The 15th Japan-Korea Joint Workshop on Frontiers of Computer Vision, 5-7 February 2009, Andong, Korea

Graph Generation from Over-segmentation for Graph Based Approaches

Hanbyul Joo† Yekeun Jeong‡ and In So Kweon§

†:KAIST , hbjoo@rcv.kaist.ac.kr
‡:KAIST, ykjeong@rcv.kaist.ac.kr
§:KAIST, iskweon@kaist.ac.kr

Abstract In the computer vision area, many algorithm use graph-based approaches to solve some problems. Conventionally, given an image, we can treat each pixel as node and the connection between pixels as nodes. By this manner, we can simply transform the problems about images as graph-based problems.

However, we can apply similar strategy to over-segmented images. In this paper, we present new method to generate graphs from the over-segmented images. In the generated graph, the edges mean the groups of edge pixels which is connected and have same adjacent segments. And, the nodes mean the junction between the edges. This graph have many advantage because it has smaller size than conventional graph generate from raw images. Moreover, because proposed graph preserve every possible way to utilize over-segmented result, it is possible to transform over-segmentation based strategy to graph-based approaches.

1 Introduction

In computer vision area, there are many approaches to treat an image as an graph to solve some problems. To transform image-based problem as an graph-based problem, we can treat each pixel as node and the connection between pixels as nodes. By this manner, it is possible to apply many graph-based algorithms for image processing and computer vision area. In [11, 7], they treat an image as an graph and they apply graph-based approach to solve the segmentation problem.

However, we can apply similar strategy for over-segmented images if we can transform the over-segmented images to graphs. That is, we can transform the over-segmentation based approach as graph-based approach by generating a graph from an over-segmented image.

Over-segmentation means the method which divides an image into multiple homogeneous regions. In the over-segmentation result, it can be assumed that each segment has similar texture or color information. Over-segmentation technique is very useful because it can reduce the computational complexity. Moreover, using the over-segmentation result, we can utilize the rich information such as texture, boundary shape, and color of the segments. In the recent years, many researches make excellent results by combining this over-segmentation technique [6, 3, 8].

However, if we can generated the graph from over-segmentation result, many graph-based algorithm can be applicable. At first, similar to ordinary edge detector, we can extract an edge map from an over-segmented image by extracting the pixels between two segments because there exist high discontinuity. Then, in our generated graph, an edge is defined by an group of the edge pixels which are connected and have same adjacent segments. Thus, in the generated graph, an edge includes the pixels which separate adjacent segments. The nodes mean the junction among more than two edges.

This graph have several useful properties. First, it has smaller size compared to the graph generated from the raw image. Secondly, the problem about this graph is equivalent to the problem of the over-segmented image. Because we define the graph edges by the groups of edge pixels which have same adjacent segments, an edge can be the unit group for the separation between adjacent segments. So, we can apply our proposed graph for region detection or foreground extraction approaches by finding pathes or cycles in the graph.

This paper is structured as follows. In section 2, we describe in detail how we generate graph from over-segmented image. The experimental results are explained in section s3, and the conclusion is given in section 4.

2 Graph Generation Algorithm

In this section, we propose graph generation method from an over-segmentation result. To generate a graph, we have to define nodes and edges from over-segmented image. In 2.2, we address the node definition, and in 2.3 we address the edge definition.

2.1 Over-segmentation and Refinement

At first, we divide an image into homogeneous regions using over-segmentation algorithm. Any over-segmentation algorithm can be used. According to our experience, the segmentation method of Felzenszwalb and Huttenlocher [7] is very fast but there exist a lot of small pieces of segments around edge regions which makes graph generation difficult. The normalize cut [11] makes relatively clean and simple over-segmentation results as an example shown in Fig. 1 (b). But, it takes a lot of processing time. In our experiment, we use the pyramid segmentation algorithm in the open source computer vision library [1]. This algorithm makes quite reasonable results with little processing time. Moreover, the boundary parts between between two segments are clear compared to [7]. The result is shown in the Fig. 1 (e).

Before generating the graph from over-segmented image, refinement is required for easier node and edge extraction step. The refinement is composed of two step. At first, we remove thin regions which have sing-pixel width. Then, we remove the isolated segments which are enclosed by a segment. So, isolated segments is merged with adjacent segments. The final over-segment result is shown in Fig. 2.

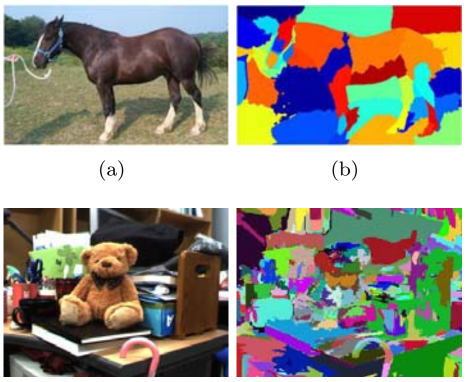
(a) (b) (c) (d)

Fig. 1: (a) input image (b) over-segmentation result using Normalized Cut [11] (c) input image (d)over-segmentation result using pyramid segmentation in OpenCV [1]

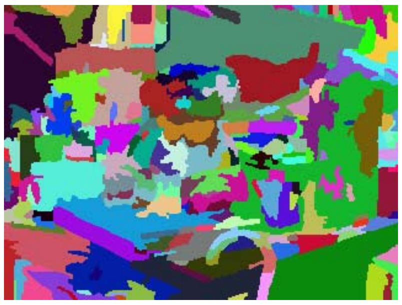

Fig. 2: Refinement result from Fig. 1 (d)

2.2 Node extraction

When we think the boundaries among segments as edge pixels, the junction points can be thought as nodes because there are several path at those points. In our work, we define a node point as the position at which more than tree segments meet.

To extract node points from an over-segmented image, we search the segment boundary pixels using 2 by 2 windows. Thus, only two kind of node can be exist show 3 (a). This approach makes it easy to extract the edge which connect two nodes which is explained in next section. The node extraction result is shown in Fig. 4.

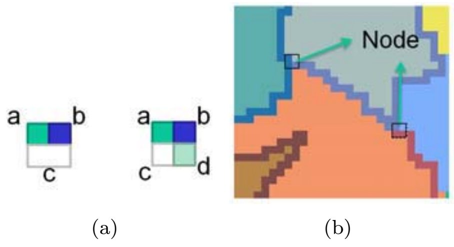

Fig. 3: (a) nodes searched by 2x2 window (b) node examples

2.3 Edge extraction

In this section, the method to extract an edge of the graph, which is a group of edge pixel, between two end nodes is explained. Among all the edge pixels which located between segments, we try to group the pixels which connect two end nodes.

However,if we just follow the connected edge pixels from a node, it could fail as shown in Fig. 5 (b). In Fig. 5, the node A is defined by segment a,b,c ,

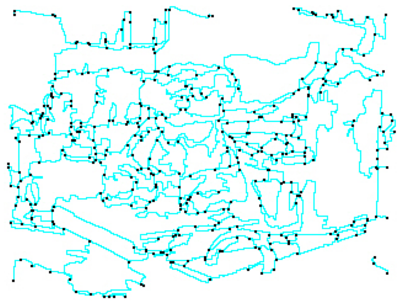
Fig. 4: Node extraction result from Fig. 2

and node B is defined by segment a,b,d. The desired group of edge pixels which connect node a and b is path 2. However, when we following the any connected neighborhood without considering, there could be wrong following like path 1 in Fig. 5 (b).

To avoid this kind of failures, we have to determine which connected neighborhood should be followed. Thus, to solve this, we consider the adjacent segments.

At first, we give a label for each edge pixel and nodes. we define the edge pixel label using adjacent segment pair. For example, in 5, the pixels between segment a,b has label (a,b) and the pixels between segment b,c has label (b,c). Intuitively, same label of pixels should be included in same group.

We also give the label for each node using the set of connected edge labels. For example, node a has a label like { (a,b), (b,c), (c,a) }, which also means that the node is the junction point of three edge group which have labels as (a,b),(b,c),(c,a). Thus, we can think theorem 1 like follows.

Theorem1 When we think an connected edge pixels which have label α and started from from the node $\{\alpha, \beta, \gamma\}$, we can reach and only reach another end node which also has the label α as an element.

Theorem1 is obvious because the reached node should include the same label of followed edge

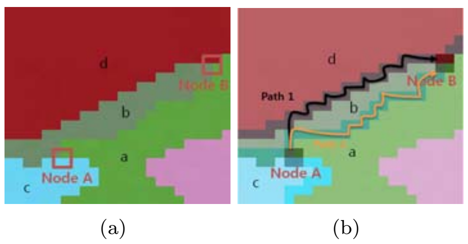

Fig. 5: (a) magnified image (b) wrong following of connect edge pixels

label. Additionally, we cannot cross to another label pixel like the path 1 in Fig. 5 (b). Thus, starting from a node, we can find the group of pixels by following the connected pixel which have the same label. And, the following is finished when we reach another end node which include followed edge label. Finally, these group of pixels are composed of an edge between the two end nodes. By this way we can define the edges of the graph.

3 Experimental Results

The final experimental results for several images are shown in Fig. 6. As mentioned 3.1, we used the pyramid segmentation algorithm in the open source computer vision library [1] for over-segmentation. Then, we extract nodes from over-segmented image. Fig. 6 (a) are input images. And the refined over-segmented images are shown in Fig. 6 (b).

Node extraction results are shown in Fig. 6 (c). The red rectangles mean nodes and the pixels between two segment is colored as black. Here, we draw one of two adjacent edge pixel for clear visualization.

The final graph generation results are shown in Fig. 6 (d). Each edge is drawn with different random color. The results show that the proposed graph generation method works correctly even in extremely complex images.

4 Conclusion

In this paper, we presented a method to generate graphs from over-segmentation results to utilize graph-based approaches. In the generated graph, the edges mean the groups of edge pixels which is connected and have same adjacent segments. And, the nodes mean the junction between the edges.

By utilizing the proposed graph, it is possible to treat the problems about computer vision as graph-based problems. The proposed graph has smaller size compared to the graph generated from the raw image, so we can reduce the computational complexity. Additionally, because proposed graph preserve every possible way to utilize over-segmented result, it is possible to transform over-segmentation based strategy to graph-based approaches. So, we can apply our proposed graph for region detection or foreground extraction approaches by selecting the path or cycle in the graph.

Acknowledgment This work was supported by the IT R&D program of MKE/IITA. [2008-F-030-01, Development of Full 3D Reconstruction

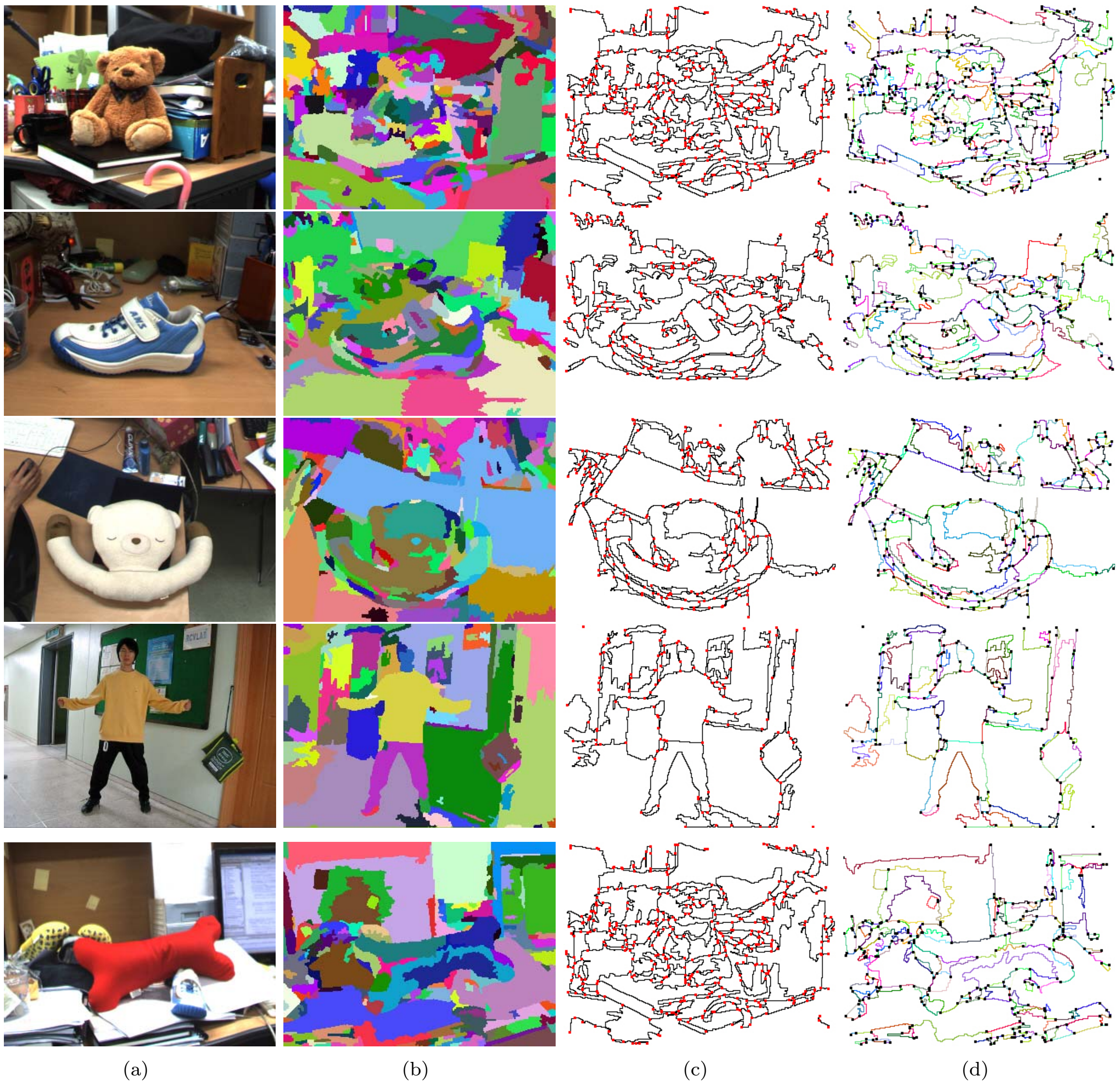

Fig. 6: (a) original images (b) refined over-segmented images (c) node extraction results (d) finally generated graphs

Technology for Broadcasting Communication Fusion]

References

[1] Open source computer vision library. *http://www.intel.com/technology/computing /opencv/index.htmE*, 2002.

[2] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching:two new techniques for image matching. *5th Int. Joint Conf. Artificial Intelligence*, pages 659–663, 1977.

[3] E. Borenstein, E. Sharon, and S. Ullman. Combining Top-Down and Bottom-Up Segmentation. In *Computer Vision and Pattern Recognition Workshop, 2004 Conference on*, pages 46–46, 2004.

[4] G. Borgefors. Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE trns. Pattern Analysis Machine Intelligence*, 10(6):849–865, 1998.

[5] J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 679–698, 1986.

[6] T. Cour and J. Shi. Recognizing objects by piecing together the segmentation puzzle. *CVPR*, 2008.

[7] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[8] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. In *Int. Conf. on Computer*

Vision and Pattern Recognition, pages 2137–2144, 2006.

[9] H. Joo, Y. Jeong, and I. S. Kweon. Bottom-up segmentation based robust shape matching in the presence of clutter and occlusion. *International Workshop on Advanced Image Technology*, 2009.

[10] A. Opelt, A. Pinz, and A. Zisserman. A boundary fragment model for object detection. *ECCV*, 2:575–588, 2006.

[11] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, pages 888–905, 2000.

[12] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. *ICCV*, 2005.

[13] A. Thayananthan, B. Stenger, P. Torr, and R. Cipolla. Shape context and chamfer maching in cluttered scenes. *CVPR*, pages 127–134, 2003.