# Image Segmentation with Ratio Cut

Song Wang, *Member*, *IEEE*, and Jeffrey Mark Siskind, *Member*, *IEEE*

**Abstract**—This paper proposes a new cost function, cut ratio, for segmenting images using graph-based methods. The cut ratio is defined as the ratio of the corresponding sums of two different weights of edges along the cut boundary and models the mean affinity between the segments separated by the boundary per unit boundary length. This new cost function allows the image perimeter to be segmented, guarantees that the segments produced by bipartitioning are connected, and does not introduce a size, shape, smoothness, or boundary-length bias. The latter allows it to produce segmentations where boundaries are aligned with image edges. Furthermore, the cut-ratio cost function allows efficient iterated region-based segmentation as well as pixel-based segmentation. These properties may be useful for some image-segmentation applications. While the problem of finding a minimum ratio cut in an arbitrary graph is NP-hard, one can find a minimum ratio cut in the connected planar graphs that arise during image segmentation in polynomial time. While the cut ratio, alone, is not sufficient as a baseline method for image segmentation, it forms a good basis for an extended method of image segmentation when combined with a small number of standard techniques. We present an implemented algorithm for finding a minimum ratio cut, prove its correctness, discuss its application to image segmentation, and present the results of segmenting a number of medical and natural images using our techniques.

**Index Terms**—Graph partitioning algorithms, cut ratio, cycle ratio, perfect matching, perceptual organization, edge detection, image segmentation, machine vision.

## 1 Introduction

There has been significant interest in graph-based approaches to image segmentation in the past few years [1], [2], [3], [4], [5], [6], [7], [8], [9]. The common theme underlying these approaches is the formation of a weighted graph where each vertex corresponds to an image pixel or region and each edge is weighted with some measure of the desire for the pixels or regions connected by that edge to be in the same segment. This graph is partitioned into components in a way that minimizes some specified cost function of the vertices in the components and/or the boundary between those components. One way of partitioning a graph into more than two components is to recursively bipartition the graph until some termination criterion is met. Often, the termination criterion is based on the same cost function that is used for bipartitioning.

Wu and Leahy [1] were the first to introduce the general approach of segmenting images by way of optimally partitioning an *undirected* graph using a global cost function. They minimized a cost function formulated as a boundary-cost metric, the sum of the edge weights along a cut boundary. They used *minimum cut*, a polynomial-time algorithm for finding optimal bipartitions with this cost function. This cost function, however, has a bias toward short boundaries. Cox et al. [2] attempted to alleviate this bias by normalizing the boundary-cost metric. They proposed a cost function, *ratio regions*, formulated as a *ratio* between a boundary-cost metric and a segment-area metric. They also gave a polynomial-time algorithm for finding optimal bipartitions in an undirected graph with this cost function. Shi and Malik [3], [7] and Sarkar and Soundararajan [6], [9] adopted different cost functions, *normalized cut* and *average cut*, formulated as sums of two ratios between boundary-cost and segment-area-related metrics, also in undirected graphs. Ratios between boundary costs and segment areas have a bias toward larger rounder areas and shorter smoother boundaries. Jermyn and Ishikawa [4], [5] were the first to adopt a cost function formulated as a ratio of two different boundary-cost metrics in a *directed* graph. They gave polynomial-time algorithms for finding bipartitions that minimize the ratio of the sums of two different edge-weight functions in such a directed graph. Cost functions formulated as the ratio of two boundary-cost metrics can alleviate area-related biases in appropriate circumstances. In this paper, we present a new cost function, *cut ratio*, namely, the ratio of the corresponding sums of two different weights associated with edges along the cut boundary in an *undirected* graph. Cut ratio generalizes the *mean cut* cost function introduced by Wang and Siskind [10], namely, the mean edge weight of the cut boundary in an undirected graph. We also generalize the polynomial-time algorithm presented by Wang and Siskind [10] for minimizing the mean-cut cost function to yield a polynomial-time algorithm for finding a cut that minimizes the cut-ratio cost function. This generalization allows us to perform efficient iterated region-based segmentation.

The approach discussed in this paper exhibits the following collection of properties:

1. It allows the image perimeter to be segmented.
2. It guarantees that the components produced by bipartitioning are connected.
3. It does not introduce a size, shape, smoothness, or boundary-length bias. The lack of bias allows it to

- *S. Wang is with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208. E-mail: songwang@cse.sc.edu.*
- *J.M. Siskind is with the School of Electrical and Computer Engineering, Purdue University, 1285 Electrical Engineering Building, West Lafayette, IN 47907-1285. E-mail: qobi@purdue.edu.*

*Manuscript received 8 Nov. 2001; revised 9 July 2002; accepted 24 Oct. 2002. Recommended for acceptance by D. Jacobs and M. Lindenbaum. For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 117707.*

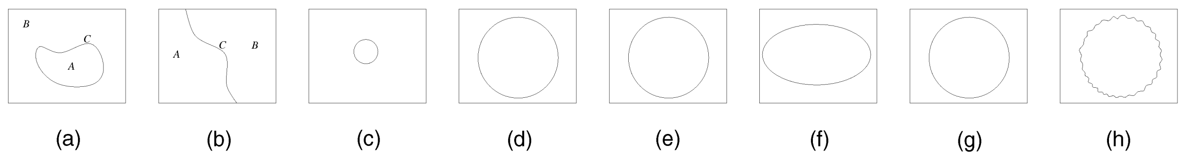


Fig. 1. Various two-segment image segmentations. (a) The image perimeter lies within a single segment. (b) The image perimeter lies within both segments. The segments in (c) and (d) differ in size though they exhibit similar shape and smoothness. The segments in (e) and (f) differ in shape though they exhibit similar size and smoothness. The segments in (g) and (h) differ in smoothness though they exhibit similar size and shape. Our method adopts no prior preference between these alternate segmentations.

produce segmentations where boundaries are aligned with image edges.

4. An optimal bipartition can be found in polynomial time.

These properties may be useful for some image-segmentation applications. We illustrate many of these properties by way of real images in Sections 5, 6, and 8.

Let us elaborate on the first of the above properties. By the ability to segment the image perimeter, we mean that our method can not only segment an image into segments where the image perimeter lies totally within one segment, as illustrated in Fig. 1a, it can also segment an image into segments where the image perimeter lies in one or more segments, as illustrated in Figs. 1a and 1b. This ability is shared by some other approaches to graph-based image segmentation, for example, minimum cut [1], normalized cut [3], [7], average cut [6], [9], and Jermyn and Ishikawa [5]. This ability allows a segment to correspond to an object that is partially occluded or partially outside the field of view. It is also important when recursively bipartitioning an image because one step of recursive bipartitioning may partition a pair of objects as one segment and a subsequent step may separate these two objects. The boundary between these two objects in the later bipartitioning step might touch the perimeter of the segment produced in the earlier bipartitioning step. The fact that our approach can segment an image into segments where the image perimeter lies in one or more segments follows from the reduction that will be discussed in Section 3.1.

Let us elaborate on the second of the above properties. The fact that components produced by bipartitioning are guaranteed to be connected is shared by some other approaches to graph-based image segmentation, for example, minimum cut [1] and Jermyn and Ishikawa [4], [5]. We prove that our method has this property in Section 3.1.

Let us elaborate on the third of the above properties. By lack of size bias, we mean that our method has no preference for larger-area segments over smaller-area ones or vice versa. By lack of shape bias, we mean that our method has no preference for segments with specific shape, such as rounder segments over elongated ones. By lack of smoothness bias, we mean that our method has no preference for boundaries with low average curvature over those with high average curvature. For example, our method will exhibit no preference between the segments in Figs. 1c, 1d, 1e, 1f, 1g, and 1h. This lack of bias is shared by some other approaches to graph-based image segmentation, for example, the method of Jermyn and Ishikawa [4], [5] when the intensity gradient is consistently oriented inward or outward. The fact that our approach lacks such biases follows from the formulation of our cost function: It does not incorporate any area metric and incorporates only a normalized boundary-length metric.

Finally, let us elaborate on the fourth of the above properties. The ability to find an optimal bipartition in polynomial time is shared by some other approaches to graph-based image segmentation, for example, minimum cut [1], ratio regions [2], and Jermyn and Ishikawa [4], [5]. We present our polynomial-time algorithm for finding a global minimum ratio cut in Section 3.

Throughout this paper, we use the term *ratio cut* to refer generically to the cut-ratio cost function and the associated optimization methods. We present two methods for image segmentation using the cut-ratio cost function. The *baseline* method, presented in Section 5, forms graphs where the vertices correspond to pixels and weights are taken to be some decreasing function of the intensity difference between adjacent pixels. This method recursively bipartitions an image along the minimum ratio cut, terminating when the segments are sufficiently homogeneous. We use the cut-ratio cost function as a measure of segment homogeneity. While the baseline method exhibits the properties discussed above, it has certain disadvantages. It is sensitive to salt-and-pepper noise and blurry edges, it can produce spurious cuts, and it can be slow in practice even though we use a polynomial-time algorithm. Sensitivity to noise and blurry edges, as well as the spurious-cut problem, result, in part, from the fact that our edge weights are pixel-based rather than region-based. To address these disadvantages, we present an *extended* method in Section 6. This method adds the following extensions: First, we postprocess the result to merge very small segments into neighboring segments, as discussed in Section 6.1. Second, we use the same cut-ratio cost function to partition a graph whose vertices correspond to image regions instead of image pixels to incorporate region information into the edge weights. This process can be iterated, performing an initial segmentation based on pixels and having subsequent iterations use the segments produced by the previous iteration as the vertices in the graph. This is discussed in Section 6.2. Finally, we show a blocking heuristic for quickly segmenting a large image by combining the segmentation results for subimages. This is discussed in Section 6.3. While this heuristic compromises the provable optimality of the solution, in practice, we find that it does not appreciably reduce the quality of the resulting segmentations.

The remainder of this paper is organized as follows: Section 2 presents a definition of the cut-ratio cost function. Section 3 presents a polynomial-time algorithm for finding a minimum ratio cut in a connected planar graph. Section 4 shows that the problem of finding a minimum ratio cut in an arbitrary graph is NP-hard. Section 5 discusses the baseline method for using ratio cut for image segmentation. Section 6 discusses the extended method. Section 7 discusses some of the details of our implementation. Section 8 presents the results of applying the extended method to various images and compares these results to normalized cut [11].[1] Section 9 summarizes the main contributions of this work.

## 2 Background

We formulate 2D image segmentation as a process of bipartitioning a weighted undirected graph $G = (V, E)$ with nonempty sets of vertices and edges. $G$ will correspond to the image, the vertices will correspond to regions, and the edges will correspond to adjacency relations between these regions. We perform iterated region-based segmentation where the regions in the first iteration contain individual pixels and may contain multiple pixels in subsequent iterations. We refer to the graph used in the first iteration as the *underlying* graph. Each vertex in the graphs used in subsequent iterations corresponds to a set of vertices in the underlying graph. If $G$ is a graph in a subsequent iteration, then $U(G)$ denotes the graph underlying $G$. If $u$ is a vertex in $G$, let $U(u)$ denote the subset of vertices in $U(G)$ that corresponds to $u$.

A *cut* $(A, B)$ of $G$ is a partition of $V$ into two disjoint nonempty sets $A$ and $B$. A cut corresponds to a segmentation while $A$ and $B$ correspond to segments. We refer to the set of edges between $A$ and $B$ as the *boundary* of $(A, B)$. This corresponds to the segmentation boundary in the image. A *path* is an alternating sequence of vertices and edges, beginning and ending with a vertex, such that each edge connects the preceding vertex to the following vertex. A *cycle* is a path that starts and ends at the same vertex. A cycle is *simple* if it does not contain a vertex more than once, except for the starting and ending vertices. A cycle is *degenerate* if it contains an edge more than once. For the remainder of this paper, we use the term "cycle" without a qualifier to mean simple nondegenerate cycle. For simplicity, we treat a cycle $C$ as a sequence or set of edges, implicitly omitting the vertices and possibly ignoring the order.

The graphs that we construct have a pair of weights $w_1(u, v)$ and $w_2(u, v) > 0$ associated with each edge $(u, v)$. We refer to $w_1(u, v)$ as the *(first) (edge) weight*, to $w_2(u, v)$ as the *second (edge) weight*, and to $\frac{w_1(u,v)}{w_2(u,v)}$ as the *(edge-) weight ratio*. Let us adopt the following notation:

$$c_1(A, B) \triangleq \sum_{u \in A, v \in B, (u,v) \in E} w_1(u, v)$$

$$c_2(A, B) \triangleq \sum_{u \in A, v \in B, (u,v) \in E} w_2(u, v)$$

$$c_1(C) \triangleq \sum_{(u,v) \in C} w_1(u, v)$$

$$c_2(C) \triangleq \sum_{(u,v) \in C} w_2(u, v).$$

We refer to $c_1(A, B)$ as the *(first) boundary cost*, $c_2(A, B)$ as the *second boundary cost*, $c_1(C)$ as the *(first) cycle cost*, and $c_2(C)$ as the *second cycle cost*.

The first edge weight denotes the affinity between two regions, some measure of the desire for those two regions to be in the same segment. Thus, the first boundary cost corresponds to the total affinity between the regions in one segment and the regions in another segment. In the first iteration, we take the second edge weight to be unity since each vertex corresponds to a single pixel. In subsequent iterations, we take the second edge weight between $u$ and $v$ to be the number of edges between $U(u)$ and $U(v)$ in $U(G)$. Thus, the second edge weight corresponds to the length of the boundary that separates the two regions and the second boundary cost corresponds to the length of the segmentation boundary. The edge-weight ratio thus corresponds to the average affinity per unit boundary length between the two regions.

We seek a bipartition that minimizes the *cut-ratio* cost function formulated as follows:

$$\mathrm{Rcut}(A, B) \triangleq \frac{c_1(A, B)}{c_2(A, B)}.$$

We refer to a cut with the smallest cut ratio as a *minimum ratio cut* and to the cut ratio of such a cut as the *minimum cut ratio*. The cut-ratio cost function generalizes the *mean-cut* cost function, $\mathrm{Mcut}(A, B)$, presented in Wang and Siskind [10]. $\mathrm{Mcut}(A, B)$ is the same as $\mathrm{Rcut}(A, B)$ when $w_2(u, v) = 1$. Wang and Siskind [10] presented a polynomial-time algorithm for finding a minimum mean cut. In this paper, we generalize that algorithm to find a minimum ratio cut in polynomial time.

We use the greater generality of the cut-ratio cost function to perform iterated region-based segmentation as discussed in Section 6.2. The mean-cut cost function corresponds to the average affinity per element of the cut boundary. In the first iteration, but not subsequent iterations, this is the average affinity per unit length of the segmentation boundary. The cut-ratio cost function normalizes the first boundary cost by the second boundary cost. Since the second boundary cost corresponds to the length of the segmentation boundary, the cut-ratio cost function always corresponds to the average affinity per unit length of the segmentation boundary.

## 3 A Polynomial-Time Algorithm for Finding a Minimum Ratio Cut in Connected Planar Graphs

We now present a polynomial-time algorithm for finding a minimum ratio cut of a connected planar graph. In Section 4, we show that the problem of finding a minimum ratio cut of an arbitrary graph is NP-hard. Our method, limited to connected planar graphs, consists of three reductions: minimum ratio cut to minimum ratio cycle, minimum ratio cycle to negative-cost cycle, and negative-cost cycle to minimum-cost perfect matching. The above reductions all operate on undirected graphs. The second reduction, from minimum ratio cycle to negative-cost cycle, uses the same binary and linear search techniques, discussed by Ahuja et al. [13, pp. 150-157], where they are used to find the minimum ratio cycle in a directed graph. The third reduction, from negative-cost cycle to minimum-cost perfect matching, was also motivated by a

1. Fig. 8 compares baseline implementations of ratio cut and normalized cut where both techniques use their respective cost functions for recursive bipartitioning and the termination criterion with the same edge-weight functions and no further enhancements. Figs. 15 and 16 compare extended implementations of both ratio cut and normalized cut. This is not strictly an apples-to-apples comparison because each implementation has different extensions. In particular, the implementation of extended normalized cut that we use [12] incorporates techniques from Malik et al. [11], such as texture, that go beyond finding cuts that minimize the normalized-cut cost function. We chose to compare our extended method to this work because it is commonly viewed as reflective of the current state of the art.

similar reduction, discussed by Ahuja et al. [13, pp. 496-497], that is used to find shortest paths in undirected graphs.

### 3.1 Reducing Minimum Ratio Cut to Minimum Ratio Cycle

Let us define the *cycle ratio* of a cycle $C$ as the ratio of the first and second cycle costs:

$$\mathrm{CR}(C) \triangleq \frac{c_1(C)}{c_2(C)}.$$

We refer to a cycle with the smallest cycle ratio as a *minimum ratio cycle* and to the cycle ratio of such a cycle as the *minimum cycle ratio*. We reduce the problem of finding a minimum ratio cut to the problem of finding a minimum ratio cycle.

The reduction from minimum ratio cut to minimum ratio cycle requires that $G$ be a connected planar graph. Initially, we construct a graph corresponding to the entire image using methods that we will describe in Section 5. This initial graph will be connected and planar by construction. However, recursive application of the bipartitioning algorithm, as will be described in Section 5.1, applies this algorithm recursively to components produced by a single application of the bipartitioning algorithm. These components correspond to image segments. In order to use our algorithm to recursively bipartition a graph, we need to show that the components produced at each step in the recursion are connected and planar. The preservation of planarity is obvious. The following lemma proves the preservation of connectivity:

**Lemma 1.** *There exists a minimum ratio cut $(A, B)$ of every connected graph $G$ that satisfies the condition that both $A$ and $B$ are connected.*

(Proofs of all lemmas have been omitted to conserve space. The proofs are available in [14] and in the supplemental material in the Digital Library http://computer.org/publications/dlib.)

We say that a cut $(A, B)$ is *connected* when components $A$ and $B$ are both connected. There may be multiple minimum ratio cuts $(A, B)$. Some of these cuts may be connected while others may not be. By the above lemma, at least one of these cuts will be connected. The algorithm that we present in this section finds a connected minimum ratio cut. By induction, recursive application of this algorithm preserves planarity and connectivity.

The reduction from minimum ratio cut to minimum ratio cycle constructs a *dual* graph $\hat{G} = (\hat{V}, \hat{E})$ from $G$. The dual $\hat{G}$ is constructed from a planar embedding of $G$ by taking each face in $G$ as a vertex in $\hat{G}$ and constructing an edge $\hat{e}$ in $\hat{G}$ for every pair of faces in $G$ that share an edge $e$. Note that there can be different duals of $G$, each corresponding to a different planar embedding of $G$. This is illustrated in Fig. 2.

Each edge $\hat{e}$ in $\hat{G}$ crosses exactly one edge $e$ in the planar embedding of $G$ used to construct $\hat{G}$ and vice versa. Thus, there is a one-to-one correspondence between the edges in $G$ and the edges in any dual $\hat{G}$ of $G$. We take the weights of the edges in $\hat{G}$ to be the same as the weights of the corresponding edges in $G$. Furthermore, for any cycle $\hat{C} = \{\hat{e}_1, \ldots, \hat{e}_l\}$ in $\hat{G}$, removing the edges $\{e_1, \ldots, e_l\}$ from $E$ bipartitions $G$ into two connected components and vice versa. Thus, there is a one-to-one correspondence between connected cuts $(A, B)$ of $G$ and cycles $\hat{C}$ in any dual $\hat{G}$ of $G$. Additionally, if the cut $(A, B)$ of $G$ corresponds to the cycle $\hat{C}$ in $\hat{G}$, then $\mathrm{Rcut}(A, B) = \mathrm{CR}(\hat{C})$. Thus, we have the following lemma:

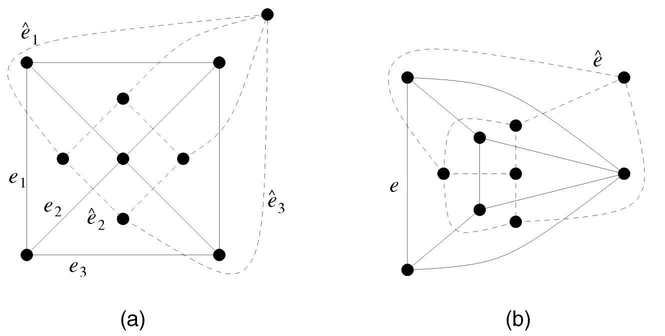


Fig. 2. Different duals of the same planar graph with different planar embeddings.

**Lemma 2.** *For every dual $\hat{G}$ of $G$, there is a one-to-one correspondence between the connected minimum ratio cuts in $G$ and the minimum ratio cycles in $\hat{G}$.*

### 3.2 Reducing Minimum Ratio Cycle to Negative-Cost Cycle

In the previous section, we reduced the problem of finding a minimum ratio cut to the problem of finding a minimum ratio cycle in an *undirected* graph. Jermyn and Ishikawa [4] use a dynamic-programming method introduced by Karp [15] for finding a cycle in a directed graph with minimum mean edge weight. Jermyn and Ishikawa [5] use a method from Ahuja et al. [13] to solve a more general problem, namely, finding a cycle in a *directed* graph with minimal cycle ratio. The methods of Karp [15] and Ahuja et al. [13] guarantee that the cycles found will be simple but those methods alone may yield degenerate cycles. The particular graphs constructed by Jermyn and Ishikawa [4], [5] further guarantee that the resulting cycles are nondegenerate.

On the surface, it might appear that we can solve our problem, namely, finding a (simple nondegenerate) cycle with minimum cycle ratio in an *undirected* graph, using the techniques that Jermyn and Ishikawa [5] present for *directed* graphs. However, this is not the case. Clearly, one cannot solve the problem for undirected graphs using the method used by Jermyn and Ishikawa [5] simply by assigning an arbitrary direction to each edge. One might consider reducing the problem for undirected graphs to one for directed graphs by constructing two directed edges $(u, v)$ and $(v, u)$ in the derived graph for each undirected edge $(u, v)$ in the original graph, assigning the weights of the two derived edges to be the same as the corresponding original edge, applying the method used by Jermyn and Ishikawa [5] to the derived graph, and mapping edges in the minimum ratio cycle found in the derived graph to the corresponding edges in the original graph. However, the result prior to mapping back will always consist of the single pair of derived edges that correspond to the original edge $(u, v)$ with minimal edge-weight ratio. This maps to a degenerate cycle in the original undirected graph.

We now reduce the problem of finding a minimum ratio cycle in an *undirected* graph to the problem of finding a negative-cost cycle. We first show the following:

**Lemma 3.** *Transforming the first edge weights of an undirected graph by the linear function $w'_1 = aw_1 - bw_2$ of the original* *edge weights $w_1$ and $w_2$ with $a > 0$ does not change its minimum ratio cycles.*

A corollary of this lemma is that such a transformation of the edge weights does not change the minimum ratio cuts of the original graph $G$. A second corollary of this lemma is that the problem of finding a cut $(A, B)$ that minimizes $\mathrm{Mcut}(A, B)$ or $\mathrm{Rcut}(A, B)$ is well-defined even when we allow the first edge weights to be negative.

Let us refer to a cycle with the smallest cycle cost as a *minimum cost cycle* and to the cost of such a cycle as the *minimum cycle cost*. The above lemma, along with the following lemma, can be used to reduce the problem of finding a minimum ratio cycle to the problem of finding a negative-cost cycle. The reduction, adapted from Ahuja et al. [13, pp. 150-57], relies on the following lemma:

**Lemma 4.** *Let $\hat{G}'(b)$ denote $\hat{G}$ with edge weights $w'_1$ derived from the edge weights $w_1$ in $\hat{G}$ by the transformation $w'_1 = w_1 - bw_2$. A graph $\hat{G}$ has a minimum ratio cycle $C$ with cycle ratio $b^*$ if and only if the minimum cycle cost of $\hat{G}'(b^*)$ is zero.*

If $\hat{G}'(b)$ has a negative-cost cycle, then $b^* < b$. Likewise, if $\hat{G}'(b)$ does not have a negative-cost cycle, then $b^* \geq b$. If we have an algorithm for determining whether a graph has a negative-cost cycle, $b^*$ can be found by binary search. Let $\underline{r}$ and $\overline{r}$ be the smallest and largest edge-weight ratios in $\hat{G}$, respectively. Initialize $\underline{b}$ to $\underline{r}$ and $\overline{b}$ to $\overline{r}$. We know that $\underline{b} \leq b^* \leq \overline{b}$. Repeatedly, let $b$ be the mean of $\underline{b}$ and $\overline{b}$. If $\hat{G}'(b)$ has a negative-cost cycle, then set $\overline{b}$ to $b$. Otherwise, set $\underline{b}$ to $b$.

We have not been able to prove that binary search must terminate with real-valued edge weights. Limiting edge weights to integers, however, guarantees termination in (pseudo)polynomial time. Given integral edge weights, the cycle ratio will be rational with a denominator in the range $1, \ldots, W_2$, where $W_2 = \sum_{(u,v)\in E} w_2(u, v)$. Thus, the denominator will always be a factor of $W_2!$. By Lemma 3, we can multiply all first edge weights by $W_2!$ without changing the minimum ratio cycles or minimum ratio cuts. Under such a scale transformation, the minimum cycle ratio of $\hat{G}$ and the minimum cut ratio of $G$ will be integers. Since the search range is now from $W_2! \cdot \underline{r}$ to $W_2! \cdot \overline{r}$, and since $W_2! < W_2^{W_2}$, binary search is now guaranteed to terminate in at most $W_2 \lg W_2 + \lg(\overline{r} - \underline{r})$ iterations. Since our application uses edge-weight functions that are bounded by a polynomial function of the size of the graph, the above is also a polynomial function of the size of the graph.

The above binary-search algorithm yields only $b^*$, the minimum cycle ratio, but not a minimum ratio cycle itself. To recover a minimum ratio cycle, we can find one negative-cost cycle for $\hat{G}$, when its edge weights are transformed by $w'_1 = W_2! \cdot w_1 - (W_2! \cdot b^* + \frac{1}{2})w_2$, using the algorithm discussed in Section 3.3. This negative-cost cycle is a desired minimum ratio cycle.

Since, as shown in Section 3.3, the negative-cost cycle can be reduced to minimum-cost perfect matching, which is polynomial time [16], [17], the above binary-search algorithm yields a polynomial-time algorithm for minimum ratio cut. While this is true in theory, it might not be practical because $W_2!$ can be large. Accordingly, our implementation uses a different technique, adapted from Ahuja et al. [13, pp. 150-157], for finding $b^*$ and a corresponding minimum ratio cycle.

Our implemented technique works as follows: Like the binary-search algorithm, we start by initializing $\overline{b}$ to $\overline{r}$. We know that $b^* \leq \overline{b}$. We use the algorithm discussed in Section 3.3 to find a a negative-cost cycle $C$ in $\hat{G}'(\overline{b})$, i.e., $c'_1(C) < 0$. Recall that the first edge weights $w'_1$ in $\hat{G}'(\overline{b})$ were derived from the edge weights $w_1$ and $w_2$ in $\hat{G}$ by the transformation $w'_1 = w_1 - \overline{b}w_2$. Since $w_2 > 0$, $c_2(C) > 0$.

$$b^* \leq \frac{c_1(C)}{c_2(C)} = \overline{b} + \frac{c'_1(C)}{c_2(C)} < \overline{b}.$$

We thus update $\overline{b}$ with $\overline{b} + \frac{c'_1(C)}{c_2(C)}$, reconstruct a new $\hat{G}'(\overline{b})$, and repeat this process until no negative-cost cycle is found. This last $\overline{b}$ is $b^*$. Note that this last $\overline{b}$ is the cycle ratio of the cycle $C$ detected in the penultimate iteration. Therefore, the desired minimum ratio cycle is the $C$ that was found in this penultimate iteration. Since the algorithm discussed in the next section can find more than one negative-cost cycle, we obviously choose $C$ to be the one with smallest $\frac{c'_1(C)}{c_2(C)}$. Also, note that no more than $|V|$ cycles can be detected in each iteration because, using the technique to be described in Section 3.3, the detected cycles cannot share vertices and edges.

The following argument shows that this linear-search technique also converges in (pseudo)polynomial time for integral edge weights. Given integral edge weights, the cycle ratio will be rational with a numerator in the range $1, \ldots, W_1$ and a denominator in the range $1, \ldots, W_2$, where $W_1 = \sum_{(u,v)\in E} w_1(u, v)$ and $W_2 = \sum_{(u,v)\in E} w_2(u, v)$. Note that no more than $W_1 \cdot W_2$ iterations are needed, since each iteration must reduce the cycle ratio of the detected cycle. We also find that this linear-search technique works well in practice, typically converging in a few iterations. Of all 192,316 attempts to find a minimum ratio cycle for the experimental runs illustrated in Figs. 8, 10, 12, 15, and 16, we find that 22,307 (11.60 percent), 69,686 (36.24 percent), 64,551 (33.57 percent), 29,241 (15.20 percent), 6,072 (3.16 percent), 452 (0.24 percent), and 7 (0.00 percent) take 2, 3, 4, 5, 6, 7, and 8 iterations of negative-cost–cycle detection, respectively. None require more than 8 iterations.

Both the binary and linear search techniques described above were used in Ahuja et al. [13, pp. 150-157] to find a minimum ratio cycle in a directed graph. To apply these techniques to undirected graphs, we need a method for finding negative-cost cycles in undirected graphs. It is this problem that we address in the next section.

## 3.3 Reducing Negative-Cost Cycle to Minimum-Cost Perfect Matching

We can determine whether a graph $\hat{G} = (\hat{V}, \hat{E})$ has a negative-cost cycle by asking whether a new graph $G' = (V', E')$ has a negative-cost perfect matching. A *perfect matching* in a graph is a subset of the edges such that each vertex has one incident edge from that subset. The *cost* of a perfect matching is the sum of the weights of the edges in that perfect matching. The reduction, adapted from Ahuja et al. [13, pp. 496-497] and illustrated in Fig. 3, constructs $G'$ from $\hat{G}$ as follows:

1. For each vertex $u$ in $\hat{G}$, $G'$ contains two corresponding vertices, $u_1$ and $u_2$, and one corresponding zero-weight edge $(u_1, u_2)$.
2. For each edge $(u, v)$ in $\hat{G}$, $G'$ contains two corresponding vertices, $u_v$ and $v_u$, and five corresponding edges

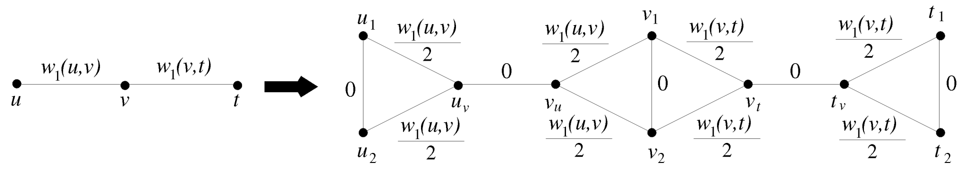


Fig. 3. An illustration of the method for constructing $G'$ from $\hat{G}$ to reduce negative-cost cycle to minimum-cost perfect matching. Adapted from Fig. 12.22 of Ahuja et al. [13].

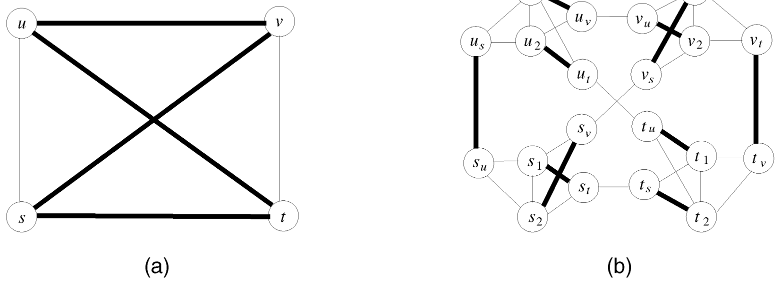


Fig. 4. An illustration of the correspondence between a cycle, the thick edges in (a), and a perfect matching, the thick edges in (b), using the reduction from Fig. 3.

with weights as follows: $w(u_1, u_v) = w(u_2, u_v) = w(v_1, v_u) = w(v_2, v_u) = \frac{1}{2}w_1(u, v)$ and $w(u_v, v_u) = 0$.

A graph has a negative-cost perfect matching if and only if its minimum-cost perfect matchings have negative cost. Edmonds [16], [17] gives a polynomial-time algorithm for finding a minimum-cost perfect matching. We can show that $G'$ always contains a perfect matching by constructing a trivial perfect matching containing all the edges of the form $(u_1, u_2)$ and $(u_v, v_u)$ for each $u \in \hat{V}$ and $(u, v) \in \hat{E}$. Since the cost of this perfect matching is zero, the minimum-cost perfect matchings in $G'$ must have nonpositive cost. The next lemma, adapted from an argument in Ahuja et al. [13, pp. 496-497], addresses the relation between the existence of a negative-cost cycle in $\hat{G}$ and the existence of a negative-cost perfect matching in $G'$.

**Lemma 5.** *$\hat{G}$ contains a negative-cost cycle if and only if $G'$ has a negative-cost perfect matching.*

Fig. 4 illustrates the correspondence between a cycle and a perfect matching using this reduction. The proof for Lemma 5 not only shows how to determine whether $\hat{G}$ has a negative-cost cycle, by a reduction to minimum-cost perfect matching, it also shows how to construct a set $S$ of negative-cost cycles from a minimum-cost perfect matching $M$.

## 4 NP-Hardness of Finding a Minimum Ratio Cut in General Graphs

We now show that the problem of finding a minimum ratio cut of an arbitrary graph is NP-hard. We do this by a reduction from the *commodity-cut-ratio*[2] problem. In the commodity-cut-ratio problem, one is given an undirected graph $G = (V, E)$ with nonnegative real-valued *capacities* $c(e)$ on the edges and a set of $k$ commodities $\{(s_1, t_1, D_1), \dots, (s_k, t_k, D_k)\}$. Each $D_i$ denotes a *demand* from *source* vertex $s_i$ to *target* vertex $t_i$. The problem is to find a subset $S \subset V$ that minimizes

$$\frac{\sum\limits_{e \in E \cap (S \times \overline{S})} c(e)}{\sum\limits_{\substack{i=1 \\ (s_i,t_i) \in ((S \times \overline{S}) \cup (\overline{S} \times S))}}^{k} D_i}.$$

Aumann and Rabani [18] state that this problem is NP-hard. Our minimum-ratio-cut problem is a special case of this problem where $w_1(u, v)$ corresponds to the capacities and $(u, v, w_2(u, v))$ corresponds to the commodities. The only difference is that commodity cut ratio allows demands between arbitrary pairs of vertices while minimum ratio cut allows second weights only on edges. We reduce an instance of the commodity-cut-ratio problem to an instance of the minimum-ratio-cut problem as follows: We add edges for all demands for which there do not exist edges in the original graph and set the capacities of these new edges to zero. We add a demand of zero for each edge $(u, v)$ for which there does not exist a demand between $u$ and $v$ in the original graph. Furthermore, if the commodity-cut-ratio problem contains two commodities $(u, v, D_{uv})$ and $(v, u, D_{vu})$, we set the demand $D_{uv}$ to $D_{uv} + D_{vu}$ and eliminate the commodity $(v, u, D_{vu})$. Now, we take $w_1$ to be the capacities and $w_2$ to be the commodities. This will reduce any commodity-cut-ratio problem to a minimum-ratio-cut problem. Thus, finding the minimum ratio cut of an arbitrary graph is also NP-hard.

The only remaining difficulty is that the minimum-ratio-cut problems produced by the above reduction may contain edges whose second edge weight is zero. The algorithm presented in Section 3 only supports problems where edges do not have zero second edge weights. It might be the case that such a limitation would allow the problem on arbitrary graphs to be solved in polynomial time as well. The following lemma shows that this is not the case:

2. Aumann and Rabani [18] call this problem *minimum cut ratio*. We change the name here since we use the term "minimum cut ratio" to mean something different in this paper.

**Lemma 6.** *For any graph $G$ with integral $w_1$ and nonnegative integral $w_2$, where all cuts $(A, B)$ satisfy $c_2(A, B) > 0$, there exists a rational $\epsilon > 0$ such that the minimum ratio cuts of $G$, when replacing all $w_2(u, v) = 0$ with $w_2(u, v) = \epsilon$, are also minimum ratio cuts of $G$ with the original $w_2$.*

This lemma shows that any minimum-ratio-cut problem with nonnegative second edge weights can be reduced to a minimum-ratio-cut problem with positive second edge weights. If the original problem has only integral first and second edge weights, the new problem has only integral first edge weights but might have nonintegral rational second edge weights. However, Lemma 3 shows that a minimum-ratio-cut problem with rational first and second edge weights can be reduced to a minimum-ratio-cut problem with integral first and second edge weights by scaling the weights. This scaling factor is the denominator of $\epsilon$ and is a polynomial function of the original edge weights. Thus, the problem of finding a minimum ratio cut of an arbitrary graph remains NP-hard even under the constraint that all second edge weights are positive.

## 5 The Baseline Method for Using Ratio Cut for Image Segmentation

One can use ratio cut to segment images at the pixel level. Because the algorithm presented in Section 3 is limited to planar graphs, we construct grid graphs from images where vertices correspond to pixels and edges correspond to Manhattan-neighboring pixel pairs. We take some decreasing function of the intensity difference between neighboring pixels as the first edge weights. We will describe the various edge-weight functions that we have used in greater detail in Sections 5.2 and 6.2. We first discuss a method for partitioning an image into multiple segments by recursively bipartitioning the image. We refer to the image-segmentation method that recursively bipartitions an image using the edge-weight functions and termination criterion discussed in this section as the *baseline* method.

### 5.1 Recursive Bipartitioning

The algorithm presented in Section 3 bipartitions a graph into two components. For image segmentation, we may wish to partition the corresponding graph into more than two components. We do this by recursively bipartitioning the graph, terminating the recursive bipartitioning when the segment corresponding to a component is sufficiently homogeneous. A natural question arises: What is a good measure of the homogeneity of a segment or component? One popular definition is the variance of the pixel intensity within the segment or component. Because this method ignores the spatial relations between pixels, it may not agree with human perceptual judgment. A simple example is shown in Fig. 5. Figs. 5a and 5b have the same intensity variance; however, the image shown in Fig. 5a is perceptually more homogeneous than that in Fig. 5b. The main reason for this is that the pixel intensities in Fig. 5a vary gradually, without any abrupt changes due to edges. Thus, pixel-intensity variance is not a suitable homogeneity measure.

We propose, instead, using the minimum cut ratio of a graph $G$ as a measure of its homogeneity. We refer to this homogeneity measure as $H(G)$. This homogeneity measure will correctly determine that Fig. 5a constitutes a single homogeneous segment while Fig. 5b does not.

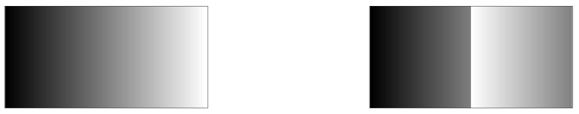


Fig. 5. An illustration of why pixel-intensity variance is ill-suited as a measure of segment homogeneity. The intensity of image (a) varies uniformly from $0$, at the left, to $255$, at the right. The intensity of image (b) varies uniformly from $0$, at the left, to $128$, in the center, then abruptly changes to $255$, and finally varies uniformly from $255$ to $129$, at the right. While the sets of pixel intensities in these two images have the same statistical properties, (a) appears to contain a single segment, while (b) appears to contain two segments.

A desired property of a recursive segmentation process is that, at each step, the child segments be more homogeneous than their parent. The following lemma shows that using $H(G)$ as the homogeneity measure and ratio cut as the segmentation method has this desired property:

**Lemma 7.** *If $(G_1, G_2)$ is a minimum ratio cut of the connected planar graph $G$ and $(G_{11}, G_{12})$ is a minimum ratio cut of $G_1$, then*

$$\mathrm{Rcut}(G_1, G_2) \leq \mathrm{Rcut}(G_{11}, G_{12}).$$

Fig. 6 illustrates the cases that arise when proving this lemma. From this lemma, it follows that $H(G_1) \geq H(G)$ and $H(G_2) \geq H(G)$ when $(G_1, G_2)$ is a minimum ratio cut of $G$. Thus, it makes sense to select a homogeneity threshold $H_T$ and terminate the recursive bipartitioning when segment homogeneity is larger than this threshold.

### 5.2 Edge-Weight Functions

In principle, one can use numerous different methods for computing edge weights for $G$ that incorporate a variety of image features such as intensity, color, texture, depth, or even higher-level knowledge. In this paper, we limit our discussion to one particular class of edge-weight functions based on intensity difference. We take some decreasing function $g$ of the intensity difference $f(u, v)$ between neighboring pixels that correspond to the vertices $u$ and $v$ as the first edge weights.[3] Let $f(u, v) = |I(x(u), y(u)) - I(x(v), y(v))|$ be the absolute intensity difference between the pixels that correspond to the vertices $u$ and $v$ in image $I$, where $(x(u), y(u))$ denotes the image coordinates of the pixel corresponding to vertex $u$. Two possible candidates for $g$ are a *Gaussian* decreasing function:

$$g(z) = e^{-\frac{z^2}{\sigma^2}}$$

and a *linear* decreasing function:

$$g(z) = -z.$$

Since the former has a free parameter $\sigma$, it is less desirable than the latter. We take $w_1(u, v) = g(f(u, v))$.

3. Because our implementation uses integer-valued edge weights, it linearly normalizes the output of the edge-weight functions to integers in the range $[0, 3 \times 255]$.

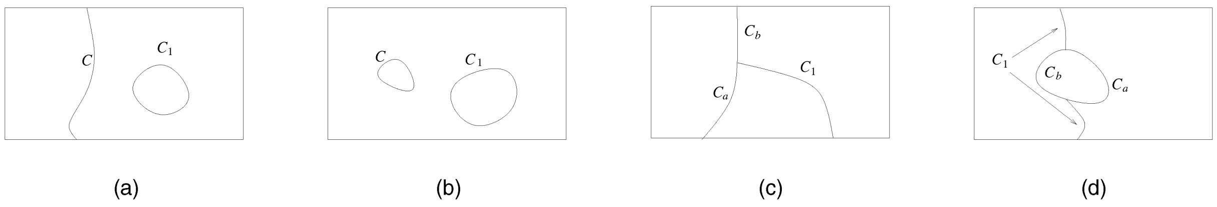


Fig. 6. The relations between the cut boundaries during recursive bipartitioning.

Previous authors [1], [2], [3], [8] have noted that graph-based image segmentation is often sensitive to the choice of edge-weight function. Prior work typically adopts a Gaussian edge-weight function. One reason for this is that, with some cost functions, Gaussian edge-weight functions tend to produce fewer spurious cuts than linear edge-weight functions. We find that this is not so with the baseline method. Furthermore, with Gaussian edge-weight functions, segmentation can be sensitive to the choice of $\sigma$.

Our baseline method is less sensitive to the choice of decreasing function $g$. This is illustrated by the synthetic binary image in Fig. 7. With the baseline method, setting $w_1(u,v) = g(f(u,v))$ and $w_2(u,v) = 1$, where $f(u,v)$ is defined as above, choosing any decreasing function $g$ will result in the proper segmentation, as shown in Fig. 7.

To further illustrate the relative insensitivity of the baseline method to the choice of decreasing function $g$, we processed the five images shown in column 1 of Fig. 8 with the baseline method using both a Gaussian decreasing function $g$, as shown in columns 2 and 3, and a linear decreasing function $g$, as shown in columns 4 and 5. This illustrates that the baseline method produces similar results with both decreasing functions. This is in contrast to performing recursive bipartitioning with the normalized-cut cost function. Columns 6 and 7 of Fig. 8 show the results of processing the same input images with a baseline method based on the normalized-cut cost function. Note that this is *not* the implementation of normalized cut due to Tal [12] that is used in Figs. 15 and 16. It does not include texture and is our own implementation of the baseline technique from Shi and Malik [7]. This baseline normalized-cut method recursively bipartitions an image using the normalized-cut cost function for both bipartitioning and the termination criterion and uses the same edge-weight functions as our baseline ratio-cut method. We use analogous recursive bipartitioning methods, edge-weight functions, and termination criteria for both ratio cut and normalized cut to focus the comparison on the cost functions of the corresponding baseline methods without possible extensions. For this experiment, we used the same value of $\sigma$ when using a Gaussian decreasing function for all images. For each image, a different value of $\sigma$, sometimes higher sometimes lower, can yield a better segmentation using normalized cut. However, we wish to compare the methods with a uniform set of parameters. Thus, we have chosen a single value for $\sigma$ that yields the best aggregate segmentations on all images according to informal human judgment. Note that, while the baseline normalized-cut method yields dramatically different results depending on the choice of decreasing function $g$, the results of the baseline ratio-cut method do not vary as widely. Since the linear decreasing function has fewer free parameters than the Gaussian decreasing function and is thus more desirable, the fact that the baseline method works well with a linear decreasing function gives some indication that it might form the basis of a good method for low-level image segmentation.

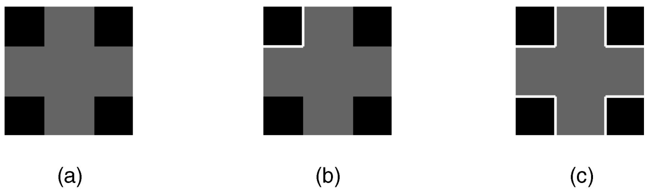


Fig. 7. Applying the baseline method to a synthetic binary image. (a) Input image. (b) Segmentation after the first bipartitioning step. (c) Segmentation after four bipartitioning steps.

## 6 The Extended Method for Using Ratio Cut for Image Segmentation

While the baseline method exhibits the properties discussed earlier, it has certain disadvantages. It is sensitive to salt-and-pepper noise and blurry edges, it can produce spurious cuts, and it can be slow in practice even though we use a polynomial-time algorithm. Sensitivity to noise and blurry edges, as well as the spurious-cut problem, result, in part, from the fact that our edge weights are pixel-based rather than region-based. To address these disadvantages, we extend the baseline method with a number of extensions.

First, we postprocess the result to merge very small segments into neighboring segments. Second, we use the same cut-ratio cost function to partition a graph whose vertices correspond to image regions instead of image pixels to incorporate region information into the edge weights. This process can be iterated, performing an initial segmentation based on pixels and having subsequent iterations use the segments produced by the previous iteration as the vertices in the graph. Finally, we show a blocking heuristic for quickly segmenting a large image by combining the segmentation results for subimages. While this heuristic compromises the provable optimality of the solution, in practice, we find that it does not appreciably reduce the quality of the resulting segmentations. We refer to the image segmentation method that incorporates these extensions into the baseline method as the *extended* method.

The baseline method can produce segmentations that contain spurious cuts that do not correspond to image edges. The cause of this phenomenon is illustrated in

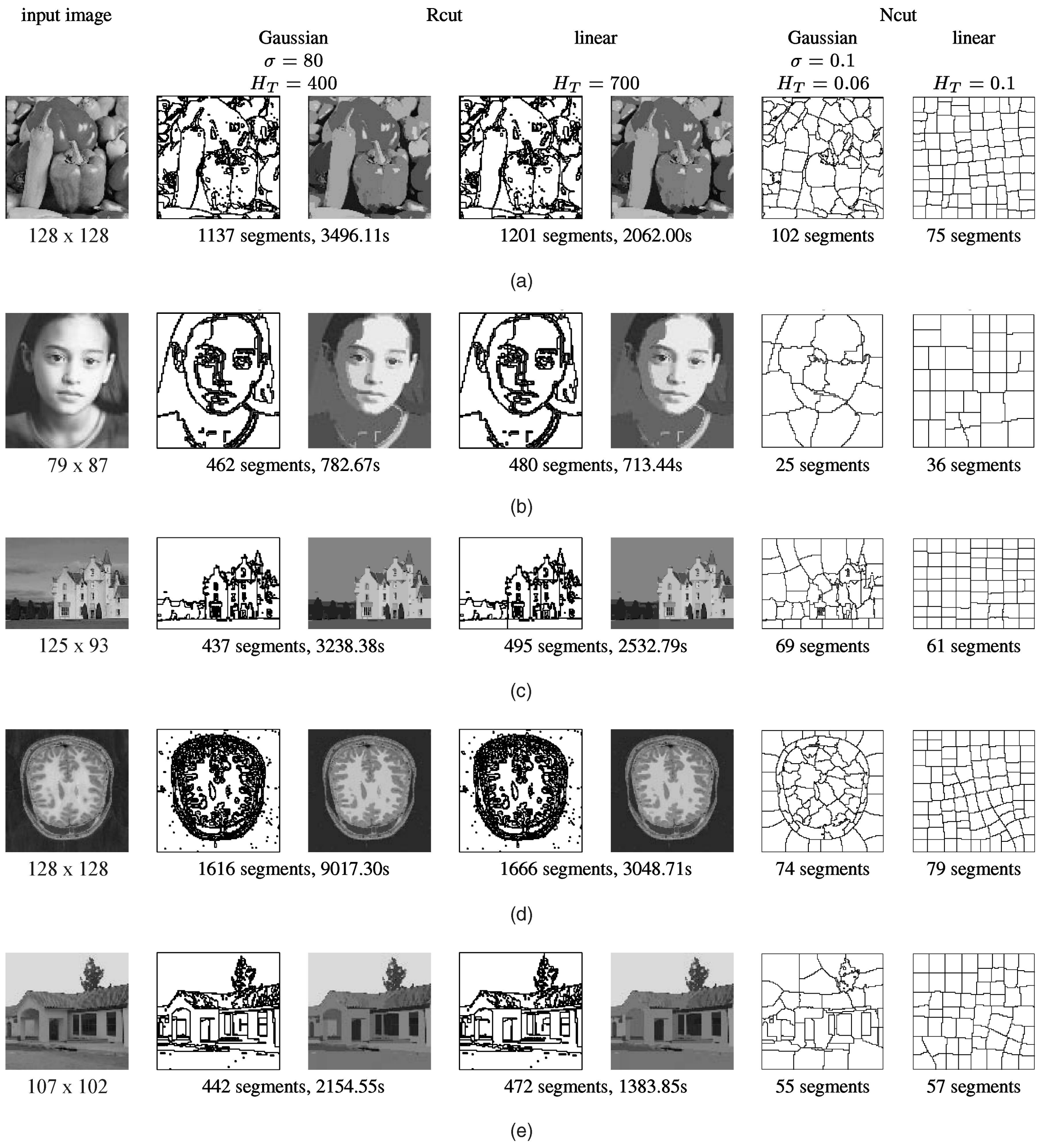


Fig. 8. Applying the baseline method to five images. The results are depicted both by the segmentation boundaries (columns 2 and 4) and by painting each segment with the average intensity in that segment (columns 3 and 5). We use $\sigma = 80$ for ratio cut with a Gaussian decreasing function. This is not excessively large given that we normalize pixel-intensity values to the range $[0, 3 \times 255]$. For comparison, we show the results of an analogous baseline method using the normalized-cut cost function instead of the cut-ratio cost function. Following Shi and Malik [7], our baseline normalized-cut implementation normalizes $f(u, v)$ to $[0, 1]$, thus our values of $\sigma$ and $H_T$ for normalized cut are different than those used for ratio cut. Images (a), (b), (c), and (d) are the same as those in Figs. 16c, 16d, 16e, and 15c except that they have been subsampled by a factor of 2 to allow them to be processed by our baseline normalized-cut implementation.

Fig. 9. Suppose that the desired cut, that corresponds to image edges, is $(A, B \cup C)$. Suppose that the cut $(B, C)$, that does not correspond to image edges, has a large cut ratio. In this case, the minimum ratio cut of $A \cup B \cup C$ may be $(A \cup C, B)$, when $\mathrm{Rcut}(A, B) < \mathrm{Rcut}(A, C)$ and $c_2(B, C) \ll c_2(A, C) < c_2(A, B)$. In other words, the cut ratio of the spurious cut $(A \cup C, B)$

$$\frac{c_1(A, B) + c_1(B, C)}{c_2(A, B) + c_2(B, C)}$$

can be less than the cut ratio of the desired cut $(A, B \cup C)$

$$\frac{c_1(A, B) + c_1(A, C)}{c_2(A, B) + c_2(A, C)}$$

even though the boundary between $A$ and $C$ has greater total image intensity difference than the boundary between $B$ and $C$ (i.e., $c_1(A, C) < c_1(B, C)$, because $c_1$ measures the negation of the total image intensity difference) when $c_2(B, C)$, the length of the boundary between $B$ and $C$, is much smaller than $c_2(A, C)$, the length of the boundary between $A$ and $C$.

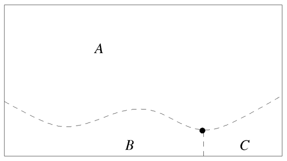


Fig. 9. An illustration of the cause behind spurious cuts.

Nonetheless, subsequent recursive bipartitioning will likely segment region $A \cup C$ into $A$ and $C$. The iterated region-based segmentation method introduced in Section 6.2 mitigates the problem of spurious cuts to a large extent by incorporating region information into the edge-weight functions.

## 6.1 Removing Very Small Segments

The baseline method is overly sensitive to salt-and-pepper noise in the input image. Because the cut-ratio cost function does not include a bias against small segments, salt-and-pepper noise leads to many such segments. Not only does this affect the quality of the segmentations produced, it also negatively impacts the running time of the algorithm. Furthermore, the effectiveness of the baseline method is reduced when the input image has blurry edges. Again, because of a lack of bias against small segments, this leads to many such segments along the blurry edge boundaries. One simple way of mitigating this problem is to postprocess the output of the baseline method to remove very small segments. We repeatedly search for the smallest segment $A_i$, with fewer than $A_T$ pixels, and merge it with the neighboring segment $A_j$, with the largest cut ratio. Note that such postprocessing uses the cut-ratio cost function. Following Wu and Leahy [1], all experimental results for the remainder of this paper use $A_T = 5$. This postprocessing approach to dealing with salt-and-pepper noise and blurry edges does not address the running time of the algorithm. This aspect of the problem is mitigated, to a large extent, by the iterated region-based segmentation method and the blocking heuristic introduced in Sections 6.2 and 6.3.

## 6.2 Iterated Region-Based Segmentation

So far, we have been segmenting images at the pixel level and taking the affinity between neighboring pixels to be some decreasing function of the absolute intensity difference between those pixels. This is a very local affinity measure. Incorporating information about the region around a pixel can give a better measure of affinity and lead to better segmentation results. One way to do this is to perform an initial segmentation using only the local affinity measure and then use this segmentation to remeasure the affinity between segments and use these segments and associated affinities for subsequent resegmentation. Such iterated region-based approaches have been investigated by Sharon et al. [19], [20] for normalized-cut–like cost functions.

So far, we have been using ratio cut in the restricted case with unity second edge weights applied to grid graphs whose vertices correspond to pixels in the input image. This corresponds to the mean-cut method of Wang and Siskind [10]. Generalizing mean cut to ratio cut, however, allows us to perform iterated region-based segmentation where graph vertices in subsequent iterations correspond to image regions instead of pixels. These regions are derived from segments produced by earlier iterations and lead to nongrid graphs.

More specifically, we perform an initial segmentation using mean cut. We then repeatedly derive a new segmentation from the previous segmentation. In this iterated process, we take the segments from the previous segmentation as vertices in a new graph and construct edges between vertices corresponding to neighboring segments. One can use any region-similarity measure to derive the first edge weights of this new graph. Like before, we define $w_1(u,v) = g(f(u,v))$ and retain the same decreasing functions $g$ as before. We generalize the definition of $f(u,v)$ to be a region difference instead of a pixel intensity difference as follows: First, let $R(u)$ denote the region corresponding to vertex $u$, $A(u)$ denote the set of vertices in the previous iteration that has been condensed into vertex $u$ in the current iteration, $\overline{I}(R)$ denote the average pixel intensity of region $R$, and $h(u,v) = |\overline{I}(R(u)) - \overline{I}(R(v))|$. Note that $h(u,v)$ is the absolute difference between the average intensities of the regions corresponding to vertices $u$ and $v$. We take $f(u,v) = \alpha h(u,v)c_2(A(u),A(v)) + (1-\alpha)c_1(A(u),A(v))$ We then take $c_2(A(u),A(v))$ in the previous graph as the $w_2(u,v)$ in the new graph. This maintains the invariant that $w_2(u,v)$ in any iteration corresponds to the length of the boundary that separates $R(u)$ and $R(v)$. The blending factor $\alpha$ governs the relative importance of information gained during a previous segmentation in constructing a new segmentation. When $\alpha = 0$, the first edge weights are preserved from one iteration to the next. In this case, the segmentation of this new graph will be the same as the segmentation of the previous graph. When $0 < \alpha \leq 1$, information gained in a previous segmentation influences the new segmentation process. All experimental results in this paper use $\alpha = 0.5$. Fig. 10 illustrates the results of applying the baseline method augmented with removal of segments with fewer than five pixels and iterated region-based segmentation to the same images as in Fig. 8. This experiment, and all remaining experiments in this paper, use four iterations, with the termination criteria $H_T = 740$, $H_T = 720$, $H_T = 700$, and $H_T = 680$, respectively.

Note that iterated region-based segmentation makes use of two properties that ratio cut generalizes over mean cut. First, in all but the first iteration, the second edge weights may not be unity. Second, in all but the first iteration, the graph may not be grid-like.

## 6.3 Heuristics for Speeding Up the Implementation

In practice, the running time of our method appears to be dominated by the time needed to compute minimum-cost perfect matchings. Gabow [21] gives an upper bound of $O(|V|^{\frac{3}{4}}|E| \log N)$ for computing minimum-cost perfect matchings on arbitrary graphs with integral edge weights, where $N$ is the maximal edge-weight magnitude. In practice, we find that a bounded number of iterations of minimum-cost perfect matching are needed to compute a minimum ratio cut. (See Section 3.2 for statistics.) Furthermore, in the graphs that we construct, $|E|$ is bounded by a constant times $|V|$. Thus, in practice, our method takes $O(|V|^{\frac{7}{4}})$ to find a minimum ratio cut. Since, in the first iteration of region-based segmentation, $|V|$ is the number of pixels in the image, our method scales with approximately the fourth power of the image resolution.

While, in theory, we have a polynomial-time algorithm for finding a minimum ratio cut, in practice, recursively bipartitioning a large image with ratio cut using the

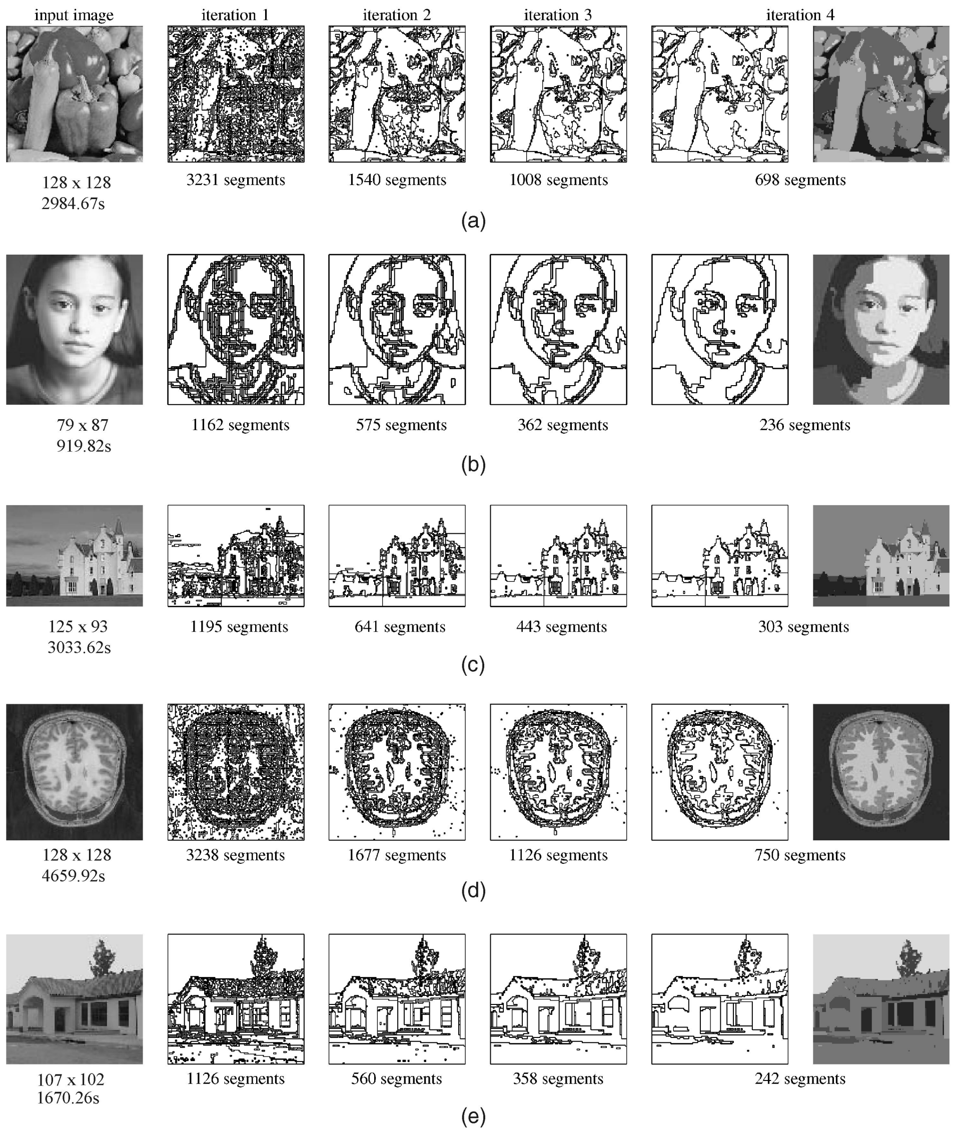


Fig. 10. An illustration of the results of applying the baseline method, augmented with removal of segments with fewer than five pixels and iterated region-based segmentation, to the same images as in Fig. 8. Subsequent iterations are shown from left to right. Since subsequent iterations can only contain boundaries that appear in earlier iterations, the leftmost (first) iteration intentionally oversegments the input image to avoid missing potential boundaries.

techniques described above can be slow. Nonetheless, the iterated region-based technique described in Section 6.2 can be used to derive a faster approach. The first iteration of the region-based technique is slow because it operates on a large graph containing vertices for all pixels in the input image. But, subsequent iterations are fast because the graphs contain vertices only for image regions that correspond to segments produced by earlier iterations. Thus, we replace the first iteration with the following method:

1. Divide the input image into overlapping subimages that cover the input image as shown in Fig. 11.
2. Segment each subimage using mean cut, setting the termination criterion to intentionally oversegment the subimage.
3. Form the set of all segments produced in Step 2. Whenever there are two overlapping segments $R_1$ and $R_2$, replace them with the set of connected components of $\{R_1 \cap R_2, R_1 \setminus R_2, R_1 \setminus R_2\}$. This will produce a disjoint partition of the input image into regions.

We then perform subsequent iterations using the same region-based technique as described in Section 6.2. Sharon et al. [19], [20] have employed a similar blocking heuristic for speeding up image segmentation with normalized-cut-like cost functions.

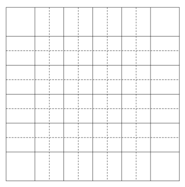
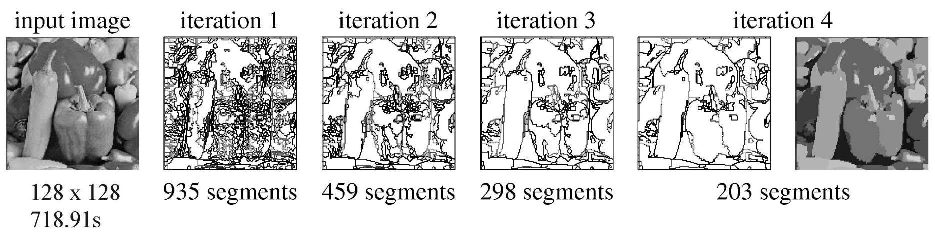
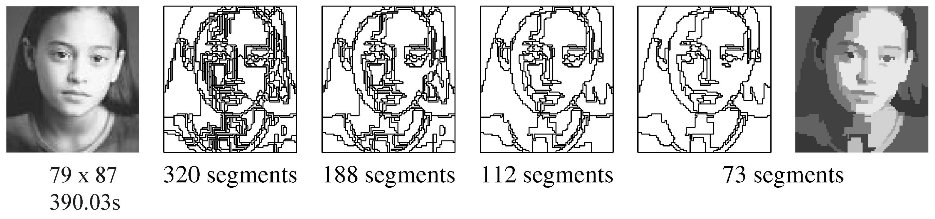


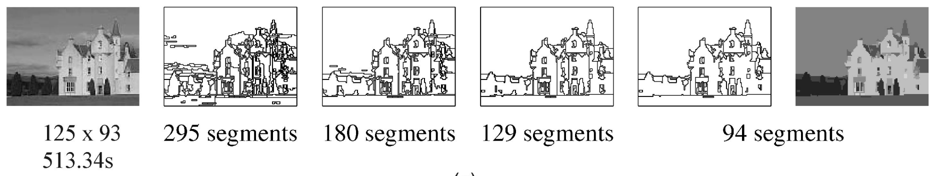
Fig. 11. An illustration of the blocking heuristic for computing the first iteration of an iterated region-based segmentation using overlapping subimages. The solid and dashed lines indicate the boundaries between two different overlapping sets of subimages.



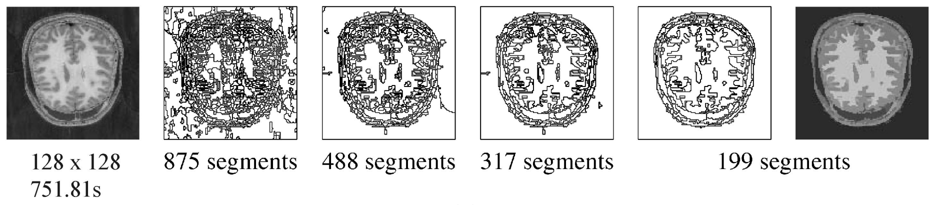
(a)



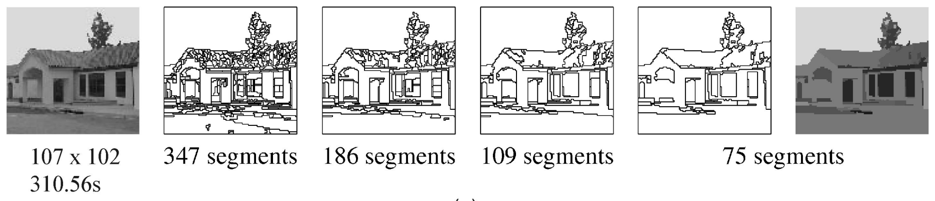
(b)



(c)



(d)



(e)

Fig. 12. A repetition of the same experiment as shown in Fig. 10 where the first iteration has been modified to use the blocking heuristic. The CPU times (in seconds) for the full segmentation process are shown. Note that the method is significantly faster than the method illustrated in Fig. 10, yet the results are similar.

This new first iteration is substantially faster than the original technique since it operates on small subimages rather than the whole image. We divide the image into overlapping subimages and choose to intentionally oversegment these subimages to reduce the chance that we fail to include some image edge in the input to subsequent iterations. The artifacts introduced by the subimage boundaries and oversegmentation will be eliminated in subsequent iterations. While this may lead to suboptimal partitions in theory, in practice, it yields results that are very similar to those produced by the original technique. Fig. 12 illustrates the results of repeating the same experiment as shown in Fig. 10, where the first iteration has been modified to use this blocking heuristic and all other parameters remain unchanged. This experiment, and all remaining experiments in this paper, use a block size of $32 \times 32$. This constitutes our full extended method that incorporates removal of segments with fewer than five pixels, iterated region-based segmentation, and the blocking heuristic.

## 7 Implementation

We have implemented the techniques described in this paper.[4] Our implementation of ratio cut uses the `blossom4` implementation of minimum-cost perfect matching [22]. We use a special-purpose method, limited to grid graphs, for computing the dual graph of the graph corresponding to the input image. This method, illustrated in Fig. 13, is summarized as follows:

4. The source code for our implementation and images and scripts for producing all of the examples in this paper are available from ftp://ftp.ecn.purdue.edu/qobi/ratio-cut.tar.Z.

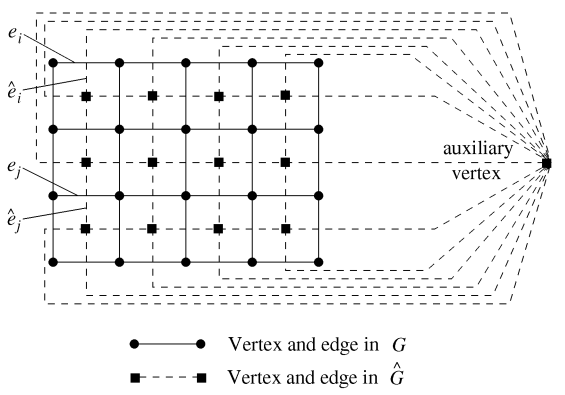


Fig. 13. An illustration of the special-purpose method for constructing the dual $\hat{G}$ of a grid graph $G$. The mapping from $e_i$ to $\hat{e}_i$ illustrates the transformation of border edges, while the mapping from $e_j$ to $\hat{e}_j$ illustrates the transformation of nonborder edges.

1. For every grid in $G$, $\hat{G}$ contains a corresponding vertex located in the center of this grid. These vertices are called *basic* vertices and form a new grid system.
2. $\hat{G}$ contains a distinct vertex for each border of $G$. These vertices are called *auxiliary* vertices.
3. Each nonborder edge $e \in E$ is mapped to a corresponding edge $\hat{e} \in \hat{E}$, with the same weights, that crosses $e$, in the grid system of $\hat{G}$, as shown in Fig. 13.
4. Each border edge $e \in E$ is mapped to a corresponding edge $\hat{e} \in \hat{E}$, with the same weights, that crosses $e$ and connects a border vertex to the auxiliary vertex for that border, as shown in Fig. 13.

Since this method is applicable only to grid graphs, we use LEDA [23] for computing the dual of nongrid graphs.

Nominally, we need to compute the dual graph at every recursive bipartitioning step. However, we have developed a technique for deriving the dual graphs of the child components produced by minimum ratio cut from the dual graph of the parent. This eliminates the need to compute the dual at recursive bipartitioning steps. With this technique, the first iteration of a region-based segmentation requires the computation of a single dual of a grid graph using our special-purpose method. Each subsequent iteration requires the computation of a single dual of a nongrid graph using LEDA.

Our method for deriving $\hat{G}_1$ and $\hat{G}_2$ from $\hat{G}$, after ratio cut bipartitions $G$ into $G_1$ and $G_2$, is illustrated in Fig. 14 and can be summarized as follows:

1. Remove edges in $\hat{G}$ that are in the minimum ratio cycle $C$ that corresponds to the cut $(G_1, G_2)$.
2. Label each edge $\hat{e} \in \hat{G}$ with $i \in \{1, 2\}$, if the corresponding edge $e$ from $G$ is in $G_i$.
3. Merge all vertices mentioned in $C$ into a single new auxiliary vertex $a$. Replace any edge that is incident on a vertex mentioned in $C$ with an edge of the same weights and label that is instead incident on $a$.
4. Label each vertex in $\hat{G}$ with the same label as its incident edge. Note that it is not possible for a vertex to have two incident edges with different labels.
5. Split $a$ into new auxiliary vertices $a_1$ and $a_2$, labeled 1 and 2, respectively. Each edge labeled $i$ that is incident on $a$ is replaced with an edge of the same weights and label that is incident on $a_i$.
6. $\hat{G}_i$ is the collection of vertices and edges with label $i$.

The following lemma shows that this technique is correct.

**Lemma 8.** *The graphs $\hat{G}_1$ and $\hat{G}_2$ produced by the above algorithm are the duals of $G_1$ and $G_2$.*

## 8 Experimental Results

An accurate and thorough comparison of the performance of ratio cut relative to the other graph-based image segmentation techniques discussed in Section 1 is beyond the scope of this paper. Nonetheless, it is instructive to illustrate the performance of our extended ratio-cut method on a collection of images and informally compare the results with those produced by another current state-of-the-art image-segmentation program. For this comparison, we have chosen the implementation of normalized cut due to Tal [12] which implements the techniques of Malik et al. [11].

We have processed 10 images with our extended ratio-cut method and compared the results to normalized cut. This image set includes five medical images, shown in Fig. 15, and five natural images, shown in Fig. 16. Our extended ratio-cut method was run with the same parameters on all 10 images. All extended-method results illustrate the last iteration of a four-iteration segmentation using a linear decreasing function for all iterations. Normalized cut was also run with the same parameters on all 10 images, namely, the default parameters given in the file `ncuts_params.txt` included in Tal [12]. These default parameters specified the use of a Gaussian decreasing function.

These images illustrate some of the important properties of ratio cut. Most, if not all, of the images require the image perimeter to be segmented. This is most clearly visible in Figs. 16c, 16d, and 16e, where image edges extend to the image perimeter. The lack of a size bias proves useful in segmenting the facial features of the bear in Fig. 16a and the

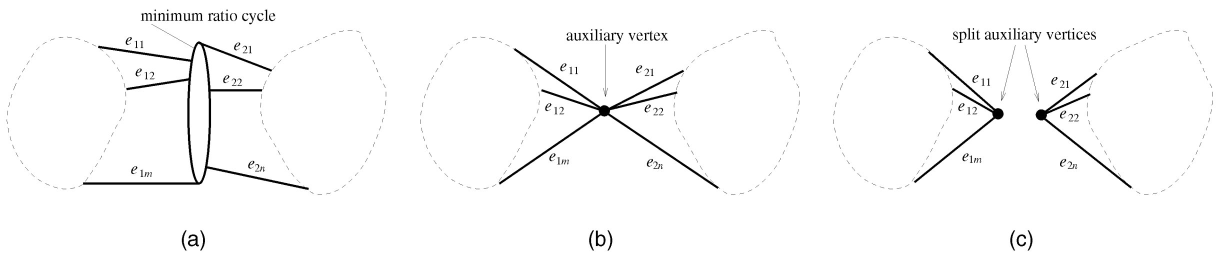


Fig. 14. Illustration of the algorithm for splitting $\hat{G}$ into $\hat{G}_1$ and $\hat{G}_2$.

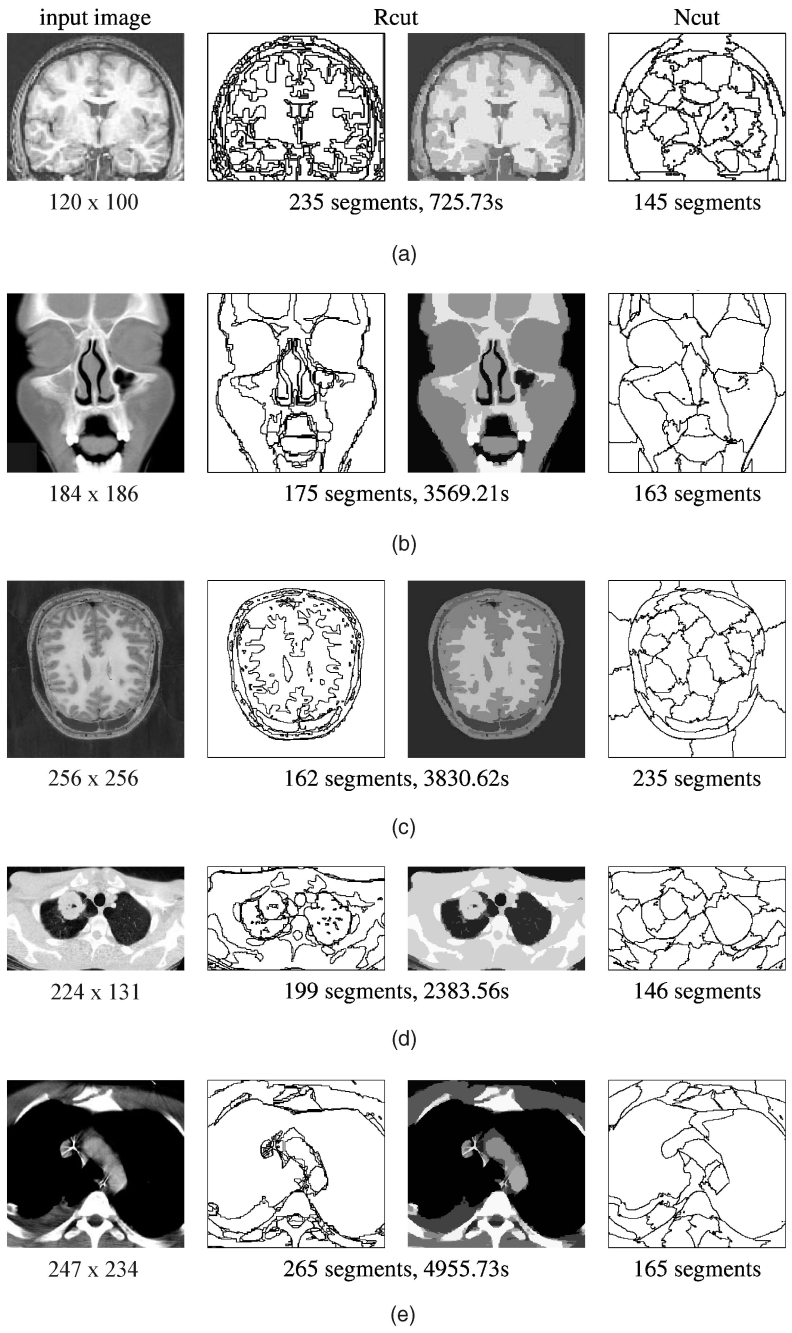
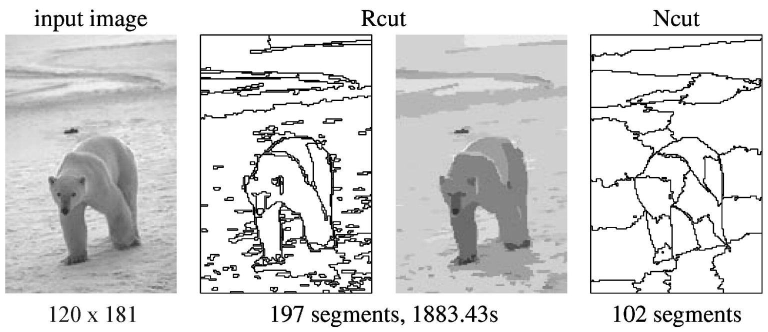


Fig. 15. A comparison between the extended ratio-cut method and normalized cut on five medical images.

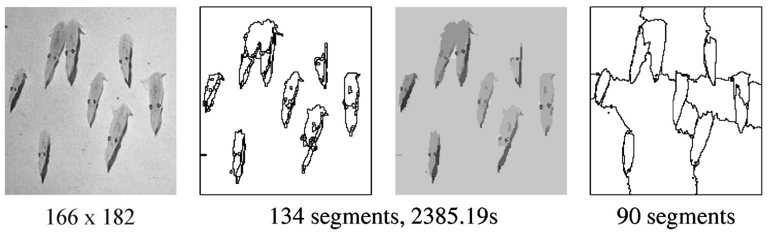
young woman in Fig. 16d as well as the small windows and doors in Fig. 16e. The lack of a shape, smoothness, or boundary-length bias proves useful in segmenting the gray matter regions of the brain images in Figs. 15a and 15c. Finally, most of the images illustrate the fact that ratio cut produces segmentation boundaries that are well-aligned with image edges. This is particularly evident in Figs. 15b, 15d, 15e, and 16b.

## 9 Conclusion

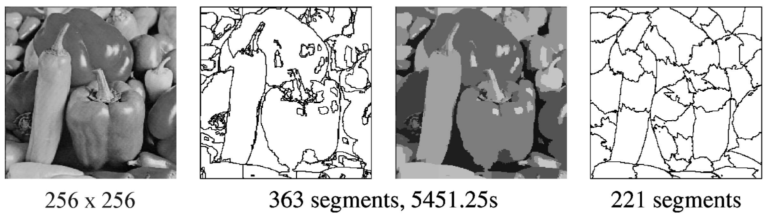
We have presented cut ratio, a new cost function for graph-based image segmentation, and ratio cut, a new algorithm for finding a cut that minimizes this cost function. Using this cost function for graph-based image segmentation has several properties. It allows the image perimeter to be segmented. It guarantees that the components produced by bipartitioning are connected. It does not introduce a size, shape, smoothness, or boundary-length bias. The lack of bias allows it to produce segmentations where boundaries are aligned with image edges. An optimal bipartition can be found in polynomial time. These properties may be useful for some image-segmentation applications. We have presented two methods for segmenting images using ratio cut. One, a baseline method, recursively bipartitions an image at the pixel level using ratio cut with an edge-weight function defined as some decreasing function of the absolute



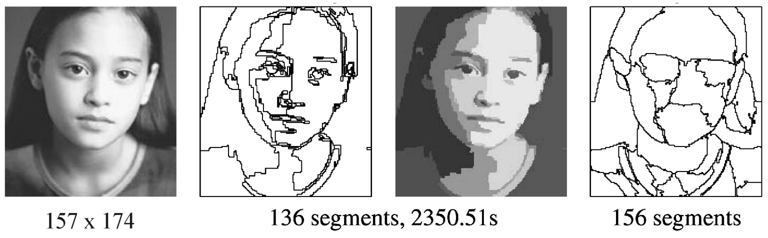
(a)



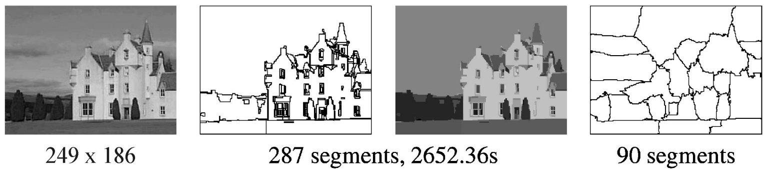
(b)



(c)



(d)



(e)

Fig. 16. A comparison between the extended ratio-cut method and normalized cut on five natural images.

intensity difference between neighboring pixels and a homogeneity measure based on cut ratio as the termination criterion. The other, an extended method, adds several extensions to the baseline method: removal of very small segments, iterated region-based segmentation, and a blocking heuristic. We have implemented the methods described in this paper and make our implementation available to other researchers. Finally, we have illustrated that our methods produce results that are comparable with current state-of-the-art graph-based image-segmentation methods on a set of medical and natural images. These results lead us believe that the cut-ratio cost function is a promising approach to image segmentation.

## Acknowledgments

This work was conducted while S. Wang was at the University of Illinois at Urbana-Champaign and supported, in part, by a Beckman Graduate Fellowship 2000-2001. Part of this work was conducted while S. Wang visited the NEC Research Institute, Inc. Part of this work was conducted while J.M. Siskind was at the NEC Research Institute, Inc. The authors would like to thank Tugkan Batu, Lance Fortnow, David Jacobs, Satish Rao, Warren Smith, Olga Veksler, Kevin Wayne, and the anonymous referees for useful discussions related to this paper. K. Wayne pointed us toward Ahuja et al. [13] and T. Batu, L. Fortnow, and W. Smith helped find the reductions in Sections 3.2 and 3.3.

## REFERENCES

[1] Z. Wu and R. Leahy, "An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 11, pp. 1101-1113, Nov. 1993.

[2] I.J. Cox, S.B. Rao, and Y. Zhong, "Ratio Regions: A Technique for Image Segmentation," *Proc. Int'l Conf. Pattern Recognition,* pp. 557-564, Aug. 1996.

[3] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 731-737, 1997.

[4] I.H. Jermyn and H. Ishikawa, "Globally Optimal Regions and Boundaries," *Proc. Seventh Int'l Conf. Computer Vision,* pp. 904-910, 1999.

[5] I.H. Jermyn and H. Ishikawa, "Globally Optimal Regions and Boundaries as Minimum Ratio Cycles," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 10, pp. 1075-1088, Oct. 2001.

[6] S. Sarkar and P. Soundararajan, "Supervised Learning of Large Perceptual Organization: Graph Spectral Partitioning and Learning Automata," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 5, pp. 504-525, May 2000.

[7] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.

[8] O. Veksler, "Image Segmentation by Nested Cuts," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 339-344, 2000.

[9] P. Soundararajan and S. Sarkar, "Analysis of Mincut, Average Cut, and Normalized Cut Measures," *Proc. Third Workshop Perceptual Organization in Computer Vision,* 2001.

[10] S. Wang and J.M. Siskind, "Image Segmentation with Minimum Mean Cut," *Proc. Eighth Int'l Conf. Computer Vision,* vol. 1, pp. 517-524, July 2001.

[11] J. Malik, S. Belongie, T. Leung, and J. Shi, "Contour and Texture Analysis for Image Segmentation," *Int'l J. Computer Vision,* vol. 43, no. 1, pp. 7-27, June 2001.

[12] D. Tal, "Normalized Cuts Software for Image Segmentation," http://www.cs.berkeley.edu/doron/software/ncuts, Dec. 2000.

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

[14] S. Wang and J. M. Siskind, "Image Segmentation with Ratio Cut–Extended Version," Technical Report TR-ECE-02-07, Purdue Univ., 2002.

[15] R. Karp, "A Characterization of the Minimum Cycle Mean in a Digraph," *Discrete Math.,* vol. 23, pp. 309-311, 1978.

[16] J. Edmonds, "Maximum Matching and a Polyhedron with 0,1-Vertices," *J. Research Nat'l Bureau of Standards,* vol. 69B, pp. 125-130, 1965.

[17] J. Edmonds, "Paths, Trees and Flowers," *Canadian J. Math.,* vol. 17, pp. 449-467, 1965.

[18] Y. Aumann and Y. Rabani, "An $O(\log k)$ Approximate Min-Cut Max-Flow Theorem and Approximation Algorithm," *SIAM J. Computing,* vol. 27, no. 1, pp. 291-301, Feb. 1998.

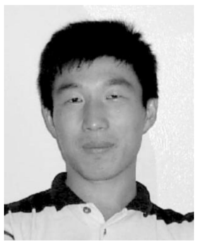
[19] E. Sharon, A. Brandt, and R. Basri, "Fast Multiscale Image Segmentation," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2000.

[20] E. Sharon, A. Brandt, and R. Basri, "Segmentation and Boundary Detection Using Multiscale Intensity Measurements," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2001.

[21] H.N. Gabow, "A Scaling Algorithm for Weighted Matching on General Graphs," *Proc. 26th Ann. Symp. Foundations of Computer Science,* pp. 90-100, 1985.

[22] W. Cook and A. Rohe, "Computing Minimum-Weight Perfect Matchings," http://www.or.nibonn.de/home/rohe/matching.html, Aug. 1998.

[23] K. Mehlhorn and S. Naher, *LEDA: A Platform for Combinatorial and Geometric Computing.* New York: Cambridge Univ. Press, 1999.



**Song Wang** received the PhD degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 2002. From 1998 to 2002, he also worked as a research assistant in the Image Formation and Processing Group of the Beckman Institute, University of Illinois at Urbana-Champaign. Since August 2002, he has been an assistant professor in the Department of Computer Science and Engineering at the University of South Carolina. His research interests include computer vision, medical image processing, and machine learning. He is a member of the IEEE and the IEEE Computer Society.

**Jeffrey Mark Siskind** received the BA degree in computer science from the Technion, Israel Institute of Technology in 1979, the SM degree in computer science from MIT in 1989, and the PhD degree in computer science from the Massachusetts Institute of Technology in 1992. He did a postdoctoral fellowship at the University of Pennsylvania Institute for Research in Cognitive Science from 1992 to 1993. He was an assistant professor at the University of Toronto Department of Computer Science from 1993 to 1995, a senior lecturer at the Technion Department of Electrical Engineering in 1996, a visiting assistant professor at the University of Vermont Department of Computer Science and Electrical Engineering from 1996 to 1997, and a research scientist at the NEC Research Institute, Inc., from 1997 to 2001. He joined the Purdue University School of Electrical and Computer Engineering in 2002, where he is currently an associate professor. His research interests include machine vision, artificial intelligence, cognitive science, computational linguistics, child language acquisition, and programming languages and compilers. He is a member of the IEEE and the IEEE Computer Society.