AN INTERACTIVE IMAGE SEGMENTATION USING MULTIPLE USER INPUT'S

Swathika P[1], Pradhiba Selvarani M[2], Arulmary S[3], Krishnalakshmi B[4]

[1, 2, 3, 4] *Assistant Professor, CSE, Kamaraj College Of Engineering and Technology College, Tamilnadu, Virudhunagar,*
Swathika.me@gmail.com, mpradhiba@gmail.com, arulmarys@gmail.com, krishna11.kcet@gmail.com

Abstract

In this paper, we consider the Interactive image segmentation with multiple user inputs. The proposed system is the use of multiple intuitive user inputs to better reflect the user's intention. The use of multiple types of intuitive inputs provides the user's intention under different scenario. The proposed method is developed as a combined segmentation and editing tool. It incorporates a simple user interface and a fast and reliable segmentation based on 1D segment matching. The user is required to click just a few "control points" on the desired object border, and let the algorithm complete the rest. The user can then edit the result by adding, removing and moving control points, where each interaction follows by an automatic, real-time segmentation by the algorithm. Interactive image segmentation involves a proposed algorithm, Constrained Random walks algorithm. The Constrained Random Walks algorithm facilitates the use of three types of user inputs. 1. Foreground and Background seed input 2. Soft Constraint input 3. Hard Constraint input. The effectiveness of the proposed method is validated by experimental results. The proposed algorithm is algorithmically simple, efficient and less time consuming.

Keywords: *Interactive image segmentation, Interactive image segmentation, digital image editing, multiple user inputs, random walks algorithm.*

----------***----------

1. INRODUCTION

Interactive image segmentation, an important branch of image segmentation aims to make use of a small amount of user interaction to extract the foreground object out from the background. The main challenge of interactive image segmentation systems is to get as much information as possible from limited user inputs. Interactive image segmentation involves minimal user interaction to incorporate user intention into the segmentation process. The users only need to roughly indicate the location and region of the object and background by using strokes, which are called markers. The user has to initialize the foreground and background seeds to separate the foreground and background objects.

For good interactive image segmentation, there are two basic requirements: First, given a certain user input, the constrained random walks algorithm should produce intuitive segmentation that reflects the user intent. Second, the algorithm must be efficient so that it can produce better segmented results.

Interactive segmentation minimizes user interaction, and also allows users to draw strokes matting on the boundary of the selected object. Completely automatic segmentation never seems to be perfect and so interactive tools are becoming more and more popular. This means that if the results obtained by the first segmentation are not satisfactory, the user can refine the selection. The process of extracting the foreground from the background of an image is known as segmentation. There are many tools that already exist which perform this function. These tools either use texture (color) information or edge (contrast) information to perform the segmentation. Magic wand (color) and Intelligent scissors (contrast) are two good examples. Early interactive image segmentation algorithms utilize either regional properties such as Adobe's magic wand or boundary properties such as active contour [7] and intelligent scissors [10], [11]. The proposed Interactive segmentation method introduces an innovative approach of using both color and contrast information in order to do the segmentation.

The purpose of interactive segmentation is to split an image into two parts – the background and foreground. In order to do this the user has to give 'clues' to the algorithm indicating which regions of the image are part of the foreground and which are part of the background. This can be done by using 'seeds'. A pixel can either be labeled as definitely background or definitely foreground. This is a means of hard segmentation. In other words that pixel must become part of the foreground or background. These pixels then become seeds. Seeds allow the algorithm to automatically calculate the rest of the foreground or background while making sure that the hard constraints are met.

In this paper, the foreground image has to be segmented accurately in still images using constrained random walks algorithm. In Constrained random walks algorithm, three types

of constraints are given as an input to the segmentation process. The user has to draw the strokes manually, in the foreground and in Background region. Three cases arise for providing a input to the images. In the first case, only Foreground and Background seeds are identified from the image. From the seed, the foreground and background strokes are given as input. In the second case, the soft constraint is given as input, which allows a user to draw the strokes to indicate the region that the boundary should pass through. In the third case, the hard constraint is given as input, which allows a user to specify the pixels that the boundary must align with.

The fundamental task of segmenting or partitioning an image into disjoint regions is used for applications ranging from medical image analysis, quality control, or military surveillance and tracking. Although the general segmentation problem involves separating N distinct partitions, a piecewise assumption of two sets is generally made. That is, the image is assumed to be comprised of two homogeneous regions, often referred to as "object and "Background". The goal of segmentation is to accurately capture these regions.

The rest of the paper is organized as follows: We discuss about the related works in Section 2. We define the proposed work in Section 3. We present the constrained random walks algorithm for interactive image segmentation in Section 4. We present the foreground and background seed generation in Section 5. We present the soft and hard constrained inputs in Section 6. In Section 7, we present the results of the proposed framework and in Section 8, we present our conclusion.

2. RELATED WORKS

There have been many published studies of interactive image segmentation. Active Contour Models where first proposed by Kass [7] in 1987 as an Interactive Segmentation method for 2D images. Active Contour Model treats the desired contour as a time evolving curve and the segmentation process as an optimization over time of an adequate energy functional. Live Wire [10] (also known as intelligent scissors) is a Dynamic Programming method for extracting a desired object's contour. The image is treated as a 2D directed weighted graph in which the nodes are image pixels and the arcs are 8-connectivity neighbour's links. Finding the boundary is implemented as finding the shorted path in the graph, where low cost path corresponds to passing near edges and other boundary-related features. For finding optimal path, Dijkstra's algorithms are used, which proves the path globally optimal.

Graph Cut segmentation was originally introduced by Boykov and Jolly [1]. It is shown here as an image segmentation technique but all arguments hold for N-D as well (in fact, it was originally introduced as an N-D method). The number of objects is also arbitrary. The user marks "foreground" and "background" pixels using free-form, wide-brush strokes. The system uses the marked pixels as foreground and background seeds and implies "hard constraints" on them, meaning, the seed pixels must be labels exactly as the user marked, in the final segmentation.

Lazy Snapping [8] enables the user to start by specifying foreground and background marks, and edit the result by border editing. Furthermore, it attacks the performance problem of extensive region-based calculation by working first in higher scale. This is well suited with the dual interaction scheme. GrabCut algorithm by Rother [12] takes the above Graph Cut segmentation even further. It applies the min-cut algorithm iteratively, inferring unknown knowledge from one iteration to another. This allows it to minimize user interaction by just specifying the background (and not the foreground) pixels. User interaction is therefore minimized to placing a rectangle or a lasso around the object (with respectable distance) instead of marking brush-strokes.

Simple Interactive Object Extraction (SIOX) is an algorithm for extracting foreground objects from colour images and videos with very little user interaction. SIOX works well with noise and videos [4]. It has been implemented as "foreground selection" tool in the GIMP. Although the algorithm was originally designed for videos, virtually all implementations use SIOX primarily for still image segmentation. The random walks algorithm [5], [6] model an image as a graph, have been adopted for various image processing tasks. The random walks algorithm requires the input of foreground and background seeds. The random walks algorithm is essentially an approach that minimizes Dirichlet energy with boundary conditions, where different boundary conditions results in different harmonic functions.

The framework proposed in this paper is different from the above in the sense that it considers a foreground extraction with minimal user interaction. Further, the objective of our framework is to extract the foreground object more accurately, even in camouflage (which allows a visible object to remain unseen) when background and foreground colour distributions overlap at least in part.

3. PROPOSED WORK

In the existing system, existing interactive image segmentation algorithms lack more intelligent ways to understand the intention of user inputs. Sometimes the user intends to provide cues for regions while at other times the user intends to focus on the boundaries. When segmenting a difficult image such as a cluttered or camouflaged scene, the user frequently struggles with laborious local editing because the user cannot effectively communicate his intentions with the underlying segmentation engine. Even with more and more scribbles, the segmentation result still may not be what the user wants.

In this paper, we propose a constrained random walks algorithm that facilitates the use of three types of user inputs: 1) foreground and background seed input, 2) soft constraint input, and 3) hard constraint input, as well as their combinations. The

foreground and background seed input allows a user to draw strokes to specify foreground and background seeds. The soft constraint input allows a user to draw strokes to indicate the region that the boundary should pass through. The hard constraint input allows a user to specify the pixels that the boundary must align with. In comparison, the three types of user inputs and any of their combinations can be supported by our coherent computational framework consisting of the constrained random walks algorithm and a local editing algorithm that imposes soft and hard constraints and allows more precise contour refinements.

4. CONSTRAINED RANDOM WALKS

In this section, the extension to the random walks algorithm is proposed. The extension is known as constrained random walks algorithm to facilitate the use of various user inputs in interactive image segmentation. The human input could be valuable in steering the segmentation process in order to obtain more easily and reliably accurate results according to the visual perception. The overall design of the proposed interactive image segmentation method is shown in the Fig. 1.

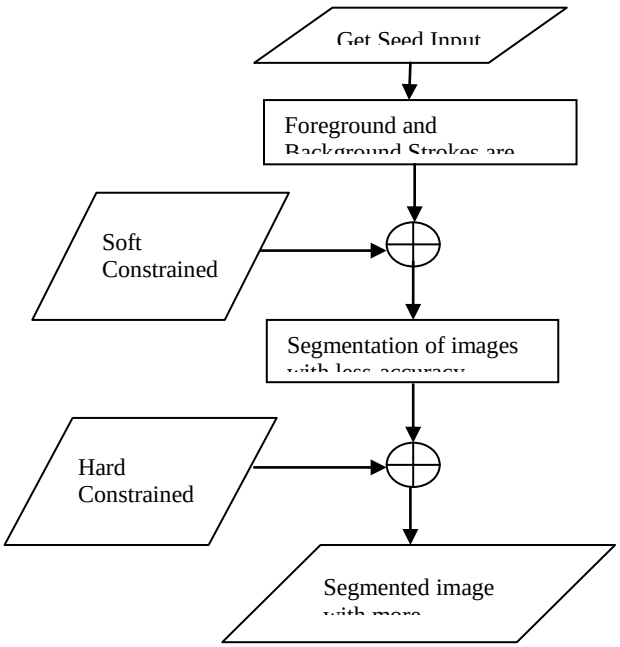


Fig. 1: Overall design of the proposed method

3.1 Graph Construction

A graph is constructed, in which each node represents a pixel and neighbouring nodes are connected with undirected edges. A graph is represented by its vertices and edges as $g = \langle v, \varepsilon \rangle$, where $v = \{v_i\}$ is a set of vertices v_i and v_j and $\varepsilon = \{e_{ij}\}$ is a set of edges e_{ij} bounded by vertices v_i and v_j. The weight for edge e_{ij} is denoted as w_{ij} and the degree of node v_i is defined as $d_i = \sum_j w_{ij}$, that is the sum of the weights of all the edges that incident on v_i.

3.2 Edge weights

The edge weights play a critical role since each edge weight w_{ij} describes the likelihood of a random walker moving to the neighbouring node. Each weight should be defined based upon the distance between two neighbouring pixels/nodes. In the random walks algorithm, the distance between two nodes is defined as $d_{ij} = \|g_i - g_j\|$, which is the Euclidean distance in colour. The prior models have also been incorporated into the random walks algorithm, where two virtual nodes are added in the graph to represent the foreground and the background. The foreground and background node is connected to every pixel node and the edge weight is assigned to the probability that the pixel fits the foreground/background prior model. In the proposed method, the prior models are incorporated directly into the distance function to avoid the distance connection problem. The edge weight w_{ij} and the corresponding distance are defined as

$$w_{ij} = \exp\left(-\beta \cdot d_{ij}^2\right)$$

$$d_{ij}^2 = (1-\alpha)\|g_i - g_j\|^2 + \alpha\left(\Pr(v_i) - \Pr(v_j)\right)^2 \quad (1)$$

The second term $\left(\Pr(v_i) - \Pr(v_j)\right)^2$ is the prior term, $\Pr(v_i)$ denotes the normalized probability that node v_i fits the foreground prior model, $\alpha \in [0,1]$ is a trade off factor, and β is a scaling factor. The values of $\|g_i - g_j\|^2$ and $\left(\Pr(v_i) - \Pr(v_j)\right)^2$ are normalized to [0, 1] individually.

The foreground and the background are modelled by the Gaussian mixture model (GMM) and their seeds S_F and S_B are used to estimate the model parameters. Let $\Pr\langle v_i | F\rangle$ $\left(\Pr\langle v_i | B\rangle\right)$ denote the probability that node v_i fits the foreground and the background GMM. The normalized probability $\Pr(v_i)$ is defined as

$$\Pr(v_i) = \frac{-\log \Pr\langle v_i | F\rangle}{-\log \Pr\langle v_i | F\rangle - \log \Pr\langle v_i | B\rangle} \quad (2)$$

According to equation (1), it is clear that the second term $(\Pr(v_i) - \Pr(v_j))^2$ should dominate when the foreground and background colours are well separable. Otherwise, the first term $\|g_i - g_j\|^2$ should dominate. Let α be the distance between the foreground and background GMMs. The Monte-Carlo simulation is used to approximate the KL-divergence between F and B. α can be defined as,

$$\alpha = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\log \Pr\langle v_i | F\rangle - \log \Pr\langle v_i | B\rangle}{\log \Pr\langle v_i | F\rangle + \log \Pr\langle v_i | B\rangle}\right| \quad (3)$$

Where n is the total number of pixels. After determining the parameter α, the distance measure is calculated. The edge weight can be applied to graph cuts, random walks, normalized cuts.

5. FOREGROUND AND BACKGROUND SEED GENERATION

The foreground and background seed input allows a user to draw strokes to specify foreground and background seeds. It identifies the foreground and background object clearly. After foreground and background seeds are identified and located, the strokes are drawn. The use of only foreground and background strokes as input allows the user to segment foreground object accurately. The design of the foreground and background seed generation is shown in Fig. 2.

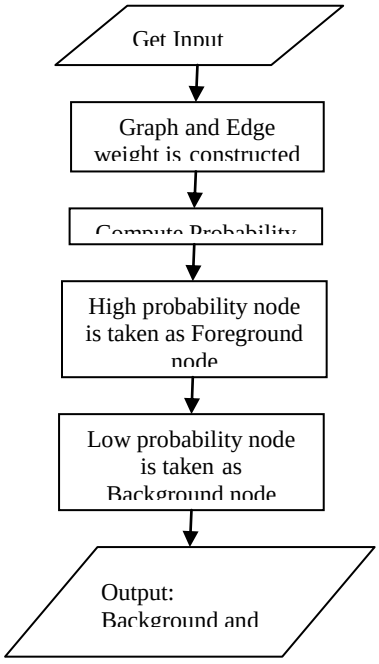


Fig. 2: Design of the Foreground and Background seed generation

First, the image is given as input. Graph and edge weight is constructed. The user input includes foreground and background seeds. Let p_i be the probability that an edge weight will arrive at one of the foreground seeds. Clearly, for the probability p_i, if $p_i = 1$, the high probability node is considered as a Foreground seed. If the probability is $p_i = 0$, the low probability node is considered as a Background. The user input includes foreground seeds S_F and background seeds S_B, where $S_F \subset \upsilon$, $S_B \subset \upsilon$, and $S_F \cap S_B = \phi$. Let p_i be the probability, $p_i = 1$ for any $\upsilon_i \in S_F$ and $p_i = 0$ for any $\upsilon_i \in S_B$. A vertex on which the soft constraint is imposed has the property that the difference its probability is ½, lies within a small prescribed range [$-\in, \in$]. A vertex on which the hard constraint is imposed has a probability of ½.

Let S_S and S_H denote soft boundary seeds and hard boundary seeds, respectively. The Constrained random walks are given by the following equations:

$$p_i = \frac{1}{d_i}\sum_{e_{ij}\in\varepsilon} w_{ij}.p_j, \qquad \upsilon_i \in \upsilon \setminus (S_F \cup S_B \cup S_H)$$

$$s.t., \begin{cases} p_i = 1, & v_i \in S_F \\ p_i = 0, & v_i \in S_B \\ p_i = 0.5, & v_i \in S_H \\ |p_i - 0.5| \le \varepsilon, & v_i \in S_S, \varepsilon \approx 0^+ \end{cases} \quad (4)$$

The equation (4) can be reformulated into

$$\min \sum_{e_{ij}\in\varepsilon} w_{ij}(p_i - p_j)^2 + \lambda \sum_{v_i \in S_S}(p_i - 0.5)^2$$

$$s.t., \begin{cases} p_i = 1, & v_i \in S_F \\ p_i = 0, & v_i \in S_B \\ p_i = 0.5, & v_i \in S_H \end{cases} \quad (5)$$

Where λ is a trade off factor controlling the importance of the difference between the probability of each soft constraint vertex and ½. Differentiating the objective function of equation (5) with respect to each p_i for $v_i \in \upsilon \setminus (S_F \cup S_B \cup S_H)$ and setting it equal to zero. The constrained random walks solve the linear system of equation given in equation (6)

$$p_i = \begin{cases} \frac{1}{d_i + \lambda}\left(\sum_j w_{ij} \cdot p_j + \lambda \cdot 0.5\right), & v_i \in S_S \\ \frac{1}{d_i}\left(\sum_j w_{ij} \cdot p_j\right), & v_i \in \upsilon \setminus (S_F \cup S_B \cup S_H \cup S_S) \end{cases}$$

(6)

The above equation (6) can be considered as adding a virtual neighbour vertex with probability of ½ to each soft constraint vertex through a virtual edge with weight λ .

The logical step in using Random walks algorithm is to generate a suitable initialization (seed points). In our approach, a list of potential seed points are generated, that are used to locate the foreground and background regions. The seed points are chosen as the maxima points in the distance map that are located in the foreground and background. Maxima points are located by dilating the distance map and finding those points that do not change in value. It is easy to see that dilation by a mask does not change the value at only the maxima points. The size of the mask determines the minimum separation distance between any two seed-points.

6. CONSTRAINED INPUTS

Two other types of user inputs as constraints into the random walks algorithm, the extension is known as constrained random walks. The two constrained inputs are soft constrained input and hard constrained input.

6.1 Soft Constrained Input

The soft constraint input allows a user to draw strokes to indicate the region that the boundary should pass through. After the foreground and background seeds are located, the foreground and background strokes are drawn. By only inputting the foreground and background strokes, the weak boundary between the object and the background cannot be detected precisely. To overcome this problem, a soft stroke is drawn. With only one additional soft boundary stroke, which is partially drawn on the boundary provides better results.

6.1.1 Local Contour Deformation for Soft Constraint Input

A method of soft constraint input is the local refinement step, where only additional hard and soft boundary constraints are allowed to be used to indicate the boundaries. This local refinement still needs to be performed if there is any soft or hard boundary stroke being input in the global stage. This is because the constrained random walks algorithm itself cannot achieve the purpose of both hard and soft boundary constraints. For example, the neighbouring nodes of soft constraint have probability values. Let λ be the weighting parameter, if the λ value is too large, the pixels masked by the soft boundary strokes will have probability values very close to ½, for which the precise boundary is difficult to locate by thresholding.
For the local refinement, first determine the pixels/positions that the contour must pass and then build the correspondences between these pixels and the pixels on the initial contour. After that, the initial contour is deformed with the correspondences used as positional constraints and the rest of the pixels on the initial contour as stay-put constraints.

6.1.2 Optimal Path in Soft Boundary Stroke

Within each soft boundary stroke, an optimal path called soft boundary path needs to be searched first. The Dijkstra's shortest path algorithm is used to find the shortest path. Let v_p and v_q denote two neighbouring pixels. The local cost e_{pq} of the directed edge from v_p to v_q is defined as a weighted sum of three components: Laplacian zero crossings f_z, gradient magnitude f_g, and the gradient directional cost $f_d(p,q)$, i.e,

$$e_{pq} = \frac{\{0.1 \cdot f_z(q) + 0.6 \cdot f_g(q) + 0.3 \cdot f_d(p,q)\}}{len} \quad (7)$$

Where the division by len, denotes the length of the path from v_p to v_q is to avoid the "shortcut" problem. The weights specified in equation (7) are empirical values. The purpose of having the Laplacian zero-crossing term in equation (7) is for edge localization. The gradient magnitude is to distinguish between a strong edge and a weak edge. The gradient direction or orientation term adds a smoothness constraint to the boundary by associating a relatively high cost for sharp changes.

6.1.3 Finding Correspondences in Soft Constraint Input

Correspondences between the soft user input and the initial contour C obtained in the global stage are established before contour deformation. The correspondence for a soft boundary constraint matches the points from the initial contour to the soft boundary path inside the stroke. First, the two end points of the soft boundary path are matched to the points on the contour with minimum geometric distance, and then the in-between points of the soft boundary path and the contour are matched by bilinear interpolation. The correspondence set is given as:

$$\{v_1^s, v_2^s \ldots\ldots\ldots v_{k2}^s\} \rightarrow \{s_1, s_2 \ldots\ldots\ldots, s_{k2}\} \quad (8)$$

Where s_i is the soft boundary constraint point, and v_i^s is the corresponding point on the initial contour.

6.1.4 Refinement by Contour Deformation

The contour deformation can be summarized as: for the initial contour C, the positional constraints resulted from the correspondences for the hard and soft boundary points. The contour deformation is given as:

$$\arg\min_{v'}\left(\left\|Lv'-Lv\right\|^2+\omega\sum_j\left\|v_j^{s'}-s_j\right\|^2\right)+\sum_{j\in C_{stay}}f\left(dist(j)\right)\left\|v_j'-v_j\right\|^2,\ s.t.,v_j^{h'}=h_j \quad (9)$$

Where ω is a trade off parameter for soft boundary constraint points, and v and v' denote two column vectors whose elements v_j and v_j' are the vertices on the initial contour C and the deformed contour C' respectively. The first term $\left\|Lv'-Lv\right\|^2$ is adapted from the Laplacian mesh processing to preserve the global shape with the transform matrix L defined as

$$L=I-D^{-1}A \quad (10)$$

Where D is a matrix with D_{ii} =2 and A is the adjacent matrix defined in equation (11)

$$A_{ij}=\begin{cases}1, & |i-j|=1\\ 0, & otherwise\end{cases} \quad (11)$$

The distance function $f\left(dist(i)\right)$ is considered as the weight. $dist(i)$ is the normalized geometric distance from pixel $v(i)$ to the input strokes and the function f can be any monotonic increasing function such that pixels near the input strokes have small weights and those far away from the input strokes have larger weights.

6.2 Hard Constrained Input

The hard constraint input allows a user to specify the pixels that the boundary must align with. In the soft constraint input, the user draws the strokes along the boundary and the object is fairly well segmented except some small inaccuracies i.e., the boundary is not aligned with in the region. To overcome this problem, the hard boundary constraint input method is proposed. Boundary pixel selector, which selects pixels on the desired contour, is introduced as the hard constraint. A vertex on which the hard constraint is imposed has a probability of ½. Specifically, by marking five hard boundary pixels, the local editing algorithm snaps the initial contour to the specified boundary pixels through local contour deformation, which results in a smoother and more accurate object contour.

6.2.1 Local Contour Deformation for Hard Constraint Input

Local Contour Deformation is also performed in hard constraint input. Deforming contours occur either due to changing region of partial occlusions or when the object of interest is actually deforming its shape over a time or space sequence of images. Higher speed is achieved by the local segmentation determined only on the vicinity of the route initialization contour and the segmentation is not performed on the entire input image. The size of the vicinity of the initialization contour which is taken into account by the processing of the image is determined interactively by the user. Terminals of the object and background in this procedure are determined automatically from initialization contours specified by the user. The neighbouring nodes of a hard constraint node have probabilities larger than ½ or less than ½, for which the resulting contour will not pass through the hard constraint node.

6.2.2 Finding Correspondences in Hard Constraint Input

Correspondences between the soft user input and the initial contour C is obtained in the global stage are established before contour deformation. The correspondence for a hard constraint point is defined as the point on the contour with minimum geometric distance. The point from the initial contour to the hard boundary path inside the stroke is matched. First, the two end points of the hard boundary path are matched to the points on the contour with minimum geometric distance. The correspondence set is given as:

$$\left\{v_1^h, v_2^h .....v_{k1}^h\right\} \rightarrow \left\{h_1, h_2, ...h_{k1}\right\} \quad (10)$$

Where h_i, is the hard boundary constraint point, and v_i^h is the corresponding point on the initial contour.

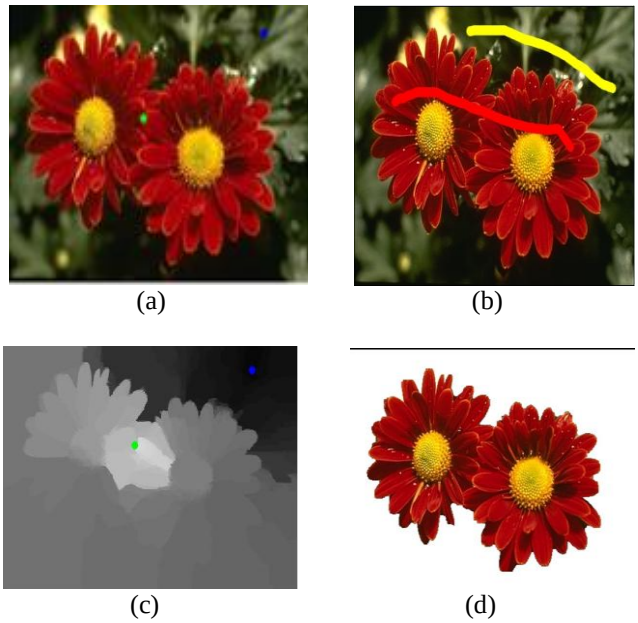
7. RESULTS AND DISCUSSIONS

The proposed method has been implemented and tested on various still images. The proposed system is to accurately segment the foreground image from the background with the minimal user interaction using interactive image segmentation. Figure 3 (a) shows the input image. This image is given as input for the foreground and background seed generation. The object with high intensity is considered as the foreground object. The green seed indicates the foreground object. The blue seed indicates the background object. Figure 3 (b) shows the foreground and background seed input in the image.

(a) (b)

Fig .3(a) Original input image (b) Image with Foreground and Background Seeds. Green seed indicates the foreground object. Blue seed indicates the background object.

The foreground and background seeds are generated based on graph and edge weight calculation. Figure 4 (a) shows the image with seed input. Figure 4(b) shows the image with foreground and background strokes. Figure 4 (c) shows the probability map for the given image. Figure 4(d) shows the segmented output for the given image.

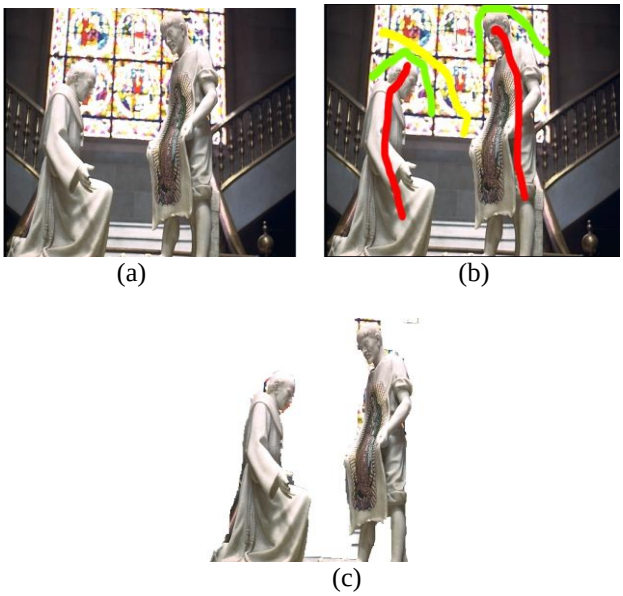


(a) (b)

(c) (d)

Fig. 4 (a) Image with Foreground and Background Seeds. Green seed indicates the foreground object. Blue seed indicates the background object (b) Foreground and Background seeds (c) Probability map for the Constrained Random walks Algorithm (d) segmented result for the Foreground and background strokes.

If the object is in complex background, foreground and background strokes are drawn on the image. The probability map provides the good separation between the foreground and background object. Figure 5 (a) shows the input image. Figure 5 (b) shows the image with soft constraint input. Figure 5 (c) shows the segmented output for the soft constraint input.



(a) (b)

(c)

Fig. 5 (a) Original input image (b) Soft Constraint input. Red stroke indicates the foreground object. Yellow stroke indicates the background object. Green stroke indicates the Soft constraint input. (c) Segmented result for the soft constraint input.

The soft constraint input allows the user to draw the strokes to indicate the region that the boundary must align with. The Soft Constraint input is given with the boundary brush that roughly mark parts of the boundary.

Figure 6 (a) shows the hard constraint input. Figure 6 (b) shows the segmented output.

(a) (b)

Fig. 6(a) hard constraint input, Black pixels are the hard boundary pixel selector. (b) Segmented Output.

The hard constraint input allows a user to specify the pixels that the boundary must align with. In the soft constraint input, the user draws the strokes along the boundary and the object is fairly well segmented except some small inaccuracies i.e., the boundary is not aligned with in the region. To overcome this problem, the hard boundary constraint input is given.

7.2 *Performance Measure*

The error rate is defined as the ratio of number of misclassified pixels to the number of pixels in unclassified region. The error rate is calculated using the formula:

$$\varepsilon = \frac{no.\ misclassified\ \ pixels}{no.\ pixels\ \ in\ \ unclassified\ \ region}$$

Where misclassified pixels are the incorrect pixels within the contour region. The unclassified region specifies the pixels that are present in the contour region. For most of the images in the MSRC data set, the proposed method achieves very low error rates. High error rates occur in images where the input foreground seeds only cover a small portion of the foreground and, thus, do not cover all the distinct colors of the foreground, while pixels with similar colors are masked by the background seeds. For such cases, the GrabCut algorithm and the random walks algorithm also perform poorly.

Table 1: Performance comparison with different images

Images	Error rate for Random Walks in (%)	Error rate for Contour Deformation in (%)	Error rate for Constrained Random Walks in (%)
Statues.jpg	5.85	4.92	4.82
Building.jpg	5.66	4.84	4.75
Penguin.jpg	5.42	4.8	4.63
Tiger.jpg	5.43	4.63	4.45
Boy.jpg	5.38	4.23	4.36
Crow.jpg	5.08	4.16	4.14
Bear.jpg	4.9	4.02	4.08
Flower.jpg	4.5	3.98	3.8

In the existing random walks algorithm, the error rate is high, while for contour deformation the error rate gets decreased. In the proposed constrained random walk algorithm, the error rate gets decreased compared to existing algorithm. Therefore the proposed algorithm is much better than the existing algorithm which is shown in the Table 1. It is also inferred from the table that for complex images the error rate gets decreased compared to the existing method.

CONCLUSIONS AND FUTURE WORK

Interactive image segmentation consists of two components: Constrained random walks algorithm and local contour deformation. The proposed method supports multiple intuitive types of user inputs and also combines the advantages of different user interactions. The proposed method can be applicable for camouflaged images and images with thin structures, if the user inputs are carefully placed.

The foreground and background strokes are carefully drawn at each side of the object boundary in order to well segment the object. With a certain amount of user interaction, some inaccuracies can occur for thin structures whose colour overlapped with the background colour. In this case, the soft constraint and hard constraint input is given by the user. The soft constraint input allows the user to draw soft strokes on the boundary with the boundary brush. With the soft constraint input, small inaccuracies occur in the boundary. To overcome this problem, hard constraint input is provided. The hard constraint input allows the user to mark the pixels on the boundary. Specifically, by marking hard boundary pixels on the inaccurate regions are marked with the boundary pixel selector. This results in smoother and more accurate segmented output.

The proposed method uses three types of user inputs, and able to provide more efficient and accurate results. The proposed method is faster compared to other techniques. Constrained random walks algorithm requires minimal user interaction to extract the region. The error rate gets reduced compared to existing methods.

The proposed method can be extended in two ways. First, its runtime can be further improved. The response can be improved through the graphics processing unit. Second, the proposed method is extended to video segmentation. Extending the method to video segmentation is not straight forward. This is because video data consists of thousands of video frames. So the user has to provide inputs for each video frame. It is also possible to create a metric to measure the amount and the complexity of interaction that is required for an interactive image segmentation to achieve better results.

REFERENCES

[1] Boykov Y. and Jolly M.P., "Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images," in Proc. IEEE Int. Conf. Computer Vision, 2001, pp. 105–112.

[2] Bai X. and Sapiro G., "A geodesic framework for fast interactive image and video segmentation and matting," in Proc. IEEE Int. Conf. Computer Vision, 2007, pp. 1–8.

[3] Falcao A., Stolfi J., and Lotufo R., "The image foresting transform: Theory, algorithms, and applications," IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 1, pp. 19–29, Jan. 2004.

[4] Friedland G., Jantz K., and Rojas R., "SIOX: Simple interactive object extraction in still images," in Proc. IEEE Int. Symp. Multimedia, Dec. 2005, pp. 253–259.

[5] Grady L., "Multilabel random walker image segmentation using prior models," in Proc. IEEE Int. Conf. Computer Vision Pattern Recognition, 2005, pp. 763–770.

[6] Grady L., "Random walks for image segmentation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 11, pp. 1768–1783, Nov. 2006.

[7] Kass M., Witkin .A, and Terzopoulos .D, "Snakes: Active contour models," Int. J. Comput. Vis., pp. 321–331, 1988.

[8] Li Y., Sun J., Tang C.K. and Shum H.Y., "Lazy snapping," ACM Siggraph, pp. 303–308, 2004.
[9] Miranda P., Falcao A., and Udupa J., "Synergistic arc-weight estimation for interactive image segmentation using graphs," Comput. Vis. Image Understand., vol. 114, no. 1, pp. 85–99, Jan. 2010.
[10] Mortensen E. N. and Barrett W. A., "Intelligent scissors for image composition," Comput. Graph. Int. Techniques, pp. 191–198, 1995.
[11] Mortensen E. N. and Barrett W. A., "Interactive segmentation with intelligent scissors," Graph. Models Image Process., vol. 60, no. 5, pp. 349–384, 1998.
[12] Rother C., Kolmogorov V., and Blake A., "Grabcut: Interactive foreground extraction using iterated graph cuts," ACM Siggraph, pp. 309–314, 2004.
[13] Sinop A. K. and Grady L., "A seeded image segmentation framework unifying graph cuts and random walks which yields a new algorithm," in Proc. IEEE Int. Conf. Computer Vision, 2007, pp. 1–8.

BIOGRAPHIES:

Swathika. P received her UG - B.Tech (Information Technology) degree in 2009 from Kamaraj College of Engineering and Technology. She then completed her P.G - M.E degree in Computer Science and Engineering at Mepco Schlenk Engineering College, Sivakasi in 2011. She is currently working as an Assistant Professor in Kamaraj College of Engineering and Technology, Virudhunagar, India. She is a Life Member of Indian Society of Technical Education (ISTE). Her area of interest encompasses Digital Image Processing, image segmentation and Visual cryptography.

Pradhiba Selvarani. M received her UG - B.Tech (Information Technology) degree in 2009 from Idhaya Engineering College for Women, chinnasalem. She then completed her P.G - M.E degree in Computer and Communication Engineering at National Engineering College, Kovilpatti in 2011. She is currently working as an Assistant Professor in Kamaraj College of Engineering and Technology, Virudhunagar, India. She is a Life Member of Indian Society of Technical Education (ISTE). Her area of interest encompasses Digital Image Processing and Visual Cryptography.

Arul Mary.S received her UG - B.E (Computer Science) degree in 2009 from Sri Ramakrishna Engineering College, Coimbatore . She then completed her P.G - M.E degree in Computer Science and Engineering at Mepco Schlenk Engineering College, Sivakasi in 2011. She is currently working as an Assistant Professor in Kamaraj College of Engineering and Technology, Virudhunagar, India. She is a Life Member of Indian Society of Technical Education (ISTE). Her area of interest encompasses Digital image processing, Biometric authentication and Visual cryptography.

Krishnalakshmi. B received her UG - B.E (Computer Science) degree in 2009 from P.S.R.Engineering College, Sivakasi. She then completed her P.G - M.E degree in Computer Science and Engineering at Anna University of Technology, Coimbatore in 2011. She had worked as an Assistant Professor in RVS educational group of institutions, Coimbatore. She is currently working as an Assistant Professor in Kamaraj College of Engineering and Technology, Virudhunagar, India. She is a Life Member of Indian Society of Technical Education (ISTE). Her area of interest encompasses Network Security, Cryptography.