A Real-time Interactive Tool for Image Cutout

Chen Liu[†], Fengxia Li, Yan Zhang, and Shouyi Zhan

The Computer Science School, Beijing Institute of Technology, Beijing, 100081, China.

Abstract

We present an interactive tool for extracting foreground objects from image in real-time. As seen from the object boundary, the segmentation is to split pixel-pairs right on the boundary. The system utilizes the user input while the user roughly paints the stroke along the object boundary. Either side of the stroke is assumed to lie in foreground or background, and the mouse trace is supposed to coarsely indicate the object contour. We use the appearance, gradient and contour information to formulate the segmentation problem to an energy minimization problem, which is solved by graph cut. The appearance model is built via local sampling, since it often provides more discriminative model in local areas than that from global sampling. In addition, our system can dynamically adjust the stroke and the optimization region for which segmentation needs to be computed. The optimization region is much smaller than the whole image. This greatly reduces the computational complexity in each iteration and also gives the system the ability of incrementally segmenting. Experiments show the effectiveness of our methods in improving the segmentation performance.

Categories and Subject Descriptors (according to ACM CCS): I.4.6 [Computer Graphics]: Segmentation—Pixel classification

1. Introduction

Interactive image cutout is popular in both commercial and academic areas. Comparing to fully automatic and manual image cutout methods, a small amount of user inputs can often resolve ambiguities in complex cases e.g. low-contrast edges, camouflages, and save users from tedious jobs of segmenting images pixel by pixel as well.

We refer to several straps of related works. First, the boundary-based methods, such as Photoshop's magnetic lasso, cut out the foreground by fitting a curve to enclose it when the user traces along the boundary. These techniques reduce the degree of user inputs, but they require images contain strong edges between fore- and back- ground and demand extremely carefulness for tracing. Second, the region-based methods segment the image by matching pixels to user-specified regions of features, e.g., Photoshop's magic wand, and Graphcut [BJ01] and Grabcut [RKB04]. They often produce decent results while the user specifies parts of the image are foreground or background, but they may fail or need lots of user inputs when image contain ambiguous situations. Recently, researchers have improved the segmentation using shape prior knowledge [KTZ05]. The interesting conclusion they have made is even simple shape cues of foreground can help to produced excellent results. And our interface is inspired by the Soft Scissors [WAC07], which provides a real-time interactive tool for matting.

In this paper, we propose a novel interactive tool for extracting image objects in real-time. The flowchart in each iteration of our system is illustrated in Figure 1. Our system cuts out the object in real-time while the user roughly draws the stroke along its contour (boundary). The stroke is assumed to be painted with its left side supposed to lie in the background (red), its right side assumed to lie in the foreground (yellow), and the mouse trace supposed to roughly indicate the object contour.

The segmentation is computed in an incremental manner, which the system only handles a small portion of pixels in each iteration. First, the system determines a newly-added region of pixels since the previous iteration. And based on this newly-added pixels, new foreground and background samples and the optimization region can be estimated. The optimization region is much smaller than the whole image,

[†] liuchen.1983@gmail.com

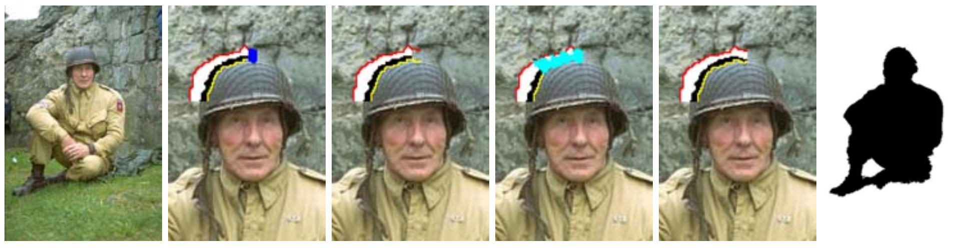

Figure 1: *The flowchart of our system. While the user paints along the solider's (a) contour, the system first decides a newly-added region (blue in (b)) since the previous iteration. New fore- (yellow) and back-ground (red) samples ((c)) on either side of the new region and the optimization region (green in (d)) is determined for the segmentation (e). (f) is the whole segmentation.*

so it reduces the complexity of optimization. In each iteration, the segmentation is solved via graph cuts on the optimization region. And as this region of adjacent iterations overlap by some pixels, we incrementally compute the segmentation by reusing the result in the previous iteration. This is more efficient than recomputing from the scratch. Three main contributions are made in this paper: 1) The novel intelligent user interface; 2) The adaptivelocal appearance model and 3) the incremental method for segmentation.

2. The Proposed Algorithm

2.1. Defining the Energy Function for Segmentation

Suppose the set $S = \{s_i | i = 1,...,n\}$ represents pixels need to be segmented. The segmentation is defined by a set $A = \{A_i | i = 1,...,n\}$, A_i specifies a labeling of s_i, $A_i \in L$. $L = \{l_j | j = 1,...,m\}$ is the label set. Take our application for example, $L = \{0,1\}$, for '1' denoting the foreground and '0' for the background. This segmentation problem can be formulated as an energy minimization problem by:

$$E(A) = \lambda \cdot \sum_{s_i \in S} E_1(A_i) + (1-\lambda) \cdot \sum_{(s_i,s_j)\in N} E_2(A_i, A_j) \quad (1)$$

The unary term E_1 reflects the cost for assigning s_i to A_i , the more they fit each other, the less cost needs to pay. And the pair-wise term E_2 is a penalty when neighboring pixels are assigned with different labels. It encourages similar adjacent pixels to take the same label. N is a neighborhood system defined on S, which contains all pairs of neighboring pixels, under the standard 8-connectivity. And $\lambda \in [0,1]$ specifies the relative importance between two terms.

The unary term is formulated by:

$$E(A_i) = -\log(P(s_i|\Theta_{A_i})) \quad (2)$$

$P(\cdot|\cdot)$ is the conditional probability given the appearance model of the foreground and the background, which is represented by the Gaussian Mixture Models (GMMs):

$$P(s_i|\Theta_{A_i}) = \sum_{m=1}^{k} \alpha_m p(s_i|\theta_m) \quad (3)$$

$$p(s_i|\theta_m) = (2\pi)^{-\frac{3}{2}} |\Sigma_m|^{-\frac{1}{2}} \cdot e^{(-\frac{1}{2}(s_i-\bar{s}_m)^\top \Sigma_m^{-1}(s_i-\bar{s}_m))} \quad (4)$$

where each GMM has $k = 5$ components. α_m is the mixture weight parameter for $\Sigma^k \alpha_m = 1$. $\theta_m = (\bar{s}_m, \Sigma_m)$ contains the mean and the covariance vectors of the m-th component.

The pair-wise term is defined using gradient information:

$$E_2(A_i, A_j) = |A_i - A_j| \cdot \varphi(i, j),$$
$$\varphi(i, j) = \delta \cdot dist_1(s_i, s_j)^{-1} \cdot \exp(-\beta^{-1} \cdot \| s_i - s_j \|^2) \quad (5)$$

where $\| s_i - s_j \|^2$ measures the distinction between neighboring pixels in the color space and $dist_1()$ is their Euclidean distance. δ is a scale parameter and β is a penalty threshold.

While the stroke runs along the object contour, the contour information can be coasely obtained from the mouse trace. As seen from the object boundary, the segmentation is to split pixel-pairs locating right on the object contour. It also implies the closer pixel-pair is near the contour the more likely it should be split. This can be defined by:

$$E_2'(A_i, A_j, \Phi) = |A_i - A_j| \cdot \phi(i, j, \Phi),$$
$$\phi(i, j, \Phi) = \omega \times (dist_2^2(s_i, s_j, \Phi) + 1)^{-1} \quad (6)$$

Φ denotes the object contour, $dist_2()$ measures the Euclidean distance between the pixel-pair and the contour. The parameter ω determines how much the energy value changes according to the distance. Then Equation 1 is extended by:

$$E'(A, \Phi) = \lambda \cdot \sum_{s_i \in S} E_1(A_i) +$$
$$(1-\lambda) \cdot \sum_{(s_i,s_j)\in N} (E_2(A_i, A_j) + E_2'(A_i, A_j, \Phi)) \quad (7)$$

2.2. Minimizing the Energy Function via Graph Cuts

Energies like Equation 1 can be solved using the graph cut if they are sub-modular [KZ04]. The sub-modularity condition requires the pair-wise term taking less value for two variables taking the same label than that for them taking different labels, which can be formulated as:

$$E(0,0) + E(1,1) \le E(0,1) + E(1,0) \quad (8)$$

Equation 7 satisfies the condition, thus it can be sovled by the graph cut algorithm. A graph $g =< v, \varepsilon >$ is created. The vertex set is $v = S \bigcup L$ with S corresponding to pixels and $L = \{0, 1\}$, two terminals, representing fore/background respectively. The edge set is $\varepsilon = \{(s_i, s_j)|(s_i, s_j) \in N\} \bigcup \{(s_i, 1), (s_i, 0)|s_i \in S\}$, where N is the neighborhood system on S. We ultilize the graph cut by projecting terms on graph edges' costs:

$$\begin{cases} w(s_i, s_j) = (1-\lambda) \cdot (\varphi(i, j) + \phi(i, j, \Phi)) \\ w(s_i, 1) = \lambda \cdot E_1(A_i = 0) \\ w(s_i, 0) = \lambda \cdot E_1(A_i = 1) \end{cases} \tag{9}$$

Figure 2 gives an example, for the proof of the min-

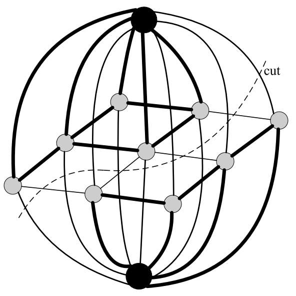

Figure 2: *An example of a graph and its cut. Black nodes are terminals, and gray ones are normal nodes; and the dashed line shows a cut passing all cut-edges.*

cut denoting the optimal segmentation, you can refer to [BJ01] [KZ04]. In our case, the graph topology in adjacent iterations is similar, so we adopt the dynamical graph cut algorithm [KT07] which is quite efficient for reusing results of the last iteration than re-calculating from the scratch.

3. The Intelligent User Interface

As the user traces along the object contour, the system automatically adjusts: 1) the length and width of the stroke, 2) the newly-added region, 3) the optimization region and 4) new fore- and back-ground samples, as illustrated in Figure 3.

3.1. Estimating the Stroke and the Newly-added Region

Wider stroke might involve noise pixels; on the other hand, thinner stroke probably can not provide enough foreground and background information. Both will result in indiscriminative appearance models which lead to inaccurate results. In order to alleviate this, our system dynamically estimates the stroke according to the local statistics of the image.

At each iteration t, it first calculates the direction of current stroke, o_t, then by sampling some lines of pixels parallel and perpendicular to this direction, the length and width of the new stroke are estimated by considering the color variation, and these also give rise to the newly-added region.

For estimating the stroke length, we use n sampling lines parallel to o_t, where n initially set to be 5, and is changed to the width of the previous stroke at each iteration. The stroke length, len_t, is fixed when new sample pixels obtained by extending sampling lines step by step increase the variance rate of color, namely $len_t = x$ when $|VAR(S_{x+1})| \cdot |VAR(S_{x-1})| \geq |VAR(S_x)|^2$, where x is the distance from new sample pixels to the previous stroke, $x = 1, \ldots, 8$, S_x represents the set of sample pixels within the length x, and $|\cdot|$ denotes the determinant.

Then, to estimate the current stroke width, we sparsely sample pixels distributed along len_t sampling lines perpendicular to those for the stroke length with the maximum width of 12 pixels. Based on the previous appearance model, each sample pixel takes the value, $W_{s_i} = \frac{E_1(A_i=0)}{E_1(A_i=0)+E_1(A_i=1)}$. The width is determined to cover all pixels with $W_{s_i} \in [0.2, 0.8]$. Finally, the length and width of the stroke are determined, and certainly the newly-added region is obtained.

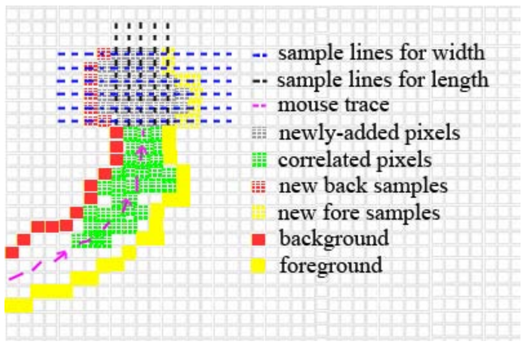

Figure 3: *The illustration of our intelligent interface.*

3.2. Local Sampling vs. Global Sampling

Although global sampling methods [BJ01] [RKB04] often produce decent results, they may fail when the foreground and the background overlap in color space. This partially because the global appearance model is not discriminative enough in local areas, for it only covering the statistical information of the whole image rather than capturing variations in local areas near the foreground boundary. Nevertheless, in a certain local area, the foreground and the background can be distinguished in most cases using the local appearance model via local sampling.

We assume that the left and the right edge of the stroke lie in the background and the foreground respectively. Then the GMMs are used to build foreground and background color models by densely sampling pixels on either side of the newly-added region (Figure 3). The GMMs are updated synchronously using recently added foreground and background samples. And if the number of correlated pixels in the previous iteration is less than that of newly-added pixels, we re-compute the GMMs again using new sample pixels.

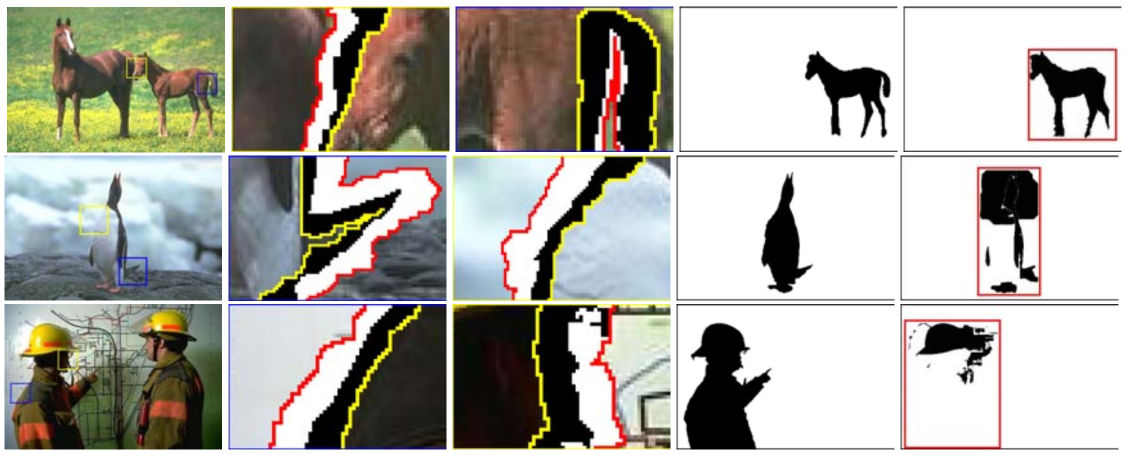

Figure 4: *Comparison examples of our method and GrabCut. The first column shows the images, and the 2nd and 3rd columns are segmentation details of our method. Results from our mehtod and Grabcut are presented in column 4 and 5 respectively.*

3.3. Solving the Optimization Region

New foreground and background samples will affect the labeling of nearby correlated pixels in the previous iteration. So the optimization region in the current iteration, for which the segmentation needs to be computed, should involve both newly added pixels and all correlated pixels in the previous iteration. To determine pixels that are affected by the newly-added region, we again use local statistics of the image. First, newly-added pixels are clustered into few small regions, $\{C_k\}$, $k = 5$ in our system. Then we represent the impact of the newly-added region on pixels in previous iteration by simply computing their distance in color space:

$$I_i = \min_k(\|s_i - C_k\|^2) \tag{10}$$

Pixels satisfy $I_i \leq 50$ are chosen in the optimization region. Comparing to compute the segmentation on the whole image, the optimization region in each iteration is much smaller so that it greatly reduces the computational complexity.

4. Experiments

Experiments were run on a 1.5GHz laptop with 768M memories. And images are from the Berkeley Segmentation dataset `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench`. We set $\lambda = 0.3, \delta = 2.0, \beta = 50.0, \omega = 0.4$ to setup our system.

Figure 4 are some results which show our method produces more accurate segmentation than the Grabcut [RKB04]. In addition, we noticed that GrabCut needs more interactions when the fore- and back-ground overlap in color space. However, our method could give comparatively accurate results using the local appearance model.

5. Conclusions and Future Work

We have presented the real-time tool for interactive image cutout while the user roughly traces the mouse along the object contour. A intelligent user interface is designed as well. And our evaluation shows that our method performs better both in quality and efficiency. And we find it's useful for producing ground-truths for recognition and classification etc.

As manually tracing the mouse along the object boundary puts a heavy load on the user, we'd like to add a Snake-like method in the system; on the other hand, the method of dynamically adjusting the stroke affects a lot in our system, a more robust way needs to be explored.

Acknowledgement We'd like to thank anonymous reviewers for their advices. And this work is partially supported by the Pre-research Project of the '11th Five-Year-Plan' of China (No. B2220061084 and No. 513040103).

References

[BJ01] Boykov V., Jolly M. P.: Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proc. ICCV '01* (2001), pp. 105–112.

[KT07] Kohli P., Torr P. H. S.: Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transaction on PAMI 29*, 12 (Dec. 2007), 2079–2088.

[KTZ05] Kumar M. P., Torr P. H. S., Zisserman A.: Obj cut. In *Proc. CVPR '05* (2005), vol. 1, pp. 18–25.

[KZ04] Kolmogorov V., Zabih R.: What energy functions can be minimized via graph cuts. *IEEE Transaction on PAMI 26*, 2 (Feb. 2004), 147–159.

[RKB04] Rother C., Kolmogorov V., Blake A.: Grabcut - interactive foreground extraction using iterated graph cut. In *Proc. SIGGRAPH '04* (2004).

[WAC07] Wang J., Agrawala M., Cohen M. F.: Soft scissors: An interactive tool for realtime high quality matting. In *Proc. SIGGRAPH '07* (2007).