Estimating Object Region from Local Contour Configuration

Tetsuaki Suzuki
NEC Corporation
Kawasaki, Kanagawa, Japan.
t-suzuki@du.jp.nec.com

Martial Hebert
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
hebert@cs.cmu.edu

Abstract

In this paper, we explore ways to combine boundary information and region segmentation to estimate regions corresponding to foreground objects. Boundary information is used to generate an object likelihood image which encodes the likelihood that each pixel belongs to a foreground object. This is done by combining evidence gathered from a large number of boundary fragments on training images by exploiting the relation between local boundary shape and relative location of the corresponding object region in the image. A region segmentation is used to generate a likely segmentation that is consistent with the boundary fragments out of a set of multiple segmentations. A mutual information criterion is used for selecting a segmentation from a set of multiple segmentations. Object likelihood and region segmentation are combined to yield the final proposed object region(s).

1. Introduction

Bottom-up segmentation of regions corresponding to individual objects in images without top-down information is a difficult task because the only available information is in the form of low level cues such as pixel intensity, color and edges. Some of the previous bottom-up approaches rely on segmentation based on the homogeneity of the regions [1][2][3][4]. After the segmentation process, detection methods estimate regions corresponding to objects by ranking the regions based on features such as depth [5], or compactness [6].

Figure/ground segmentation [7] approaches determine to which regions perceived occlusion boundaries belong. In the computer vision domain, computational approaches to figure/ground segmentation have been proposed using convexity [8], familiar configuration [9], and T-junction [9][10][5] to extract the important regions or to determine to which regions estimated boundaries are attached.

Recently, some object detection methods that exploit the shape of the occlusion boundaries have been proposed [11][12][13][14]. The traditional template based methods [15][16] use only the internal texture of the target objects and do not use the contour information explicitly, even though contour information is a natural cue for object detection. The boundary based methods show the usefulness of the occlusion boundaries to detect the objects while remaining robust to texture changes.

In this paper, we explore ways to use partial contour information, in the form of boundary fragments estimated from the image to segment out regions corresponding to objects in the scene. At the beginning of the whole process, we detect the occlusion boundary candidates by using the approach described in [17]. Second, we use the segmentation algorithm of [18], yielding multiple candidate segmentations that are consistent with the occlusion boundary candidates. After obtaining the multiple segmentations, we select the optimal segmentation among all the segmentations based on a normalized mutual information criterion. We estimate also the object regions in the image by first generating a region hypothesis for each candidate boundary fragment based on its shape. Then all of the hypotheses are combined into a single object region likelihood map which encodes the likelihood that each pixel belongs to an object region. Finally, we integrate the results of the segmentation and the figure region estimation to obtain the object regions in the images. We use the image database of [19] for most of the experiments.

2. Segmentation from candidate occluding contours

The first component of the algorithm is to generate image segmentations that are consistent with candidate contour fragments corresponding to occluding boundaries. To generate the initial contours, we use the occlusion boundary detector proposed in [17]. The boundary detector starts with an over-segmentation of the input image by applying watershed segmentation to the output of the *Pb* contour detector [20]. We call the over-segmented regions "segments" and we call "fragments" the boundary between two segments. The detector applies a classifier to the fragments to distinguish between occlusion boundary fragments and non-occlusion boundary fragments. The classifier is trained using Adaboost based on features such as edges, colors and morphological features. Once the

classifier is applied, the detector groups the fragments into consistent extended boundaries with global inference.

We modified the algorithm of [17] in several ways. We use additional features such as the chi-square distance between the texton histograms of the two regions abutting to each fragment for unary features, and the chi-square distance between the texton histograms of the figure segment and ground segment and the chi-square distance between the texton histograms of two figure regions for the pairwise features. We also use the global *Pb* detector proposed in [21] instead of the original *Pb* detector since global *Pb* detects better boundary candidates.

Table 1 shows the F-values [20] of global *Pb* and the improved occlusion boundary detector. The F-values of modified detector are 0.71.

Table 1 Evaluation of boundary detector.

	Global *Pb*[21]	Independent Labeling	After Global Inference
F-value	0.70	0.71	0.71

We use the contours to generate multiple segmentations by using Stein's segmentation method described in [18], which we briefly summarize now. This segmentation method is based on Normalized Cuts [3] but differs in the way the weights used in building the affinity matrix. Stein's method uses the fact that one of the segment connected to a boundary fragment must be in the background, while the other one must be in the foreground. For each fragment, these two regions are used as seeds to a matting process [22], which assigns to each pixel a α-value vector. The α value at a pixel characterizes the likelihood that the pixel is a foreground pixel. The weight associated with each of the graph edges is the correlation of the α-value vectors of the two nodes to which the edge is connected. From these weights, an affinity matrix is built and used in Normalized Cuts, which yields a segmentation of the image.

This object segmentation method requires the target number of regions to be specified in advance. In this paper, we use a method based on normalized mutual information [23] to find the "best" segmentation among the multiple segmentations obtained with different values of the number of regions. Normalized mutual information relies on the assumption that a good segmentation should share as much information as possible with all of the remaining segmentations. In this approach, normalized mutual information becomes the metric used to evaluate how well the image is segmented. More precisely, if S_a and S_b are two segmentation candidates, the normalized mutual information $\phi^{(\mathrm{NMI})}$ between two segments is computed as:

$$\phi^{(NMI)}(S_a, S_b) = \frac{\sum_{h=1}^{|S_a|}\sum_{l=1}^{|S_b|}|R_{h,l}|\log\frac{n\cdot|R_{h,l}|}{|R_h||R_l|}}{\sqrt{\sum_{h=1}^{|S_a|}|R_h|\log\frac{|R_h|}{n}\sum_{l=1}^{|S_b|}|R_l|\log\frac{|R_l|}{n}}}, \quad (1)$$

where $|R_h|$ and $|R_l|$ are the areas of region h and l, respectively, $|R_{h,l}|$ is the area in common between regions h and l, n is the number of pixels in the image, and $|S_a|$ and $|S_b|$ are the numbers of segments in segmentations S_a and S_b, respectively. Since a "good" segmentation shares more regions with the other segmentations, the segmentation S^o whose averaged normalized mutual information (ANMI) is the highest of all segmentations is selected as the final segmentation. The averaged mutual information is:

$$\phi^{(ANMI)}(\hat{\boldsymbol{S}}, \Lambda) = \frac{1}{N}\sum_{q=1}^{N}\phi^{(NMI)}(\hat{\boldsymbol{S}}, \boldsymbol{S}_q), \quad (2)$$

where $\Lambda = \{ S_1,.., S_N\}$ is the set of N input segmentations. An example of segmentation selected by using this criterion is shown in Fig. 1. In our experiments, the multiple segmentations are obtained by varying the number of regions between 2 and 20.

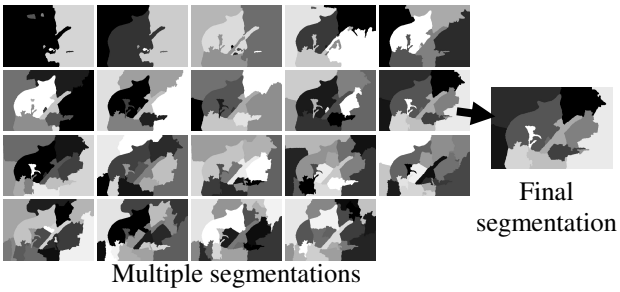


Fig. 1 The segmentation selected by the normalized mutual information criterion.

3. Object region likelihood estimation from boundary fragments

The next step is to generate an object likelihood image, which encodes the likelihood of object/background membership at each pixel, from the fragments. To do this, we exploit the fact that the shape of a contour fragment and the location of the corresponding object region are correlated. For instance, the contours of human heads are usually convex and the head object is located in the inside of the convex contours as shown in Fig. 2. If there are other relations between the shapes of contours and the location of the corresponding object regions, these relations can be strong cues to decide which segments are the object regions in images.

To estimate the object regions from the fragments found in the input images, we use an exemplar-based codebook which encodes the relations between the fragments and the corresponding object regions in the training images. In this section, we first explain how the codebook is generated. Then, we describe how the object region is estimated using the codebook. We use the figure/ground labeling data [9] to learn and evaluate the estimation of the object likelihood image.

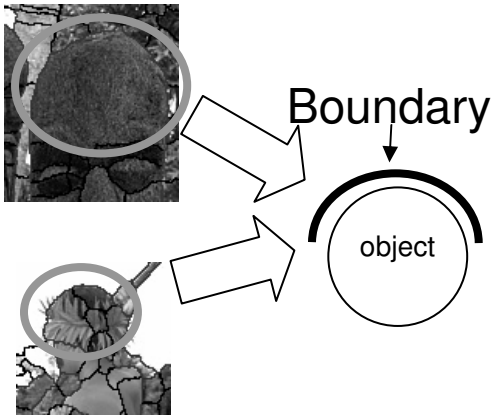


Fig. 2 Relation between boundary and object region.

3.1. Fragment codebook

The exemplar based codebook is built by collecting boundary fragments and their corresponding object regions from the training data. The boundary fragments are collected from the training images through the following procedure:

1. Over-segment the training data and obtain the fragments (Fig. 3 (b)).
2. Assign figure/ground labels in the ground truth data (Fig. 3 (c)) to the fragments by bipartite matching [9] (Fig. 3 (d)).
3. Discard the fragments which do not have enough consistent figure/ground labels connected to them (Fig. 3 (e)).
4. Normalize the center positions of all the fragments and their corresponding object regions.
5. Collect the pairs of the fragments and their corresponding object regions (Fig. 3 (f)).

We divide the large set of fragments into a smaller set of clusters. This is necessary because of the large number of pairs (fragments, regions) which prevents their direct use, *e.g.*, in a nearest-neighbor approach.

To cluster the fragments, we first construct a fully connected graph in which the nodes correspond to the fragments and the edges are weighted by t_e-$d(f_i,f_j)$, where t_e is the threshold and $d(f_i,f_j)$ *is* the distance between the two corresponding fragments to the nodes connecting the target edges. The distance between two fragments is computed by using the chamfer distance [25]. The distance between two fragments, f_i and f_j is:

$$d(f_i,f_j)=\frac{1}{N_i}\sum_{\mathbf{t}\in f_i}\min_{\mathbf{e}\in f_j}\|\mathbf{t}-\mathbf{e}\|^2+\frac{1}{N_j}\sum_{\mathbf{t}\in f_j}\min_{\mathbf{e}\in f_i}\|\mathbf{t}-\mathbf{e}\|^2, \qquad (3)$$

where N_i and N_j are the numbers of pixels of f_i and f_j, respectively.

To estimate the fragment clusters, we use the clique partitioning algorithm of [24]. This clique partitioning algorithm divides the graph into several disjointed sub-graphs so as to maximize the total weights of the remaining graph edges. The detailed algorithm is:

1. Construct the initial clique set C so that each clique c in C contains one node of the graph.
2. For each clique c in C, compute the best clique $a(c)$ with which to merge and the corresponding score $b(c)$:
$$\boldsymbol{a}(\boldsymbol{c})=\arg\max_{t\in C}\boldsymbol{m}(\boldsymbol{c},\boldsymbol{t}),\ \ \boldsymbol{b}(\boldsymbol{c})=\max_{t\in C}\boldsymbol{m}(\boldsymbol{c},\boldsymbol{t}), \qquad \textbf{(4)}$$
where $m(c,t)$ is the sum of the weights of all the edges between c and t:
$$\boldsymbol{m}(\boldsymbol{c},\boldsymbol{t})=\sum_{i\in c_1,j\in c_2}\boldsymbol{w}_{ij}\ . \qquad \textbf{(5)}$$
3. Merge cliques c_i, c_j if and only if $a(c_i)=c_j$ and $a(c_j)=c_i$ and $b(c_i)=b(c_j)>0$.

At step 2, the algorithm selects the pairs of cliques to merge next and based on the merging score of Eq. (4). At step 3, the algorithm merges two cliques if each one represents the best merging option for the other and if merging them increases the total score. Steps 2 and 3 are iterated until no clique can be merged.

Clique partitioning produces a set of sub-graphs whose total edge weight is maximized. The set of fragments corresponding to the nodes in each sub-graph is grouped into a fragment cluster and the distances between the fragments in the fragment cluster are minimized because the edge weights increase as the distances between two corresponding fragments decrease and are maximized by the clique partitioning. The initial fragment set F is divided into m subsets, $F_1,..,F_m$ by applying clique partitioning to the graph. The fragment whose total distance to all the other fragments in the same cluster is the shortest is selected as the prototype fragment for that cluster. The prototype fragment is the one used for computing the distance between a new input fragment and a cluster. In our experiments, we set the threshold t_e to 2.0.

3.2. Estimating the object likelihood map from a single fragment

We use the codebook described in the previous section to estimate which image pixels are likely to belong to the object region attached to each fragment. We start by associating to each fragment f from a training image a binary mask p_f, such that $p_f(x)$ is 1 if the pixel at position x in the image belongs to an object, 0 if it belongs to the background. To ensure invariance to translation of the object in the image plane, the position x is relative to the fragment. More precisely, x is the vector of pixel coordinates obtained by using the center of the fragment f as the origin of the coordinate system. We denote by f_j^o the prototype fragment associated with cluster j. For each cluster j, we estimate an object likelihood image such that if x is the position of a pixel relative to the prototype fragment f_j^o, the corresponding likelihood value is:

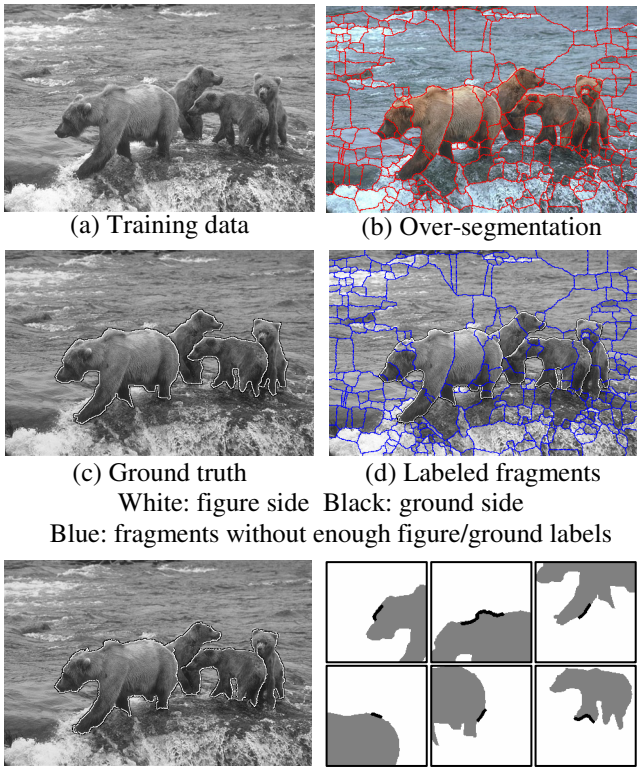


Fig. 3 The procedure for estimating the fragment labels (see text).

$$\boldsymbol{p}_j(\boldsymbol{x}) = \frac{1}{\boldsymbol{N}_{cj}} \sum_{f_l \in \boldsymbol{F}_j} \boldsymbol{p}_{f_l}(\boldsymbol{x}), \qquad (6)$$

where N_{cj} is the number of fragments belonging to cluster j. Informally, $p_j(x)$ measures the likelihood that a pixel at relative position x belongs to an object instead of the background, given a fragment from cluster j. Fig. 4 shows six examples of the prototype fragments and the object likelihood images of the corresponding clusters. In Fig. 4, the first and the third rows show the prototype fragment in black and all the fragments in the cluster in gray. The white diamonds are the center positions of the fragments. The second and the bottom rows show the object likelihood images. As the object likelihood increases, the pixel intensity becomes darker. The white diamonds are the center position of the corresponding fragments. The object likelihoods of pixels under the horizontal lines (cluster 1) are higher than those above it. This bias arises because the horizontal lines in the training data tend to be the part of the horizon lines and the figure regions of the horizon lines are assigned to the regions under the horizon lines, that is, the ground regions. The object likelihoods for the vertical lines (cluster 2) are unbiased because the object regions could be

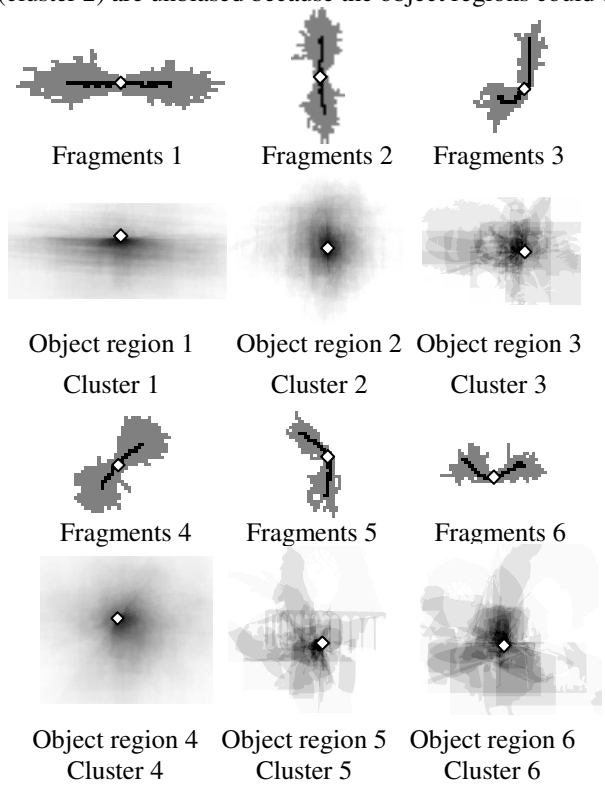


Fig. 4 Examples of obtained fragment clusters (see text).

on either side of the vertical lines. The other examples show the results of the convex fragments (cluster 3-6). The likelihoods of pixels inside the convex fragments are higher than those of the pixels on the outside. These results are consistent with our initial observations in Section 3.

Given an input image, we estimate a similar likelihood distribution for every fragment f from the input image. The likelihood image for a single fragment f is estimated by summing the likelihood images from all of the clusters in the codebook, with higher weights given to clusters closer to f. More precisely, if f is a fragment from the input image, the likelihood map relative to f at pixel i is:

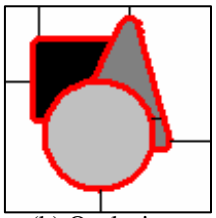
$$\boldsymbol{L}(\boldsymbol{i}, \boldsymbol{f}) = \frac{1}{\boldsymbol{Z}} \sum_j \boldsymbol{w}(\boldsymbol{f}, \boldsymbol{f}_j^o) \boldsymbol{p}_j(\boldsymbol{x}_i), \qquad (7)$$

where Z is the normalization factor: $Z = \Sigma_j\, w(f,f_j^o)$. As a slight abuse of notations, we denote by x_i the relative position of the pixel with respect to a fragment, even though we do not indicate explicitly which fragment since it is implicit in the form of the equation. The weight w is designed to favor clusters close to the input fragment f and it is computed as:

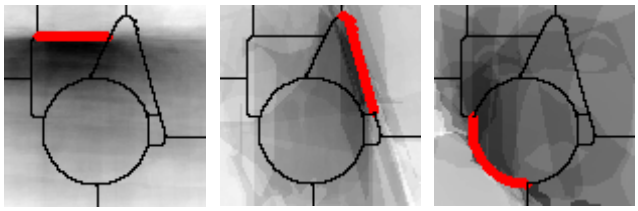
$$\boldsymbol{w}(\boldsymbol{f}, \boldsymbol{f}_j^o) = \boldsymbol{N}_{cj} \exp(-\boldsymbol{k} \cdot \boldsymbol{d}(\boldsymbol{f}, \boldsymbol{f}_j^o)), \qquad (8)$$

where k is set to 4 in the experiments below and $d(.,.)$ is the distance defined in Eq. (3).

(a) Input image

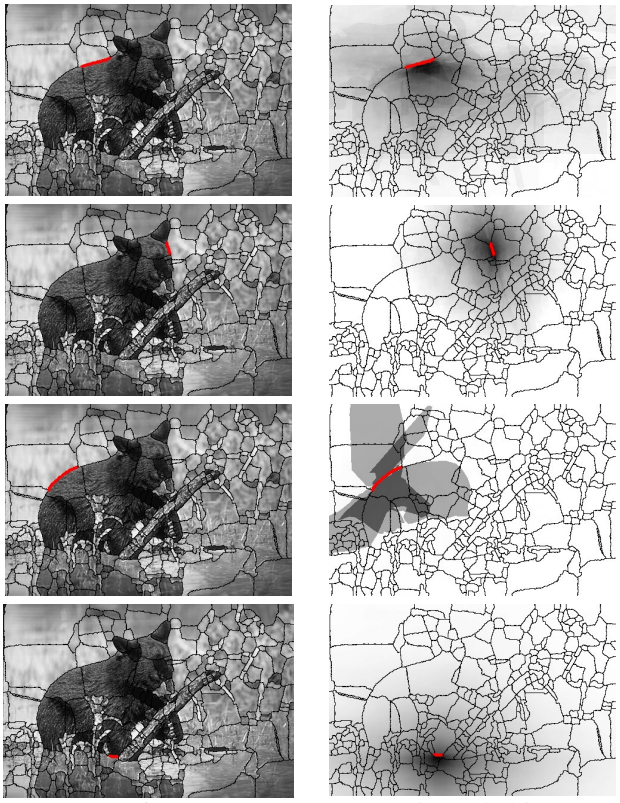


(b) Occlusion boundaries (red lines)



(c) Object likelihood

Fig. 5 Object likelihood for a single fragment (see text).



Target fragment (red lines)

Estimated object likelihood

Fig. 6 Object likelihood for a single fragment with natural image (see text).

Fig. 5 shows examples of object likelihood maps. With the input image (Fig. 5 (a)) and the potential occlusion boundaries estimated by the algorithm of Section 2 (Fig. 5 (b, red lines)), the object likelihood for every potential occlusion boundaries is estimated. Fig. 5 (c) shows the object likelihood for a horizontal fragment, a slanted fragment and a convex fragment from left to right. In Fig. 5 (c), the object likelihood increases as the pixel color becomes darker. The red lines in Fig. 5 (c) are the target fragments with which the object likelihood is estimated. In the horizontal fragment and the slanted fragment case, the likelihoods of pixels under the fragments are higher than those above them. In the convex fragment case, the likelihoods of pixels inside the convex fragment are higher than those of the pixels on the outside. These biases are obtained with the fragment and their regions in the codebook automatically.

Fig. 6 shows examples of object likelihood estimated from a natural image. The object likelihoods of the pixels belonging to the true objects (bear) are higher than those of the other regions when estimated with the slanted fragments in the top two rows. Though the object likelihoods estimated with the convex fragment in the third row images is composed of a few object region exemplars, the object likelihoods of the pixels inside the convex fragments are higher than those of the pixels on the outside. With the small horizontal fragments in the bottom row images, the object likelihoods of the water region become higher than those of the bear regions. However, the object likelihoods of the pixels above the target fragment are also high and they are still useful for the estimating the object likelihood map for the whole image describes in the following section.

3.3. Estimating the object likelihood map from the whole image

After obtaining the set of candidate occlusion boundary fragments $f_1,..,f_n$, we estimate the likelihood of pixel i to be on an object in the image by combining the contributions of all the fragments to pixel i:

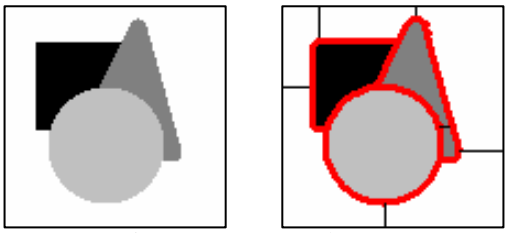
$$p_i = \sum_j p_j(x)c(f_j). \qquad (9)$$

The same abuse of notation is used as before for x. If $C(f_j)$ is the value of the output of the boundary detection algorithm of Section 2., $c(f_j)$ is the confidence normalized over the entire image:

$$c(f_j) = \frac{1}{\sum_{f_i \in F} C(f_i)} C(f_j). \qquad (10)$$

Fig. 7 shows the result of the object region estimation on a toy image. Fig. 7 (a) shows the input image, (b) shows the occlusion boundary fragments (red lines), (c) is the estimated result and (d) is the averaged likelihood of the segments. Brighter regions or segments correspond to higher likelihood values in (c) and (d). As seen in Fig. 7 (d), the pixels on the objects have higher likelihood than the background.

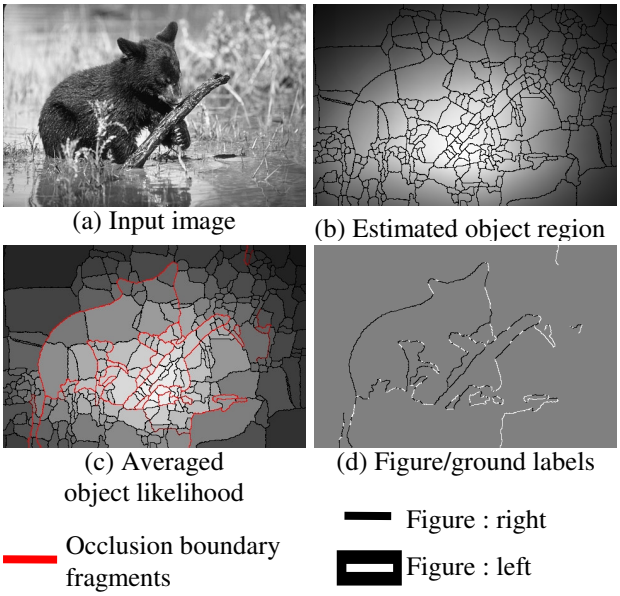
In all of the examples shown in this section, we used 100 training images from the data set of [9] from which the algorithm of Section 2 generated 9852 fragments which are summarized in 142 clusters.



(a) Input image (b) Fragments

(c) Estimated object region (d) Averaged object likelihood

Fig. 7 Likelihood estimation for the whole image.



(a) Input image (b) Estimated object region

(c) Averaged object likelihood (d) Figure/ground labels

Fig. 8 Labeling boundary fragments.

3.4. Estimation results

To evaluate the performance of the object region estimation, the fragments in the images are assigned the figure/ground labels through the following algorithm:

1. Estimate the object regions for the whole image (Fig. 8 (b)).
2. Average the likelihood in each segment (Fig. 8 (c)).
3. Assign figure/ground labels to each occlusion fragment by comparing the averaged figure probability (Fig. 8 (c)) of the two segments connected to the fragment (Fig. 8 (d)).

After the figure/ground labels are obtained, the evaluation is performed by following the same protocol as in [9] and the result is compared with the local shapeme and global CRF based on the *Pb* boundaries described in [9]. The precision rate of the figure/ground estimation, 65.1%, is comparable to the precision rate of Ren's method [9] based on local shapemes. It is lower by 4% than Ren's global CRF because we do not use an additional step of global smoothing. This is a natural extension for future work. This result shows that this approach is competitive with the most related approach with the key difference that it produces additional information in the form of the object likelihood maps. In the next section, we combine the likelihood maps with the segmentation of Section 3. to generate the final object segmentation.

4. Combining segmentation and object region estimation

The last step is to combine the segmentation selected by the algorithm of Section 3 with the likelihood map generated by the algorithm of Section 4. The proposed method obtains the object regions by integrating the optimal segmentation chosen by normalized mutual information and the estimated object likelihood images. The integration is performed as follows:

1. Generate the boundary fragments of the segmentation (Fig. 9 (b)) chosen by normalized mutual information (Fig. 9 (c)).
2. Obtain the average object likelihood (AOL) from the object likelihood image estimated in Section 4 (Fig. 9 (d)) in two thin regions around each fragment (Fig. 9(e)).
3. Assign figure/ground labels to the fragments (Fig. 9(f)) by calculating the confidence values c_f :
$$c_f = (Left\ AOL) - (Right\ AOL) \cdot \qquad (11)$$
Each fragment is oriented and the left and right sides are defined according to the fragment direction.
4. Threshold the absolute confidence values (Fig. 9(g)).
5. Derive the figure confidence map (Fig. 9(h)) by assigning to each segment a figure confidence c_s :
$$c_s = \frac{1}{N_p}((\#\ \textbf{\textit{figure labels}}) - (\#\ \textbf{\textit{ground labels}}), \qquad (12)$$
where N_p is the number of pixels on the all fragments surrounding the segment.
6. Decide the threshold which divides the figure confidence map into the figure region and the background region by separating the two regions so that their combined variance is minimal.
7. The final result is obtained by thresholding the figure confidence map with the threshold obtained at the previous step (Fig. 9(i)).

Fig. 10 shows other examples on natural images. Even though each object region includes some background, this method detects the approximate object regions. Table 1 summarizes the quantitative evaluation of the system on the

data of [9]. We used the same procedure as Ren's [9] to derive the precision rate. The generation of the likelihood image from boundary fragments yields similar performance as the local approach of [9] based on shapemes and *Pb* boundaries. However, it enables us to combine the evidence from the boundary fragments with region segmentation to yields higher performance than the global version of [9] in which the local evidence is combined into a global interpretation by using a CRF based on *Pb* boundaries. The performance is evaluated on 100 test images, using 100 images for training. As in Section 4, we used 142 clusters to represent the fragments from the 100 training images.

Fig. 11 shows some failure examples. The shadow regions are detected in the first row image. The proposed method cannot distinguish the shape of the physical boundaries and the shape of the shadow boundaries because the shadow shape reflects the physical boundaries. This method also finds the trees in the second image because there are some similar boundaries between the sky and the trees in the training data and the tree regions are more figural than the sky. The polar bears in the third row image are not detected. The boundaries of the right polar bear's neck are assigned the wrong labels because the downside boundaries of the neck are almost horizontal and this method estimates that the figure regions for these boundaries are under them.

Table 2 Precision of figure/ground labeling

Object region from boundary fragments (Section 4)	Combination boundary fragments and region segmentation (Section 5)	Local shapeme [9]	Global CRF [9]	Human labeled ground truth
65.1%	78.0%	64.9%	68.9%	88%

5. Conclusion

This paper describes a general approach to estimating object regions from local contour configurations. The shape of the object contours implies the positions of the objects. We extract the relations between the contour shape and the object position and estimate the figure region in the image with the relations. We also automatically choose the optimal segmentation among the multiple segmentations with normalized mutual information. By integrating the figure region estimation and the optimal segmentation, the important object regions are segmented out without the object specific knowledge.

Acknowledgment

This work was partially supported by NSF Grant IIS0713406.

References

[1] P.F. Felzenszwalb, and D.P. Huttenlocher, Efficient Graph-Based Image Segmentation, IJCV, V. 59, No. 2, 2004

[2] D. Comaniciu, and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI, 2001.

[3] J. Shi and J. Malik, Normalized Cuts and Image Segmentation, PAMI, Vol.22, No.8, 2000.

[4] J. Malik, S. Belongie, T. Leung, and J. Shi, Contour and Texture Analysis for Image Segmentation, IJCV, Vol. 43, pp. 7-27, 2001.

[5] M. Dimiccoli, and P. Salembier, Exploiting T-Junctions for Depth Segregation In Single Images, ICASSP, 2009.

[6] D. Hoiem, A. Stein, A. Efros, and M. Hebert, Recovering Occlusion Boundaries from a Single Image, ICCV, 2007.

[7] E. Rubin, Figure-Ground Perception, In Readings in perception, 1958.

[8] H.K. Pao, D. Geiger, and N. Rubin, Measuring convexity for figure/ground separation, ICCV, 1999.

[9] X. Ren, C. Fowlkes, and J. Malik, Figure/Ground Assignment in Natural Images, ECCV, 2006.

[10] S. Yu, T. Lee, and T. Kanade, A Hierarchical Markov Random Field Model for Figure-ground Segregation, EMM CVPR, 2001.

[11] J. Shotton, A. Blake, and R. Cipolla, Contour-Based Learning for Object Detection, ICCV, 2005.

[12] V. Ferrari, T. Tuytelaars, L.Van.Gool, Object Detection by Contour Segment Networks, ECCV, 2006.

[13] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, Groups of Adjacent Contour Segments for Object Detection, PAMI, Vol. 30, No. 1, pp. 36-51, 2008.

[14] Q. Zhu, L. Wang, Y, Wu, and J. Shi, Contour Context Selection for Object Detection: A Set-to-Set Contour Matching Approach, ECCV, 2008.

[15] P. Viola, and M. Jones, Robust Real-time Object Detection, ICCV, 2001.

[16] N. Dalal, and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR, 2005.

[17] A. Stein, D. Hoiem, and M. Hebert, Learning to Find Object Boundaries Using Motion Cues, ICCV, 2007.

[18] A. Stein, T. Stepleton, and M. Hebert, Toward Unsupervised Whole-Object Segmentation: Combining Automated Matting with Boundary Detection, CVPR, 2008.

[19] D. Martin, C. Fowkles, D. Tal, and J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, ICCV, 2001.

[20] D. Martin, C. Folkes, and J. Malik, Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues, PAMI, Vol. 26, No. 5, 2004.

[21] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, Using Contours to Detect and Localize Junctions in Natural Images, CVPR, 2008.

[22] A. Levin, D. Lischinski, and Y. Weiss, A closed form solution to natural image matting, CVPR, 2006.

[23] P. Wattuya, K. Rothaus, J-S. Prasni, and X. Jiang, A Random Walker Based Approach to Combining Multiple Segmentations, ICPR, 2008.

[24] V. Ferrari, T. Tuytelaars, and L. Van Gool, Real-time affine Region Tracking and Coplanar Grouping, CVPR, 2001.

[25] P.F. Felzenszwalb, and D.P. Huttenlocher, Distance Transforms of Sampled Functions, Cornell Computing and Information Science Technical Report, 2004.

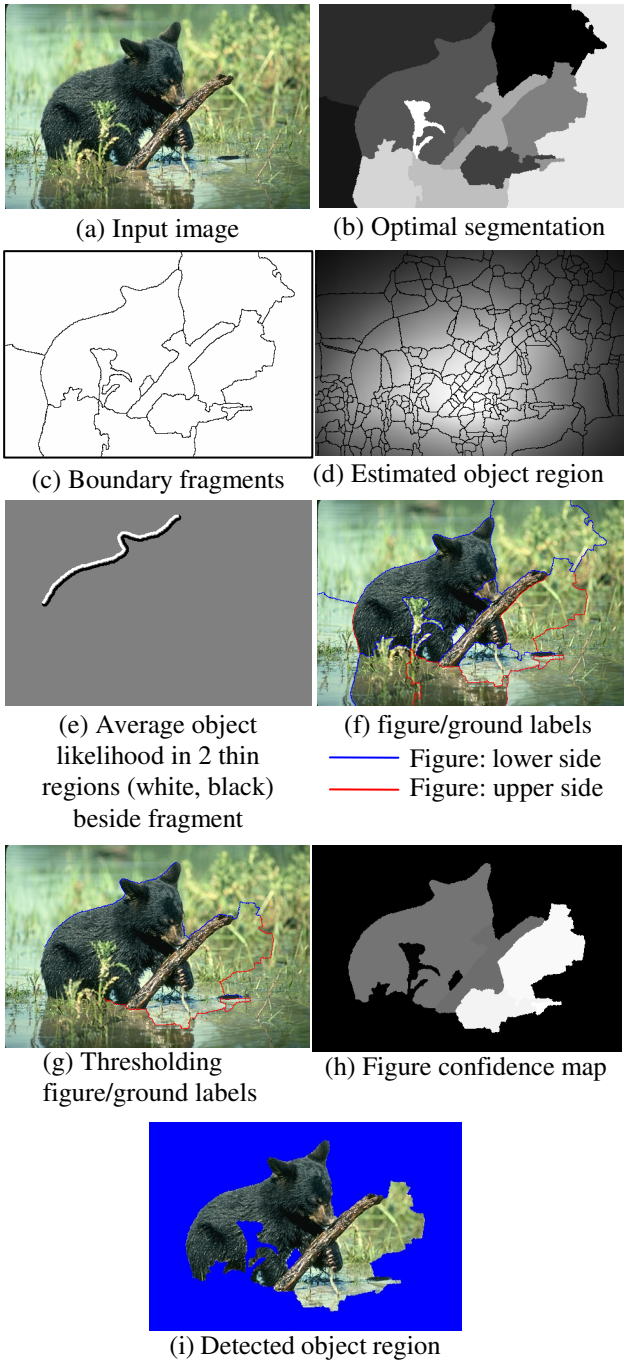


Fig. 9 Combining segmentation and object likelihood.

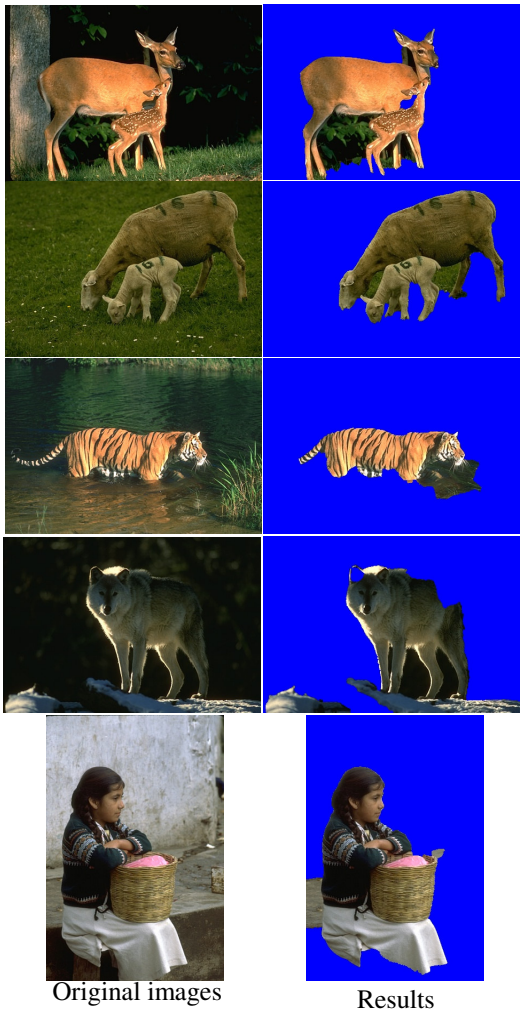


Fig. 10 General object segmentation results.

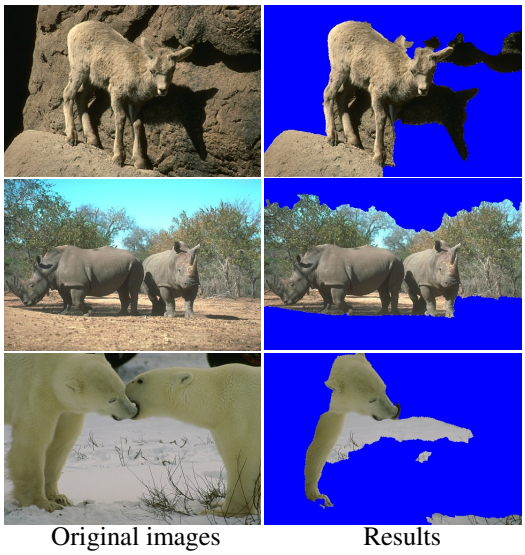


Fig. 11 Example of incorrect segmentations.