Quantitative Evaluation of a Novel Image Segmentation Algorithm

Francisco J. Estrada and Allan D. Jepson
Department of Computer Science
University of Toronto
Toronto, ON., M5S 3G4, Canada
{strider,jepson}@cs.utoronto.ca

Abstract

We present a quantitative evaluation of SE-MinCut, a novel segmentation algorithm based on spectral embedding and minimum cut. We use human segmentations from the Berkeley Segmentation Database as ground truth and propose suitable measures to evaluate segmentation quality. With these measures we generate precision/recall curves for SE-MinCut and three of the leading segmentation algorithms: Mean-Shift, Normalized Cuts, and the Local Variation algorithm. These curves characterize the performance of each algorithm over a range of input parameters. We compare the precision/recall curves for the four algorithms and show segmented images that support the conclusions obtained from the quantitative evaluation.

1 Introduction

Image segmentation continues to be a challenging problem. Despite continuous improvements, the task of accurately segmenting an arbitrary image remains basically unsolved. At the same time, only recently has a significant effort been dedicated to developing suitable quantitative measures of segmentation quality that can be used to evaluate and compare segmentation algorithms.

In this paper we present a quantitative evaluation of a novel technique that uses spectral embedding together with the minimum-cut algorithm to generate high quality segmentations. We compare SE-MinCut against three of the leading image segmentation algorithms: the Mean-Shift algorithm of Comaniciu and Meer [5, 6], the Normalized Cuts algorithm of Shi and Malik [16], and the Local Variation algorithm of Felzenszwalb and Huttenlocher [10]. The algorithms are evaluated on the Berkeley Segmentation Database [14, 13], which provides human segmentations for a large collection of images.

The contributions of our paper are: 1) A simple definition of precision and recall measures for segmentation quality, these measures are sensitive to over- and under-segmentation and can be computed efficiently. 2) The introduction of precision/recall curves that characterize the segmentation quality of a given algorithm. These curves allow for a robust comparison of segmentation quality that is independent of the choice of input parameters. 3) A quantitative evaluation and comparison of segmentation quality for the four segmentation algorithms over the complete BSD which, to our knowledge, is the first direct comparison of current segmentation algorithms presented in the literature.

We will start with an overview of the SE-MinCut framework and place it in the context of current research involving minimum-cut for image segmentation. We will then show segmentation results for the four algorithms on images from the BSD, proceed to the complete quantitative evaluation of the algorithms, and discuss our results.

2 Spectral Embedding and Min-Cut

The minimum cut algorithm is a graph-partitioning technique that has received a significant amount of attention in recent years as a useful framework for image segmentation. Any image $I(\vec{x})$ can be viewed as a graph $G(V, E)$ where V is a set of nodes that correspond to pixels in the image, and E is a set of edges that connect nodes in the graph. E is usually set up so that only neighboring pixels are connected, and the strength of the connection between two pixels is given by the weight of the edge that spans them. The graph $G(V, E)$ is usually stored as an affinity matrix. For an $n \times m$ image, we can build an $nm \times nm$ affinity matrix A whose elements $A_{i,j}$ are proportional to the similarity between pixels i and j and correspond to the weight of the edges $E_{i,j}$. Given this affinity matrix, the min-cut algorithm computes the subset of edges $E_{i,j}$ that must be removed from $G(V, E)$ so that the graph is partitioned into two disjoint sets, and the sum of weights for the removed edges is minimal.

Computation of the minimum cut usually involves defining two special nodes called source and sink that are linked to elements within one of the disjoint sets in $G(V, E)$.

The cut itself can be computed using a max-flow formulation [4]. The minimum cut framework is attractive for several reasons: the min-cut separating source and sink nodes can be computed efficiently [2, 4], it is guaranteed to be a global minimum, and given appropriate source and sink regions, the resulting cut will partition an image along salient image boundaries. Several algorithms have been proposed that use minimum cut for image segmentation, Wu and Leahy [18] propose trying every pair of pixels as source and sink and selecting from the resulting partitions the cut with the minimum weight, while Veksler [17] proposes placing sink regions outside the image and using individual image pixels as source. However, both of these techniques can become impractical due to the large number of partitions that have to be computed. At the same time, the final selected cut may not correspond to salient image structure. Boykov et al. [2, 1] on the other hand, rely on user interaction to select suitable source and sink regions to produce a segmentation efficiently. Finally, Boykov et al. [3] propose a method to optimize an initial label field using min-cut to perform label swap and region expansion operations.

2.1 Spectral Embedding

We will briefly review the algorithm proposed in [8], which uses spectral embedding to define suitable source and sink regions for min-cut. The algorithm uses a simple affinity measure based on the grayscale difference between neighboring pixels

$$A_{i,j} = e^{-(I(\vec{x}_i)-I(\vec{x}_j))^2/(2\sigma^2)}, \qquad (1)$$

where σ represents the typical gray-level variation between neighboring pixels. Without loss of generality, we assume A is sparse, and the entries $A_{i,j}$ are non-zero only for elements within the 5×5 pixel neighborhood centered at pixel $\vec{x}_i$ (although any neighborhood structure including the complete image can be used).

We generate a Markov matrix M by normalizing the columns of A using $D_j \equiv \sum_{k=1}^{nm} A_{k,j}$, and $M_{i,j} = A_{i,j}/D_j$. M defines a Markov chain representing a random walk over image pixels (see Meila and Shi [15]). Let $p_t(\vec{x}_j)$ denote the probability that a particle undergoing the random walk is at pixel $\vec{x}_j$ at time t, and let the nm dimensional vector $\vec{p}_t$ represent $p_t(\vec{x})$. We can estimate $\vec{p}_{t+1}$ using

$$\vec{p}_{t+1} = M\vec{p}_t, \quad \text{where } M = AD^{-1}, \qquad (2)$$

and D is a diagonal matrix formed by the normalization factors D_j.

Given an initial distribution $\vec{p}_0$, it follows from (2) that the distribution after t steps of the random walk is $\vec{p}_t = M^t\vec{p}_0$. Here, M^t can be expressed in terms of its eigenvectors and eigenvalues, obtained conveniently from

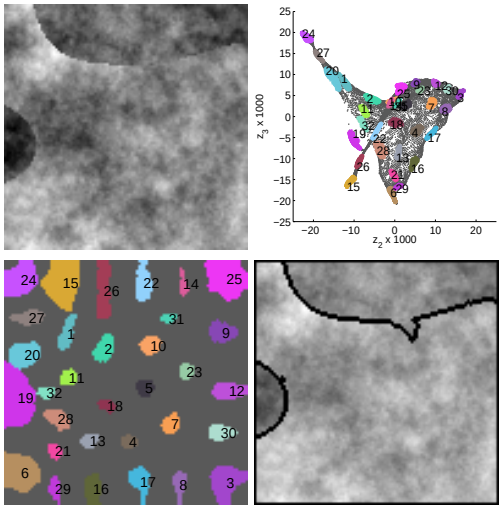

Figure 1. Fractal image (3 regions plus noise, 100×100 pixels), seed regions generated by our algorithm, second and third components of the embedding, and final segmentation. For this image $d = 15$ and t was calculated so that $|\lambda_{d+1}|^t \simeq 1/3$.

the *similar, symmetric* matrix $L = D^{-1/2}MD^{1/2} = D^{-1/2}AD^{-1/2}$. Since L is symmetric, its eigendecomposition has the form $L = U\Delta U^T$, where U is an orthogonal matrix whose columns are the eigenvectors of L, and Δ is a diagonal matrix of eigenvalues $\lambda_i, i = 1, \ldots, nm$. We assume that the eigenvalues have been sorted in decreasing order of magnitude. It can be shown from the shape of D and A that $\lambda_i \in (-1, 1]$ and that at least one eigenvalue is equal to 1. So, without loss of generality, we assume that $\lambda_1 = 1$. This eigenvalue problem is equivalent to that of Shi and Malik [16] except for the choice of the affinity matrix.

From the eigendecomposition of L and (2) we obtain

$$\vec{p}_t = M^t\vec{p}_0 = (D^{1/2}U)\Delta^t(U^TD^{-1/2})\vec{p}_0. \qquad (3)$$

Suppose that we start the random walk at pixel $\vec{x}_i$. The initial distribution is given by $\vec{p}_{0,i} = \vec{e}_i$, which is a vector with 1 in the i^{th} row and zeros elsewhere. Then, after t steps of the random walk, we obtain the diffused distribution $\vec{p}_{t,i} = M^t\vec{e}_i$, which we call a blur kernel. Since the probability that a particle will jump across a strong image boundary is small, blur kernels will diffuse little across such boundaries, and the stronger the boundary, the smaller the diffusion across it. As a result of this, blur kernels for pixels separated by a strong boundary will have their probability masses distributed over different subsets of pixels, and their inner product $\vec{p}^T_{t,i}\vec{p}_{t,j}$ must vanish. Conversely, blur kernels for neighboring pixels in homogeneous regions will be sim-

ilar, and their inner product will be large.

We approximate these blur kernels using only the top d eigenvectors and eigenvalues of the Markov matrix M

$$\vec{p}_{t,i} \simeq \vec{q}_{t,i} \equiv (D^{1/2}U_d)\vec{w}_{t,i}, \quad (4)$$

where $\vec{w}_{t,i} = \Delta_d^t U_d{}^T D^{-1/2}\vec{e}_i$ is the projected blur kernel for pixel $\vec{x}_i$ at time t, U_d contains the first d columns of U, and Δ_d is a $d \times d$ diagonal matrix formed with the first d eigenvalues λ_i. We can approximate the inner product of the original blur kernels with $\vec{p}_{t,i}^T\vec{p}_{t,j} \simeq \vec{q}_{t,i}^T\vec{q}_{t,j} = \vec{w}_{t,i}^T Q\vec{w}_{t,j}$, with $Q = U_d{}^T DU_d$. Defining $\vec{z}_{t,i} = Q^{1/2}\vec{w}_{t,i}$ we have $\vec{q}_{t,i}^T\vec{q}_{t,j} = \vec{z}_{t,i}^T\vec{z}_{t,j}$. The approximation error depends on the magnitude of the terms $|\lambda_i|^t$ for $i > d$, the largest of which is $|\lambda_{d+1}|^t$. We find a good approximation when d and t are chosen so that $|\lambda_{d+1}|^t < 1/3$. Fig. 1b shows a plot of two of the dimensions in the embedding. Other properties and details of the embedding are discussed in [8].

2.2 Seed Region Selection

The projected blur kernels cluster in the d-dimensional space induced by the embedding. Because blur kernels from homogeneous image regions are similar to one another, these clusters correspond to local groups of similar pixels. Furthermore, the density of blur kernels between neighboring clusters is indicative of whether there is a smooth continuation between the groups of pixels corresponding to the clusters. A low density valley between two clusters indicates the existence of an image boundary.

To find clusters, we map the $\vec{z}$ points onto the unit sphere $\vec{s}_{t,i} = \vec{z}_{t,i}/||\vec{z}_{t,i}||$, and generate a set of initial guesses for the cluster centers $\{\vec{m}_k\}_{k=1}^K$ by randomly sampling points from $\vec{s}_{t,i}$. Successive samples are constrained to have an inner product of at least τ_0 from previous samples with $\tau_0 = 0.8$. The number of clusters K is chosen to be the maximum number of samples that can be drawn before all points $\vec{s}_{t,i}$ have an inner product of at least τ_0 with some center $\vec{m}_k$, this ensures that we have enough clusters to span the embedding.

The location of each cluster center $\vec{m}_k$ is then updated iteratively by computing a weighted average of the points $\vec{s}_{t,i}$ in the neighborhood of $\vec{m}_k$

$$\vec{m}_k' = \frac{\sum_{\vec{s}_{t,i}^T\vec{m}_k \geq \tau_1} (\vec{s}_{t,i}^T\vec{m}_k)\vec{s}_{t,i}}{\sum_{\vec{s}_{t,i}^T\vec{m}_k \geq \tau_1} (\vec{s}_{t,i}^T\vec{m}_k)}. \quad (5)$$

The weights correspond to the inner product between the points $\vec{s}_{t,i}$ and the current estimate for the cluster center. Only the points on the unit sphere whose inner product with regard to the cluster center is greater than a threshold τ_1 contribute to the weighted average. In what follows we use $\tau_1 = 0.95$.

The resulting cluster centers can be used to define seed regions in the image that consist of groups of similar pixels. Seed region S_k originating from cluster k is obtained by thresholding the inner products of $\vec{s}_{t,i}$ with $\vec{m}_k$, $S_k \equiv \{\vec{x}_j | \vec{s}_{t,j}^T\vec{m}_k \geq \tau_1\}$. Seed regions obtained in this fashion are illustrated in Fig. 1. Notice here two important properties of the seed regions: they include a significant fraction of the image area, and they do not cross salient image boundaries. Combinations of seed regions are used to define source and sink nodes for min-cut. Details about how this is accomplished are found in [8].

We compute a minimum cut for every source/sink combination and store the resulting partitions. After all the cuts are done, we generate an intermediate segmentation as the intersection of all the partitions generated by min-cut. This intermediate segmentation is usually over-segmented, so we perform an additional stage of region merging. Region merging is performed by examining the distribution of links along the boundary between two regions. For partitions that correspond to salient image boundaries most links should be weak (smaller than .1 in our case). If less than a certain portion τ_m of the links along the boundary are suitably weak, we merge the regions.

2.3 Experimental Results

Figure 2 shows the segmentation results generated with our algorithm on several images from the Berkeley Segmentation Database (BSD) [13]. Results generated with Normalized Cuts [16], Mean Shift [5, 6], and the Local Variation [10] algorithm are shown for comparison. The choice of parameters for each algorithm is noted in the figure and is based on the comparative graphs described in the next section. These results indicate that the regions extracted with our algorithm better capture the perceived structure of the images, and the boundaries of regions are more closely aligned with salient image boundaries (this is an expected result of the use of min-cut). The next section presents a quantitative comparison of the segmentation results produced with the four algorithms on the BSD.

3 Comparing Segmentation Algorithms

The images in Fig. 2 offer compelling evidence that our segmentation algorithm performs well on a variety of images from different domains. Such visual comparisons have been used extensively in the past as a means of illustrating the capabilities of segmentation algorithms. However, we would like a quantitative measure of segmentation quality that can be used to compare the algorithms directly.

In this section, we will evaluate the performance of our algorithm on the Berkeley Segmentation Database (BSD) [13]. We will discuss the organization of the BSD,

Figure 2. Segmentation results, from top to bottom: Original image, SE-MinCut, Mean-Shift, Local Variation, Normalized Cuts, and human segmentations. Parameters for the algorithms are: $d = 40$, and $\tau_m = .25$ for SE-MinCut, spatial bandwidth $SB = 4$, and range bandwidth $RB = 6$ for Mean-Shift, $k = 100$ for Local variation, and the number of regions for Normalized Cuts was set to 64 (see text for the explanation of the choice of parameters). SE-MinCut clearly generates segmentations that more closely capture the structure of the scenes, and does so with less over-segmentation.

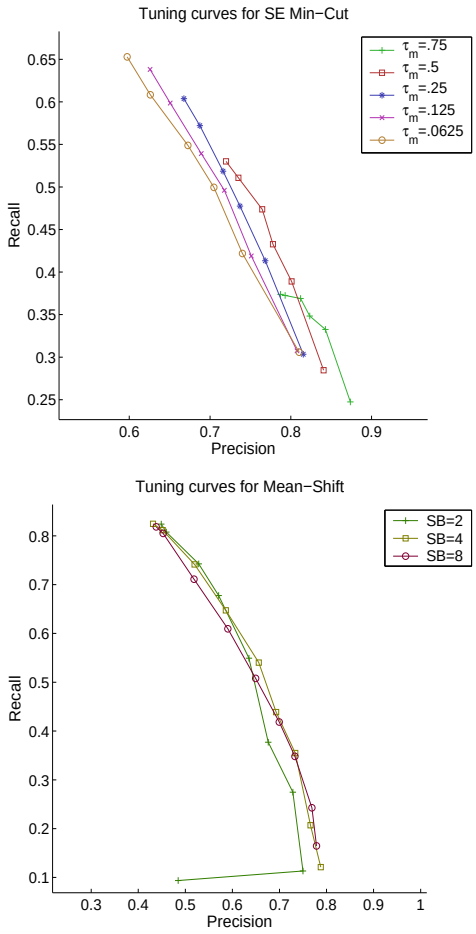

Figure 3. Tuning curves for SE-MinCut (top) and Mean-Shift (bottom) (notice that the scales are not the same). For SE-MinCut the points along each curve correspond to variations of the parameter d within $[5, 40]$. For Mean-Shift the points along each curve correspond to variations of the range bandwidth within $[1 - 20]$. From the above plots, we selected for comparison the curve for $\tau_m = .25$ for SE-MinCut, and the curve for a spatial bandwidth $SB = 4$ for Mean Shift. These curves offer the best compromise between precision and recall for these algorithms.

develop appropriate measures of segmentation quality, and use these measures to generate tuning curves that characterize the behavior of each algorithm over a range of input parameters. Finally, we will present quantitative performance results for SE-MinCut [8], Normalized Cuts [16], Mean-Shift [5, 6], and Local Variation [10].

3.1 Evaluation Measures and the BSD

The current public version of the BSD [13] consists of 300 colour images. The images have a size of 481×321 pixels and are divided into two sets, a training set containing 200 images that can be used to tune the parameters of a segmentation algorithm, and a testing set that contains the remaining 100 images. For each image, and separately for the colour and grayscale versions of the image, a set of human segmentations is provided. This set contains between 4 and 7 segmentations specified as labeled images in a special format.

Martin et al. [14] show that the human segmentations, though varying in detail, are consistent with one another in that regions segmented by one subject at a finer level of

detail can be merged consistently to yield the regions extracted by a different subject at a coarser level of detail. Based on this consistency, Martin et al. propose two measures to evaluate segmentation performance and use them to compare the Normalized Cuts algorithm against the human segmentations. However, these measures are not sensitive to over and under-segmentation (both of which can yield low error values). This effect was also noted by Martin [12] who proposed several alternative measures for comparing segmentation algorithms. Among these, he discussed the use of precision/recall values that characterize the agreement between the region boundaries of two segmentations. In his work, these measures are computed using a bi-partite matching formulation that matches boundary pixels using their location and orientation.

Here we propose a simpler method for matching boundaries between a source segmentation S_1 and a target segmentation S_2. Matching is performed by examining the immediate neighborhood of each boundary pixel b_i in S_1 for potential matches. The neighborhood is searched in order of increasing distance in a fixed pattern following the pixel grid, up to a maximum distance of τ_d. Pixels are matched to the nearest boundary element b_x in S_2 as long as there is no other boundary pixel b_j in S_1 between b_i and b_x. This procedure allows for many-to-one matchings between boundary pixels, which is useful when corresponding boundaries in the two segmentations have different length due to small localization errors. However, it is possible for (at most) two boundaries in the source segmentation to be matched to the same boundary in the target segmentation. This would occur when the target boundary is 'sandwiched' between the two boundaries from the source segmentation.

Using this matching strategy, we define precision and recall to be proportional to the total number of unmatched pixels between two segmentations S_1 and S_2. Unmatched pixels are those for which a suitable match cannot be found within a particular distance threshold τ_d. Our precision/recall measures are defined as follows:

$$Precision(S_1, S_2) = \frac{Matched(S_1, S_2)}{|S_1|}, \qquad (6)$$

where $Matched(S_1, S_2)$ is the number of boundary pixels in S_1 for which a suitable match was found in S_2, and $|S_1|$ is the total number of boundary pixels in S_1. Similarly

$$Recall(S_2, S_1) = \frac{Matched(S_2, S_1)}{|S_2|}. \qquad (7)$$

Precision is low when there is significant over-segmentation, or when a large number of boundary pixels have localization errors greater than τ_d. A low recall value is typically the result of under-segmentation and indicates failure to capture salient image structure.

The principal advantage of using precision and recall for the evaluation of segmentation results is that we can compare not only the segmentations produced by different algorithms, but also the results produced by the same algorithm using different input parameters. By systematically changing the value of the input parameters, we can produce tuning curves that characterize the performance of a particular segmentation technique for a wide range of its input parameters, thus providing a more complete evaluation of the quality of the segmentations that can be generated with that algorithm. The tuning curves also allow for the selection of input parameters that will yield the desired combination of precision and recall within the operating range of each algorithm.

3.2 Experimental Setup

We will use the measures defined above to compare the segmentations produced by the four algorithms. However, neither our algorithm nor the Normalized Cuts implementation from [7] can work directly on the images from the BSD due to their size. In what follows, we will use the grayscale images from the BSD after they have been appropriately blurred and downsampled by a factor of 4 to a size of 121×88 pixels. The human segmentations of each image have also been downsampled to the appropriate size.

Since human segmentations of the same image vary in level of detail, precision and recall would change significantly depending on which target segmentation is chosen. Instead of comparing against individual human segmentations, we choose to compare against a composite segmentation that is formed by the union of all region boundaries extracted by human observers for the same image. This had already been suggested by Martin [12]. Since the automatic segmentation results and the human segmentations in the BSD consist of labeled images, we use an identical procedure to generate the region boundaries for all the segmentations involved. This procedure is simple and consists of marking as a boundary any pixel that has at least 1 neighbor with a different label.

We used the implementations of the algorithms made available by the authors. For Normalized Cuts see [7], for Local Variation see [9], and for Mean-Shift see [11]. We tested each algorithm over a range of values for its input parameters. Matching was carried out using a distance threshold $\tau_d = 5$, which is reasonably large given the resolution of the images. Since there is no training phase involved, we ran each algorithm for each combination of input parameters over the full 300 images of the BSD. For each of these runs, we calculate the median precision and recall values and use these values to generate tuning curves that characterize the algorithm's performance. For algorithms with a single input parameter (Normalized Cuts and Local Varia-

tion), we obtain a single curve. For algorithms with two parameters (SE-MinCut, and Mean-Shift), we get a curve for each value of one input parameter, while the values along the curve correspond to variations of the second parameter. The ranges for the input parameters of each algorithm were determined experimentally to produce significant over- and under-segmentation at the extremes, while values within this range were chosen so as to yield segmentations with perceptible differences.

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 values 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 [10]); 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 [5, 6] for details). In practice, the segmentation software for Mean-Shift uses an additional parameter (the size in pixels of the smallest allowed region). We didn't test the effect of this parameter; since it only imposes an artificial limit on over-segmentation, it was kept fixed at 25 pixels (which is the same size as the search window used for boundary matching). Experimentation showed that the largest differences between segmentations were obtained when varying the range bandwidth parameter. Thus, we evaluated the algorithm using three values for the spatial bandwidth within $[2, 8]$, and for each of these values, we computed a tuning curve that corresponds to variations of the range bandwidth within $[1, 20]$.

For SE-MinCut, we tested two parameters. One is d, which determines the number of eigenvectors to use in the embedding. We then use d to determine t so that $|\lambda_{d+1}|^t \simeq 1/3$ as described in the previous section. The other parameter is the merging threshold τ_m. The remaining internal parameters mentioned in the text were kept fixed at the original values proposed in [8]. The largest variation between segmentations is obtained by changing the value of d. Following the same methodology used with Mean-Shift, we chose 5 values for the merging threshold between $[1/16, 3/4]$, and for each of these we computed a tuning curve that corresponds to variations of d within $[5, 40]$.

Finally, we generated precision and recall data for human segmentations. Given the set of human segmentations for a particular image, we selected each segmentation in turn and compared it against a composite of the remaining segmentations for that same image. We then computed the median precision and recall for all observers. The resulting precision and recall points are useful for comparing the performance of segmentation algorithms against human observers.

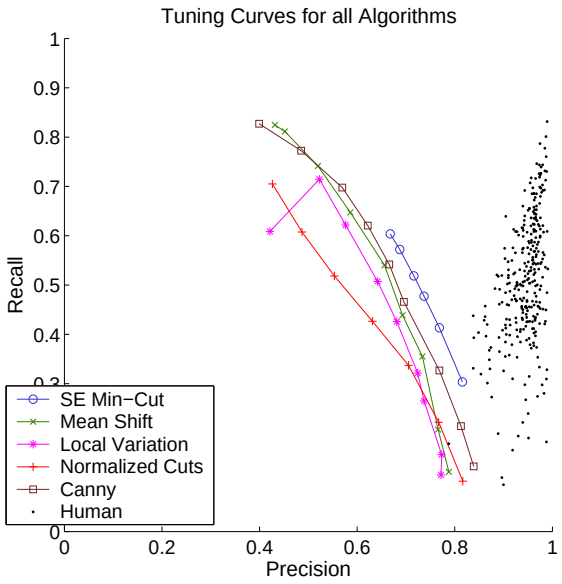

Figure 4. Tuning curves for all segmentation algorithms plus canny edges, and precision/recall values for human segmentations. SE-MinCut clearly has the best performance of all algorithms over its range of input parameters.

4 Tuning Curves and Algorithm Comparison

Since we have 3 tuning curves for the Mean-Shift algorithm, and 5 curves for the SE-MinCut algorithm, we selected for comparison the curves that corresponds to the parameter combinations that yield the best compromise between precision and recall. Fig. 3 shows the tuning curves obtained for SE-MinCut and Mean-Shift. We have chosen the tuning curve that corresponds to $\tau_m = .25$ as the representative for SE-MinCut, and the tuning curve that corresponds to a spatial bandwidth $SB = 4$ is selected as the representative for Mean-Shift.

The selected curves for SE-MinCut and Mean-Shift together with the curves for Normalized Cuts and Local Variation are shown in Fig. 4. The figure also shows the points that correspond to the median precision and recall of human segmentations for the 300 images of the BSD. This figure shows that SE-MinCut outperforms the other segmentation algorithms across its range of input parameters. For a given recall value, SE-MinCut achieves the highest precision; conversely, for a given precision value, our algorithm achieves the best recall. This improved performance indi-

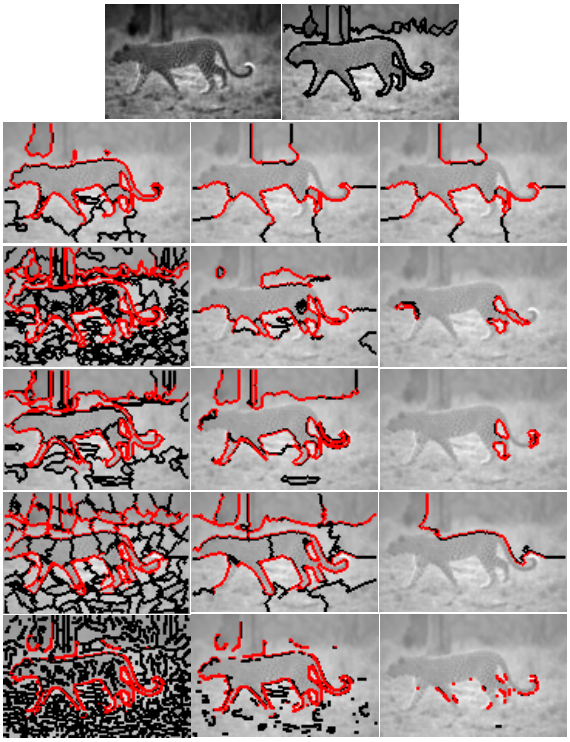

Figure 5. From top to bottom: Original image and composite human segmentation, SE-MinCut segmentations, Mean-Shift results, Local Variation results, Normalized Cuts segmentations, and Canny edges. Boundary pixels that were matched to the composite human segmentation are shown in red. For each algorithm the leftmost column corresponds to the highest possible recall, the middle column is for the center of the tuning curve, and the rightmost column corresponds the highest precision.

cates that SE-MinCut finds salient regions with less over-segmentation, and the boundaries of these regions correspond more closely to the human-marked boundaries. This agrees with the results displayed in Fig. 2, the segmentations shown there correspond to the parameters that yield a recall value closest to .6 for each algorithm. Notice that the Local Variation curve has two points close to .6 on the recall axis; we chose the point with the largest precision value.

The reader may wonder about the significance of the larger range of recall values achieved by the other algorithms when compared to SE-MinCut. Figure 5 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, highest precision, and middle of the corresponding tuning curves. 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 partition small, high-contrast regions that do not necessarily capture the structure of the image. Over-segmentation is limited for SE-MinCut by the fact that the leading eigenvectors 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 eigenvectors used during segmentation. Results in Fig. 5 also agree visually with the information provided by the tuning curves. Over-segmentation is characterized in the curves by high recall but low precision, and the converse is true for under-segmented images.

We have also included results from running the Canny edge detector (with no hysteresis thresholding) on the input images. A tuning curve was produced by varying a single threshold on normalized gradient magnitude between 0.05 and 0.6. Region boundaries are 2 pixels wide, so the Canny edges were artificially dilated to 2 pixels before computing precision/recall values for each test. It is worth noting that the comparison with the Canny edge detector is unfair in that Canny edges are not required to form closed contours (see Fig. 5), and do not produce a segmentation of the image. We show the Canny results here to provide a hint of how well the segmentation algorithms perform purely as boundary detectors.

Perhaps not surprisingly, there is a significant gap in performance between all the segmentation algorithms and human observers (though humans segmented the images at high resolution, which gives them an advantage especially with finer image structure). The data for human segmentations confirms the observation that humans segment images consistently; this is reflected in the high precision scores obtained by most human segmentations. On the other hand, the large variation in recall scores reflects the fact that different observers will segment an image at different levels of detail. Some images show more variability than others and thus receive a lower recall score.

Though the gap is still significant, we believe that there has been consistent progress in the field of image segmentation and expect the gap to become smaller as research in image segmentation continues. We should note that we expect the tuning curves shown here (and in general, any measure designed to compare segmentation results) to be sensitive to image resolution. However, we have computed tuning curves using the original precision/recall definitions and matching algorithm of Martin [12], and the results are very similar. We are confident that our results are sound and provide a fair comparison between the algorithms. It

is worth mentioning that SE-MinCut produces good quality segmentations using a simple gray-scale based affinity measure. We expect that better affinity measures will enhance the quality of the segmentations produced by our algorithm.

In terms of run-time, the best performance is achieved by the Local Variation algorithm, which takes around 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; finally, SE-MinCut takes between 1 and 7 min. depending on the number of eigenvectors used for the embedding. These times were measured on a 1.9GHz Pentium IV machine. Both Normalized Cuts and SE-MinCut are partly implemented in Matlab. In the case of SE-MinCut there are three main components: the spectral embedding and region proposal step, min-cut, and the merging stage. We expect that the first and last of these components can be optimized for increased efficiency. Our goal here was to show that min-cut with automatic source and sink selection, followed by a simple post-processing stage can produce higher quality segmentations than other current algorithms.

5 Conclusion

We have shown that the SE-MinCut algorithm is capable of generating high quality segmentations using a simple affinity measure based on gray-scale similarity. We presented simple precision and recall measures of segmentation quality, and a matching algorithm that can be used to compute them efficiently. With these measures, we evaluated the quality of the segmentations produced by several well known algorithms over the Berkeley Segmentation Database, and presented tuning curves that characterize the performance of these algorithms over a range of their input parameters. This is to our knowledge the first quantitative comparison of leading segmentation algorithms on a standard set of images.

The tuning curves show that the SE-MinCut algorithm produces better segmentations for any desired value of recall within its tuning curve. Though the other algorithms have a wider range of recall values, larger recall and precision values come at the cost of significant over- or under-segmentation. It should be noted that there is no in-principle reason why finer structure cannot be segmented with SE-MinCut using the recursive approach that has also been proposed for Normalized Cuts. Specifically, once coarse regions have been extracted (and we know from the above that such regions are likely to agree closely with perceptually salient image structure), each region can be individually segmented using SE-MinCut.

References

[1] Y. Boykov and M. Jolly. *Interactive Graph Cuts* for optimal boundary & region segmentation of objects in n-d images. In *ICCV*, pages 105–112, 2001.

[2] Y. Boykov and V. Kolmogorov. 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, pages 359–374, 2001.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast, approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.

[4] Cherkassky and Goldberg. On implementing push-relabel method for the maximum flow problem. In *Proc. IPCO-4*, pages 157–171, 1995.

[5] D. Comaniciu and P. Meer. Robust analysis of feature spaces: Color image segmentation. In *CVPR*, pages 750–755, 1997.

[6] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *ICCV*, pages 1197–1203, 1999.

[7] T. Cour, S. Yu, and J. Shi. Normalized cuts matlab code. code available at **http://www.cis.upenn.edu/˜jshi/software/**.

[8] F. J. Estrada, A. D. Jepson, and C. Chennubhotla. Spectral embedding and min cut for image segmentation. In *BMVC*, pages 317–326, 2004.

[9] P. Felzenszwalb and D. Huttenlocher. Image segmentation by local variation code. code available at **http://www.ai.mit.edu/people/pff/seg/seg.html**.

[10] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. In *CVPR*, pages 98–104, 1998.

[11] B. Georgescu and C. M. Christoudias. The Edge Detection and Image SegmentatiON (EDISON) system. code available at **http://www.caip.rutgers.edu/riul/research/code.html**.

[12] D. Martin. *An Empirical Approach to Grouping and Segmentation*. PhD thesis, University of California, Berkeley, 2002.

[13] D. Martin and C. Fowlkes. The Berkeley segmentation database and benchmark. **http://www.cs.berkeley.edu/projects/vision/grouping/segbench/**.

[14] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, pages 416–425, 2001.

[15] M. Meila and J. Shi. Learning segmentation by random walks. In *NIPS*, pages 873–879, 2000.

[16] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 2000.

[17] O. Veksler. Image segmentation by nested cuts. In *CVPR*, pages 339–344, 2000.

[18] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *PAMI*, 15(11):1101–1113, Nov. 1993.