Error-tolerant Interactive Image Segmentation using Dynamic and Iterated Graph-Cuts

Ozan Şener
Middle East Technical University
Ankara, Turkey
ozansener@eee.metu.edu.tr

Kemal Uğur
Nokia Research Center
Tampere, Finland
kemal.ugur@nokia.com

A. Aydın Alatan
Middle East Technical University
Ankara, Turkey
alatan@eee.metu.edu.tr

ABSTRACT

Efficient and accurate interactive image segmentation have significant importance in many multimedia applications. For mobile touchscreen-based applications, efficiency is more crucial. Moreover, due to small screens of the mobile devices, error tolerance is also a crucial factor. In this paper, a method for interactive image segmentation, tailored for mobile touch screen devices, is proposed. As an interaction methodology, coloring is presented. An automatic stroke-error correction methodology to correct the inaccurate user interaction is also proposed. For the efficient computation of the solution, a novel dynamic and iterative graph-cut solution is formulated. Efficiency and error tolerance of the proposed method are tested by using various sample images. Subjective evaluation of the interactive segmentation algorithms for mobile-touch screen is also performed. Indeed, for the challenging examples, the superior performance of the proposed method is obtained by the experiments.

Categories and Subject Descriptors

I.4.6 [**Image Processing and Computer Vision**]: Segmentation; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*input devices and strategies, interaction styles*

Keywords

Interactive Image Segmentation, Graph Cuts, Mobile Devices

1. INTRODUCTION

Extracting the object of interest from the non-trivial background is a crucial step in many interactive multimedia applications. Although, fully automatic image segmentation algorithms have been improved significantly, it is still not possible to apply an automatic image segmentation algorithm with a guaranteed performance in the general case. Therefore, interactive image segmentation algorithms are becoming more popular.

Robust interactive segmentation algorithms are desirable in any application domain; however, applications on mobile touch-screen devices put extra constraints on the algorithm. Due to the inaccuracy of user inputs on small screens, such algorithms should be able to tolerate some errors. Moreover, it is also attractive not to include any correction stage in the algorithm in order to avoid tedious zooming and correction processes. In this paper, we propose an interactive image segmentation algorithm satisfying all of the these constraints.

Interactive segmentation methods in the literature can be divided into two main classes, as boundary-based [13, 17] and region-based methods [19, 18, 3, 5, 14, 15]. Boundary-based methods require the user to select an approximate boundary around the object; then, try to find the correct boundary. Such methods accomplish their goal by either minimizing an energy function locally via graph search [17] or minimizing an energy function globally by using graph-cut and then letting user perform local changes on the result [13]. Main drawback of such algorithms is their convergence to a local minimum in highly textured regions due to the high edge profile.

On the other hand, region-based methods either exploit region grow/merge like methods or region cut like approaches. Region grow/merge type of methods starts from the interacted regions and try to enlarge this region by the help of a measure using color or texture profile. In [18], color histogram based similarity measure is used to perform the maximal similarity based merging after oversegmentation of an image. In [5], information theoretical measure is used to merge pixels. Moreover, in a different approach [15], merging performance is improved by using an automatic refinement of suspicious boundaries. The main advantage of these algorithms is their computational efficiency. On the other hand, the solution, obtained by these methods, could be a local minimum of the specified energy function; therefore, it might not be the globally optimum solution.

Graph-cut is a powerful method which is capable of finding the global minimum of range of energy functions efficiently. Type of energy functions, which can be minimized globally, have been analytically obtained in [12]. The algorithm in [3] is one of the first algorithm that uses trimaps and probabilistic color models to interactively segment any given image. Due to their efficiency and performance, graph cuts have been used and improved in the literature. GrabCut [19] is an iterative version of graph-cut formulation, which works

IMMPD'12, November 2, 2012, Nara, Japan.

efficiently by a small amount of interaction. On the other hand, color models are also improved by using discriminative clustering of colors [14]. In [2], a method for the estimation of graph parameters is further proposed.

Isoperimetric graph partitioning [9] is another graph based segmentation method defined as finding the segment with minimum isoperimetric constant. Although it was proposed as an automatic image segmentation algorithm, it has also been used as an interactive image segmentation algorithm [8, 10]. In [9], a graph partitioning problem is defined as finding the partition with the minimum isoperimetric constant. Then the partition having the minimum isoperimetric constant is obtained via solution of a linear system. On the other hand, this method requires selection of ground node. Although it can be selected automatically, supervised selection of ground nodes results in interactive image segmentation algorithm [10].

Among the aforementioned algorithms, the proposed method is mostly related to [3] and [19]. Our main contributions over these algorithms can be stated as follows: 1) A novel method to detect and correct the erroneous user interaction; 2) Dynamic version of the graph-cut approach that decreases the computational complexity of the algorithm; 3) User friendly interaction method, namely coloring-.

2. PROPOSED METHOD

In most of the interactive image segmentation algorithms, a user first selects the object of interest by drawing strokes on the object and/or background or drawing a rectangle around the object; then, the user gives a command to execute segmentation. Then, the user might have a chance to correct possible mistakes. To the best of our knowledge, only exception to these types of interaction is paint selection tool [16]. Paint selection tool minimize the energy function locally and efficiently by using multi-core multi-level banded graph-cut and shows the result dynamically to the user. User draw scribbles for both foreground and background and can zoom-in and zoom-out.

The proposed method is a dynamic process. In the proposed algorithm, when the user selects a color image, the gray scale version of this image is initially displayed to the user. Then, the user starts to colourize the object of interest by the finger strokes on the screen. With each stroke, global segmentation is performed, and the result on the display is updated in real-time. Main difference between paint selection tool and proposed method is that our method does not require any scribbles on the background. In addition to these, our method finds the globally optimum solution of an energy minimization problem. On the other hand, paint selection tool uses an approximate energy minimization method. In addition to these, classical mouse-based interfaces use left click for foreground and right click for background. However, in touch based interfaces, there is only finger stroke. Therefore, using only foreground (or only background) scribbles is crucial for the user-centered interaction.

From the correction point of view; in the proposed method, a user always has a chance to correct foreground classified as background. However, user might not have a chance to correct background classified as foreground without restarting the algorithm from scratch. Therefore, it is better solving these types of errors before they occur. In our experiments, we also observed that this type of errors caused by interaction errors. For this purpose, we propose an algorithm to classify and correct erroneous interactions in Section 2.3.

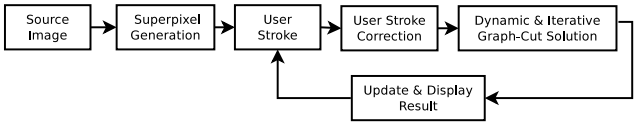


Figure 1: Block diagram of the overall algorithm.

In order for coloring to be an intuitive gesture, effects of scribbles should be local. In order to satisfy this locality requirement, with each stroke on the screen, only the results of neighbouring pixels are updated, although the segmentation for the whole image is solved globally. Although this locality increase the time spent to segment a single image, it increases the quality of the interaction. Moreover, in order to decrease computational complexity, input image is initially over segmented by using SLIC [1] algorithm. After this oversegmentation step, the image is represented as a graph of superpixels and all the remaining algorithms are applied to the graph of these superpixels. In Figure 1, complete image segmentation method is summarized as a block diagram.

2.1 Image Segmentation using Graph-Cut

Although the interaction methodology and the solution method differs, representation of the energy function of the proposed algorithm is still based on [3] and [19]. In [19], an input image is represented as a color vector of the form $\mathbf{z} = (z_1, ..., z_n, ..., z_N)$. We modify this representation of an image from pixel-based to superpixel-based. In our framework, z_i is the concatenated color vectors of the pixels of superpixel i. Segmentation of the image is represented as a binary vector of form $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n, ..., \alpha_N)$ with $\alpha_i = 1$, if the superpixel i is foreground and vice versa.

As the color model, Gaussian Mixture Model(GMM) is used; concatenation of mean and variance vectors of each Gaussian model is stored in a parameter θ. Then, the energy function, which is in the form of a Gibbs energy [7], is formulated as

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

In this energy function, $U(\boldsymbol{\alpha}, \theta, \mathbf{z}, \mathbf{k})$ corresponds to a fit measure of the estimated color models θ to the segmentation mask $\boldsymbol{\alpha}$. This term is used as the sum of the fitness term of each superpixel and fitness term of each superpixel is represented by the average of fitness terms of its pixels. Therefore, if z_{n_i} represents the color vector of i^{th} pixel of n^{th} superpixel and $\|z_n\|$ represents the number of pixels in the superpixel n,

$$U(\boldsymbol{\alpha}, \theta, \mathbf{z}, \mathbf{k}) = \sum_n \frac{1}{\|z_n\|} \sum_{i \in n} D(\alpha_n, \theta, z_{n_i}, k_n)$$

where $D(\alpha_n, \theta, z_{n_i}, k_n)$ represents the fitness of the pixel color value to its label. Moreover, it is defined as the inverse of the conditional-log-likelihood of the color vector z_{n_i} as:

$$D(\alpha_n, \theta, z_{n_i}, k_n) = -\log p(z_{n_i} | \alpha_n, k_n, \theta)$$

where p is a normal distribution by an estimated mean vector and covariance matrix.

In order to learn the parameters $\theta = (\mu(\alpha, k), \Sigma(\alpha, k))$, at each step of the interaction, the interacted superpixels have been considered as foreground, while all the other superpixels have been considered as background. Then, model parameters are computed by using Expectation Maximization(EM) algorithm iteratively [19]. It should be noted that user selects the superpixels; therefore, even during the initialization of the algorithm, there is enough number of pixels to be used for EM algorithm.

On the other hand, $V(\boldsymbol{\alpha}, \mathbf{z})$ corresponds to the coherency of the segmented foreground and background. This term is a typical smoothness term defined via color profile of the foreground-background edge and defined as

$$V(\boldsymbol{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n)\in \mathbf{C}} [\alpha_m \neq \alpha_n] exp(-\beta dis(m,n))$$

where $[\psi]$ is the indicator function, which gives 1, if ψ is true and 0 otherwise. $\mathbf{C}$ represents the set of neighbouring superpixels. Moreover, $dis(m,n)$ represents the distance between superpixel m and n. This distance is taken as the Euclidean distance between mean color vectors of corresponding superpixels. Finally, the normalization constant β is selected as [3]:

$$\beta = (2 < dis(m,n) >)^{-1}$$

where $< . >$ denotes expectation over the all superpixel edges in the whole image. As explained in [3], global minimum of this energy function can be efficiently obtained via min-cut/max-flow method. Details on the minimization method with the proposed improvements are given in Section 2.2

2.2 Dynamic and Iterated Graph-Cut

Proposed algorithm is inherently iterative due to the proposed interaction methodology. By each interaction of a user, energy minimization is updated by the re-estimation of its parameters. Therefore, even in the case of using the original graph-cut, the proposed algorithm should work iteratively and dynamically. On the other hand, due to the high computational cost of the algorithm, iteration of the whole approach at each interaction does not seem possible. In order to solve this drawback, we use the residual graph concept [11] with a novel spatial dynamicity improvement.

Stated energy minimization problem can be converted to the min-cut/max-flow problem on two terminal (source and sink) graph $G(V, E)$, where V is set of nodes (superpixels) and E is set of directed edges. Global solution to the energy minimization problem is equivalent to the minimum cost cut that separates source and sink nodes in this graph. It is shown that, finding the minimum cost cut is equivalent to the determining maximum flow from source to sink. Moreover, solution to the max-flow problem is obtained by augment paths algorithm [4]. This algorithm can be explained by using the residual capacities and augmenting paths.

Residual capacity r_{ij} of edge $(i, j) \in E$ is the maximum additional flow that can be sent from i to j through edge (i, j). Initially, residual capacities are set as edge weights. The augmenting path is the path from source to sink through unsaturated residual edges. Augmenting paths algorithm [4] uses the fact that pushing any flow through an augmenting path does not change the solution. In other words, the solution (resulting minimum cut) to the original graph G, and the graph G' which results from pushing a flow through an augmenting path is equivalent. Augmenting flow algorithm [4] finds a valid path on the residual graph from source to sink and push the maximum possible flow that can go through active edges of the obtained path. This search and pushing flow steps are iterated until there exist no valid path. In other words, when the max-flow is obtained, residual graph is the graph with no possible augmenting path. Moreover, saturated edges correspond to the min-cut solution.

In the proposed method, the structure of the graph does not change at all, and the edge weights change slightly throughout the iterations. Therefore, if the residual graph of the previous iteration is used, computational burden might significantly decrease. In [11], a method for this edge update is developed. If a weight ,w_{ij}, of edge (i, j) is changed to w'_{ij}, then the solution to the new graph can be determined by solving the updated residual graph with update $r'_{ij} = r_{ij} + w'_{ij} - w_{ij}$. This updated graph results in the same result, since this update corresponds to pushing all the flows in the previous graph to the new graph. However, this update might result in negative edge capacities. Hence, a method to solve negative edges is also developed in [11].

2.2.1 Spatially Dynamic and Iterated Graph-Cut

Iterative solution [11] improves the time efficiency significantly but still there exist some room for additional improvement. In our interaction method, user colourize the object of interest locally; therefore, the solution required to be obtained should also be a local one. However, min-cut/max-flow solution is determined for the whole graph; therefore, there must be some redundant processing. A straightforward solution to this problem is solving the sub-graph including the user interaction. However, the performed experiments showed that finding a generic size for this sub-graph is not possible. Therefore, an adaptive method for finding an appropriate size of this subgraph is proposed.

Assume that smallest possible sub-graph (bounding box of interacted superpixels) is selected and graph-cut is run and the residual graph is obtained. Then, solution for a subset of connected nodes R having the same segmentation result can not be changed simultaneously by the external flow, if the condition in Equations (1a) and (1b) is satisfied. Simultaneous change corresponds to the flipping the label of all nodes in region R.

If R is foreground (connected to source)

$$\sum_{i\in R} w_{iS} - w_{iT} > \sum_{iR, j\notin R} w_{ij} \tag{1a}$$

If R is background (connected to sink)

$$\sum_{i\in R} w_{iT} - w_{iS} > \sum_{j\notin R, i\in R} w_{ji} \tag{1b}$$

where w_{iS} and w_{iT} denote the terminal weight of node i with source and sink respectively.

This condition holds since the cost of the changing the solution (cutting terminal edges) is larger than the cost of cutting all the non-terminal edges. Solution to the part of the nodes in R might still change; however, the result of the whole R can not change. In other words, only some part of the nodes in region R can flip their labels.

On the other hand, when the sub-graph and region R is selected, this condition can also be defined in terms of edge weights between sub-graph and the rest of the global graph. It should be noted that there is no available path within the sub-graph, since this conflicts with the augmenting paths algorithm [4]. Therefore, all the paths which change the solution should go through edges between the sub-graph and the rest of the global graph. Moreover, if the sum of the maximum flows through these paths is less than the terminal weights of the nodes, the resultant labelling can not be changed via enlargement of the sub-graph. Since, cost of the changing the solution (cutting only the terminal edges) is larger than cutting all edges between sub-graph and the rest of the graph.

The condition for the sub-graph case can be represented as follows, if N is the set of nodes, which are not in the sub-graph but neighbour to the nodes in the sub-graph, and $\exists Path(i,j)$ indicates the existence of an augmented path between i and j.

If R is foreground (connected to source)

$$\sum_{i \in R} w_{iS} - w_{iT} > \sum_{\substack{i \in R, j \in N \\ \exists Path(i,j), e \in E \cap Path(i,j)}} min(w_e) \qquad (2a)$$

If R is background (connected to sink)

$$\sum_{i \in R} w_{iT} - w_{iS} > \sum_{\substack{i \in R, j \in N \\ \exists Path(j,i), e \in E \cap Path(j,i)}} min(w_e) \qquad (2b)$$

Minimum weight in the path is used as flow value, since it corresponds to the maximum amount of the flow which can go through the path. It should be noted that, when the conditions in Equations (2a) and (2b) are satisfied, label of all the nodes in region R can not change simultaneously. However, the solution to the part of the nodes in R might still change.

If R is taken as single super-pixel and all the superpixels in the selected sub-graph satisfy this condition, the solution can not change, when sub-graph is enlarged. However, it is not efficient to check this condition. Indeed, this condition is too strict to be satisfied. One can relax this condition by using the fact that when the colored region is large enough, the solution to the bounding box of the interacted superpixels gives satisfying results. Therefore, our method need to handle only small sub-graph cases. For the small subgraphs, we experimentally observed that initial segmentation supplied by GMM result is reliable. When sub-graph is segmented via GMM such that if assigned GMM component is foreground, super-pixel is considered to be foreground and vice versa; the segmentation supplied by GMM is generally correct. Most of the error is actually caused by small foreground or background (a.k.a shrinking bias). Hence, we can safely argue that if both foreground and background generated by GMM satisfy Equations (1a) and (1b), selected sub-graph can be used. When the Equations (1a) and (1b) are satisfied, terminal edge weights are larger than non-terminal ones; therefore, there is no shrinking bias. On the other hand, using Equation (2a) and (2b) is expected to have better performance with its greater computational cost. Moreover, in our experiments, segmentation solution to the sub-graph obtained via both metrics yielded the same result as global segmentation in all test samples. Therefore, we concluded that both metrics can be used safely, and the computationally efficient one is preferred in the final algorithm. Hence, the resulting algorithm first starts with bounding box of user interaction, then enlarges this sub-graph until the condition in equations (1a) and (1b) is satisfied. In order to use residual graph idea, weights of the edges between selected sub-graph and the rest of the graph should be assumed 0. Moreover, when the graph is enlarged, these edges can be properly updated via the update rule in [11].

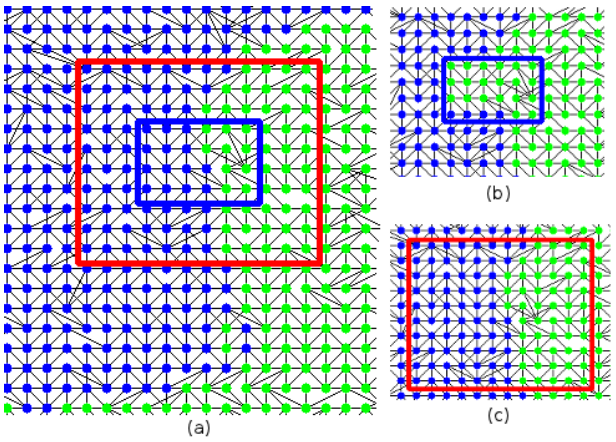


Figure 2: Visualization of automatic sub-graph finding.

Consider the case in Figure 2.a, where green nodes represent the ground-truth foreground and blue nodes represents the ground-truth background nodes. In Figure 2.a, the blue rectangle is the bounding box of the interacted superpixels, and this rectangle does not satisfy the condition. Enlarged version, which satisfies the condition, is computed via proposed algorithm and shown as red rectangle. The corresponding solutions are given in Figure 2.b and 2.c, respectively. In Figure 2.b and 2.c, green and blue nodes represent computed foreground and background nodes respectively. As presented in this figure, although the initial solution is erroneous, the enlarged rectangle leads to the correct solution.

2.3 Error Correction

Due to the small screens of the mobile devices, users generally make stroke errors during interaction and these errors typically occur around the boundary of the object of interest. In order to solve such interaction errors, we propose a correction method which is summarized in Algorithm 1. We assume that user starts interaction within the object. Then, the algorithm accumulates the color statistics of the current region in a single color Gaussian model. When user moves from current superpixel to a new one, algorithm checks the new superpixel. If new superpixel fits to the learned model, the algorithm accepts this new superpixel. If not, the algorithm stores the superpixel which user left the object. Then, new superpixels are stored in a temporary queue and not inserted to the algorithm. In the mean time, color model of these new superpixels are stored in another single color gaussian model. When another superpixel is examined, if this new superpixel fits to the previously learned color model, superpixels accumulated in the queue is discarded, and the correct path between

the superpixel which user left and returned back to the object is calculated and inserted in to the dynamic graph-cut. If the user also leaves the next region (multi-color case), the algorithm calculates the correct path and insert to the dynamic graph-cut. Leaving the next region means not fitting to the temporary color model. As a fit measure, Euclidean distance between color GMM means are used; i.e. $dist$(NewOS,ColorModel) $= |z_{NewOS} - \mu_{ColorModel}|$ and scalar multiple of standard deviation ($k\sigma_{ColorModel}$) is used as a threshold.

In order to find the correct path between the superpixel which user left the object and returned to object, a minimum cost path finding problem is defined and solved. Correct path is assumed to be the path between these two points which minimizes the following cost function;

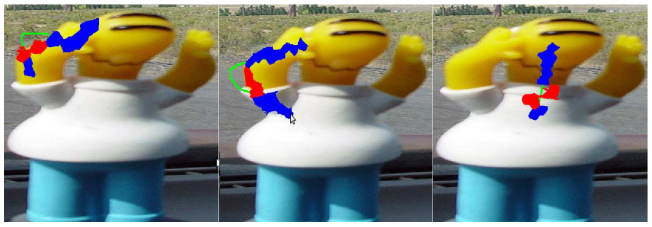
$$Cost(path) = \sum_{u,v \in path} |\overline{x_u} - \overline{x_v}| + \lambda |\overline{I_u} - \overline{I_v}| \qquad (3)$$

where, u and v are the nodes incident to the same edge in a path, $\overline{x_i}$ is mean position vector of superpixel i and $\overline{I_i}$ is mean RGB vector of superpixel i. Moreover, this minimum cost path is obtained via Dijkstra's algorithm [6] over the superpixel graph. In our experiments, we have fixed the value of parameter $\lambda = 0.5$.

Algorithm 1 User input correction algorithm

Initialization: $PossibleError \leftarrow 0$, **clear** TempQueue
InsertNewOversegment(NewOS):
if not $PossibleError$ **then**
  if $dist$(NewOS,CurrentMdl) $\leq k\sigma_{current}$ **then**
    insert NewOS **to** Dynamic-Graph-Cut
    update CurrentMdl **with** NewOS
  else
    $PossibleError \leftarrow 1$, LeftOS$\leftarrow$NewOS
    PrevMdl $\leftarrow$ CurrentMdl
  end if
else
  if $dist$(NewOS,PrevMdl) $\leq k\sigma_{prev}$ **then**
    CurrentMdl $\leftarrow$ PrevMdl, $PossibleError \leftarrow 0$
    FindPath(LeftOS,NewOS)
    insert founded path **to** Dynamic-Graph-Cut
  else if $dist$(NewOS,CurrentMdl) $\leq k\sigma_{current}$ **then**
    insert NewOS **to** TempQueue
    update CurrentMdl **with** NewOS
  else
    $PossibleError \leftarrow 0$, FindPath(LeftOS,NewOS)
    insert founded path **to** Dynamic-Graph-Cut
  end if
end if

In order to visualize the performance of the error correction algorithm, we summarize the main error correction scenarios in Figure 3a,3b and 3c on a sample image. In Figure 3a,3b and 3c; blue superpixels are the ones accepted as the correct user interaction, whereas green line is the discarded user interaction (considered as an interaction error). Moreover, red superpixels are obtained from minimum cost path finding solution. Therefore, only the blue and red regions are used for the dynamic and iterated graph-cuts algorithm. It should be noted that, these lines are not shown to the user. These markers are drawn to explain the algorithm.



(a) Error - 1 color (b) Error - 2 colors (c) No Error

Figure 3: Visualization of the proposed error correction algorithm

In the final version, only the current segmentation result is shown to the user.

In Figure 3a, user first left the object boundary accidentally and then returns back to foreground object pixels. The proposed algorithm discards the erroneous interaction and finds the correct path. In Figure 3b, the user left the object accidentally from the yellow coloured region, then comes back to a white coloured region. Proposed algorithm also handles this case successfully. In Figure 3c, user lefts the yellow region (multi-color case) and then continues along the white region. When the user enters the region with shadow, algorithm detects the color change and solve path finding problem. Indeed, the resulting path is also correct. In other words, in the case of the false error alarm, resulting minimum path is also expected to be contained in the object; therefore, false error alarm has no side effects for the algorithm.

3. EXPERIMENTAL RESULTS

The proposed algorithm is tested by using an extensive dataset with various color and texture profiles. Some of the interactions and their corresponding results are shown in Figure 4. Proposed method is compared against Intelligent Scissors [17], Grabcut [19] and isoperimetric segmentation [9]. The reason for selection of these algorithms is their interaction methodology. Using interactions, such as right-click and keyboard press, are not desirable in touch-screen application interfaces. Moreover, these algorithms are the ones which do not require such types of interactions. Therefore, these methods are assumed to be the only natively applicable ones to touch-screen based scenarios. In [19], a bounding box drawn around the object of interest is used as an interaction. In [17], a roughly drawn boundary of the object is used as an interaction. Moreover, in [9], scribbles on foreground are used as an interaction.

As it can be observed from the results in Figure 4, intelligent scissors algorithm [17] requires many seed points for robust operation. Even with dense input seed points; the performance is quite limited, if there exist local texture around the boundary. For all the test images, the resulting segmentation neither smooth nor correct. Grabcut [19] algorithm performs well, if the object of interest has a different colour characteristics than background, as in the case of last row. Performance of the Grabcut is limited in other cases. In Figure 4, in the first row, there is a second object in the rectangle apart from the object to be segmented. The algorithm could not separate objects, since both are contained in the rectangle. In the second row, color characteristics

Figure 4: Comparison of the interactive segmentation methods. Columns show interaction and corresponding result for Grabcut[19], Intelligent Scissors [17] and proposed method respectively.

of the head of the bird is similar to the background; therefore, head of the bird is segmented as background. Finally, in the third row, there is not enough background information; therefore, Grabcut fails to segment the object. Isoperimetric segmentation [9] algorithm is quite powerful for automatic segmentation scenario. However, selecting ground nodes is not a strong prior. Therefore, single stage segmentation with isoperimetric segmentation algorithm does not yield segments corresponding to the objects, especially in the case of complex backgrounds. Except the trivial case in the last row, isoperimetric segmentation algorithm fails for other test samples. The proposed method has superior performance in all the test examples. In the first and last rows, the proposed method yields smooth and correct segmentation results. In the second row, the foot of the bird is not segmented correctly due to the limitations of touch screen. However, proposed method still outperforms the others. In the third row, there is an erroneous additional small head around the shoulder of the cyclist. This artifact is caused by the shrinking bias.

3.1 Analysis of Computation Time

Computation time of the proposed method is compared against the min-cut/max-flow solution [4] and dynamic graph-cut [11] solution. Two different evaluations are performed. First, for the segmentation of a single image, computation time of three different algorithms are calculated for each iteration (each scribble of the user) of the algorithm. Then, plot of the computation time throughout the whole process is obtained as in Figure 5. Only one segmentation is performed, and interaction is dumped to a file. Then, this interaction is fed to all three algorithms. Therefore, in each iteration same interaction is fed to all algorithms.

Obviously the original min-cut/max-flow algorithm [4] has the largest computational time, since it is the baseline algorithm for other two. Dynamic graph-cut [11] starts with the full solution of graph-cut. After the initial iteration, dynamic graph-cut only updates the residual solution, hence it has a better time complexity. Moreover, proposed algorithm initially starts with a quite small sub-graph and it is much more efficient than other methods. Throughout the process, computation time of the proposed method increase, since interacted region and the resulting sub-graph enlarges. Eventually the sub-graph converges to the full graph and computation time of the proposed method converges to the dynamic graph-cut. Since the proposed algorithm enlarges the sub-graph until Equations (1a) and (1b) are satisfied, there are points which enlargement and augmenting flow algorithm performed many times. At those times, the proposed method spend more time compared to dynamic graph-cut as expected. This situation is observable in Figure 5 at the 13th iteration. For all other iterations, superior time performance of the proposed method is visible in this figure.

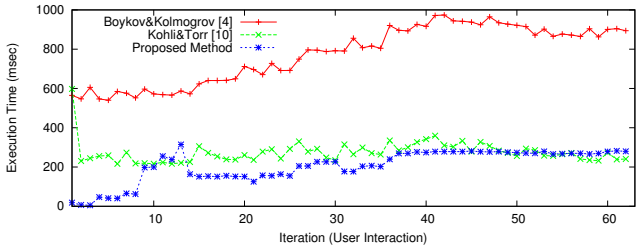


Figure 5: Execution times for each iteration

As a second test, a total of 50 segmentation for different images are performed, and average computation time of each algorithm is computed. The average computation time per interaction of these 3 algorithms is tabulated in Table 1. It is also clear from Table 1, the proposed algorithm yields the best computational time.

3.2 Analysis of Error Correction

The proposed error correction algorithm corrects errors before segmentation. Moreover, correction can also be incorporated within graph-cut. If hard constraints (terminal

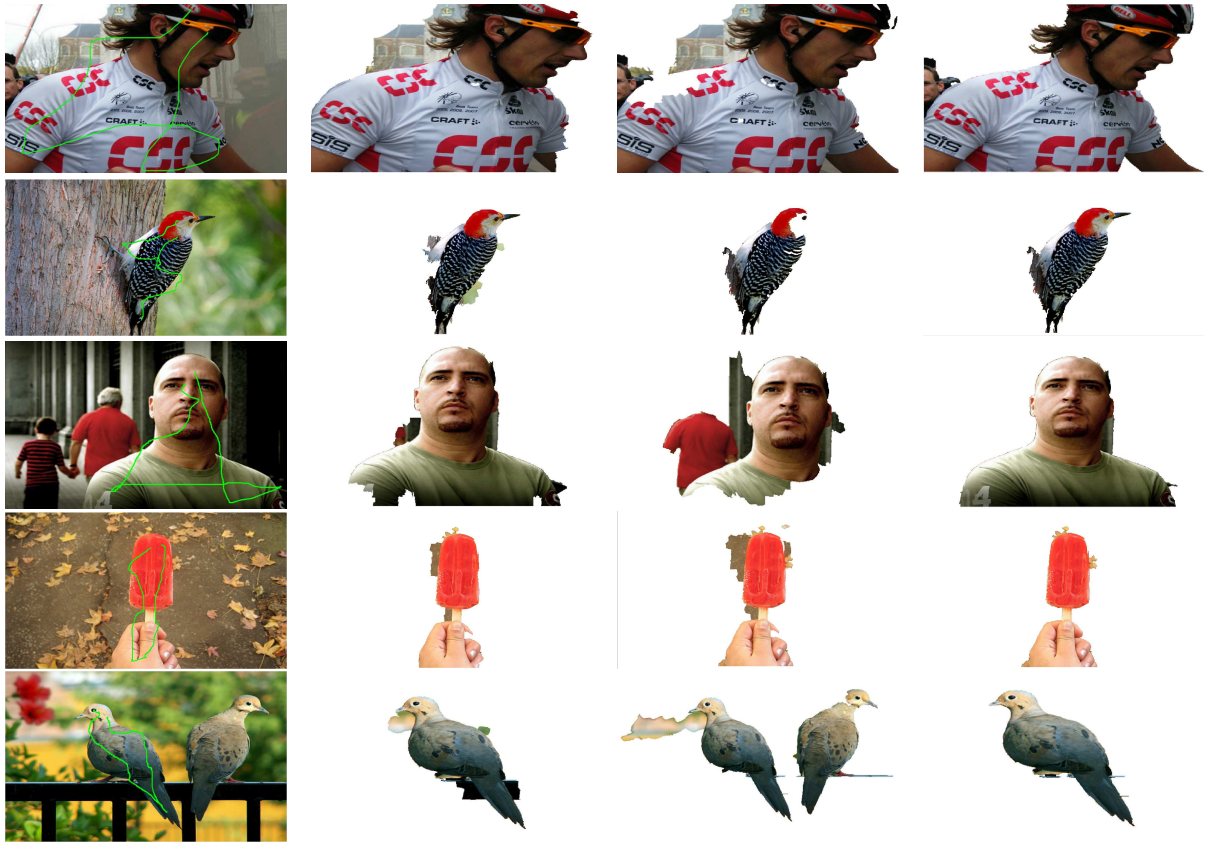


Figure 6: Comparison of the proposed error correction method with hard-label graph-cut and soft-label graph-cut. Columns show interaction and corresponding result for hard-label graph-cut, soft-label graph-cut and proposed method respectively.

Table 1: Average computation times

Boykov&Kolmogrov	Kohli&Torr	Proposed Method
771 msec	278 msec	201 msec

edges with infinite weight) are replaced with soft constraints (terminal edges with weights calculated from GMM), graph-cut framework supposed to handle this interaction errors. However, these hard labels on the foreground are quite meaningful, and the proposed algorithm tries to utilize these interactions. The performance of the error correction algorithm is compared to hard labels and soft labels graph-cut. For the graph-cut with hard labels, interaction is directly fed to algorithm. For the graph-cut with soft labels, likelihood of GMM's are used as terminal weights for all nodes. Non-terminal weights are same in all methods. The results of this experiment are shown in Figure 6.

Figure 6 suggests that proposed algorithm has superior error correction performance. Hard-labeled graph-cut leads to a solution with extra erroneous regions, since no error correction is utilized. In soft-label graph-cut case, discarding hard-labels results in much worse segmentation performance. In the first and third row, input image has highly complex color structure and color profile of foreground and background is also similar to each other. Therefore, using only likelihood terms failed to properly segment the image. In the last row, there are two objects in the image and the user is interested with only one of them. Soft-label graph-cut fails to distinguish these two objects; since both of the objects have almost same color profile. Indeed, it also erroneously classify part of the background as foreground due to the color similarity. In the second row, there is no interaction around the head of the bird. Indeed, color of the head is also available in the background. Moreover, in the forth row, color of the hand and the ground are similar to each other. Therefore, founded foregrounds by soft-label graph-cut are erroneous in both cases. It is surprising that soft-label graph-cut yields worse performance than no error correction(hard-label graph cut). This result actually shows the importance of the hard-labels(infinite terminal weights). In conclusion, not using definite foreground markers (infinite terminal weights) fails to properly segment the image.

3.3 Analysis of Interaction Method

For the subjective evaluation of interactive segmentation algorithms, we have proposed and practiced a subjective evaluation. We have compared 3 different interaction methods, namely intelligent scissors [17], grabcut [19] and proposed method. In order to evaluate the algorithms, 4 different evaluation metrics are used namely *segmentation performance*, *entertainment*, *easiness* and *overall satisfaction*.

First each subject is shown a tutorial about usage of each algorithm. Then, the user is asked to segment 4 images

randomly chosen from the dataset composed of 10 images with various difficulty. For each image, the algorithms are applied in a random order. Then, user is asked to rate each algorithm for each of 4 metrics. Rating is conducted by grading at the level of 1-5. The tests were conducted with a capacitive mobile touch screen. 15 subjects, composed of undergraduate engineering students, have been participated in the tests. Median ratings for each algorithm as well as interquartile ranges (IQR) and standard deviations (STD) is summarized below. Dependent ANOVA test is applied to find p-values and p-values are same for each metric and equal to 0.0005.

Table 2: Interaction Quality Test Results (Median:IQR:STD), P-values are founded via dependent ANOVA test and they all are equal to 0.0005

	Perf.	Easiness	Entertain	Overall
Proposed Met.	5:1:.45	4:0:.86	5:1:.74	4:1:.45
GrabCut[19]	3:2:.92	4:1:.75	2:1:.61	3:1:.77
Int. Scissor[17]	3:1:.51	2:1:.74	3:2:.89	2:1:.76

The proposed method has the best segmentation performance result. This behavior is also visible in Figure 4. Moreover, Grabcut only requires a rectangle around the object so it is expected to be the easiest one subjectively. However, the proposed method is thought to be as easy as grabcut. We believe, this result is due to the intuitive coloring gesture used in the algorithm. Intelligent scissors has the worst easiness result, since the selection of landmarks and movement around the boundary is quite unattractive. Most of the subjects actually commented about this unattractiveness of intelligent scissors. The proposed method has the best entertainment result. It is intuitive since coloring is an entertaining process. On the other hand, Grabcut has worst entertainment result. This result is surprising, since comments of the users suggest that intelligent scissors is an unattractive process. We believe this surprising result is due to the time spent for the interaction. When the user spends more time, he/she likes the process more. In addition to these, the proposed technique has the best overall satisfaction result. With its superior performance and interaction quality this is also expected.

4. SUMMARY & CONCLUSIONS

We addressed the interactive image segmentation problem with particular emphasis on mobile touch-screens. A new interactive image segmentation method that utilizes an intuitive interaction method coloring is proposed with improvements on interaction error correction. Dynamic and iterated graph-cuts method is proposed to increase the speed of the algorithm without compromising on performance. Performance of the proposed method is compared against the available algorithms in the literature. Experiments suggest that proposed method has superior error robustness and computational complexity. In addition to these, the performed user study suggests that algorithms designed for classical mouse based interfaces results in poor interaction quality and overall satisfaction. Therefore, it is necessary to revisit standard problems in the multimedia literature for their mobile extensions.

5. REFERENCES

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC Superpixels. Technical report, EPFL, 2010.

[2] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *ECCV*, pages 428–441, 2004.

[3] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. *ICCV*, 1(July):105–112, 2001.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE PAMI*, Sept. 2004.

[5] F. Calderero and F. Marques. Region merging techniques using information theory statistical measures. *IEEE TIP*, 19(6):1567–86, June 2010.

[6] E. W. Dijkstra. A note on two problems in connexion with graphs. *NUMERISCHE MATHEMATIK*, 1(1):269–271, 1959.

[7] S. Geman and D. Geman. *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, pages 452–472. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[8] L. Grady, M.-P. Jolly, and A. Seitz. Segmentation from a box. In *ICCV*, pages 367–374, 2011.

[9] L. Grady and E. L. Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE PAMI*, 28:469–475, 2006.

[10] L. Grady, Y. Sun, and J. Williams. Interactive graph-based segmentation methods in cardiovascular imaging. In *Handbook of Mathematical Models in Computer Vision*, pages 453–469. Springer, 2006.

[11] P. Kohli and P. H. S. Torr. Dynamic Graph Cuts for Efficient Inference in Markov Random Fields. *IEEE PAMI*, 29(12):2079–2088, 2007.

[12] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE PAMI*, 26(2):147–59, Mar. 2004.

[13] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Transactions on Graphics*, Aug. 2004.

[14] D. Liu, K. Pulli, L. G. Shapiro, and Y. Xiong. Fast interactive image segmentation by discriminative clustering. In *Proceedings of the 2010 ACM multimedia workshop*, MCMC '10, New York, NY, USA

[15] D. Liu, Y. Xiong, L. G. Shapiro, and K. Pulli. Robust interactive image segmentation with automatic boundary refinement. In *ICIP*, pages 225–228, 2010.

[16] J. Liu, J. Sun, and H.-Y. Shum. Paint selection. In *ACM SIGGRAPH 2009*, pages 69:1–69:7, NY, USA, 2009. ACM.

[17] E. N. Mortensen and W. a. Barrett. Intelligent scissors for image composition. *SIGGRAPH '95*, 84602(801):191–198, 1995.

[18] J. Ning, L. Zhang, D. Zhang, and C. Wu. Interactive image segmentation by maximal similarity based region merging. *Pattern Recognition*, 43(2):445–456, Feb. 2010.

[19] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23:309–314, 2004.