Error-tolerant Scribbles Based Interactive Image Segmentation

Junjie Bai Xiaodong Wu
Department of Electrical and Computer Engineering, The University of Iowa, Iowa City, IA, US
junjie-bai, xiaodong-wu@uiowa.edu

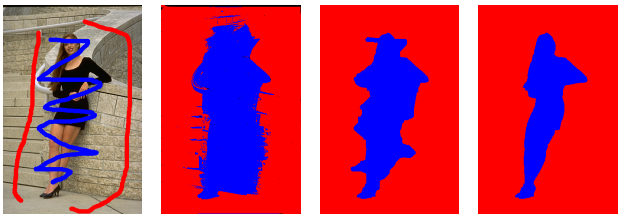
Abstract

Scribbles in scribble-based interactive segmentation such as graph-cut are usually assumed to be perfectly accurate, i.e.*, foreground scribble pixels will never be segmented as background in the final segmentation. However, it can be hard to draw perfectly accurate scribbles, especially on fine structures of the image or on mobile touch-screen devices. In this paper, we propose a novel ratio energy function that tolerates errors in the user input while encouraging maximum use of the user input information. More specifically, the ratio energy aims to minimize the graph-cut energy while maximizing the user input respected in the segmentation. The ratio energy function can be exactly optimized using an efficient iterated graph cut algorithm. The robustness of the proposed method is validated on the GrabCut dataset using both synthetic scribbles and manual scribbles. The experimental results show that the proposed algorithm is robust to the errors in the user input and preserves the "anchoring" capability of the user input.*

1. Introduction

Image segmentation/object selection is widely used in image processing. While fully-automatic segmentation methods can provide satisfactory result in some cases, human interaction is needed to produce high quality segmentation in more challenging images. Among various interactive approaches, two of the most popular ones are the boundary-based segmentation[9] and the scribble-based segmentation[11, 12, 1, 6, 8, 7]. The boundary-based interactive segmentation such as intelligent scissors [9] requires the user to trace the whole boundary of the object, which is usually time-consuming and tedious for users. Scribble-based interactive segmentation, on the other hand, is based on a number of foreground and optionally background scribbles. The algorithm will automatically label the pixels as either foreground or background based on the information such as location, color, texture, *etc*. provided by the scribbles.

Classical scribble-based interactive segmentation takes the scribbles as hard constraints, *i.e*., all foreground and background scribbles are guaranteed to be foreground and background, respectively, in the segmentation results. This requires the scribbles to be highly accurate, otherwise the segmentation gets compromised. This requirement can be hardly met on the mobile touch-screen devices, which has increasingly found wide applications. Even on a big screen with a mouse, it is hard to draw perfectly accurate scribbles on challenging images with fine structures, such as a thin bush stem or legs of a table. The scribble-based approaches have been widely used in image editing [1, 6] and image segmentation [2]. In the interactive image editing, users first specify sparse scribbles and the corresponding edits to be performed on each scribble, such as tone, color and/or material changes. These edits are then propagated to all the other pixels in the image with a modulated "editing strength", which can be seen as a certain soft segmentation.



(a) scribbles (b) Subr's [12] (c) graph-cut (d) proposed

Figure 1. The proposed method tolerates errors in user-specified scribbles and is better at segmenting fine structures in an image.

An *et al*. [1] formalizes the image editing problem as a quadratic optimization problem based on the pixel affinity matrix in the pixel appearance space. The affinity matrix is defined on all pairs of pixels. A low-rank stochastic approximation is applied to obtain an approximate solution. This method does not build an explicit appearance from the whole set of scribbles, instead it builds an implicit model by propagating information between all pairs of pixels. This method relies on that information propagation among all pairs of pixels to tolerate user input errors. Li *et al*. [6] observed that the optimization formulation in An *et al*.'s work [1] essentially is a smooth function with a sparse set of

constraints. Based on this observation, they approximately decomposed the given editing strength on the scribble pixels into a series of radial-based editing functions. The editing strength on all the other pixels are then interpolated using these radial-based editing functions for their appearance representation. This method runs extremely fast. However, the quality of the results highly depends on the representation capability of the series of radial-based editing functions for the user's intentions. In addition, both An *et al.*'s and Li *et al.*'s methods only produce a continuous "editing strength" map, which reflects how similar each pixel is to the foreground seed, instead of binary segmentation.

Various methods have been proposed to alleviate the problem of user input errors in the binary image segmentation. Liu *et al.*'s method [8] allows the user to override the erroneous scribbles by specifying new scribbles, which overlap partially with the inaccurate old scribbles. The new scribbles are then enforced as new hard constraints while the old scribbles are regarded as soft constraints for the segmentation. Clearly, this method still highly replies on the accuracy of the new scribbles. Sener *et al.* [11] developed an error-tolerant interactive segmentation method using dynamic and iterated graph-cuts. Essentially, the method removes the inaccurate scribble pixels from being used as seeds with some heuristics in the preprocessing step. Subr *et al.* [12] make use of a dense conditional random field (CRF) to infer the segmentation from possibly inaccurate scribbles. The dense CRF model contains a simple unary term and a fully connected CRF among all pairs of pixels in the image. To solve the dense CRF, they embedded pixels in a low-dimensional Euclidean space with a metric that approximates the desired high-dimensional affinity function.

We introduce a novel ratio-form energy function which consists of a graph-cut energy term to utilize both region and boundary information from the input image, and a user-scribble utility term to encourage the user scribbles to be respected. Essentially, optimizing the ratio energy function is equivalent to minimizing the graph-cut energy while at the same time respecting the user input as much as possible. The user scribbles are enforced as a soft constraint instead of a hard constraint, which allows the proposed method to tolerate user input errors. In contrast to the methods which deal with user-input errors using heuristics such as Sener *et al.*'s work [11], a global optimization framework is utilized to handle the user input errors. Comparing to the fully connected CRF method [12], our method enjoys the sparsity of the constructed graph from the neighborhood setting, as in the graph cut method [2]. Our experiments demonstrated that the energy function in the proposed method can be optimized efficiently and can produce spatially coherent segmentations.

2. Methods

We formalize the segmentation problem as the optimization of a ratio energy in which the numerator is the graph-cut energy and the denominator is a utility function which increases as more user input is respected.

2.1. Energy Formulation

The energy we aim to minimize is

$$E(\mathbf{x}) = \frac{E_{gc}(\mathbf{x})}{M + U(\mathbf{x})}, \tag{1}$$

in which $\mathbf{x} \in \mathcal{L}^{\mathcal{P}}$ is the labeling of all the pixels $\mathcal{P}$ from a binary set of available labelings $\mathcal{L} = \{`ob', `bg'\}$.

$E_{gc}(\mathbf{x})$ is the graph-cut energy [2], which consists of a region term $D_p(x_p)$ and a boundary term $V_{pq}(x_p, x_q)$ (Eq.(2)). The region term measures how likely each pixel p belongs to object ('ob') or background ('bg'). Unlike classical graph-cut [2] which assigns infinite region term weight to seed pixels to ensure them as hard constraint, we assign region term weights to seed pixels just like any other non-seed pixels. Thus no hard constraint is enforced in Eq.(2). The boundary term $V_{pq}(x_p, x_q)$ penalizes the discontinuity between the object and background, that is, $V_{pq}(x_p, x_q)$ is the penalty of assigning labels x_p and x_q to two neighboring pixels p and q according to the neighborhood setting $\mathcal{N}$. We use 8-neighborhood setting for the rest of the paper. More precisely,

$$E_{gc}(\mathbf{x}) = \sum_{p \in \mathcal{P}} D_p(x_p) + \eta \sum_{(p,q) \in \mathcal{N}} V_{pq}(x_p, x_q), \tag{2}$$

where, η is a balancing constant between the region term and the boundary term. $V_{pq} = 0$ if $x_p = x_q$, and $V_{pq} > 0$ if $x_p \neq x_q$.

$U(\mathbf{x})$ is a nonnegative utility function which increases as more user input information is respected in the segmentation result. Assume that $\mathcal{S}_F$ and $\mathcal{S}_B$ are the sets of pixels included in the foreground and background scribbles, respectively. Denote by $d_b(x_p)$ the distance between pixel p and the nearest scribble boundary. Two user input utility functions that we use in this paper are defined, as follows.

$$U_1(x_p) = \begin{cases} 1 & \text{if } p \in \mathcal{S}_F \text{ and } x_p = `ob' \\ 1 & \text{if } p \in \mathcal{S}_B \text{ and } x_p = `bg' \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$U_2(x_p) = \begin{cases} d_b(x_p)^2 & \text{if } p \in \mathcal{S}_F \text{ and } x_p = `ob' \\ d_b(x_p)^2 & \text{if } p \in \mathcal{S}_B \text{ and } x_p = `bg' \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The first utility function $U_1(x_p)$ simply counts the number of the user scribble pixels that are respected in the final segmentation. This utility function is used when large amount

of user input error is expected (such as in the image editing application), since in this case we usually do not know which portion of seed is more important than the other portions. The rational behind the design of the utility function $U_2(x_p)$ is that a user is more likely to draw a scribble whose centerline is correct while the boundary of the scribble has more bias to be wrong (such as in the image segmentation application). This utility function is used when we expect the user to make small mistakes that mostly happen around the scribble boundary. For example, a careful user may rarely make any mistake except at drawing scribbles on very thin structures such as table legs. More sophisticated $U(\mathbf{x})$ can be designed as long as it increases while more user input information is respected, and is nonnegative for all configurations due to optimization consideration.

M is a constant that controls how "flexible" the method is w.r.t. user-specified scribbles. The larger M is, the more scribble pixels are likely to be allowed foreground-background swap in the segmentation. To see this, imagine the extreme case in which $M \gg U(\mathbf{x})$, then essentially the denominator of Eq.(1) is constant M and we are just optimizing the numerator, which is the graph-cut energy. Note the seed pixels are regarded equivalent to those non-seed pixels in numerator. In this case, seed pixels have no special roles at all in the energy function, and cannot "anchor" segmentation anymore. In our experiment, we set M to be a multiple of the maximum possible utility function value.

$$M = \alpha \sum_{p \in \mathcal{P}} [U(x_p = \text{`}ob') + U(x_p = \text{`}bg')] \qquad (5)$$

Note that by minimizing the energy function in Eq.(1), we attempt to minimize the graph-cut energy $E_{gc}(\mathbf{x})$ while maximizing the respect to the user input scribbles. The optimization process is to find out the "best" set of scribble pixels such that the swap of their foreground-background labels enables the maximum reduction of the energy $E_{gc}(\mathbf{x})$ (*i.e.*, those erroneously identified as foreground or background scribbles), thus achieving our goal of error-tolerance.

2.2. Optimization of Ratio Energy

We use Newton's method for ratio optimization [3] to minimize the ratio energy function $R(\mathbf{x}) = P(\mathbf{x})/Q(\mathbf{x})$, in which functions $P, Q : \mathcal{X} \rightarrow \mathbb{R}, \mathcal{X} = 2^{\mathcal{V}}$ and $Q(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \mathcal{X}$. The main idea is to iteratively minimize a related linear function instead of the ratio function until convergence. The optimal solution of the ratio energy is given by the optimal solution of the linear function after convergence. The related linear function, which is called λ-*function*, is defined as

$$E_R^{\lambda}(\mathbf{x}) = P(\mathbf{x}) - \lambda Q(\mathbf{x}) \qquad (6)$$

More formally, the Newton's method for ratio optimization is defined in Alg.1. Theorem.1 claims the correctness of the algorithm. In our experiment, Alg.1 always converged in a few iterations (less than 5 iterations).

Input: Min-ratio problem $\min_{\mathbf{x} \in \mathcal{X}} R(\mathbf{x})$
Output: Opt sln $\mathbf{x}^*$ for $\lambda^* = \min_{\mathbf{x} \in \mathcal{X}} R(\mathbf{x})$
Select some $\mathbf{x}^0 \in \mathcal{X}$. $\lambda^0 \leftarrow R(\mathbf{x}^0)$. $k \leftarrow 0$.
while $\lambda^k \neq \lambda^{k-1}$ **do**
  Compute $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x} \in \mathcal{X}} E_R^{\lambda^k}(\mathbf{x})$ and
  $\lambda^{k+1} \leftarrow R(\mathbf{x}^{k+1})$
  $k \leftarrow k + 1$
end
return $\mathbf{x}^* = \mathbf{x}^k$ and $\lambda^* = R(\mathbf{x}^*)$
Algorithm 1: Newton's method for ratio optimization

Theorem 1. *[3] Algorithm 1 outputs an optimal solution to minimizing $R(\mathbf{x}) = P(\mathbf{x})/Q(\mathbf{x})$. In addition, the generated sequence $\{\lambda^k\}$ is strictly decreasing,* i.e.*, $\lambda^{k+1} < \lambda^k$.*

For Alg.1 to work properly, the λ-function in Eq.(6) needs to be efficiently optimized. Note that in Eq.(1), $P(\mathbf{x}) = E_{gc}(\mathbf{x})$ is the graph-cut energy consisting of a unary term and a pairwise term, and $Q(\mathbf{x}) = M + U(\mathbf{x})$ is a unary function of $\mathbf{x}$. Thus, the λ-function, as shown in Eq.(7), consists of only unary terms and a pairwise term, which can be optimized by the graph-cut method. Since M is a constant, the removal of M in the linear form of the optimization problem does not affect the final solution.

$$E^{\lambda}(\mathbf{x}) = E_{gc}(\mathbf{x}) - \lambda U(\mathbf{x}) \qquad (7)$$

As a result, we will use graph-cut to optimize λ-function in each iteration. Instead of computing a new max-flow from scratch, we used the dynamic graph-cut implementation [4] to reuse results from previous iteration as an initialization. Note we do not need to modify the weights of all arcs in the graph for each iteration. Since $U(\mathbf{x})$ is nonzero only for those user scribble pixels, thus, only the weights of those arcs associated with scribble pixels need to be updated in each iteration, and there are no weight changes for all the other arcs from one iteration to another. Thus, the overhead of updating arc weights are pretty light during the whole optimization process, and the algorithm runs efficiently in practice. Fig.2 shows the graph construction.

3. Experiments

3.1. Experiment Settings

To validate our algorithm, we use GrabCut dataset [10], which contains 50 images including a variety of objects such as person, car, goat, *etc*. Ground truths for all 50 images are available.

In this paper, we used some simple cost function designs for each term in the energy function to achieve our

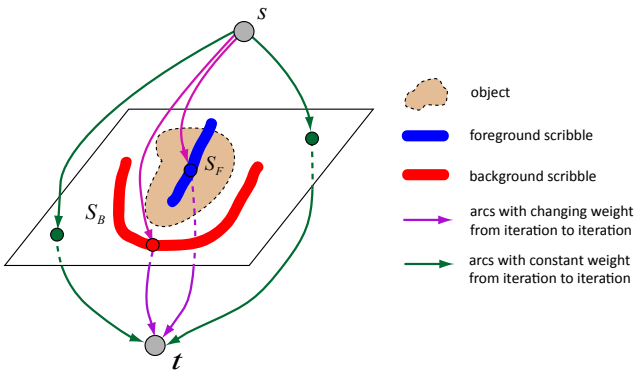


Figure 2. Only arcs connected to scribble pixels need to update their weights in each iteration. All other arcs have constant weights during all iterations. (Best viewed in color.)

goal of proof-of-the-concept. More comprehensive cost designs will likely improve the performance of our proposed method. The region term is generated by building two Gaussian Mixture Models (GMM) for foreground and background, respectively, in the Lab color space. These two GMM models are then applied on all pixels to generate the region term. The boundary term is obtained by computing the gradients on a smoothed image with the bilateral image filtering. For the weighting coefficient η between the region term and the boundary term in the graph-cut energy in Eq.(2), we set it to be 1 for all of our experiments. For those experiments in which the number of the erroneous scribble pixels is expected to be small (Sec.3.3), we use the utility function Eq.(4). Otherwise, we use the utility function Eq.(3). The coefficient α in Eq.(5) is set to be 1 for the experiments in Sec.3.3, to strongly enforce the user scribbles as anchor points. It's set to be 10 for those experiments in Sec.3.2 to tolerate large synthetic scribble errors, and is set to 100 for the experiments in Sec.3.4 to allow even larger and more spatially coherent scribble errors. We will discuss the issue of choosing appropriate α parameter in Sec.4.

Two metrics are used to measure the segmentation accuracy: the labeling accuracy and the Dice similarity coefficient. The metric labeling accuracy is defined as percentage of pixels correctly labeled in the final segmentation. More precisely, assume that TP, TN, FP, and FN represent the number of pixels that are true positive, true negative, false positive, and false negative, respectively, in the segmentation. Then the labeling accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$. The second metric is the Dice Similarity Coefficient (DSC), which measures the overlap between two segmented object volume. Suppose we have two segmentations O_1 and O_2 for the object, the DSC between the two segmentations is defined as $\frac{2|O_1 \cap O_2|}{|O_1|+|O_2|}$, or equivalently $\frac{2TP}{2TP+FP+FN}$.

Compared to the labeling accuracy, the computation of DSC does not rely on TN, which means that DSC is not sensitive to large areas of background in the image. For an image with large background, a segmentation even assigning every pixel to be background will have high labeling accuracy. However, DSC is able to tell that there is zero overlapping between the segmented object and the ground truth in this case.

To demonstrate our algorithm's ability to handle user input errors (*i.e.*, the user scribble pixels with wrong foreground/background labels), we compare our segmentation result to the classical graph-cut method [2] which regards the seeds (i.e., the user input scribbles) as a hard constraint. We also compare the proposed method to Subr *et al.*'s approach [12], which also tolerates the user input errors using an optimization framework. The author's publicly available code is used in our comparison. There are multiple tunable parameters in their implementation, an oracle is used to test all possible combinations of the parameters to find the one that results in best accuracy. The resulting set of parameters is used in the comparison.

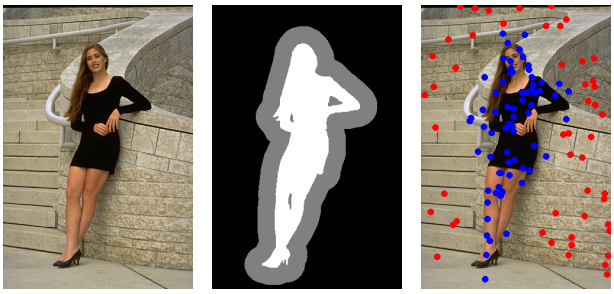
3.2. Experiment using Synthetic Scribbles

To quantitatively measure how robust the proposed algorithm is provided different degrees of user input errors, we follow the procedure used in Subr *et al.*'s work [12]. First 50 foreground pixels and 50 background pixels are randomly selected based on ground truth. They are assigned as foreground or background scribbles, respectively. Then an error-zone for each image is defined as background pixels that are less than a distance D from the foreground, in which D is defined as 5% of the image diagonal (Fig.3(b)). We randomly select 0 to 50 pixels in the error zone and assign them as foreground scribbles to simulate different degrees of user input errors.

Note in our experiment, the randomly selected points are dilated by a radius of 5 pixels before they are used as scribbles, in contrast to the isolated pixels used in [12]. Because isolated point cannot effectively anchor segmentation in the graph-cut method. Moreover, our manual scribble in the next experiment has a width of 10 pixels. Dilating the randomly selected pixels by a radius of 5 will result in a small circle that's similar to a point scribble drawn manually (Fig.3(c)). We randomly select 0, 5, 10, 20, 30, 40, 50 erroneous sample pixels from error zone to simulate the error percentage of 0%, 10%, 20%, 40%, 60%, 80%, 100% in the user input. Fig.3 shows one set of synthetic scribbles which contains 20 erroneous samples from the error zone.

The performance of each method is shown in Fig.4. We can observe that all methods perform quite well when no error is in user input. However, as more and more user input errors are added in the scribbles, the performances of graph-cut and Subr's method [12] get compromised quickly, while our method stays performing pretty well.

The reason that Subr's method's performance is worse than the graph-cut based approaches could be due to the use



(a) input (b) error zone (c) synthetic scribbles

Figure 3. Error zone mask and a set of synthetic scribbles. The foreground scribbles consists of 50 foreground pixels and 20 error zone pixels. The background scribbles consists of 50 background pixels.

of a fully-connected CRF model, instead of a sparse MRF model as used in the graph-cut based approaches. Thus, it generates a less coherent segmentation since it can easily propagate (wrong) information to remote pixels (see the third row in Fig.6). In fact, as more and more errors are added to the user input, Subr's method can quickly propagate those erroneous information to remote pixels instead of stopping the error in a small local region.

3.3. Experiment Aiming for High Accuracy

The scribbles drawn by users aiming high segmentation accuracy may not make as much mistake as shown in Fig.1(a). In this case, the user input error usually comes from drawing scribbles on fine structures such as the bush stem, vase handle, sheep legs shown in the second row of Fig.5. To validate our algorithm for this type of user input, a user was asked to draw scribbles on all 50 images in GrabCut dataset manually in a *natural* way, *i.e.*, the user neither intentionally makes mistake, nor makes excessive efforts to accurately draw the scribbles. As a result, no scribble errors are found in 20 images out of 50. The remaining 30 images have scribble errors in different degrees. On average, each of those 30 image scribbles contains 1.5% errors, *i.e.*, 1.5% of the foreground scribble pixels are actually background with respect to the ground true. The maximum scribble error is 8.4%.

The proposed robust segmentation (RS) method, the graph-cut (GC) method, and Subr's error-tolerant method [12] are used to segment those 20 images with *error-free* manual scribbles (reported as "RS/GC/Subr's(correct)" in Table.1), and those 30 images with *erroneous* manual scribbles ((reported as "RS/GC/Subr's(error)" in Table.1)). To gain better understanding about how the errors affect the performance of graph-cut, we use an oracle to correct the errors by removing erroneous foreground scribble pixels that are actually background. Then, the graph-cut method runs on those corrected scribbles and reported as "GC(err-corrected)" in Table.1. Note that this method is not really a fair reference. It can be excessively "accurate" because when the oracle removes the scribble pixels outside the object, it is actually drawing the perfect boundary locally using the ground truth.

Table 1. Manual scribble experiment result.

method (scribble type)	DSC (%)	label accuracy (%)
Subr's [12] (correct)	92.88	96.77
GC (correct)	97.64	98.91
RS (correct)	**97.65**	**98.92**
Subr's (error)	82.68	92.23
GC (error)	95.19	98.44
RS (error)	**95.26**	**98.46**
GC (err-corrected)	95.72	98.61
Subr's (all)	86.76	94.05
GC (all)	96.17	98.63
RS (all)	**96.22**	**98.64**
GC(all, err-corrected)	96.49	98.73

Table.1 shows the performance of each method. When no error happens, both the proposed method and the graph-cut method achieve high accuracy. But when the user scribbles contain errors, the proposed method performs better than the graph-cut method due to its tolerance of user input errors. The difference in accuracy metrics does not seem very high, which is understandable because users only make small errors around object boundary. However, the boundary accuracy of the segmentation indeed is improved by using the proposed robust segmentation method. Fig.5 shows the improved boundary by using the proposed method.

In the first column of Fig.5, the stem segmented by the proposed method is much thinner than graph-cut, which is closer to the ground truth. The proposed method also generates more accurate vase handle in the second column, more accurate sheep and person legs in the third and forth column. Subr's method performs even worse than the graph-cut method. It usually does not generate a spatially coherent segmentation and propagates errors in the segmentation to remote regions from the erroneous scribbles, as shown in the third row in Fig.5.

For the 30 images with erroneous scribbles, the proposed robust segmentation result corrected 60.1% of the erroneous scribble pixels on average.

3.4. Illustrative Results

Scribbles with large errors frequently happens on mobile touch-screen devices due to the use of less accurate pointing devices (such as fingers). In these experiments, the user scribbles contain much more errors. Illustrative results are shown in Fig.6. Subr's method can easily propagate foreground scribble errors to background due to its fully con-

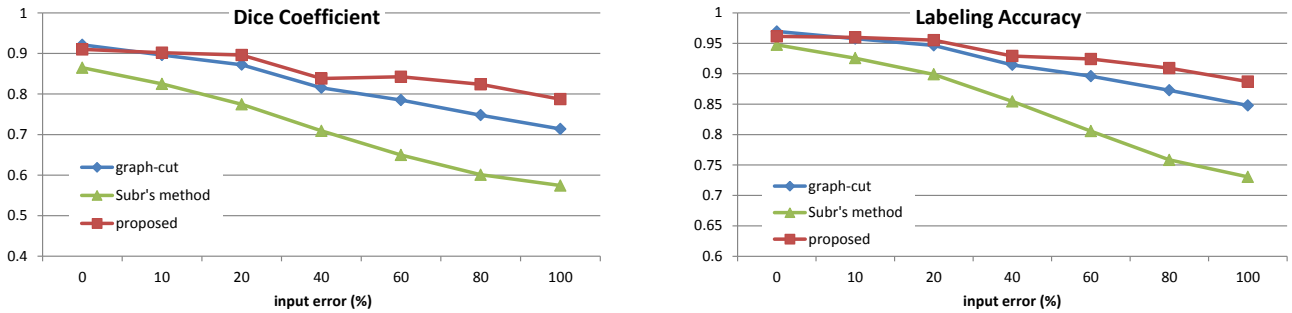


Figure 4. DSCs and the labeling accuracies of Subr's error-tolerant segmentation method, the graph-cut method, and the proposed robust segmentation method, given different percentage of errors in *randomly-generated synthetic* user input.

nected CRF formulation, as shown in the bear and the lady images. Another issues is that it does not generate spatially coherent segmentation for textured object, as shown in the grave tombstone and the sheep images. The graph-cut method uses the user scribbles as hard seeds, and thus is not able to correct errors in user input. In contrast, the proposed method performs very well in this type of situation.

One interesting case is the sheep (last column in Fig.6). In order to separate the two legs shown in the image, the user adds one background scribble between them. Graph-cut simply follows the boundary of the added background scribble to separate the two legs. Our proposed method, however, is able to reject part of the erroneous background scribble pixels and segments the two legs with higher accuracy.

4. Discussion

Ratio energy has been used for image segmentation in different ways. Wang et al.[13] used a ratio energy maximizing the average intensity difference between foreground and background for segmentation. Kolmogorov et al.[5] alleviates the shrinkage bias of graph-cut segmentation by using foreground volume as the denominator in the ratio energy. Here we use a different idea that the denominator is a utility function which encourages the user input to be respected as much as possible. How strong this encouragement is can be tuned by the parameter M in Eq.(1), or, equivalently α in Eq.(5). Thus, we can adjust the proposed method for users of different styles as we have shown in the different experiments in Sec.3.2, 3.3, 3.4.

To choose an appropriate α value, the basic guideline is that the larger α is, the more tolerant of user input errors the algorithm is. In Fig.7(b), although by using $\alpha = 1$ our method can tolerate the erroneous scribble at head, it does not tolerate many other errors of the scribble. In Fig.7(c), with $\alpha = 10$ the proposed method is doing a much better job. However, there is still erroneous segmentation at the elbow. In Fig.7(d), using $\alpha = 100$, our method achieves the best segmentation among the uses of those three α values.

On the other hand, a large α value means that the "anchoring" ability of the scribble pixels is reduced. In challenging images, the "anchoring" ability can be crucial to the correct segmentation. Fig.8 shows a challenge case of cheetah. When $\alpha = 1$, the paws and the tail of the cheetah are correctly segmented by following the guidance of the foreground scribbles. However, when $\alpha = 100$, the paws and part of the tails are incorrectly segmented as background.

Thus, for those images with the expectation of large user input errors, we can use large α value to accommodate those errors. For challenge images (e.g, with very fine structures, or similar foreground and background profile), the "anchoring" ability can be crucial to the accurate segmentation. Thus a small α value should be selected, and users are advised to provide more accurate scribbles in order to achieve an accurate segmentation.

However, we should note that no matter what α value is chosen, if the evidence for the foreground/background shown by the graph-cut energy is strong enough, the erroneous scribbles can be automatically corrected. Introducing parameter α allows us to control how to determine the evidence is strong enough.

5. Conclusion

We propose a novel ratio energy function to tolerate user input errors in scribble-based interactive segmentation. It aims to minimize a graph-cut energy which incorporates both region and boundary information and maximizes the portion of scribbles that are respected in the segmentation result. The ratio energy can be optimized exactly using an efficient algorithm. Experiments based on synthetic and manual scribbles are conducted to validate the algorithm's robustness to large amount of user input errors and the ability to achieve high segmentation accuracy when presented with user input errors. Promising results are shown in our the experiments.

Acknowledgment This work was supported in part by NSF grants CCF-0844765 and CCF-1318996.

Figure 6. Illustrative results on user scribbles with large amount of errors.

References

[1] X. An and F. Pellacini. AppProp: all-pairs appearance-space edit propagation. *ACM Transactions on Graphics (TOG)*, 27:40, 2008.

[2] Y. Boykov and G. Funka-Lea. Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006.

[3] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18:30, 1989.

[4] P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2079–2088, 2007.

[5] V. Kolmogorov, Y. Boykov, and C. Rother. Applications of parametric maxflow in computer vision. In *IEEE 11th Inter-*

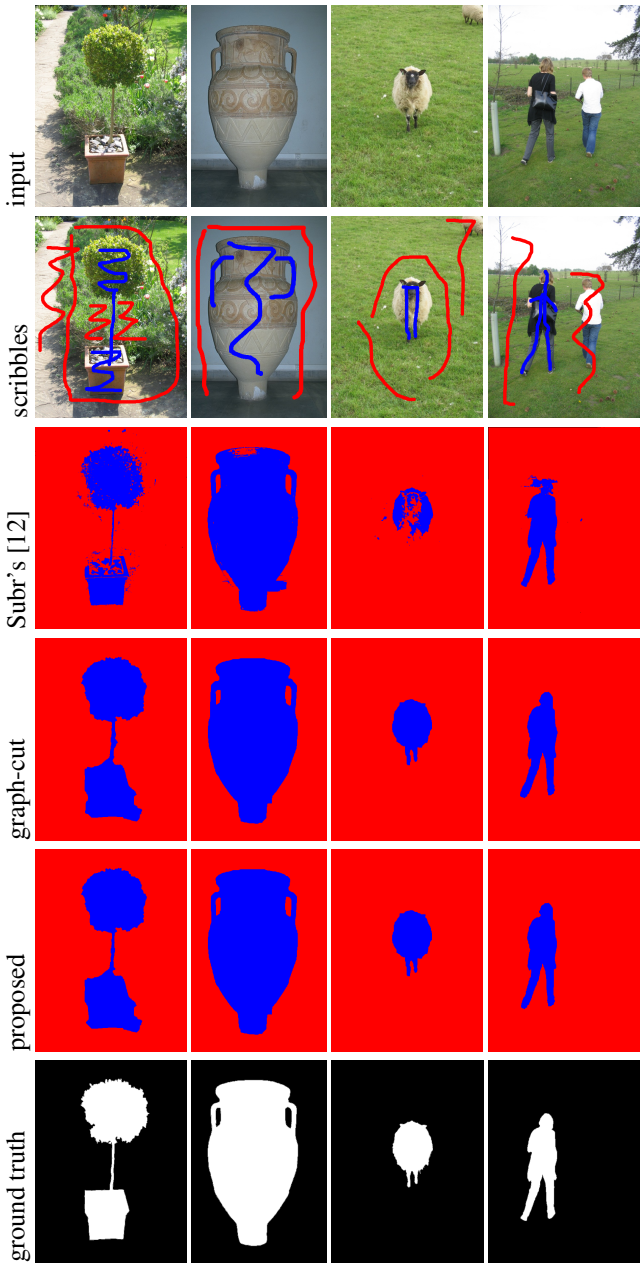
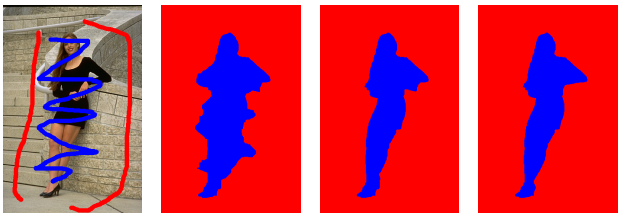
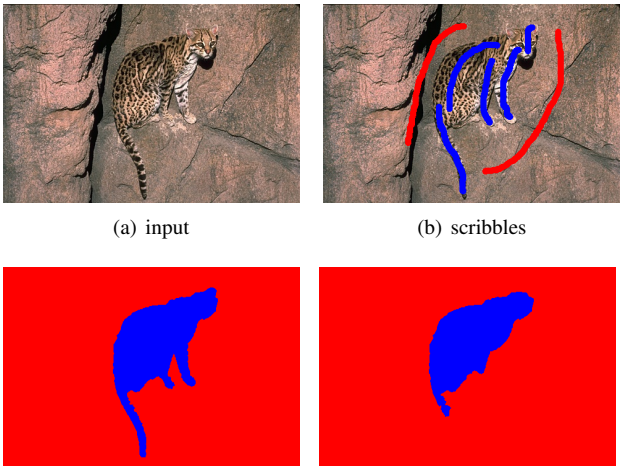


Figure 5. Improved boundary accuracy by using the proposed robust segmentation method. Note the proposed method improved boundary segmentation accuracy of the bush stem, vase handle, sheep legs and person's legs.



(a) scribbles (b) $\alpha = 1$ (c) $\alpha = 10$ (d) $\alpha = 100$

Figure 7. The larger α is, the more error-tolerance the algorithm can achieve.



(a) input (b) scribbles

(c) $\alpha = 1$ (d) $\alpha = 100$

Figure 8. The larger α is, the less "anchoring" ability the scribble has. Note when $\alpha = 100$, the paws and part of the tail of the cheetah is segmented as background although specified as foreground seeds.

national Conference on Computer Vision, 2007. ICCV 2007, pages 1–8, 2007.

[6] Y. Li, T. Ju, and S. Hu. Instant propagation of sparse edits on images and videos. *Computer Graphics Forum*, 29:2049–2054, 2010.

[7] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, Aug. 2004.

[8] J. Liu, J. Sun, and H. Shum. Paint selection. *ACM Transactions on Graphics (ToG)*, 28(3):69, 2009.

[9] E. N. Mortensen and W. a. Barrett. Intelligent scissors for image composition. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95*, 84602(801):191–198, 1995.

[10] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[11] O. Sener, K. Ugur, and A. Alatan. Error-tolerant interactive image segmentation using dynamic and iterated graph-cuts. *Proceedings of the 2nd ACM international workshop on Interactive multimedia on mobile and portable devices*, 2012.

[12] K. Subr, S. Paris, C. Soler, and J. Kautz. Accurate Binary Image Selection from Inaccurate User Input. *Computer Graphics Forum*, 32, 2013.

[13] H. Wang, N. Ray, and H. Zhang. Graph-cut optimization of the ratio of functions and its application to image segmentation. *ICIP*, 2008.