Black and White Rendering in Complex Background

KUI CHEN[†1] CLAUS ARANHA[†2]
HITOSH KANOH[†2]

In this paper, we present an algorithm for abstracting images with complex background into monochrome art images. Mould et al. rendered monochrome images by inputting photos which have only one object and have no complex background. To render images with complex background, we firstly segment the input image and extract the foreground from the background automatically. Then the belief propagation algorithm based on Hidden Markov Model is used to render the foreground objects and Otsu method is used to render the background. Finally, we combine the results and obtain the black and white image as the output. Comparing with the results gained with only one rendering method, by using different method to render the foreground and the background, we get more details of the foreground objects and stand out them from the background.

1. Introduction

There are many scenes where an input color image should be turned into a black and white art image, especially in documentaries and comics. This paper addresses the problem of rendering black and white art image from a color photo which has complex background. The input image not only has a single object, but also has complex background. A single object image and image with complex background are shown in Figure 1.

a b

Figure 1. The Comparison of Simple Image and Image with Complex Background.

In Figure 1, image (a) has a single object in the middle of it, so the background can be ignored. On the other hand, (b) has not only a single object but a little complex background which cannot be ignored in the process. Here, we define “complex background” as below: the foreground object does not dominant the whole image while the background occupies a large part of input image and cannot be ignored in the process.

The aims in this paper include: segmenting foreground objects from the background, making the foreground objects stand out from the background, and drawing the details of foreground objects carefully.

This paper is organized as follow. In section 2, we review some scientific works related to our goal. We also point out the problems in these works. Next, in section 3, we describe the details of proposed method. Finally, we show results of applying proposed method to different input images and give some commentary on the effectiveness of this method in section 4.

†1 Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba.
†2 Division of Information Engineering, Faculty of Engineering, Information and Systems, University of Tsukuba.

2. Overview

2.1 Previous approach

The simplest method used to convert images to binary images is setting threshold. First of all, the color image is converted into a grayscale image in which one pixel is 8 bits. Then a threshold value between 0 and 255 is selected and each pixel’s gray level is compared with it. The pixels whose gray level is smaller than the threshold value are given black, otherwise white. This method is very intuitive but does not respect the details in the input image.

Setting threshold is not an acceptable method in most application scenes. However, some improvements can be done to grab the details of the input image and produce a better result. Xu [1] proposed an optimization-based method called “artistic threshold”. Firstly, segmentation is applied to a source image. Then, based on the result of segmentation, a region adjacency graph is generated and an energy function that measures the quality of different black and white colorings of the segment is established. Finally, the algorithm searches for a black and white assignment that minimizes the energy function. The optimization is controlled via a set of intuitive user-selected weights that can produce distinct results.

Hidden Markov Model can also be used to create a black and white image. Mould [2] presented a Hidden Markov Model-based method. First of all, the gray levels of pixels is regarded as observations of Hidden Markov Model and two labels, 0 and 1, are defined as states which represent black and white respectively. For each given pixel p in the foreground object, we associate a label l_p with it. A label can be either black or white, and the cost of each pixel is defined as:

$$c(l_p) = \begin{cases} -1 & l_p = black \\ 1 & l_p = white \end{cases} \quad (1)$$

Then, let the mean intensity of the image be $\overline{v}$, the intensity of each pixel be v . The global energy component for a pixel is determined by comparing $\overline{v}$ and v :

$$\delta_G = \frac{v - \overline{v}}{M + 1} \quad (2)$$

Where M is the possible maximum pixel intensity, in our case,

it is 255. Finally the global energy cost of assigning a pixel with label l_p is defined as:

$$G(l_p) = -\ln(0.5 + 0.5 \cdot c(l_p) \cdot sign(\delta_G) \cdot |\delta_G|^{\gamma}) \quad (3)$$

The function $sign(\delta_G)$ is to ensure the sign (positive or negative) of δ_G retained after raising to a power.

Similarly, defining the mean intensity of 8 neighbors of each pixel as $\overline{v}_8$, then the local energy component for a pixel is defined as:

$$\delta_L = \frac{v - \overline{v}_8}{M + 1} \quad (4)$$

The local energy cost of assigning a pixel with label l_p is defined as:

$$L(l_p) = -\ln(0.5 + 0.5 \cdot c(l_p) \cdot sign(\delta_L) \cdot |\delta_L|^{\gamma}) \quad (5)$$

Now the data cost function is defined by combining 3 and5:

$$D(l_p) = (1 - \alpha)G(l_p) + \alpha L(l_p) \quad (6)$$

High value of α keeps the local details of input image and on the other hand, the low value of α creates larger segmentation of one color.

The other part of energy function is smoothness function, which indicates the cost between two neighbour pixels p and q. Let g_p and g_q represent the gradient magnitude at pixels p and q respectively, and GM is the maximum gradient magnitude value in the image. Similar to the data cost function, we define smoothness function V_{pq} as:

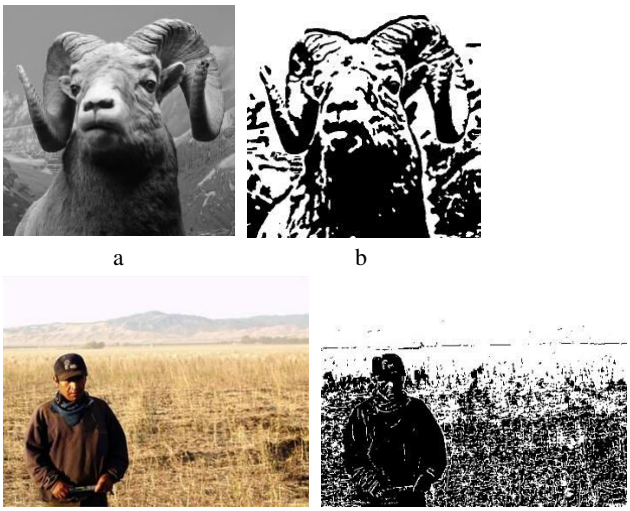
$$\delta_V(p,q) = 1 - \frac{\max(g_p, g_q)}{GM} \quad (7)$$

$$V_{pq}(l_p, l_q) = -\ln(0.5 + 0.5 \cdot c(l_p, l_q) \cdot \delta_V(p,q)^{\gamma V}) \quad (8)$$

Smoothness function V_{pq} punishes neighbour pixels assigned to different labels with a high energy value.

Once the energy functions are defined, then Loopy Belief Propagation algorithm is used to minimize the Gibbs energy function. Finally, if Loopy Belief Propagation converges, the label, 0 or 1, is assigned to each pixel.

Both Xu and Mould's methods can generate nice results but the shortcoming is also clear. These algorithms proposed above can only draw black and white images from photographs which contain only single item in the middle of the image and have no complex background. If the input image has complex background, the detail in result becomes worse because the foreground object is affected by the background seriously. (See Figure 2.)



a b c d

Figure 2. The Results of Simple Image and Image with Complex Background Using Mould's Method. (a) and (c) are the input images, (b) and (d) are the result of Mould's method.

2.2 Related work

To generate a black and white image from a photograph which has complex background, it is important to recognize the foreground objects from the background.

"Grab cut" [3] is a very famous algorithm that can extract the foreground objects from the background. Generally, unlike "Graph cut" [4], users only need to drag a rectangular loosely around an object. Based on the hint given by users, color data is modeled by Gaussian Mixture Model (GMM) and an iteration is done on an energy function. The iteration terminates when the minimum value of the energy function is found. The result is a high performance of accurate segmentation of object from background.

However, instead of dragging a rectangular around an object manually, we can use saliency map based on histogram to do the same thing automatically [5]. First of all, we use the method proposed by Cheng [5] to generate the saliency map of input image. Then, GrabCut based on the generated saliency map is applied to extract the foreground object from the background.

Once the foreground object is separated from the background, we can render the foreground object and the background respectively. Instead of rendering the whole image with only one method, we use two different algorithms to obtain the result.

Otsu algorithm [6] is used to automatically perform clustering-based image thresholding and reduces a grayscale image to a binary image. The algorithm assumes that the image contains two classes of pixels following bi-model histogram, it then calculates the optimum threshold separating the two classes so that their inner-class variance is maximal. Otsu algorithm is very fast and can be complemented easily. However, the result is a little rough, and many details are ignored. Another shortcoming of Otsu algorithm is that it cannot be adjusted by user or parameters because it is based only on histogram of input image.

On the other hand, Belief Propagation [7], which is a method

based on Hidden Markov Model, can also be used to generate black and white image. By defining the Gibbs energy function and adjusting the parameters, more details can be preserved. Belief propagation is not as fast as Otsu algorithm, but can obtain better result. However, if it is used on the whole input image, the background will be rendered excessively and it is difficult to make the foreground object stand out.

In our implementation, according to the result from GrabCut, we apply belief propagation on the foreground object so that the details of foreground object can be preserved, and Otsu algorithm is used to obtain the background result. Finally, the two results are combined and the final result is output.

The novelty of our approach lies first in the handling of rendering. Unlike the previously mentioned methods which usually render the whole image with only one algorithm, we use two different methods to render the foreground object and background respectively so that the foreground details are preserved and background details are suppressed. The result is that the foreground object is stood out. Secondly, our method is automatic, and no intervention from user need after inputting the original images.

3. Proposed method

The process of the proposed method is shown in Figure 3.

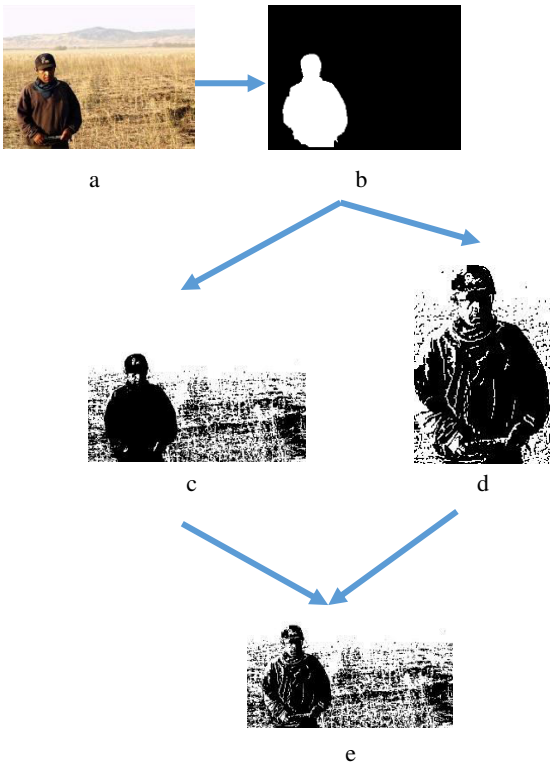


Figure 3. The Process of Proposed Method.

First of all, we calculate the saliency map of input image (a) and then based on the saliency map, GrabCut is used to separate the foreground object from the background (b). Secondly, Otsu method is used on the whole input image and belief propagation is used on the foreground object only (c and d). Finally, we combine the results of foreground a background together to produce the final result (e).

3.1 Separating foreground object from background

Given a source color image, we use GrabCut based on saliency map proposed by [3] and [5]. The foreground object can be separated from background automatically by this method. The result gained by this step is a mask of input image, where the foreground is represented with white and the background is indicated with black. The result is shown in Figure 4.

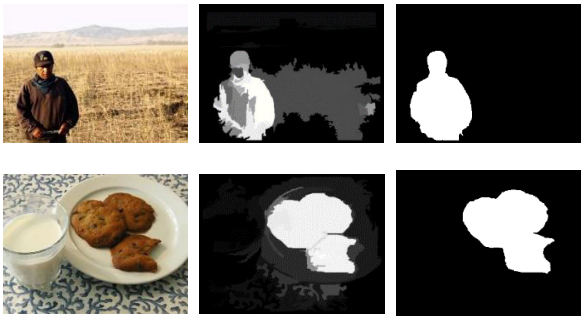


Figure 4. GrabCut based on Saliency Map. (a) Input images. (b) Saliency map. (c) The results generated from GrabCut based on the saliency map (b).

3.2 Rendering background with Otsu method

Otsu method is used to rendering the background of the input image. First, the source color image is converted into a grayscale image. Then, Otsu method is applied on the whole grayscale image. After that, using the mask generated in section 2.1, the background is cut out from the result of Otsu method. We also try to apply Otsu method on the foreground object only but find no improvement. This process is shown in Figure 5.

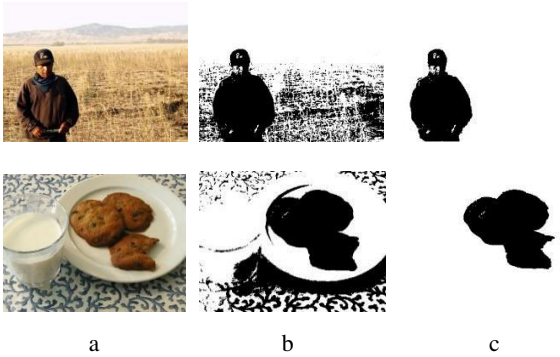


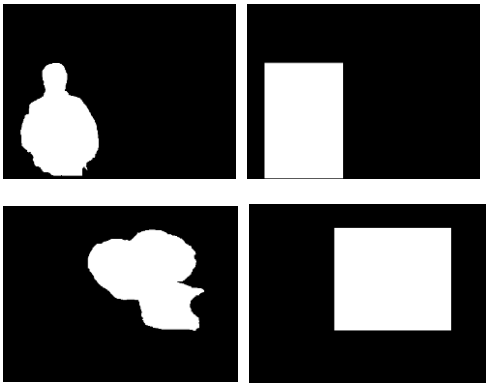
Figure 5. Otsu Method Using on the Whole Image and on the Mask Only. (a) Input image. (b) The output image generated by Otsu method on the whole image. (c) The output image generated by Otsu method on the mask only. There is no improvement for the details of foreground objects. The masks used here are identical to the ones in Figure 4.

3.3 Rendering foreground with loopy belief propagation

The foreground object is rendered by loopy belief propagation

(LBP). The input image is also converted into grayscale image first. Then the mask is used to cut the foreground object out from background. We designed the energy function based on [2], and do some improvements.

First of all, the mask generated in section 3.1 is improved so that it is suitable to be applied in LBP. Because LBP is applied on 2D Markov Model, we use the minimum enclosing rectangular of originally mask as the mask of LBP. This is shown in Figure 6



a b

Figure 6. The Masks Used in Belief Propagation Algorithm. (a) The original mask produced by GrabCut. (b) The masks used in LBP.

Then, in our implementation, we found that the ln function was easy to underflow. To avoid this, we modified the implementation, so that 2 is added to all ln function to avoid underflow:

$$G(l_p) = -\ln(0.5 + 0.5 \cdot c(l_p) \cdot sign(\delta_G) \cdot |\delta_G|^{\gamma} + 2) \quad (9)$$

$$L(l_p) = -\ln(0.5 + 0.5 \cdot c(l_p) \cdot sign(\delta_L) \cdot |\delta_L|^{\gamma} + 2) \quad (10)$$

$$V_{pq}(l_p, l_q) = -\ln(0.5 + 0.5 \cdot c(l_p, l_q) \cdot \delta_V(p,q)^{\gamma V} + 2) \quad (11)$$

Adding 1 is not a good idea because many function values will turn to 0. The functions 9 and 10 punish a label that falls on the opposite side of the mean intensity as the original pixel intensity. The parameter γ is used to control the magnitude of energy. The higher γ, the more severe punishment is for even small values of δ_L and δ_G. Smoothness function 11 punishes neighbour pixels assigned to different labels with a high energy value. The data cost function is identical to function 6.

After defining the energy function, instead of max-product, which is frequently used in LBP, we invoke the min-sum method to calculate the message delivered from pixel p to its neighbour pixel q:

$$msg_{p \to q}(l_p) = \min_{l'}(D(l_p) + V_{pq}(l_p, l') + \sum_k msg_{k \to p}(l')) \quad (12)$$

Min-sum is a minimization problem because we are trying to find the smallest cost. In function 12, l' is the label which can be either white or black and k is the neighbour of pixel p except q. In our implementation, we use 4 neighbourhood (right, left, up and down). Min-sum is more computationally efficient than the max-product because it does not have any expensive exponential functions and uses addition operations mainly so that it is not as easy to underflow as max-product.

Finally, to avoid overflow, we normalize each message before sending it to its neighbour:

$$A = \log \sum_{l_p} \exp(msg_{p \to q}(l_p)) \quad (13)$$

$$msg_{p \to q}(l_p) = msg_{p \to q}(l_p) - A \quad (14)$$

After this step, all messages has been restricted to interval (0, 1).

The LBP is executed on the foreground object. The target is to assign a set of labels over all pixels that minimizes the total energy:

$$E = \sum_p D(l_p) + \sum V_{pq}(l_p, l_q) \quad (15)$$

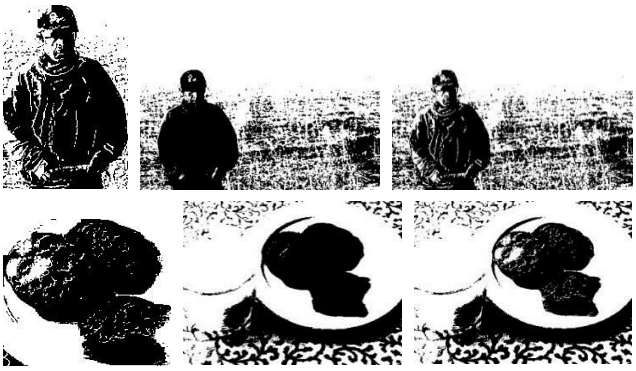
In function 15, the data term is $\sum_p D(l_p)$, which means the total data cost is the summation of all pixels. The smoothness term is $\sum V_{pq}(l_p, l_q)$, where the summation is taken over all neighbouring pixel pairs (in our implementation, 4 neighbours model is used, that is right, left, up and down). The result is shown in Figure 7.

a b c

Figure 7. Results Produced by Belief Propagation Algorithm. (a) Input image. (b) Results produced by BP without masks. (c) Results produced by BP only on the foreground objects.

3.4 Combination

We combine the results above and produce the final result. The background of the final result comes from Otsu method and the foreground of it comes from LBP. The final result is shown in Figure 8.



a b c

Figure 8. Combining. (a) Foreground object generated by LBP with mask. (b) Otsu method used on the whole image. (c) Combining foreground object in (a) and background in (b) with mask (a), we obtain the final result.

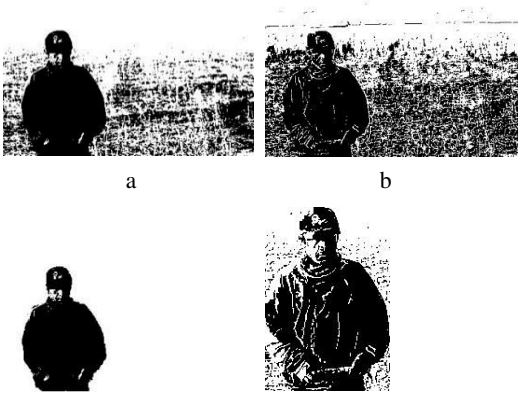
4. Experiments

All experiment images are from MSRA10K benchmark dataset and can be downloaded from http://mmcheng.net/zh/salobj/ .

4.1 Comparison of different methods

We show that using LBP with mask can gain good results with respect to render details of foreground object in Figure 9 and Figure 10. The parameters of LBP used in our experiment are: iteration time=10, $\gamma = 0.3333 \quad \gamma V = 16 \quad \alpha = 0.7$

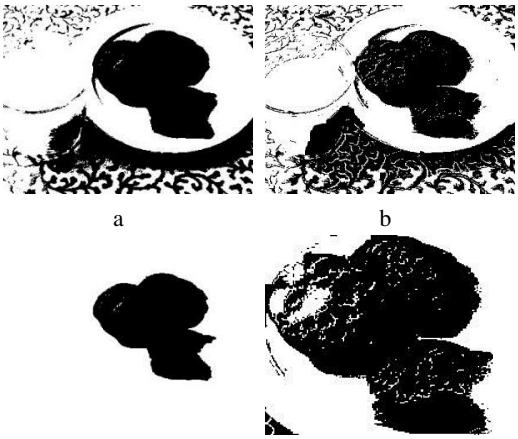
From the results, we find that Otsu method cannot enhance the details of foreground object even if the mask is used because it only calculates the grayscale histogram of input image once while LBP method with mask on the foreground object can improve the details significantly.



a b c d

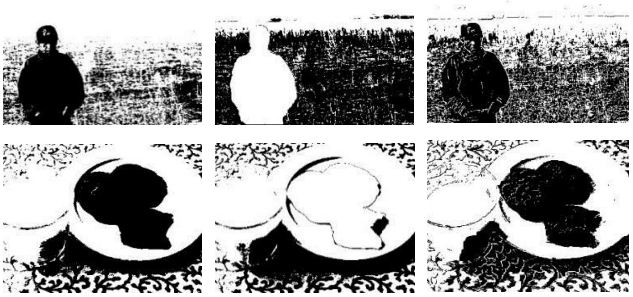
Figure 9. Comparison of Different Methods. (a) Otsu method on the whole image without mask. (b) BP on the whole image without mask. (c) Otsu method on the foreground object only with mask. (d) BP on the foreground object only with mask.

We also observed that Otsu method on the whole image can deduce the background details. To make the foreground objects stand out, we combine the background from Otsu method and the foreground from the LBP. See Figure 11.



a b c d

Figure 10. Another Instance of Comparison of Different Methods. (a) Otsu method on the whole image without mask. (b) BP on the whole image without mask. (c) Otsu method on the foreground object only with mask. (d) BP on the foreground object only with mask.



a b c

Figure 11. Comparison of the Background Produced by Different Methods. (a) Otsu method on the whole image without mask. (b) Otsu method on the background only with mask. (c) BP on the whole image without mask.

4.2 CPU time

We also compute the CPU time by testing 105 input images. The CPU time of each step is listed in Table 1. According to this table, the most expensive operation is the LBP on the whole image without mask. But fortunately, we just use the LBP on the foreground and Otsu on the whole image only, the total average CPU time is 8.58 + 0.19 + 0.08 = 8.85s.

Table 1. The Run Time of Different Methods in Each Step

Method	Average CPU Time
LBP on the whole image	25.61s
LBP on the foreground only	8.58s
Otsu on the whole image	0.19s
Combining	0.08s

4.3 More results

More results are listed in Figure 12.

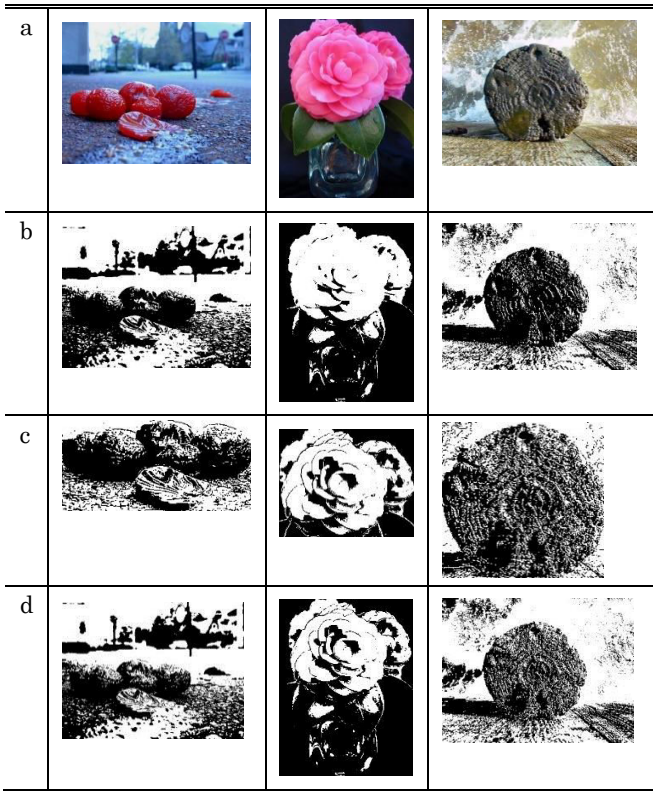


Figure 12. More Results. (a) Input image. (b) Otsu on the whole image. (c) BP on the foreground object only. (d) Combining results.

5. Conclusions

In this paper, we presented a method for converting a color image with complex background into a black and white art image. Mask is used to separate the foreground object and the background. By using different algorithm to render the foreground and background respectively, our method can keep the details of foreground object as far as possible and deduce the details of the background so that foreground object can stand out from the background.

There are avenues for future work. First, although the input is color image, we convert it into grayscale image before processing it and the color information is ignored here. We want to find a method to process color image directly without any conversion.

Secondly, for some images, our method cannot improve the result. See Figure 13. There is no enhancement for the details of foreground object even if mask is used. The reason, in our opinion, is that the contrast of the flower is too low. The Otsu method only calculates the histogram of the input image while the data cost function and smoothness function in BP only concern with the grayscale gradient, the change of color and contrast is ignored. We will improve this problem in the future work.

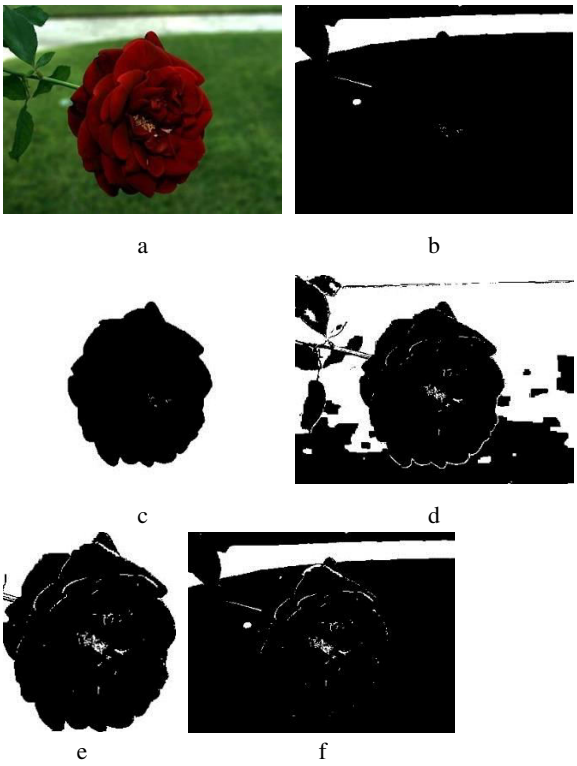


Figure 13. A Bad Result. (a) The input image. (b) Otsu method on the whole image without mask. (c) Otsu method on the foreground object only. (d) BP on the whole image without mask. (e) BP on the foreground object only. (f) Result.

Reference

1) Jie Xu, Craig S. Kaplan: Artistic thresholding. ACM-NPAR 2008: 39-47
2) David Mould, Kevin Grant: Stylized black and white images from photographs. ACM-NPAR 2008: 49-58
3) Carsten Rother, Vladimir Kolmogorov, Andrew Blake: "GrabCut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. 23(3): 309-314 (2004)
4) Yuri Boykov, Marie-Pierre Jolly: Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. ICCV 2001: 105-112
5) Ming-Ming Cheng, Guo-Xin Zhang, Niloy J. Mitra, Xiaolei Huang, Shi-Min Hu: Global contrast based salient region detection. CVPR 2011: 409-416
6) Nobuyuki Otsu: A threshold selection method from gray-level histograms. IEEE Trans. Sys., Man., Cyber. 9 (1): 62–66, 1979
7) Weiss, Y., And Freeman, W: On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. IEEE Transactions on Information Theory 47, 2001.