Understanding and Implementing GrabCut

I. Understanding GrabCut

The GrabCut algorithm [1] is the successor of the well-known GraphCut scheme proposed in [2]. It is a heuristic but popular way even nowadays to extract foreground object from a ROI selected by users. Compared with its predecessor, GrabCut extends the function domain to RGB images, and meanwhile requires even fewer manual operations. It utilizes Gaussian mixture models (GMMs) to model the foreground and background iteratively, which sets a precedent, making it possible to do segmentation on colored images. Furthermore, the min-cut strategy is applied as well to accelerate the convergence, and border matting is introduced to optimize the fringes. In this report, I majorly realized the most important segmentation procedure by implementing iterative foreground and background GMMs optimization. The contents of this report is arranged as follows: In 1.1, the majorly-involved Gaussian mixture modeling theory is described; In Section 2, the procedure of my implementation is provided in 2.1, and the examples of results are given with descriptions in 2.2. Finally, in Section 3, the implementation procedure as well as the result are concluded.

1.1 Gaussian Mixture Modeling (GMM)

The basic idea of GMM is assuming that the distribution of any sample data is generated by a number of Gaussian distributions with different expectations and covariances. Under this assumption, a GMM is made up of several singal Gaussian models, which is capable of characterizing any data contribution.

A Gaussian distribution for multivariate data is shown below:

$$P(x|\theta) = \frac{1}{2\pi^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2})$$

Here, μ is the expectation, Σ is the covariance, and D is the dimension of data. Before given the GMM probability distribution, some key variables are defined first. Denote x_j as the j-th observed sample, where $j = 1, 2, \dots, N$; K represents the number of single Gaussian models, and $k = 1, 2, \dots, K$ denotes the k-th model within a GMM; $\alpha_k \geq 0$ is the probability of the observed data generated from the k-th model, and $\sum_{k=1}^{K} \alpha_k = 1$; $\Phi(x|\theta_k)$ refers to the Gaussian probability density function of the k-th model, where $\theta_k = (\mu_k, \sigma_k^2)$. The GMM probability distribution can be expressed as:

$$P(x|\theta) = \sum_{k=1}^{K} \alpha_k \Phi(x|\theta_k)$$

Here, $\theta = (\widetilde{\mu_k}, \widetilde{\sigma_k}, \widetilde{\alpha_k})$, which are the expectation, covariance, and probability of occurrence of each sub-model within the GMM.

For a single Gaussian model, the maximum likelihood can be utilized to estimate the value of θ:

$$\theta = argmax_{\theta} L(\theta),$$

$$L(\theta) = \prod_{j=1}^{N} P(x_j|\theta).$$

Usually, we apply maximum log likelihood, since the product result can be too small to be analyzed. The expression of the maximum log likelihood is written as:

$$logL(\theta) = \sum_{j=1}^{N} log\, P(x_j|\theta)$$

For the GMM, its log-likelihood function can be referred as:

$$logL(\theta) = \sum_{j=1}^{N} log\, P(x_j|\theta) = \sum_{j=1}^{N} log\, (\sum_{k=1}^{K} \alpha_k \Phi(x|\theta_k))$$

Due to the fact that for each sample of data, it is agnostic which sub-model it belongs to in advance, thus using differentiation to directly retrieve the θ maximizing $L(\theta)$ is impossible in this case. Fortunately, Dempster *et al.* proposed the EM algorithm [3] which can be applied to the maximum likelihood estimation for parameters of probabilistic models with hidden variables. Luckily, there's no need for us to implement GMM by ourselves, since there have been a lot of existing open-source implementation, i.e. encapsulated classes and functions, for developers to deploy freely. In my implementation of GrabCut, the 'Gaussianmixture' module from the Scikit-learn ML library for Python is utilized for GMMs' maximum likelihood estimation, which will be announced in the next section as well.

II. Implementation Procedure and Results

2.1 Implementation Procedure

The implementation procedure of GrabCut, especially the iterative GMMs optimization scheme, is demonstrated as Fig.1.

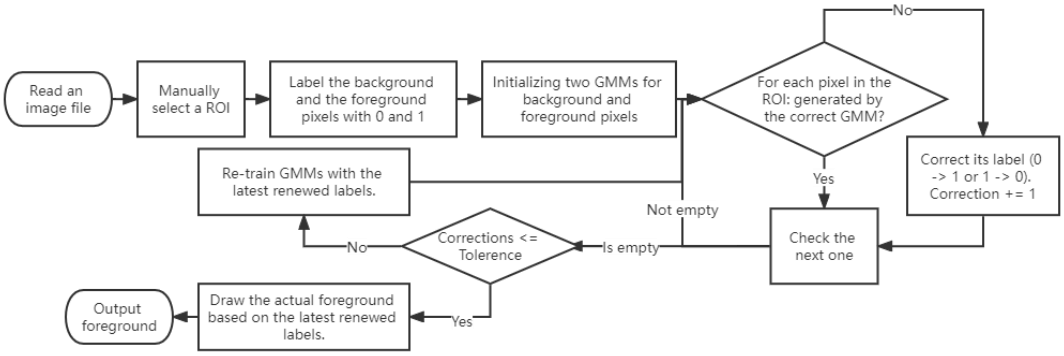

Fig.1 The schematic diagram of my GrabCut programme

Notice that when the label of a pixel is corrected, i.e. a currently foreground-labeled pixel turns out to score higher (return higher log-likelihood result) after being substituted into the current background GMM than being substituted into the foreground GMM, such pixel becomes a background-labeled pixel and is added to the background pixels list (which is also removed from the foreground as well) for re-training the two GMMs with the renewed labeled pixels in the final step of the loop. Furthermore, a convergence mechanism is designed and embedded in the programme which isn't fully shown in the diagram. A tolerance value is set to guide the progress of convergence. Note that the GMMs aren't considered converged until the value of correction times has been within the tolerance for more than r rounds, namely the r-round validation standard. In my experiment, the r is set to 3. Such convergence validation standard is based on the idea that the correction times recorded after each round is not always declining, and sometimes causes false convergence. An intuitive way to deal with such problem is to wait for a stationary region, where all corrections can be tolerated within a few rounds.

Since the computing hardware are much more advanced and capable these days, I iteratively fine-tune the GMM parameters by using the entire ROI in each round rather than apply min-cut, which turns out to be still feasible with a bit more time-consuming. The results of my programme is shown in 2.2.

2.2 Result & Analysis

Fig.2 to Fig.6 display the output examples of the scheme.

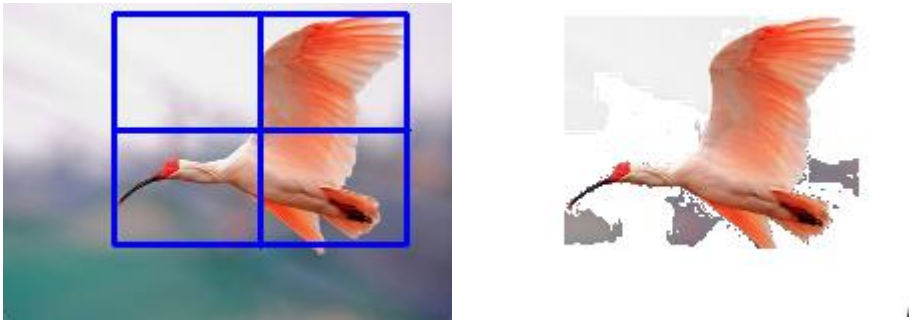

Fig.2 Output example 1

Fig.3 Output example 2

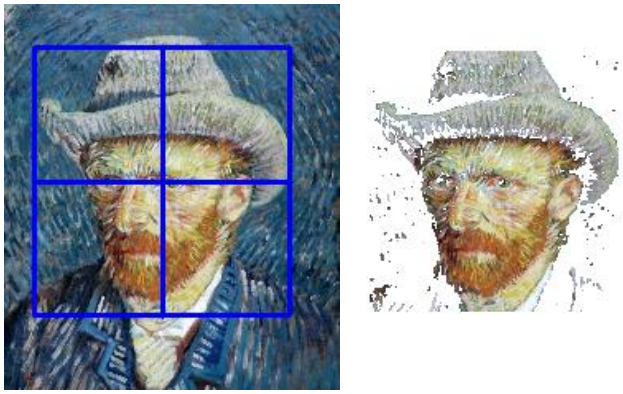

Fig.4 Output example 3

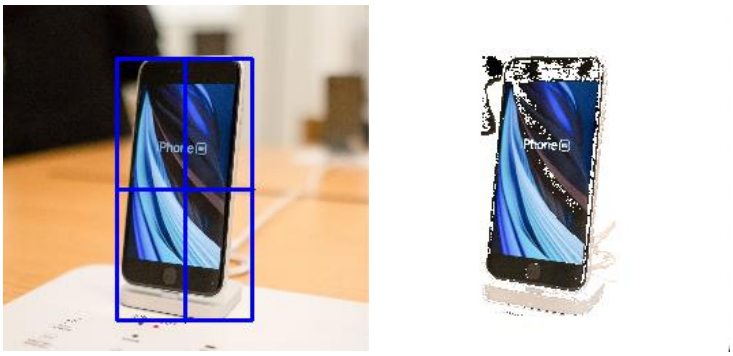

Fig.5 Output example 4

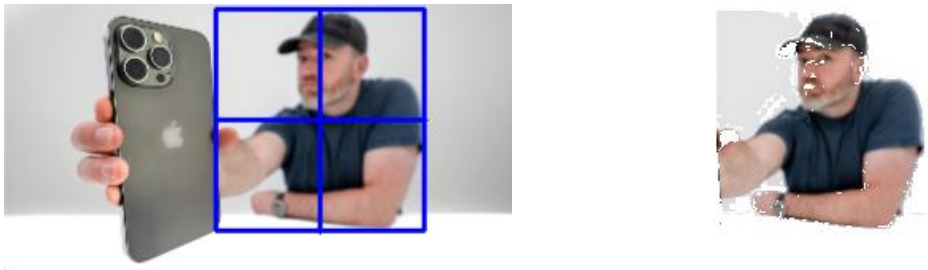

Fig.6 Output example 5

As shown in the above examples, the algorithm is able to remove most of the background pixels within the ROIs. The problem is that, i.e. Fig.2, Fig.3, and Fig.6, noticeable areas of forged foreground parts are retained. In the meantime, Fig.5 shows conspicuous deficiency on the remained foreground. I think that the RGB-based modeling might be a reason for the flaws. Take Fig.5 for example. In Fig.5, both the

foreground (the smartphone's front face) and the background (especially the dim figure) contain considerable proportion of visually black pixels, and as is seen from the segmentation result, most of the lost foreground pixels are originally black, which offers a clue that, regions with similar proportion and color between the foreground and the background act as distractors which can disturb the GMM from ideally converging. It is also suggested that the false retention of the background pixels in Fig.2, Fig.3, and Fig.6 encounters similar causes. Besides, the native contrast ratio of an image, ratio of ROI to original image, ratio of foreground region to ROI, etc. may all contribute to segmentation imperfection. The original GrabCut scheme achieves better effect because it takes more factors into consideration, i.e. contrast ratio, border matting.

III. Conclusion

In this report, the GrabCut algorithm is reproduced by utilizing the iterative GMMs optimization. Especially, an r-round validation standard in introduced for locating a relatively stable convergence of the two GMMs. In addition, the results of my implementation is listed and analyzed, and one important conclusion is that color deception and unawareness of the native contrast can lead to forged foreground pixels and incomplete preservation of the ideal foreground area. Such algorithm has a wide range of use in the field of computer vision, and with the development of neural networks, heuristic means are gradually fading away, but schemes like GrabCut and GraphCut surely deserves the titles as groundbreaking foundations.

Reference

[1]Rother, C., V. Kolmogorov, and A. Blake. "Interactive foreground extraction using iterated graph cuts, 2004." *SIGGRAPH*. 2004.
[2]Boykov, Yuri Y., and M-P. Jolly. "Interactive graph cuts for optimal boundary & region segmentation of objects in ND images." *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*. Vol. 1. IEEE, 2001.
[3]Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977): 1-22.