Journal of Computer and Communications, 2018, 6, 48-55
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Hierarchical Grab Cut Image Segmentation Algorithm

Lan-Rong Dung[1], Yao-Ming Yang[1], Yin-Yi Wu[2]

[1]Institute of Electrical and Control Engineering, National Chiao Tung University, Taiwan
[2]Chung-Shan Institute of Science Technology, Taiwan
Email: lennon@faculty.nctu.edu.tw

How to cite this paper: Dung, L.-R., Yang, Y.-M. and Wu, Y.-Y. (2018) A Hierarchical Grab Cut Image Segmentation Algorithm. *Journal of Computer and Communications*, **6**, 48-55.
https://doi.org/10.4236/jcc.2018.62005

Received: November 27, 2017
Accepted: February 25, 2018
Published: February 28, 2018

Abstract

This paper aims to speed up a segmentation algorithm "Grab Cut" by separating the process of segmentation into hierarchical steps. The Grab Cut algorithm segments images by means of the color clustering concept and the process requires a lot of iteration for it to get converged. Therefore, it is a time-consuming process which we are interested in improving this process. In this study, we adopt the idea of hierarchical processing. The first step is to compute at low resolution to make the iteration much faster, and the second step use the result of the first step to carry on iteration at original resolution so that the total execution time can be reduced. Specifically speaking, segmentation of a low resolution image will lead to high-speed and similar-segmentation result to the segmentation at original resolution. Hence, once the iterations at low resolution have converged, we can utilize the parameters of segmentation result to initialize the next segmentation on original resolution. This way, the number of iteration of segmentation at original resolution will be reduced through the initialization of those parameters. Since the execution time of low resolution images is relatively short, the total hierarchical execution time will be reduced consequently. Also, we made a comparison among the four methods of reduction on image resolution. Finally, we found that reducing the number of basins by "Median Filter" resulted in best segmentation speed.

Keywords

Image Segmentation, Image Processing, Median Filter, Clustering

1. Introduction

The efficiency of interactive foreground/background segmentation is practically

important for image editing. There are two kinds of segmentation methods: "Auto" [1] and "Semi-Auto". Auto segmentation processes image without any other information. Therefore, the results are not reliable enough.

On the other hand, Semi-Auto segmentation added the user information, for example, a desired region assigned by user, will be more accurate and faster than auto segmentation. There is a powerful, popular semi-auto segmentation algorithm which is called "Graph-Cut" [2] [3]. Users only need to plot some points on the image for deciding which is background or foreground. Then the algorithm will segment the image with the user information automatically. Recently, an algorithm, which improved graphcut, is called "Grab Cut" [4]. Grab cut adds one more step in the processing. Before assigning the points, users need to mark two corners of a rectangle so that the first segmentation result will show up in the rectangle. Therefore, users don't have to specify so many points.

Our work focuses on the rectangle segmentation stage. Since the rectangle segmentation utilizes the clustering concept [5] [6], the process will need a lot of iterations to achieve converging. As a result, we provided the process that makes the original process faster. Specifically speaking, there are two steps of the process. The first step is reducing the resolution of the image and doing the segmentation which has the similar result as the original segmentation. Second, we take the converged result parameters of the first stage, then go back to the original resolution and use the parameters to do segmentation iterations. Since the result from low resolution stage is similar to original segmentation and will help the segmentation on original resolution to get converged with lesser iterations. Still the speed under low resolution process is faster than the original process. So, we will make the grand process faster than the traditional method.

The practical measure is about "Lazy Snapping" [7] which mainly discusses a concept of considering a basin produced by the watershed algorithm [8] [9] as a pixel. This way, the number of pixels will decrease heavily and the speed will improve greatly. Because a basin is now considered as a pixel, we use the smooth filter to reduce basins [10] as the way of decreasing the resolution. Additionally, we experiment with four methods of decreasing resolution which are median filter, mean filter, gaussian filter, and down sample. The experiment result is that the median filter is the best choice to our algorithm.

2. Background

2.1. Gaussian Mixture Model

The implementation of grab cut segmentation is based on "Clustering" and "Max-Flow Min-Cut Theorem". Clutering is a concept and also an algorithm that separates multi-dimension data into several groups. Gaussain Mixture Mo-del (GMM) [11] is one of the existing clustering algorithms. The concept of GMM is to distribute several gaussian probability models to every data point and find out the model with the highest probability for each point. As a result, every data point belongs to one gaussian model. This way, we may distribute all the

data points to numbers of group.

The equation form of the probability produced by the k-th gaussian model (component) is (1).

$$r(i,k)=\frac{\pi_k N(x_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_i \mid u_j, \Sigma_j)} \tag{1}$$

Now we have learned each data point belongs to the model with the highest probability. Therefore, we can calculate the mean value and the variance (matrix) of every gaussian probability model (component), the results are (2) and (3). Then we do the steps above until the results to get converged.

$$\mu_k=\frac{1}{N_k}\sum_{i=1}^{N} r(i,k)\,x_i \tag{2}$$

$$\Sigma_k=\frac{1}{N_k}\sum_{i=1}^{N} r(i,k)(x_i-\mu_k)(x_i-\mu_k)^{\mathrm{T}} \tag{3}$$

2.2. Max-Flow Min-Cut Theorem

If we build a flow network as Figure 1, we can see there are sources, "S", and a sinks "T". The denominator means the capacity of the flow. The numerator means the amount of current flow. All the flow cannot be disappeared; they must be the same amount as the converge flow. In a flow network, there can be different flow path from S to T, as Figure 1. So, there are different path to achieve max flow. If we separate two regions for S and T, it will be a "cut", as Figure 2. Although the capacity of cuts differs from each other, the real flow

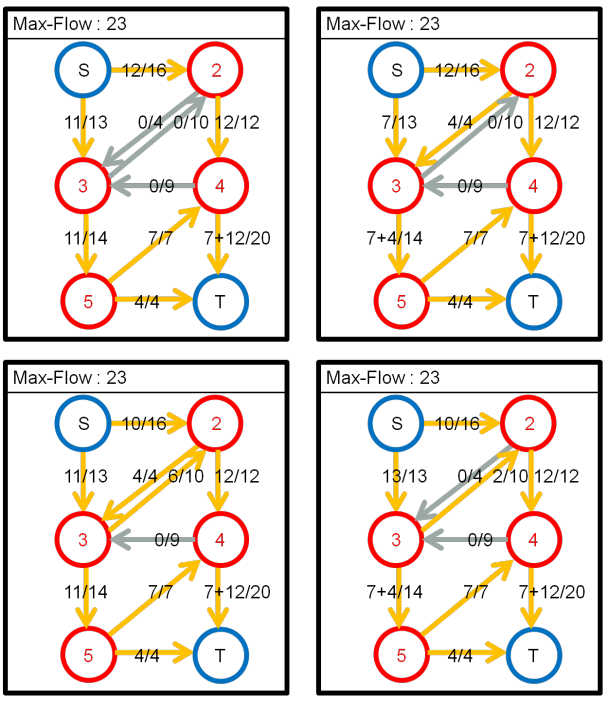


Figure 1. Max-flows of different flow networks.

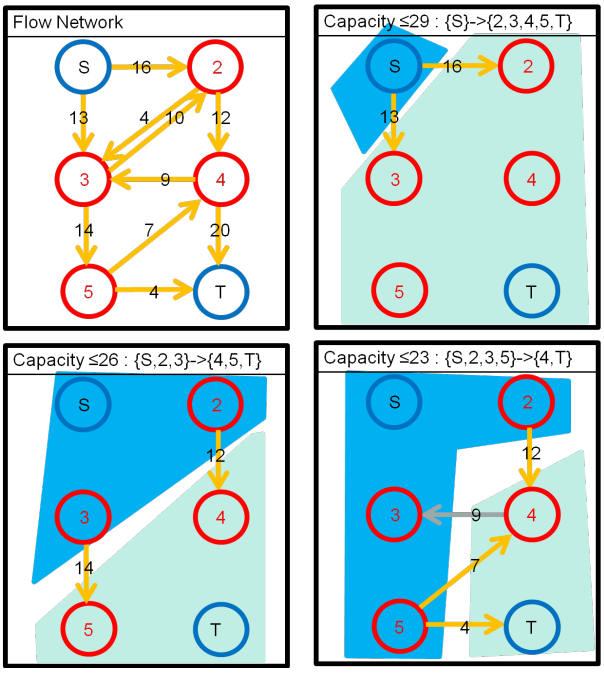


Figure 2. Different bottle necks of a network.

amount of them are all the same. Because current flow value must be identical to the previous flow, or the flow will disappear. We can find out that in **Figure 2**, the capacities are 29, 26, 23, but the real flow amount must be 23 which just equal to the max flow amount in **Figure 1**. The reason is that the max flow is constrained to the bottle necks of a flow network, and the cut with capacity 23 is so called "Min-cut". As a result, if we want to find the max flow value, we can find the min cut and its capacity; if we want to find a min cut capacity, we can find the max flow value.

If we image there is a chessboard, the cross point is the pixel, we can apply this theorem [12] to images.

2.3. Grab-Cut Algorithm

For a segmentation algorithm, the cost function is always the necessity, they are as Equation (4)-(6), where k_n means the n-th pixel belonging to k-th component of GMMs, and α_n only possess value 0 and 1 which means the n-th pixel belonging to back/foreground. θ are the parameters of GMMs; they are weighting coefficients, mean values, and covariance matrix. z_n is the pixel value of n-th pixel. Moreover, $COV(\alpha_n, k_n)$ refers to the covariance matrix of the component that belongs to back/foreground and corresponds to the n-th pixel.

$$U(\alpha, k, \theta, z) = \sum_n D(\alpha_n, k_n, \theta, z_n) \tag{4}$$

$$\begin{aligned} D(\alpha_n, k_n, \theta, z_n) = &-\log\left[\pi(\alpha_n, k_n)\right] + \frac{1}{2}\log\left\{\det\left[COV(\alpha_n, k_n)\right]\right\} \\ &+ \frac{1}{2}\left[z_n - \mu(\alpha_n, k_n)\right]^{\mathrm{T}}\left[COV(\alpha_n, k_n)\right]^{-1}\left[z_n - \mu(\alpha_n, k_n)\right] \end{aligned} \tag{5}$$

$$\theta = \{\pi(\alpha,k), \mu(\alpha,k), COV(\alpha,k), \alpha = 0,1, k = 1,\cdots,K\} \quad (6)$$

We can see that the function D in (4)-(5) is a muti-dimension gaussian probility distribution. Therefore, function D means how possible a pixel belongs to back/foreground. The total cost function is as (7) which includes function U and V. The function V is written as (8), where

$f(\alpha) = 0$ for $\alpha_n = \alpha_m$, and 1 for $\alpha_n \neq \alpha_m$.

$$E(\alpha,k,\theta,z) = U(\alpha,k,\theta,z) + V(\alpha,z) \quad (7)$$

$$V(\alpha,z) = r\sum_{(m,n)\in C} f(\alpha)\exp\left(-\beta\|z_m - z_n\|^2\right) \quad \text{where } f(\alpha) = \begin{cases} 0, & \alpha_n = \alpha_m \\ 1, & \alpha_n \neq \alpha_m \end{cases} \quad (8)$$

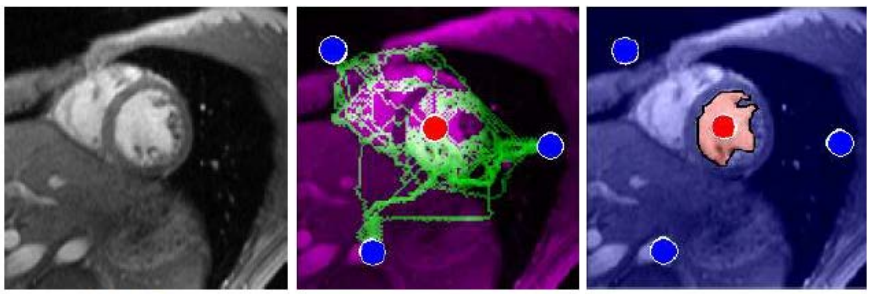
If two neighbor pixels have large difference in value, they can be considered as background and foreground with low cost. If the values of two neighbor pixels are close, the cost of assigning them different back/foreground will be large. The segmentations executing with max-flow min-cut theorem and cost function (7) are as Figure 3.

In the end, the process of grab cut can be divided into four steps. First, user marks two corner of a rectangle to ensure a segmentation region, and the pixels inside will be used to build foreground GMMs while the pixels outside are used to build background GMMs. Second, these GMMs are used to form the cost function U, and the segmentation will begin. Third, the segmentation result forms new back/foreground GMMs. We need to check if the new GMMs are identical to the old ones. If they didn't change, it means that the segmentation is converged, the program will go to step four. If they did change, the program will back to step two until it converges. The fourth step is using graph cut to mark the imperfect region to complete the segmentation.

3. The Hierarchical Speeding up Method

3.1. Lazy Snapping

Since the computation complexity of graph cut is $O(mn^2)$, which n is the pixel number and m is the flow number. It is a third power of pixel number. So, [7] has proposed an algorithm that uses watershed segmentation and consider a



(a) (b) (c)

Figure 3. The results of (a) Original image (b) the maximum flow and (c) minimum cut algorithms.

basin as a pixel. This way, the new pixel number reduces rapidly.

Our work is based on lazy snapping and [10]. [10] has proposed a method that is using median filter to smooth the image and the number of basins will be decreased. Therefore, we can consider different degree of smooth as different image resolution, due to the reduction of basins.

3.2. Speeding up Method

Our proposed method is as **Figure 4**; the first step is to segment image for 2 iterations so that the result will be similar to the result of traditional method for sure. We can see **Figure 5**, for all experiment patterns, the energy (cost) at least decreased the amount of getting converged by 80%.

The second step is to reduce resolution by using smooth filter, then use the GMMs parameters of the first step to continue segmentation. As a result, the GMMs parameters of the second step are closer to those converged parameters.

The third step is to go back to the original image and use the final parameters of step two to carry out a new segmentation. Although the process is more complicated, the execution time decreases due to the replacing in part of entire process by high speed computation in low resolution stage.

4. Simulation and Comparisons

We pick four approaches to accomplish the proposed speeding up algorithm. They are median filter, mean filter, Gaussian filter, and down sampling. By using these filters and down sampling, we can lower the number of basins so that the speed of execution on these images is increased.

We use a method called "normalized ranking" with four methods to find out which method has the best execution time. The normalized ranking is to set the longest execution time as 1, the others are divided by the longest one. Therefore, we will get a grand score by summing scores on all patterns. Obviously, the largest grand score means the longest execution time. In our simulation, the median filter has the best grand score among four methods. The simulation platform is a personal computer with Intel Core i3-530, 2.93 GHz, and 4 G byte DRAM.

Moreover, in **Table 1**, we can see the average speeding up rate for various mask size of median filter. The number "9-1" means the segmentation was processed on an image pretreated by a 9×9 mask median filter, and then back to the original image. We can find out that the speeding up rate is from 20% to 30% for median filter of those sizes.

5. Conclusion

Image segmentation is such a highly developed field and so many researchers are still trying to make some improvement. Our improvement is mainly about speeding up and we used four methods to make the segmentation faster. Finally, we find out that median filter has the best performance among four methods. In the future, if there exists any need is strong to real-time while with less emphasis

o n

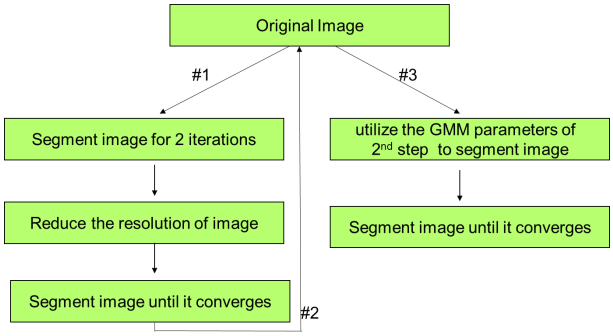


Figure 4. The flow chart of speeding up method.

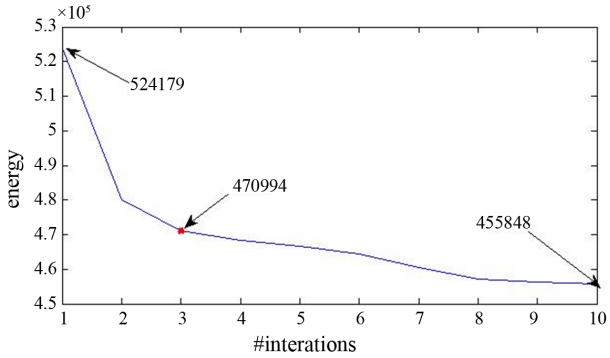


Figure 5. The energy decreasing plot.

Table 1. Speeding up rate on different mask size of median filter.

Mask size	Speeding up rate
9-1	27.6724%
7-1	28.6286%
5-1	27.8606%
3-1	20.4462%
9-5-1	29.5481%
9-3-1	26.7035%

quality of image, such as object tracking, our proposed algorithm may be useful for these kind of needs.

Acknowledgements

The authors would like to thank the Editor and the referee for their comments. This work was supported in part by the National Science Council, Taiwan, under Grant No.98-2221-E-009-138.

References

[1] Han, D., Li, W., Lu, X., Wang, T. and Wang, Y. (2006) Automatic Segmentation Based on AdaBoost Learning and Graph-Cuts. *ICIAR*06, Póvoa de Varzim, 18-20

September 2006, 215-225. https://doi.org/10.1007/11867586_21

[2] Boykov, Y. and Funka-Lea, G. (2006) Graph Cuts and Efficient N-D Image Segmentation. *International Journal of Computer Vision,* **70**, 109-131. https://doi.org/10.1007/s11263-006-7934-5

[3] Boykov, Y. and Jolly, M.P. (2001) Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. *Proceeding of ICCV*, **1**, 105-112. https://doi.org/10.1109/ICCV.2001.937505

[4] Rother, C., Kolmogorov, V. and Blake, A. (2004) GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Transactions on Graphics*, **23**, 309-314. https://doi.org/10.1145/1015706.1015720

[5] Forgy, E. (1965) Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classifications. *Journal of the International Biometric Society*, **21**, 768-769. http://blog.pluskid.org/?cat=4&paged=3

[6] Li, Y., Sun, J., Tang, C.K. and Shum, H.Y. (2004) Lazy Snapping. *ACM Transactions on Graphics*, **23**, 303-308. https://doi.org/10.1145/1015706.1015719

[7] Khalifa, F.A. and Alouani, A.T. (2009) Survey of Watershed Modeling and Sensor Data Fusion. *Proc. SSST,* March 2009, 39-43. https://doi.org/10.1109/SSST.2009.4806786

[8] Hong, M.C. (2000) Muti-Resolution and JND Based Image Segmentation Algorithm. Master's Thesis, EE Department, NYUST.

[9] Peng, B., Zhang, L. and Zhang, D. (2001) Image Segmentation by Iterated Region Merging with Localized Graph Cuts. *Pattern Recognition*, **44**, 2527-2538. https://doi.org/10.1016/j.patcog.2011.03.024

[10] Bradley, P. and Fayyad, U. (1998) Refining Initial Points for K-Means Clustering. *Proc.* 15*th ICML*, **66**, 91-99.

[11] Wang, J.D., Lee, J.G. and Zhang, C.G. (2003) Kernel GMM and Its Application to Image Binarization. *Proc. ICME,* **1**, 533.

[12] Dempster, A.P., Laird, N.M. and Rubin, D.B. (1997) Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society B*, **39**, 1-38. http://ccckmit.wikidot.com/st:em