Chapter 5

Segmentation Results and Quantitative Evaluation

In this chapter we present experimental results for our algorithm. First, we show segmentations produced by SE-MinCut on several indoor and outdoor scenes. We include for comparison the segmentation results obtained with three alternative segmentation algorithms: Mean-Shift [21, 22], Local Variation [29], and Normalized Cuts [98]. These algorithms were chosen because they are representative of current methods in image segmentation, and because implementations of the algorithms have been made available by their authors.

Following the visual comparison, we lay the foundations for a quantitative study of segmentation quality. The goal of this part of our research is to offer quantitative evidence that the use of min-cut together with automatically generated seed regions results in superior segmentation quality. For this purpose, we use the Berkeley Segmentation Database of human segmented images. We examine the problem of evaluating segmentation quality, propose suitable metrics that can be evaluated efficiently, and present comparative results of segmentation quality for SE-MinCut and the three alternative segmentation methods mentioned above. Finally, we show a quantitative comparison of the segmentation algorithms on synthetic data.

5.1 Segmentation Results

Here we present the segmentations produced by our algorithm on several images. For all of our tests we use $d = 15$; we chose this value experimentally by noting that it is large enough to yield seed regions within each salient image region in our test images. Given d, the value for t was calculated so that $|\lambda_{d+1}|^t \simeq 1/3$. Our implementation uses the push-relabel min-cut code described in [18]. [1] We also use a Matlab implementation of the multi-scale SVD algorithm described in [17].

Figure 5.1 shows results for indoor scenes, while Figure 5.2 shows the results for outdoor images. In both figures, the segmentations obtained with the Normalized Cuts algorithm [98] (using the implementation from [23]), the Mean Shift algorithm [21] (using the implementation provided by the authors at [37]), and the Local Variation algorithm [29] (for which we use the implementation available at [28]) are shown for comparison purposes.

The Normalized Cuts algorithm requires as input the number of regions into which the image should be partitioned. The number of regions used was the same as the number of regions produced by our algorithm (though it may be the case that several of the regions produced by SE-MinCut cover only a couple of pixels along some image boundary, and do not generate perceivable over-segmentation). We set the parameters of the Local Variation algorithm once, and used the same parameters to process all test images. The parameters were chosen so that the resulting segmentations would compare most favorably with regard to our own results. The parameters for the Mean Shift algorithm were set using the same criterion.

From the figures shown here, it is clear that the boundaries of regions produced by our algorithm agree more closely with the boundaries of salient image regions; the min-cut algorithm is indeed capable of finding the salient boundaries even using simple affinities based on gray-level difference in a pixel's 8-neighborhood. In contrast, Normalized Cuts uses a more complicated affinity measure based on gray-level difference, spatial proximity, and intervening

[1] We thank A. V. Goldberg for supplying C-code (i.e. hi_pr.c) for this algorithm

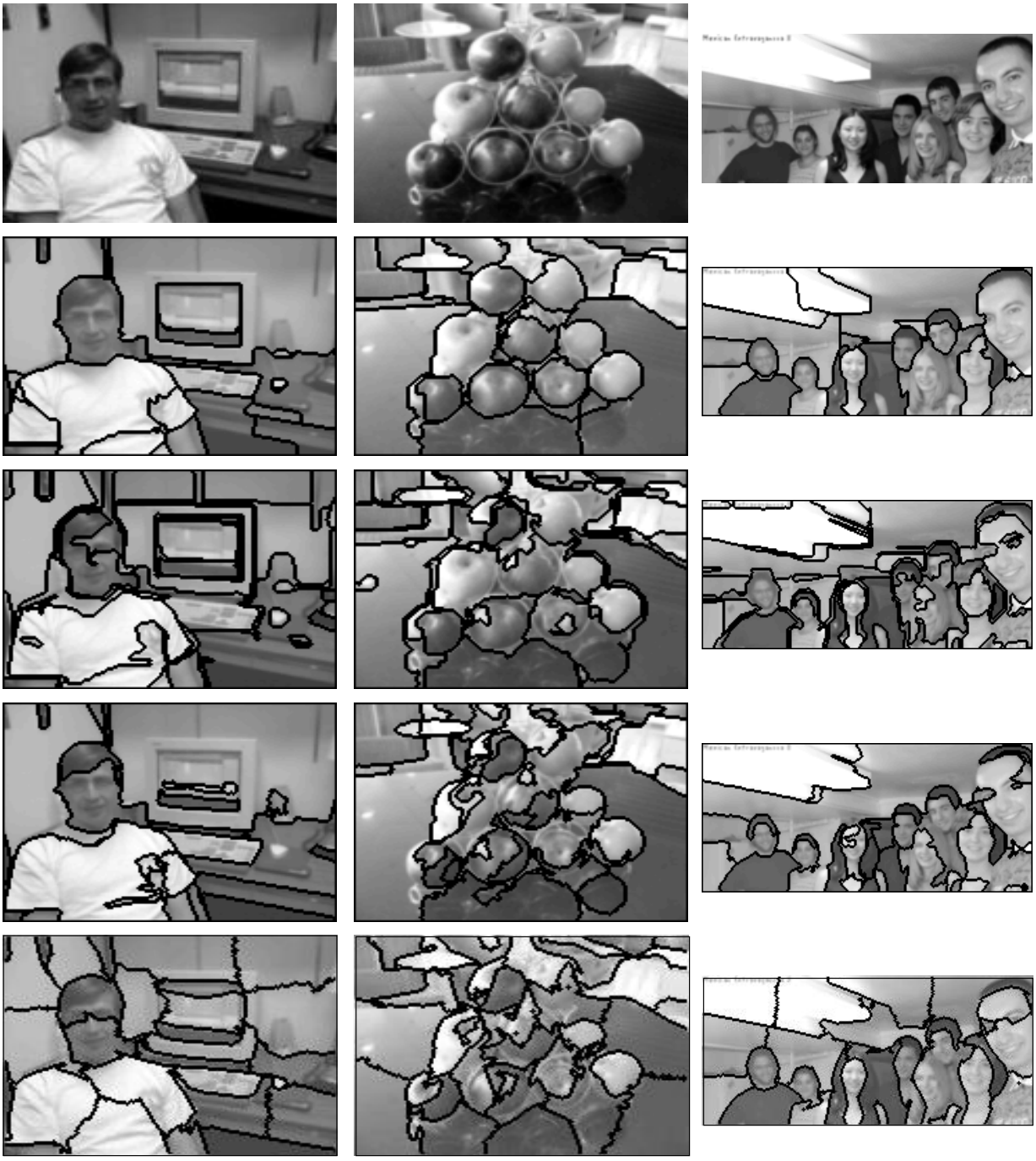

Figure 5.1: Segmentation results for indoor scenes, $d = 15$ and t is set as described in the text. From top to bottom: Original images, segmentations produced by our algorithm, segmentations generated by the Local Variation algorithm, Mean Shift segmentations, and segmentations from Normalized Cuts. From left to right, the images are 160×120, 160×120, and 239×108 pixels in size.

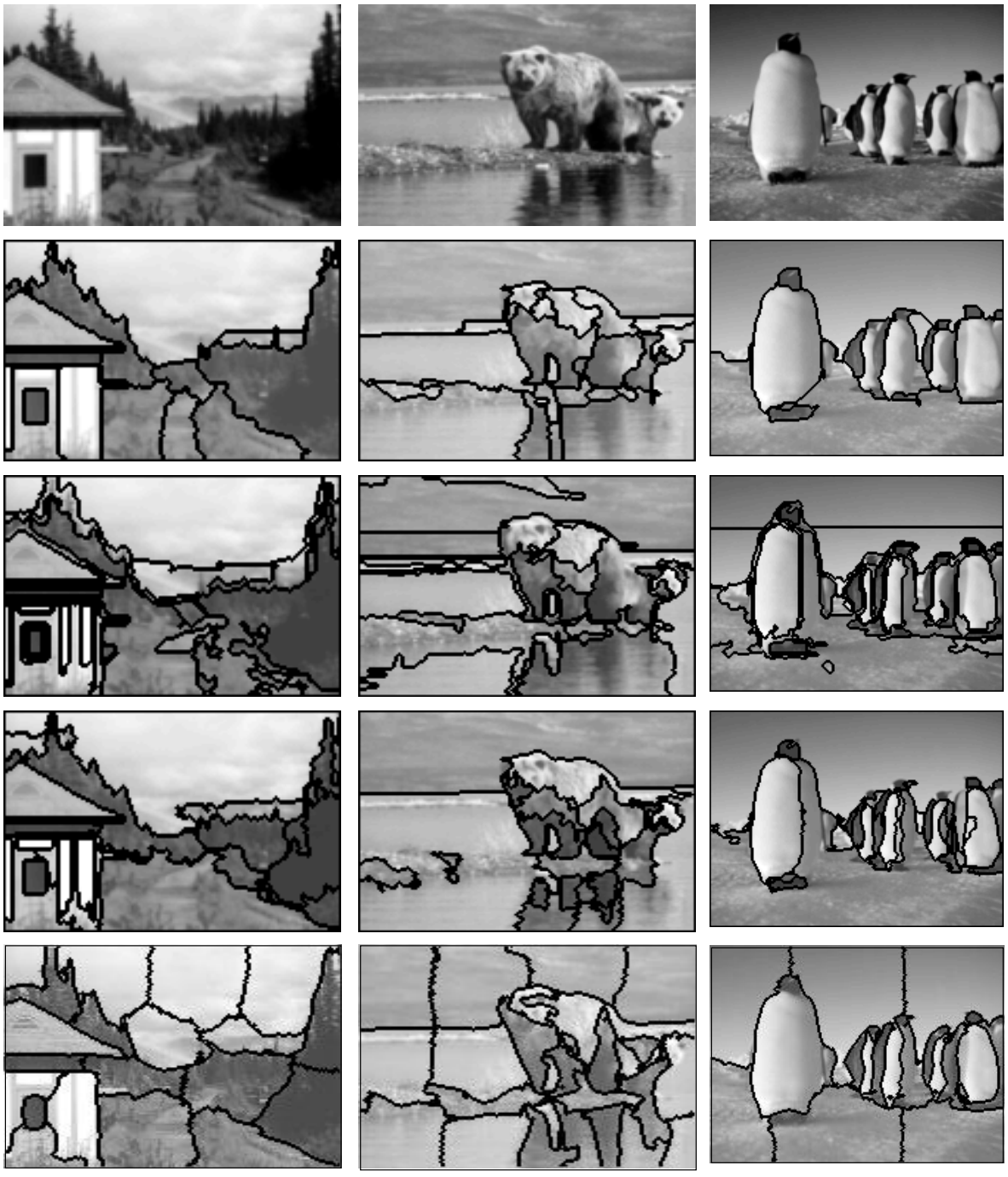

Figure 5.2: Segmentation results for outdoor scenes, $d = 15$ and t is set as described in the text. From top to bottom: Original images, segmentations produced by our algorithm, segmentations generated by the Local Variation algorithm, Mean Shift segmentations, and segmentations from Normalized Cuts. From left to right, the images are 160×120, 160×120, and 200×150 pixels in size.

contours, and the affinities themselves are calculated across a larger image neighborhood. The Local Variation algorithm uses accumulated statistics within each region to perform the segmentation. We expect that affinity measures that incorporate other image cues such as colour and texture would only enhance the results obtained with our segmentation algorithm.

Though in the results discussed above we have set the value of d so as to obtain a large enough number of seed regions to cover all salient natural image segments, particular applications may benefit from coarser or finer segmentations. The amount of detail in the final segmentation can be influenced by changing the value of d appropriately, larger values of d yield a larger number seed regions, and the seed regions themselves become smaller. Smaller values of d yield fewer and larger seed regions. Figure 5.3 shows three of our test images, and the segmentations obtained for different values of d, with t calculated as described in the text.

For the images in Figs. 5.1, and 5.2, and others of similar sizes, our algorithm running on a 2.5GHz. P4 machine requires roughly 30 secs. to compute the 40 leading principal vectors of L (which is roughly a $20K \times 20K$ matrix!), seed region generation takes between 30 sec. and 1 min., the computation of minimum cuts between source and sink regions takes from 1 to 3 min., and the final merging stage consumes between 1 and 2 minutes. The variability in run times is the result of having a different number of seed region proposals for each image.

The above results demonstrate that our procedure for selecting candidate seed regions is effective, and that a segmentation algorithm based on S-T min-cut, using our seed regions produces good quality segmentations of natural images. Visual inspection of the results shows that our algorithm produces segmentations whose boundaries more closely agree with those of natural image segments. To validate these results, a quantitative evaluation of our algorithm against Normalized Cuts, Mean Shift, and the Local Variation algorithm is presented in the next section.

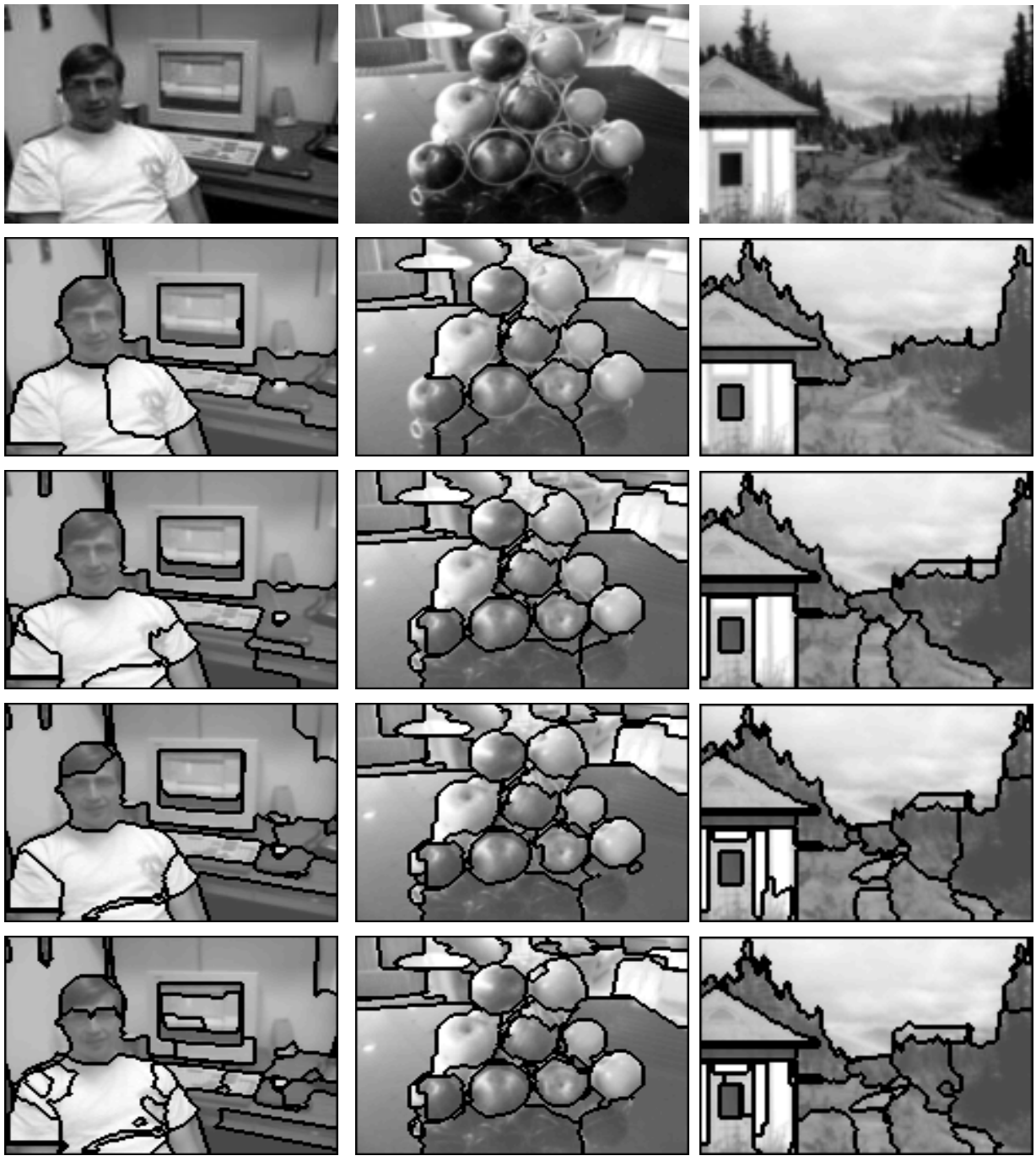

Figure 5.3: Segmentation results for different values of d. First row: original images. Second row: $d = 5$. Third row: $d = 15$. Fourth row: $d = 25$. Fifth row: $d = 35$. The value of t was calculated in each case as described in the text.

5.2 Quantitative Evaluation on the Berkeley Segmentation Database

The images in the previous section 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, there has been recent progress in developing quantitative measures of segmentation quality that can be used to evaluate and compare different segmentation algorithms.

In this section, we will evaluate the performance of our algorithm on the Berkeley Segmentation Database (BSD) [65]. We will discuss the characteristics of the error metrics previously defined by Martin et al. [66] explore potential problems with these metrics, introduce precision/recall curves that use the BSD to evaluate the quality of a segmentation, and show that these curves can be used to tune the parameters of a segmentation algorithm, and to characterize its performance over a range of parameter values. Finally, we will present quantitative performance results for our segmentation algorithm and three others: Normalized Cuts [98], Mean-Shift [21], and Local Variation [29].

5.2.1 The Berkeley Segmentation Database

The current public version of the Berkeley Segmentation Database [65] is composed 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 containing the remaining 100 images on which the final performance evaluations should be carried out.

For each image, and separately for the colour and grayscale versions of that image, a set of human segmentations is provided. This set contains between 4 and 8 segmentations specified as labeled images in a special format. Martin et al. [66] show that the human segmentations,

Figure 5.4: Selected images from the training image set, and four human segmentations for each image. Notice that the level of detail varies, but the segmentations are consistent with each other. Notice as well that there are small variations in the shape of region boundaries.

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 (see Fig. 5.4).

Based on this consistency, Martin et al. propose two metrics that can be used to evaluate the consistency of a pair of segmentations. The measures are designed to be tolerant to refinement, that is, if subsets of regions in one segmentation consistently merge into some region in the other segmentation the consistency error should be low. In order to compute the consistency

error for a pair of images, they first define a measure of the error at each pixel p_i

$$E(S_1, S_2, p_i) = \frac{|R(S_1, p_i) \backslash R(S_2, p_i)|}{|R(S_1, p_i)|}, \tag{5.1}$$

where $R(S_j, p_i)$ is the region in segmentation j that contains pixel p_i, $\backslash$ denotes set difference, and $|\cdot|$ denotes set cardinality. This measure evaluates to 0 if all the pixels in S_1 are also contained in S_2 thus achieving the tolerance to refinement discussed above. It is important to note that this measure is not symmetric, so for every pixel it must be computed twice, once in each direction.

Given the error measures at each pixel, two segmentation error measures are defined

$$GCE(S_1, S_2) = \frac{1}{n} min(\sum_i E(S_1, S_2, p_i), \sum_i E(S_2, S_1, p_i)), \tag{5.2}$$

and

$$LCE(S_1, S_2) = \frac{1}{n} \sum_i min(E(S_1, S_2, p_i), E(S_2, S_1, p_i)). \tag{5.3}$$

The Global Consistency Error (GCE) assumes that one of the segmentations must be a refinement of the other, and forces all local refinements to be in the same direction. The Local Consistency Error (LCE) allows for refinements to occur in either direction at different locations in the segmentation.

Martin et al. show that, as expected, when pairs of human segmentations of the same image are compared, both the GCE and the LCE are low; conversely, when random pairs of human segmentations are compared, the resulting GCE and LCE are high. They use histograms of GCE and LCE computed over the complete testing image set, both for pairs of human segmentations for the same image, and for random pairs of human segmentations, as a baseline to evaluate the quality of the segmentations produced by the Normalized Cuts algorithm.

5.2.2 Properties of the GCE and LCE Metrics

To better understand how the GCE and LCE error metrics work, it is interesting to consider what the metrics report on two extreme cases: A completely under-segmented image, where

every pixel has the same label (i.e. the segmentation contains only one region spanning the whole image), and a completely over-segmented image in which every pixel has a different label.

From the definitions of the GCE and LCE we can see that both measures evaluate to 0 on both of these extreme situations regardless of what segmentation they are being compared to. The reason for this can be found in the tolerance of these measures to refinement. Any segmentations is a refinement of the completely under-segmented image, while the completely over-segmented image is a refinement of any other segmentation.

In his Ph.D. dissertation [64], Martin notes these problems, and proposes alternative ways to evaluate pairs of segmentations, three of the proposed metrics are region based, and of these, one is a variation of the LCE metric (5.3) discussed above, namely, Martin defines the Bidirectional Consistency Error as

$$BCE(S_1, S_2) = \frac{1}{n} \sum_i \max(E(S_1, S_2, p_i), E(S_2, S_1, p_i)) \tag{5.4}$$

which is no longer tolerant to refinement. The two remaining region based metrics are based on mutual information, one of them works on the affinity matrix itself, and the other compares the pairwise assignment of pixels to regions in the two segmentations.

5.3 Precision/Recall Curves

Martin also proposes the use of precision/recall curves based on the region boundaries as a means to evaluate segmentation consistency. Recall is defined as the proportion of boundary pixels in the ground truth that were successfully detected by the automatic segmentation, precision is the proportion of boundary pixels in the automatic segmentation that correspond to boundary pixels in the ground truth.

Precision and recall are attractive as measures of segmentation quality because they are sensitive to over and under-segmentation, over-segmentation leads to low precision scores, while under-segmentation leads to low recall scores. A comparison of two segmentations can

only yield high values of both precision and recall, if the boundaries in both segmentations agree in location and level of detail. Additionally, if the region boundaries are in agreement, the regions themselves must agree across the segmentations.

In [64], precision and recall are estimated by computing a minimum cost matching between the boundary pixels in two segmentations. The cost of matching two pixels b_i and b_x is proportional to their similarity in spatial location and orientation

$$w_{b_i \to b_x} = \sqrt{(\Delta x)^2 + (\Delta y)^2} + \alpha \frac{|\Delta \theta|}{\pi/2}. \tag{5.5}$$

The matching cost for each pixel is computed using a bi-partite matching algorithm. Since the number of boundary elements in the two segmentations may be different, each segmentation is augmented with a number of outlier nodes in such proportion that the total number of nodes is the same for both segmentations. The cost of matching a boundary element in either segmentation to an outlier node is set to be high, so the matching procedure will only pair a boundary pixel with an outlier node if there is no suitable match within the boundary pixels of the other segmentation. After the bi-partite matching is computed, precision and recall are calculated as the number of pixels in the corresponding segmentation that were matched to outlier nodes.

Here we propose a different matching strategy. Given two segmentations S_1 and S_2, we find a suitable match for each boundary pixel in S_1 by examining its neighborhood within a radius of ϵ for boundary pixels in S_2. We match a pixel b_i in S_1 to a pixel b_x in S_2 if

- There is no other boundary pixel b_j in S_1 between b_i and b_x, with the exception of b_i's immediate neighbor (this is because, as will be described below, we know boundaries are 2-pixel wide).

- The nearest boundary pixel in S_1 for b_x is in the general direction of b_i (though it does not have to be b_i). If b_x has several nearest neighbors, at least one of them must point in the general direction of b_i (in practice, this means that the directions from b_x to b_i, and from b_x to one of its nearest neighbors should be within 25 degrees of each other).

If more than one pixel in S_2 satisfies the above conditions for a given b_i, we choose the nearest one. Together, these conditions imply that b_i should not be matched to a boundary pixel b_x unless b_i is part of the closest boundary in S_1 to b_x's boundary in S_2. The direction condition is necessary to avoid double matches when a boundary in S_2 is flanked on both sides by boundaries in S_1. We can define

$$Precision(S_1, S_2) = \frac{|unmatched(S_1)|}{|S_1|} \tag{5.6}$$

where $unmatched(S_1)$ is the set of boundary elements in S_1 that do not have a suitable match in S_2 within a distance ϵ, and $|\cdot|$ represents set cardinality. Similarly, we can define

$$Recall(S_2, S_1) = \frac{|unmatched(S_2)|}{|S_2|}. \tag{5.7}$$

Given the matching algorithm describe above, we can estimate precision and recall for any pair of segmentations. We evaluate the performance of an algorithm by comparing its output segmentations against the human segmentations, and computing precision/recall statistics over the complete set of images from the BSD. However, instead of comparing the algorithm's output against several different human segmentations of the same image, we compare against the union of the boundaries from all human segmentations of that same image. This was already suggested in [64] as a means of accounting for the variability in the level of detail between human segmentations.

The composite segmentation contains any boundaries that human observers considered salient, and thus provides us with a more useful benchmark against which we can test for over and under-segmentation. Additionally, we consider the fact that even though different observers will segment an image at different levels of detail, some of the boundaries in the image are agreed upon by most (and in fact, usually by all) human observers. Boundaries in the composite segmentation are weighted according to how many observers marked the same boundary pixel, normalized by the number of observers. This yields a composite image with values in $[0, 1]$, where 0 represents no boundary, and 1 represents a boundary pixel that was marked by all the human observers segmenting the image. This is illustrated in Figure 5.5.

We extend the precision and recall metrics defined above to make use of the weighted boundaries. The reason for this is that the quality of the segmentation should depend more heavily on boundaries that were considered salient by more of the human observers

$$Precision(S_1, S_2) = \frac{\sum_{b_i \in unmatched(S_1)} w_{b_i}}{\sum_{b_i \notin unmatched(S_1)} w_{b_i} w_{b_{x,i}} + \sum_{b_i \in unmatched(S_1)} w_{b_i}} \tag{5.8}$$

where w_{b_i} is the boundary weight for b_i from S_1, and $w_{b_{x,i}}$ is the boundary weight for the pixel in S_2 that was matched to b_i. In a similar way (simply changing the roles of S_1 and S_2), we define

$$Recall(S_2, S_1) = \frac{\sum_{b_i \in unmatched(S_2)} w_{b_i}}{\sum_{b_i \notin unmatched(S_2)} w_{b_i} w_{b_{x,i}} + \sum_{b_i \in unmatched(S_2)} w_{b_i}}. \tag{5.9}$$

Notice that the boundary weight for automatic segmentation boundaries is set to 1, but the measures allow for the general case in which both segmentations are weighted.

We will use the weighted precision/recall metrics to evaluate the segmentation algorithms. Given the above definitions, precision will be low when there is significant over-segmentation, when significant portions of the boundary have localization errors greater than ϵ, or when a significant portion of the boundary pixels in S_1 are matched to pixels in S_2 with low boundary weight (meaning that they capture structure that was not considered salient by many human observers). A low recall value is typically the result of under-segmentation and the failure to capture salient image structure, especially if the unmatched boundaries have high boundary weight in S_2.

Precision and recall are attractive for one additional reason. Precision/recall curves can be used to characterize the performance of an algorithm over a range of input parameters (thus allowing for the selection of the optimal parameters for any desired recall value), and to compare different algorithms on an equal footing. We choose to use these curves to evaluate the performance of the segmentation algorithms on the Berkeley Segmentation Database.

Figure 5.5 shows an image from the BSD, the corresponding composite segmentation, and two segmentations showing severe over, and under-segmentation. The figure also shows the precision and recall for each case. The results are as expected, the over-segmented image

has high recall (meaning it detected most of the boundaries in the composite human segmentation), but low precision. The under-segmented image receives low recall since it fails to capture salient image structure, and also low precision (since a relatively large proportion of the boundary pixels in the automatic segmentation remain unmatched). An interesting observation from the results on the under-segmented image is that it is possible for algorithms to under-segment globally while over-segmenting locally.

5.4 Quantitative Segmentation Results

In this section we present the results of evaluating each segmentation algorithm on the images from the BSD. We use the weighted precision and recall metrics developed in the previous section, and show results for two values of the distance parameter ϵ used during boundary matching.

5.4.1 Experimental Set-Up

Neither our algorithm nor the Normalized Cuts implementation from [23] can work directly on the images from the BSD due to their size. For this reason, we have chosen to downsample the complete training and testing sets by a factor of 4. The resulting images have a size of 121×81 pixels. Furthermore, since the affinity function used by our algorithm is based on gray-level difference, we will evaluate the four segmentation algorithms using the grayscale version of the downsampled images.

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 upsampled to the appropriate size, or the human segmentations could be downsampled. For simplicity, and to reduce the computational load of evaluating the error metrics, we chose this latter option. Thus,

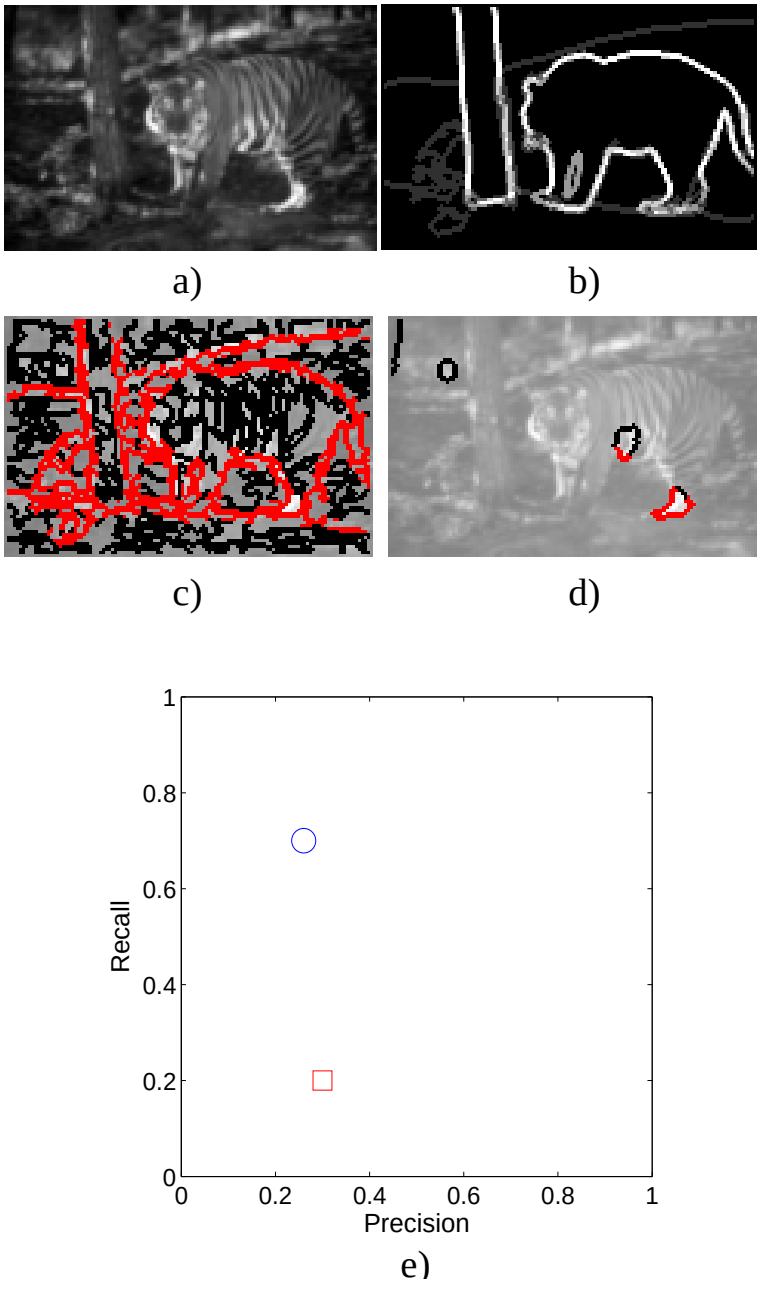

Figure 5.5: a) Downsampled original image (121 $\times$ 81 pixels), b) Composite human segmentation, higher brightness corresponds to larger boundary weight. c) Over-segmented image. d) Under-segmented image (in both c) and d) matched pixels are shown in red). e) Precision and recall for the over-segmented (blue) and under-segmented (red) images.

for error computation purposes, the human segmentations from the BSD, which have the form of labeled images, are also downsampled by a factor of 4. Figure 5.6 shows the downsampled version of the images and segmentations shown in Figure 5.4 above.

5.4.2 Segmentation Algorithms and Algorithm Parameters

We tested each algorithm across a range of its input parameters, for each combination of parameters the algorithms are run over the complete set of 300 images from the BSD (since there is no training phase, we do not need separate training and testing image sets). Given the precision and recall values for individual images, we compute and store the median precision and recall achieved by the algorithm over the BSD for each combination of parameters.

The resulting median values of precision and recall describe tuning curves that characterize the performance of the algorithm over some range of input parameters. For algorithms with a single input parameter (Normalized Cuts and Local Variation), we obtain a single curve, for algorithms with two parameters (SE-MinCut, and Mean-Shift) we get a curve for each possible value of one input parameter, while the values along the curve correspond to variations of the second parameter.

We used the implementations of the algorithms made available by the authors, for Normalized Cuts see [23], for Local Variation see [28], and for Mean-Shift see [37]. The segmentations produced by the four algorithms consist of labeled images in which pixels within the same region have an identical label. Since our metrics require region boundaries instead of region label maps, some processing is required. Normalized Cuts and Mean-Shift can be made to output boundary images, however, to ensure fairness in the evaluation process we extracted the boundaries using an identical procedure for the four algorithms. The procedure consists simply of taking the labeled image output by the algorithm, and marking each pixel that has at least one neighbor with a different label as a boundary pixel (this causes a 2-pixel wide boundary to be marked along region borders, with possibly wider boundaries at junctions of more than two regions). The ranges for the input parameters of each algorithm were determined experimen-

Figure 5.6: Downsampled images from the training set, and 4 downsampled human segmentations for each image, the boundaries look thicker because their 2-pixel width is now larger relative to the image size. Notice that the downsampled boundaries still capture the structure of the full-resolution segmentations (see Fig 5.4).

tally 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 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 [29]), 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 [21], and [22] for details). In practice, the segmentation software for Mean-Shift takes an additional parameter (the size in pixels of the smallest allowed region). We didn't test the effect of this parameter and instead kept it fixed to 25 pixels in all our tests. Experimentation showed that the largest differences between segmentations were obtained when 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 $[1, 20]$.

For Spectral Embedding Min-Cut, we have two parameters: d which determines the number of eigenvectors to use in the embedding, and the merging threshold τ_m. The largest variation between segmentations is obtained by changing the value of d, so, 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 compute a tuning curve that corresponds to variations of d within $[5, 40]$.

Finally, we also calculated precision and recall for human segmentations of the same image. Given a set of human segmentations of some image, we select each human segmentation in turn, and compare it against a composite of the remaining segmentations for that particular image. We then average precision and recall for all observers over the image (this 'leave one

out' strategy has already been suggested by Martin [64] as a means of evaluating human segmentations). The resulting precision and recall point are useful for comparing the performance of segmentation algorithms against human observers.

Since SE-MinCut and Mean-Shift have several tuning curves, we choose for comparison the curve that offers the best balance of precision/recall. Figure 5.7 shows the tuning curves for SE-MinCut, and Mean-Shift. The curves are shown for $\epsilon = 3$ and $\epsilon = 1$. From these curves we choose the one corresponding to $\tau_m = .25$ as the representative curve for SE-MinCut, and the curve corresponding to a spatial bandwidth $SB = 8$ for Mean-Shift.

Figures 5.8 and 5.9 show the tuning curves for the Normalized Cuts and Local Variation algorithms, as well as the selected curve for Mean-Shift and SE-MinCut. They also show the precision/recall points corresponding to human segmentations of each image (300 data points in total), for comparison. From these curves, appropriate parameter values can be selected that will yield a target value of either precision or recall for a given algorithm (limited, of course, by the operational range of the algorithm). Most importantly, they provide a direct way of comparing the quality of the segmentations produced by different algorithms across a wide range of input parameters, we can tell when an algorithm performs consistently better than others across its particular range of parameters, and rank algorithms by performance for particular values of precision or recall.

Figures 5.8 and 5.9 offer quantitative evidence that our algorithm outperforms the three competing segmentation methods across the range of (tested) input parameters. Particularly, we can observe that for higher recall values, SE-MinCut offers the best precision. This indicates that in general SE-MinCut captures salient image structure with less over-segmentation. This agrees with the segmentation results shown in Figures 5.1 and 5.2.

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 5.10 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

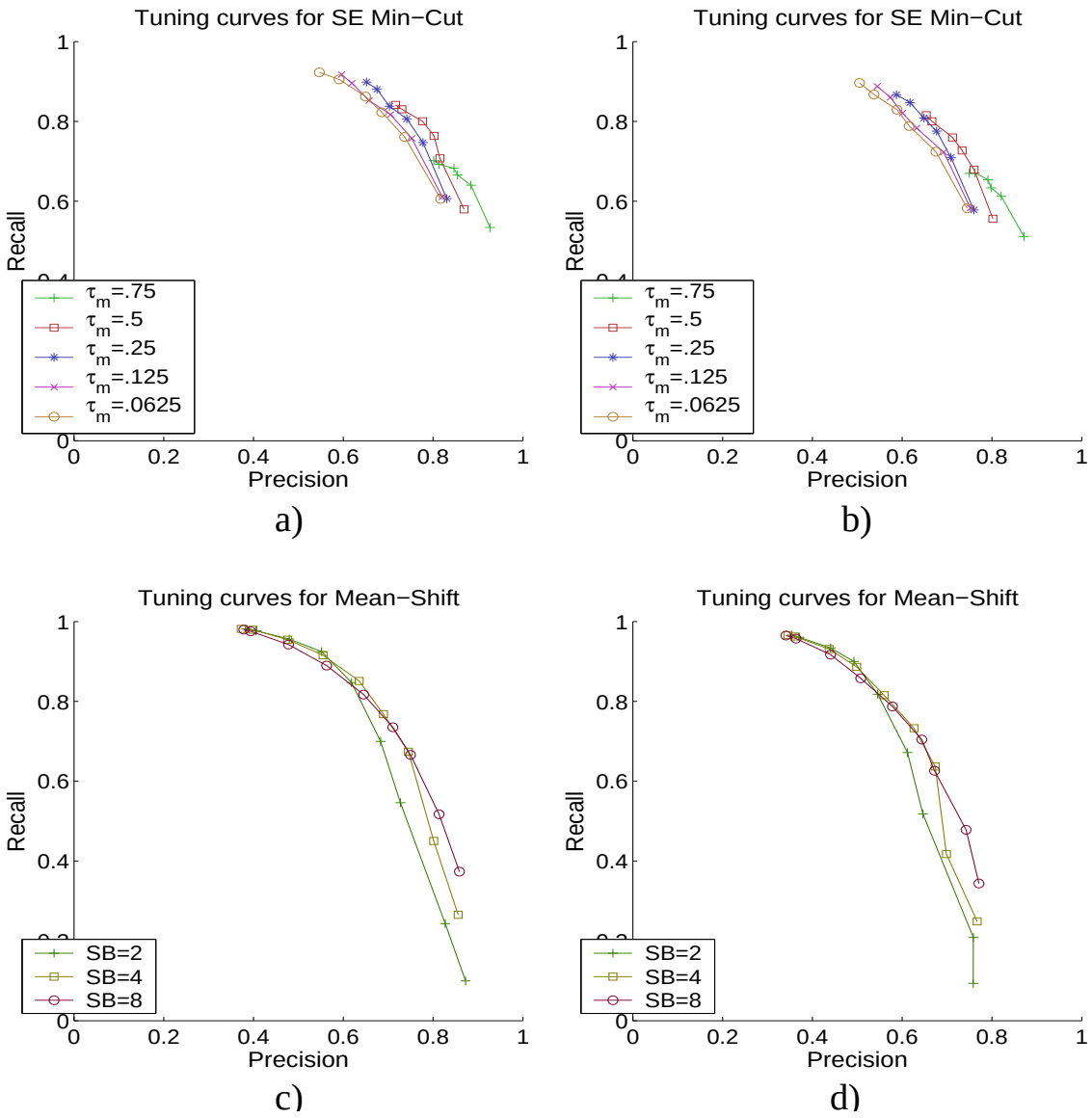

Figure 5.7: Top row: Tuning curves for SE-MinCut: a) $\epsilon = 3$, b) $\epsilon = 1$. The corresponding value for the merging parameter τ_m is indicated in the plot, points along each curve correspond to values of d in $[5, 40]$, with $d = 40$ corresponding to the highest recall. The best curve for SE-MinCut corresponds to $\tau_m = .25$. Bottom row: Tuning curves for Mean-Shift: c) $\epsilon = 3$, d) $\epsilon = 1$. The corresponding values for the spatial bandwidth parameter (SB) are shown in the plot, values along each curve correspond to variations of the range bandwidth parameter in $[1, 20]$, with $RB = 1$ yielding the highest recall. The best curve for Mean-Shift corresponds to $SB = 8$.

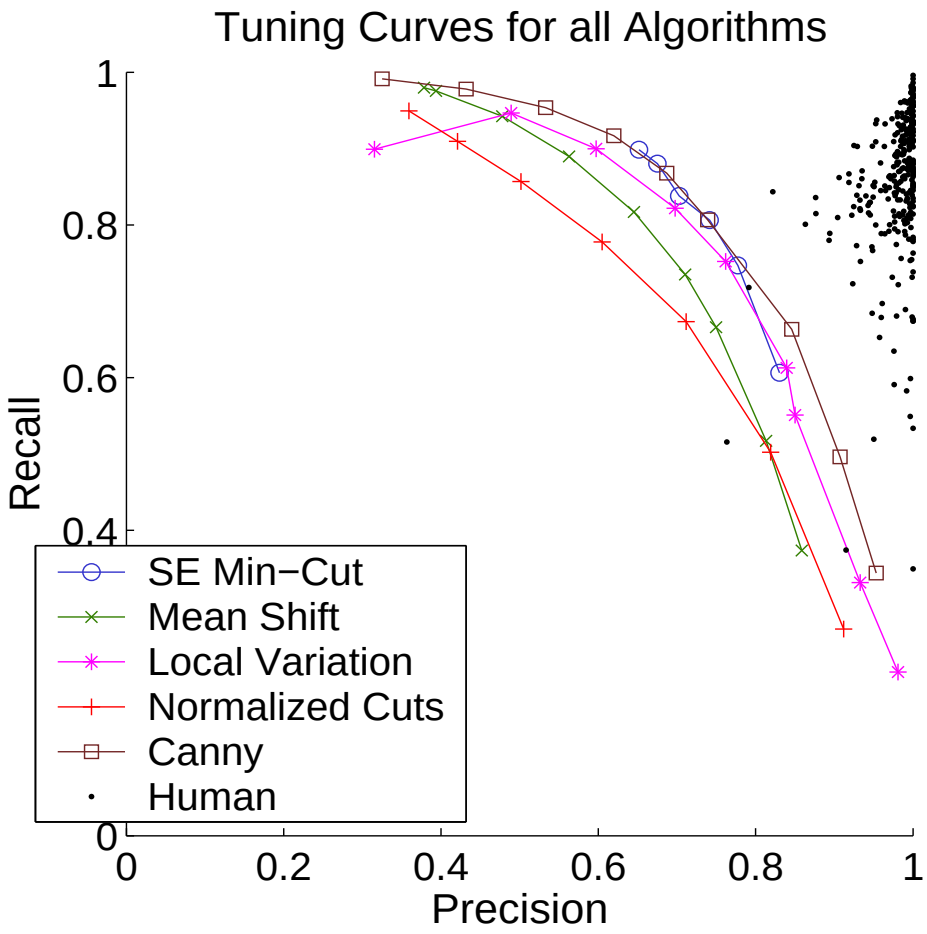

Figure 5.8: a) Tuning curves for all the segmentation algorithms algorithms plus Canny edgels, and precision/recall data for human segmentations. All curves were generated using $\epsilon = 3$ for boundary matching.

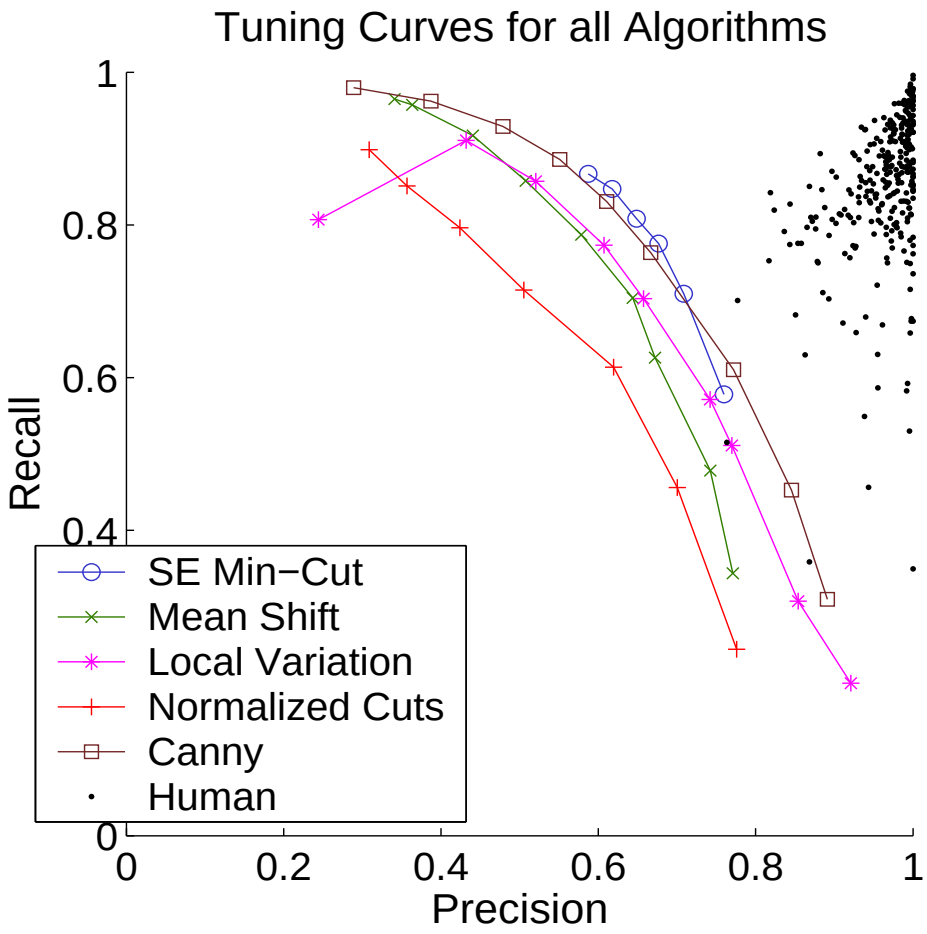

Figure 5.9: a) Tuning curves for all the segmentation algorithms plus Canny edgels, and precision/recall data for human segmentations. All curves were generated using $\epsilon = 1$ for boundary matching.

corresponding tuning curve. 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 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. While under-segmentation occurs for all algorithms, SE-MinCut and Normalized Cuts benefit from the global nature of the eigenvectors used during segmentation. Results in Fig. 5.10 provide additional evidence that SE-MinCut produces better segmentations along its tuning curve, and agree visually with the information provided by the precision/recall metrics. 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 in Figures 5.8, 5.9, and 5.10 results for the Canny edge detector (with no hysteresis thresholding). For Figs. 5.8, and 5.9 a tuning curve was produced by varying a single threshold on normalized gradient magnitude between $.05$ and $.6$. 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. 5.10), and do not give a segmentation of the image. To generate a segmentation, we would need to group Canny edgels into closed boundaries. This is a difficult problem, but the results presented here motivate research in this direction. We will discuss the grouping of edges in the perceptual grouping part of the thesis. For now, we show the Canny tuning curve to provide a hint of how well the segmentation algorithms perform as purely 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

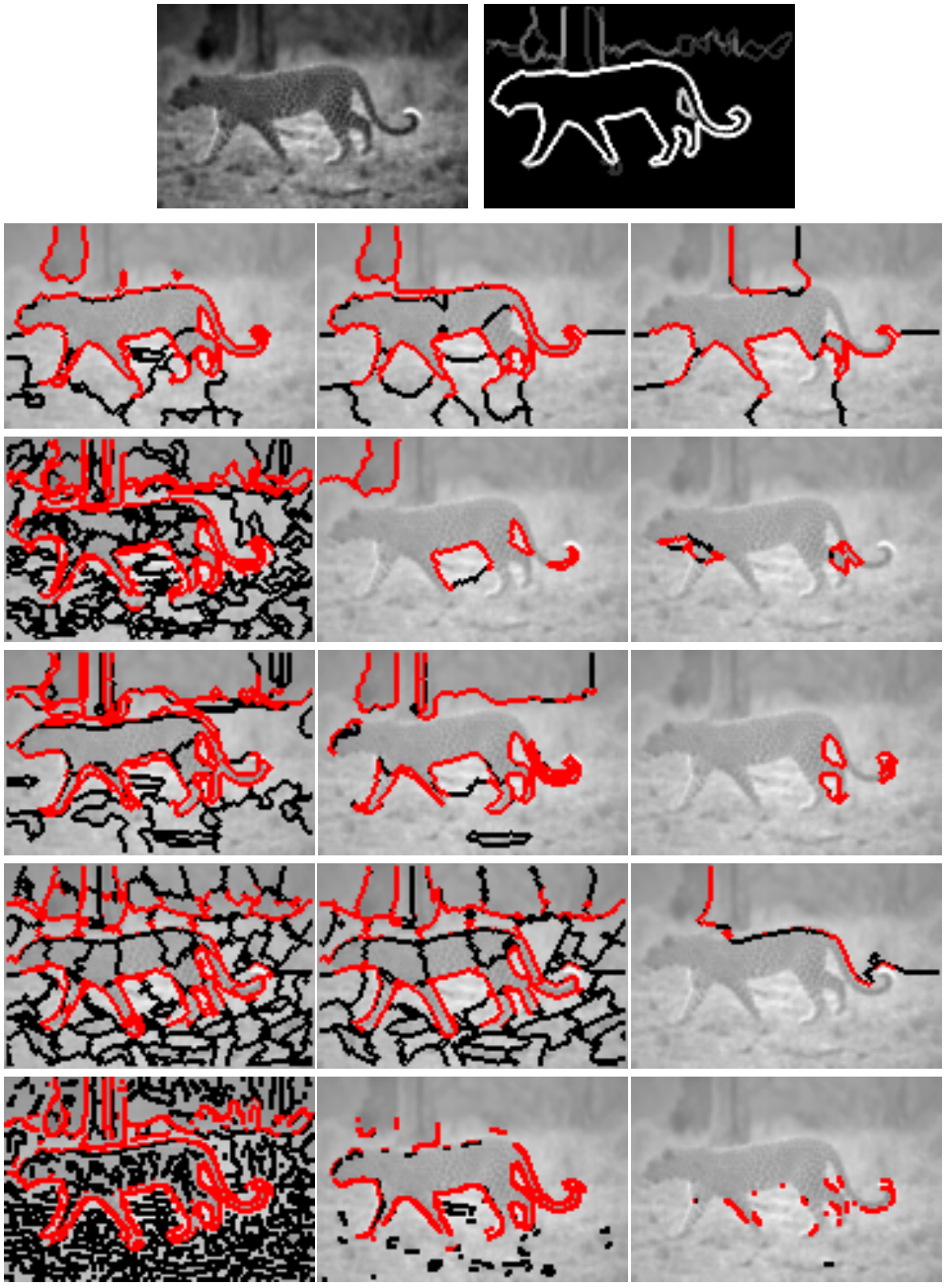

Figure 5.10: From top to bottom: Original image and composite human segmentation, SE-MinCut segmentations, Mean-Shift results, Local Variation, Normalized Cuts, and Canny edges. On each image, boundary pixels that were matched to the composite human segmentation are shown in red. The leftmost column corresponds to the parameters that yield highest recall, the middle column is for parameters at the middle of the tuning curve for each algorithm, and the rightmost column corresponds to the parameters that yield highest precision.

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. This is illustrated in Figure 5.11 which shows two images that received very high and very low recall values, along with the human segmentations of each image.

5.5 Quantitative Evaluation on Synthetic Data

We expect that part of the reason why there is such a large gap in segmentation quality between human observers and segmentation algorithms, is that human observers use high level knowledge and visual processing to perform segmentation. This allows human observers to accurately segment objects when the photometric information provided by the image is ambiguous, or contradictory (for example, by merging regions that have very different appearance, such as a zebra's stripes, into a single object, namely, the zebra). This raises the question of how well the segmentation algorithms perform on a segmentation task that is characterized exclusively by the perceptual difference between regions.

To gain some insight into this matter, we [2] tested the segmentation algorithms on a set of fractal images. Each image consists of a few regions with smooth boundaries and with uniform brightness, to which we have added fractal noise. Figure 5.12 shows four such images, as well as the ground truth and the segmentations produced by SE-MinCut. We have 107 fractal images in total, and ground truth corresponding to the true region boundaries for the noiseless versions of each image.

Even though the only factor that differentiates one region from another is their gray-level difference, segmenting the fractal images is not a trivial task due to the rather significant amount of noise added to the original images. Noise may introduce leaks across region boundaries, and its fractal nature may lead to small clusters of noisy pixels that actually look different

[2] The author would like to acknowledge Midori Hyndman's contribution in generating the fractal images, and in running the tests on the segmentation algorithms for this part of the thesis.

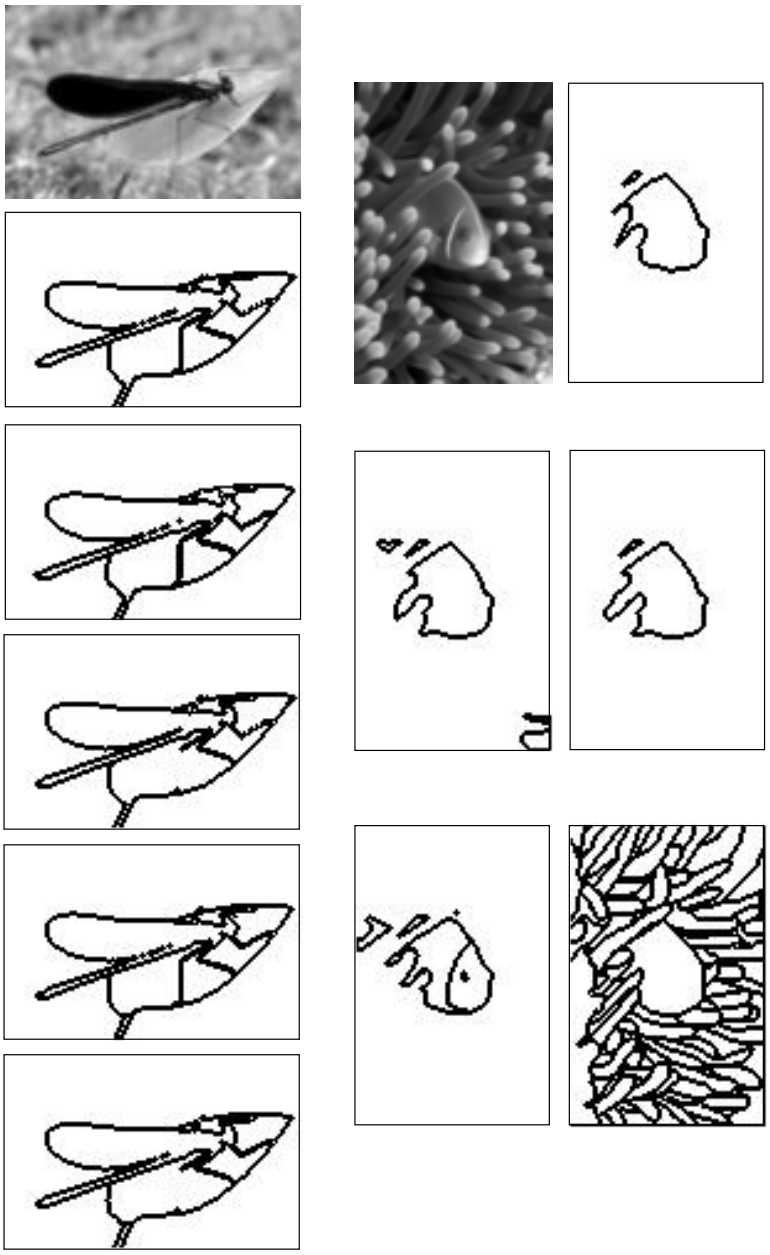

Figure 5.11: Images with high (left) and low (right) recall values for human segmentation, and corresponding human segmentations. Notice that the segmentations for the first image are quite similar, yielding a median recall of .99. For the second image, one observer segmented the scene at a much finer level of detail, when compared against a composite including this detailed segmentation, all other human segmentations receive a low recall value. The median recall for this image was .36.

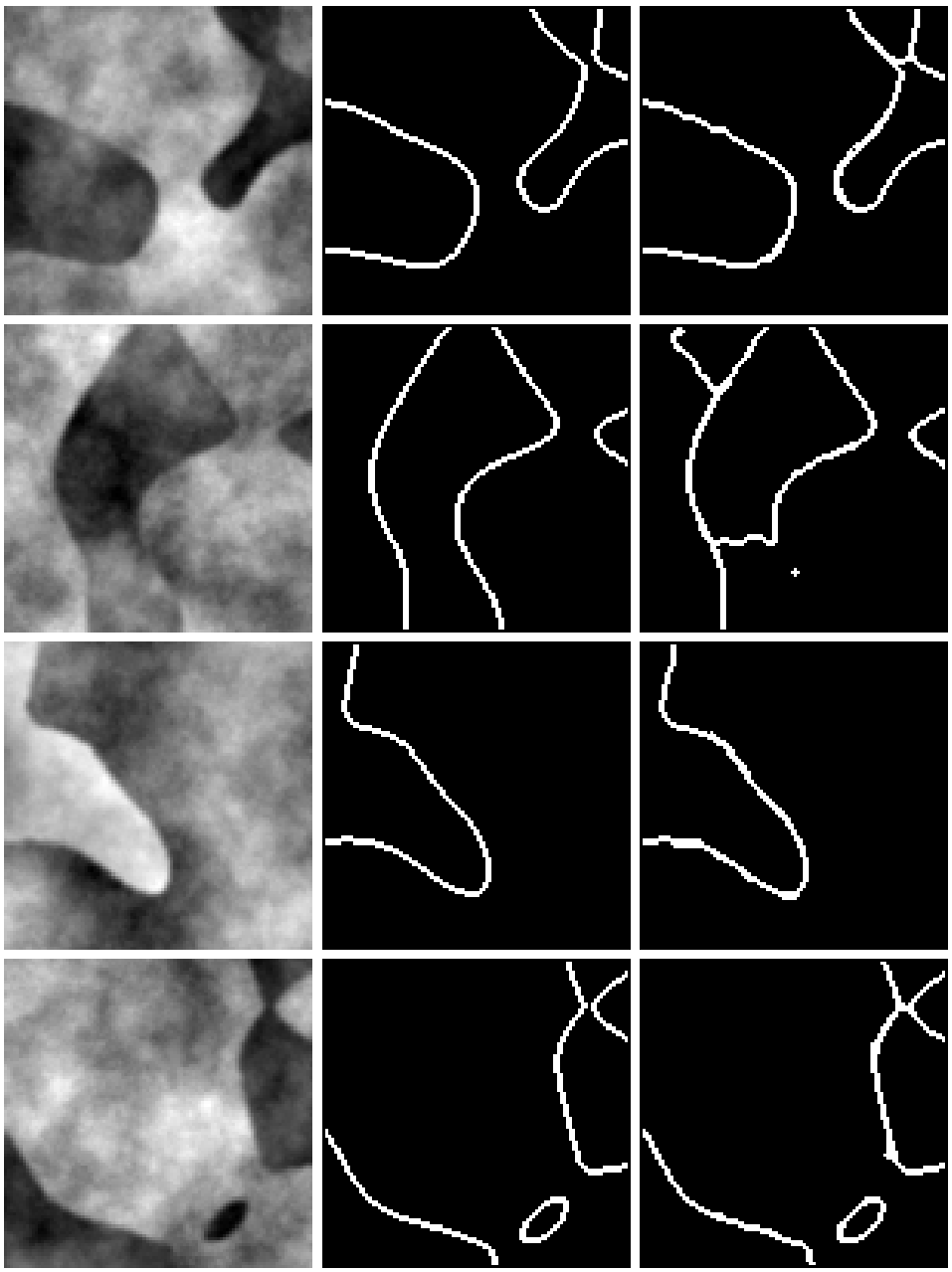

Figure 5.12: Left: Sample fractal images (100×100 pixels in size). Middle: Ground truth (original region boundaries). Right: SE-MinCut segmentations for $d = 15$, and $\tau_m = .5$.

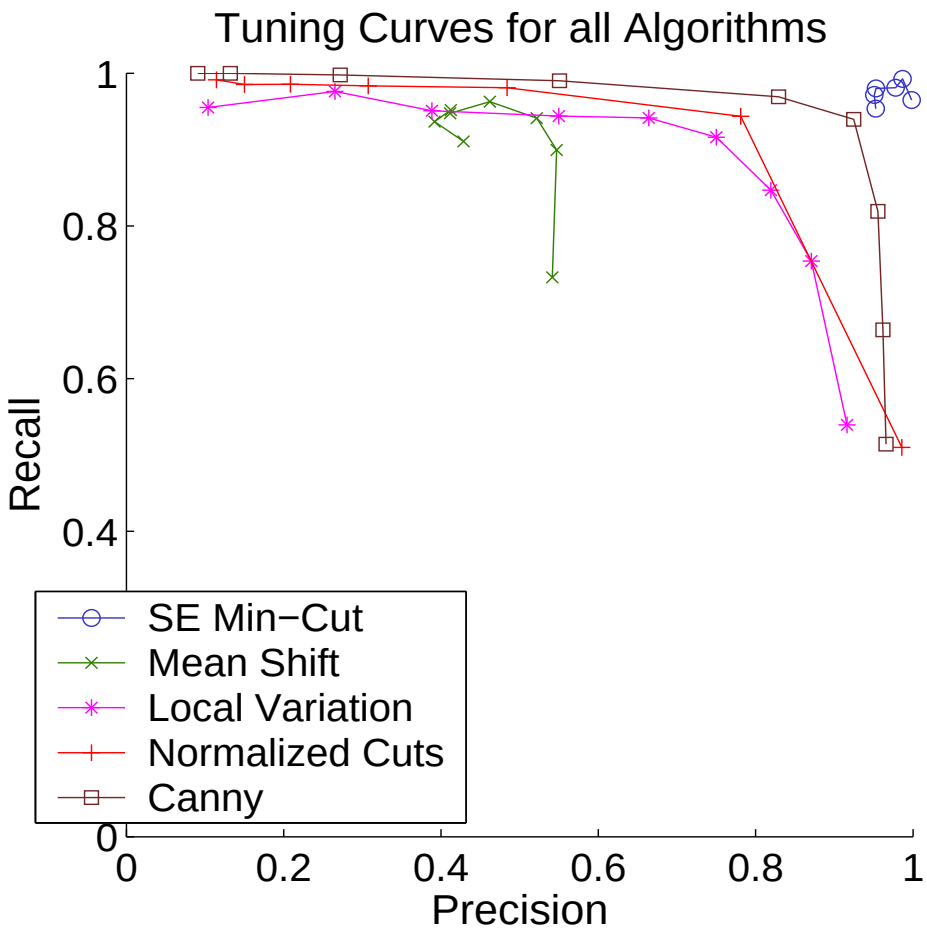

Figure 5.13: Tuning curves for all segmentation algorithms, plus Canny edges. The curves were generated with a radius of $\epsilon = 1$ for boundary matching. Only the best curves for SE-MinCut, and Mean-Shift are shown.

from the region in which they are located.

We ran the four segmentation algorithms, and the Canny edge detector on the fractal images, using the same parameter combinations described above, for the tests on the BSD. In the same manner, we calculated the median precision and recall for each run, and generated tuning curves for each algorithm. Figure 5.13 shows the results obtained for each algorithm on the set of fractal images. It is clear that SE-MinCut outperforms all other segmentation algorithms by a significant margin. In particular, the degree of over and under-segmentation incurred by our algorithm is quite small in comparison to competing segmentation methods. The good

results obtained by SE-MinCut can be explained in part by the anisotropic smoothing effect introduced by the blur kernels, which greatly reduces the influence of noise on the clustering algorithm, and segmentation process.

It is particularly interesting that the Normalized Cuts algorithm performs significantly better on the fractal images than on the BSD. This is likely to be because the eigenvectors used by Normalized Cuts provide global information about the image, and are less easily affected by noise. In addition, we should note that we could expect a higher precision/recall score for Normalized Cuts if we provided the algorithm with the correct number of regions for each fractal image (instead of running with a fixed number of segments on the complete database).

5.6 Discussion

A few observations should be made with regards to the precision/recall plots shown above. The tuning curves for SE Min-Cut were obtained using a simple measure of affinity based on gray-level difference between neighboring pixels. We expect that better affinity measures will enhance the quality of the segmentations produced by the algorithm on images such as those of the BSD.

We expect the tuning curves to be resolution dependent, the larger the image resolution, the more sensitive the curves will become to small variations in boundary localization (this will also affect the human precision/recall scores, since humans segmentations tend to show perceptible variations in localization of region boundaries). The precision and recall values will also be affected by over-segmentation due to additional detail and texture markings that are more salient at higher resolution.

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, 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 1.9GHz 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 improved for increased efficiency. Our goal here was to show that min-cut with automatic source and sink selection, and followed by a simple post-processing stage can produce higher quality segmentations than current algorithms.

This concludes the part of this thesis that deals with bottom-up image segmentation, we have made several contributions to the image segmentation field:

- We have presented an explicit connection between spectral embedding and anisotropic smoothing.
- We have shown that spectral embedding can be used to select clusters of similar pixels within homogeneous image regions, and described an algorithm to generate source and sink proposals for min-cut from these seed regions.
- We have developed a complete segmentation algorithm and shown segmentation results for our algorithm and several well known segmentation algorithms.
- We have presented a quantitative performance evaluation of our algorithm and others. Our results show that SE-MinCut consistently produces higher quality segmentations on the Berkeley Segmentation Database, and that it outperforms by a significant margin the competing segmentation methods on synthetic data.

Region segmentation algorithms are currently capable of separating an image into perceptually different regions using cues such as colour, grayscale intensity, texture, and some edge information. However, in most real world applications, the appearance of objects varies significantly over the object's surface. Changes in colour, texture, or illumination result in segmentations that split a single object into a number of image segments.

If these regions are to be grouped together into complete objects an additional stage of processing is required. In general, such a grouping stage must rely on cues that are not directly available from colour, intensity, or texture information. Instead, grouping should look for regularities in the data that suggest that particular parts of the image have a common origin (even if their appearance is quite different). In computational vision, this task is usually called perceptual grouping.

In the following chapters we will present our work on perceptual grouping starting with a comprehensive review of the relevant literature. We will then develop an efficient grouping algorithm in the context of convex polygon extraction and show that it yields significant computational advantages, finally, we will extend this algorithm to address the problem of general contour extraction, and show that the resulting grouping method remains efficient and can successfully extract complex contours in cluttered scenes.

The final part of this thesis will briefly examine the relationship between segmentation and perceptual grouping in terms of what classes of images are suitable for analysis with each methodology, and to what degree information provided by segmentation and grouping algorithms is complementary. It will also hint at how information from both domains could be integrated into a more comprehensive figure-ground separation algorithm.