IMAGE SEGMENTATION USING HIERARCHICAL MESHES

Dong-Keun Lim and Yo-Sung Ho

Kwangju Institute of Science and Technology
1 Oryong-Dong Puk-Gu, Kwangju, 500-712, Korea
Email : {dklim,hoyo}@kjist.ac.kr

ABSTRACT

The object boundary of an image plays an important role for image interpretation. In this paper, we introduce a concept of hierarchical mesh-based image segmentation for finding object boundaries. In each hierarchical layer, we employ neighborhood searching and boundary tracking methods to refine the initial boundary estimate. We also apply a local region growing method to define closed contours. Experimental results indicate that reliable segmentation of objects can be accomplished by the proposed low complexity technique.

1. INTRODUCTION

Image segmentation is known to be one of the most challenging topics in the field of image processing. In recent years, several algorithms for image segmentation have been proposed for certain applications, such as video conferences, where only one or two speakers exist in the static background [1-2].

Typical image segmentation algorithms include thresholding, region growing, split and merge, watersheds, or edge-based operations. Each operation has its own peculiar features and advantages.

Initial segmentation is performed on the first frame of the video sequence by partitioning the frame into homogeneous regions based on intensity values. The watershed algorithm [3-4] and the region growing algorithm [5-7] are popularly employed for initial segmentation; however, both of them require a large amount of computation time. In order to overcome this problem, we propose a hierarchical approach for image segmentation using a mesh structure. The proposed algorithm also increases the robustness of linkage of object boundaries.

2. A NEW SEGMENTATION METHOD

In this paper, we consider the imaging system as a two-stage structure. The first stage performs low-level image processing, where we extract the image structure. The second stage performs high-level image processing, where hierarchical structures are inferred from the output of the first stage.

The proposed segmentation method is based on hierarchical mesh classification with a pyramid data structure. The advantage of the hierarchical approach lies in the possibility of making a rough classification at a coarse level and then continuing into finer resolution to improve the segmentation accuracy. The pyramid structure also provides the important property of focusing attention on interesting parts of the image. Our algorithm pays attention only to meshes along object boundaries. The proposed algorithm is computationally efficient since only a fraction of all pyramid nodes is processed during the top-down classification [1,4].

The main idea behind our proposal is to perform hierarchical segmentation of the first frame based on object boundary refinement in the mesh structure. Figure 1 shows the flow diagram of the hierarchical image segmentation, which is similar to the divide-and-conquer algorithm for boundary detection.

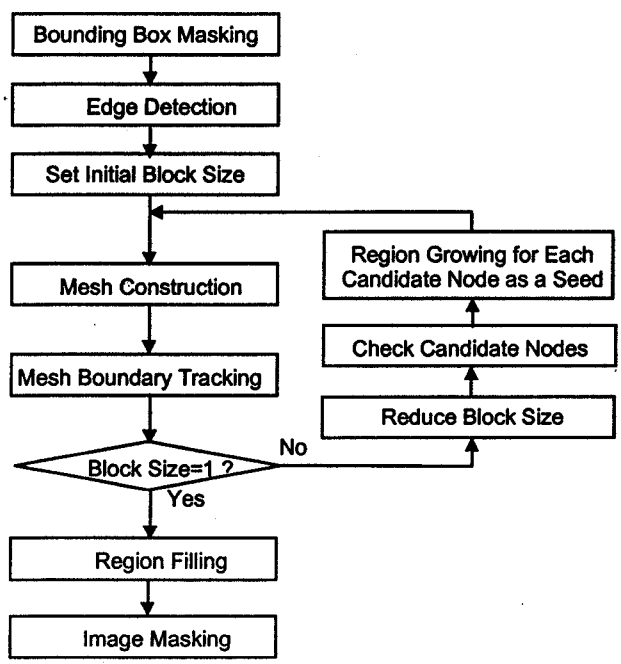


Figure 1: Hierarchical Mesh-based Segmentation

0-7803-5467-2/99/$10.00 © 1999 IEEE

We construct a hierarchical structure of meshes of different sizes using the edge information. Meshes are constructed by connecting the centroid of each candidate block. The segmentation mask region is obtained by searching the mesh boundary. To follow the mesh boundary, we use a left-hand or a right-hand rule of a maze search based on the previous search direction. The algorithm works counterclockwise for the left-hand rule or clockwise for the right-hand rule along the meshes on the object boundary. The resulting boundary meshes are candidates of the next process in the mesh hierarchy.

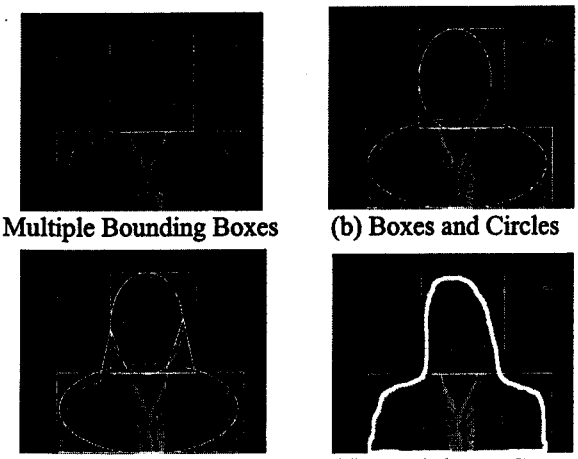
The candidate meshes are split into smaller meshes. We apply the mesh boundary tracking method and link the mesh boundary. This process continues iteratively until we reach one pixel resolution. In each level, local searches are carried out on the previous approximate boundaries.

Since the resulting object boundary may not be accurate, we need to correct the boundary. We employ the region growing algorithm to refine the boundary starting from the previous boundary as a seed. A segmentation mask is obtained by filling the refined object boundary.

2.1 Bounding Box Masking

Due to semantic ambiguity and content complexity, automatic segmentation algorithms are applied only on specific situation. For general applications, the object is usually defined by the user. Semantic objects in the video frame can be identified by bounding boxes.

As shown in Figure 2, multiple rectangular boxes, circles and hand-drawn lines can be used to define the region of interest(ROI). We can also combine those bounding shapes.



(a) Multiple Bounding Boxes (b) Boxes and Circles

(c) Boxes, Circles and Polygons (d) Hand-drawn Curve

Figure 2: Boundary of Region of Interest

2.2 Boundary Tracking

An image can be represented by a mesh structure [2,8]. Meshes are generally located on long object boundaries, which are important in the hierarchical structure since they provide the meaning of the objects. The object boundaries can be obtained by a polygonal approximation. In order to obtain coherent edge features from individual local edge segments, we perform several steps of segmentation in the hierarchical mesh structure.

After we find locations of candidate boundary points, we refine the object boundaries. Figure 3 explains a mechanism of hierarchical mesh construction and boundary linking. At Level k, edges are located in three quadrants. Those regions are candidates for Level k+1. In the same way, at Level k+1, edges are located in seven regions, which are candidate locations for Level k+2.

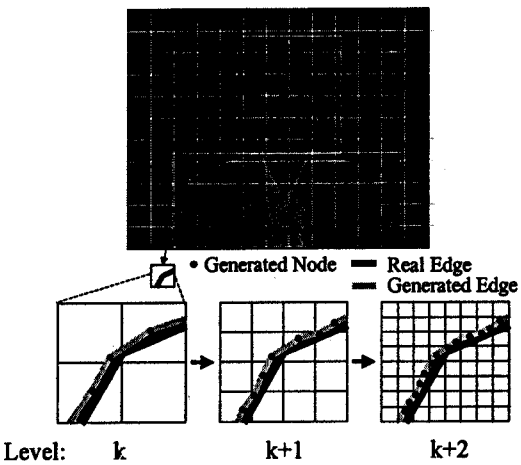


Figure 3: Object Locations at Different Levels

Figure 4 shows an example of complex object boundary tracking. As proceeding to the next level, we can refine the object boundary.

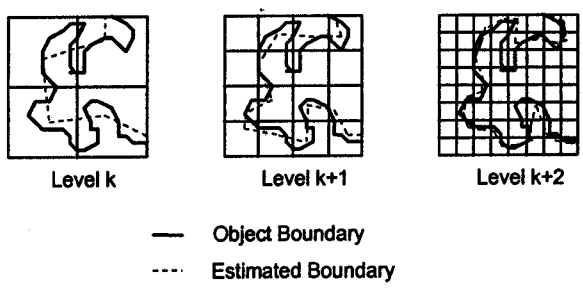


Figure 4: Boundary Refinement

A mesh structure can be generated by linking the centroid (center of mass) position (Cx, Cy) of each block, whose coordinate values are defined by

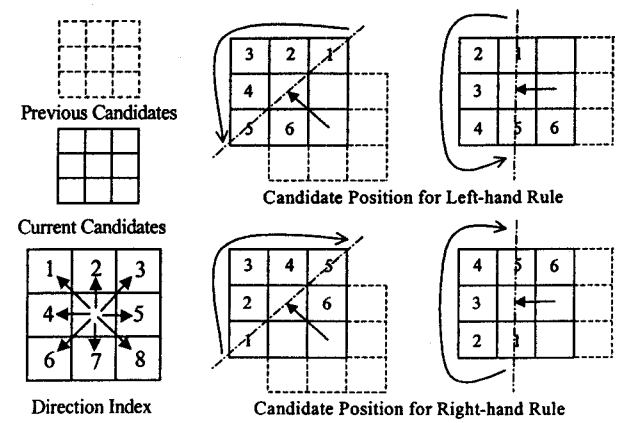
$$c_x^{i,j} = \frac{\sum_{k=0}^{BS-1}\sum_{l=0}^{BS-1} k \cdot I(i*BS+k, j*BS+l)}{\sum_{k=0}^{BS-1}\sum_{l=0}^{BS-1} I(i*BS+k, j*BS+l)} \quad (1)$$

$$c_y^{i,j} = \frac{\sum_{k=0}^{BS-1}\sum_{l=0}^{BS-1} l \cdot I(i*BS+k, j*BS+l)}{\sum_{k=0}^{BS-1}\sum_{l=0}^{BS-1} I(i*BS+k, j*BS+l)} \quad (2)$$

where (i,j) is the index of each block, BS stands for block size, and I(.) represents the edge intensity value of the edge position.

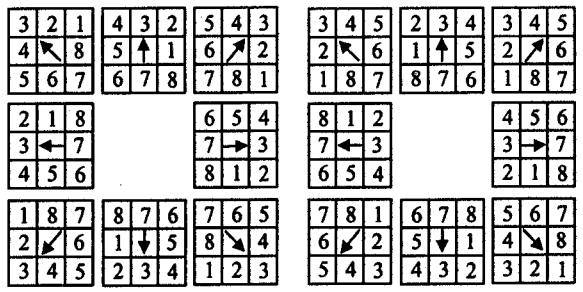
In order to track the correct edge boundary, we can use a left-hand or a right-hand rule. The following algorithm can effectively find the object boundary.

a) Starting from the upper left corner of the image, we perform raster scanning line by line until we reach the position of an edge block. This position is used as the starting point of boundary tracking. The left-hand rule is initially applied for searching.
b) Check the 3×3 neighborhood of the current position to find an edge block according to the priority of candidates. This procedure is explained in Figure 5 and Figure 6. If we miss a link because of the image boundary before we have a complete closed loop, we return to the previous starting position. We try another search with a different searching rule for the opposite direction of the object boundary. This operation is performed iteratively until all object boundaries are visited.
c) At each edge block position, we define a variable, DIR, which stores the direction of the previous movement along the edge boundary. The next boundary position is searched using the priority tables, which are determined by DIR.
d) Reduce the edge block size and perform local region growing for the object boundaries.
e) The same process is repeated until we obtain the object boundary at one pixel accuracy.



(a) Index of Directions (b) Search for Candidate Positions

Figure 5: Boundary Tracking



(a) Left-hand Rule (b) Right-hand Rule

Figure 6: Priority for Boundary Tracking

Figure 7 shows an example of boundary tracking, and Table 1 summarizes edge positions and directions. We start with the left-hand scan. When we reach the image boundary at No. 18, we switch to the right-hand scan. The next position can be found from the direction. A side branch, which exists from No. 13 to No. 16, should be removed to generate a closed segmentation mask.

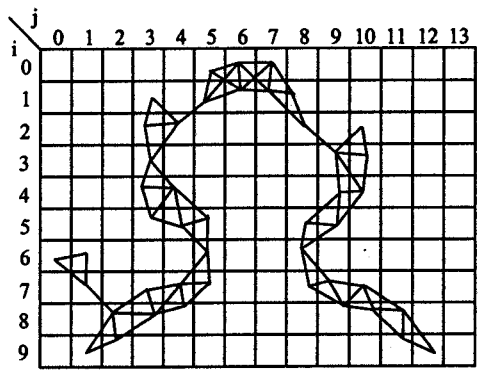


Figure 7: Example of Boundary Tracking

Table 1: Direction and Position Values

No	Pos	DIR	Scan	No	Pos	DIR	Scan
0	(0,5)	7	left	17	(8,2)	8	left
1	(1,5)	6	left	18	(9,1)	6	left
2	(2,4)	1	left	19	(0,5)	5	right
3	(1,3)	7	left	20	(0,6)	5	right
4	(2,3)	7	left	21	(0,7)	8	right
5	(3,3)	7	left	22	(1,8)	7	right
6	(4,3)	7	left	23	(2,8)	8	right
7	(5,3)	7	left	24	(3,9)	3	right
8	(5,4)	5	left	25	(2,10)	7	right
9	(6,5)	8	left	26	(3,10)	7	right
10	(7,4)	6	left	27	(4,10)	6	right
11	(7,3)	4	left	28	(5,9)	6	right
12	(8,2)	6	left	29	(6,8)	8	right
13	(7,1)	1	left	30	(7,9)	5	right
14	(6,1)	2	left	31	(7,10)	8	right
15	(6,0)	4	left	32	(8,11)	8	right
16	(7,1)	8	Left	33	(9,12)		end

2.3 Edge Detection and Region Growing

A contour pixel is defined as a local maximum in the output of an edge detector or the first derivative operator. Since image edges characterize object boundaries, they are important for image segmentation. The goal of image segmentation is to find edge segments whose contours have significant contrast changes.

Region growing is a simple scheme for image segmentation. If the gray-level difference between two adjacent pixels is below a predetermined threshold value, those pixels are merged. We can use a statistic of object segments, instead of the pixel difference.

Both edge detection and region growing are two different aspects of the same process under the assumption of step edges and smooth brightness distribution within regions. Both operations can be unified by deciding whether a point is on an object boundary or in a homogeneous region [1,7]. Homogeneity may be measured in terms of color, texture, motion, depth, etc. In this paper, we examine gray-level changes.

In the region growing method with hybrid linkage, an edge operator is applied to each pixel, which is labeled as edge or nonedge. Neighboring pixels, which are not belonging to an edge, can be joined to an image segment. Image segments are connected components of nonedge pixels. Since edge detection on noisy or complex scenes often produces missing or extra edge segments, we sometimes generate open contours.

The region growing method with statistics linkage compares a pixel value with some meaningful statistics of neighboring segments. An example of the statistics is the average pixel value of the rectangular block. Two neighboring segments can be merged if their means are similar or if their adjacent pixels along the boundary have similar intensity values. This method always produces connected regions, but it requires high computational complexity.

The two region growing methods can be combined to overcome the problems of missing edges and high computational complexity. The combined region growing method is applied to the object boundaries locally. This method generally produces a closed edge line. When we reduce the block size, we check whether each block has a significant edge in its region. For this purpose, we compare the average intensity value of each block with a given threshold value. When we perform local region growing, we use the output of an edge detector.

In this work, we employ the Sobel operator. The Sobel operator is popularly used as a simple edge detector, which takes the first order derivative in the vertical direction(h_1) and in the horizontal direction(h_2).

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

The output of the local region growing method is an initial candidate for the object boundary tracking in the next level.

2.4 Region Filling

In order to find the segmentation mask, we fill each region based on object boundary information. To fill a region with a certain gray value, we set each pixel lying on a scan line running from the left edge to the right edge to the same pixel value. The general polygon scan-conversion algorithm [9] handles both convex and concave polygon, even those are self-intersecting or have interior holes. It operates by computing spans that lie between left and right edges of the polygon. The span extrema are calculated by an incremental algorithm that computes a scan line-edge intersection from the intersection with the previous scan line. Figure 8 shows the region filling process. The spans can be filled in by a three-step process.

a) Find the intersections of the scan line with all edges of the polygon.
b) Sort the intersections by increasing x coordinate.
c) Fill in all pixels between pairs of intersections that lie interior to the polygon, using the odd-parity rule to determine that a point is inside a region. Parity is initially even, and each intersection encountered thus inverts the parity bit-draw when parity is odd, do not draw when it is even.

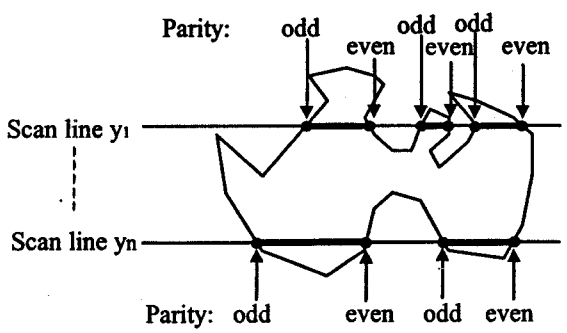


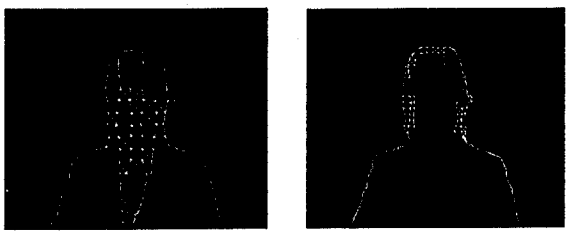
Figure 8: Region Filling

2.5 Image Masking

In order to obtain image objects, we apply the AND operation of the original image and the segmentation mask.

3. SIMULATION RESULTS

Computer simulation is performed on video conferencing images of the CIF(352×288) format. We use two head and shoulder images, CLARE and AKIYO. Figure 9 shows two-level meshes of different block sizes.

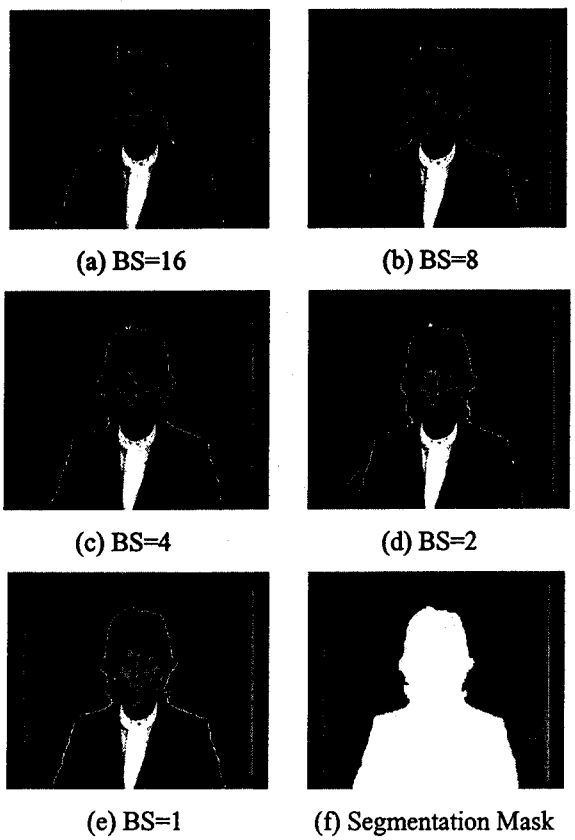


(a) BS=16 (b) BS=8

Figure 9: Hierarchical Meshes of Different Block Sizes

In the first step, we partition the image into 16×16 blocks and generate meshes in the region of interest. Meshes are mainly distributed around edge points. Using the boundary tracking process, we find object boundaries to generate closed contours at each hierarchical level. In the next step, we use 8×8 blocks and smaller meshes to refine the object boundaries. The local region growing method increases robustness of closed contour generation. Since meshes are only distributed around object boundaries, it reduces the computation time in the next level.

Figure 10 shows the outputs of the proposed method, where each frame uses a different block size(BS). The squares in the figures indicate the block size. As proceeding to the next level, we refine object boundaries.



(a) BS=16 (b) BS=8

(c) BS=4 (d) BS=2

(e) BS=1 (f) Segmentation Mask

Figure 10: Segmentation Results

Segmented images are generated by the AND operation of the original image and the segmentation mask.

4. CONCLUSION

In this paper, we propose a new image segmentation algorithm using hierarchical meshes. Reliable segmentation of objects is obtained by the proposed low complexity method. It is desirable to have mesh boundaries aligned with object boundaries and edge information of the image to achieve better perceptual quality. We obtain the shape information of the image object from the intermediate results.

ACKNOWLEDGMENTS

This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) through the Ultra-Fast Fiber-Optic Networks (UFON) Research Center at Kwangju Institute of Science and Technology (K-JIST).

REFERENCES

[1] L. Westberg, "Hierarchical contour-based segmentation of dynamic scenes," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 14, no. 9, pp. 946-952, September 1992.

[2] Y. Altunbasak, "Object-scalable mesh-based coding of synthetic and natural image objects," ICIP'97, pp. 94-97, October 1997.

[3] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," IEEE Trans. Pattern Analysis Machine Intelligence, vol. 13, pp. 583-598, June 1991.

[4] L. Najiman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 18, pp. 1163-1173, December 1996.

[5] S.A. Hojjatoleslami and J. Kittler, "Region growing: a new approach," IEEE Trans. Image Processing, vol. 7, no. 7, July 1998.

[6] T. Pavlidis and Y.T. Liow, "Integrating region growing and edge detection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 7, no.3, pp. 225-233, September 1992.

[7] M. Tabb and N. Ahuja, "Multiscale image segmentation by integrated edge and region detection," IEEE Trans. Image Processing, vol. 6, pp. 642-655, May 1997.

[8] O. Lee and Y. Wang, "Nonuniform image sampling and interpolation over deformed meshes and its hierarchical extension," Visual Comm. And Image Proc. SPIE vol. 2501, pp.389-400, May 1995.

[9] J.D. Foley, et al., Introduction to computer graphics, Addison-Wesley, 1994.