretation/matching*, where the goal is to analyze and match given boundaries, and *grouping*, where the goal is to extract salient structural boundaries from real images [10]. As shown in Fig. 1, the goal of this paper is to develop an effective method to address the latter problem of grouping for symmetric boundaries, which, as pointed out in [10], is a particularly challenging problem.

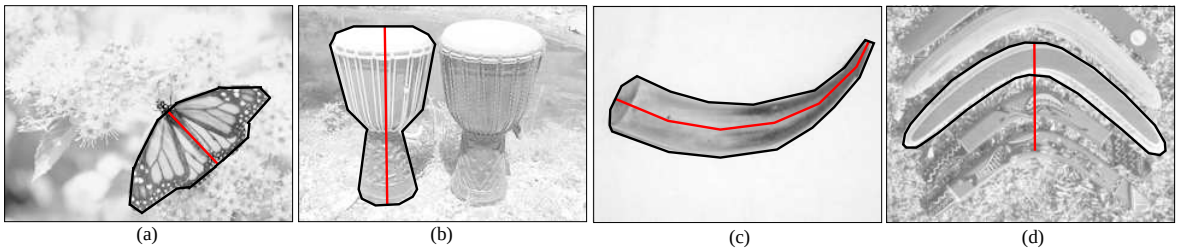


Figure 1: Four samples of structural boundaries that show symmetry. Structural boundaries are shown in black while the symmetry axes are shown in red.

Several reasons make the grouping for symmetric boundaries a challenging problem. First, unlike many other grouping cues, boundary symmetry is not a simple local measure: two symmetric fragments along the resulting boundary may be located far away from each other. As a result, it is usually difficult to encode symmetry into the simple locally-constructed grouping tokens, such as image pixels or boundary fragments, which have been widely used in previous grouping methods. Second, while symmetry is an important grouping cue, other cues, such as Gestalt laws of proximity and closure, are also crucial to achieve a successful grouping [16]. This calls for a unified *grouping cost (function)* that can flexibly integrate different grouping cues. Third, the grouping cost should be designed to avoid undesirable explicit or implicit biases, such as a bias toward shorter boundaries, which occurs in many previous grouping methods [37]. Finally, it is usually a challenging problem to develop an optimization algorithm for finding a grouping that minimizes the selected grouping cost.

In this paper, we developed a new grouping method for detecting 2D closed boundaries with symmetry. Particularly, we propose a new grouping cost function that takes a ratio form: the numerator reflects the boundary information of proximity and symmetry and the denominator reflects the region information of the enclosed area. The use of the enclosed region area makes the resulting grouping biased to detecting longer and rounder boundaries and therefore, promotes the robustness to image noise and texture. This grouping cost function can be expanded to include other boundary information, such as boundary continuity (smoothness), and region information, such as region intensity homogeneity. To quantify and encode

this grouping cost, we construct a new type of grouping token in the form of trapezoids by pairing line segments detected from the image. Based on these trapezoids, the problem of grouping for symmetric boundaries can be formulated as identifying and connecting a sequence of trapezoids into a closed boundary. Finally, we construct a new graph model where candidate boundaries are represented as special cycles in this graph, and apply a known graph algorithm to find the optimal cycle that corresponds to the boundary with the minimum grouping cost. Note that the work in this paper is only concerned with the case where the boundary of interest shows bilateral symmetry on the 2D image plane. We do not consider the case where the surface of an object shows symmetry over an axis in 3D space but the resulting 2D boundary does not show bilateral symmetry on the 2D image plane due to the perspective-projection transformation, as in [36, 18, 22, 7].

The important role and use of symmetry has been studied in both human vision and computer vision [35, 41, 30, 27, 26, 11]. Particularly, prior research has shown that symmetry is non-accidental [21, 40] and therefore, can be used as a grouping cue to distinguish salient structures from noisy background. Symmetry analysis of a given object boundary is usually conducted by deriving its *symmetry axis*. Symmetry-axis information has been incorporated to facilitate boundary interpretation, matching, and recognition in many prior research [5, 6, 19, 42]. Note that, different from the problems of boundary interpretation, matching, and recognition, the work presented in this paper aims to solve the grouping problem, where the structural boundaries are not available and our goal is to extract them from real images.

The related work includes the long-line research on *edge grouping* [2, 3, 8, 9, 12, 14, 15, 23, 28, 29, 33, 37, 38, 39]. These methods aim to extract perceptually salient boundaries from a set of line segments, which are usually detected from an image by edge detectors and line-fitting operators. In previous edge-grouping methods, the grouping cost usually combines well known Gestalt laws, such as (a) *closure*, which requires the resulting boundary to be always closed, (b) *proximity*, which requires the gap length to be short in connecting the detected line segments into a closed boundary, (c), *continuity*, which requires the resulting boundary to be as smooth as possible, and (d) *convexity*, which requires the resulting boundary to be convex. However, these edge-grouping methods do not consider the boundary symmetry in grouping.

Mohan and Nevatia [24] developed a grouping method, where boundary symmetry is considered along with closure and proximity. It applies both edge detection and corner detection to extract a set of line segments and corner points as the grouping tokens. The grouping cost is defined by a collinearity measure that actually reflects the proximity and continuity of the boundary. Symmetry is applied as a cue to couple

the extracted curves by producing a set of *ribbons*. These ribbons are then grouped into structures by some heuristic algorithms. This method does not introduce a unified grouping cost function and the developed grouping algorithm is not globally optimal. The locality of the grouping algorithm may lead to many small ribbons and may not handle the boundary occlusion very well. Another work that is closely related to the proposed work is the grouping method developed by Liu, Geiger, and Yuille [20], which identifies the local symmetry-axis segments and then applies a shortest-path algorithm to connect some of them into a complete symmetry axis. The grouping cost function is defined as the sum of a predefined local cost along the symmetry axis. By manually selecting a starting pair of points that are symmetric to each other, this method produces an open boundary. Since this method does not consider region information or other normalization in the cost function, it presents a bias toward shorter boundaries, which may have difficulty in detecting the symmetry axis shown in Fig. 1(d).

In recent years, many methods have also been developed for detecting structures with symmetric appearance [25, 13, 31, 32]. For example, Prasad and Yegnanarayana [26] develop a voting-based method to detect axes of bilateral symmetry directly from images based on edge-gradient information. Note that these methods are quite different from the grouping method proposed in this paper from the following three perspectives: (a) most of these methods assume the appearance symmetry while the proposed method only assumes the boundary-shape symmetry, (b) these methods usually assume the symmetry axes to be straight while the proposed method quantifies symmetry as a continuous value, and (c) these methods usually detect only the symmetry axes, but not the final structural boundaries, while the proposed method detects both symmetry axes and resulting structural boundaries.

The method proposed in this paper is inherited from the previous ratio-contour method [37], an edge-grouping method for detecting smooth closed boundaries. Particularly, both of them use the same graph algorithm: the minimum-ratio alternate cycle algorithm, for solving the final graph problems. However, both the research goals, the problem formulations, and the graph modelings introduced in this paper are completely different from the ones introduced in [37]. The research goal in this paper is to develop a grouping method to detect boundaries with good bilateral symmetry, which is not considered in the ratio-contour method. To achieve this goal, in this paper we introduce different grouping tokens, define a different grouping cost with a normalization over the enclosed region area, construct a different graph model with "mirror" edges and auxiliary edges, and define different graph edge weight functions to encode the region-area information. In Section 5, we also compare the performance of the proposed method and

the ratio-contour method on both synthetic data and real images.

The remainder of this paper is organized as follows. In Section 2, we formulate the problem of grouping with symmetry by introducing the general edge-grouping methodology, the new grouping tokens in the form of symmetric trapezoids, and the new unified grouping cost function. In Section 3, we introduce the graph modeling of the formulated problem and apply a graph algorithm to solve this grouping problem in a globally optimal fashion. In Section 4, we introduce a more accurate way to measure the gap length along the boundary. In Section 5, we discuss implementation details and report experiment results on both synthetic and real images. In Section 6, we discuss the possible extensions of the proposed method to incorporate other boundary or region information. In Section 7, we discuss the complexity and running time of the proposed grouping method. Section 8 presents the conclusions.

2 Problem Formulation

The proposed grouping method has its roots in *edge grouping*, where grouping tokens are a set of line segments (or more generally, curve segments) and the output is one or several perceptually-salient boundaries formed by connecting a subset of the line segments. However, to encode and quantify the boundary symmetry, we further pair the line segments to construct a new type of grouping token in the form of symmetric trapezoids. In this section, we start the problem formulation by introducing the typical process of edge grouping. We then elaborate on the trapezoid-type token construction and the grouping-cost definition.

2.1 Edge Grouping

In edge grouping, a set of line segments is first constructed from the input image, as shown in Fig. 2(a), by edge detection and line fitting operations, as shown in Fig. 2(b). A new set of line segments, as shown by dashed lines in Fig. 2(c), is then constructed to fill the gap between each pair of initial line segments. For convenience, we call the initial line segments resulting from edge detection the *detected (line) segments* and the newly constructed ones the *gap-filling (line) segments*. A *(closed) boundary* of interest is then a simple cycle that traverses a set of detected and gap-filling segments *alternately*, as shown in Fig. 2(d). Note that we do not show all the constructed gap-filling segments in Fig. 2(c) to prevent the figure from being too crowded. In the ideal case, with n detected segments, we have $2n$ segment endpoints, and therefore, we may need to construct $n(2n-2)$ gap-filling segments if we construct a gap-filling segment

between any possible two segment endpoints except for the two endpoints of the same detected segment. Finally, we define a grouping cost function for the boundaries and develop an algorithm to find from all boundaries the one with the minimum grouping cost, also as shown in Fig. 2(d). As mentioned above, various grouping cues, such as proximity, closure, continuity, and convexity have been incorporated into edge grouping trying to extract the perceptually salient structural boundaries from a noisy background [12, 23, 28, 29, 33, 37, 38]. In prior edge-grouping methods, the grouping cost is usually defined to be a function of some local weights associated to each individual detected/gap-filling segment. However, it is difficult to incorporate symmetry into these edge-grouping methods because of the difficulty of encoding symmetry into each individual line segment. In the next section, we construct new trapezoid-type tokens from these line segments to encode boundary symmetry.

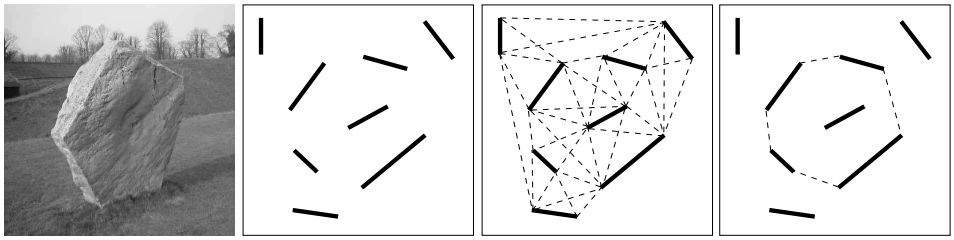


Figure 2: An illustration of the process of edge grouping.

2.2 Symmetric Trapezoids as Grouping Tokens

While it is difficult to encode symmetry into each individual line segment, symmetry can be encoded to a pair of segments. For each pair of line segments P_1P_2 and P_3P_4, as shown in Fig. 3, we can identify their symmetric portions by following three steps:

1. Find the angle-bisector line l between P_1P_2 and P_3P_4, as shown in Fig. 3(a). If $P_1P_2//P_3P_4$, then l is the line equidistant to both P_1P_2 and P_3P_4.

2. Find the projections of both segments P_1P_2 and P_3P_4 to l and denote them $P_1'P_2'$ and $P_3'P_4'$ respectively. The overlap of segments $P_1'P_2'$ and $P_3'P_4'$, as shown by $P_1'P_2'$ in Fig. 3(b), is denoted as the *axis segment* between P_1P_2 and P_3P_4. We refer to P_1' and P_2' as the *axis-segment endpoints*.

3. Map this axis segment back to segments P_1P_2 and P_3P_4. This results in a *(symmetric) trapezoid* $P_1P_2P_2''P_1''$, as shown in Fig. 3(c).

In this paper, we construct such symmetric trapezoids (with axis segments) by pairing every two detected segments, as shown in Figs. 3(a-c), and pairing each gap-filling segment with each detected segment, as shown in Figs. 3(d-e). Note that, for some pairs of segments, their projections to l have no overlap. In this case, no symmetric trapezoid will be constructed for them. These trapezoids are used as new grouping tokens in the proposed method. We construct a trapezoid by pairing a gap-filling segment with a detected segment to handle the case shown in Fig. 4, where the desired symmetric boundary (in black) and its symmetric axis (in red) are shown in Fig. 4(a). However, the symmetric portion of many detected segments, such as P_1P_2, P_3P_4, and P_6P_7, are not detected and therefore, correspond to gap-filling segments. In this case, we can represent the desired symmetric boundary only by pairing gap-filling segments with detected segments, as shown in Fig. 4(b). For convenience, in the remainder of the paper we usually display trapezoids by assuming that they are constructed from two detected segments and therefore, use solid lines for both non-parallel opposite sides, as illustrated in Fig. 3(c). However, we will consider both cases shown in Figs. 3(c) and (e) in developing the proposed grouping method.

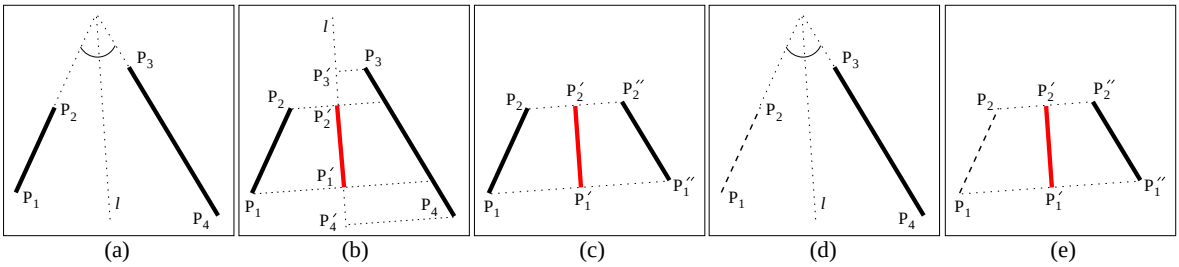


Figure 3: An illustration of constructing a symmetric trapezoid from a pair of segments: (a-c) pairing two detected segments, and (d-e) pairing a gap-filling segment P_1P_2 with a detected segment P_3P_4.

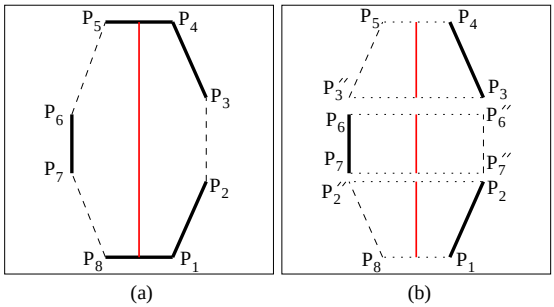


Figure 4: An illustration of the case that needs the construction of trapezoids by pairing gap-filling segments with detected ones. (a) The desired symmetric boundary with its detected segments (solid black lines), gap-filling segments (dashed black lines), and symmetric axis (red line), (b) trapezoids constructed by pairing gap-filling segments with detected ones.

2.3 Trapezoid Grouping and Grouping Cost

Analogous to edge grouping, we formulate the problem of symmetry grouping as a problem of identifying a subset of trapezoids and connecting them sequentially into a closed boundary. The gap between two sequential trapezoids in the connection is filled by a *(gap-filling) quadrilateral*. Two examples are shown in Figs. 5(a) and (b), where two gap-filling quadrilaterals $\mathcal{G}_1 = \{P_2P_3P_{10}P_{11}\}$ and $\mathcal{G}_2 = \{P_4P_5P_8P_9\}$ are constructed to connect three trapezoids $\mathcal{T}_1 = \{P_1P_2P_{11}P_{12}\}$, $\mathcal{T}_2 = \{P_3P_4P_9P_{10}\}$, and $\mathcal{T}_3 = \{P_5P_6P_7P_8\}$ into a closed polygonal boundary $P_1P_2 \ldots P_{12}$. The axis segments of $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$, shown as red solid lines, are also connected by the axis segments of the quadrilaterals $\mathcal{G}_1$ and $\mathcal{G}_2$, shown as red dashed lines, to generate the polyline $Q_1Q_2 \ldots Q_6$, which is the *(boundary) axis* of the closed boundary $P_1P_2 \ldots P_{12}$. We call Q_1 and Q_6 the *boundary-axis endpoints*. Note that the gap-filling quadrilaterals are simply constructed by connecting a parallel side of one trapezoid and a parallel side of another trapezoid. They may not be symmetric and its axis segment is constructed simply by connecting the endpoints of the axis segments of the two neighboring symmetric trapezoids. As in edge grouping, in the ideal case, we may construct quadrilaterals between each pair of trapezoids. Since the axis segment of a trapezoid has two endpoints, there are four different gap-filling quadrilaterals that can be constructed between a pair of trapezoids. As discussed later, in practice we do not need to construct all possible gap-filling quadrilaterals because many of them are not likely to be included in the desired optimal boundary.

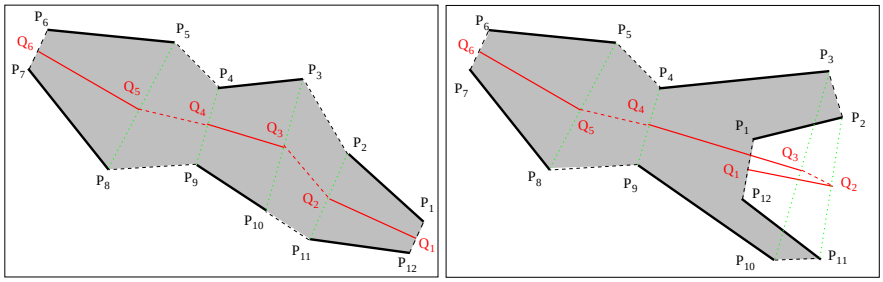


Figure 5: Two examples of grouping detected trapezoids into a closed boundary.

Based on the above formulation, we can measure the symmetry of a closed boundary using the collinearity (straightness) of its boundary axis, as shown by the red polylines in Fig. 5. Specifically, we define the grouping cost function for a closed boundary $\mathcal{B}$ as

$$\phi(\mathcal{B}) = \frac{|\mathcal{B}_D| + \lambda \cdot \rho(\texttt{axis}(\mathcal{B}))}{\texttt{area}(\mathcal{B})}, \tag{1}$$

where $|\mathcal{B}_D|$ is the total gap length along the boundary $\mathcal{B}$. Note that not only quadrilaterals but also trapezoids may contribute to $|\mathcal{B}_D|$, because (a) trapezoids may be constructed by pairing a gap-filling segment with a detected segment, as shown in Figs. 3(d-e), and (b) a parallel side of a trapezoid may contribute to the boundary gap-length $|\mathcal{B}_D|$ when this side contains a boundary-axis endpoint, e.g., the parallel sides P_1P_{12} and P_6P_7 need to be included in calculating $|\mathcal{B}_D|$ in Fig. 5. In Section 4, we will further discuss a more accurate way to measure $|\mathcal{B}_D|$ in practice. The term of $|\mathcal{B}_D|$ reflects the preference of a boundary with good proximity. The term of $\rho(\texttt{axis}(\mathcal{B}))$ is a measure related to the collinearity of $\mathcal{B}$'s axis (e.g., the red polylines in Fig. 5). This term reflects the preference of a boundary with good symmetry and we will elaborate on this term in Section 3. The term of $\texttt{area}(\mathcal{B})$ is the region area enclosed by the boundary $\mathcal{B}$. A normalization over this term sets a preference to produce larger rounder structures, which improves the robustness against image noise by avoiding a bias toward shorter boundaries. Note that, in practice such a preference may not prevent the detection of small salient structures if they show good proximity and symmetry, as we will see in many experiments reported in Section 5. $\lambda > 0$ is a preset factor that balances the weights of the proximity and symmetry terms. As discussed later, we simply set a consistent $\lambda = 10$ in all the experiments reported in this paper.

3 Graph Modeling and Algorithm

In this section we construct a new graph model to describe the above formulated problem. In this graph model, trapezoids and gap-filling quadrilaterals are represented by graph edges. By encoding the grouping cost (1) into the graph edge weights, we then reduce the grouping problem to a problem of finding an optimal cycle with minimum ratio cost in this new graph. We finally apply a known graph algorithm to address this cycle-finding problem. Specifically, the graph construction consists of two sequential steps: (a) constructing solid and dashed edges to represent trapezoids and gap-filling quadrilaterals, respectively, and (b) further constructing auxiliary edges between the vertices corresponding to potential axis endpoints. We elaborate on these two steps in the following.

3.1 Graph Construction I: Solid/Dashed Edges

We construct an undirected graph $G = (V, E)$ with vertex set V and edge set E to model the trapezoids and gap-filling quadrilaterals. To encode the enclosed region area $\texttt{area}(\mathcal{B})$, we construct a pair of *solid* edges e_T^+ and e_T^- for each trapezoid $\mathcal{T}$, and a pair of *dashed* edges e_G^+ and e_G^- for each quadrilateral $\mathcal{G}$, as

shown in Fig. 6. We call the pair of the edges constructed for the same trapezoid or quadrilateral to be the *mirror* edges of each other. For convenience, we can treat each pair of mirror edges to be an abstraction of the axis segment of the corresponding trapezoid or quadrilateral. For example, e_T^+ and e_T^- model the axis segment Q_1Q_2 in Fig. 6(a). Accordingly, we construct a pair of vertices for each axis-segment endpoint (or each parallel side of each trapezoid). For example, vertex pair $u_1^{(1)}$ and $u_1^{(2)}$ are constructed for axis-segment endpoint Q_1 in Fig. 6(a).

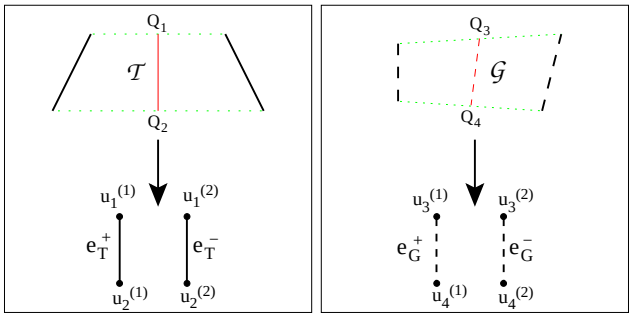


Figure 6: An illustration of the solid/dashed edges. (a) For a trapezoid $\mathcal{T}$, we construct a pair of solid edges e_T^+ and e_T^-. Each axis-segment endpoint Q_i is then modeled by two vertices $u_i^{(1)}$ and $u_i^{(2)}$. (b) For a gap-filling quadrilateral $\mathcal{G}$, we proceed similarly as in (a), except that the two constructed edges are dashed.

In the proposed grouping, two neighboring trapezoids are connected by a quadrilateral as shown in Fig. 5. Therefore, in constructing the graph $G = (V, E)$, we connect two pairs of solid edges by a pair of dashed edges, since each trapezoid and quadrilateral is represented by two mirror edges e^+ and e^-, We apply the following two steps to determine the edge connection in constructing the graph $G = (V, E)$. First, we consider only the quadrilateral that leads to a non-intersected boundary. Particularly, when connecting two trapezoids by a quadrilateral, the two sides of the selected quadrilateral that are not shared with the two trapezoids must not intersect with each other. For example, in Fig. 7, we construct the quadrilateral in the form of $P_2P_3P_6P_7$ with two sides P_2P_3 and P_6P_7, instead of the quadrilateral $P_2P_6P_3P_7$ with two sides P_2P_6 and P_3P_7, to connect $\mathcal{T}_1$ and $\mathcal{T}_2$.

Second, we distinguish two mirror edges e^+ and e^- by associating an implicit direction to the corresponding trapezoid or quadrilateral. Particularly, we set edge e^+ to imply that its corresponding trapezoid or quadrilateral has a counterclockwise direction and edge e^- to imply that its corresponding trapezoid or quadrilateral has a clockwise direction. Then the edge connection will be uniquely determined by requiring the resulting boundary to be of a consistent direction, either clockwise or counterclockwise.

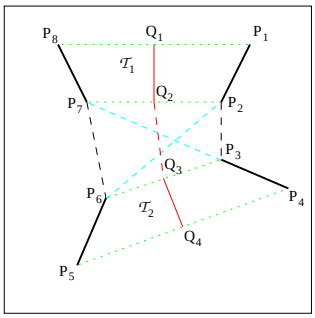


Figure 7: An illustration of determining the gap-filling quadrilateral between two trapezoids $\mathcal{T}_1 = \{P_1P_2P_7P_8\}$ and $\mathcal{T}_2 = \{P_3P_4P_5P_6\}$.

Figures 8(a-d) illustrate the four possible cases. When the clockwise $\mathcal{T}_1$ is connected to the clockwise $\mathcal{T}_2$ by a gap-filling quadrilateral (this implies that the counterclockwise $\mathcal{T}_1$ is connected to the counterclockwise $\mathcal{T}_2$), we connect their corresponding edges with same signs, as shown in Figs. 8(a) and (c). When the clockwise $\mathcal{T}_1$ is connected to the counterclockwise $\mathcal{T}_2$ by a gap-filling quadrilateral (this implies that the counterclockwise $\mathcal{T}_1$ is connected to the clockwise $\mathcal{T}_2$), we connect their corresponding edges with opposite signs, as shown in Figs. 8(b) and (d). The sign of the edge corresponding to the quadrilateral can also be uniquely determined by following the directions of the trapezoids. For example, in Fig. 8(a), the quadrilateral between the clockwise $\mathcal{T}_1$ and the clockwise $\mathcal{T}_2$ has a clockwise direction. This indicates that the dashed edge between $e^+_{T_1}$ and $e^+_{T_2}$ is e^+_G and the dashed edge between $e^-_{T_1}$ and $e^-_{T_2}$ is e^-_G.

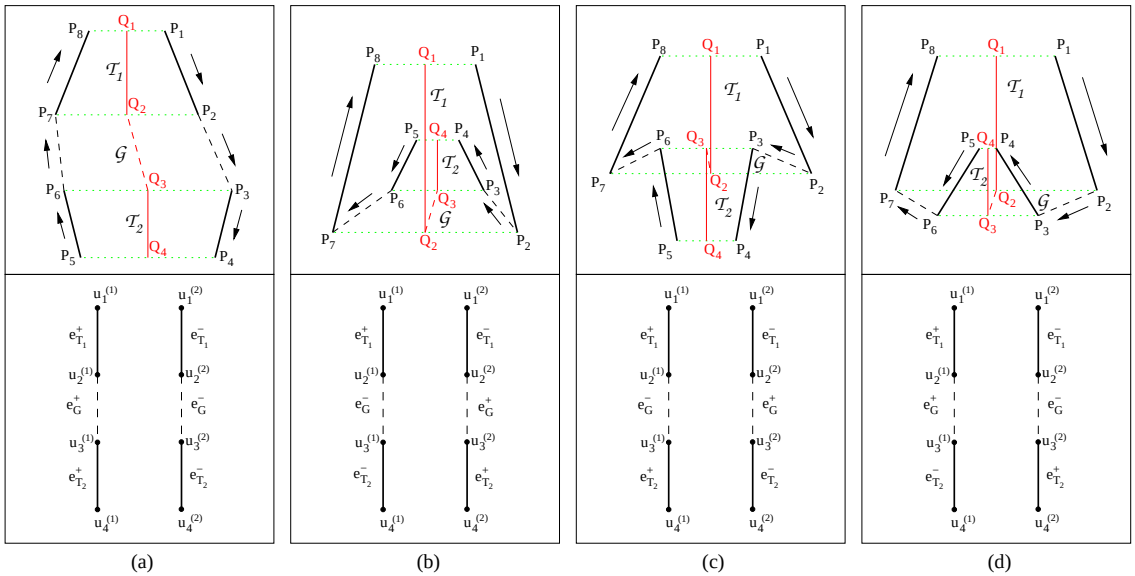


Figure 8: An illustration of the edge connection for modeling the construction of a quadrilateral between two trapezoids.

There is a special case where the constructed gap-filling quadrilateral contains a self intersection, as shown by the quadrilateral $P_2P_3P_6P_7$ in Fig. 9. Note that this self intersection is caused by the two sides P_3P_6 and P_2P_7 from the two neighboring trapezoids and will not be included in the resulting boundary. Therefore, different from the quadrilateral self-intersection mentioned before, this kind of self-intersected quadrilateral is allowed as it does not lead to self-intersected boundaries. In this case, the quadrilateral is divided into two triangles R_1 and R_2 with opposite directions, i.e., if R_1 is counterclockwise, then R_2 is clockwise, also as shown in Fig. 9. We set this quadrilateral's direction to be the direction of the triangle with a larger area and set the area of this quadrilateral as $|\texttt{area}(R_1) - \texttt{area}(R_2)|$.

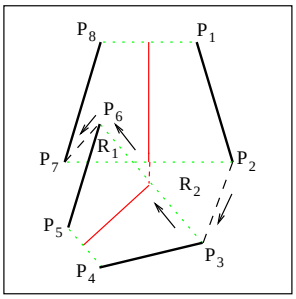


Figure 9: An illustration of an allowed self-intersected quadrilateral.

For each edge e in this graph, we define two weight functions $w_1(e)$ and $w_2(e)$. If e is solid, we define $w_1(e) = 0$ if the corresponding trapezoid was constructed from two detected segments, e.g., the trapezoid shown in Fig. 4(c), otherwise we set $w_1(e)$ to be the length of the gap-filling segment included in the trapezoid construction, e.g., $|P_1P_2|$, the Euclidean distance between P_1 and P_2, in Fig. 4(e). If e is dashed, e.g., corresponding to axis segment Q_2Q_3 in Fig. 7, we define

$$w_1(e) = |P_2P_3|_D + |P_7P_8|_D + \lambda \cdot \rho(e)$$

where $|P_2P_3|_D + |P_7P_8|_D$ is the total gap length (along the boundary) that results from the quadrilateral corresponding to e. In this paper, we use an improved gap-length measure $|P_iP_j|_D$ instead of the Euclidean distance $|P_iP_j|$ to handle the case where a gap-filling quadrilateral may (partially) coincide with detected segments. We will discuss this improved gap-length measure in Section 4. $\rho(e) = |\sin(\angle Q_1Q_2Q_4)| + |\sin(\angle Q_1Q_3Q_4)|$ measures the collinearity (straightness) of the polyline $Q_1Q_2Q_3Q_4$, giving a lower cost to polylines with good collinearity, as shown in Fig. 7. We can see that the first edge weight $w_1(e)$ is always nonnegative and for a pair of mirror edges e^+ and e^-, we have $w_1(e^+) = w_1(e^-)$. The second edge weight, $w_2(e)$, is simply the signed area of the trapezoid or quadrilateral corresponding to e, i.e.,

$w_2(e^+) = -w_2(e^-) > 0$. We introduce the negative area to handle the nonconvex boundaries, where the inclusion of an additional trapezoid or quadrilateral may contribute negatively to the resulting enclosed area $\texttt{area}(\mathcal{B})$, as shown by the trapezoid $P_1P_2P_{11}P_{12}$ in Fig. 5(b).

3.2 Graph Construction II: Auxiliary Edges

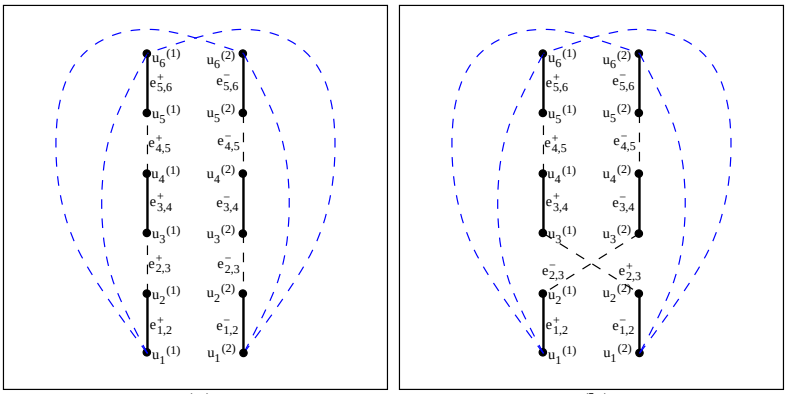
In order to include the boundary information around the two boundary-axis endpoints, e.g., the gap length along P_6P_7 and P_1P_{12} in Fig. 5, we further construct a set of new dashed edges, named *auxiliary* edges, between the vertices corresponding to the two boundary-axis endpoints. Let $u_1^{(1)}$ and $u_1^{(2)}$ be the vertex pair corresponding to an boundary-axis endpoint, e.g., Q_1 in Fig. 5(a) (or (b)), and $u_6^{(1)}$ and $u_6^{(2)}$ be the vertex pair corresponding to the other boundary-axis endpoint, e.g., Q_6 in Fig. 5(a) (or (b)). We construct four **dashed** edges $(u_1^{(i)}, u_6^{(j)}), i, j = 1, 2$ as *auxiliary* edges, as shown by blue curves in Fig. 10(a) (or (b)). For these four auxiliary edges, we set their second edge weights to be zero and the first edge weights to be the total gap length around these two boundary-axis endpoints. For example, in Fig. 5(a) (or (b)), the first weight $w_1(e)$ of the four auxiliary edges that reflects the connection between Q_1 and Q_6 is defined by

$$w_1(e) = |P_1P_{12}|_D + |P_6P_7|_D,$$

where $|P_iP_j|_D$ is the gap length between P_i and P_j, to be detailed in Section 4. Since the optimal closed boundary and its boundary-axis endpoints are unknown before the grouping, we can treat all axis-segment endpoints as potential boundary-axis endpoints and construct auxiliary edges between all of them. In practice, however, we do not need to construct auxiliary edges between all axis-segment endpoints and we will discuss this in more detail in Section 5.

In the graph $G = (V, E)$, each boundary $\mathcal{B}$ is represented by two "mirror" cycles $\mathcal{C}^+$ and $\mathcal{C}^-$, e.g., if an edge is included in $\mathcal{C}^+$, its mirror edge must be included in $\mathcal{C}^-$, and vice versa. In addition, each of them contains an auxiliary edge, as shown in Fig. 10. These two cycles traverse a sequence of solid and dashed edges alternately and therefore, we call them *alternate* cycles. It is easy to verify that (a) the total first edge weights along both $\mathcal{C}^+$ and $\mathcal{C}^-$ are always equal to the numerator of the cost $\phi(\mathcal{B})$ in Eq. (1), and (b) the total second edge weights along $\mathcal{C}^+$ and $\mathcal{C}^-$ have the same absolute value equal to $\texttt{area}(\mathcal{B})$, but with opposite signs. This way, we have

$$\phi(\mathcal{B}) = \frac{\sum_{e\in\mathcal{C}^+} w_1(e)}{\sum_{e\in\mathcal{C}^+} w_2(e)} = -\frac{\sum_{e\in\mathcal{C}^-} w_1(e)}{\sum_{e\in\mathcal{C}^-} w_2(e)},$$



(a) (b)

Figure 10: An illustration of the cycles corresponding to the boundaries shown in Figs. 5(a) and (b), respectively. (a) For the boundary in Fig. 5(a), the corresponding cycle $\mathcal{C}^+$ traverses $u_i^{(1)}, i = 1, 2, \ldots, 6$ sequentially and $\mathcal{C}^-$ traverses $u_i^{(2)}, i = 1, 2, \ldots, 6$ sequentially. (b) For the boundary shown in Fig. 5(b), the corresponding cycle $\mathcal{C}^+$ traverses $u_1^{(2)}, u_2^{(2)}, u_3^{(1)}, u_4^{(1)}, u_5^{(1)}, u_6^{(1)}$ sequentially and $\mathcal{C}^-$ traverses $u_1^{(1)}, u_2^{(1)}, u_3^{(2)}, u_4^{(2)}, u_5^{(2)}, u_6^{(2)}$ sequentially. Vertices $u_i^{(1)}$ and $u_i^{(2)}$ correspond to the axis-segment endpoint Q_i, $i = 1, 2, \ldots, 6$ in Fig. 5. Auxiliary edges are shown by blue dashed curves.

and locating the optimal boundary $\mathcal{B}$ that minimizes the cost (1) is reduced to finding an alternate cycle $\mathcal{C}$ in graph $G = (V, E)$ such that this cycle $\mathcal{C}$ minimizes the cycle ratio

$$\varphi(\mathcal{C}) = \frac{W_2(\mathcal{C})}{W_1(\mathcal{C})} = \frac{\sum_{e \in \mathcal{C}} w_2(e)}{\sum_{e \in \mathcal{C}} w_1(e)}. \tag{2}$$

The correctness of this reduction comes from the following two facts. First, any alternate cycle $\mathcal{C}$ has two mirror versions $\mathcal{C}^+$ and $\mathcal{C}^-$ with opposite signs on the total second weights. Without loss of generality, we assume $W_2(\mathcal{C}^+) = -W_2(\mathcal{C}^-) > 0$. Clearly, the cycle $\mathcal{C}$ that minimize the cost (2) must be of a $\mathcal{C}^-$ version and has a negative $W_2(\mathcal{C})$. The mirror of this optimal cycle $\mathcal{C}$ is then of a $\mathcal{C}^+$ version that maximizes $\frac{W_2(\mathcal{C})}{W_1(\mathcal{C})}$ and therefore, minimizes $\frac{W_1(\mathcal{C})}{W_2(\mathcal{C})}$ subject to $W_2(\mathcal{C}) > 0$. The ratio $\frac{W_1(\mathcal{C})}{W_2(\mathcal{C})}$ subject to $W_2(\mathcal{C}) > 0$ is exactly the same as $\phi(\mathcal{B})$, where $\texttt{area}(\mathcal{B})$ is always positive. Therefore, we have $\phi(\mathcal{B}) = -\frac{1}{\varphi(\mathcal{C})}$ and the alternate cycle $\mathcal{C}$ with the minimum cycle ratio $\varphi(\mathcal{C})$ corresponds to the boundary $\mathcal{B}$ with the minimum cost $\phi(\mathcal{B})$.

Second, we prove by contradiction that the resulting optimal alternate cycle $\mathcal{C}$ does not contain more than one auxiliary edge. Otherwise, the resulting boundary $\mathcal{B}$ would contain multiple separate closed boundaries with unaligned axes. Assume the resulting cycle $\mathcal{C}$ contains k auxiliary edges with $k > 1$. This means $\mathcal{C}$ contains k alternate paths $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_k$ after these k auxiliary edges are removed. From these k paths, we can construct k new alternate cycles $\mathcal{C}_1$, $\mathcal{C}_2$, ..., $\mathcal{C}_k$ by including the auxiliary edges between the two endpoints of each path. Given the cycle ratio defined in Eq. (2), it is not difficult to see

that $\varphi(\mathcal{C}) = \frac{W_2(\mathcal{C})}{W_1(\mathcal{C})} = \frac{\sum_{i=1}^{k} W_2(\mathcal{C}_i)}{\sum_{i=1}^{k} W_1(\mathcal{C}_i)}$. This shows that at least one cycle, say $\mathcal{C}_m$, out of $\mathcal{C}_i, i = 1, 2, \ldots, k$, has a smaller cycle ratio than $\mathcal{C}$, i.e., $\varphi(\mathcal{C}_m) = \frac{W_2(\mathcal{C}_m)}{W_1(\mathcal{C}_m)} \leq \varphi(\mathcal{C})$. This contradicts the assumption that $\mathcal{C}$ is a cycle with minimum cycle ratio.

Finally, we use an available graph algorithm to find an alternate cycle $\mathcal{C}$ that minimizes the cycle ratio (2) [37, 1]. This minimum-ratio alternate cycle algorithm finds the desired optimal cycle in polynomial time.

4 An Improved Gap-Length Measure $|P_iP_j|_D$

The main goal of introducing $|P_iP_j|_D$ is to handle the case where a gap-filling quadrilateral may coincide with detected segments. As shown in both Figs. 11(a) and (b), a gap-filling quadrilateral $P_2P_3P_6P_7$ is constructed to connect two trapezoids $\mathcal{T}_1$ and $\mathcal{T}_2$. The contribution of this quadrilateral to the term $|\mathcal{B}_D|$, if it is included in the boundary $\mathcal{B}$, is not $|P_2P_3| + |P_6P_7|$, as a portion of P_2P_3 is in fact coincident with detected segments, e.g., P_3P_9 in Fig. 11(a) and P_9P_{10} in Fig. 11(b). To get an accurate estimate of $|\mathcal{B}_D|$, we locate such coincident portions and deduct them from the calculation of the gap length.

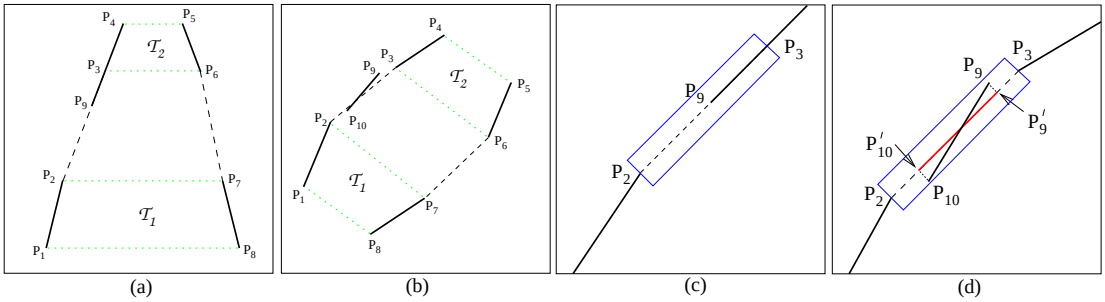


Figure 11: An illustration of measuring the gap length by excluding the projections from coincident detected segments: (a) & (b) two examples where the side P_2P_3 of the quadrilateral $P_2P_3P_6P_7$ is coincident with a detected segment, and (c) & (d) calculating $|P_2P_3|_D$ for the two examples shown in (a) and (b) respectively.

Specifically, in calculating the gap length between two points, say P_2 and P_3 in Figs. 11(c) and (d), we construct a rectangular box, shown in blue in Figs. 11(c) and (d), around the segment P_2P_3. All the detected segments, e.g., P_9P_{10} in Fig. 11(d), or the portions of the detected segments, e.g., P_9P_3 in Fig. 11(c), that are located inside of this rectangular box are projected onto the segment P_2P_3. We then only count the portions along P_2P_3 that do not overlap with any such detected-segment projections to calculate the gap-length between P_2 and P_3. For example, the gap length between P_2P_3 will be $|P_2P_3|_D = |P_2P_9|$

for the case shown in Fig. 11(c) and $|P_2P_3|_D = |P_2P'_{10}| + |P_3P'_9|$ for the case shown in Fig. 11(d). Using a small rectangular box for searching coincident detected segments improves the robustness by not requiring exact coincidence between the considered gap and the detected segments, as shown in Fig. 11(d). The only free parameter in this processing is the width of the rectangular box. In all our experiments, we set this width to be 2 pixels.

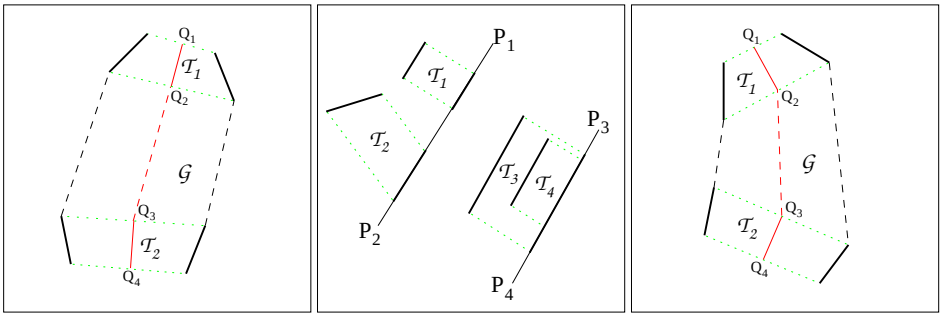
5 Experiments

We implemented the proposed method and tested its performance on synthetic data and on a set of real images[2]. For the synthetic data, we synthesize detected segments directly and all the detected segments are located within a square region of size 96×96. All the real images are scaled to be no larger than 250×250 while maintaining their aspect ratio. For all our experiments we set $\lambda = 10$ in the cost function (1). We will discuss the selection of this parameter later in Section 5.4.

In constructing the trapezoids, we pair (a) every two detected segments, and (b) every gap-filling segment with every detected segment, when they have overlapped projections on their angle-bisector line, as shown in Fig. 3. For the gap-filling segments, we consider no more than K shortest gap-filling segments incident from each detected-segment endpoints, where K is a preset constant number and we set $K = 5$ in our experiments. This way, we consider only $O(n)$ gap-filling segments out of all possible $O(n^2)$ ones, with n being the number of detected segments. This indicates that we may construct $O(n^2)$ trapezoids. If we construct quadrilaterals to fill the gaps between each pair of trapezoids, we may then have $O(n^4)$ quadrilaterals, which lead to $O(n^4)$ dashed edges in the constructed graph. To reduce the number of dashed edges, we develop three practical strategies to avoid constructing the quadrilaterals that are unlikely to be included in the desired optimal boundary.

The first strategy is to consider the construction of only one quadrilateral between each pair of trapezoids. As discussed above, we can construct four different gap-filling quadrilaterals to connect two trapezoids because the axis segment of a trapezoid has two endpoints. For example, to connect the two trapezoids shown in Fig. 12(a), the axis-segment of the constructed quadrilateral can be Q_1Q_3, Q_1Q_4, Q_2Q_3, or Q_2Q_4. Usually, only one of these four quadrilaterals is likely to be included in the desired optimal boundary. Based on the proximity preference, we only consider the quadrilateral that has the shortest axis segment out of the four choices. For the example shown in Fig. 12(a), we only consider one quadrilateral

[2]The software and images used in this section are available at `http://www.cse.sc.edu/~songwang/document/SRC.tgz`.



(a) (b) (c)

Figure 12: An illustration of the three strategies implemented to reduce the number of quadrilaterals.

with the axis segment Q_2Q_3 to connect the trapezoids $\mathcal{T}_1$ and $\mathcal{T}_2$. Furthermore, if the considered quadrilateral introduces a total gap length that is larger than a given threshold value, we simply do not construct it. In our experiments we set this threshold value to $D_1 = 30$ pixels. Each quadrilateral introduces two gaps described by its two sides across the two neighboring trapezoids, e.g., the gap length introduced by the quadrilateral $P_2P_3P_6P_7$ in Figs. 11(a) (or (b)) is $|P_2P_3|_D + |P_6P_7|_D$.

Our second strategy is to avoid constructing a quadrilateral to connect two trapezoids that share a same portion of a detected segment. For example, the two trapezoids $\mathcal{T}_3$ and $\mathcal{T}_4$ shown in Fig. 12(b) share a same portion of the detected segment P_3P_4. Therefore, no quadrilateral will be constructed between them since the resulting closed boundary should not traverse (a portion of) a line segment more than once. However, for two trapezoids $\mathcal{T}_1$ and $\mathcal{T}_2$ shown in Fig. 12(b), although both of them are constructed from the detected segment P_1P_2, we still consider constructing a quadrilateral to connect them since they use different portions of P_1P_2.

Our third strategy is to avoid constructing quadrilaterals that lead to an axis with low collinearity, because we are only interested in detecting symmetric boundaries. Specifically, for the two trapezoids shown in Fig. 12(c), we require that $|\sin(\angle Q_1Q_2Q_4)|$ and $|\sin(\angle Q_1Q_3Q_4)|$ are both less than a certain threshold for constructing a quadrilateral between them. These two terms are exactly the ones used in the grouping cost (1) for measuring the local boundary symmetry. In our experiments we set this threshold to $D_2 = 0.5$. Applying these three strategies can substantially reduce the number of dashed edges in the constructed graph.

Similarly, the number of the auxiliary edges would be $O(n^4)$ if we consider connecting every possible pair of the axis-segment endpoints. To reduce the number of auxiliary edges, we only consider the axis-segment endpoints that are likely to be boundary-axis endpoints for constructing auxiliary edges.

Specifically, we only consider the axis-segment endpoints around which the gap length is less than a given threshold. For example, in Fig. 5(a), we may consider Q_1 as a potential boundary-axis endpoint for constructing auxiliary edges because the gap length between P_1 and P_{12} is small. But we may not consider Q_5 because the gap length between P_5 and P_8 is larger than the given threshold. In our experiments we set this threshold to $D_3 = 20$. In addition, if both endpoints of a trapezoid axis segment satisfy this condition to be potential boundary-axis endpoints, we only consider the one with the smaller gap length. Note that the gap length is measured by the method introduced in Section 4.

Note that all these strategies are developed for speeding up the algorithm in practice. Without these three strategies, the number of constructed edges in the graph is still a polynomial function of n. Therefore, the complexity of the proposed grouping algorithm is still polynomial in terms of n. The robustness of the method to these thresholds is examined in section 5.4.

5.1 Experiments on Synthetic Data

To evaluate the proposed method quantitatively, we construct a set of synthetic data with a known ground truth of the desired symmetric boundary. Each synthetic data sample is constructed in the form of a set of detected segments, which come from two sources: a pair of synthetic boundaries (one desired symmetric boundary and one non-symmetric boundary) and random noise. Figure 13 shows the boundary pairs that are used for constructing synthetic data. The desired symmetric boundaries are shown in bold. We can see that the pair of boundaries in Fig.13 may or may not overlap each other since both cases may happen in practice. Particularly, Fig.13(e) simulate the case where a symmetric structure generates a non-symmetric shadow. In some cases, such as the ones shown in Fig.13(d), the desired symmetric boundary encloses a smaller area than the other non-symmetric boundary does. We intentionally design such cases to see whether the consideration of the boundary symmetry can help detect the desired symmetric boundary, even if the grouping cost prefers a boundary with a larger enclosed region area.

We sample the boundaries shown in Fig. 13 to construct disjoint detected segments with gaps. Specifically, we uniformly subdivide each boundary into a set of line segments of equal length (5% of the perimeter) and then randomly remove a certain number of these line segments. The remaining ones are then included as detected segments in constructing a synthetic data sample. We further add randomly generated detected segments to simulate the image noise and for simplicity, we call them *noise segments*. For these noise segments, their directions conform to a uniform distribution over all possible directions,

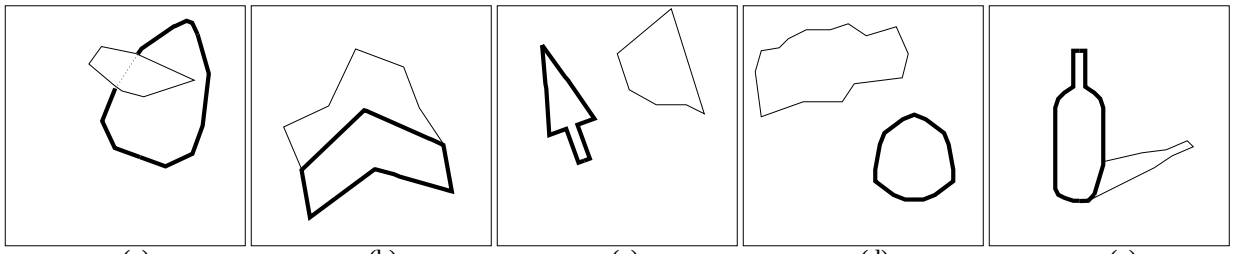


Figure 13: Five boundary pairs used for constructing synthetic data. Desired symmetric boundaries are shown in bold.

their locations conform to a uniform distribution within a square region of size 96×96, and their lengths are uniformly distributed within the range of $[5, 15]$ pixels. Figure 14(a) shows an example of adding 40 noise segments to the boundary pair shown in Fig. 13(a). Figure 14(b) shows one synthetic data sample, which consists of the detected segments sampled from the boundary pair shown in Fig. 13(a) and the noise segments shown in Fig. 14(a). Particularly, we remove 30% of the subdivided line segments along the boundary pair shown in Fig. 13(a) to construct the synthetic data sample in Fig. 14(b). Based on such a synthetic data sample, we can directly apply the proposed grouping method to detect an optimal boundary, as shown in Fig. 14(c).

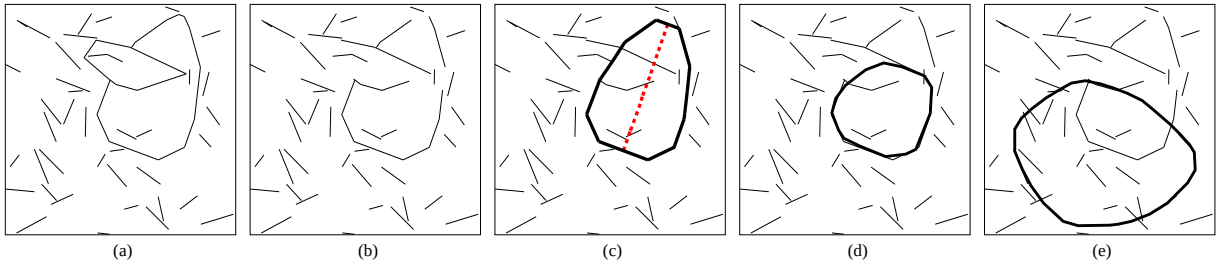


Figure 14: An illustration of a synthetic data sample. (a) A boundary pair and random noise segments. (b) Synthesized detected segments by combining the ones from the boundary pairs and the noise segments. (c) Optimal boundary detected by the proposed grouping method, with symmetry axis shown in red. (d) Optimal boundary detected by RC. (e) Optimal boundary detected by EZ.

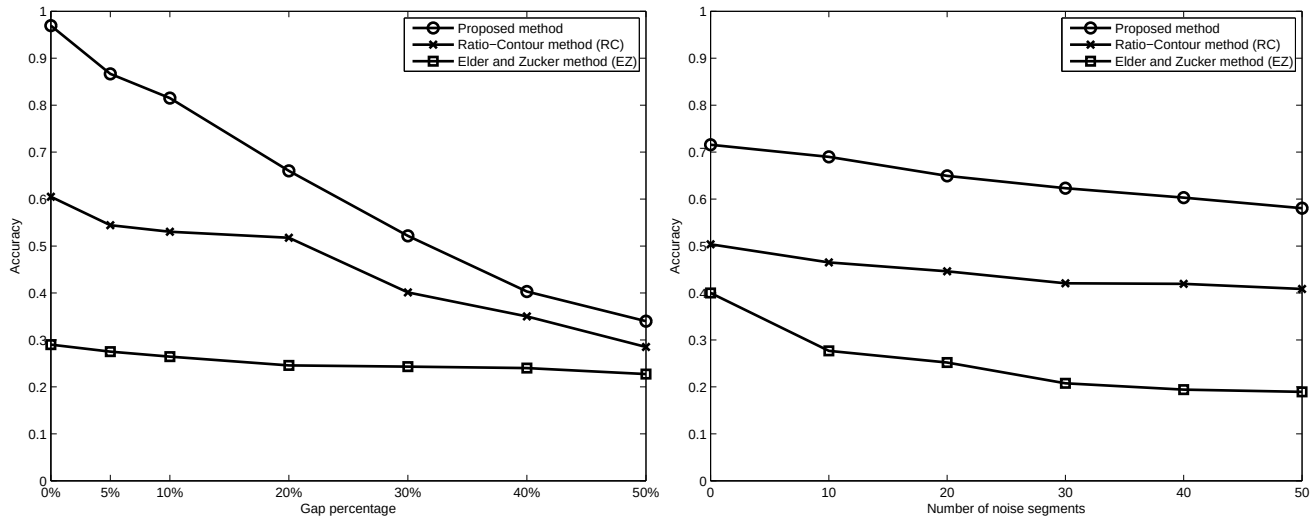
For each of the five boundary pairs shown in Fig. 13, we sample them in seven different ways so that the resulting gaps along the boundaries account for 0%, 5%, 10%, 20%, 30% , 40% and 50% of the boundary length, respectively. The number of noise segments are also chosen to be 0, 10, 20, 30, 40 and 50 in different settings. By doing all possible combination of the sampled boundary pairs and the noise segments, we have a total number of $5 \times 6 \times 7 = 210$ different settings. Under each setting, we further randomly generate the expected number of noise segments and the expected gap percentage along the

underlying boundary pair 10 times, which finally results in $210 \times 10 = 2100$ synthetic data samples. In the experiments, we apply the proposed grouping method to all these 2100 data samples. For comparison, we also apply two previous edge-grouping methods: the ratio-contour method (RC) [37] and the Elder and Zucker method (EZ) [9] to the same 2100 synthetic data. The implementation of both RC and EZ are the same as in [37]. Note that both RC and EZ consider only boundary closure, proximity, and continuity (smoothness), but not boundary symmetry and enclosed region area, in grouping. Sample grouping results by RC and EZ are shown in Fig 14(d) and (e) respectively.

As in [37], on each data sample, we define the grouping performance using a region coincidence measure $\frac{|R \cap R'|}{|R \cup R'|}$, where R and R' are the regions enclosed by the desired ground-truth boundary and the detected boundary respectively, and $|R|$ indicates the area of region R. The larger this measure, the better the coincidence between the ground truth boundary and the detected boundary. For example, the grouping results shown in Figs. 14(c), (d) and (e) have a performance of 0.98, 0.68 and 0.19 respectively. Figures 15(a) and 15(b) show the average performance of the proposed method, RC, and EZ on all 2100 data samples, in terms of the gap-percentage along the boundary pairs and the number of noise segments, respectively. Note that, for each setting of the gap percentage along the boundary pair in Fig. 15(a), the average is taken over 300 data samples since we have 5 boundary pairs, 6 levels of the number of noise segments, and 10 rounds of noise-segment and boundary-gap generation. Similarly, for each setting of the number of noise segments in Fig. 15(b), the average is taken over 350 data samples since we have 5 boundary pairs, 7 levels of the gap percentage in sampling the boundary pair, and 10 rounds of noise-segment and boundary-gap generation. From Fig. 15, we can see that when the desired boundaries are symmetric, the proposed grouping method performs better than RC and EZ, where boundary symmetry is not considered.

5.2 Experiments on Real Images

We also test the proposed grouping method on real images and compare it to RC and EZ. The test real images are selected from the Corel image database and Google image search. On real images, we construct the detected segments by edge detection and line fitting. For edge detection we use the Canny detector provided with Matlab's Image Processing Toolbox (R2006a), leaving its parameters at their default values. For line fitting we used the Matlab function developed by Kovesi [17] by setting the minimum length for an edge to be considered to 30 pixels and the maximum deviation between an edge and its fitting line to 2 pixels. Sample edge detection and line-fitting results for 20 real images are shown in the second and third



(a) (b)

Figure 15: The average performance of the proposed method, RC, and EZ on all 2100 synthetic data samples, in terms of (a) the gap percentage along the synthetic boundary pairs, and (b) the number of noise segments.

columns of Figs. 16 and 17 respectively. Based on these detected segments, all the other settings for the proposed grouping method are the same as the ones used in the above synthetic-data experiments.

The fourth, fifth and sixth columns of Figs. 16 and 17 show the optimal boundaries obtained by the proposed grouping method, RC and EZ respectively. From Figs. 16(a-d,f,i,j) and 17(a,c,d,f,h-j), we can see that, by considering the symmetry cue, the proposed method can detect boundaries with good symmetry. These experiments on real images also show that, when the whole structure is not symmetric, the proposed method may only detect a symmetric component of the structure, as shown in Figs. 16(c,e) and 17(a,b,e,f). In such cases, the edge-grouping methods without considering symmetry may detect the structural boundary more completely, as shown in Fig. 17(b). On several images, such as the ones shown in Figs. 16(a,e) and 17(e,g), RC or EZ produce similar symmetric boundaries as the proposed method does. This indicates that the symmetric boundaries found in these several images by the proposed method also minimize the RC or EZ grouping costs, which consider closure, proximity and continuity, but not symmetry.

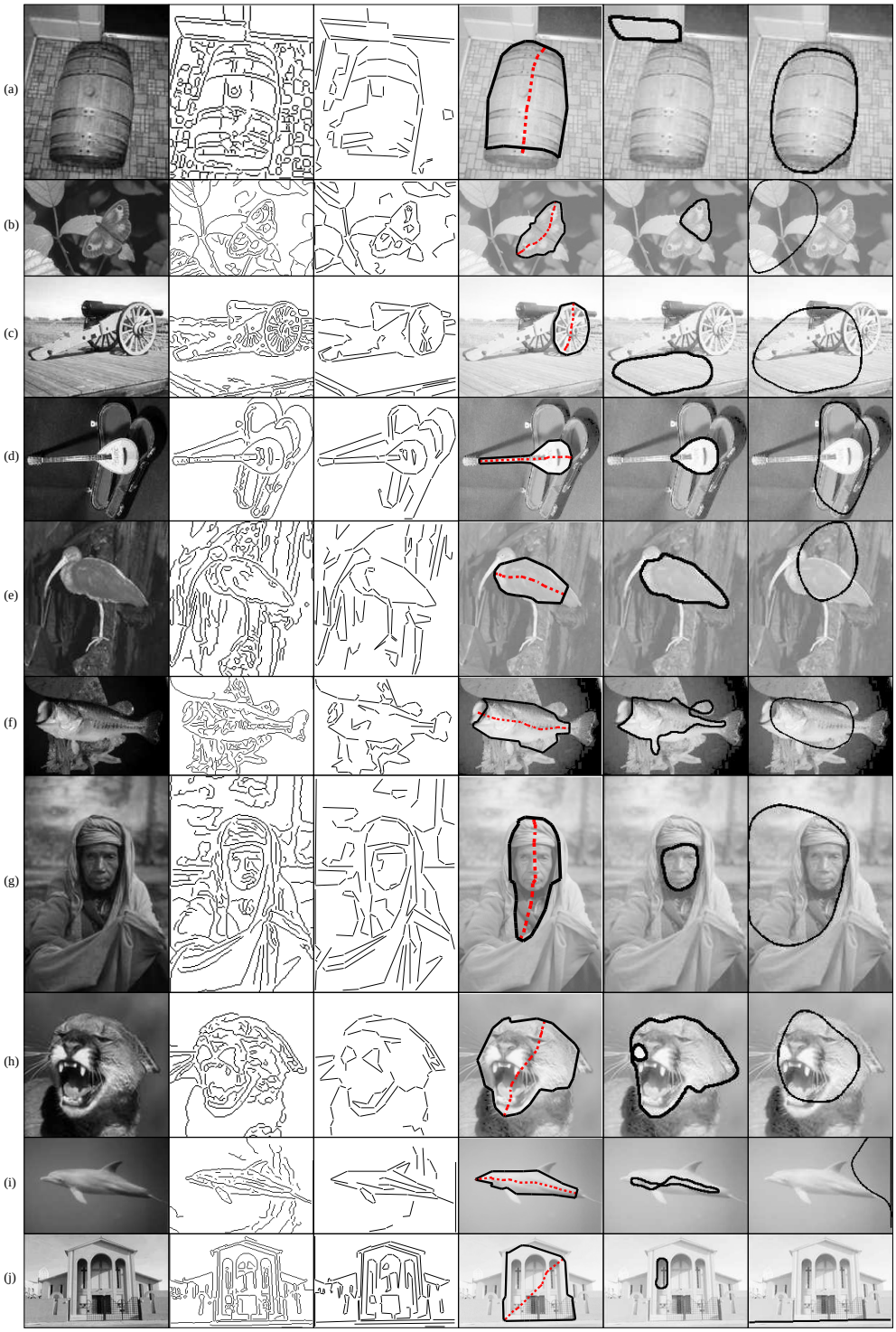


Figure 16: Grouping results on 10 real images. Column 1: input real images; Column 2: Canny edge detection results; Column 3: detected segments after line fitting; Column 4: the optimal boundary obtained by the proposed grouping method. Detected boundary axes are shown by red dashed curves; Column 5: the optimal boundary obtained by RC; and Column 6: the optimal boundary obtained by EZ.

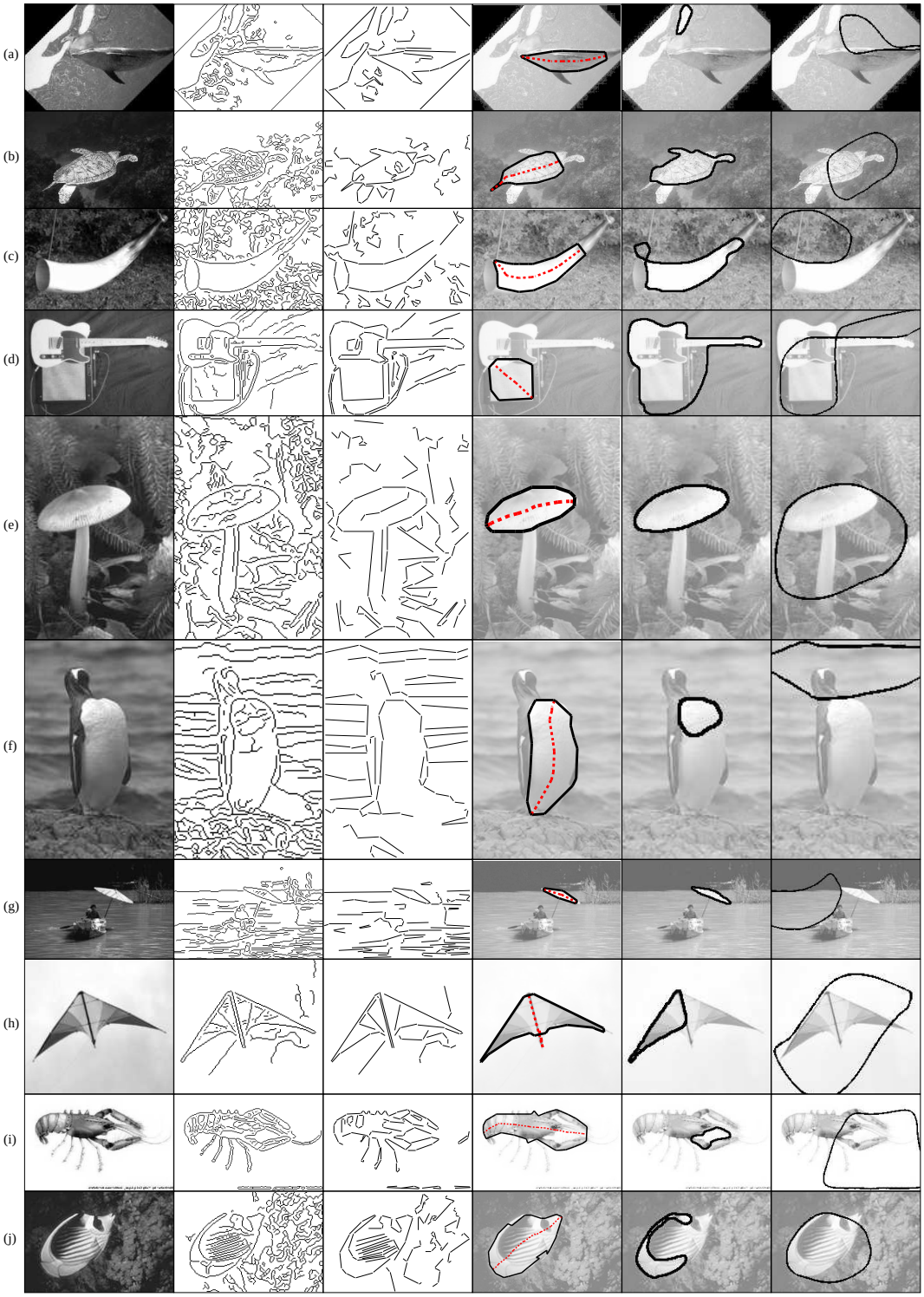


Figure 17: Grouping results on 10 more real images. The columns depict the same information as in Fig. 16.

5.3 Detecting Multiple Boundaries

In the previous sections, the proposed grouping method is developed and tested to detect the single optimal boundary that minimizes the grouping cost. In practice, most real images contain multiple structures of interest and there is a strong motivation to extend the proposed grouping method to detect multiple symmetric boundaries from the same image. In addition, due to noise, texture and other undesirable structures in real images, the optimal boundary that minimizes the grouping cost may not be the desired one. A more reasonable way is to detect a small number of optimal boundaries with small grouping cost and expect that the desired ones are among them.

We extend the grouping method to detect multiple optimal boundaries by repeating the proposed grouping method on the same image: after detecting the first optimal boundary, we remove all the trapezoids along the detected boundary and then repeat the same grouping method on the remaining trapezoids to detect the second optimal boundary. This process can be further repeated to detect multiple symmetric boundaries. The implementation of this strategy is simple: each trapezoid corresponds to two edges, and we only need to delete the edges present in the detected optimal cycle (and its mirror cycle) and re-run the minimum-ratio alternate cycle algorithm on the remaining graph to detect the next optimal boundary.

One problem of this multiple-boundary detection is that the same boundary may be detected in different rounds when repeating the proposed grouping algorithm. An example is shown in Fig. 18: the first round of grouping may produce a boundary consisting of the trapezoids $\mathcal{T}_1$ and $\mathcal{T}_2$. After removing $\mathcal{T}_1$ and $\mathcal{T}_2$, the second round of grouping may produce a boundary consisting of the trapezoids $\mathcal{T}_3$ and $\mathcal{T}_4$. We can see that the boundaries produced in these two rounds are in fact the same boundary. However, this mainly happens for a boundary with multiple different symmetry axes, as shown in Figs. 18(b) and (c). In practice, this is not a serious problem since we can easily check the detected multiple boundaries and for the boundaries that are detected more than once, we only keep one of them and discard the redundant ones.

We conducted experiments on real images by detecting the first three optimal boundaries, as shown in columns four, five, and six of Fig. 19. We can see that, the optimal boundaries detected in the second or third rounds may be more desirable than the one detected in the first round. In Fig. 20, we further show several examples where the proposed grouping method may detect different symmetric boundaries in different rounds. It also shows an example where the proposed grouping method may detect the same symmetric boundary in different rounds, as shown in columns $3, 4$ and 5 of Fig. 20(a).

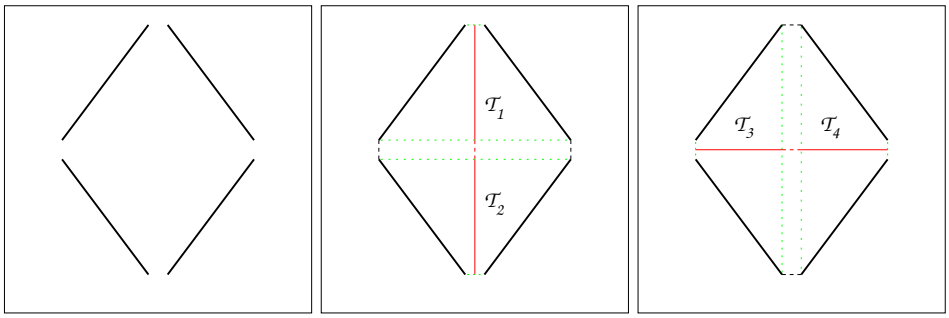


Figure 18: An illustration of detecting the same boundary when repeating the proposed grouping method. (a) A set of detected segments. (b) A symmetric boundary consisting of two trapezoids $\mathcal{T}_1$ and $\mathcal{T}_2$. (c) The same symmetric boundary consisting of two different trapezoids $\mathcal{T}_3$ and $\mathcal{T}_4$.

Note that we do not remove all the detected/gap-filling segments along the detected boundary and then repeat the same grouping method on the remaining line segments to detect multiple boundaries. The reason is that, removing one segment may correspond to removing many trapezoids, given that the same segment may be used to construct different trapezoids, and this may introduce two problems: (a) for an image with two neighboring structures that share a portion of the their boundaries, the detection of one may prevent the detection of the other, since the segments corresponding to the shared boundary may be removed after the detection of the first boundary, and (b) if the boundaries detected in the previous rounds are not the desired ones but include some segments along the desired boundaries, the removal of such segments may prevent the detection of the desired symmetric boundaries in the later rounds.

5.4 The Selection of λ and Other Thresholds

The proposed grouping cost function (1) has a free parameter λ, which needs to be selected by the user. This parameter balances the boundary proximity and the boundary symmetry in grouping. A larger λ may lead to more symmetric boundaries with poorer proximity and a smaller λ may lead to less symmetric boundaries with better proximity. In most of our experiments, we found that, the same or similar grouping results are obtained when λ takes a value in a certain range. Figure 21 shows an example of the proposed grouping with different values of λ. For this image, when λ takes a value in the range $[1, 50]$, the detected boundaries are very similar and well aligned with the barrel present in this image. In general, the selection of λ is related to the image size, since the proximity term $|\mathcal{B}_D|$ is related to image size and the symmetric term $\rho(\texttt{axis}(\mathcal{B}))$ is not. As mentioned earlier, we scale the images to be no larger than 250×250 while maintaining their aspect ratio in all our experiments on real images. We empirically select $\lambda = 10$ for all

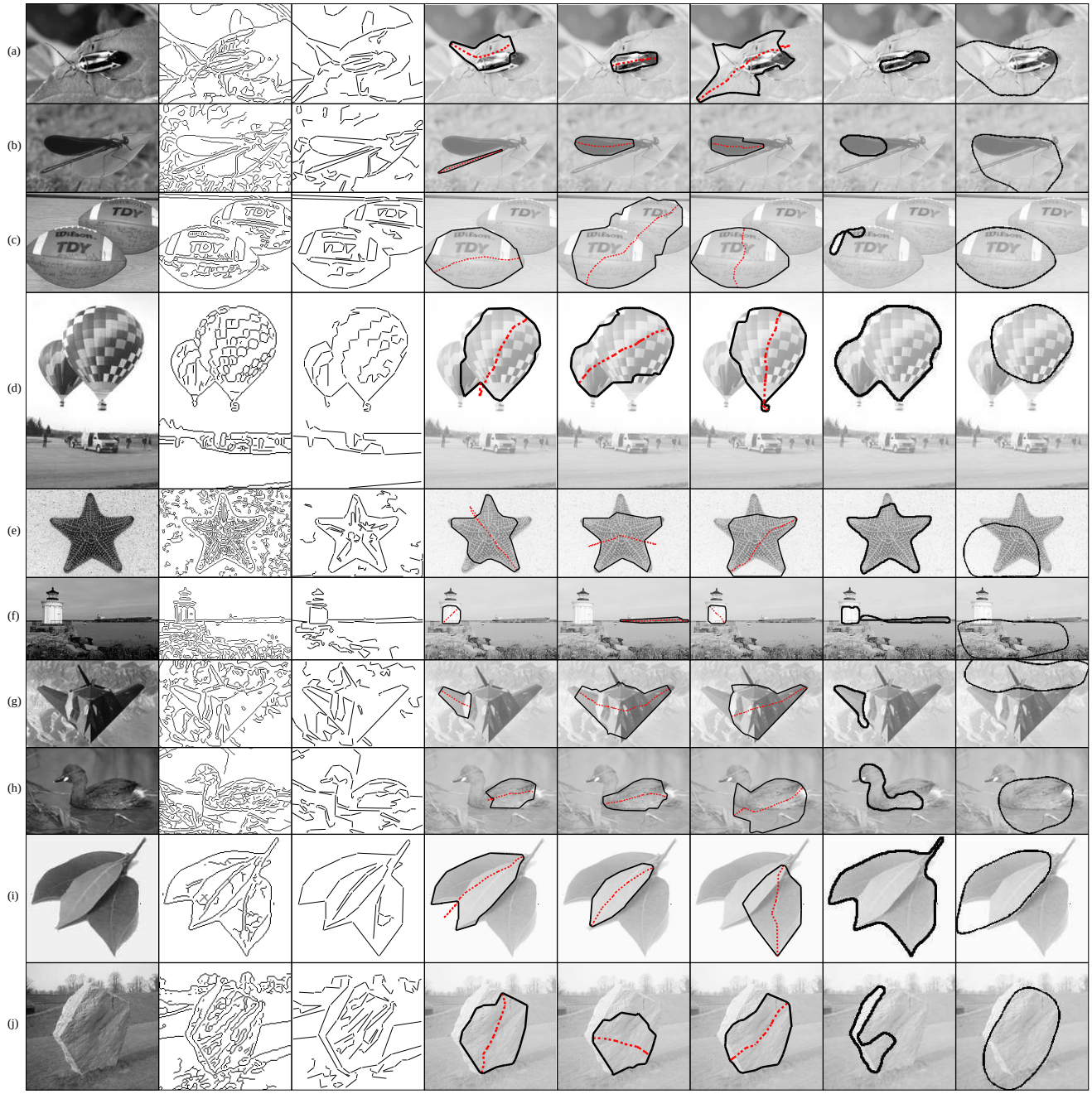


Figure 19: Experiment results of detecting multiple optimal boundaries from 10 real images. Column 1: input real images; Column 2: Canny edge detection results; Column 3: detected segments after line fitting; Column 4, 5 and 6: the first three optimal boundaries by repeating the proposed grouping method; Column 7: the optimal boundary obtained by RC; and Column 8: the optimal boundary obtained by EZ.

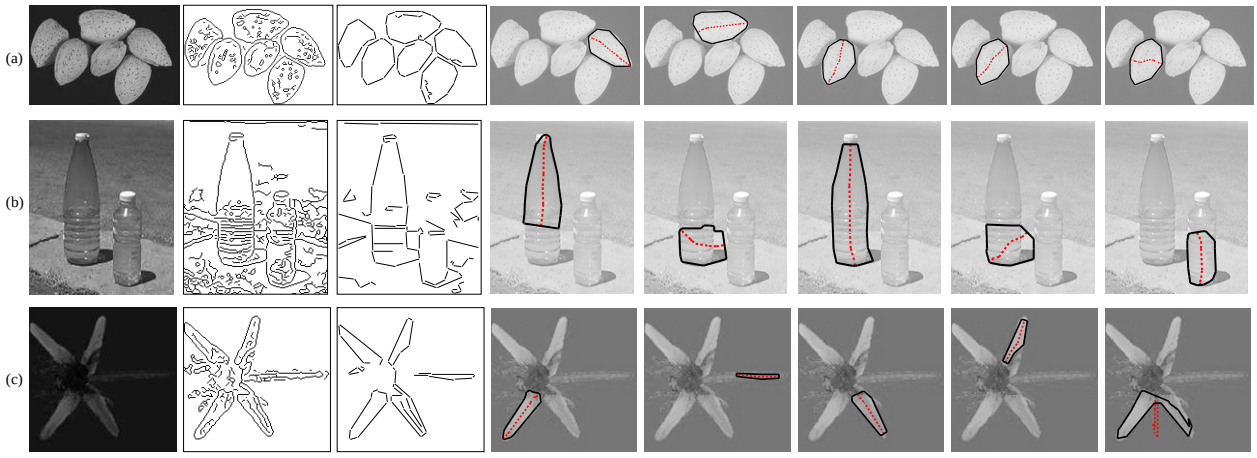


Figure 20: An illustration of detecting different boundaries or the same boundary in different rounds when repeating the proposed grouping method. From left to right, Column 1: input real images; Column 2: Canny edge detection results; Column 3: detected segments after line fitting; Columns 4-8: the first five optimal boundaries by repeating the proposed grouping method.

our experiments.

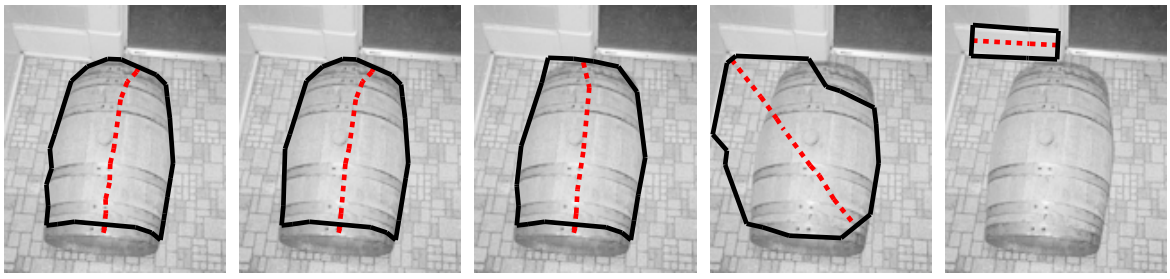


Figure 21: An illustration of the optimal boundaries detected by the proposed method using different values of λ. (a) $\lambda = 1$ or 5, (b) $\lambda = 10$ or 25, (c) $\lambda = 50$ or 75, (d) $\lambda = 100$, and (e) $\lambda = 1000$. The input image and the detected segments are the same as the ones shown in Fig. 16(a).

At the beginning of Section 5, we also introduce several thresholds to reduce the constructed graph size. Particularly, we choose $D_1 = 30$, $D_2 = 0.5$ and $D_3 = 20$ in our experiments. Using these thresholds, we can avoid constructing the gap-filling quadrilaterals and boundary-axis endpoints that are unlikely to be included in the optimal boundary with the minimum grouping cost. Figure 22 shows the grouping results by varying the value of these thresholds. By choosing larger values for D_1, D_2 and D_3, we construct more gap-filling quadrilaterals and consider more potential boundary-axis endpoints. However, the grouping results largely keep unchanged when we choose them to be larger than the ones used in our experiments. In fact, by turning off all three thresholds, i.e., setting $D_1 = +\infty$, $D_2 = 1.0$ and $D_3 = +\infty$, we get the

same grouping result as shown in the first column of Fig. 22. However, when turning off all of them, the constructed graph has $3,384,762$ edges and the proposed grouping takes 678.69 seconds, compared with $87,292$ edges and 3.95 seconds when setting $D_1 = 30$, $D_2 = 0.5$ and $D_3 = 20$.

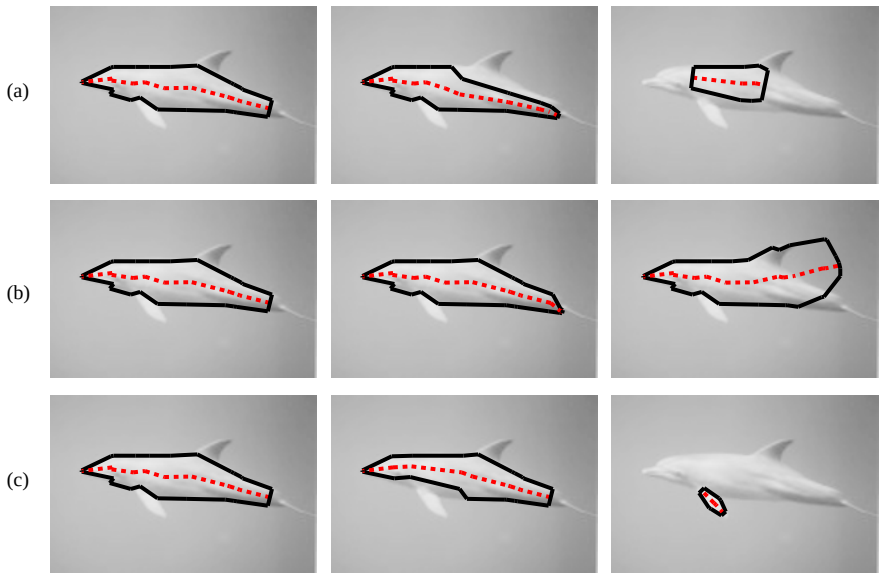


Figure 22: An illustration of the optimal boundaries detected by the proposed method using different values of D_1, D_2 and D_3. (a) Fixing $D_2 = 0.5$, $D_3 = 20$ and varying $D_1 = 25, 30, 40, 50, 100, 150, 200, 250$ or $+\infty$ (left), $D_1 = 15$ or 20 (middle) and $D_1 = 5$ or 10 (right). (b) Fixing $D_1 = 30$, $D_2 = 0.5$ and varying $D_3 = 10, 20, 30, 40, 50, 100, 150$ or $+\infty$ (left), $D_3 = 1, 3$ or 5 (middle) and $D_3 = 0$ (right). (c) Fixing $D_1 = 30$, $D_3 = 20$ and varying $D_2 = 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$ or 1.0 (left), $D_2 = 0.2$ or 0.3 (middle) and $D_2 = 0.1$ (right). The input image and the detected segments are the same as the ones shown in Fig. 16(i).

5.5 Two Special Cases

In Section 3.2, we proved that the minimum-ratio alternate cycle contains no more than one auxiliary edge. However, it is possible that the minimum-ratio alternate cycle does not contain any auxiliary edges. A minimum-ratio alternate cycle without an auxiliary edge may correspond to two special kinds of grouping results that are produced occasionally in practice. The first kind of grouping result consists of two disjoint closed boundaries that are symmetric to each other over an axis in between, as shown in Figs. 23(a-c). Specifically, from the detected segments shown in Fig. 23(a), we construct a set of trapezoids, three of which are shown in Fig. 23(b). One possible grouping result is to connect these three trapezoids to form two disjoint closed boundaries as shown by black curves in Fig. 23(c), where the red polygon with solid/dashed lines is the resulting symmetry axis (shown a little misaligned to visualize it

better). Along this symmetry axis, red solid lines represent the axis segments of the trapezoids and red dashed lines represent the axis segments of the gap-filling quadrilaterals. We can see that the alternate cycle corresponding to this grouping result contains no auxiliary edges, i.e., no boundary-axis endpoints can be identified. However, such a grouping result is more likely to have a relatively smaller $\texttt{area}(\mathcal{B})$ (the total area enclosed by the two resulting disjoint closed boundaries) and a relatively larger $|\mathcal{B}_D|$ (because of the longer total boundary perimeters). Therefore, such a grouping usually has a larger grouping cost and does not happen frequently in the proposed grouping. In Fig. 24, we show an example of such a special grouping result on a real image. This is produced in the 2nd round of repeating the proposed grouping method on this image. In practice, however, we can achieve more such grouping results by not constructing any auxiliary edges in the graph $G = (V, E)$. The application of the same graph algorithm will be forced to detect such symmetric boundary pairs. In practice, this may extend the proposed method to detect the objects in pairs, such as eyes, eyeglasses, windows, and so on.

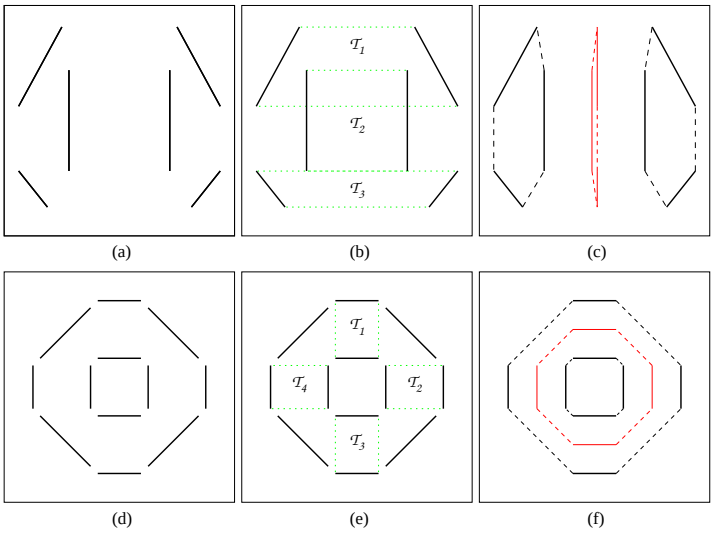


Figure 23: An illustration of two special cases in the proposed grouping. Top row: the first special case of detecting a symmetric pair of boundaries. Bottom row: the second special case of detecting two disjoint boundaries that form a ring.

The second kind of special grouping result consists of two disjoint closed boundaries that form a ring, as shown in the second row of Fig. 23. Based on the detected segments shown in Fig. 23(d), we can construct trapezoids as shown in Fig. 23(e). One possible grouping result is to connect these four trapezoids to form two disjoint closed boundaries as shown by black curves in Fig. 23(f), where the red polygon with solid/dashed lines is the resulting symmetry axis. In this case, the enclosed region area $\texttt{area}(\mathcal{B})$ is the

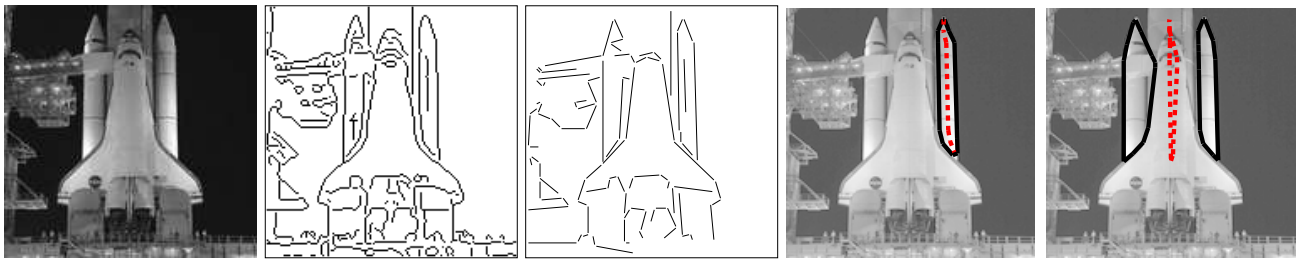


Figure 24: An example of the special grouping result on a real image. Column 1: original image; Column 2: Canny edge detection result; Column 3: detected segments after line fitting; Columns 4-5: the first and second optimal boundaries obtained by repeating the proposed grouping method.

one bounded by these two disjoint closed boundaries. This special kind of grouping result also occurs very rarely in practice since the enclosed region area is relatively small and the collinearity of the resulting symmetry axis is usually poor, which results in a large grouping cost. In all our experiments, so far we did not encounter any such special grouping results.

6 Extensions to Incorporate Other Boundary and Region Information

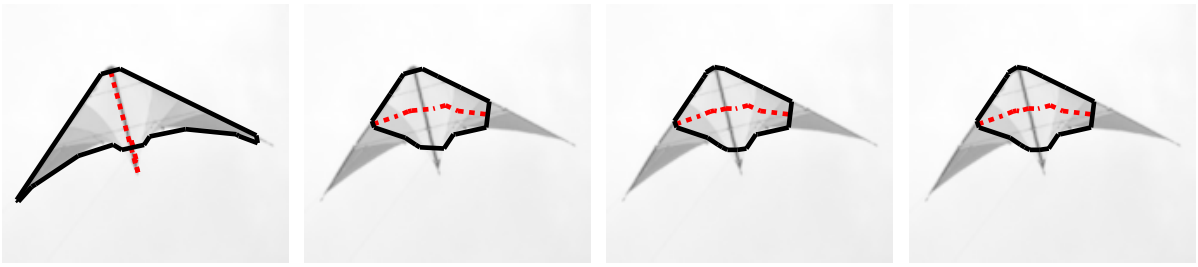
The proposed grouping method can be extended to incorporate other boundary or region information. For example, boundary continuity, or smoothness, is widely used in previous edge grouping methods [9, 39, 37], and it can be incorporated in the proposed grouping by adding another term into the numerator of the grouping cost (1) as

$$\phi(\mathcal{B}) = \frac{|\mathcal{B}_D| + \lambda \cdot \rho(\texttt{axis}(\mathcal{B})) + \lambda_2 \cdot \int_{\mathcal{B}} \kappa^2(t)dt}{\texttt{area}(\mathcal{B})}, \tag{3}$$

where $\kappa(t)$ is the curvature along the boundary $\mathcal{B}$ and λ_2 is a weighting factor for this continuity term. Since all the considered boundaries are polygons in this paper, cubic spline interpolations are used to estimate the boundary curvature [34].

By making a small change on the first edge weight function, the proposed graph modeling and algorithm can still be used to find the optimal boundary that minimizes new grouping cost [34]. However, we found that, while the boundary continuity is important in previous edge grouping methods, it is not critical in the proposed grouping for symmetric boundaries. The reasons are two fold: (a) with a normalization over the enclosed region area in the grouping cost, the proposed method has a preference to produce boundaries that enclose a large round area. This preference implicitly reflects some level of continuity, and (b) as many symmetric boundaries are not smooth everywhere, an explicit inclusion of a curvature term in the

grouping cost (3) may affect the detection of such symmetric boundaries. Figure 25 shows an example where the inclusion of the curvature-based continuity term prevents the detection of a symmetric boundary with high-curvature points.



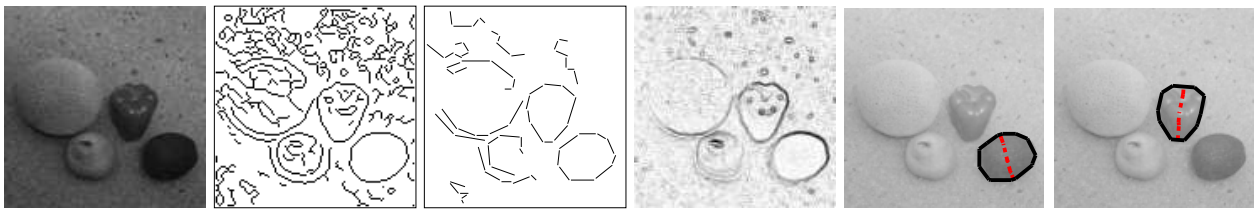
(a) (b) (c) (d)

Figure 25: An example of the grouping by using the modified grouping cost (3): (a) the optimal boundary obtained by the proposed method based on the original grouping cost (1), (b-d) the optimal boundaries obtained by the proposed method based on the modified grouping cost (3) with $\lambda_2 = 0.5, 1$ and 5, respectively. The input image and the detected segments are the same as the ones shown in Fig. 17(h).

Other region information can also be incorporated by modifying the denominator of the grouping cost (1). For example, we can extend the proposed method to detect a boundary that encloses a region with good intensity homogeneity by modifying the grouping cost to

$$\phi(\mathcal{B}) = \frac{|\mathcal{B}_D| + \lambda \cdot \rho(\texttt{axis}(\mathcal{B}))}{\iint_{R(\mathcal{B})}(1 - |\nabla I(x,y)|)dxdy}, \tag{4}$$

where $R(\mathcal{B})$ is the region enclosed by the boundary $\mathcal{B}$. Normalized to the range $[0, 1]$, $|\nabla I(x, y)|$ is the magnitude of the intensity gradient at pixel (x, y). To find the optimal boundary that minimizes this modified grouping cost, we only need to modify the definition of the second edge weight in the graph construction from the signed enclosed area $\iint_{R(\mathcal{T})} dxdy$ (or $\iint_{R(\mathcal{G})} dxdy$) to the signed value of $\iint_{R(\mathcal{T})}(1 - |\nabla I(x, y)|)dxdy$ (or $\iint_{R(\mathcal{G})}(1 - |\nabla I(x, y)|)dxdy$), where $R(\mathcal{T})$ (or $R(\mathcal{G})$) is the region enclosed by the trapezoid $\mathcal{T}$ (or the quadrilateral $\mathcal{G}$) corresponding to the considered edge. The same graph modeling and algorithm can then be used to find the optimal boundary. Figure 26(e) shows an example of the grouping using the modified grouping cost (4). As a comparison, Figure 26(f) show the grouping result on the same image using the grouping cost (1). We can see that, with this extension, the proposed method detects a boundary whose enclosed region has better intensity homogeneity.



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

Figure 26: An example of grouping by incorporating intensity homogeneity: (a) input real image, (b) Canny edge detection result, (c) detected segments after line fitting, (d) the (normalized) magnitude of the image intensity gradient, where darker pixels indicate larger gradient magnitudes, (e) the optimal boundary obtained by the proposed grouping method based on the modified grouping cost (4), and (f) the optimal boundary obtained by the proposed method based on the original grouping cost (1).

7 Complexity Analysis and Running Time

As discussed in Section 5, given n detected segments, we may need to construct $O(n^2)$ trapezoids in the worst case. If we construct quadrilaterals to fill the gaps between each pair of trapezoids, we may then have $O(n^4)$ quadrilaterals, which lead to $O(n^4)$ dashed edges and $O(n^4)$ auxiliary edges in the constructed graph. The minimum-ratio alternate cycle algorithm has a time complexity of $O(|E|^{\frac{7}{4}})$, where $|E|$ is the number of edges in the graph [37], in the worst case. The worst-case complexity of the proposed grouping algorithm is then $O(n^7)$. This is a very high complexity and an algorithm with such a complexity is not usually useful in practice.

However, two reasons make the proposed grouping method still practically useful: (a) n, the number of detected segments, is usually much smaller than the number of pixels in an image, and (b) we developed several special strategies to substantially reduce the number of edges in the constructed graph (see Section 5). While it may be difficult to analytically derive a tighter estimate of the algorithm complexity, we implement the proposed grouping method using C++ and check its running time in processing a set of real images. Table 1 shows the running time on the 20 images shown in Figs. 16 and 17 and the size of the graph constructed for them. Our experiments were run on Linux computers equipped with a 3.4GHz Xeon processor and 4GB RAM.

8 Conclusions

In this paper, we developed a new grouping method for detecting closed boundaries that show good bilateral symmetry. Particularly, the proposed method can detect both boundaries and their symmetry

	Size	Lines	Trapezoids	Quads.	Auxiliary Edges	Total Edges	Prop. Method CPU(sec)	RC CPU(sec)
Fig. 16(a)	124×150	335	10400	251154	343620	605174	416.55	29.77
Fig. 16(b)	218×145	580	21620	523218	843700	1388538	1134.85	33.08
Fig. 16(c)	179×150	353	11124	256668	292612	560404	325.22	28.53
Fig. 16(d)	176×150	309	9282	229228	286524	525034	448.60	13.29
Fig. 16(e)	150×161	456	17256	588778	721200	1327234	2033.32	37.53
Fig. 16(f)	220×150	460	14270	245566	435244	695080	545.48	27.01
Fig. 16(g)	150×225	559	25838	699136	1307344	2032318	2468.40	68.11
Fig. 16(h)	150×150	290	7672	132560	135720	275952	73.85	16.85
Fig. 16(i)	200×133	161	3638	48010	35644	87292	3.95	14.04
Fig. 16(j)	231×150	512	25912	899050	1547040	2472002	2192.04	57.44
Fig. 17(a)	210×150	314	10210	200366	205440	416016	100.97	16.82
Fig. 17(b)	230×150	422	11574	223262	346944	581780	81.64	24.00
Fig. 17(c)	200×136	538	16964	321902	509040	847906	698.81	29.64
Fig. 17(d)	212×150	397	14102	362208	459840	836150	863.27	25.62
Fig. 17(e)	150×225	547	19718	340776	505012	865506	787.04	54.69
Fig. 17(f)	123×181	334	10274	166230	200344	376848	246.79	30.81
Fig. 17(g)	225×150	404	14324	416190	424120	854634	1740.18	53.79
Fig. 17(h)	166×150	143	3228	58618	62304	124150	6.47	12.45
Fig. 17(i)	231×150	412	12710	307544	448404	768658	525.75	20.04
Fig. 17(j)	184×125	390	13066	472338	619384	1104788	1433.91	18.56

Table 1: Running time and the constructed graph size for the 20 images shown in Figs. 16 and 17.

axes. This is achieved by (a) defining a new grouping cost that combines the different boundary and region information, (b) constructing a new type of trapezoidal grouping tokens by pairing line segments detected from the input image, and (c) constructing a new graph model that can transform the proposed grouping problem into a graph problem of detecting a cycle that minimizes a given ratio-form cost. We show that this graph problem can be addressed by an available graph algorithm with polynomial-time complexity. We implemented the proposed grouping method and tested its performance on a set of synthetic data and real images. We also conducted experiments to compare its performance to the performance of two previous edge-grouping methods. These experiments showed that the proposed method performs more favorably when the desired structure has a boundary with good bilateral symmetry.

Acknowledgements

This work was funded, in part, by NSF-EIA-0312861. Part of the experiments were run on the "NICK" Dell PC cluster provided by the College of Engineering and Computing at the University of South Car-

olina. Some preliminary results of this paper were published in a conference [34].

References

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

[2] T. Alter and R. Basri. Extracting salient contours from images: An analysis of the saliency network. In *International Journal of Computer Vision*, pages 51–69, 1998.

[3] A. Amir and M. Lindenbaum. A generic grouping algorithm and its quantitative analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):168–185, 1998.

[4] H. Blum. Biological shape and visual science. *Journal of Theoretical Biology*, 38:205–287, 1973.

[5] H. Blum and R. N. Nagle. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10:167–180, 1978.

[6] M. J. Brady and H. Asada. Smoothed local symmetries and their implementation. Technical Report AIM-757, Massachusetts Institute of Technology, February 1984.

[7] H. Cornelius and G. Loy. Detecting bilateral symmetry in perspective. In *5th Workshop on Perceptual Organization in Computer Vision (POCV)*, pages 191–198, 2006.

[8] J. H. Elder, A. Krupnik, and L. A. Johnston. Contour grouping with prior models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):661–674, 2003.

[9] J. H. Elder and S. W. Zucker. Computing contour closure. In *European Conference on Computer Vision*, pages 399–412, 1996.

[10] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2003.

[11] A. Gupta, V. Prasad, and L. Davis. Extracting regions of symmetry. In *IEEE International Conference on Image Processing*, volume 3, pages 133–136, 2005.

[12] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *International Journal of Computer Vision*, 20(1):113–133, 1996.

[13] H. J. A. M. Heijmans and A. V. Tuzikov. Similarity and symmetry measures for convex shapes using Minkowski addition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):980–993, 1998.

[14] D. Huttenlocher and P. Wayner. Finding convex edge groupings in an image. *International Journal of Computer Vision*, 8(1):7–29, 1992.

[15] D. Jacobs. Robust and efficient detection of convex groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):23–27, 1996.

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

[17] P. D. Kovesi. Matlab functions for computer vision and image analysis. School of Computer Science & Software Engineering, The University of Western Australia. http://www.csse.uwa.edu.au/∼pk/research/matlabfns/.

[18] S. Lazebnik, C. Schmid, and J. Ponce. Semi-local affine parts for object recognition. In *British Machine Vision Conference*, volume 2, pages 959–968, 2004.

[19] M. Leyton. *Symmetry, Causality, Mind*. MIT Press, Cambridge, 1992.

[20] T. Liu, D. Geiger, and A. L. Yuille. Segmenting by seeking the symmetry axis. In *International Conference on Pattern Recognition*, pages 994–998, 1998.

[21] D. G. Lowe. *Perceptual Organization and Visual Recognition*. Boston: Kluwer Academic Publishers, 1985.

[22] G. Loy and J.-O. Eklundh. Detecting symmetry and symmetric constellations of features. In *European Conference on Computer Vision*, pages 508–521, 2006.

[23] S. Mahamud, L. R. Williams, K. K. Thornber, and K. Xu. Segmentation of multiple salient closed contours from real images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):433–444, 2003.

[24] R. Mohan and R. Nevatia. Perceptual organization for scene segmentation and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):616–635, 1992.

[25] R. L. Ogniewicz and O. Kübler. Hierarchic Voronoi skeletons. *Pattern Recognition*, 28(3):343–359, 1995.

[26] V. S. N. Prasad and B. Yegnanarayana. Finding axes of symmetry from potential fields. *IEEE Transactions on Image Processing*, 13(12):1559–1566, 2004.

[27] E. Saber and A. Tekalp. Frontal-view face detection and facial feature extraction using color, shape and symmetry based cost functions. *Pattern Recognition Letters*, 19(8):669–680, 1998.

[28] S. Sarkar and K. Boyer. Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 478–483, 1996.

[29] A. Shashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *IEEE International Conference on Computer Vision*, pages 321–327, 1988.

[30] D. Shen, H. Ip, K. Cheung, and E. Teoh. Symmetry detection by generalized complex (GC) moments: a close-form solution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):466–476, 1999.

[31] H. Shroff and J. Ben-Arie. Finding shape axes using magnetic fields. *IEEE Transactions on Image Processing*, 8(10):1388–1394, 1999.

[32] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker. The Hamilton-Jacobi skeleton. In *IEEE International Conference on Computer Vision*, volume 2, pages 828–834, 1999.

[33] J. S. Stahl and S. Wang. Convex grouping combining boundary and region information. In *IEEE International Conference on Computer Vision*, volume 2, pages 946–953, 2005.

[34] J. S. Stahl and S. Wang. Globally optimal grouping for symmetric boundaries. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1030–1037, 2006.

[35] D. Terzopolous, A. Witkin, and M. Kass. Symmetry-seeking models and 3D object reconstruction. *International Journal of Computer Vision*, 1(3):211–221, 1988.

[36] T. Tuytelaars, A. Turina, and L. V. Gool. Noncombinatorial detection of regular repetitions under perspective skew. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):418–432, 2003.

[37] S. Wang, T. Kubota, J. Siskind, and J. Wang. Salient closed boundary extraction with ratio contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):546–561, 2005.

[38] S. Wang, J. S. Stahl, A. Bailey, and M. Dropps. Global detection of salient convex boundaries. *International Journal of Computer Vision*, 71(3):337–359, 2007.

[39] L. Williams and K. K. Thornber. A comparison measures for detecting natural shapes in cluttered background. *International Journal of Computer Vision*, 34(2/3):81–96, 2000.

[40] A. P. Witkin and J. M. Tenenbaum. On the role of structure in vision. In J. Beck, B. Hope, and A. Rosenfeld, editors, *Human and Machine Vision*, pages 481–543. Academic Press, New York, 1983.

[41] H. Zabrodsky, S. Peleg, and D. Avnir. Symmetry as a continuous feature. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1154–1166, 1995.

[42] S. C. Zhu and A. Yuille. FORMS: a flexible object recognition and modelling system. In *IEEE International Conference on Computer Vision*, pages 465–472, 1995.