Int J Comput Vis (2009) 85: 167–181
DOI 10.1007/s11263-009-0251-z

Benchmarking Image Segmentation Algorithms

Francisco J. Estrada · Allan D. Jepson

Received: 13 May 2006 / Accepted: 13 May 2009 / Published online: 28 May 2009

Abstract We present a thorough quantitative evaluation of four image segmentation algorithms on images from the Berkeley Segmentation Database. The algorithms are evaluated using an efficient algorithm for computing precision and recall with regard to human ground-truth boundaries. We test each segmentation method over a representative set of input parameters, and present tuning curves that fully characterize algorithm performance over the complete image database. We complement the evaluation on the BSD with segmentation results on synthetic images. The results reported here provide a useful benchmark for current and future research efforts in image segmentation.

Keywords Computer vision · Image segmentation · Quantitative evaluation · Boundary matching

1 Introduction

Bottom-up image segmentation continues to be a challenging problem despite a sustained research effort that spans several decades. In recent years, several algorithms have been proposed and demonstrated on a handful of images, but the question of how these algorithms compare to one another has not been properly addressed. This is mostly due to the unavailability until recent years of a suitable, standard set of images and associated ground-truth, as well as the lack of publicly available implementations of major segmentation algorithms.

F.J. Estrada (✉) · A.D. Jepson
Department of Computer Science, University of Toronto, 6 King's College Road, M5S 3G4 Toronto, ON, Canada
e-mail: strider@cs.utoronto.ca

A.D. Jepson
e-mail: jepson@cs.utoronto.ca

In 2001 Martin et al. (2001) introduced the Berkeley Segmentation Database (BSD) as a repository of natural images with corresponding human segmentations. Since then, the BSD has been used to develop and evaluate boundary extraction algorithms based on image statistics (see Martin et al. 2004, and Kaufhold and Hoogs 2004). Concurrently with the development of the BSD several image segmentation algorithms have been made available by their authors. This has allowed different research groups to extend and improve known segmentation methods, and to visually compare the resulting segmentations.

However, a thorough quantitative evaluation of current image segmentation algorithms has yet to be reported. The lack of a suitable benchmark means that new image segmentation algorithms are still evaluated by providing qualitative comparisons with regard to other methods. While qualitative comparisons appeal to our intuitive perception of images and object boundaries, it may be quite difficult to determine visually whether a particular algorithm provides a significant advantage over alternate methods.

The goal of this paper is to present a quantitative evaluation of four current segmentation algorithms. To obtain a meaningful comparison, each algorithm must be tested over many possible combinations of input parameters. This, in turn, leads to the need for an efficient method for comparing potentially thousands of segmentations. We propose a simple procedure for quickly computing precision and recall measures of segmentation quality. This method is analogous to that of Martin (2002) but eliminates the need to solve an expensive bi-partite matching problem. The results of our study are presented in the form of tuning curves that fully characterize the performance of each algorithm over a wide range of input parameters. The segmentation algorithms we will compare in this paper are: Spectral-Embedding MinCut (SE-MinCut) (Estrada and Jepson 2004), Normalized Cuts

(Shi and Malik 2000), Mean-Shift (Comaniciu and Meer 2002), and Local Variation (Felzenszwalb and Huttenlocher 1998). In addition to this, the code that implements the matching algorithm described here has been made available online at http://www.cs.utoronto.ca/~strider/Benchmark so that other research groups can test and compare their own segmentation methods.

The paper is organized as follows. Section 2 gives a brief overview of the segmentation algorithms selected for comparison. Section 3 explains the quantitative evaluation framework and describes the fast matching algorithm used to compute precision and recall. Section 4 describes the experimental setup we used to perform the evaluation and comparison of the segmentation methods. Section 5 presents our experimental results and discussion including segmentation quality benchmarks for BSD images as well as for a set of synthetic scenes. Finally, Sect. 6 provides a conclusion to this paper. A preliminary version of this work appears in Estrada and Jepson (2005).

2 Segmentation Algorithms

Before we proceed to the evaluation part of this paper, it is convenient to briefly review the segmentation algorithms we have selected for comparison. Since three of the selected algorithms (SE-MinCut, Normalized Cuts, and Local Variation) are based on graph-theoretic principles, we will first describe the general graph representation for images. An input image I is transformed into a graph $G(V, E)$ in which V is a set of nodes corresponding to image elements (which may be pixels, feature descriptors, image patches, etc.), and E is a set of edges linking these nodes together. The weight of an edge $w_{i,j}$ is proportional to the similarity between image elements i and j (according to some appropriate measure of similarity), and is usually referred to as the affinity between i and j.

The similarity measure can use any of a number of image cues including image brightness, colour, and texture. It is also common to add a distance term (or to enforce a particular neighborhood structure) to ensure that the graph is sparse by linking together only image elements that are in close spatial proximity. Once the graph is built, segmentation consists of determining which subsets of nodes in V correspond to homogeneous regions in the image. Graph-theoretic segmentation algorithms exploit the notion that nodes that belong to the same region or cluster should be joined together by edges with large weights, while nodes that are joined by weak edges are likely to belong to different regions.

2.1 Spectral Embedding MinCut

Spectral-Embedding MinCut (Estrada and Jepson 2004) builds an affinity matrix W using a simple affinity measure based on the gray-level difference between neighboring pixels. From the affinity matrix, a Markov matrix M is formed by normalizing each column of W to add up to 1:

$$M = W D^{-1} \tag{1}$$

where D is the diagonal matrix with $D_{i,i} = \sum_k W_{k,i}$. This Markov matrix M can be used to determine patterns of diffusion that correspond to random walks started at different image pixels. In particular, if $p_{0,j}$ represents the probability distribution that corresponds to a particle starting a random walk at pixel $\vec{x}_j$ (i.e. with all of its probability mass located at pixel $\vec{x}_j$), the state of the random walk after t steps is given by

$$\vec{p}_{t,j} = M^t \vec{p}_{0,j},$$

where the term M^t can be computed efficiently from the eigenvectors and eigenvalues of M. The vectors $\vec{p}_{t,k}$ that result from starting the random walk at different pixels k in the image are called blur kernels. These blur kernels capture important properties of the neighborhood of the corresponding image pixels.

It is shown in Estrada and Jepson (2004) that the blur kernels for neighboring pixels in homogeneous-looking image regions will be very similar (more formally, their inner product will be close to 1), while for neighboring pixels separated by an image discontinuity the dot product will tend to vanish. This suggests that clusters of similar blur kernels should correspond to clusters of homogeneous-looking image pixels.

SE-MinCut uses spectral embedding to generate a low-dimensional representation of these blur kernels. Tight clusters of embedded blur-kernels correspond to seed regions of similar-looking pixels in the image. These seeds generally don't cover the complete image, so the minimum cut algorithm (Wu and Leahy 1993) is used to accurately locate the boundaries that separate different seed regions.

A cut through a graph is defined as the sum of the weights of the links that have to be removed to split the graph into two disconnected components R_1 and R_2. The minimum cut through a graph is the cut whose overall value (or capacity) is minimum

$$cut(R_1, R_2) = \sum_{v_i \in R_1, v_j \in R_2} w_{i,j},$$

$$MinCut = \min_{R_1, R_2} cut(R_1, R_2).$$

However, instead of looking for the minimum cut through the entire graph, SE-MinCut computes the minimum cut that separates especially designated source and sink nodes. S-T MinCut, as this particular version of min-cut is known, has been previously studied by Boykov and Kolmogorov (2001), and has been demonstrated as an efficient tool for

high-quality, interactive image segmentation by Boykov and Jolly (2001).

In SE-MinCut, source and sink nodes are defined automatically from the seed regions generated from the embedding. A final step of region merging to remove trivial cuts produces a segmentation of the image. Current algorithms related to SE-MinCut minimum-cut include Veksler (2000), Blake et al. (2004), Boykov et al. (2001), Ishikawa and Geiger (1998), Wang and Siskind (2001), Gdalyahu et al. (2001), Wang and Siskind (2003), Shental et al. (2003), and of course, the normalized cuts algorithm described next.

2.2 Normalized Cuts

Shi and Malik (2000) propose that the optimal segmentation of the image corresponds to the partition of the graph that minimizes the normalized cut measure

$$Ncut(R_1, R_2) = \frac{cut(R_1, R_2)}{assoc(R_2, V)} + \frac{cut(R_1, R_2)}{assoc(R_2, V)},$$

where $assoc(R, V)$ is the sum of the weights between nodes in R and the full graph V, and measures the strength of the association between the image elements in R and the whole image. It is shown in Shi and Malik (2000) that the cut that minimizes this normalized cut measure also maximizes the association between elements that belong to the same region.

Computing the exact solution to the normalized cut is an NP-complete problem. However, an approximate solution is provided by the eigenvector with the second smallest eigenvalue of the system

$$(D - W)y = \lambda Dy, \tag{2}$$

where W and D are as in (1). Furthermore, eigenvectors with successively larger eigenvalues can be used to split previously generated partitions. The segmentation algorithm consists of building the affinity matrix W, computing the eigenvectors and eigenvalues of the system shown in (2), and thresholding a number of these eigenvectors to obtain partitions of the image. The intersection of these partitions yields the final segmentation. The implementation of normalized cuts used in this paper employs an affinity measure that includes gray-level similarity, spatial proximity, and a measure of edge energy (see Malik et al. 2001 for details).

Extensions to the normalized cuts algorithm have been proposed by Belongie and Malik (1998), Malik et al. (1999, 2001), Fowlkes et al. (2001, 2004), Belongie et al. (2002), Yu and Shi (2003). Other segmentation methods related to normalized cuts include Sharon et al. (2000, 2001), Yu (2004, 2005).

Normalized cuts and SE-MinCut are closely related. In particular, Meila and Shi (2000) show that the normalized cuts eigenvectors are equivalent to those of the Markov matrix M in (1). However, despite the two algorithms solving an equivalent eigenvector problem, there are fundamental differences in the way the solution to this eigenvector problem is used to determine the segmentation. Normalized cuts discretizes and thresholds individual eigenvectors. Each eigenvector provides a two-way cut that depends on the discretization and thresholding method used. Individual eigenvectors provide good partitions when there are strongly separated clusters in the data, however in real world images this is not always the case. Weak region boundaries result in eigenvectors that are not piece-wise constant. Such eigenvectors are hard to threshold appropriately (leading to inaccurate boundary localization), and often lead to incorrect partitions.

Instead of discretizing individual eigenvectors, SE-MinCut uses a subset of the eigenvectors with largest eigenvalues to form a linear-subspace that represents a random-walk over the image pixels as described above. The result of simulating a certain number of steps of this random walk starting at each pixel yields information about image discontinuities and regions of homogeneous appearance in the neighborhood of each pixel, this provides considerably more information for segmentation purposes than the pairwise affinities used by normalized cuts, it also reduces the effects of noise. Clustering of the resulting blur-kernels yields sets of pixels that are highly likely to belong in homogeneous regions, and min-cut is used to determine the optimal partitions thereby achieving good boundary localization even along weak region boundaries.

2.3 Local Variation Algorithm

Felzenszwalb and Huttenlocher (1998) describe an efficient graph theoretic algorithm for image segmentation. Its principle of operation is that the image should be partitioned in such a way that for any pair of regions, the variation across neighboring regions should be larger than the variation within each individual region. They define two measures of variation:

$$Int(A) = \max_{e \in MST(A,E), e=(v_i,v_j)} w_{i,j}$$

and

$$Ext(A, B) = \min_{v_i \in A, v_j \in B, (v_i,v_j) \in E} w_{i,j},$$

where A is a region, $Int(A)$ is the internal variation within the region, $MST(A, E)$ is a minimum spanning tree of A, and $Ext(A, B)$ is the external variation between regions A and B.

The proposed algorithm works by merging together regions when the external variation between them is small with regard to their respective internal variations

$$Ext(A, B) \leq MInt(A, B),$$

where

$$MInt(A, B) = \min(Int(A) + \tau(A), Int(B) + \tau(B)),$$

and the threshold value $\tau(A) = \kappa/|A|$ determines how large the external variation can be with regards to the internal variation for two regions to still be considered similar. The analysis presented in Felzenszwalb and Huttenlocher (1998) concludes that the algorithm is nearly linear in complexity with regard to the number of pixels in the image.

2.4 Mean Shift Segmentation

Comaniciu and Meer (2002), describe a segmentation method based on the mean-shift algorithm. The mean-shift algorithm (Cheng 1995) is designed to locate points of locally-maximal density in feature space. Feature vectors containing gray-scale or colour information as well as pixel coordinates are computed for each pixel. Then the algorithm looks at a local neighborhood in feature space centered at each pixel's feature vector, and repeats the following steps iteratively:

- Compute a weighted mean of the feature vectors within this local neighborhood. The weight for each point is based on the distance between the point and the center of the local neighborhood (using an appropriate distance measure).
- Shift the center of the local neighborhood to this newly estimated weighted mean (hence the Mean-Shift algorithm)

These steps are repeated until a convergence condition is satisfied, and the location of convergence for each point is recorded.

In the original formulation, the local neighborhood is defined as a unit sphere in R^d, and the density estimate takes the form

$$\hat{f}(x) = \frac{1}{n \cdot h^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where h is the sphere's radius, and K is the weighting kernel. The kernel they use is given by

$$K(x) = \begin{cases} \frac{1}{2 \cdot c_d} \cdot (d + 2) \cdot (1 - x^t x) & \text{if } x^t x < 1 \\ 0 & \text{otherwise} \end{cases}$$

where c_d is the volume of the d-dimensional hypersphere. This results in the following expression for the sample's mean shift

$$M_h(x) \equiv \frac{1}{n_x} \sum_{x_i \in S_h(x)} (x_i - x)$$

where $S_h(x)$ is a hypersphere of radius h centered on x, and n_x is the number of feature vectors within the hypersphere. It is noted in Comaniciu and Meer (2002) that the use of a different kernel leads to a weighted mean computation.

The above algorithm can be easily generalized to include additional image cues within the feature vectors. The reader is referred to Comaniciu and Meer (2002) for more details about the structure of the feature space, the definition of the local neighborhood around a feature vector, and the mean-shift update rule. The mean-shift procedure will converge from an initial location in feature space to a region of locally maximal density. Pixels that have the same point of convergence are given the same segmentation label. Peaks in local density that are too close to each other are merged, and regions smaller than a user defined threshold are eliminated.

The algorithms described above are representative of current research in image segmentation, however; there are many other recent algorithms not discussed here. The reader is referred to Estrada (2005) for a thorough survey of current research in image segmentation. Finally, we should point out the important distinction between the segmentation algorithm itself and the similarity measure used to determine pixel similarity. The above algorithms support the use of different image features such as colour or texture. Here we examine their performance based purely on gray-level similarity, the objective is to evaluate the segmentation component of the algorithms on an equal footing independently of the different possible choices of similarity measure.

3 Quantitative Evaluation

There are two schools of thought with regards to the evaluation of computational vision algorithms in general. The first maintains that vision algorithms should be evaluated in the context of particular task (see for example Borra and Sarkar 1997). In the context of image segmentation and other low- and mid-level vision tasks this translates to measuring how much a particular algorithm contributes to the success of higher-level procedures that carry out, for example, object recognition.

The second school of thought holds that vision algorithms can be evaluated in terms of their performance with regard to some suitably defined ground truth (see for example Martin et al. 2001). In this paper we adopt this latter

philosophy, and present the results of evaluating the performance of the segmentation algorithms described above on a large database of natural images for which the ground truth is known. This has been made possible by the introduction of the Berkeley Segmentation Database (BSD) (Martin and Fowlkes 2001) by Martin et al. (2001).

The current public distribution of the BSD contains 300 colour images of size 481 × 321 pixels. For each of these images, the database provides between 4 and 9 human segmentations in the form of label maps. The segmentations are provided separately for the grayscale and colour versions of each image, and the complete database is split in two sets: A training image set consisting of 200 images and their corresponding segmentations, and a testing data set consisting of the remaining 100 images and their human segmentations.

The BSD was originally introduced as a tool for gathering statistics of natural images and for evaluating image segmentation algorithms. In particular, Martin et al. (2001) and Martin (2002) studied the problem of defining suitable measures of segmentation quality. In general terms, there are two possible types of measures: those that compare the overlap between regions in two segmentations of the same image; and those that evaluate the agreement between corresponding segmentation boundaries. Here we use the latter approach. Specifically, we compute precision and recall of segment boundaries with regard to human segmentations (see Martin 2002, Chap. 3). A similar approach has also been used to design and evaluate boundary detection algorithms (see Martin et al. 2002, 2004, and Kaufhold and Hoogs 2004).

Precision and recall are particularly attractive as measures of segmentation quality because they are not biased in favour of over- or under-segmented images. Precision measures the percentage of boundary pixels in the automatic segmentation that correspond to a boundary pixel in the ground truth and is sensitive to over-segmentation. Recall measures the percentage of boundary pixels in the ground-truth that were detected in the automatic segmentation and is sensitive to under-segmentation. More importantly, the use of precision and recall will allow us characterize the performance of segmentation algorithms in a way that is independent of particular choices of input parameters.

3.1 Precision and Recall

Given a source segmentation S_{source}, and a target segmentation S_{target}, precision is defined as the proportion of boundary pixels in S_{source} for which we can find a matching boundary pixel in S_{target}

$$\text{Precision} = \frac{\text{Matched}(S_{source}, S_{target})}{|S_{source}|},$$

where $|S_{source}|$ is the total number of boundary pixels in the source segmentation. In a similar way, recall is defined as the proportion of pixels in S_{target} for which we can find a suitable match in S_{source}

$$\text{Recall} = \frac{\text{Matched}(S_{target}, S_{source})}{|S_{target}|}.$$

To compute precision and recall, we need a method for determining correspondence between boundary pixels in the two segmentations. Martin (2002) proposed to solve this problem using bi-partite matching. In his formulation, a matching cost between pairs of boundary pixels is calculated and an optimization process determines the minimum-cost matching between boundary elements from the two segmentations. The matching cost has terms for proximity and boundary orientation similarity. Outlier elements with a large matching cost are introduced to equalize the number of boundary elements in the segmentations. Precision and recall are then computed using the percentage of boundary pixels in S_{source} and S_{target} respectively that were matched to outliers. The bi-partite matching between segmentation boundaries can be expensive to compute. It is also unclear whether forcing a one-to-one matching between boundary pixels is necessarily appropriate.

Here we improve upon the original formulation by Martin by proposing a significantly faster method for computing precision and recall. We use a matching algorithm based on boundary pixel proximity that has linear complexity in the number of source boundary pixels. We have observed empirically that it produces good quality matchings between corresponding boundaries at a fraction of the computational cost of the original bi-partite matching algorithm proposed by Martin. This speed-up is especially important since we will need to evaluate several thousand pairs of segmentations in order to benchmark any particular segmentation algorithm. The matching algorithm is described in detail in Sect. 3.3.

3.2 About the Segmentation Boundaries

The segmentations output by each algorithm are stored as labeled images where pixels within the same region have identical labels. To extract the region boundaries from the labeled images we could, as a first approach, simply mark as a boundary any pixels that have a neighbor with a different label. This, however, yields boundaries that are two pixels thick. Thick boundaries create two problems. First, very thin or very small regions will disappear altogether, having been replaced by solid clusters of boundary pixels within which the region-structure of the image is lost. Secondly, thick boundaries complicate the matching procedure and are likely to introduce unwanted artifacts in the resulting precision/recall scores (incidentally, the implementations we are

using of both Normalized Cuts and Mean Shift can be made to output region boundaries, but these boundaries are also two pixels thick). We could also attempt to mark pixels uniformly on one side (e.g. to the left and above) of region boundaries but this also introduces artifacts.

To eliminate these problems, we generate boundary images that have twice the resolution of the segmentations. In these higher-resolution images, boundaries can be accurately localized to lie between the pixels corresponding to the original regions in the low resolution segmentation. The procedure for generating the super-sampled boundaries is simple: We super-sample each individual region in the original segmentation in turn and find its boundary in the higher-resolution image. The final boundary map is formed by the union of the super-sampled boundaries of the individual regions. Figure 1 shows a segmentation, the boundaries extracted by marking as boundary any pixels that have neighbors with a different label, and the super-sampled boundary map.

We compute super-sampled boundary maps for all the segmentations generated by the four image segmentation algorithms. The same procedure is applied to the human segmentations from the BSD. Given two super-sampled boundary maps, we can compute correspondence maps for precision and recall (in general, the correspondence maps will not be identical since the matching procedure is not symmetric).

3.3 Matching Algorithm

The matching algorithm is quite simple. For each boundary pixel $p = (x_p, y_p)$ in the source segmentation, we look at a circular window of radius ϵ centered at (x_p, y_p) in the target segmentation. Any boundary pixels within this window are potential matches for p. We say that boundary pixel q within the search window is a suitable match for p the following conditions are satisfied:

(1) There are no other boundary pixels in S_{source} between p and q (no intervening contours constraint).
(2) The source pixel r that is closest to q and the source pixel p being matched to q must be on the same side of the target boundary q is apart of (same side constraint).

Figure 2 illustrates the two matching constraints. The first condition mentioned above is meant to avoid matching across multiple boundaries, we can only match two boundary pixels if there are no other boundary pixels between them. The second condition deserves further comment, without it, the matching algorithm will indiscriminately match boundary pixels on both sides of individual target boundaries. The result is that instead of obtaining a tight match between one boundary in S_{source} and one boundary in S_{target}, we will get a band of matched pixels in S_{source} all of which are within distance ϵ of a single target boundary. For a match between source pixel p and target pixel q to occur,

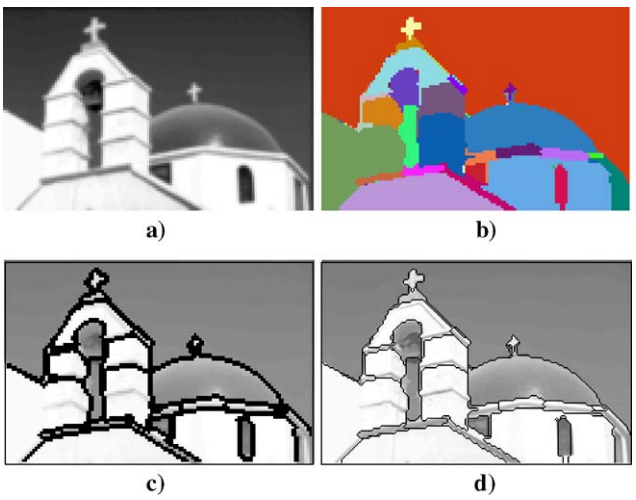

Fig. 1 (**a**) Original image (121 × 81). (**b**) Segmentation labels (SE-MinCut). (**c**) Boundaries generated by marking every pixel that has at least one neighbour with a different label (121 × 81, boundaries are 2 pixels thick). Notice that very small or very thin regions become indistinguishable clumps of boundary pixels. (**d**) Super-sampled boundary map (242 × 162) boundaries are now more accurately localized, and the structure of small and thin regions is preserved

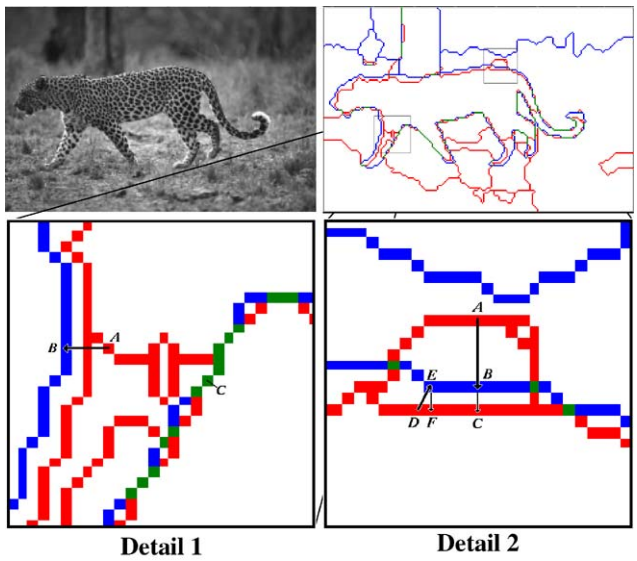

Fig. 2 (Color online) The *top row* shows an image from the BSD and an overlay of one human segmentation (*blue*), and a segmentation produced with SE-MinCut (*red*, *green pixels* indicate overlap). **Detail 1** shows the application of the first of our matching constraints. Pixel **A** from the source segmentation can't be matched to pixel **B** in the target segmentation because there are other source (*red*) boundary pixels between the two. An implication of this constraint is that any source boundary pixel that overlaps a target boundary pixel (for example, consider the pixel marked **C**) prevents any other source boundary pixels from matching to the same target pixel. **Detail 2** shows the application of the second matching constraint. In this case, the pixel labeled **A** cannot be matched to target pixel **B** because **A** is not on the same side of the boundary as pixel **C**, which is the source pixel closest to **B**. Conversely, pixel **D** can be successfully matched to pixel **E** since it is on the same side as **F**

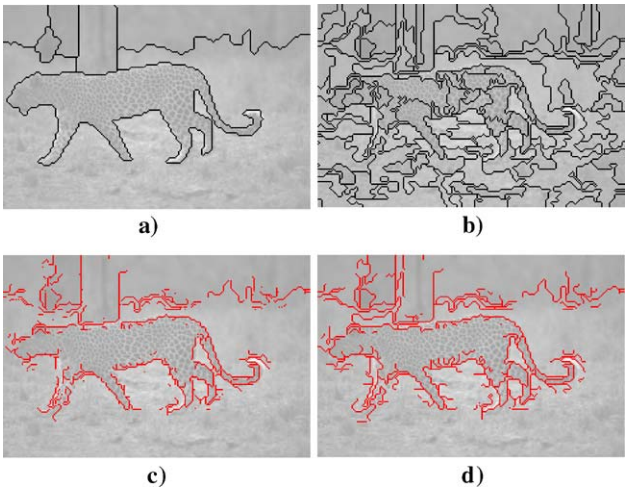

Fig. 3 (**a**) One human segmentation from the BSD. (**b**) Automatic segmentation (Mean-Shift, parameters set for over-segmentation). (**c**) Matched boundary pixels ($\epsilon = 7$) using both of the constraints mentioned in the text. (**d**) Matched boundary pixels ($\epsilon = 7$) without the second (same side) constraint. Notice that without the 'same side' constraint the matching procedure finds matches for any pixels that lie within a band of radius ϵ around the human segmentation boundaries. This would increase the precision score for the algorithm artificially

both p and the source boundary pixel r that is closest to q must be on the same side of the boundary q is part of.

Figure 3 provides examples of matching with and without this constraint. It is still possible that boundary pixels from S_{source} that lie on opposite sides but at the same distance of a boundary in S_{target} will be matched to the same boundary, but we have found that in general the matching algorithm described above generates good correspondence maps.

We should point out that ϵ here refers to pixel distance in the boundary maps input to the matching algorithm. Since our boundary maps are half the resolution of the original BSD images, the reader can determine the actual distance in terms of the full size BSD images by taking the ϵ values stated in the experimental results and multiplying them by two. This should allow the reader to get an idea of how big the allowed error in boundary localization is with regard to the actual images that the human observers segmented.

For a fixed value of ϵ, the matching algorithm described above has linear complexity in the number of source boundary pixels. This is true because the matching procedure will examine at most a constant number of pixels in the target segmentation for every boundary pixel in the source segmentation.

4 Experimental Setup

Due to the computationally intensive nature as well as practical limitations of SE-MinCut and Normalized Cuts, we have carried out the evaluation on images that have been down-sampled by a factor of 4 from their original size in the BSD. The input images have a size of 121×81 pixels, and we use only the grayscale version of each image.

Since the human segmentations contained in the BSD have the same size as the original images, we are faced with the problem of comparing segmentations produced at a lower level of detail against the full resolution segmentations produced by humans. This could be achieved in either of two ways: The resulting segmentations could be interpolated and up-sampled to the appropriate size, or the human segmentations could be down-sampled. For simplicity, and to reduce the computational cost of evaluating the error measures, we chose this latter option. Thus, for error computation purposes the human segmentations from the BSD are also down-sampled by a factor of 4.

To obtain a meaningful benchmark, we must test each algorithm for different combinations of its input parameters in a systematic way. The overall testing procedure for a particular algorithm is as follows: for each combination of parameters, the algorithm is run over the complete set of 300 images from the BSD. Note that there is no training phase since the algorithm will be tested using different combinations of input parameters, hence, we do not need to separate training and testing image sets, and we can use all the available images for the evaluation. For each image, we compute the average precision and recall with respect to the available human segmentations. The score of the algorithm for that particular combination of input parameters is the median of the precision and recall scores obtained for the individual images. We chose to use the median because the distribution of precision/recall values is typically non-Gaussian. However, using the average instead of the median was observed to yield similar results.

The median precision and recall values computed for different combinations of input parameters yield tuning curves that fully characterize the performance of the algorithm. For algorithms that have only one input parameter, we get a single tuning curve, whereas algorithms with more parameters will have multiple curves each of which corresponds to precision/recall scores obtained by changing one input parameter while holding all other parameters fixed.

4.1 Segmentation Algorithms and Algorithm Parameters

We used our own implementation of SE-MinCut, as well as the following publicly available segmentation programs: for Normalized Cuts see Cour et al. (2006), for Local Variation see Felzenszwalb and Huttenlocher (2004), and for Mean-Shift see Georgescu and Christoudias (2003). The ranges for the input parameters of each algorithm were determined experimentally so as to produce segmentations that go from severely under-segmented to severely over-segmented, with increments chosen so as to yield visually perceptible differences within the selected range. We evaluated the algorithms

exclusively on the parameters that are user-tunable in the respective implementation. Any internal parameters were kept fixed.

The input parameters and ranges are as follows: For Normalized Cuts the only input parameter is the desired number of regions, we tested the algorithm for a number of output regions within [2, 128]. The Local Variation algorithm also takes a single input parameter k that roughly controls the size of the regions in the resulting segmentation (for details please refer to Felzenszwalb and Huttenlocher 1998), smaller values of k yield smaller regions and favour over-segmentation. For this algorithm we tested values of k within [10, 1800].

For Mean-Shift we have two parameters: The spatial bandwidth, and the range bandwidth. These parameters are related to the spatial and gray-level intensity resolution of the analysis (see Comaniciu and Meer 2002 for details). In practice, the segmentation software for Mean-Shift accepts an additional parameter: the size in pixels of the smallest region allowed. We didn't test the effect of this parameter and instead kept it fixed to 25 pixels (corresponding to an area roughly equal to the size of the window used for computing precision and recall). Experimentation showed that the largest differences between segmentations were obtained while varying the range bandwidth parameter. Thus, we evaluate the algorithm using three values for the spatial bandwidth within [2, 8], and for each of these values, we compute a tuning curve that corresponds to variations of the range bandwidth within [1, 20] (see Fig. 4).

For SE-MinCut we have two parameters: d which determines the number of eigenvectors to use in the embedding, and τ_m which is the threshold used during the merging step. The largest variation between segmentations is obtained by changing the value of d, so following the same methodology used with Mean-Shift we use 5 values for the merging threshold between [.0625, .75], and for each of these we compute a tuning curve that corresponds to variations of d within [5, 40] (see Fig. 4).

5 Comparison Results

Since SE-MinCut and Mean-Shift have several tuning curves (shown in Fig. 4 for two values of ϵ), we have chosen for comparison the curves that are most favorable for each method. For SE-MinCut we will use the curve that corresponds to $\tau_m = .25$, and for Mean-Shift we will use the curve that corresponds to a spatial bandwidth $SB = 8$.

Figure 5 shows the tuning curves for all algorithms for different values of ϵ, also shown are error bars in the directions of precision and recall corresponding to the median standard error $mse = 1.25 * \sigma/\sqrt{N}$ where N is the number of samples (in the case of the BSD images $N = 300$), and σ is the corresponding standard deviation along the precision or recall axis. The tuning curves describe the behavior of the algorithms as input parameters change, and can be used to determine the input parameters that will yield a desired combination of precision and recall within the algorithm's operating range. Since the curves fully characterize algorithm performance, they provide a direct comparison of the quality of the segmentations produced by different algorithms independently of the choice of input parameters. The curves show when an algorithm performs consistently better than the others, and allow us to rank algorithms by performance for particular values of precision or recall.

The tuning curves shown in Fig. 5 show that the SE-MinCut algorithm generates the segmentations that agree most closely with the ground truth. Figure 6 shows the segmentations produced by each of the algorithms for several images from the BSD using the parameters at the middle of the tuning curve for each method. These segmentations provide a visual comparison of the segmentations generated by each algorithm, and complement the results from Fig. 5. It is encouraging that the precision/recall results for each algorithm agree well with the visually perceived quality of the corresponding segmentations.

Not surprisingly, the scores for all algorithms fall sharply as ϵ is decreased. Figure 7 shows the distribution of distances between matching boundary pixels in corresponding human segmentations. The distribution was generated by pairing human segmentations of the same image, computing a correspondence map between them using the algorithm described above, and generating a histogram of the distances between matching boundary pixels for all pairs of corresponding human segmentations. The distribution confirms the consistency of human segmentations previously observed by Martin et al. (2001), and supports the use of a small matching radius during evaluation. Additionally, the observed distribution indicates that the matching distance ϵ should be at least 2 to obtain meaningful precision/recall scores.

The reader may wonder about the practical usefulness of segmentations with different precision/recall scores along the tuning curves. While the answer to this is likely to be task-dependent, we can provide a visual impression of what the segmentations on different points along the tuning curve look like. Figure 8 shows segmentations of the same image produced with different parameter choices for each of the algorithms. The parameters used correspond to the points of highest recall, middle of the tuning curve, and highest precision. The images also show (in red) which boundary pixels agree with one of the human segmentations for the image. These results provide an intuitive understanding of what different points along the tuning curve represent. It can be seen in Fig. 8 that over-segmentation is characterized by high recall but low precision, and that under-segmented images correspond to high precision, but low recall.

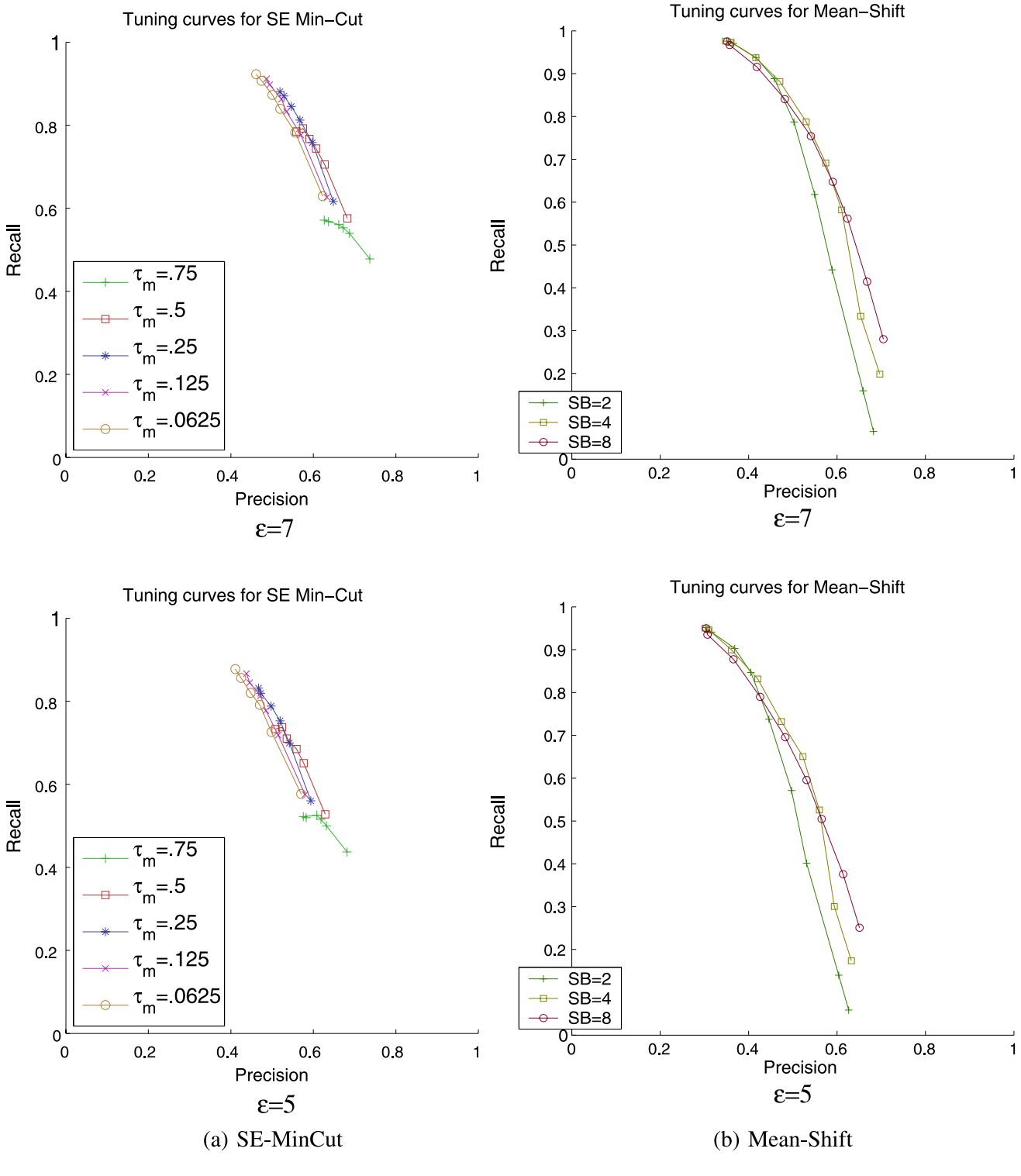

Fig. 4 Tuning *curves* SE-MinCut (*left column*) and Mean-Shift (*right column*) for $\epsilon = 7$ and (**b**) $\epsilon = 5$. From these *curves*, we chose the one that is most favorable for each method. For SE-MinCut we selected the *curve* for $\tau_m = .25$, and for Mean-Shift we selected the *curve* for $SB = 8$

With regard to the segmentation algorithms we can observe that at the point of highest recall, Mean-Shift, Normalized Cuts, and Local Variation produce notoriously over-segmented results, while at the point of highest precision Mean-Shift and Local Variation produce segmentations that capture small, high-contrast regions that do not necessarily correspond to actual object boundaries. Over-segmentation is limited for SE-MinCut by the fact that the leading eigen-

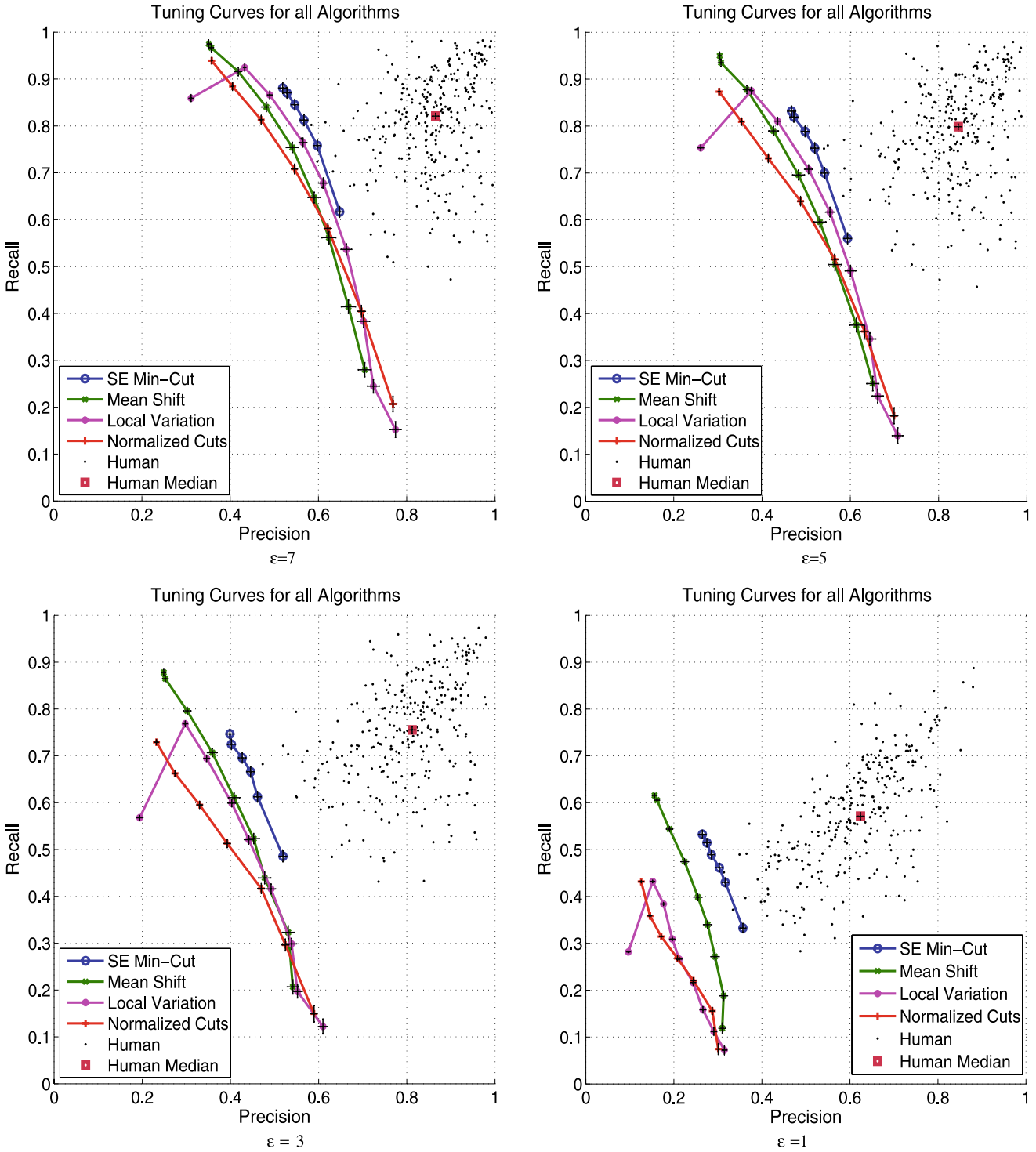

Fig. 5 Tuning *curves* for all algorithms. The *curves* are shown for values of $\epsilon = 1, 3, 5,$ and 7. The scores obtained by each algorithm decrease sharply as we reduce the matching radius ϵ (it's worth pointing out that for $\epsilon = 1$ the *curves* may be influenced by discretization artifacts in our super-sampled boundary maps). Even so, the *curves* provide a stable performance ranking of the segmentation methods. Also shown are the mean precision/recall scores for human observers on each image as well as the median human performance

vectors of the Markov matrix usually capture coarse properties of the random walk, as well as by the algorithm's merging stage. Under-segmentation occurs for all algorithms, but SE-MinCut and Normalized Cuts benefit from the global nature of the information provided by the eigenvectors of a Markov matrix or an affinity matrix respectively. Overall,

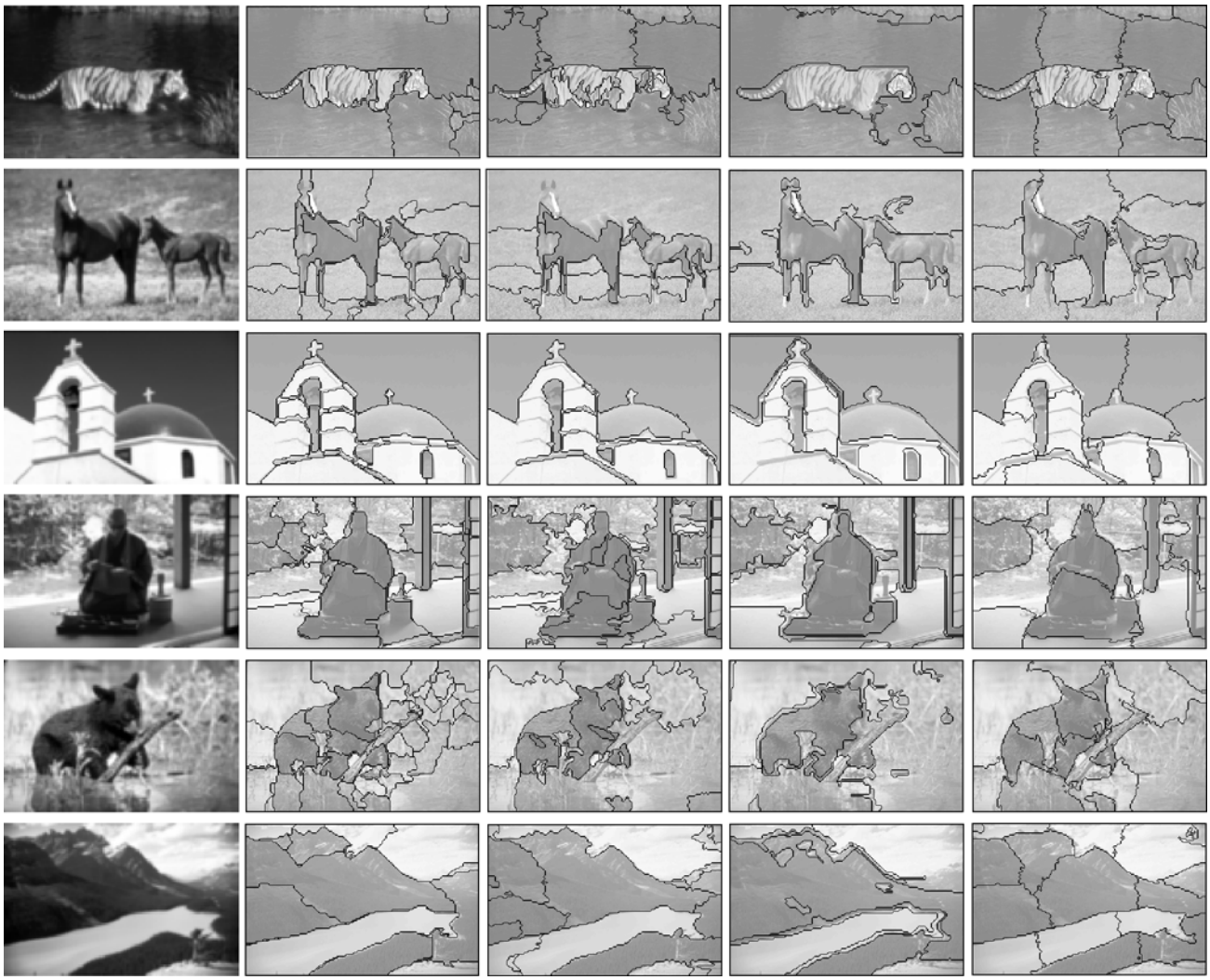

Fig. 6 From left to right: Original image, SE-MinCut segmentation, Mean-Shift segmentation, Local Variation output, and Normalized Cuts regions. The segmentations correspond to the parameter settings at the middle of the tunning *curve* for each algorithm

visual inspection suggests that SE-MinCut produces better segmentations.

In terms of run-time, the best performance is achieved by the Local Variation algorithm which takes less than 1 sec. to segment images of the size used here. Mean-Shift takes between 1 and 7 sec. depending on the spatial bandwidth parameter, Normalized Cuts takes between 10 sec. and 1.5 min. depending on the number of regions requested, and SE-MinCut takes between 1 and 7 min. depending on the number of eigenvectors used for the embedding (these times were measured on a 1.9 GHz Pentium IV machine). Both Normalized Cuts and SE-MinCut are partly implemented in Matlab, while Local Variation and Mean Shift are stand-alone binary modules. These differences in speed performance are significant and important. Depending on the task, it may be more appropriate to use a faster algorithm that yields somewhat less accurate boundaries but computes them in a fraction of the time. Given the run-times noted above, the tuning curves provided in the paper can be used to select the fastest algorithm that will work above specified lower bounds on precision and recall.

In addition to the tuning curves, we have included precision/recall scores for the human observers. These were computed by taking pairs of human segmentations of the same image and computing the mean precision and recall just as we have done for the automatic segmentation methods. This yields one data point per image describing the mean precision/recall achieved by human observers on that particular image. The median precision and recall for human observers is also shown. We can see a significant gap in performance between all the segmentation algorithms and the human results which clearly shows that bottom-up segmentation algorithm still fall short of human performance on natural images.

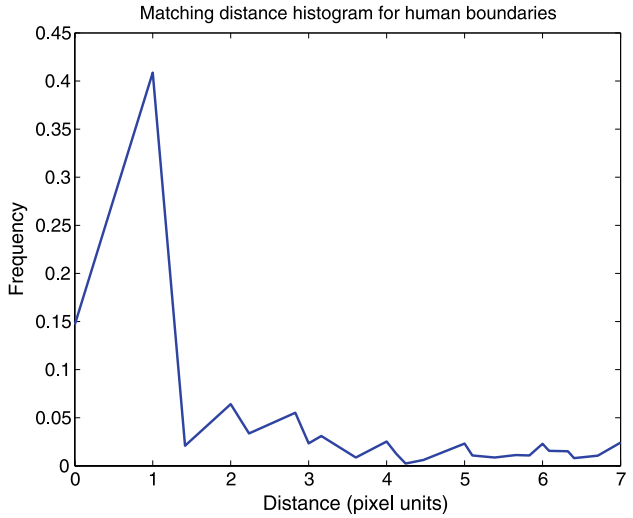

Fig. 7 Distribution of distances between matching boundary pixels in human segmentations. The observed distribution supports the use of a matching distance of at least $\epsilon = 2$ for images of the resolution used in this study

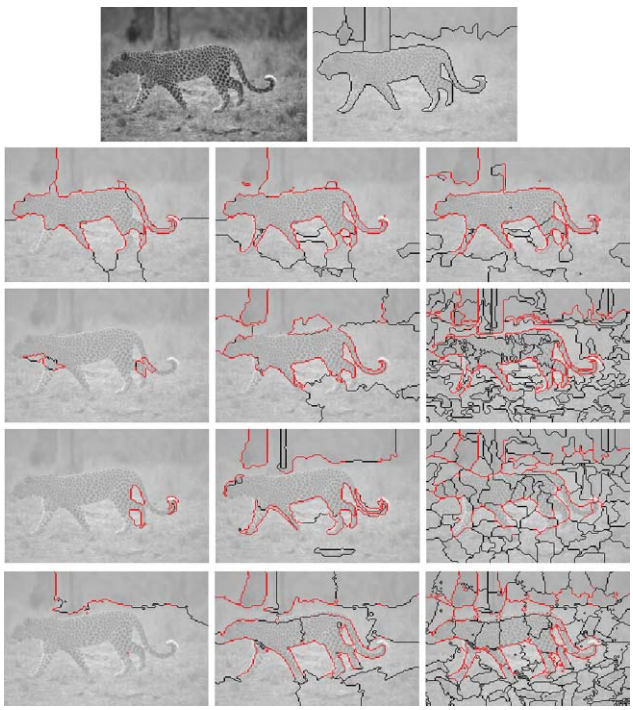

Fig. 8 (Color online) *Top row*: Original image and one human segmentation. From second row to bottom: SE-MinCut segmentations, Mean-Shift results, Local Variation, and Normalized Cuts. On each image, boundary pixels that were matched to the composite human segmentation are shown in *red*. The *leftmost column* corresponds to the parameters that yield highest precision, the *middle column* is for parameters at the middle of the tuning *curve*, and the *rightmost column* corresponds to the parameters that yield highest recall

However, we should point out that human segmentations were produced on the full-size images, and that human observers are most likely using high-level knowledge and reasoning as well as bottom-up perceptually guided visual processing to determine the segmentation for a given image. The observers were instructed to *"Divide each image into pieces, where each piece represents a distinguished thing in the image..."* (Martin et al. 2001). This is not necessarily equivalent to segmenting an image into regions of distinct visual appearance (which is what bottom-up image segmentation algorithms are meant to do). In addition to the above we should remember that the segmentation algorithms rely on simple gray-level similarity between pixels to compute the segmentations. Most objects on the BSD are either heterogeneous-looking, textured, or both, putting the segmentation algorithms at an even greater disadvantage with regard to humans.

A fair comparison between human observers and automatic segmentation algorithms would require the use of images that lack semantic interpretation, thus making the perceptual component of the segmentation task the dominant source of information. This is a task beyond the scope of our paper; however, we have studied the behaviour of the segmentation algorithms on images that have been specifically designed so that regions are indeed characterized by gray-level brightness. The goal is to measure the performance of each of the algorithms when their expectations about what defines a region are actually valid.

To accomplish this, we generated a set of 110 images each of which consists of a small number of uniformly bright regions with smooth boundaries. To make the segmentation problem interesting, each image was corrupted by adding fractal noise. We then ran each of the segmentation algorithms using the same combinations of parameters described above. Figure 9 shows 5 randomly selected images from our test set; the ground truth boundaries obtained from the original, noiseless images; and the segmentations produced by each of the segmentation methods (using the parameters that yield the median precision/recall closest to $(precision = 1, recall = 1)$). We computed tuning curves for each of the algorithms using the procedure described above, these curves are shown in Fig. 10 along with the corresponding error bars showing the median standard error as described above.

Perhaps not surprisingly, all algorithms do significantly better on these images than on the more complex BSD scenes. In particular, recall is quite high with most of the variation occurring along the precision axis. Since on these images the defining characteristic of region boundaries is a large change in image brightness, this is a reasonable outcome. The tuning curves in this case indicate how sensitive each of the algorithms is to local changes in brightness that may be the result of noise. It is clear that normalized cuts and SE-MinCut obtain the best results, and we conjecture that this is due to the global nature of their formulation, which

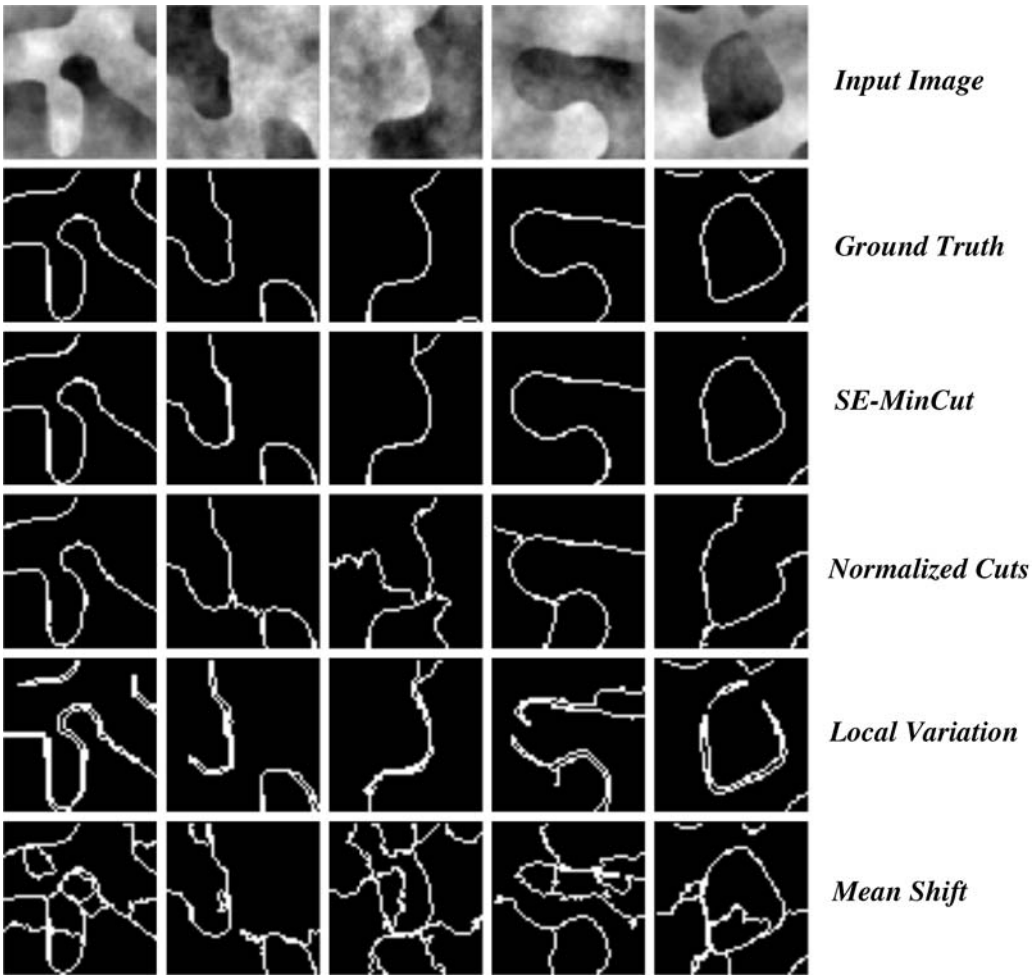

Fig. 9 Five images from our test set, notice that each is composed of a few uniformly bright regions corrupted by fractal noise. The noise can make region boundaries appear smooth. Also shown are the ground-truth boundaries, and the segmentations produced by each of the segmentation algorithms

gives these two methods higher robustness to local brightness variations caused by the fractal noise.

The results on the set of artificial images is intended to give the reader an idea of how well the algorithms perform when faced with images for which they have the right similarity measure (i.e. images in which the relevant cues are indeed image brightness and spatial proximity). Further studies are required to determine the behaviour of these methods for different image cues. We believe that the use of artificial images in addition to natural scenes such as those in the BSD will be helpful in separating the bottom-up component of the segmentation task from the high-level component that depends on context and high-level knowledge. Ultimately, we would like to determine what is the fundamental performance ceiling for bottom-up segmentation algorithms. Even an approximate answer to this question would be very useful in focusing future research efforts toward aspects of the segmentation task that are more likely to yield significant improvements in the current state-of-the-art.

As a final point, and to provide closure with regard to the prior work by Martin (2002), Fig. 11 shows the tuning curves produced using Martin's original bi-partite matching formulation (using the publicly available implementation from Martin and Fowlkes 2001). Notice that the resulting tuning curves are very similar to those obtained using our matching method and a matching distance of $\epsilon = 3$. It is difficult to argue that either of the matching algorithms is superior, but we believe that the simplicity and low computational cost of our matching method are advantageous if a large number of tests have to be performed. In support of this, consider the following: the results presented above required us to generate a total of 35,400 segmentations with their corresponding boundary map, each of these had to be compared to at least 5 human segmentations. This yields at least 177,000 boundary image pairs for which the matching algorithm had to be invoked (this had to be repeated for each value of ϵ). Clearly, having a cheap and effective matching algorithm is desirable. Our code can compute precision

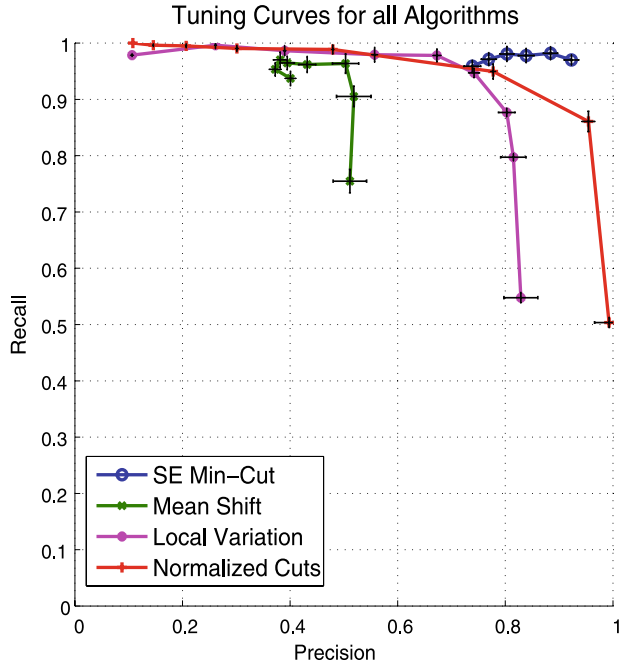

Fig. 10 Tuning *curves* for the segmentation algorithms on the set of 110 artificial images with $\epsilon = 5$. Only the best *curves* for SE-MinCut and Mean Shift are shown

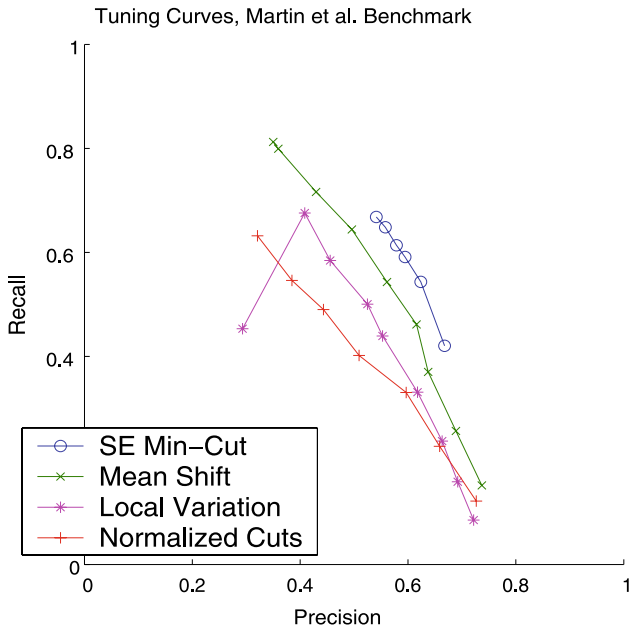

Fig. 11 Tuning *curves* produced using the bi-partite matching algorithm of Martin (2002). Notice that the results are qualitatively similar to those obtained using our matching algorithm and a matching distance of $\epsilon = 3$. The ranking of the segmentation algorithms is consistent, though the distance between the *curves* varies

and recall for 100 segmentations in an average of 1 minute for $\epsilon = 7$ (it becomes faster with smaller ϵ). In comparison, computing precision and recall for the same 100 images using the bi-partite matching method takes 40 minutes. The implementation provided with the BSD includes a faster, approximate matching algorithm. The fast matching algorithm takes on average 2.5 minutes to process 100 segmentations. All these times were measured in a P4, 3.6 GHz. machine. Results shown in Fig. 11 were generated using the slow (more accurate) matching computation.

6 Conclusion

The results presented above agree very well with visual inspection of the segmentations generated by each algorithm. They are also qualitatively similar to the precision/recall scores previously reported for boundary extraction methods (with the observation that, boundary extraction methods require an additional and usually difficult perceptual grouping step to yield closed boundaries). This leads us to conclude that the tuning curves presented above provide a robust and meaningful evaluation of the selected algorithms.

It would be worthwhile to study the trade-off between speed and segmentation quality further. There are several segmentation methods that depend on the agreement between multiple segmentations of the image that differ from one another in some way (see for example Shental et al. 2003, and Cho and Meer 1997). The multiplicity of segmentations could be produced by one of the fast segmentation algorithms studied here by giving it the same amount of time taken by one of the slow segmentation algorithm, and allowing it to run with different parameter settings. Such an enhanced algorithm would be interesting by itself, but it requires a research effort that is beyond the scope of this paper and must remain a matter for future work. The matter of evaluating different similarity measures also remains for future work. One can imagine evaluating the performance of the same algorithm using different similarity measures, perhaps with different image cues or with varying spatial support. This would provide useful information for choosing the optimal similarity measure for segmenting natural images.

Future work aside, segmentations produced by other algorithms (whether they already exist, are extensions from existing methods, or are completely new) can be quickly and easily evaluated using the framework described here. The resulting tuning curves can be compared directly to the results presented in this paper. It is our hope that this benchmark will prove useful for researchers working on image segmentation, and that it will become richer and more complete as results for other segmentation algorithms become available.

References

Belongie, S., & Malik, J. (1998). Finding boundaries in natural images: A new method using point descriptors and area completion. In *European conference on computer vision* (pp. 751–766).

Belongie, S., Fowlkes, C., Chung, F., & Malik, J. (2002). Spectral partitioning with indefinite kernels using the Nyström extension. In *European conference on computer vision* (pp. 21–31).

Blake, A., Rother, A., Brown, M., Perez, P., & Torr, P. (2004). Interactive image segmentation using an adaptive GMMRF model. In *European conference on computer vision* (pp. 428–441).

Borra, S., & 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.

Boykov, Y. Y., & Jolly, M.-P. (2001). Interactive Graph Cuts for optimal boundary & region segmentation of objects in N-D images. In *IEEE international conference on computer vision* (pp. 105–112).

Boykov, Y., & Kolmogorov, V. (2001). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *International workshop on energy minimization methods in computer vision and pattern recognition*. Lecture Notes in Computer Science (pp. 359–374).

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(11), 1222–1239.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(8), 790–799.

Cho, K., & Meer, P. (1997). Image segmentation from consensus information. *Computer Vision and Image Understanding*, *68*(1), 72–89.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(5), 603–619.

Cour, T., Yu, S., & Shi, J. (2006). Normalized cuts Matlab code. Computer and Information Science, Penn State University. code available at http://www.seas.upenn.edu/~timothee/software_ncut/software.html.

Estrada, F. J. (2005). Advances in computational image segmentation and perceptual grouping. PhD thesis, University of Toronto, Department of Computer Science.

Estrada, F. J., & Jepson, A. D. (2004). Spectral embedding and min-cut for image segmentation. In *British machine vision conference* (pp. 317–326).

Estrada, F. J., & Jepson, A. D. (2005). Quantitative evaluation of a novel image segmentation algorithm. In *IEEE international conference on computer vision and pattern recognition*.

Felzenszwalb, P., & Huttenlocher, D. (2004). Image segmentation by local variation code. MIT Artificial Intelligence Lab. Code available at http://people.cs.uchicago.edu/~pff/segment/.

Felzenszwalb, P. F., & Huttenlocher, D. P. (1998). Image segmentation using local variation. In *IEEE conference on computer vision and pattern recognition* (pp. 98–104).

Fowlkes, C., Belongie, S., & Malik, J. (2001). Efficient spatiotemporal grouping using the Nyström method. In *IEEE conference on computer vision and pattern recognition* (pp. 231–238).

Fowlkes, C., Belongie, S., Chung, F., & Malik, J. (2004). Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(2), 214–225.

Gdalyahu, Y., Weinshall, D., & Werman, M. (2001). Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(10), 1053–1074.

Georgescu, B., & Christoudias, C. M. (2003). The Edge Detection and Image segmentation (EDISON) system. Robust Image Understanding Laboratory, Rutgers University. Code available at http://web.archive.org/web/20060617020918/www.caip.rutgers.edu/riul/research/code/EDISON/index.html.

Ishikawa, H., & Geiger, D. (1998). Segmentation by grouping junctions. In *IEEE conference on computer vision and pattern recognition* (pp. 125–131).

Kaufhold, J., & Hoogs, A. (2004). Learning to segment images using region-based perceptual properties. In *IEEE international conference on computer vision and pattern recognition* (vol. 2, pp. 954–961).

Malik, J., Belongie, S., Leung, T., & Shi, J. (1999). Textons, contours and regions: Cue integration in image segmentation. In *IEEE international conference on computer vision* (pp. 918–925).

Malik, J., Belongie, S., Leung, T., & Shi, J. (2001). Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, *43*(1), 7–27.

Martin, D. (2002). An empirical approach to grouping and segmentation. PhD thesis, University of California, Berkeley.

Martin, D., & Fowlkes, C. (2001). The Berkeley segmentation database and benchmark. Computer Science Department, Berkeley University. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/.

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE international conference on computer vision* (pp. 416–425).

Martin, D. R., Fowlkes, C. C., & Malik, J. (2002). Learning to detect natural image boundaries using brightness and texture. In *Neural information processing systems*.

Martin, D., Fowlkes, C., & Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(5), 530–549.

Meila, M., & Shi, J. (2000). Learning segmentation by random walks. In *NIPS* (pp. 873–879).

Sharon, E., Brandt, A., & Basri, R. (2000). Fast multiscale image segmentation. In *IEEE conference on computer vision and pattern recognition* (pp. 70–77).

Sharon, E., Brandt, A., & Basri, R. (2001). Segmentation and boundary detection using multiscale intensity measurements. In *IEEE conference on computer vision and pattern recognition* (pp. 469–476).

Shental, N., Zomet, A., Hertz, T., & Weiss, Y. (2003). Learning and inferring image segmentations using the GBP typical cut algorithm. In *IEEE international conference on computer vision* (pp. 1243–1250).

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905.

Veksler, O. (2000). Image segmentation by nested cuts. In *IEEE conference on computer vision and pattern recognition* (pp. 339–344).

Wang, S., & Siskind, J. M. (2001). Image segmentation with minimum mean cut. In *IEEE international conference on computer vision* (pp. 517–524).

Wang, S., & Siskind, J. M. (2003). Image segmentation with ratio cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(6), 675–690.

Wu, Z., & Leahy, R. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*(11), 1101–1113.

Yu, S. X. (2004). Segmentation using multiscale cues. In *IEEE international conference on computer vision and pattern recognition* (pp. 247–254).

Yu, S. X. (2005). Segmentation induced by scale invariance. In *IEEE international conference on computer vision and pattern recognition*.

Yu, S. X., & Shi, J. (2003). Multiclass spectral clustering. In *IEEE International Conference on Computer Vision* (pp. 313–319).