Intelligent Object Extraction Algorithm Based on Foreground/Background Classification

Jhing-Fa Wang, Han-Jen Hsu, and Jyun-Sian Li

Department of Electrical Engineering, National Cheng Kung University,
No.1, Ta-Hsueh Road, Tainan, Taiwan
wangjf@csie.ncku.edu.tw, hjhsu@icwang.ee.ncku.edu.tw
http://icwang.ee.ncku.edu.tw

Abstract. In this paper, we propose an intelligent object extraction algorithm based on foreground/background classification. The proposed algorithm can offer the users more friendly interface for object extraction from image without unnecessary steps. After the interactive steps from user (marking the foreground and background parts), the wanted object is extracted from the background automatically. The proposed algorithm processes the input image by watershed to produce the regions. Then, the regions are labeled after marking parts of regions. We also introduce an implementation of hierarchical queues to store the unlabeled regions. The classification of foreground and background will generate the final image with selected object. In our experimental results, the proposed algorithm provides the output image with high efficiency. The wanted object is generated after user marking the foreground and background parts less than one second. In addition, the application of this work also can be used in image synthesis or object removal in other fields of image processing.

Keywords: Object extraction, image editing tool, image segmentation.

1 Introduction

In traditional researches, image segmentation means to detect the boundaries in digital image. However, the boundaries of whole image may contain the boundaries of textures or the inner structure of object in the foreground. In this work, we need to find the contour of wanted object which separates the image into only two parts-foreground and background. Therefore, object extraction can be considered as a binary labeling problem.

The related works include boundary-based methods and region-based methods. For boundary-based methods, they tried to approximate the pre-assigned curves near the object to the object contour, such as intelligent scissor [1], image snapping [2], and Jetstream [3]. The users need to draw the curves which enclose the whole object. This disadvantage of these methods is high complexity in user interface. The user needs to draw the shape of object explicitly.

In order to reduce the redundant steps in user interface, several researches are mentioned to improve the previous works in boundary-based methods. Recently, the re-

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 101 – 110, 2005.

gion-based methods are proposed. The accuracy is increased without as much efforts as that in boundary-based methods. The primitive concerns of this problem are two points. One problem is using less effort to acquire more accurate result. Another problem is the detail information of the object also needed to be preserved. Sun *et al.* proposed a smart method "Poisson Matting" [4] focused on preserving fine features of object, such as hairs and feathers. The object can be cut from original image and pasted on another target image. Rother *et al.* proposed "GrabCut" [5] to achieve foreground extraction using iterated Graph Cuts. The target object is extracted by dragging a square window around the target. Another work "Lazy snapping" [6], is presented by drawing the foreground and background parts at first. The user interface is similar to our system. An example of our proposed algorithm is provided in Fig. 1. The user draws two kinds of lines, green lines for foreground seeds and yellow lines for background seeds, in the input image in Fig. 1(a). The wanted object is acquired easily in the output image shown in Fig. 1(b). Based on foreground and background classification, the regions belonged to wanted object are given as foreground label (F). The other remaining regions are classified to background label (B). However, the object also can be labeled as background by opposite assignments of foreground and background. The obtained result is the input image for object removal.

The organization of this paper is shown as follows. In Section 2, we will describe our algorithms in detail. In Section 3, the experimental results are shown to provide subjective measurement. Finally, the conclusion and future work are provided in Section 4.

(a) (b)

Fig. 1. The example of our proposed algorithm

2 Proposed Intelligent Object Extraction Algorithm

The flow diagram of our proposed algorithm is shown in Fig. 2. At first, the input image in Fig. 2(a) is pre-processed to reduce additive noise. The edge detection and watershed algorithm are applied to produce many small regions shown in Fig. 2(b). Then, the user marks the image by foreground (green lines) and background (yellow lines) markers, respectively, as shown in Fig. 2(c). Once the user marks the image, according to the two sets of markers, two sets of regions are labeled as F and B, respectively. And the non-marked regions are defined as unlabeled.

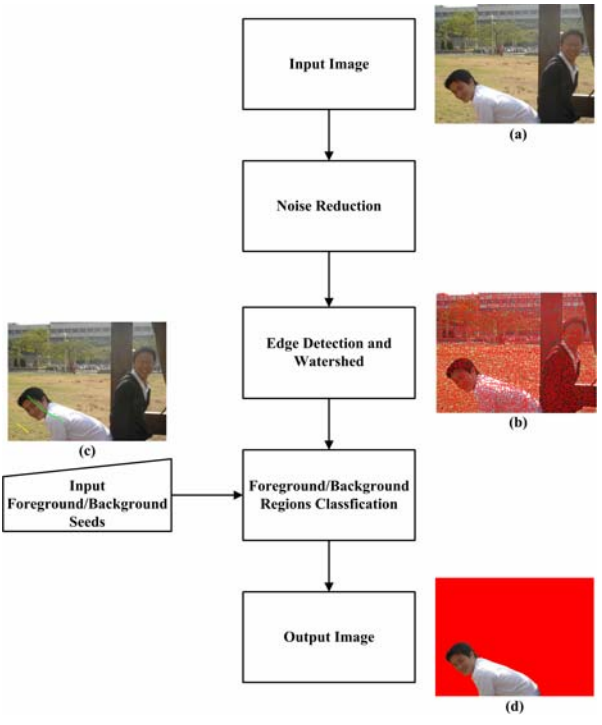


Fig. 2. The flow diagram of our proposed algorithm

After all, the unlabeled regions are classified into foreground or background to generate the final image as shown in Fig. 2(d). The detail description of the proposed algorithm is shown as follows.

2.1 Noise Reduction

The pre-processing is applied to remove the noise which may affect the output result. We use median filter and mean filter to reduce the noise [7]. The median filter is applied first to avoid the operation of averaging at impulse noise in mean filter.

2.2 Edge Detection and Watershed

After noise reduction, edge detection and watershed are used to segment the input image into large number of regions. In order to enhance the luminance and color variation near the boundary of object, we adopt a simple and efficient method which incorporates the morphological gradient of luminance and color in L*a*b* color space from [8]. The gradient value is decided by the erosion and dilation. The color space transformation of RGB to L*a*b* is described from [9]. The RGB values to XYZ(D65) is shown as in (1).

$$\begin{bmatrix} X_{D65} \\ Y_{D65} \\ Z_{D65} \end{bmatrix} = \begin{bmatrix} 0.3935 & 0.3653 & 0.1916 \\ 0.2124 & 0.7011 & 0.0865 \\ 0.0187 & 0.1119 & 0.9582 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

The pixel values in L*a*b* color space transformation is provided as in (2).

$$L^* = \begin{cases} 116\left(\frac{Y}{Y_n}\right)^{1/3} - 16, & \left(\frac{Y}{Y_n}\right) > 0.008856 \\ 903.3\left(\frac{Y}{Y_n}\right), & \left(\frac{Y}{Y_n}\right) \le 0.008856 \end{cases}$$

$$a^* = 500 \times \left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right] \tag{2}$$

$$b^* = 200 \times \left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right]$$

where $f(t) = \begin{cases} t^{1/3}, & t > 0.008856 \\ 7.787t + \frac{16}{116}, & t \le 0.008856 \end{cases}$

We have used D65 as the CIE L*a*b* reference white point. Thus, the constant values X_n, Y_n, and Z_n are equal to 0.9504, 1.0, and 1.0889, respectively.

After edge detection, we use the watershed algorithm from [7] to chunk the image into many regions. We then construct the region adjacency graph (RAG) [10]-[11], which represents the relation between each region and its neighborhood. With this useful step, we can extract the object more efficiently. The RAG is defined as an undirected graph, $G = (V, E)$, where $V = \{1,2,3,\cdots,k\}$ is the set of graph nodes, k is the number of regions obtained from watershed, and $E \subset V \times V$ is the set of graph edges. Each region is represented by a graph node and there exists a graph edge (x, y) if the two graph nodes x and y are adjacent.

A weight of each graph edge stands for the regional color distance of the two adjacent regions, as shown in (3).

$$\mathrm{RCD}(x, y) = \| C(x) - C(y) \| \tag{3}$$

where $C(\bullet)$ denotes the mean color vector of a region in L*a*b color space, $x \in V$, $y \in V$, and $(x, y) \in E$.

2.3 Input Foreground/Background Seeds

The all regions are all unlabeled regions before this step. In this step, we mark the image with foreground and background seeds. More drawing the lines of the foreground and background will lead more exact result. However, our algorithm needs less effort to generate the wanted object. The foreground seeds are selected in green and the background seeds are selected in yellow, respectively. Once the user marks the image, foreground and background marked regions are labeled as F and B, respectively. Therefore, the marked regions are the labeled regions inevitably. And the non-

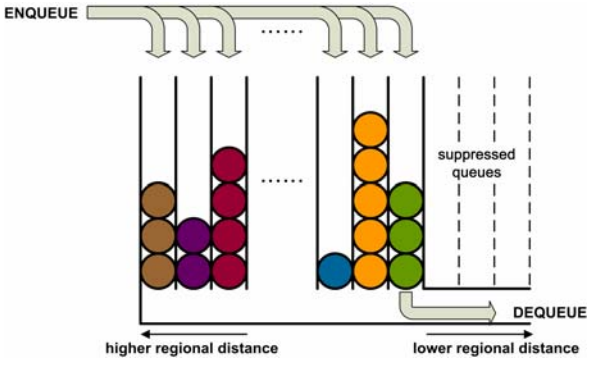


Fig. 3. The notation diagram of hierarchical queues

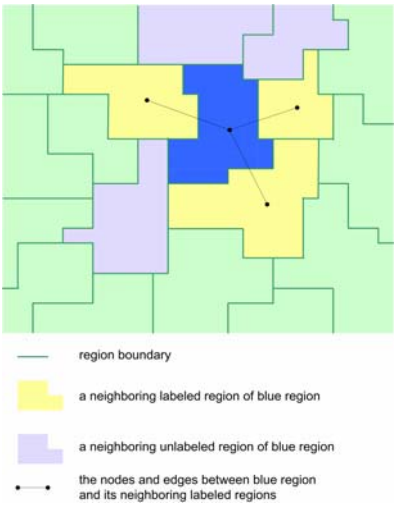


Fig. 4. The generic notation diagram of region labeling

marked regions are regarded as unlabeled regions. In the next step, the remaining non-marked regions are processed by *F*/*B* classification.

2.4 F/B Classification (Region Labeling)

In this Section, the other unlabeled regions which are not belonged to user-defined foreground and background regions are labeled in this Section. *F*/*B* classification (Region labeling) is the most important step in our algorithm, which affects the quality of final result. The pseudocode of region labeling is shown in Table. 1.

In our implementation, we adopt the hierarchical queues shown in Fig. 3 from [12]. In each queue of hierarchical queues, the regions with the same index number are put in the same queue. For example, there are three green balls in the same queue which means three regions with the same index number. This data structure is used in image

segmentation originally. However, this mechanism is also suitable for region labeling. It is composed of two steps: Initialization of the Hierarchical Queues and the Flooding Step. The generic notation diagram is shown in Fig. 4, the blue region is denoted as **A**, in initialization of hierarchical queues. The regional color distance between region **A** and neighboring labeled regions is used to decide which index number of region **A** in hierarchical queues is. On the other hand, the blue region is denoted as **R** in the flooding step. Therefore, the regional color distance between region **R** and neighboring labeled regions is used to decide which label of region **R** is. The detail descriptions of these two steps are shown as below.

Table 1. The pseudocode of region labeling in our proposed algorithm

Step	Description
	Initialization of the Hierarchical Queues:
	For each labeled region **L** For each neighboring region **A** of **L** if **A** is a unlabeled region outside of the hierarchical queues enqueue **A** into the hierarchical queues according to its index number.
	Flooding Step:
	Repeat the following steps until the all hierarchical queues are empty: 1. dequeue a region **R** from the hierarchical queues from the lowest index number; 2. region **R** has at least one labeled region in its neighborhood. It is assigned to the same label with its neighboring labeled region which has the smallest distance from **R**; 3. the neighboring regions of **R** that have not been labeled and are still outside the hierarchical queues are enqueued into the hierarchical queues with the index number not lower than the index number of **R**

Initialization of the Hierarchical Queues

The initialization of region labeling is enqueued the neighboring unlabeled regions of the user marked regions into the hierarchical queues. Each neighboring unlabeled region (**A**) of user marked regions is enqueued into the hierarchical queues according to its index number as in (4). The index number of a region is denoted as q.

$$q = floor(\min_{m}(\mathrm{RCD}(\mathbf{R_i}, \mathbf{A})) + 0.5) \qquad where\ i = 1\ to\ m \tag{4}$$

where $(\mathbf{R_i}, \mathbf{A}) \in E$ and m is the number of the labeled regions adjacent to **A**.

Flooding step

After the initialization of the hierarchical, we start to dequeue the regions from the queue with the lowest index number. Any queue which is empty will be suppressed

and no longer be enqueued. The hierarchical queues are processed until all the queues are empty. The flooding step is similar to the initialization of the hierarchical queues. Each region in the hierarchical queues is dequeued and compared the similarity with the neighboring labeled regions. The labeled regions may contain the user marked regions and the regions which are already labeled in this step. As shown in Table. 1, after we dequeue a region **R** from the hierarchical queues, we have to find a neighboring labeled region of **R** which is most similar to **R** as in (5).

$$\mathbf{R}^* = \arg\min_{\mathbf{LR}} \mathrm{RCD}(\mathbf{LR}, \mathbf{R}), \quad \text{where } \begin{cases} \mathbf{LR} \in N(\mathbf{R}) \\ \mathbf{R}^* \in N(\mathbf{R}) \end{cases} \tag{5}$$

$\mathbf{R}^*$ is the most similar region to **R**, and N(**R**) denotes the neighboring labeled region of **R**. Then, **R** is assigned to the same label as $\mathbf{R}^*$.

On the other hand, the neighboring unlabeled regions (**B**) of region **R** which is still out of the hierarchical queues is enqueued into the q-th queue as in (6).

$$t = floor(\min_n (RCD(\mathbf{R_j}, \mathbf{B})) + 0.5) \quad where\ j = 1\ to\ n$$
$$\begin{cases} q = t, & \text{if } t \geq z \\ q = z, & otherwise \end{cases} \quad \text{where } z \text{ is the index of } \mathbf{R} \tag{6}$$

where $\mathbf{R_j}$ is labeled, $(\mathbf{R_j}, \mathbf{B}) \in E$, and n is the number of the labeled regions adjacent to **B.**

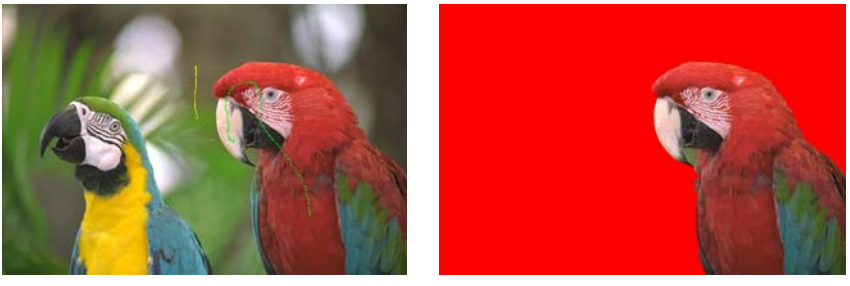
After this, if the queue with the same index number of **R** is empty, it will be suppressed. In the later processing, if we obtain a region which will be enqueued into the suppressed queue, we put the regions into the lowest un-suppressed queue. Finally, until the whole regions are labeled (the hierarchical queues is empty), the wanted objects are extracted from the input image.

3 Experimental Results

We provide some experimental results in this Section. The experimental results show our proposed algorithm can produce excellent output images. The test image from Kodak test images "parrot" is chosen by the user shown in Fig. 5(a). After our proposed algorithm, the red parrot is extracted from input image shown in Fig. 5(b). The use only needs to draw little lines of foreground and background. Another example is provided in Fig. 6. The wanted lighthouse is extracted from the image shown in Fig. 6(b). The test image from [13] in Fig. 7 shows three chairs on the grass. The red chair is chosen to be the target object. We draw two points in green and one line in yellow as background. The middle red chair is extracted from the image shown in Fig. 7(b).

Besides of natural images, we also use indoor image to test our proposed algorithm. In Fig. 8(a), we use the test image from [14]. The operation of marking the foreground and background are reversed to generate the opposite result. The wanted object is selected as background which we want to remove from the image. In another work, "object removal", the result shown in Fig. 8(b) is used to be the input image.

We take another photograph as shown in Fig. 9. The left man is chosen to be the object which we want to remove. The output result is shown in Fig. 9(b). The computation time analysis is given in Table. 2. The computation time costs most time in edge detection and watershed algorithm. The final result is obtained immediately after the user drawing. The simulation environment is on AMD 1.7G with 1GB of RAM and implemented in C++.



(a) (b)

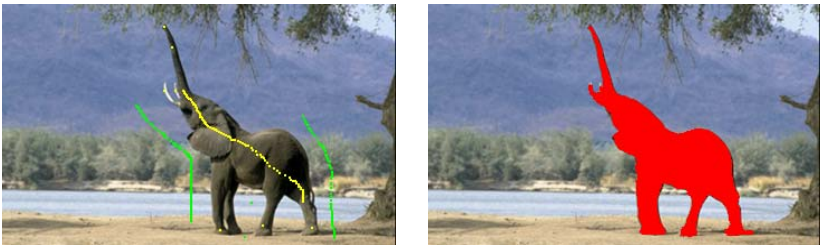
Fig. 5. The experimental result for object extraction

(a) (b)

Fig. 6. The experimental result for object extraction

(a) (b)

Fig. 7. The experimental result for object extraction



(a) (b)

Fig. 8. The experimental result for object removal

(a) (b)

Fig. 9. The experimental result for object removal

Table. 2. The computation time analysis of our proposed algorithm

Image no.	Image Resolution	Noise Reduction	Edge Detection	Watershed	F/B Classification
1	768*512	1.390 sec	5.469 sec	2.250 sec	0.031 sec
6	768*512	1.421 sec	5.406 sec	2.172 sec	0.016 sec
7	768*512	1.406 sec	5.422 sec	2.234 sec	0.032 sec
8	352*234	0.282 sec	1.141 sec	0.453 sec	0.001 sec
9	450*339	0.500 sec	2.078 sec	0.813 sec	0.001 sec
10	352*211	0.266 sec	1.031 sec	0.422 sec	0.001 sec
11	352*264	0.329 sec	1.266 sec	0.484 sec	0.001 sec

4 Conclusion and Future Work

In this work, we propose an intelligent object extraction algorithm based on foreground/background classification. The regions after watershed are classified into

foreground and background. The hierarchical queues are implemented to increase the efficiency. The user interface is easy to use, even the general users without the tips in image processing can acquire the wanted object. The proposed algorithm can produce good results in real-time.

Our future work is preserving the fine features on the boundary of the object. Currently, the fine features of the object need to be selected as foreground to ensure the excellent result. We are going to improve the weakness of our proposed algorithm. To develop the smart camera system, we also plan to integrate with object removal from our previous work.

References

1. Mortensen, E. N., Barrett, W. A.: Toboggan-based intelligent scissors with a four parameter edge model. In *Proceedings of CVPR'99*. (1999)
2. Gleicher, M.: Image snapping. In *Proceedings of ACM SIGGRAPH'95*. (1995)
3. Perez, P., Blake, A., Gangent, M.: Jetstream: Probabilistic contour extraction with particles. In *Proceedings of ICCV 2001*. (2001)
4. Jian Sun, Jiaya Jia, Chi-Keung Tang, Heung-Yeung Shum: Poisson matting. ACM Transactions on Graphics (TOG), Volume 23 Issue 3 (2004)
5. Carsten Rother, Vladimir Kolmogorov, Andrew Blake: "GrabCut": interactive foreground extraction using iterated graph cuts. ACM Transactions on Graphics (TOG), Volume 23 Issue 3 (2004)
6. Yin Li, Jian Sun, Chi-Keung Tang, Heung-Yeung Shum: Lazy snapping. August ACM Transactions on Graphics (TOG), Volume 23 Issue 3 (2004)
7. Rafael C. Gonzalez, Richard E. Woods.: Digital Image Processing. Prentice-Hall, Inc. (2002)
8. Hai Gao, Wan-Chi Siu, Chao-Huan Hou: Improved Techniques for Automatic Image Segmentation. Circuits and Systems for Video Technology, IEEE Trans. on Volume 11, Issue 12, pp. 1273 - 1280 (2001)
9. J. M. Kasson, W. Plouffe: An Analysis of Selected Computer Interchange Color Spaces. ACM Transactions on Graphics, Vol. 11, No. 4, October, Pages 373-405 (1992)
10. D. Ballard and C. Brown: Computer Vision. Englewood Cliffs, NJ: Prentice-Hall (1982)
11. X. Wu: Adaptive split-and merge segmentation based on piecewise least-square approximation. IEEE Trans. Pattern Analysis Machine Intelligence Vol. 15, Page 808-815 (1993)
12. Meyer, F.: Color Image Segmentation. Image Processing and its Applications, International Conference 303 - 306 (1992)
13. I. Drori, D. Cohen-Or, H. Yeshurun: Fragment-based image completion, in ACM Trans. Graphics (TOG), vol.22, no. 3, pp. 303-312 (2003)
14. J. Jia, and C. K. Tang: Inference of segmented color and texture description by tensor voting. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 26, June (2004)