DIP Final:Grabcut

1st Jun-Chen Hong
Dept of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
r11944050@ntu.edu.tw

2nd Chin Ting Hsu
Dept of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
b09902030@ntu.edu.tw

3rd Xin-Yi Li
Dept of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
b09902022@ntu.edu.tw

I. Motivation

The k-means algorithm implemented in homework 3 reminds us of a common technique used in image editing called background removal. Often, we only need the foreground objects and prefer to remove the background from an image. In the manual approach of selecting and removing the background, having an intelligent algorithm that can update the image to better match the user's desired outcome would make the background removal process more efficient. That's why we came across this paper that introduces the GrabCut technique, which can quickly and accurately separate the foreground and background, making it easier to extract the desired foreground image. It is also interactive, allowing users to provide annotations to refine the results and make them closer to the intended target.

II. Problem defintion

Given an image and a bounding box, we aim to initially perform a rough segmentation into foreground and background. Subsequently, with each user click, our algorithm should swiftly and accurately update the desired cropping area. This allows for faster and more satisfactory background removal, catering to the user's preferences.

III. Algorithm

A. graphcut

the foundation of [1]grabcut is the segmentation method developed by Boykov and Jolly which is called [2]graphcut.

Graph-cut performs segmentation using the min-cut approach. A graph is constructed based on the weights defined in the paper, and a min-cut is applied. The weights are defined according to factors such as pixel classification and neighboring relationships mentioned in the Fig.1.

Once the min-cut algorithm is implemented, the image will undergo segmentation, resulting in a distinction between foreground and background regions.

The following table gives weights of edges in $\mathcal{E}$

edge	weight (cost)	for
$\{p,q\}$	$B_{\{p,q\}}$	$\{p,q\} \in \mathcal{N}$
$\{p,S\}$	$\lambda \cdot R_p(\text{“bkg”})$	$p \in \mathcal{P},\ p \notin \mathcal{O} \cup \mathcal{B}$
	K	$p \in \mathcal{O}$
	0	$p \in \mathcal{B}$
$\{p,T\}$	$\lambda \cdot R_p(\text{“obj”})$	$p \in \mathcal{P},\ p \notin \mathcal{O} \cup \mathcal{B}$
	0	$p \in \mathcal{O}$
	K	$p \in \mathcal{B}$

where

$$K = 1 + \max_{p \in \mathcal{P}} \sum_{q:\ \{p,q\} \in \mathcal{N}} B_{\{p,q\}}.$$

Fig. 1. table of edge weight

B. grabcut

Based on Graph-cut,Grabcut extended graphcut to iterated segmentation.Grabcut use two Gaussian Mixture Model(GMM) as learning model for background and foreground.

In the segmentation process, Gibbs energy, as shown in Fig.3, is utilized.

The variable α denotes the specific Gaussian Mixture Model (GMM) to which a pixel belongs. The variable "k" indicates the Gaussian component within the GMM to which the pixel belongs. The variable θ represents the GMM parameters, encompassing properties such as the mean, standard deviation, and weights of each Gaussian distribution. Lastly, "z" represents the pixel value.the symbols also have same meaning in

Initialisation

- User initialises trimap T by supplying only T_B. The foreground is set to $T_F = \emptyset$; $T_U = \overline{T}_B$, complement of the background.
- Initialise $\alpha_n = 0$ for $n \in T_B$ and $\alpha_n = 1$ for $n \in T_U$.
- Background and foreground GMMs initialised from sets $\alpha_n = 0$ and $\alpha_n = 1$ respectively.

Iterative minimisation

1. *Assign GMM components to pixels:* for each n in T_U,
$$k_n := \arg\min_{k_n} D_n(\alpha_n, k_n, \theta, z_n).$$
2. *Learn GMM parameters* from data $\mathbf{z}$:
$$\underline{\theta} := \arg\min_{\underline{\theta}} U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z})$$
3. *Estimate segmentation:* use min cut to solve:
$$\min_{\{\alpha_n:\, n \in T_U\}} \min_{\mathbf{k}} \mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}).$$
4. Repeat from step 1, until convergence.
5. Apply border matting (section 4).

User editing

- *Edit:* fix some pixels either to $\alpha_n = 0$ (background brush) or $\alpha_n = 1$ (foreground brush); update trimap T accordingly. Perform step 3 above, just once.
- *Refine operation:* [optional] perform entire iterative minimisation algorithm.

Fig. 2. Iterative image segmentation in GrabCut

$$\mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) + V(\underline{\alpha}, \mathbf{z}),$$

Fig. 3. gibbs energy

Fig.2.

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

$$D(\alpha_n, k_n, \underline{\theta}, z_n) = -\log \pi(\alpha_n, k_n) + \frac{1}{2} \log \det \Sigma(\alpha_n, k_n) + \frac{1}{2}[z_n - \mu(\alpha_n, k_n)]^\top \Sigma(\alpha_n, k_n)^{-1} [z_n - \mu(\alpha_n, k_n)].$$

Fig. 4. U term

In Fig.4, the energy formula includes a term referred to as "U," which is associated with the data term. This term quantifies how the data within the Gaussian Mixture Model (GMM) contributes to the overall Gibbs energy, taking into account the GMM's parameters. The U term reflects the influence of the GMM's data on the energy computation within the segmentation process.

In Fig.5,the enrergy formula includes a term refereed to as "V",which is associated with smoothness term. It is basically unchanged from the monochrome case except that the contrast

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n) \in \mathbf{C}} [\alpha_n \neq \alpha_m] \exp -\beta \|z_m - z_n\|^2.$$

Fig. 5. smoothness term

term is computed using Euclidean distance in colour space.

C. border matting

In our study, the input consists of an image along with its corresponding original trimap. Our first step is to construct a new trimap that expands the original TF (foreground) and TB (background) regions to include an additional region, TU (region close to the contour). To identify TU, we need to first determine the boundaries of TF and TB. We treat the original trimap as a binary image and aim to identify the boundaries of the image. For this purpose, we employ the Canny edge detection algorithm. The resulting boundaries are referred to as the contour C, and we store the pixels along the contour in a list in sequential order, completing the contour construction.

Next, we proceed to compute the new trimap. For each pixel, we aim to determine which point on the contour it is closest to. Although the paper does not explicitly mention the method for finding the closest point, examining all the contour pixels for each pixel would be time-consuming. However, since the contour is continuous and neighboring pixels should correspond to nearby contour pixels, we can accelerate the grouping process by creating a dynamic programming (DP) table. This table leverages precomputed groups of neighboring pixels to expedite the grouping of the current pixel.

By employing this approach, we can efficiently find the closest distance between each pixel on the image and the corresponding point on the contour C. Pixels within a distance of 6 are marked as TU.

Now, we have an energy function composed of a data term and a smoothing regularizer in the latter part. Our goal is to test which sigma-delta pairs can minimize the energy. The equation is as follows:

$$E = \sum_{n \in T_U} \tilde{D}_n(\alpha_n) + \sum_{t=1}^{T} \tilde{V}(\Delta_t, \sigma_t, \Delta_{t+1}, \sigma_{t+1})$$

Fig. 6. energy function

$$\tilde{V}(\Delta, \sigma, \Delta', \sigma') = \lambda_1 (\Delta - \Delta')^2 + \lambda_2 (\sigma - \sigma')^2,$$

Fig. 7. smoothing regularizer

Let's first explain the relatively straightforward smoothing regularizer. We aim for a smooth variation of alpha, so we prefer smaller differences between neighboring sigma-delta pairs. Essentially, we calculate the differences for each parameter separately and then take a weighted sum. Here, lambda 1 is set to 50 and lambda 2 to 1000.

$$\tilde{D}_n(\alpha_n) = -\log \mathbf{N}\left(z_n; \mu_{t(n)}(\alpha_n), \Sigma_{t(n)}(\alpha_n)\right)$$

Fig. 8. data term

$$\begin{aligned} \mu_t(\alpha) &= (1-\alpha)\mu_t(0) + \alpha\mu_t(1) \\ \Sigma_t(\alpha) &= (1-\alpha)^2\Sigma_t(0) + \alpha^2\Sigma_t(1). \end{aligned}$$

Next, let's examine the data term. For each pixel in TU, we calculate delta Dn using the following formula, where N represents the Gaussian PDF. Let's first discuss how to compute the mean and variance of this distribution. We create a 41x41 matrix on the boundary pixels and find the intersection region with TF and TB. From these two parts, we compute the sample mean and sample variance. Then, we interpolate using the following formula, where each pixel calculates its alpha value based on distance and the current sigma-delta values. By applying this formula, we obtain the mean and variance, which allows us to compute the PDF and ultimately calculate the data term.

IV. Experimental results

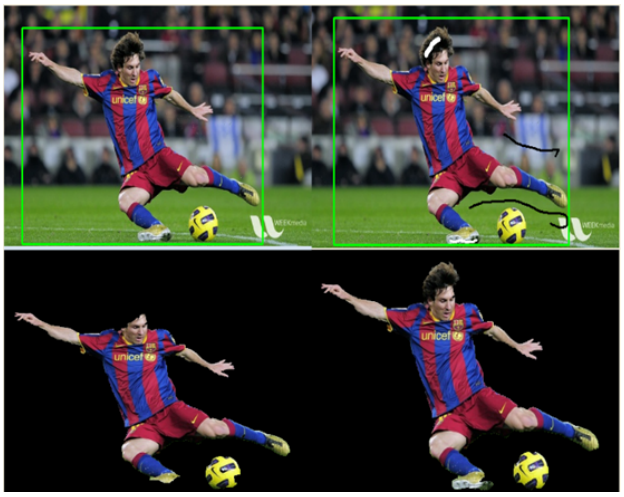

Fig. 9. complex background result

In Fig.9, it demonstrates the impact of the brush on the segmentation result. Without the use of the brush, the algorithm struggles to properly segment the image with just a single bounding box. Therefore, the brush serves as a helpful tool in achieving accurate segmentation. Due to the iterative nature of the process, we can improve the result obtained from the initial single bounding box by incorporating the brush later on. This iterative approach allows for refinement and enhancement of the segmentation result.

In Fig.10, the performance of grabcut on an easy background with a single bounding box is showcased. Clearly, when compared to the result obtained on a complex background, it appears to have an acceptable outcome in this particular example. These two examples highlight the influence

Fig. 10. easy background result

of grabcut in different usage cases on overall performance and convenience. It demonstrates that grabcut can yield satisfactory results in scenarios where the background is relatively simple, and it offers a convenient solution for segmentation tasks. However, when faced with more complex backgrounds, additional techniques or brush may be necessary to achieve desired results.

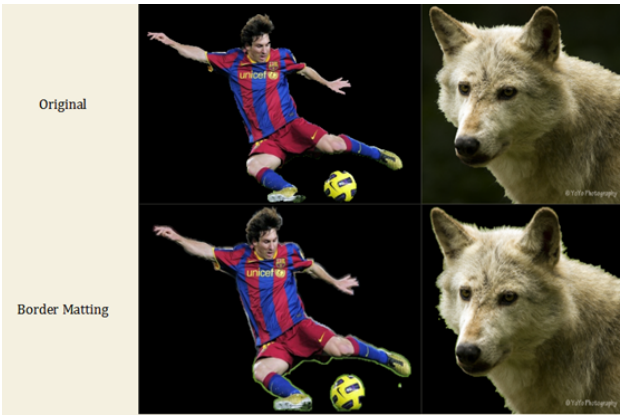

Fig. 11. border matting result

In Figure 11, the influence of border matting on the borders is illustrated. It appears to have a more pronounced effect on the result obtained on a complex background. Border matting helps to enhance the natural appearance of the image within the picture. Notably, there are noticeable inconsistencies in the original result, such as in the depiction of shoes, hands, and other details. However, after applying border matting, the visual performance of the image improves significantly, creating a more natural and seamless integration within the picture.

V. Discussions

In final project, we implement grabcut algorithm, border matting and user interface. We prove that grabcut have good performance on distinguish item from picture, and we also show that border matting can smooth item's border and make it more clear.

VI. Division of the work

- Jun Chen Hong:grabcut algorithm
- Chin Ting Hsu:border matting
- Xin-Yi Li:frontend

References

[1] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": Interactive foreground extraction using iterated graph cuts," in *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, (New York, NY, USA), p. 309–314, Association for Computing Machinery, 2004.

[2] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary region segmentation of objects in n-d images," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, pp. 105–112 vol.1, 2001.