International Journal of Computer Vision 71(3), 337–359, 2007

DOI: 10.1007/s11263-006-8427-2

Global Detection of Salient Convex Boundaries

SONG WANG*, JOACHIM S. STAHL, ADAM BAILEY AND MICHAEL DROPPS
Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208
songwang@cse.sc.edu
stahlj@cse.sc.edu
baileya6@cse.sc.edu
droppsm@cse.sc.edu

Received July 26, 2005; Revised March 22, 2006; Accepted March 22, 2006

First online version published in June, 2006

Abstract. As an important geometric property of many structures or structural components, convexity plays an important role in computer vision and image understanding. In this paper, we describe a general approach that can force various edge-grouping algorithms to detect only convex structures from a set of boundary fragments. The basic idea is to remove some fragments and fragment connections so that, on the remaining ones, a prototype edge-grouping algorithm that detects closed boundaries without the convexity constraint can only produce convex closed boundaries. We show that this approach takes polynomial time and preserves the grouping optimality by not excluding any valid convex boundary from the search space. Choosing the recently developed ratio-contour algorithm as the prototype grouping algorithm, we develop a new convex-grouping algorithm, which can detect convex salient boundaries with good continuity and proximity in a globally optimal fashion. To facilitate the application of this convex-grouping algorithm, we develop a new fragment-connection method based on four-point Bezier curves. We demonstrate the performance of this convex-grouping algorithm by conducting experiments on both synthetic and real images. In addition, we provide a comparison with some prior edge-grouping algorithms. Finally, we show that the proposed convex-grouping algorithm can be further extended to detect convex open boundaries, derive region-based image hierarchies, and incorporate some simple human-computer interactions.

Keywords: boundary detection, convexity, edge detection, edge grouping, graph models, perceptual organization

1. Introduction

A fundamental problem in computer vision is the automatic detection of the *desirable* structures from a noisy image. *Grouping* (or *perceptual organization*) aims to address this problem by clustering some image tokens into a small number of desirable groups (Lowe, 1985; Sarkar and Boyer, 1994; Forsyth and Ponce, 2003). As one of the most challenging problems in computer vision, grouping usually faces two difficulties: (a) how to define the *desirability* of a grouping, and (b) how to find an effective algorithm to achieve the grouping with the maximum desirability. The definition of the grouping desirability is usually application dependent. For example, in general-purpose grouping, the desirability may consider some Gestalt laws, such as closure, continuity, proximity, parallelism, symmetry, and so on. In a specific application, such as face detection, prior knowledge, such as the shape of the human face, is usually important to the grouping-desirability definition.

In this paper, our major goal is to find a way to enforce *convexity* in grouping, i.e., detecting only convex structures. Convexity is an important and very useful

*Correspondence author.

factor in grouping for several reasons. First, convexity plays an important role in measuring the desirability of the general-purpose grouping in both human and computer vision. For example, many psychophysical studies (Kanisza and Gerbino, 1976; Liu et al., 1999; Bertamini, 2001) have shown that convexity may dominate other important factors, such as continuity, closure, and symmetry in human vision. Jacobs (1996b) shows that convexity is statistically nonaccidental in real images and can be used to distinguish well-organized structures from the random noise. Second, in many computer-vision applications, the desirable structures may be *a priori* known to be convex, such as the silhouette of some buildings, various kinds of fruits, and some anatomical structures in biomedical images. In such applications, convexity needs to be enforced to get desirable groupings. For example, the use of convexity has been a key factor in detecting various cells in digital micrographs (Nattkemper, 2004). Third, even if the desirable structure is not fully convex, it may consist of several convex components. In this case, a convex grouping may produce more useful and robust results (Jacobs, 1987). For example, Borra and Sarkar (1997) compare several grouping algorithms in a recognition task and find that the one with the convexity constraint (Jacobs, 1996b) leads to the best recognition performance, although the desirable structures are not necessarily convex.

To consider convexity, this paper focuses on *edge grouping*, a category of grouping methods that has been widely investigated in the past decades (Shashua and Ullman, 1988; Alter and Basri, 1998; Jacobs, 1996b; Sarkar and Boyer, 1996; Guy and Medioni, 1996; Elder and Zucker, 1996; Amir and Lindenbaum, 1998; Williams and Thornber, 2000; Mahamud et al., 2003; Wang et al., 2005). In edge grouping, clustering tokens are a set of open *boundary fragments*, or for brevity, *fragments*, and the desirable structure is detected by identifying a subset of these fragments and connecting them sequentially into a boundary. In practice, these fragments are usually constructed using some simple image-processing methods. For example, in color/gray-scale images, fragments can be constructed by edge detection and line/curve approximation (Huttenlocher and Wayner, 1992; Jacobs, 1996b). In hand-drawn sketches and line art, fragments can be constructed by thresholding and contour tracing (Saund and Moran, 1995; Saund, 2003). In Section 2, we will review some related past work on edge grouping. Note that some prior edge-grouping methods, such as Jacobs (1996b), Elder and Zucker (1996), Mahamud et al. (2003), Wang et al. (2005), produce only closed boundaries, and others, such as Shashua and Ullman (1988), Sarkar and Boyer (1996), Alter and Basri (1998), Williams and Thornber (2000), do not. Since our goal is to detect convex structures, we are only interested in edge-grouping methods that detect closed boundaries (as described in Section 3) in this paper. For the remainder of this paper, we use the term "edge grouping" to indicate the ones that detect only closed boundaries when there is no special claims.

In this paper, we first develop a new general approach to enforce the strict boundary convexity in edge grouping that detects closed boundaries. This new approach has two important properties: (a) it is applicable to different edge-grouping algorithms with different desirability measures, (b) it preserves the grouping optimality by not excluding any valid convex boundary from consideration. Many prior works on convex grouping usually exhibit only one of these two properties. The basic idea of the proposed approach is to remove some fragments and fragment connections so that, on the remaining ones, the application of a prototype edge-grouping algorithm without the convexity constraint will produce only convex closed boundaries.

Using this convexity-enforcement approach, we develop a new convex-grouping algorithm, called *convex ratio contour*, by choosing the ratio-contour algorithm (Wang et al., 2005) as the prototype edge-grouping algorithm. This convex-ratio-contour algorithm is able to detect convex salient boundaries with good continuity and proximity in a globally optimal fashion. To facilitate the application of this convex-grouping algorithm, we develop a method for the fragment connection based on four-point Bezier curves, from which we can conveniently model the boundary continuity and evaluate the boundary convexity. To demonstrate the general applicability of this convexity-enforcement approach to various edge-grouping algorithms, we also apply it to two other edge-grouping algorithms (Mahamud et al., 2003; Elder and Zucker, 1996) to detect convex boundaries with their respective desirability measures.

In the remainder of this paper, Section 2 reviews the related past work on edge grouping and convex grouping. Section 3 gives a precise formulation of the problem. Section 4 describes the algorithm that enforces convexity in edge grouping. Section 5 extends the ratio-contour algorithm to the convex-ratio-contour algorithm. Section 6 discusses the fragment connection based on Bezier curves. Section 7 reports

the experimental results on synthetic and real images and compares these results with the experimental results from several other edge-grouping algorithms with or without the convexity constraint. In Section 8, we discuss several important extensions, including open-boundary detection, hierarchical image segmentation, and human-guided boundary detection. A brief conclusion is given in Section 9.

2. Related Work

2.1. Edge Grouping

In general-purpose edge grouping, the grouping desirability is often called the boundary *saliency*, which is usually based on some Gestalt laws (Wertheimer, 1938), such as boundary closure, continuity, and proximity. *Closure* means that the resulting boundary should be closed and fully partition a foreground structure out of the background. *Proximity* means that the gaps that are filled when connecting the fragments into the resulting boundary should be small. *Continuity* means that the resulting boundary should be smooth. As mentioned above, since our goal is to detect convex structures, we are only interested in edge grouping for closed boundaries in this paper.

Shashua and Ullman (1988) construct a local parallel network to model the detected fragments and define a boundary-saliency measure that favors longer boundaries with better proximity and continuity. An iterative update algorithm is developed to search for an optimal boundary that are usually not closed. Alter and Basri (1998) 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.

Elder and Zucker (1996) develop a globally optimal edge-grouping algorithm in a probabilistic framework, where local proximity and continuity associated with each fragment and fragment connection are modelled by probability functions. Boundary saliency is then defined as the likelihood, or the product of the probability, along the boundary and a shortest-path algorithm is applied to find the closed boundary with the maximum saliency. It has been shown that this algorithm has a bias to produce a boundary that contains fewer fragments (Wang et al., 2004, 2005).

To avoid the bias toward boundaries with fewer fragments, (Williams and Thornber, 2000; Mahamud et al., 2003) suggest a boundary-saliency measure using the geometric mean of the likelihood along the boundary. A spectral-analysis algorithm is used to encode the closure into the local probability function and a strongly-connected-components algorithm is used to achieve final closed boundaries. However, this algorithm has no guarantee to produce globally optimal solutions.

Wang et al. (2003, 2005) develop a *ratio-contour* algorithm for edge grouping by considering closure, proximity, and continuity. In this algorithm, boundary saliency is defined in a ratio form, with a normalization over the boundary length, to avoid the bias toward shorter boundaries or boundaries with fewer fragments. A graph-theoretic algorithm is developed to find the globally optimal solutions in polynomial time. In addition, this algorithm is guaranteed to produce closed boundaries. A recent comparison study (Wang et al., 2004, 2005) shows that the ratio-contour algorithm performs more favorably than Elder and Zucker's algorithm and Williams and Thornber's algorithm.

Saund (2003) uses edge grouping to find salient and compact closed boundaries in hand-written sketches and line art. This application is particularly suitable to be handled by edge grouping, because there is no useful intensity, color, or texture information in hand-written sketches and line art. In this work, a bidirectional search algorithm is developed for finding the salient closed boundaries, starting from some initially selected seed fragments. Saund (2003) also gives an overview of some other works on edge grouping that aim at detecting closed boundaries.

Elder et al. (2003) introduce more specific prior knowledge of the resulting boundary to edge grouping and use it to locate the lake boundaries from satellite imagery. In this work, the grouping desirability is defined in a probabilistic framework to combine both local fragment features and global prior knowledge. The developed edge-grouping algorithm employs a constructive search technique to compute candidate closed boundaries, from which the most desirable boundary is chosen to be the one with maximum posteriori probability.

2.2. Convex Grouping

Prior work on convex grouping was mainly carried out on the edge-grouping framework, because the structural convexity can be conveniently evaluated on fragments and fragment connections (Huttenlocher and Wayner, 1992). Based on a set of straight-line fragments, Huttenlocher and Wayner (1992) reduce the

possible number of fragment connections by a constrained triangulation. Based on the triangulation result, a local convexity graph is constructed and the salient convex boundaries are detected by a greedy-based search for long cycles with good proximity. However, this approach only considers the most possible connection incident from each fragment. This substantially reduces the search space and only a very small subset of the valid convex boundaries are considered. Therefore, it does not guarantee the global optimality of the resulting boundary in terms of the selected desirability measure.

Jacobs (1996b) develops a globally optimal algorithm to detect convex closed boundaries with good proximity. In this work, fragments and fragment connections are also straight-line segments. The boundary saliency is defined by the non-gap proportion along the boundary. By employing a sequential-search technique, this grouping algorithm detects all the convex boundaries that have a saliency higher than a pre-assigned threshold. This algorithm has a polynomial-time complexity in average, although it has an exponential-time complexity in the worst case. However, the convex-enforcement operations developed in this approach are not shown to be applicable to other edge-grouping algorithms with other desirability measures, especially the measures that consider the boundary continuity. Jacobs' convex-grouping algorithm has been used in object recognition tasks with better performance than several other edge-grouping algorithms (Borra and Sarkar, 1997).

Parvin and Viswanathan (1995) use dynamic programming to search for salient convex boundaries from straight-line fragments and the results are applied to object tracking. This algorithm combines the proximity and continuity. However, by measuring the proximity using the total-gap length along the boundary, this grouping algorithm has a bias to produce shorter boundaries. Estrada and Jepson (2004a,b) develop a search-control procedure to reduce the search complexity in convex edge grouping and then present a ranking system to identify the most desirable boundary. While this procedure improves the speed of convex grouping, there is no proof that it will always find the globally optimal boundary in terms of its desirability measure.

3. Problem Formulation

As mentioned above, edge grouping is operated on a set of n (boundary) fragments, which can be represented by straight-line segments with two endpoints Γ_k^- and Γ_k^+, $k = 1, 2, \ldots, n$, as shown in Fig. 1(a). As in Williams and Thornber (2000), this paper further sets the length of the fragment to zero, and therefore the kth fragment can be equally represented by (Γ_k, θ_k), where $\Gamma_k = (x_k, y_k)$ is the location of the fragment and $\theta_k \in [0, \pi)$ is the slope angle of the fragment. The equality of these two representations can be built by setting $\Gamma_k^- = (x_k - \delta\cos\theta_k, y_k - \delta\sin\theta_k)$ and $\Gamma_k^+ = (x_k + \delta\cos\theta_k, y_k + \delta\sin\theta_k)$, with a positive infinitesimal $\delta \to 0$.

Zero-length fragments can be constructed by tracing and sampling the edge-detection output (or edge-detection-like output (Saund, 2003)), which consists of a set of *(pixel) tracks*, i.e., connected pixel sequences. An example is shown in Fig. 1(b), where the edge

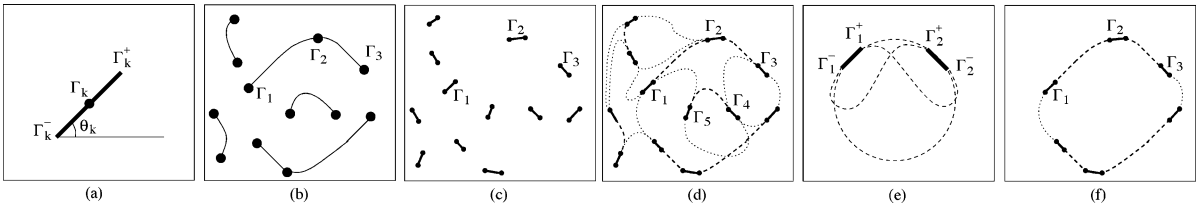

Figure 1. An illustration of fragments, detected and gap-filling arcs, and closed boundaries. (a) Two ways of representing a fragment: a line segment with two endpoints or a point with a slope angle. (b) Zero-length fragments (black dots) obtained by sampling the edge-detection output. (c) Redraw the 12 fragments in (b) by setting a small fragment length to make their slope angles and two endpoints visible. (d) Construct detected or gap-filling arcs to connect the fragments. The line-dashed curves represent the detected arcs and the dotted-dashed curves represent the gap-filling arcs. (e) Each arc is constructed by interpolating two fragment endpoints with G^1-continuity, i.e., continuous tangent directions. Therefore, between two fragments, we can construct four arcs (dashed curves) to connect all possible pair of fragment endpoints. Note that arcs interpolating the two endpoints of the same fragment are ignored in this paper. (f) A closed boundary is derived from the fragments and arcs shown in (d).

detection produces five tracks and we sample them with 12 fragments (shown as 12 black dots in Fig. 1(b)). The slope angle θ_k at a fragment can be derived from its neighboring pixels along the track. In Fig. 1(c), we redraw these 12 zero-length fragments by assuming a small length so that their slope angles and endpoints are visible. Note that this way of drawing a zero-length fragment will be repeatedly used in many other illustrative figures in this paper. To achieve closed boundaries, we construct a set of *linking arcs* (or for brevity, *arcs*), as shown by the dashed curves in Fig. 1(d), to connect the constructed fragments.[1] These arcs are constructed in a way that each of them interpolates two fragment endpoints with G^1-continuity, i.e., continuous locations and continuous tangent directions (Foley et al., 1995), as shown Fig. 1(e). These constructed arcs have explicit parametric representations, which provide necessary geometric information for calculating the boundary desirability. Some arcs approximate a sequence of detected track pixels between two endpoints (line-dashed curves in Fig. 1(d)), and some arcs fill the gap between two endpoints (dotted-dashed curves in Fig. 1(d)). We name the former ones *detected* arcs and the latter ones *gap-filling* arcs. In Section 6, we will discuss one method of constructing both detected and gap-filling arcs.

Theoretically, to consider all possible fragment connections, we need to construct gap-filling arcs between each possible pair of fragment endpoints, as shown Fig. 1(e). In practice, however, we may only need to fill the gaps that are likely to be filled by a valid arc. Specifically, considering the proximity preference, in this paper we only construct gap-filling arcs between two fragment endpoints that are sufficiently close to each other. This will be discussed in more detail in Sections 6 and 7. Note that a constructed gap-filling arc, such as the one between Γ_4 and Γ_5 in Fig. 1(d), may intersect another detected or gap-filling arc. However, in Section 4, we will show that our developed convex grouping algorithm is guaranteed to produce simple boundaries without self-intersections.

As mentioned above, we are only interested in edge grouping that detects closed boundaries. Based on the constructed fragments and arcs, a valid closed boundary can be defined as a cycle that traverses a subset of fragments and arcs *alternately*, as shown in Fig. 1(f). The goal of edge grouping is then to find from all such valid closed boundaries the one that has the largest desirability. As reviewed in Section 2, based on different grouping-desirability measures, many edge-grouping algorithms have been developed for this purpose in the past decades.

In this paper, our major goal is to develop a general approach to enforce convexity into edge grouping. This problem can be described as

Problem 1. From the constructed fragments and arcs, given a *prototype* edge-grouping algorithm $\mathcal{E}$ that can find the optimal closed boundary B^* in terms of a desirability $\phi(B)$, develop a convex-grouping algorithm to find the optimal closed boundary B_c^* that maximizes the same desirability $\phi(B)$ subject to the constraint that B_c^* is convex.

Certainly, we assume that the convexity constraint is not considered in the prototype edge-grouping algorithm $\mathcal{E}$. As mentioned above, we expect to solve this problem by developing a general approach that is independent of the specific edge-grouping desirability $\phi(B)$ and prototype algorithm $\mathcal{E}$. In the next section, we show that this goal can be achieved by an arc-pruning algorithm.

4. Arc-Pruning Algorithm

The basic idea of the arc-pruning algorithm is to remove some arcs from consideration so that running the prototype algorithm $\mathcal{E}$ on the remaining fragments and arcs will always produce convex boundaries. To address Problem 1, we first address the problem with an additional constraint:

Problem 2. From the constructed fragments and arcs, given a prototype algorithm $\mathcal{E}$ that can find the optimal closed boundary B^* in terms of a desirability $\phi(B)$, develop a convex-grouping algorithm to find the optimal closed boundary B_{ci}^* that maximizes the same desirability $\phi(B)$ subject to two constraints: (a) B_{ci}^* is convex, and (b) B_{ci}^* traverses the fragment (Γ_i, θ_i).

With an algorithm solving Problem 2, we can repeat it n times to find n optimal boundaries B_{ck}^* that traverse the fragment (Γ_k, θ_k), $k = 1, 2, \ldots, n$, respectively. Among these n optimal boundaries, the one with the maximum desirability $\phi(B)$ is then the optimal boundary B_c^* desired in Problem 1. In the following, we show that B_{ci}^* can be achieved by pruning some arcs and then running the prototype algorithm $\mathcal{E}$ on the remaining fragments and arcs.

First, all the nonconvex arcs can be pruned, as a convex boundary cannot traverse any nonconvex arcs.

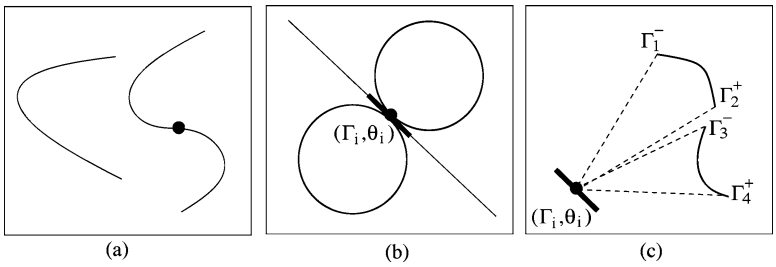

Figure 2. An illustration of arc pruning for finding the optimal convex closed boundary that traverses the fragment (Γ_i, θ_i). (a) A convex arc (left) and a nonconvex arc with an inflection point (right). (b) Two semiplanes partitioned by the straight line coincident with the fragment (Γ_i, θ_i). Note that any convex boundaries traversing (Γ_i, θ_i) must fall completely in one semiplane. (c) Arcs that are convex and nonconvex with respect to (w.r.t) (Γ_i, θ_i).

Examples of a convex arc and nonconvex arc (with one inflection point) are shown in Fig. 2(a). An *inflection point* is a point on a curve at which the sign of the curvature changes and there is no inflection point on any convex boundary. In the implementation, we in fact accomplish this step of pruning by only constructing convex arcs, as discussed in Section 6 and Appendix.

Second, the straight line coincident with the fragment (Γ_i, θ_i) partitions the plane into two semiplanes, as shown in Fig. 2(b). Since all the arcs have G^1-continuity at the interpolated fragment endpoints, the optimal boundary B_{ci}^* must be located completely in one of these two semiplanes, also as shown in Fig. 2(b). Therefore, we only need to solve Problem 2 in two semiplanes separately. First, we prune all the arcs located in the left semiplane and find the optimal boundary in the right semiplane. Similarly, we prune all the arcs in the right semiplane and find the optimal boundary in the left semiplane. The optimal boundaries found in these two semiplanes are then compared and the one with the larger desirability $\phi(B)$ is the desired B_{ci}^* in Problem 2. Note that the arcs with two endpoints located in different semiplanes are always pruned.

Third, in the considered semiplane, we can further prune all the arcs that are nonconvex *with respect to* (*w.r.t*) the given fragment (Γ_i, θ_i), i.e., these pruned arcs are convex by themselves, but they cannot be included in a convex boundary that traverses the fragment (Γ_i, θ_i). Since the considered arc is completely located in one semiplane as discussed above, its convexity w.r.t a given fragment can be determined by checking the convexity of the cone formed by the involved arc and fragment. As shown in Fig. 2(c), an arc $\Gamma_1^-\Gamma_2^+$ is convex w.r.t (Γ_i, θ_i) only if the closed curve made up of the arc $\Gamma_1^-\Gamma_2^+$ and the two straight line segments $\overline{\Gamma_1\Gamma_i}$ and $\overline{\Gamma_2\Gamma_i}$ is convex. In Fig. 2(c), it is easy to see that the arc $\Gamma_1^-\Gamma_2^+$ is convex w.r.t (Γ_i, θ_i), but the arc $\Gamma_3^-\Gamma_4^+$ is not.

After pruning the arcs in the above three steps, we perform the prototype algorithm $\mathcal{E}$ on the remaining arcs to achieve a closed boundary that maximizes the desirability $\phi(B)$, without imposing any constraint. The following lemma shows that the achieved boundary is the optimal boundary B_{ci}^* desired in Problem 2.

Lemma 1. *Running the prototype algorithm $\mathcal{E}$ on the remaining arcs after the above three steps of arc pruning produces a convex, simple, and closed boundary, which is the optimal boundary B_{ci}^* desired in Problem 2.*

Proof: Without loss of generality, let's consider solving Problem 2 in the semiplane above the fragment (Γ_i, θ_i), as shown in Fig. 3(a–c). We prove this lemma by contradiction.

First, let's assume that the detected boundary does not traverse the fragment (Γ_i, θ_i). Note that Γ_i must be located outside or on the resulting boundary, because we only consider the boundary in one semiplane. Let's further assume that Γ_i lies completely outside the resulting boundary, as shown in Fig. 3(a). Then there must exist an arc, shown as $\Gamma_1^-\Gamma_2^+$ in Fig. 3(a), which is on the resulting boundary, but is nonconvex w.r.t (Γ_i, θ_i). This contradicts the fact that all the arcs after the pruning are convex w.r.t (Γ_i, θ_i). Therefore, the resulting boundary must traverse the fragment (Γ_i, θ_i).

Second, let's assume that the resulting boundary is nonconvex by having an inflection point. Since all the arcs on the boundary are convex after the arc pruning, such an inflection point can only appear at a fragment, say (Γ_j, θ_j), which is a joint of two neighboring arcs, as illustrated in Fig. 3(b). With the G^1-continuity at (Γ_j, θ_j), the line coincident with the fragment (Γ_j, θ_j) is the tangent line of the boundary. We further construct the normal line at Γ_j, which is perpendicular to

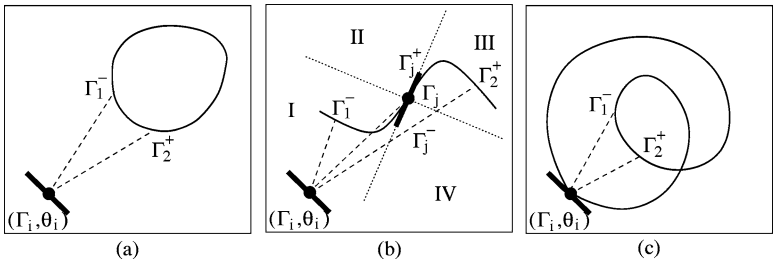

Figure 3. Counter examples used in proving Lemma 4.

the tangent line. The tangent and normal lines partition the plane into four domains, I, II, III, IV, as shown in Fig. 3(b). Since Γ_j is an inflection point, the two neighboring arcs, $\Gamma_1^-\Gamma_j^-$ and $\Gamma_2^+\Gamma_j^+$ (or the portions of these two arcs that are near Γ_j), as shown in Fig. 3(b), must be located in two different domains that are symmetric over the point Γ_j, i.e., they are located in either I and III, or II and IV. Certainly, the given fragment (Γ_i, θ_i) can be located in any of these four domains. However, whichever domain the fragment (Γ_i, θ_i) is located in, either the arc $\Gamma_1^-\Gamma_j^-$ or the arc $\Gamma_2^+\Gamma_j^+$ is nonconvex w.r.t (Γ_i, θ_i). An example is shown in Fig. 3(b), where Γ_i is located in domain I and the arc $\Gamma_1^-\Gamma_j^-$ is nonconvex w.r.t (Γ_i, θ_i). This also contradicts the fact that all the arcs after the pruning are convex w.r.t (Γ_i, θ_i). Therefore, the resulting boundary has no inflection point.

Third, let's assume that the detected closed boundary contains no inflection point but is not simple, as shown in Fig. 3(c). In this case, (Γ_i, θ_i) must be located on the outermost loop of the boundary given that the boundary is located in one semiplane. As shown in Fig. 3(c), we can always find an arc, say $\Gamma_1^-\Gamma_2^+$, on the inner loop of the boundary so that this arc is nonconvex w.r.t (Γ_i, θ_i). Similarly, this contradicts the fact that all the arcs after the pruning are convex w.r.t (Γ_i, θ_i). Therefore, the resulting boundary must be simple. As pointed out in Section 3, the constructed arcs may intersect each other. The property proved here ensures that any two intersected arcs would not appear simultaneously on the resulting boundary.

Finally, since all the arcs along a convex boundary that traverses (Γ_i, θ_i) must be convex w.r.t (Γ_i, θ_i) and all such arcs are not pruned in the above arc pruning, the convex-grouping algorithm combining the arc-pruning algorithm and the prototype grouping algorithm $\mathcal{E}$ provides the globally optimal boundary B_{ci}^* desired in Problem 2. □

The following summarizes this convex-grouping algorithm, which in fact needs to run $\mathcal{E}$ $2n$ times because each run of $\mathcal{E}$ can only find the optimal boundary in one semiplane.

Convex-Grouping Algorithm:

```
Construct fragments (Γ_i, θ_i), i = 1, 2, ... n
Construct detected and gap-filling arcs
Loop over fragment (Γ_i, θ_i), i = 1, 2, ... n
    On semiplane 1
        Run the arc-pruning algorithm
        Run the edge-grouping algorithm E to get a boundary B_1
    On semiplane 2
        Run the arc-pruning algorithm
        Run the edge-grouping algorithm E to get a boundary B_2
    Take B*_ci = B_1 if φ(B_2) < φ(B_1), and B*_ci = B_2 otherwise
Output B*_c = {B*_cm | φ(B*_ci) ≤ φ(B*_cm), i = 1, 2, ..., n}
End
```

Note that, in this convex-grouping algorithm, the arc-pruning operations are independent of the prototype algorithm $\mathcal{E}$ and preserves the grouping optimality if $\mathcal{E}$ is a globally optimal algorithm. However, as reviewed in Section 2, many edge-grouping algorithms are not globally optimal. A natural question then is whether they can still be used as the prototype algorithm $\mathcal{E}$ to achieve a convex grouping. In fact, only if $\mathcal{E}$ is guaranteed to produce a closed boundary, the above proof is still valid on the claims that the arc-pruning algorithm leads to a convex, simple boundary that traverses the fragment (Γ_i, θ_i). Therefore, edge-grouping algorithms without the global optimality can still be used as the prototype algorithm $\mathcal{E}$ to achieve convex groupings. The only problem is that the resulting boundary is not globally optimal in terms of the grouping desirability. In general, a globally optimal grouping algorithm is independent of other heuristic operations and is able to provide a more convincing evaluation of the selected desirability measures. In the next section, we develop a new convex-grouping algorithm where the globally optimal ratio-contour algorithm (Wang et al., 2005) is used as the prototype algorithm $\mathcal{E}$.

5. Convex Ratio Contour

Ratio contour (RC) (Wang et al., 2005) is a general-purpose edge-grouping algorithm developed for detecting perceptually salient closed boundaries. This algorithm considers the Gestalt laws of closure, continuity, and proximity. A recent comparison study (Wang et al., 2004, 2005) shows that it performs more favorably than some other edge-grouping algorithms. By adopting it as the prototype algorithm $\mathcal{E}$, we achieve a new convex-grouping algorithm, called *convex ratio contour* (*CRC*), which can detect salient convex boundaries in a globally optimal fashion.

In RC, all the arcs are constructed to be smooth curve segments to achieve good boundary continuity. Let $\mathbf{q}(t) = (x(t), y(t))$, $t \in [0, L(B)]$ be the arc-length parameterized representation (Bartels et al., 1987) of a valid closed boundary B formed by alternately connecting a sequence of arcs and fragments, where $L(B)$ is the boundary length. We know that $\mathbf{q}(L(B)) = \mathbf{q}(0)$ as the boundary is closed. In RC, the boundary *cost* $R(B)$ (negatively related to the boundary saliency/desirability, e.g., $\phi(B) = e^{-R(B)}$) is defined as

$$R(B) = \frac{\int_0^{L(B)} [\sigma(t) + \lambda \cdot \kappa^2(t)] dt}{L(B)}, \qquad (1)$$

where $\sigma(t) = 1$ if $\mathbf{q}(t)$ is on a gap-filling arc and $\sigma(t) = 0$, otherwise. $\kappa(t)$ is the curvature of the boundary at $\mathbf{q}(t)$. Since all the fragments are of zero length, they have no direct contribution to this cost. In the numerator of (1), the first term $\int_0^{L(B)} \sigma(t) dt$ makes it biased to a boundary with longer detected arcs and shorter gap-filling arcs. This reflects the preference of better proximity. The second term $\int_0^{L(B)} \kappa^2(t) dt$ reflects the favor of smoother boundaries, or better continuity. The denominator normalizes the cost by the boundary length $L(B)$ to avoid a bias to shorter boundaries. $\lambda > 0$ is a regularization factor that balances the proximity and continuity in the cost function. As in Wang et al. (2005), we choose a uniform λ for all the experiments.

In Wang et al. (2005), a graph-based RC algorithm was developed to find the optimal closed boundary that globally minimizes the cost (1). This algorithm has a polynomial-time complexity in the worst case. By using RC as the prototype algorithm $\mathcal{E}$ for the convex-grouping algorithm developed in Section 4, CRC is a globally optimal grouping algorithm that can detect convex salient boundaries by considering the properties of convexity, proximity, continuity, and closure. To facilitate the application of CRC, we discuss the fragment and arc construction in the next section.

6. Fragment and Arc Construction

6.1. Fragment Construction

As mentioned in Section 3, we sample all the *tracks* in an edge-detection output (or an edge-detection-like output (Saund, 2003)) to construct a set of zero-length fragments. Between these fragments, we then construct detected or gap-filling arcs. In the fragment construction, the sampling rate should not be too sparse because sparse sampling may produce long detected arcs that are only partially present in the desirable closed boundary. Since in this paper the detected arcs are not allowed to be divided in the edge grouping, such long detected arcs may prevent the correct detection of the desirable boundary (Jacobs, 1992, 1996b).[2] On the other hand, we also need to avoid overly dense sampling, which may introduce unnecessary computational load. For simplicity, we use a uniform-sampling technique to construct the zero-length fragments in this paper. First, all the endpoints of the tracks are always chosen as fragments. In the edge-detection output, these endpoints correspond to the detected pixels with only one neighboring detected pixel or with more

than two neighboring detected pixels. Second, we uniformly sample each track to construct some additional fragments. This way, each track is partitioned into a set of subtracks with the given sampling length. Slope angle of each fragment is estimated by differentiating the locations of the detected pixels that are neighboring to this fragment along the the track.

With these fragments, we can construct parametric detected arcs to approximate the pixel-based subtracks. In the detected-arc construction, we may encounter the cases where a subtrack is still too long to be well approximated by a convex detected arc. Then we may further insert additional fragments to this subtrack to divide it into shorter subtracks for detected-arc construction. Therefore, we may have more fragments added in the following arc-construction step.

6.2. *Arc Representation and Construction*

In CRC, all the constructed arcs should be convex because nonconvex arcs will be removed in the arc pruning. In addition, all the constructed arcs should be of a simple and consistent parametric form. This will simplify their storage, facilitate the calculation of the boundary curvature and length needed in Eq. (1), and unify the arc convexity evaluation. In this paper, we use four-point Bezier curves (Boehm et al., 2002) to represent all detected/gap-filling arcs, with which the resulting boundary is guaranteed to be smooth, as required in RC and CRC. As shown in Fig. 4(a), to represent an arc interpolating the fragment endpoints Γ_1^+ and Γ_2^-, we find another two control points P_1 and P_2 to construct a Bezier curve

$$\mathbf{p}(u) = (1-u)^3\Gamma_1 + 3u(1-u)^2 P_1 + 3u^2(1-u)P_2 + u^3\Gamma_2,$$

with $u \in [0, 1]$. This Bezier curve interpolates at Γ_1^+ and Γ_2^-, and the tangent vectors at Γ_1^+ and Γ_2^- are $\mathbf{r}_1 = 3(P_1 - \Gamma_1)$ and $\mathbf{r}_2 = 3(\Gamma_2 - P_2)$, respectively. Since the resulting arc is required to have G^1-continuity at Γ_1^+ and Γ_2^-, P_1 and P_2 must lie on the straight lines coincident with the fragments (Γ_1, θ_1) and (Γ_2, θ_2) respectively.

To construct a detected arc, we construct a Bezier curve to approximate a subtrack, or a sequence of connected pixels, between two fragments, as shown in Fig. 4(b). For a gap-filling arc, there is no detected track or subtrack to approximate. The main goal is to find the perceptually possible arc to fill the gap between two fragment endpoints. This is a typical gap-completion problem, which has been investigated by many researchers (Bruckstein and Netravali, 1990; Mumford, 1994; Williams and Jacobs, 1997; Sharon et al., 2000; Srivastava et al., 2003; Mio et al., 2004). For simplicity and consistency, in this paper we still use above Bezier-curve model to estimate the gap-filling arcs, as shown in Fig. 4(c). The technical details of constructing the detected and gap-filling arcs using Bezier curves are documented in Appendix.

Arc convexity w.r.t a fragment. From the Bezier-curve representation, the length and the curvature of the arcs can be conveniently calculated and then supplied to RC and CRC. The convexity of a Bezier curve can be simply determined by checking the convexity of the quadrilateral that consists of the two fragment endpoints Γ_1^+ and Γ_2^- and the two control points P_1 and P_2, as shown in Fig. 4. In practice, we only construct convex arcs, as described in detail in Appendix. The only problem is then to efficiently determine whether an arc, say $\Gamma_1^+\Gamma_2^-$, is convex w.r.t a given fragment (Γ_i, θ_i). To do this, we need to check the convexity

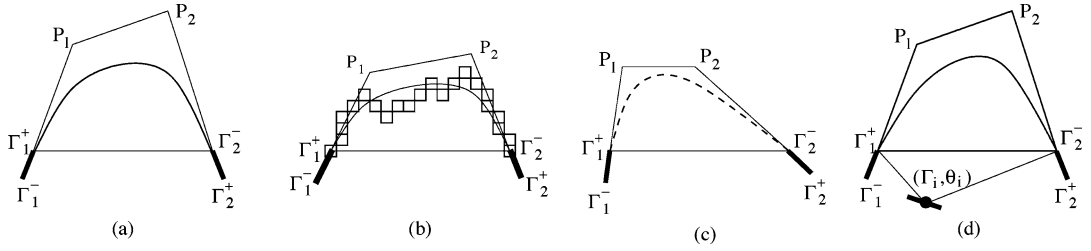

Figure 4. An illustration of the arc construction and the arc-convexity evaluation w.r.t to a fragment: (a) Bezier curve and its control points. (b) A detected arc constructed as a Bezier curve. Small squares in this figure represent the detected pixels along the subtrack. Note that the zigzag curve formed by sequentially connecting the track pixels is nonconvex, but the constructed arc is convex. (c) A gap-filling arc (dashed curve) constructed between two fragment endpoints Γ_1^+ and Γ_2^-. (d) Evaluating the convexity of the constructed arc $\Gamma_1^+\Gamma_2^-$ w.r.t the given fragment (Γ_i, θ_i).

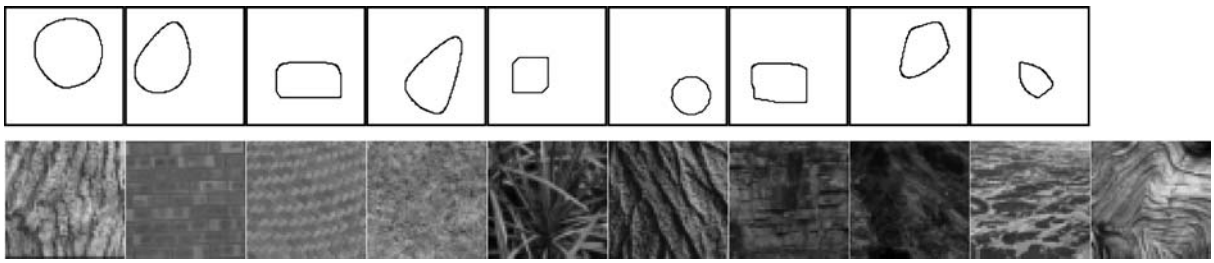

Figure 5. The nine convex boundaries and the 10 texture images used for constructing synthetic data.

of the curve made up of this arc and two straight line segments $\overline{\Gamma_i\Gamma_1}$ and $\overline{\Gamma_i\Gamma_2}$, as shown in Fig. 4(d). According to the variation-diminishing property (Boehm et al., 2002) of the Bezier curves, any straight line intersects the arc $\Gamma_1^+\Gamma_2^-$ no more times than intersecting the piecewise linear curve $\overline{\Gamma_1 P_1 P_2 \Gamma_2}$. Therefore, the arc $\Gamma_1^+\Gamma_2^-$ is convex w.r.t the fragment (Γ_i, θ_i) when the pentagon $\Gamma_i\Gamma_1 P_1 P_2\Gamma_2$ is convex. The convexity of a pentagon can be determined very efficiently (Cormen et al., 1990).

7. Experiments

7.1. Synthetic Data

We first construct some synthetic data that simulate the edge-detection outputs for testing CRC. With known ground-truth boundaries, the experiments on these synthetic data can provide a quantitative evaluation of the grouping performance. Specifically, we collect nine convex boundaries and 10 texture images, as shown in Fig. 5, where the 10 texture images come from Williams and Thornber (2000). As shown in Fig. 6, each synthetic data sample is constructed by combining some background tracks and some foreground tracks. The background tracks are constructed by applying the standard Canny edge detection on a selected texture image with the threshold $T = 0.4$ and the smoothing parameter $\sigma = 1.0$. The foreground tracks are constructed from a selected convex boundary by (1) rasterizing this convex boundary into a closed pixel track (Foley et al., 1995), (2) uniformly partitioning this closed track into a set of subtracks with a fixed length (in this experiment, this length is set to 13 pixels), and (3) randomly removing some of these subtracks to construct gaps, as shown in Fig. 6(b). The grouping performance is then evaluated by comparing the detected boundary with the ground-truth convex boundary that is used for constructing the foreground tracks.

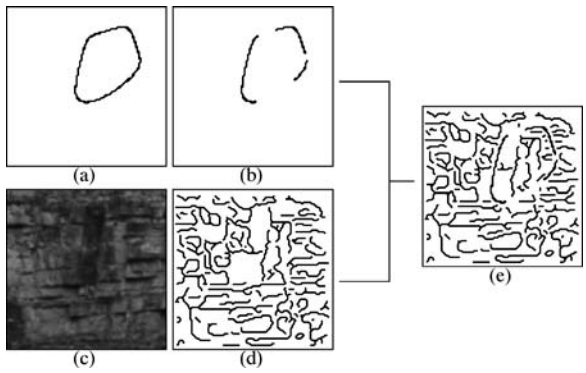

Figure 6. An illustration of constructing a synthetic data sample. (a) A rasterized convex boundary. (b) Foreground tracks. (c) A texture image. (d) Background tracks resulting from Canny edge detection on (c). (e) A synthetic data sample constructed by combining the foreground and background tracks.

For CRC, the method developed in Section 6 is used to construct fragments and arcs, as shown in Fig. 7(a), where the black dots represent the constructed fragments and the curve segment between two neighboring black dots represents a detected arc. We can see that all the detected arcs are convex. In all the experiments, the tracks with a length less than 13 pixels are removed as noise before arc construction. The uniform sampling length is set to 13 pixels in the fragment construction. Note that the number of gap-filling arcs may be very large ($O(n^2)$ with n being the number of fragments) if we fill all the possible gaps. To control the number of gap-filling arcs, we only consider filling the gaps with a length of less than 100 pixels. Figure 7(b) demonstrates some of the constructed gap-filling arcs. To avoid making this figure too crowded, we only show the convex gap-filling arcs incident from one specific fragment in Fig. 7(b). Note that we still consider constructing gap-filling arcs between fragments that are extracted from the same track/subtrack. This way, even if some portions of a long track/subtrack can not be approximated

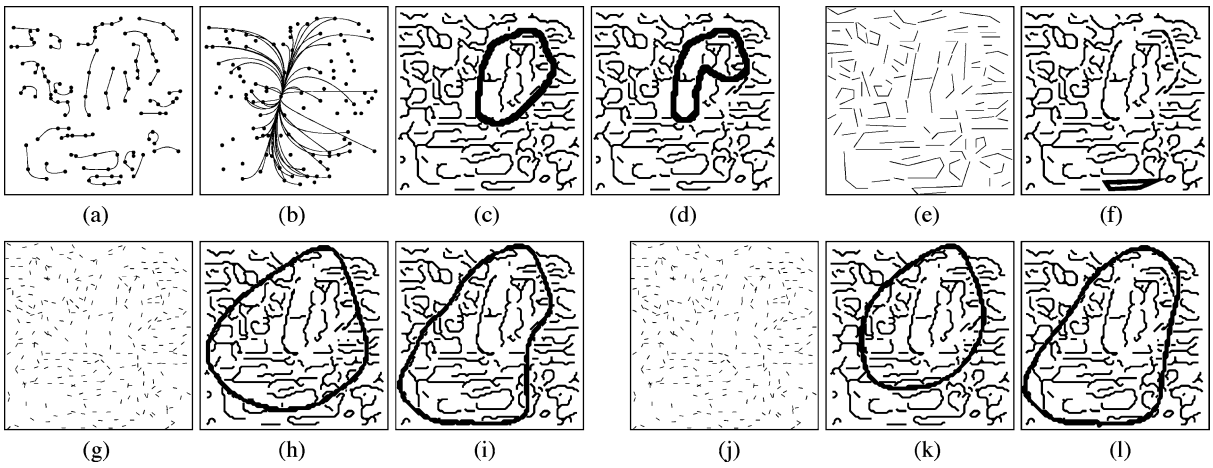

Figure 7. Edge-grouping results on the synthetic data sample shown in Fig. 6(e). (a) Detected fragments (black dots) and detected arcs using the method introduced in Section 6. (b) Constructed convex gap-filling arcs incident from one fragment using the method introduced in Section 6. (c) CRC grouping result. (d) RC grouping result. (e) Detected arcs in Jacobs' algorithm. (f) The optimal grouping result using Jacobs' algorithm. (g) Fragments constructed in WT. (h) Convex-WT (CWT) grouping result. (i) WT grouping result. (j) Fragments constructed in EZ. Note they are same as the ones shown in (g). (k) Convex-EZ (CEZ) grouping result. (l) EZ grouping result.

by convex detected arcs, this long track/subtrack will not be completely excluded from the grouping. The regularization parameter λ is set to 10 in the cost (1). Figure 7(c) illustrates the CRC edge-grouping result on the sample shown in Fig. 7(a).

We try all possible combinations between the nine convex boundaries and the 10 texture images in constructing the synthetic data samples. Furthermore, in constructing the foreground tracks, we try the cases where gaps count for 5%, 10%, 20%, 30%, 40% and 50% of the ground-truth boundary. This way, given one gap percentage, we have $9 \times 10 = 90$ data samples and totally we have $90 \times 6 = 540$ data samples. As in Wang et al. (2005), we define the grouping accuracy by a region-based coincidence measure $\frac{|R_D \cap R_G|}{|R_D \cup R_G|}$, where R_D and R_G are the regions bounded by the detected boundary and the ground-truth boundary respectively, and $|R|$ indicates the area of R. The performance of CRC on these data samples is shown in Fig. 8, where the accuracy at each gap percentage is the average over 90 data samples.

Figure 8 also shows the performances of several other edge-grouping methods. (i) RC ($\lambda = 10$) on the same fragments/arcs used in CRC, (ii) Jacobs' algorithm (Jacobs, 1996b), where arcs are constructed as straight-line segments, as shown in Fig. 7(e), and the grouping result is a convex polygon, as shown in Fig. 7(f). (iii) Williams and Thornber's algorithm (WT) (Williams and Jacobs, 1997; Williams and Thornber, 2000; Mahamud et al., 2003) and the convex-WT algorithm (CWT), where WT is used as the prototype algorithm in the proposed convex-grouping algorithm. (iv) Elder and Zucker's algorithm (EZ) (Elder and Zucker, 1996) and the convex-EZ algorithm (CEZ), where EZ is used as the prototype algorithm in the proposed convex-grouping algorithm. We include EZ and WT in this experiment to show that the proposed convexity-enforcement approach is applicable to various edge-grouping algorithms with different desirability measures. In Wang et al. (2005), we have shown that RC usually produces comparable or better performance than EZ and WT. Figure 7 shows the constructed fragments and grouping results when using these several edge-grouping methods on a synthetic data sample.

As in Wang et al. (2005), the fragments in WT and EZ, as shown in Figs. 7(g) and (j), are constructed from the edge-detection-like output (Fig. 6(e)) with a sampling rate of 8 pixels and all the arcs are then constructed as gap-filling ones using the stochastic-completion-field method (Williams and Jacobs, 1997). Specifically, the parameters used in the stochastic-completion-field method are $\gamma = 0.15, T = 0.004$, and $\tau = 5.0$, as suggested in Williams and Jacobs (1997). For an arc between the fragment endpoints i and j, an affinity measure p_{ij} is defined to describe its likelihood to be included in the resulting salient boundary. In EZ,

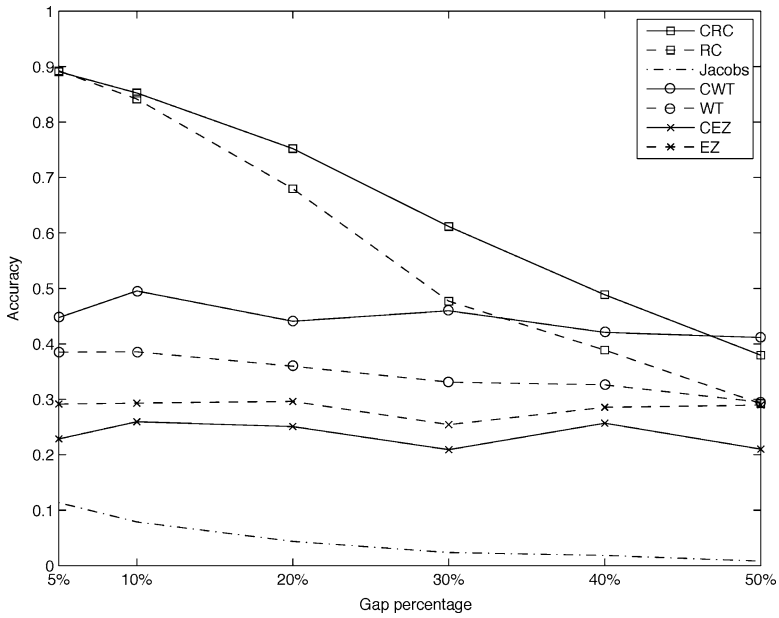

Figure 8. Grouping performance of CRC, RC, Jacob's algorithm, CWT, WT, CEZ, and EZ on 540 data samples.

a shortest-path algorithm is developed to find a closed boundary B that maximizes the desirability measure $\phi_{ez}(B) = \prod_{(i,j)\in B} p_{ij}$. In WT, the arc affinity measure is first enhanced by an eigenvalue decomposition and then a strongly-connected-component algorithm is used to find a closed boundary B that aims at maximizing the desirability measure $\phi_{wt}(B) = (\prod_{(i,j)\in B} p_{ij})^{\frac{1}{|B|}}$, where $|B|$ is the number of the arcs along B.

Without considering the boundary continuity, Jacobs' algorithm uses the proximity as the boundary desirability, i.e., the saliency of a boundary B is defined as $\phi_J(B) = \frac{g(B)}{L(B)}$, where $g(B)$ is the total length of detected arcs along the boundary B, and $L(B)$ is the total length of the boundary B. From the performance curves shown in Fig. 8, we can clearly see that, (a) given that the desirable structures are convex, CRC does produce better performance than RC, especially when the gap percentage along the desirable boundary increases; and (b) without considering the boundary continuity, Jacobs' algorithm may produce overly small or thin structures. For WT, incorporating convexity constraint also improves the grouping performance, when the desirable structure is convex. In fact, the desirability measures used in WT and RC are very similar to each other (Wang et al., 2005). For EZ, the incorporation of convexity constraint does not improve the edge-grouping performance. The reason may be that EZ has a bias to produce boundaries that traverse fewer fragments and arcs. Such a bias may result in a boundary that is completely different from the ground-truth one. In this case, enforcing convexity would not improve the edge-grouping results.

Note that these experiments, as well as the experiments reported in the next section, only compare the single optimal boundaries produced by different edge grouping methods. For EZ, RC, and Jacobs' methods, they are globally optimal in terms of their respective desirability measures, these experiments in essence compare the desirability measures underlying these methods. For WT, there is no proof that the extracted boundary has the maximum desirability $\phi_{wt}(B)$, but we believe its performance is mainly determined by this selected desirability measure. In practice, Jacobs's method detects a set of convex boundaries that can be ranked by their desirability. The ground-truth boundary may appear as one of these detected convex boundaries but not as the top-ranked one with maximum desirability (Estrada and Jepson, 2004b). Similarly, in some data samples, the optimal boundary detected by RC or CRC may not coincide with the ground-truth boundary, but the further iterations of RC or CRC may find the desired ground-truth boundary. For simplicity, we only consider and compare the single optimal boundaries detected by these methods in this paper.

7.2. *Real Images*

In this section, we use CRC to detect closed boundaries on a set of real images, where pixel tracks are constructed by edge detection. However, the performance of a general-purpose grouping on real images is difficult to evaluate quantitatively and objectively, given the complexity of real images and the unavailability of the ground truth. In this section, we show grouping results on some sample real images as has been done in most prior work on edge grouping. We believe these sample results still qualitatively provide some insights that are consistent to the conclusion drawn from the above experiments on synthetic data.

On real images, pixel tracks are simply constructed by standard Canny edge detection with the threshold $T = 0.4$ and the smoothing parameter $\sigma = 1.0$. The other parameters for fragment/arc construction and CRC grouping are the same as the ones used in the above experiments on synthetic data. The fourth column (from the left) of Fig. 9 shows the optimal boundaries detected on 15 natural and medical images using the proposed CRC. As a comparison, the fifth and ninth columns show the grouping results using RC and Jacobs' algorithm.[3] Similar to the results on synthetic data, we see that CRC does improve the detection of the salient boundaries over RC given that the structures of interest are convex. This is demonstrated by the results on the images shown in Figs. 9(a), (b), (e), (i), (j) and (n). Particularly, CRC is guaranteed to produce simple boundaries without self-intersections, which may not be the case for RC without the convexity constraint, as shown in Figs. 9(a) and (b). However, if the structures of interest are nonconvex, RC sometimes produces more desirable boundaries than CRC, as shown in Figs. 9(d) and (m). Comparing the results in the fourth and ninth columns, we can see that the incorporation of the boundary continuity in CRC avoids producing overly short boundaries or thin structures as in Jacobs' algorithm. For example, on the images shown in Figs. 9(a), (b), (c), (d), (f), (i), (j), (k), (m), (n), and (o), Jacobs' algorithm detects overly small or thin structures, while CRC handles this problem well. This is consistent to the above experimental results on synthetic data.

Finally, we briefly discuss the running time of the proposed method. The ratio-contour algorithm, which incorporates a step of finding the minimum weight perfect matching in an undirected graph, has the worst-case time complexity $O(n^{\frac{7}{4}})$ (Wang et al., 2003), where n is the number of fragments. Since we need to repeat the ratio-contour algorithm $2n$ times as mentioned in Section 4, an upper-bound of the worst-case time complexity of the proposed CRC is then $O(n^{\frac{11}{4}})$, which looks very high. However, in practice, the arc-pruning step usually removes a large number of arcs from consideration and the number of the remaining arcs supplied to the ratio-contour algorithm is usually much smaller than n. Table 1 shows the CPU time used by CRC and RC on these 15 images. We can see that the running time of CRC is much smaller than $2n$ times of the the running time of RC. For Jacobs' algorithm, the average time complexity is $O(n^2 \log(n) + nm)$ when we wish to find the m most salient boundaries. Table 1 also shows the CPU time used by Jacobs' algorithm.[4] We can see that the current RC and CRC implementations takes more CPU time than Jacobs' algorithm. This results from (a) the graph algorithm used in RC and CRC has relatively higher complexity and (b) we did not optimize the current RC/CRC implementations, given that the speed is not our priority in this paper. We believe that the speed of the RC/CRC implementations can be substantially improved with more carefully-designed programs. One natural question is then whether we can extend the Jacobs' sequential-search algorithm to solve the convex-ratio-contour problem. Unfortunately, we find this to be nontrivial because Jacobs' sequential-search algorithm is highly dependent on the proximity-based boundary desirability (Jacobs, 1996b) and is difficult to be extended to other edge-grouping methods. Also note that, while Jacobs' method takes less CPU time in these

Table 1. The CPU time (in seconds) used by CRC, RC, and Jacobs' algorithm in processing the 15 images shown in Fig. 9. This CPU time includes the Canny edge detection, fragment/arc construction, and edge grouping. The experiments are conducted on a Dell Precision Workstation equipped with a 3.06 GHz Intel Xeon Processor with a 512 K L2 Cache.

Algorithm	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)	(m)	(n)	(o)
CRC	75.70	17.10	15.95	45.49	183.32	25.77	60.36	13.90	19.10	52.93	98.55	37.27	13.09	3.89	20.56
RC	26.59	11.85	9.27	14.82	34.18	13.06	17.86	8.21	11.12	21.19	26.94	14.55	7.00	3.20	8.50
Jacobs	3.17	2.40	1.09	2.19	4.89	1.90	3.13	1.94	1.93	2.69	1.16	0.70	1.08	1.18	1.32

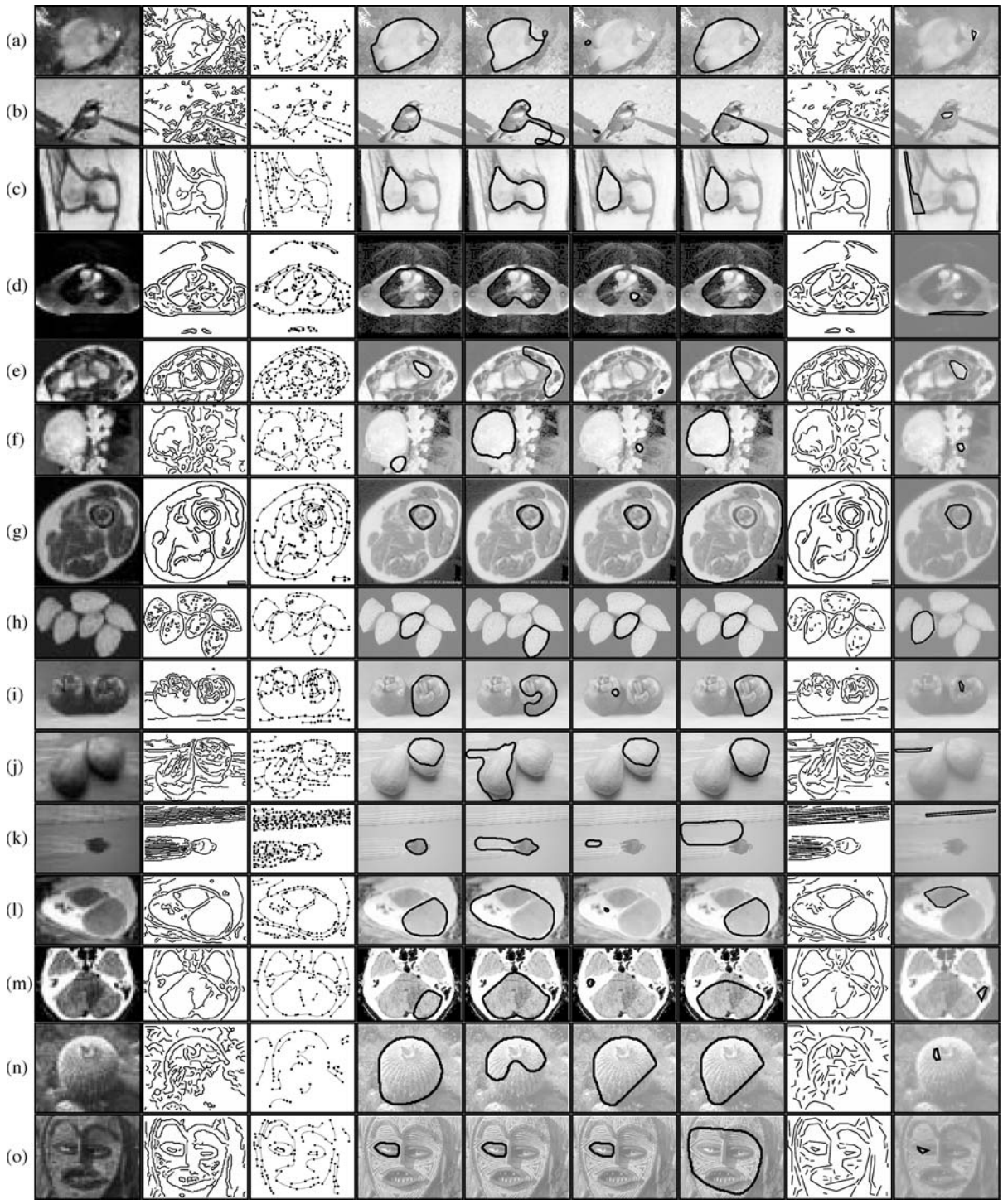

Figure 9. Edge-grouping results on 15 natural and medical images. Column 1: original images. Column 2: Canny edge detections. Column 3: constructed fragments (black dots) and detected arcs using the methods developed in Section 6. Column 4: CRC grouping results ($\lambda = 10$). Column 5: RC grouping results ($\lambda = 10$). Column 6: CRC grouping results ($\lambda = 0$). Column 7: CRC grouping results based on cost (2) ($\lambda = 10$). Columns 6 and 7 correspond to the experiments discussed in Section 8.3. Column 8: constructed detected arcs in Jacobs' algorithm. Column 9: grouping results from Jacobs' algorithm.

images, its algorithm complexity is non-polynomial in the worst case. On the contrary, the proposed convex grouping algorithm is always of polynomial-time complexity.

8. Extensions

In this section, we discuss several important extensions of the proposed convex-grouping algorithm.

8.1. *Open Boundaries and Hierarchical Segmentation*

In practice, most images contain multiple structures that can be described by a top-down segmentation hierarchy. This usually requires us to detect structural boundaries recursively from the original image and subsequent regions. In this section, we discuss the iterative application of CRC to achieve a hierarchical image segmentation. For this purpose, we first extend CRC from only detecting closed boundaries to detecting both closed and open boundaries. As shown in Figs. 10(a) and (b), a closed boundary indicates that the structure of interest is located completely in the image perimeter, while an open boundary indicates that the structure of interest is only partially located in the image perimeter. Note that there is fundamental difference between an open boundary and an arc, although both of them are open curve segments. An open boundary, just like a closed boundary, must fully partition the image into subregions, as shown in Fig. 10(b), while an arc cannot accomplish a region partitioning, as shown in Fig. 10(c). Therefore, the two endpoints of an open boundary on an image must be located on the image perimeter, also as shown in Fig. 10(b).

Supporting both open- and closed-boundary detection is particularly important for hierarchical image segmentation. An example is shown in Fig. 10(d), where a closed boundary B_0 is detected first to partition the image into two regions. In the second step, an open boundary B_1 may be detected to further partition one of the regions into two subregions. Therefore, we need open boundaries to segment not only the original image, but also the subsequent regions. The latter case is in fact very common in practice as many structures may be partially overlapped or occluded by others.

Next, we describe the CRC extension for detecting both open and closed boundaries in the original image. This extension can also be easily implemented for open/closed-boundary detection in subsequent regions. To allow open boundaries, we construct some gap-filling arcs that are located partially outside the image perimeter. An example is shown in Fig. 10(e), where a large portion of a gap-filling arc constructed between the fragments Γ_1 and Γ_2 is located outside the image perimeter. We call such a gap-filling arc an *external* arc. In the arc pruning, all the external arcs will be examined in the same way as other arcs that are located completely in the image perimeter. But when computing the curvature and length terms in the numerator and denominator of the cost function (1), we omit the contribution from the portions outside the image perimeter. This way, the cost (1) is consistently defined for both open and closed boundaries.

With this modification, CRC may detect an optimal closed boundary that contains some external arcs, as shown in Fig. 10(e), because the long gaps outside the image perimeter are not counted into the cost (1). Such a boundary, when we only view the portions located in the image perimeter, is in fact an open boundary, as shown in Fig. 10(f). In the implementation, we consider the external-arc construction between all fragment endpoints that are located near the image perimeter. Note that these fragment endpoints may be located far from each other. In our experiments, a fragment is regarded as being near the image perimeter if its distance to the

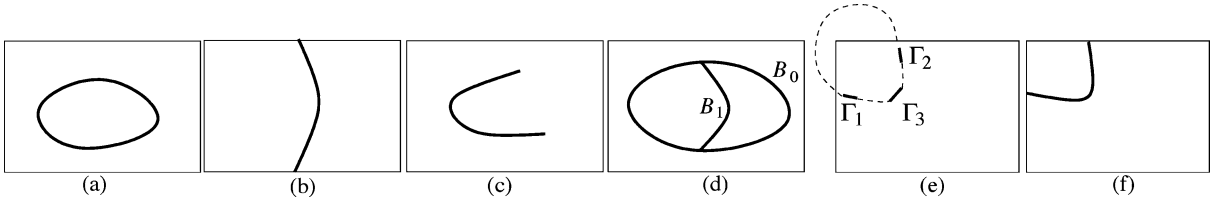

Figure 10. An illustration of open and closed boundaries and the open-boundary detection. (a) A closed boundary. (b) An open boundary. (c) An arc, or an incomplete segment of a boundary. (d) A hierarchical segmentation with both closed and open boundaries. (e) A closed boundary contains an external arc that is partially located outside the image perimeter. (f) Cropped by the image perimeter, the proposed method produces an open convex boundary.

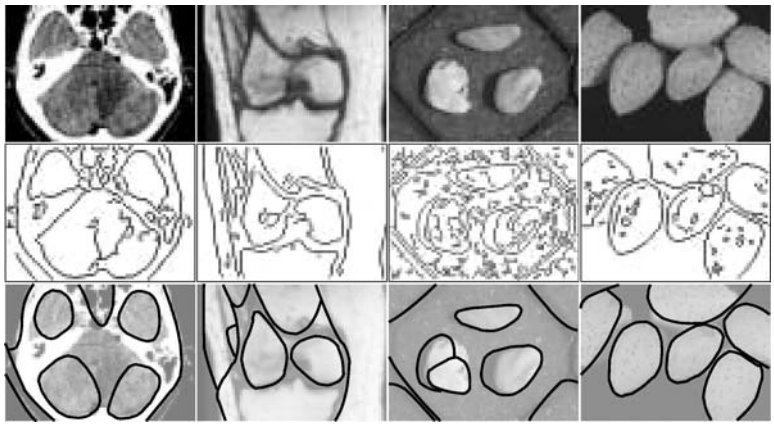

Figure 11. Hierarchical image segmentation on four real images using CRC. The top row shows the original images. The middle row shows the Canny edge detection outputs. The bottom row shows the hierarchical image segmentation results by iteratively extracting convex open/closed boundaries.

image perimeter is less than 20 percent of the longer side of the image. To detect open boundaries in an irregular-shape region, the only difference is to use region perimeter instead of image perimeter in constructing the external arcs. Figure 11 demonstrates the hierarchical image segmentation on four real images using this extended CRC ($\lambda = 10$). We can see that, in all four examples, both open and closed boundaries are detected in some steps.

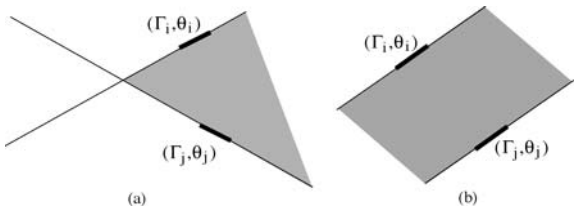

Figure 12. An illustration of the arc pruning in detecting a convex boundary that is required to traverse two specified fragments (Γ_i, θ_i) and (Γ_j, θ_j).

8.2. Human-Guided Convex Boundary Detection

Another advantage of CRC is its capability to incorporate some simple human-computer interactions that can help detect the desired boundaries. In this section, we consider the following two kinds of human-computer interactions: (a) interactively specify some fragments (points) and then find the optimal convex boundary that traverses these fragments, and (b) interactively specify a point and then find the optimal convex boundary that surrounds this specified point.

For the first problem, let's first consider a simple case: specifying only one fragment manually and finding the optimal convex boundary that traverses this fragment. This is exactly Problem 2 that is formulated in Section 4. In this case, we can set the specified fragment to be (Γ_i, θ_i), prune arcs that are not convex w.r.t (Γ_i, θ_i), and then run RC on the remaining arcs to achieve the optimal boundary. This type of human interaction speeds up the grouping because we do not need to repeat RC over each fragment (Γ_i, θ_i) as discussed in Section 4. Certainly, in practice we should allow certain error when specifying a fragment manually. Figures 13(a) and (b) demonstrate two examples of the CRC-based boundary detection with a specified fragment (labelled by a cross).

Similarly, we can specify multiple fragments and then find the optimal convex boundary that traverses all of them. To achieve this goal, we first prune all the arcs (and fragments) that are not located inside the region bounded by these fragments. Figure 12 shows an example where two fragments, (Γ_i, θ_i) and (Γ_j, θ_j), are specified. The two lines coincident with these two fragments partition the plane into four regions, as shown in Fig. 12(a), or three regions when they are parallel to each other, as shown in Fig. 12(b). We can see that only arcs and fragments located in the shaded region (the region bounded by the two specified fragments) can be present in the resulting convex boundary. Therefore, we can prune all the arcs and fragments that are located outside the shaded region. Second, we prune all

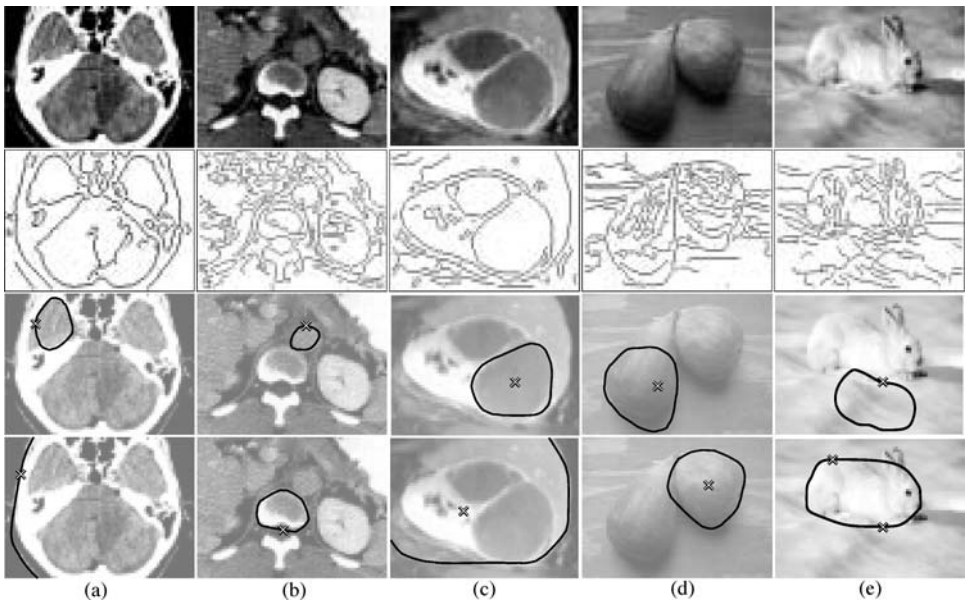

Figure 13. Some examples of human-guided boundary detection. The first row shows the original images. The second row shows the Canny edge-detection outputs. The third and fourth rows show the detected boundaries with various human-computer interactions.

the arcs that are not convex w.r.t any one of these specified fragments. Lemma 4 can then be easily extended to show that, with these two steps of arc pruning, the resulting boundary from the prototype edge-grouping algorithm must be convex and traverse all the specified fragments. The fourth row of Fig. 13(e) demonstrates an example of CRC-based convex grouping with two specified fragments. We can see that specifying a second fragment may produce a more desirable boundary than only specifying one fragment, as shown in the third row of Fig. 13(e). With this extension, we can interactively specify additional fragments until we get the desired boundary.

For the second problem where our goal is to detect a boundary around a specified point, we need to add another arc-pruning step before solving Problem 1. In this additional step, we remove all the arcs that are not convex w.r.t to this specified point. After this additional arc pruning, we run CRC as usual to find the optimal convex boundary. The correctness of this operation can be proved by the following two facts. (a) Since all the arcs along the resulting convex boundary must be convex w.r.t to this specified point, this specified point must be located inside the resulting convex boundary. (b) This additional arc-pruning step does not exclude any valid convex boundary that surrounds this specified point from the grouping search space. Note that, in this human-guided boundary detection, we still need to repeat the prototype RC over each fragment (Γ_i, θ_i) as discussed in Section 4. If we only run RC once on the remaining fragments and arcs after this additional arc-pruning step, we cannot exclude the non-simple boundaries shown in Fig. 3(c) where the specified point may be located inside the inner loop of the boundary. Figures 13(c) and (d) demonstrates two examples of CRC grouping around some specified points.

8.3. *Without Continuity Preference*

In some real applications, the structure of interest is convex but its boundary may not be very smooth, i.e., it may contain some high-curvature points. In this case, we may want to turn off the continuity (i.e., smoothness) preference and only consider the proximity, convexity, and closure in grouping, as in Jacobs' algorithm (Jacobs, 1996b). In CRC, boundary continuity can be easily turned off by setting the regularization factor $\lambda = 0$ in the cost function (1). In the sixth column (from the left) of Fig. 9, we show the grouping results using CRC with $\lambda = 0$. Comparing the results in the sixth and ninth columns, we can see that the results from Jacobs' algorithm and CRC with $\lambda = 0$ share similar problems of detecting very small or thin structures that

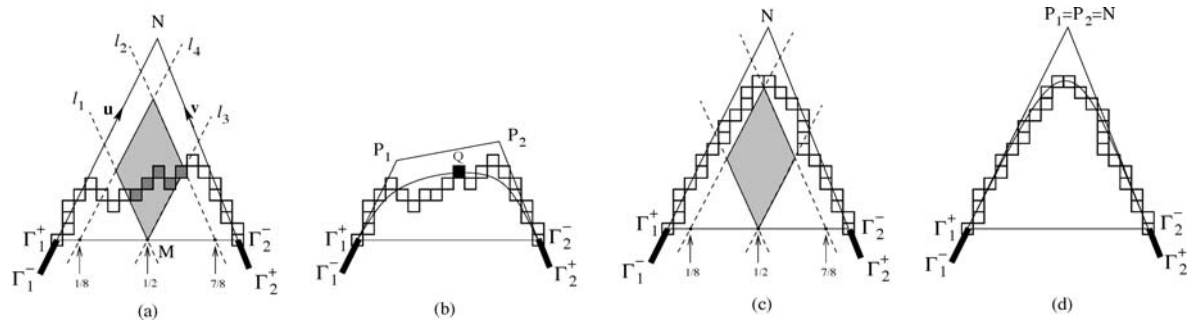

Figure 14. An illustration of selecting the point Q and constructing a detected arc. (a) The parallelogram indicates the valid region of selecting the point Q. Small squares in this figure represent the detected pixels along the subtrack. (b) The point Q is selected to be a detected pixel in the valid region and with the farthest distance to the line $\overline{\Gamma_1\Gamma_2}$. Note that the zigzag curve formed by sequentially connecting the track pixels is nonconvex, but the constructed arc is convex. (c) A special case where there is no detected pixels in the valid parallelogram region. (d) We pick $P_1 = P_2 = N$ in this special case for the arc construction.

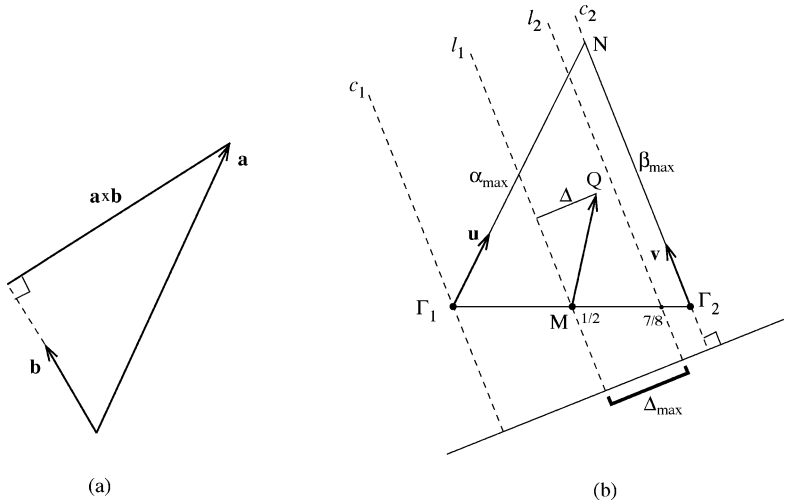

Figure 15. An illustration of deriving the valid parallelogram region for selecting the point Q. (a) Geometric explanation of the 2D scalar product. (b) The derivation of the four sides of the parallelogram region.

may not be salient from a global perspective, as illustrated by the results on the images shown in Figs. 9(a), (b), (d), (f), (i), and (m).

One way to solve this problem is to exclude the continuity term from the cost while introducing a preference toward longer closed boundaries. Certainly, we expect a resulting new cost function to have the similar format as the cost (1) so that the optimal boundary can still be found using RC algorithm. In this paper, we propose to use the following new cost function to achieve this goal:

$$R(B) = \frac{\int_0^{L(B)} [\sigma(t) + \lambda \cdot |\kappa(t)|] dt}{L(B)}. \qquad (2)$$

Different from the cost (1), this new cost function uses $|\kappa(t)|$, the absolute value of the curvature, to replace $\kappa^2(t)$.

Since the solution is constrained to convex boundaries and the boundary is arc-length parameterized, we have $\int_0^{L(B)} |\kappa(t)| dt = 2\pi$, which is a constant. Therefore, this new cost function can be rewritten as

$$R(B) = \frac{\int_0^{L(B)} \sigma(t) dt + 2\pi\lambda}{L(B)},$$

which does not have an explicit continuity term but has a preference toward longer boundaries. The new cost function (2) is of the same format as the cost (1), and

therefore, can be described by the same graph model as in RC and optimized by the same arc-pruning and RC algorithms. The only difference is to use $|\kappa(t)|$ instead of $\kappa^2(t)$ in constructing the graph model for RC (Wang et al., 2005). The seventh column of Fig. 9 shows the CRC grouping results based on this new cost (2) with $\lambda = 10$. By using this new cost function, longer boundaries are usually obtained, as shown in all the results in this column, while high-curvature points may be included in the resulting boundaries, as illustrated in the images shown in Fig. 9(b), (i) and (n). While in principle this extension can be used to detect boundaries with very high-curvature points or even corner-like points, our current CRC implementation may not handle this well because the arcs constructed as Bezier curves are always smooth and the adopted numerical algorithm may not estimate the curvature-integration term accurately in Eq. (2) around a corner-like point.

9. Conclusions

This paper presented a new general approach to enforce convexity to various edge-grouping algorithms that detects closed boundaries. This is achieved by an arc-pruning algorithm, which removes some fragments and fragment connections so that, on the remaining ones, a prototype edge-grouping algorithm without the convexity constraint can only produce convex closed boundaries. Based on this approach, this paper developed a convex-grouping algorithm, called convex ratio contour (CRC), which takes the ratio-contour algorithm (RC) as the prototype edge-grouping algorithm. CRC is able to detect convex salient boundaries with good continuity and proximity in a globally optimal fashion. To facilitate the application of CRC, we developed a new fragment-connection method based on four-point Bezier curves, which is suitable for detecting smooth boundaries as assumed in RC and CRC. Experiments on synthetic data and real images show that (a) CRC improves the performance of RC when the structures of interest are convex, and (b) by considering the continuity, CRC avoids detecting very small or thin structures as in Jacobs' convex-grouping algorithm. In summary, the proposed CRC has several properties: (a) it always produces smooth convex boundaries, which may be useful in some applications. (b) it produces the globally optimal boundaries in polynomial time in the worst case, and (c) it can be easily extended to incorporate some useful human-computer interactions, such as detecting the optimal boundary that traverses or surrounds some manually selected points. We also discussed several important extensions of CRC, including the detection of convex open boundaries and region-based image hierarchies.

Appendix: Bezier-Curve Based Arc Construction

The Bezier curve shown in Fig. 4(a) interpolates at Γ_1^+ and Γ_2^-, and the tangent vectors at Γ_1^+ and Γ_2^- are $\mathbf{r}_1 = 3(P_1 - \Gamma_1)$ and $\mathbf{r}_2 = 3(\Gamma_2 - P_2)$, respectively. Since the resulting arc is required to have G^1-continuity at Γ_1^+ and Γ_2^-, P_1 and P_2 must lie on the straight lines coincident with the fragments (Γ_1, θ_1) and (Γ_2, θ_2) respectively. In addition, since this arc is expected to interpolate at the endpoints Γ_1^+ and Γ_2^-, we know that P_1 and Γ_1^- must be located on the opposite sides of Γ_1^+, and P_2 and Γ_2^+ must be located on the opposite sides of Γ_2^-, as shown in Fig. 4(a). For simplicity, we denote $\mathbf{u} = \frac{\mathbf{r}_1}{\|\mathbf{r}_1\|}$ and $\mathbf{v} = -\frac{\mathbf{r}_2}{\|\mathbf{r}_2\|}$ as the unit vectors pointing from Γ_1 to P_1 and from Γ_2 to P_2 respectively. This way, we have

$$\begin{aligned} P_1 &= \Gamma_1 + \alpha\mathbf{u} \\ P_2 &= \Gamma_2 + \beta\mathbf{v}, \end{aligned} \tag{3}$$

and for constructing this arc, we only need to determine the two scalars $\alpha \geq 0$ and $\beta \geq 0$.

Detected-Arc Construction. In constructing a detected arc, our goal is to construct a Bezier curve to approximate a subtrack, or a sequence of connected pixels, between two fragments. In this paper, we take advantage of the subdivision property of the Bezier curve to achieve this goal. Our basic idea is to select a pixel Q from this detected subtrack and then set this point $Q = \frac{1}{8}(\Gamma_1 + 3P_1 + 3P_2 + \Gamma_2)$. From the subdivision property, we know that the point Q is always located on the resulting Bezier curve. In general, this provides us two equations (for x and y coordinates, respectively), which can be used to solve for the two unknown variables α and β. This way, we locate the additional control points P_1 and P_2, and thereby construct a detected arc.

An important issue in the detected-arc construction is that the constructed arc must be convex itself. Otherwise, it will not be incorporated into convex grouping. Note that the zigzag curve formed by directly connecting the pixels along a track is usually nonconvex, even if the underlying boundary is convex. We expect that the detected-arc construction can filter out the noise

in edge detection and recover the convexity of the detected arcs. In this paper, we impose the following constraints to achieve the arc convexity. First, as shown in Fig. 14(a), we construct an arc between Γ_1^+ and Γ_2^- only if the fragments (Γ_1, θ_1) and (Γ_2, θ_2) can be extended to intersect each other at a point N along the positive direction of $\mathbf{u}$ and $\mathbf{v}$. Second, we set $\alpha_{\max} = |\overline{N\Gamma_1}|$ and $\beta_{\max} = |\overline{N\Gamma_2}|$, and constrain the desired $\alpha \in [0, \alpha_{\max}]$ and $\beta \in [0, \beta_{\max}]$. This way, the four control points Γ_1, P_1, P_2, and Γ_2 constitute a convex quadrangle, which guarantees the convexity of the constructed detected arc, according to the variation-diminishing property of Bezier curves (Boehm et al., 2002).

The next problem is then how to select a valid point Q, which leads to $\alpha \in [0, \alpha_{\max}]$ and $\beta \in [0, \beta_{\max}]$. This is illustrated in Fig. 14(a), where lines l_1 and l_2 are parallel to $\overline{N\Gamma_2}$ and the distances from l_1 and l_2 to $\overline{N\Gamma_2}$ are one-half and one-eighth of the distance from Γ_1 to $\overline{N\Gamma_2}$, respectively. Similarly, lines l_3 and l_4 are parallel to $\overline{N\Gamma_1}$ and the distances from l_3 and l_4 to $\overline{N\Gamma_1}$ are one-half and one-eighth of the distance from Γ_2 to $\overline{N\Gamma_1}$, respectively. We then have the following lemma:

Lemma 2. *If we select a point Q from the shaded parallelogram bounded by l_1, l_2, l_3, and l_4, as shown in Fig. 14(a), we can achieve $\alpha \in [0, \alpha_{\max}]$ and $\beta \in [0, \beta_{\max}]$ as*

$$\alpha = \frac{8}{3}\left(\frac{\overrightarrow{MQ} \times \mathbf{v}}{\mathbf{u} \times \mathbf{v}}\right), \quad \beta = \frac{8}{3}\left(\frac{\overrightarrow{MQ} \times \mathbf{u}}{\mathbf{v} \times \mathbf{u}}\right),$$

where M is the mid-point between Γ_1 and Γ_2, and $\mathbf{a} \times \mathbf{b} = a_x b_y - a_y b_x$ is the 2D scalar cross product between vectors $\mathbf{a} = (a_x, a_y)$ and $\mathbf{b} = (b_x, b_y)$.

Proof: According to Eq. (3), we can easily get

$$Q = \frac{1}{8}(4\Gamma_1 + 3\alpha\mathbf{u} + 3\beta\mathbf{v} + 4\Gamma_2),$$

where $\mathbf{u}$ and $\mathbf{v}$ are unit vectors. Given $\Gamma_i = (x_i, y_i)$, $\mathbf{u} = (u_x, u_y)$, $\mathbf{v} = (v_x, v_y)$, and $Q = (Q_x, Q_y)$, the above equation can be converted to

$$\frac{3}{8}\begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} Q_x - \frac{1}{2}(x_1 + x_2) \\ Q_y - \frac{1}{2}(y_1 + y_2) \end{bmatrix}.$$

Given $M = (M_x, M_y) = \frac{1}{2}(\Gamma_1 + \Gamma_2)$ be the mid-point between these two fragments, we can derive the unknowns α and β as

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{8}{3}\left(\frac{1}{u_x v_y - u_y v_x}\right)\begin{bmatrix} v_y & -v_x \\ -u_y & u_x \end{bmatrix}\begin{bmatrix} Q_x - M_x \\ Q_y - M_y \end{bmatrix}.$$

To explain it geometrically, we can rewrite it using the *2D scalar cross product* (i.e., $\mathbf{a} \times \mathbf{b} = a_x b_y - a_y b_x$) as

$$\alpha = \frac{8}{3}\left(\frac{\overrightarrow{MQ} \times \mathbf{v}}{\mathbf{u} \times \mathbf{v}}\right), \quad \beta = \frac{8}{3}\left(\frac{\overrightarrow{MQ} \times \mathbf{u}}{\mathbf{v} \times \mathbf{u}}\right).$$

A 2D scalar cross product $\mathbf{a} \times \mathbf{b}$ in fact combines two operations: (a) rotating $\mathbf{a}$ by 90° in a counterclockwise way, and (b) finding the dot product between this rotated vector and $\mathbf{b}$. When $\mathbf{b}$ is a unit vector, $\mathbf{a} \times \mathbf{b}$ represents the length of the projection line from $\mathbf{a}$ to $\mathbf{b}$, as shown in Fig. 15(a). If $\mathbf{a}$ is located on the right side of $\mathbf{b}$ given the same origin point, we have $\mathbf{a} \times \mathbf{b} > 0$. Otherwise, we have $\mathbf{a} \times \mathbf{b} \leq 0$.

Let's further write $\alpha_{\max} = |\overline{N\Gamma_1}|$ and $\beta_{\max} = |\overline{N\Gamma_2}|$ in the form of 2D scalar cross product. We can see that

$$\alpha_{\max}\mathbf{u} - \beta_{\max}\mathbf{v} = \overrightarrow{\Gamma_1\Gamma_2},$$

from which, we achieve the following results using the same method in deriving α and β:

$$\alpha_{\max} = \frac{\overrightarrow{\Gamma_1\Gamma_2} \times \mathbf{v}}{\mathbf{u} \times \mathbf{v}}, \quad \beta_{\max} = \frac{\overrightarrow{\Gamma_2\Gamma_1} \times \mathbf{u}}{\mathbf{v} \times \mathbf{u}}.$$

Since we want $\alpha \in [0, \alpha_{\max}]$, we have

$$0 \leq \frac{8}{3}\left(\frac{\overrightarrow{MQ} \times \mathbf{v}}{\mathbf{u} \times \mathbf{v}}\right) \leq \frac{\overrightarrow{\Gamma_1\Gamma_2} \times \mathbf{v}}{\mathbf{u} \times \mathbf{v}}.$$

Let's consider the case where $\mathbf{u} \times \mathbf{v} > 0$, i.e., $\mathbf{u}$ is on the right side of $\mathbf{v}$ after being moved to the same origin. In this case, we have

$$0 \leq \overrightarrow{MQ} \times \mathbf{v} \leq \frac{3}{8}(\overrightarrow{\Gamma_1\Gamma_2} \times \mathbf{v}). \tag{4}$$

Denote $\Delta = \overrightarrow{MQ} \times \mathbf{v}$ and $\Delta_{\max} = \frac{3}{8}(\overrightarrow{\Gamma_1\Gamma_2} \times \mathbf{v})$, as illustrated in Fig. 15(b), where all dashed lines are parallel to (Γ_2, θ_2), and lines l_1 and l_2 have one-half and one-eighth distances to $\overline{N\Gamma_2}$ as the fragment Γ_1 does. Equation (4) indicates that the valid point Q can only be selected from the area bounded by parallel lines l_1 and l_2. In the case where $\mathbf{u} \times \mathbf{v} < 0$, we can get the

same result. Similarly, considering the requirement of $\beta \in [0, \beta_{\max}]$ will produce the other two parallel lines l_3 and l_4 in Fig. 14(a). These four lines form a valid parallelogram region shown in Fig. 14(a). □

In practice, from all the detected track pixels in this parallelogram, we choose the one with the farthest distance to the line $\overline{\Gamma_1\Gamma_2}$ as the point Q, as shown in Fig. 14(b). One special case is that the whole subtrack is located outside this parallelogram, as shown in Fig. 14(c–d). In this case, we just simply set $P_1 = P_2 = N$ to construct a Bezier curve for this arc.

If no intersection can be found in extending (Γ_1, θ_1) and (Γ_2, θ_2) along the positive directions of **u** and **v**, we can insert a new fragment along the subtrack between Γ_1^+ and Γ_2^-. This divides the subtrack into two shorter subtracks and we then try to construct an arc for each of them. This process is repeated until the resulting subtrack can be represented by a convex arc or it is too short to be considered.

In practice, we first check the special case of the possible colinearity between two fragments before applying the above Q-point location and arc construction. Following (Jacobs, 1992), we allow a certain level of noise in determining the colinearity between two fragments. Using Fig. 4(a) as an example, we deem two fragments (Γ_1, θ_1) and (Γ_2, θ_2) to be colinear if $\angle P_1\Gamma_1\Gamma_2 < \theta_T$ and $\angle P_2\Gamma_2\Gamma_1 < \theta_T$. Here $\theta_T > 0$ is a small threshold angle. In this paper, we set $\theta_T = 10^\circ$ in all the experiments. When these two fragments are colinear, we just directly set $\alpha = 0$ and $\beta = 0$, and use a straight-line segment as an arc connecting Γ_1^+ and Γ_2^-. Straight-line arcs are treated as convex arcs in the later arc pruning and edge grouping. Strictly speaking, allowing this 10°-error in determining the fragment colinearity breaks the assumption of G^1-continuity at some fragments. However, this practical processing does improve the grouping performance given the fact that the derived fragment slope angles usually contain some random noise (Jacobs, 1992).

Gap-Filling Arc Construction. For example, let's consider constructing a gap-filling arc between fragment endpoints Γ_1^+ and Γ_2^-. To ensure the arc convexity, we only construct this gap-filling arc when P_1 and P_2 are located on the same side of the straight line $\overline{\Gamma_1\Gamma_2}$, as shown in Fig. 4(c). In this paper, we choose the variables $\alpha \geq 0$ and $\beta \geq 0$ in Eq. (3) to satisfy the following two equations,

$$\begin{cases} \alpha + \beta = |\overline{\Gamma_1\Gamma_2}| \\ \dfrac{\alpha}{\sin(\angle P_2\Gamma_2\Gamma_1)} = \dfrac{\beta}{\sin(\angle P_1\Gamma_1\Gamma_2)}. \end{cases} \tag{5}$$

Solving Eq. (5) will produce two control points P_1 and P_2 that always leads to a convex gap-filling arc. This can be proved by considering two cases. In the first case, the extension of the fragments (Γ_1, θ_1) and (Γ_2, θ_2) along the positive directions of **u** and **v** leads to an intersection point N. It is then easy to see that the solved α and β from Eq. (5) satisfies $0 \leq \alpha \leq \alpha_{\max} = |\overline{N\Gamma_1}|$ and $0 \leq \beta \leq \beta_{\max} = |\overline{N\Gamma_2}|$. Therefore, Γ_1, P_1, P_2, and Γ_2 constitute a convex quadrangle, with which the convexity of the constructed gap-filling arc is guaranteed. In the second case, the extension of the fragments (Γ_1, θ_1) and (Γ_2, θ_2) along the positive directions of **u** and **v** does not lead to an intersection point. Given $\alpha \geq 0$ and $\beta \geq 0$, four points Γ_1, P_1, P_2, and Γ_2 always constitute a convex quadrangle, with which the convexity of the constructed arc is also guaranteed. Similar to the detected-arc construction, we also check the fragment-colinearity before this Bezier-curve-based gap-filling arc construction. The same 10°-error is allowed in determining the colinearity and if the considered two fragments are colinear, we set $\alpha = 0$ and $\beta = 0$, and use a straight line segment as the gap-filling arc.

Acknowledgments

We would like to thank Stephen Fenner and David Jacobs for important help. This work was funded, in part, by NSF-EIA-0312861 and a grant from the University of South Carolina Research and Productivity Scholarship Fund. Some preliminary results of this paper were published in a conference (Stahl and Wang, 2005).

Notes

1. We do not show all the constructed gap-filling arcs in this figure. As mentioned later, in this paper we in fact construct gap-filling arcs between every two fragment endpoints that are sufficiently close to each other. Showing all these gap-filling arcs would make the figure too crowded to identify useful information.
2. In Jacobs (1992, 1996b) the problem of overly long detected arcs is solved by an additional modification of allowing a portion of a detected arc to be present in the grouping (see pp. 172–176 of Jacobs (1992) or Section 4.3 of Jacobs (1996b)). This additional

modification is easy to implement in Jacobs (1992, 1996b), because the arcs there are straight-line segments, but very difficult to implement in the proposed CRC, because CRC assumes smooth arcs/boundaries, as discussed later in this section.

3. Jacobs' convex-grouping algorithm sometimes produces boundaries that are not fully convex, as shown in Fig. 9(c). This is the result of some additions in its implementation (Jacobs, 1996a), such as allowing certain errors in determining the curve colinearity. The details can be found in the pages 172–176 of Jacobs (1992).
4. Jacobs' algorithm detects all the boundaries with a saliency $\phi_J(B)$ larger a preset threshold $\phi_T \in [0, 1]$. Jacobs (1996b) suggests this threshold to be selected in the range [0.6, 0.9]. In our experiments, we choose $\phi_T = 0.7$.

References

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

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

Bartels, R., Beatty, J., and Barsky, B. 1987. *An Introduction to Splines for use in Computer Graphics and Geometric Modelling*. Morgan Kaufmann, Los Altos.

Bertamini, M. 2001. The importance of being convex: An advantage for convexity when judging position. *Perception*, 30:1295–1310.

Boehm, W., Paluszny, M., and Prautzsch, H. 2002. *Bezier and B-Spline Techniques*. Springer-Verlag, Berlin.

Borra, S. and Sarkar, S. 1997. A framework for performance characterization of intermediate-level grouping modules. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1306–1312.

Bruckstein, A. and Netravali, A. 1990. On minimal energy trajectories. *Computer Vision, Graphics, and Image Processing*, 49:283–296.

Cormen, T.H., Leiserson, C.E., and Rivest, R.L. 1990. *Introduction to Algorithms*. Cambridge: MIT Press/New York: McGraw Hill.

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

Elder, J. and Zucker, S. 1996. Computing contour closure. In *European Conference on Computer Vision*, pp. 399–412.

Estrada, F. and Jepson, A. 2004a. Perceptual grouping for contour extraction. In *International Conference on Pattern Recognition* Vol. 2, pp. 32–35.

Estrada, F. and Jepson, A. 2004b. Controlling the search for convex groups. Technical Report CSRG-482, Department of Computer Science, University of Toronto.

Foley, J.D., Dam, A., Feiner, S.K., and Hughes, J.F. 1995. *Computer Graphics: Principles and Practice in C*. 2nd Edn. Reading, MA: Addison Wesley.

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

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

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

Jacobs, D. 1987. GROPER: A grouping based object recognition system for two-dimensional objects. In *Proceedings of IEEE Workshop on Computer Vision*, pp. 164–169.

Jacobs, D. 1992. Recognizing 3D objects using 2D images. Technical Report MIT AI-1416, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Jacobs, D. 1996a. Convex grouping code. http://www.cs.umd.edu/djacobs/convex-grouping.tar.

Jacobs, D. 1996b. Robust and efficient detection of convex groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):23–37.

Kanisza, G. and Gerbino, W. 1976. Convexity and symmetry in figure-ground organization. In *Vision and Artifact*. M. Henle, (Eds.), New York: Springer.

Liu, Z., Jacobs, D., and Basri, R. 1999. The role of convexity in perceptual completion: Beyond good continuation. *Vision Research*, 39:4244–4257.

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

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

Mio, W., Srivastava, A., and Klassen, E. 2004. Interpolations with elasticae in euclidean spaces. *Quarterly of Applied Mathematics*, 62(2):359–378.

Mumford, D. 1994. Elastica and computer vision. In *Algebraic Geometry and Its Applications*, C. Bajaj, (Ed.), Springer Verlag, pp. 491–506.

Nattkemper, T.W. 2004. Automatic segmentation of digital micrographs: A survey. In *Proceedings of 11th World Congress on Medical Informatics (MEDINFO)*, San Franzisco, USA.

Parvin, B. and Viswanathan, S. 1995. Tracking of convex objects. In *Proceedings of the International Symposium on Computer Vision*, pp. 295–298.

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

Sarkar, S. and Boyer, K.L. 1994. *Computing Perceptual Organization in Computer Vision*. Singapor: World Scientific.

Saund, E. 2003. Finding perceptually closed paths in sketches and drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):475–491.

Saund, E. and Moran, T. 1995. Perceptual organization in an interactive sketch editing applications. In *International Conference on Computer Vision*, pp. 597–604.

Sharon, E., Brandt, A., and Basri, R. 2000. Completion energies and scale. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1117–1131.

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

Srivastava, A., Mio, W., Klassen, E., and Liu, X. 2003. Geometric analysis of constrained curves for image understanding. In *Proceedings of 2nd IEEE Workshop on Variational, Geometric, and Level Set Methods (VLSM) in Vision*, Nice, France.

Stahl, J. and Wang, S. 2005. Convex grouping combining boundary and region information. In *International Conference on Computer Vision*, Beijing, China, pp. II: 946–953.

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

Wang, S., Kubota, T., and Siskind, J.M. 2003. Salient boundary detection using ratio contour. In *Neural Information Processing Systems Conference*, pp. 1571–1578.

Wang, S., Wang, J., and Kubota, T. 2004. From fragments to salient closed boundaries: An in-depth study. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. II:291–298.

Wertheimer, M. 1938. Laws of organization in perceptual forms (partial translation). In *A Sourcebook of Gestalt Psychology*, W.D. Ellis, (Ed.), New York: Harcourt, Brace, pp. 71–88.

Williams, L. and Jacobs, D. 1997. Stochastic completion fields: A neural model of illusory contour shape and salience. *Neural Computation*, 9:849–870.

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