Improving Interactive Image Segmentation via Appearance Propagation

ID: short5045

affiliations

Abstract

We present Propagate-and-Graphcut (PG), a system that improves interactive image segmentation by leveraging the appearance propagation research. Users' efforts are minimized, and inaccurate inputs are handled. We show experimentally that PG is capable of handling complex scenario settings and outperforms previous approaches.We suggest that PG could serve as an alternative to other general interactive image segmentation methods under most circumstances.

Categories and Subject Descriptors (according to ACM CCS): I.4.6 [Image Processing and Computer Vision]: Segmentation

1. Introduction

Interactive foreground(object) and background segmentation techniques such as [RKB04] [LSTS04] [BJ01] show great promise in different applications. Fig. 1(a) 1(b) illustrates this idea. Though graph cut is widely recognized to be the key in solving the optimization problem for segmentation, challenges on how to exploit users' input still remain and stand in the way of the real deployment for this class of methods.

GrabCut [RKB04] and LazySnapping [LSTS04] apply different clustering algorithms to distinguish the color histograms of the user-selected foreground and background. They both require subtle user refinements for most cases, unless images to be handled have significant appearance difference between the foreground and background as pointed out by [RKMB04]. Other related projects include shape prior [FZ05], probabilistic learning [BRB*04], to name a few.

Our work is based on appearance propagation for HDR image and material editing in [AP08] [PL07], which we consider to be an excellent choice in modeling users' inputs. We propose PG, an improved foreground background interactive segmentation system that has the following advantages over traditional methods: 1) It models textures rather than colors; 2) It uses strokes' geometric meanings; 3) It is robust to inaccurate inputs; 4) It produces reasonable segmentation without any refinement.

We show in the next section the steps of our algorithm. Performance evaluation of PG are provided in Section. 3. Section 4 covers the conclusion and future work.

2. Algorithms

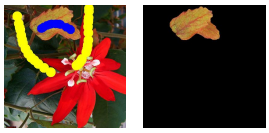
2.1. User Interaction

Users are prompted by PG to use thick strokes to paint part of the foreground and part of the background accordingly. It is not necessary for users to ensure the precision of their inputs because the system could identify and tolerate cases in which users accidentally touch the background with foreground strokes. Section 2.3 will cover this in detail.

Good foreground selection should be composed of strokes covering most of the texture variance and spanning over the whole geometric structure of the foreground, while background strokes are also supposed to bound the foreground. Fig. 1(a)-(b) gives an example input for isolating two leaves from the picture.

2.2. Normalized Appearance Propagation

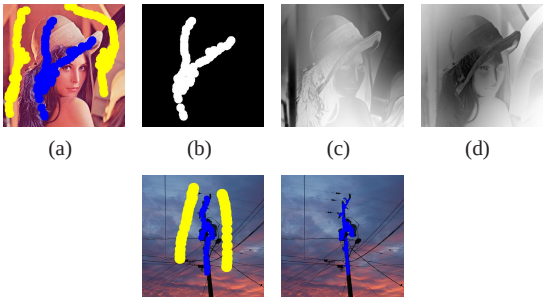
We apply the normalized appearance propagation to both the user's foreground and background labellings independently



(a) (b)

Figure 1: *In (a) the user selection for the foreground and the background are displayed in blue and yellow respectively. (b) is the output of PG.*

and results are used in segmentation. Fig. 2(a)-(d) demonstrates the intuition behind propagation. Users cover textures for both the foreground and the background as shown in Fig. 2(a). Fig. 2(b) visualizes the foreground *propagation target input* g_i for every pixel i, which is produced simply by setting the user-selected foreground pixels to be 1(white) and the rest to be 0(black). Our algorithm propagates the white region to the whole image by brightening pixels that have similar textures to those in the white area subject to geometric constraints. The more likely one pixel belongs to the foreground, the whiter it will be in the foreground propagation result in Fig. 2(c). The background propagation result in Fig. 2(d) could be interpreted in the exact same manner.



(a) (b) (c) (d)

(e) (f)

Figure 2: *The user strokes for the foreground(in blue) and the background (in yellow) on Lenna is shown in (a). (b) is the foreground propagation target input. (c) is the foreground propagation result, and (d) is the background propagation result. (e) The user coarse input with error, and (f) the selected foreground pixels(in blue) after cropping.*

2.2.1. Appearance Propagation

Most of the appearance propagation technique embedded in PG are derived from [AP08] with slight modifications to suit our problem space. The central concept of propagation is to find the value of e_i for each pixel i that maximizes Eq. 1 given the target input g_i, which is shown in Fig. 2(b), and the image itself.

$$\arg\min_{\mathbf{e}} \sum_i \sum_j z_{ij}(e_i - g_j)^2 + \lambda \sum_i \sum_j z_{ij}(e_i - e_j)^2 \quad (1)$$

$$\text{with } z_{ij} = exp(\frac{-||\mathbf{f}_i - \mathbf{f}_j||^2}{\sigma_a^2}) exp(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{\sigma_s^2}) \quad (2)$$

Where $\mathbf{e}$ is the vector composed of e_i. i, j represent each pixel in the image. $\mathbf{f}_i$ is the texture feature vector for the area around pixel i, which is constructed with the RGB values of pixel i and the RGB averages of the surrounding eight pixels in this paper. (Note that PG is adaptive to other potential texture features.) $\mathbf{x}_i$ denotes pixel i's 2D coordinate. In the first term of Eq. 1, z_{ij} encourages g_i to propagate to nearby pixels with similar textures. Smoothing is achieved in the second term to ensure that in the propagation result pixels with similar textures or neighboring pixels will have close values.

Parameters controlling the objective function are taken directly from the implementation of [AP08]: σ_a^2 is set to be 0.2, σ_s^2 is set to be 0.1, while λ is adjusted to 0.1 for all the experiments in this paper.

2.2.2. Optimization

Eq. 1 is quadratic and can be optimized by least-square estimation; However, the difficulty arises when the resolution of the image increases. A 400×400 image will lead to calculating the inverse of a $400^2 \times 400^2$ matrix. We use the row-column sampling technique to approximate the calculation and readers are referred to [AP08] and [WS00] for details due to the space limitation.

2.2.3. Normalization

In practice, we found that normalization yields a better output because Eq. 1 lacks constraints on the variance of e_i. So after row-column approximation is done, we apply the following normalization operation to e_i:

$$l_i = \frac{e_i - \min_j e_j}{\max_j e_j - \min_j e_j} \quad (3)$$

$\mathbf{l}$ will denote the vector composed of l_i in the following sections.

2.3. Foreground Cropping

In GrabCut, every pixel that is marked by users are considered to be the ground truth for segmentation. In many scenarios, it is inconvenient for users to purify their inputs, so we propose a different solution for a more intelligent experience by cropping user inputs for the foreground.

As we already have the normalized propagation result $\mathbf{l}^f$ from the previous step, we model $\mathbf{l}^f$ as a GMM with equal weights:

$$p(l_i^f) = \sum_{k=1}^{2} 0.5\mathcal{N}(l_i^f; \mu_k, \sigma_k^2) \text{ with } (\mu_1 < \mu_2) \quad (4)$$

edges	capacity	for
(p_i, p_j)	$\exp(-\beta D^2_{\{p_i,p_j\}})$	
(s, p_i)	$\exp(-\beta(K-1)^2)$	$i \in \mathcal{F}$
	$\gamma \times (1 - l_i^b)$	$i \in \mathcal{I} - \mathcal{F} - \mathcal{B}$
	0	$i \in \mathcal{B}$
(p_i, t)	$exp(-\beta(K-1)^2)$	$i \in \mathcal{B}$
	0	$i \in \mathcal{F}$
	$\gamma \times (1 - l_i^f)$	$i \in \mathcal{I} - \mathcal{F} - \mathcal{B}$

Table 1: *Edge capacity.*

We learn parameters $(\mu_1, \mu_2, \sigma_1, \sigma_2)$ through Expectation-Maximization and we exclude point m from the user-selected foreground if:

$$\Phi_{\mu_1,\sigma_1^2}(l_m^f) < 0.975 \tag{5}$$

Where Φ_{μ_1,σ_1^2} is the cumulative distribution function for Gaussian distribution, and 0.975 indicates a 95% confidence interval. The intuition behind is that μ_1, σ_1 denotes the cluster of background pixels and we should mark a pixel as unknown if it tends to be in the background. We show in Fig. 2(e) 2(f) the foreground labellings before and after this cropping process.

We also introduce new notations here: $\mathcal{I}$ denotes the set of all pixels in an image, $\mathcal{B}$ denotes the set of pixels marked by users as in the background and $\mathcal{F}$ denotes the set of user-annotated foreground pixels that is not excluded by Eq. 5.

2.4. Graph Cut Optimization

We first show how a directed graph $\{V, E\}$ is constructed followed by the underlying theory. For each pixel i, we add node p_i to V. For every two neighboring pixels i, j, we add edge (p_i, p_j) to E and define $D_{\{p_i,p_j\}}$ to be the Euclidean distance between their 3-dimensional RGB color vectors. K denotes the smallest distance among all neighboring pixels.The source and sink nodes s, t together with edges of $(s, p_i), (p_i, t)$ are also added to the graph. The capacity for each edge in the graph is described in Table. 1. It can be proven [BJ01] that the min-cut of the graph divides all the nodes and its corresponding pixels to two partitions S(on the source side) and T(on the sink side), which minimizes the following function:

$$\sum_{p_i \in S, p_j \in T, (p_i,p_j) \in E} \exp(-\beta D^2_{\{p_i,p_j\}}) - \gamma \left(\sum_{p_i \in S} l_i^f + \sum_{p_i \in T} l_i^b \right) \tag{6}$$

We design Table.1 so that the global minimum of Eq. 6 corresponds to an ideal segmentation and all pixels in the source partition form the actual foreground segmentation. The idea in Eq. 6 is to separate the foreground by cutting on the edges in the sample image (the first term) while maintaining the consistence in both the foreground and the background (the second term).

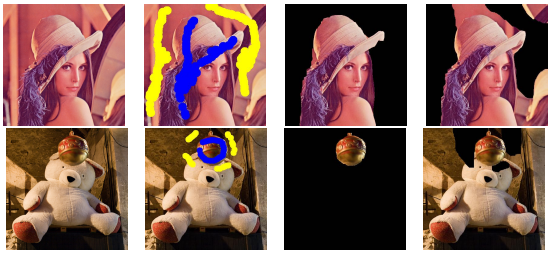
γ is empirically set to 1 in our experiments. The constant β is chosen to be $\left(2\mathrm{E}(D^2_{\{p_i,p_j\}})\right)^{-1}$, where $\mathrm{E}()$ is the expectation over all neighboring pairs in an image sample. We refer readers to [BJ01] for a more complete description on graph cut.

2.5. Refinement On User Input

PG allows user-assisted refinements on its output by adding pixels to $\mathcal{F}$ and $\mathcal{B}$ and rerunning the min-cut algorithm. We understand that sometimes refinements are still necessary for intricate images, but in this paper our only focus is to see how PG responds to the rough user inputs, rather than the segmentation after rounds of polishing. In the next section we show that PG could largely reduce the users involvement in producing high-quality cuts with fewer or no refinement efforts at all.

3. Evaluation

As suggested in Section 1, we intentionally pick images that has complex lighting conditions in our evaluation section. We implemented GrabCut for comparison purpose. Implementations for both PG and GrabCut do not include any refinement process.

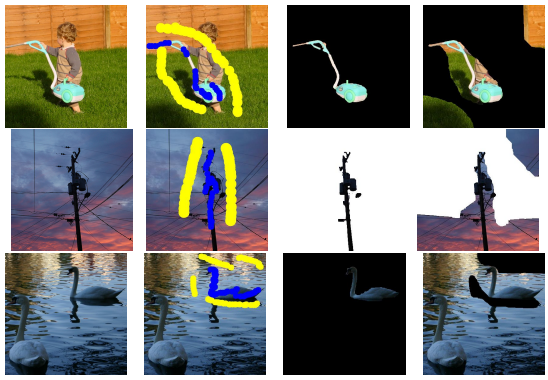


(a) (b) (c) (d)

Figure 3: *(a)Original images, (b)user inputs, (c)outputs from PG and (d)outputs from GrabCut.*

3.1. Segmentation on Intricate Images

We first show in Fig. 3 two interactive segmentation examples. We argue that these images are usually considered hard for interactive segmentation due to similar color tones in their foregrounds and backgrounds. A user attempts to cover more textures and structure information during the interaction but there is no refinement involved. PG performs reasonably well while GrabCut doesn't recognize the right foreground because its color cluster mechanism is not able to model such images. Similar results for GrabCut could be seen in [RKMB04].



(a) (b) (c) (d)

Figure 4: *(a)Original images, (b)user inputs, (c)outputs from PG, (d)outputs from GrabCut.*

3.2. Segmentation with Inaccurate User Input

Fig. 4 illustrates how PG handles inaccurate user inputs. The second row of Fig. 4, for instance, may require tedious polishing iterations with other state-of-the-art segmentation software. In our case, the user experience is more appreciating because coarse strokes are acceptable for PG.

The third row in Fig. 4 are the most challenging image in our testing set. The textures on the water surface exponentially increase the problem difficulty. PG texture measurements eliminate the inference of the reflection, and PG geometric measurements prevent the other bird from being selected. The neck of the bird is correctly selected even when users' strokes are thicker than the neck.

3.3. Simplified User Input

PG also accepts inputs where users casually mark the background with a closed rectangle as in [RKMB04] for simple segmentation tasks. We treat the area inside the closed region as foreground pixels and feed them to PG automatically.

(a) (b) (c)

Figure 5: *(a)The image, (b)the user input for the background and (c)the output from PG.*

3.4. Grayscale Image Segmentation

PG can also be used to segment grayscale images, while GrabCut is designed only for color images. We simply use the value for each pixel and the values for its surrounding pixels as the texture feature vector in processing the example in Fig. 6, More sophisticated features may be beneficial.

(a) (b) (c)

Figure 6: *(a)The gray-scale image, (b)the user input and (c)the output from PG.*

3.5. Computational Complexity Analysis

Given an $m \times m$ image and a fixed row-column sampling rate, our system takes $O(m^3)$ operations in the whole process. We implemented a prototype system in Matlab, and the run-time for a 400×400 image is around 20 seconds due to many avoidable loops. We believe a C implementation will reduce the run-time dramatically and a shorter delay would allow an iterative workflow for users.

4. Conclusion

In this paper, we demonstrate a novel interactive segmentation system which is capable of handling a wide range of images relying only on coarse user strokes. Advantages are its robustness to user error, the embedded appearance propagation process, and compelling results without refinement on intricate images. Future works are a C implementation of the system and complete evaluations on recognized segmentation testing databsets.

References

[AP08] AN X., PELLACINI F.: AppProp: all-pairs appearance-space edit propagation. In *International Conference on Computer Graphics and Interactive Techniques* (2008), ACM New York, NY, USA.

[BJ01] BOYKOV Y., JOLLY M.: Interactive graph cuts for optimal boundary and region segmentation of objects in ndimages. In *International Conference on Computer Vision* (2001), vol. 1, Vancouver, BC, Canada, pp. 105–112.

[BRB*04] BLAKE A., ROTHER C., BROWN M., PEREZ P., TORR P.: Interactive Image Segmentation Using an Adaptive GMMRF Model. *LECTURE NOTES IN COMPUTER SCIENCE* (2004), 428–441.

[FZ05] FREEDMAN D., ZHANG T.: Interactive graph cut based segmentation with shape priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (2005), vol. 1.

[LSTS04] LI Y., SUN J., TANG C., SHUM H.: Lazy snapping. In *International Conference on Computer Graphics and Interactive Techniques* (2004), ACM Press New York, NY, USA, pp. 303–308.

[PL07] PELLACINI F., LAWRENCE J.: AppWand: editing measured materials using appearance-driven optimization. In *International Conference on Computer Graphics and Interactive Techniques* (2007), ACM Press New York, NY, USA.

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: " GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG) 23*, 3 (2004), 309–314.

[RKMB04] ROTHER C., KOLMOGOROV V., MINKA T., BLAKE A.: Cosegmentation of Image Pairs by Histogram MatchingâIncorporating a Global Constraint into MRFs. In *Proc. CVPR* (2004).

[WS00] WILLIAMS C., SEEGER M.: Using the Nystrom method to speed up kernel machines. In *International Conference on Machine Learning* (2000).