INTERACTIVE IMAGE SEGMENTATION THROUGH LAGRANGE SALIENCY

P. Akshay Kumar[1] *Koteswar Rao Jerripothula*[2]

[1] CSE Department, IIIT-Delhi
[2]Faculty, CSE Department, IIIT-Delhi
akshay19094@iiitd.ac.in, koteswar@iiitd.ac.in

ABSTRACT

Interactive image segmentation is needed to counteract the weakness in existing segmentation algorithms to map the user's imagination to the segmented output that is being generated. We propose a novel technique that captures the user's imagination through multiple foreground and background scribbles in GUI. We use the spatial and intensity values of these pixels to generate a Lagrange polynomial curve which is used to interpolate rest of the pixels. Since, the essence of location and intensity is being captured, precise segmentation was possible producing good results on benchmark data sets.

***Index Terms*—** Lagrange interpolation, Saliency, Interactive Image Segmentation, Multiple user inputs, Otsu, KMeans

1. INTRODUCTION

Interactive image segmentation is an active area of research which involves user interaction. The selected points through scribbles in user interaction help attain results that are better than most of the state-of-the-art segmentation algorithms. Our goal is to develop a tool which allows user to choose foreground and background points through scribbles and use the selected points in the underlying algorithm to achieve satisfactory segmentation that matches the user imagination.

A good user segmentation algorithm must cater to two basic requirements:
1. Generate a visually satisfying segmentation that augurs well with the given user input.
2. The algorithm must be robust and efficient to generate instant results.

In our research work, we have focused on enhancing the user provided seeds in order to maximize the information passed onto the Lagrange interpolation equation in order to get better saliency maps. We have used color averaging, PCA, KMeans[1] and SLIC superpixels[2] for the purpose of enhancing the input seeds. We have further used thresholding and morphological techniques to fine tune the output so that it matches with user's imagination of the segmented foreground. Some of the techniques involve Otsu's thresholding[3], flood fill algorithm, largest connected component extraction to filter out the noise elements from the resultant saliency map. We have segregated the images into different categories to enable targeted seed selection and thresholding in order to obtain higher quality saliency map.

2. RELATED WORK

Some of the early interactive segmentation techniques were active contour method[4] and intelligent scissors algorithm[5]. Active contour method involves user placing the contour near the desired boundary the algorithm evolves the boundary. The limitation in this method is that the contour is likely to be trapped in a local minima. The intelligent scissors algorithm allows the user to mark the contour of the foreground object. It uses Dijkstra's shortest path algorithm to compute the paths connecting the neighbouring regions. The limitation is that there exists multiple paths that can be taken and mostly requires large number of user interactions to obtain a satisfactory result.

The advanced algorithms include graph cut based techniques of which GrabCut method[6] is very popular. GrabCut framework segments color image where the segmented foreground and background colours are modelled by Gaussian mixture models. It works for various input types as well achieves good performance in segmenting images whose foreground and background images are well separable. It does not work well when the color distribution of foreground and background are similar or if the image is cluttered or camouflaged. It also does not work well with noisy images.

SIOX algorithm[7] is a popular technique implemented in tools such as GIMP, Inkscape, Blender and Krita. It is heavily reliant on the foreground and background seeds chosen in the region of interest. This technique is robust to noise but suffers the same drawback as Grabcut where it misclassifies the foreground when the color distribution of pixels is similar.

Random walks algorithm[8] has been widely used in various image segmentation implementations. It is more intelligent and performs better than other graph cut algorithms. However, due to the lack of a proper global distribution model, it is highly sensitive the seeds provided as input by the user.

3. PROPOSED METHOD

The main drawback of most interactive segmentation algorithms is that the lack of diligence to understand the intentions of user inputs. User usually picks the visually striking feature of the image as foreground and the pixels around the boundaries as the background. When the image is camouflaged with the background or is blurred, then user finds it difficult to effectively communicate the intentions while picking the points or making scribbles. In such cases, even with more and more user scribbles, the segmentation result may or may not match user's imagination.

The objective of our work is to generate a method which makes use of the pixels chosen by the user as well as its locality to make intelligent decisions while generating the saliency map for the original image. In this paper, the proposed algorithm takes the input image and two types of points as input.

1) Foreground points - These are usually the main object of focus in the image chosen by the user and is denoted by red scribbles drawn by user using the tool. The intensity value for the foreground points are taken as 1.

2) Background points - These are usually subsidiary objects or the points near the boundaries and are denoted by blue scribbles drawn by user using the tool. The intensity value of background pixels are taken to be 0.

The intensity values as discussed in the above section is represented as follows

$$Intensity = \begin{cases} 1, & \text{if } foreground\ pixel \\ 0, & \text{if } background\ pixel \end{cases} \tag{1}$$

3.1. Lagrange interpolation

Lagrange interpolation[9] is a popular technique used to interpolate missing values in an image. The intensity, saliency values of the user chosen input pixels obtained from (1) are passed as parameters to Lagrange's polynomial equation to interpolate the intensity values of all the remaining pixels.

Lagrange's interpolation polynomial is generated using the following equation:

$$L(x) = \sum_{j=0}^{k} y_j l_j(x) \tag{2}$$

where L is Lagrange interpolation function, y is the j^{th} saliency value in the saliency vector $l_j(x)$ is computed as follows:

$$l_j(x) = \prod_{\substack{0 \le m \le k \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \frac{x - x_0}{x_j - x_0} \cdots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{x - x_k}{x_j - x_k} \tag{3}$$

The above equation (2) generates a polynomial curve to which the remaining pixel values are passed to get their corresponding intensity values.

3.2. Choosing the right seeds

The following subsections involves the various techniques involved in filtering the good seeds to be passed to Lagrange Interpolation equation represented by (3) to generate the saliency map.

3.2.1. Color channel averaging

In this technique, we separated the color channels (R,G,B) and then passed intensity value of a pixel corresponding to each channel individually to the generated Lagrange interpolation polynomial represented by (2). The average of saliency value for each channel (output of ((2))) is considered as the saliency value for a specific pixel of an image. We repeat this process for all the pixel values of the image which generates the saliency map.

3.2.2. Principal component analysis

In principal component analysis, we first computed the PCA coefficient matrix of the image. This was followed by a matrix multiplication of PCA coefficient matrix and the input image. The generated image was then segregated into three channels (R,G,B). Then, we used the channel averaging similar to the technique mentioned in sub-section 3.2.1 to generate the saliency map for each individual pixel value of the image.

3.2.3. Channel wise polynomial generation

This technique involves generating a Lagrange polynomial equation for each channel - R,G,B. The three curves were used in colour averaging (section 3.2.1) and principal component analysis (section 3.2.2) to get the intensity value corresponding to each channel. The obtained values corresponding to each pixel were then averaged to get the saliency map.

3.2.4. Lagrange interpolation using spatial location and intensity values of pixels

This technique involves combining the effect of the spatial location along with the intensity values of pixels for interpolation. The R,G,B values are averaged into a single intensity averaging component whereas the x,y spatial coordinates are normalized by dividing with the length of diagonal of the image and then multiplying it by 255 to ensure that it remains in the intensity values range. The normalization equations is represented as follows:

$$x = (x/diagonal(image)) * 255 \tag{4}$$

$$y = (y/diagonal(image)) * 255 \tag{5}$$

where x,y are spatial coordinates and diagonal(image) represents the length of diagonal of the image.

$$z = (r + g + b)/3 \tag{6}$$

where r,g,b are red, green and blue intensity values respectively of pixel x,y and z represents the average intensity value.

Here, we are dealing with three dimensions - x coordinate (4), y coordinate (5) and average intensity value (6) which is passed to Lagrange polynomial equation. The Lagrange equation represented by (2) is tweaked to accommodate multidimensional attributes as per following equation.

$$\prod_{k=1,k\neq i}^{n} \frac{(x-x_k)*(y-y_k)*(z-z_k)}{(x_i-x_k)*(y_i-y_k)*(z_i-z_k)} \tag{7}$$

where x,y,z represent the x coordinate (4), y coordinate (5) and average intensity value (6) respectively and n represents the number of pixels.

Once the Lagrange polynomial is generated, we interpolate the remaining pixels using the colour averaging filter represented in section 3.2.1 and PCA represented in section 3.2.2.

3.2.5. Using KMeans to filter the seeds

This technique involves passing the user scribbles to KMeans clustering[1] algorithms to filter out the good seeds. Foreground and background seeds are passed separately to the KMeans algorithms and K clustering centroids are chosen as representative points for user given input scribble. These seeds are then passed onto Lagrange Interpolation equation to generate the curve for saliency map.

3.2.6. Using SLIC superpixel algorithm to filter the seeds

Superpixels are the set of pixels which share common characteristics and carry more information than others. We used SLIC superpixels algorithm[2] to generate a label matrix for the original image. The pixels from user input scribbles are mapped with superpixel label regions and all pixels corresponding to a mapped region are extracted. We used KMeans specified in sub-section 3.2.5 to pick 1 pixel from each superpixel region as a representative point for the images which have minimum distance between foreground and background. For the images with reasonable distance, we sampled K cluster centroid from all the samples pixels from mapped superpixel regions.

3.2.7. Using normalized spatial coordinates

Spatial coordinates are important to distinguish foreground and background regions with similar intensity patterns. In addition to seeds obtained through sub-section 3.2.5 and 3.2.6, we used the normalized spatial coordinates to improve the saliency map. The normalization equation is similar to that is specified by (4) and (5). We added an additional location tuning factor to control the impact of spatial coordinates in the resultant saliency obtained. The final normalized equations are represented below in (8) and (9).

$$x = ((x*255*\alpha)/diagonal(image)) \tag{8}$$

$$y = ((y*255*\alpha)/diagonal(image)) \tag{9}$$

where x,y are spatial coordinates, diagonal(image) represents the length of diagonal of the image alpha is a tune-able location tuning factor and its value ranges from 0 to 1.

3.3. Thresholding the obtained saliency map

Thresholding is one of the most important operation in any segmentation and it determines the quality of the resultant binary saliency map. We used the following techniques to generate the binary map.

3.3.1. Otsu Thresholding

Otsu thresholding[3] is one of the popular techniques used for segmenting foreground and background. The generated saliency map through Lagrange interpolation is passed to Otsu's algorithm with number of bins obtained from histogram of the obtained saliency map. The Matlab code snippet is as follows

```
[counts,~] = imhist(fxi);
T = otsuthresh(counts);
salient_img_otsu = imbinarize(fxi,T);
```

3.3.2. Manual Thresholding

We used manual thresholding to hard code the threshold value ranging from 0.6 to 0.8. For certain category of images, manual thresholding with 0.8 threshold seems to be producing better results. The Matlab code snippet is as follows

```
T=0.8
salient_img_final = imbinarize(fxi,T);
```

Matlab function	Functionality
imhist	calculates the histogram of a grayscale image
otsuthresh	returns the thresholding value of a grayscale image
imbinarize	thresholds the image based on the thresholding value provided

Table 1: Matlab functions and their functionality

3.4. Segregation of images in dataset

It is not practically possible to get the best results for different types of images with the same set of thresholding parameters and number of seeds. We need to augment and fine tune images by segregating them based on some valid parameter. We used color distance parameter as well as spatial and color

distance between foreground and background seeds as a parameter to segregate. In the following subsection each of the parameter is discussed.

3.4.1. Color distance parameter

Color distance parameter is the distance between R,G and B intensity values of the user provided foreground and background seeds of the input image. It is computed as shown in (10)

$$d = \sum_{i=1}^{m}\sum_{j=1}^{n}\sqrt{\frac{(rf_i - rb_j)^2 + (gf_i - gb_j)^2 + (bf_i - bb_j)^2}{m * n * rmax * gmax * bmax}} \tag{10}$$

where rf,gf,bf are foreground intensity values, rb,gb,bb are background intensity values, rmax,bmax,gmax are maximum r,g,b values respectively and m,n are number of foreground and background seeds provided by the user input.

3.4.2. Spatial color distance parameter

Spatial color distance parameter is the distance between R,G and B intensity values and normalized spatial coordinate values [normalized using (4) and (5)] of the user provided foreground and background seeds of the input image. It is computed as shown in (10)

$$d = \sum_{i=1}^{m}\sum_{j=1}^{n}\sqrt{\frac{(fx_i - bx_j)^2 + (fy_i - by_j)^2 + (rf_i - rb_j)^2 + (gf_i - gb_j)^2 + (bf_i - bb_j)^2}{m * n * rmax * gmax * bmax * fxmax * fymax}} \tag{11}$$

where rf,gf,bf are foreground intensity values, rb,gb,bb are background intensity values, rmax,bmax,gmax are maximum r,g,b values respectively, fx,fy,bx,by are normalized spatial coordinates normalized using the factor specified in (4) and (5) and and m,n are number of foreground and background seeds provided by the user input.

3.5. Lagrange saliency algorithm

The proposed Lagrange saliency algorithm is depicted in the flowchart shown in Fig.1. The pixel R,G,B values of the input seeds are extracted from the input image and the color distance is computed using technique mentioned in Section 3.4,1.

Based on the color distance(d), the images are segregated into 3 categories:

1. Category 1 (d<=d1(=0.25)) - These comprise of images whose foreground and background have similar color distribution. More seeds and spatial coordinates are needed to handle images belonging to this category. We use SLIC superpixels technique[2] to extract all pixels from a superpixel region corresponding to the seed from user provided input. Centroid of each unique superpixel region are used as seeds

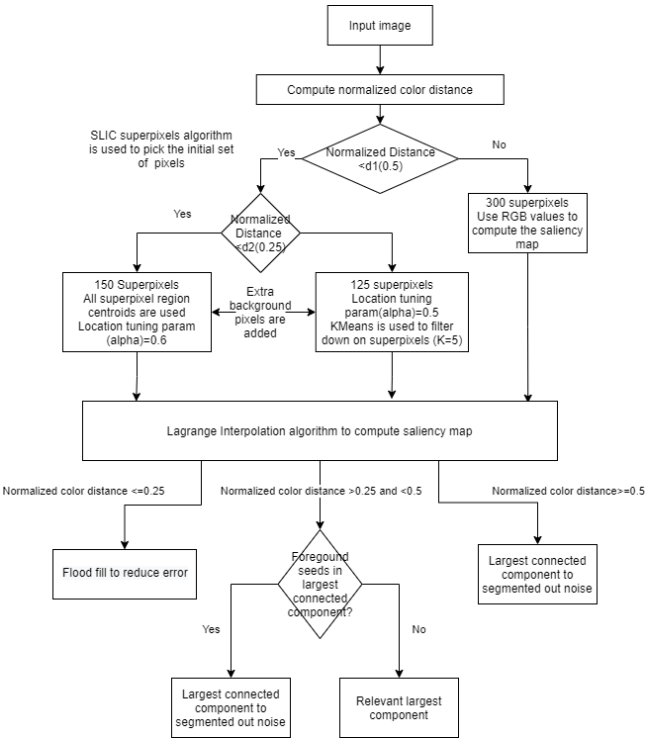

Fig. 1: Lagrange Saliency Algorithm

to be passed to Lagrange interpolation algorithm. We set the location tuning factor(α)=0.6 mentioned in sub-section 3.2.7 for this category of images. The images obtained are further manually thresholded with threshold 0.8 as specified in section 3.3.2. The resultant saliency map is further subjected to flood fill algorithm to fill any potential dark background regions within the foreground in the saliency map.

2. Category 2 (d>d1(=0.25) and d<d2(=0.5)) - There are intermediate tier images which need lesser number of seeds compared to category 1. We use SLIC superpixels technique to extract all pixels from a superpixel region corresponding to the seed from user provided input. KMeans is applied on the collected sample to obtained K(=5) centroids from the set of superpixels for foreground and background separately. We set the location tuning factor(α)=0.5 mentioned in sub-section 3.2.7 for this category of images since higher weightage will overoptimize and ends up adding unnecessary noise to the saliency map. The images obtained are further thresholded using Otsu's thresholding algorithm specified in secition 3.3.1. The largest connected component is extracted from the resultant saliency map to eliminate any potential noisy elements.

3. Category 3 (d>=d2(=0.5)) - This category comprises of images which can be segmented easily using only R,G,B values as seeds to the Lagrange interpolation algorithm. Once the saliency map is generated, the largest connected compo-

nent is extracted from the resultant saliency map to eliminate any potential noisy elements.

We also added additional background pixels along the borders of the image for category 1 and category 2 images since they use spatial coordinates as part of input being fed into Lagrange interpolation algorithm. This helps in eliminating the noisy pixels along the border of the resultant saliency map.

4. EXPERIMENTS AND RESULTS

The techniques involving PCA and colour averaging with or without channel-wise described in section 3.2.1-3.2.4 produced results which were inconsistent with grainy output. This was probably due to the similar colour composition of foreground and background causing the wrong interpolation of saliency values for the rest of the pixels. This was largely negated in the techniques used in section 3.2.4 where the location coordinates of the chosen pixels were used in addition to the red, green and blue (RGB) component. Also, it has been observed that PCA technique is highly sensitive to the scribbles chosen for foreground and background by the user. The number of foreground and background points chosen by user influences the saliency map generated - more distinguishing area of foreground covered by the scribbles, better are the results. The sample output on the MSRA10K dataset[10] of the color averaging and PCA are shown in Fig.2.

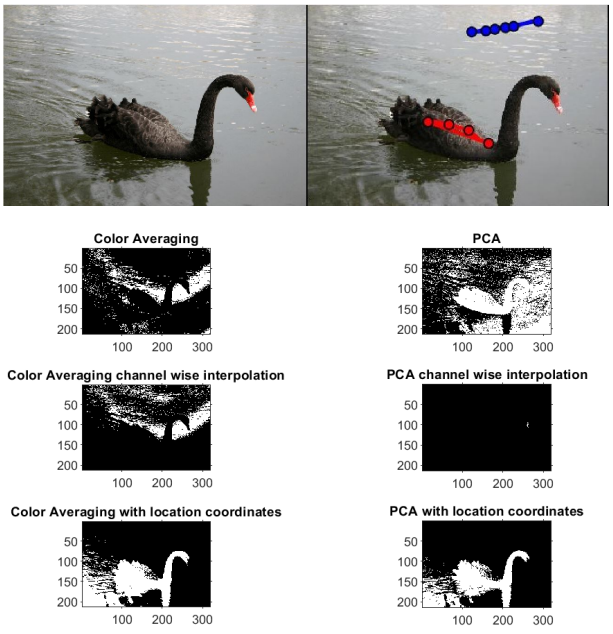

Fig. 2: Comparison of color averaging and PCA techniques

The images and ground truth data were taken from 47 images of Grabcut dataset[11] and was used to evaluate the algorithm. We used error rate represented in (12) and jaccard coefficient represented in (13) to evaluate the quality of the saliency maps.

Error rate[12] is defined as

$$\epsilon = \frac{(no.of misclassified pixels)}{(no.of pixels in unclassified region)} \tag{12}$$

Jaccard Coefficient is defined as

$$J(A,B) = \frac{(A \cap B)}{(A \cup B)} \tag{13}$$

where A is the obtained saliency map and B is the ground truth saliency map.

The KMeans technique mentioned in Section 3.2.5 solves the shortcomings of RGB and PCA based techniques by choosing better representative seeds (KMeans centroids). The error rate results are shown in Table 2 and Jaccard coefficient results are shown in Table 3. This technique has shortcomings for images whose foreground and background have similar pixel distribution as shown in Fig. 3 and Fig.4. This shortcoming is resolved to a large extent in the superpixel based seeds selection mentioned in 3.2.6.

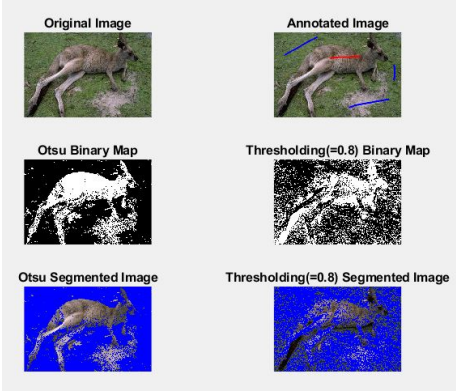

Fig. 3: Otsu and Thresholding(=0.8) Output

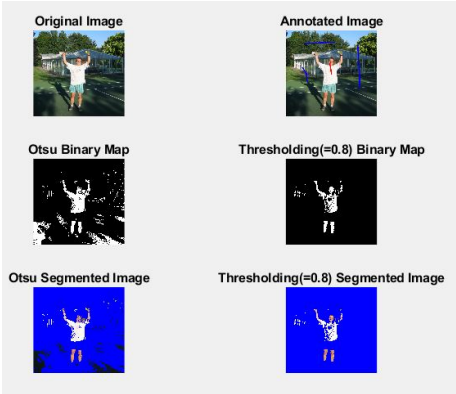

Fig. 4: Otsu output and Thresholding(=0.8) Output

Technique	Otsu	Thresholding(=0.8)
k=1	0.2381	0.1953
k=2	0.2185	0.2024
k=3	0.2252	0.1837
k=4	0.21	0.2024
k=5	0.1997	0.219
k=6	0.2355	0.2165
k=7	0.2271	0.2084
k=8	0.2269	0.2254
k=9	0.216	0.2312
k=10	0.2183	0.2277
Average Error Rate	0.2215	0.2112

Table 2: Error rate for KMeans based technique on Grabcut Dataset

Technique	Otsu	Thresholding(=0.8)
k=1	0.4154	0.3922
k=2	0.4554	0.4171
k=3	0.4564	0.4247
k=4	0.4654	0.408
k=5	0.4638	0.3825
k=6	0.4229	0.3995
k=7	0.4167	0.3841
k=8	0.4057	0.356
k=9	0.416	0.3475
k=10	0.4052	0.341
Average Jaccard coefficient	0.4323	0.3853

Table 3: Jaccard Coefficient for KMeans based technique on Grabcut Dataset

The final proposed algorithm in Fig.1 produced the best results in terms of error rate and Jaccard metric as specified in Table 4. The comparison of our technique with other state of the art techniques like SIOX[7],random walks[8] and GMMRF[14] is shown in Table 5.

The segregation of images based on color distance and having different seed choosing and thresholding techniques for each category enabled us to produce saliency maps of higher quality. The sample output from each category of images are shown in Fig. 5-7. The Lagrange saliency maps obtained for different images from the MSRC Grabcut dataset are shown in Fig. 8

Error Rate	Jaccard Coefficient
0.111	0.5865

Table 4: Final metrics of Lagrange Saliency Algorithm on Grabcut Dataset

We conducted another experiment with a different scribble set given in [13]. This scribble set comprises of scribbles in the bounded box region of interest of the foreground. The grabcut dataset [11] was used to evaluate the error rate and Jaccard coefficient using the bounded box scribble set and the results are shown in Table 6 and 7 respectively.

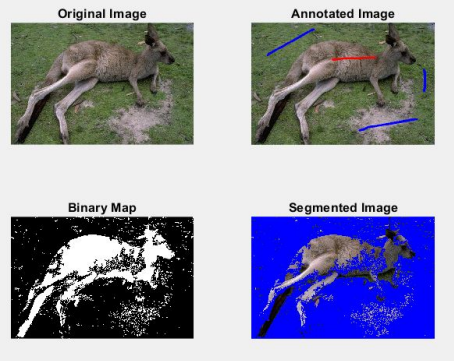

Fig. 5: Category 1 Output

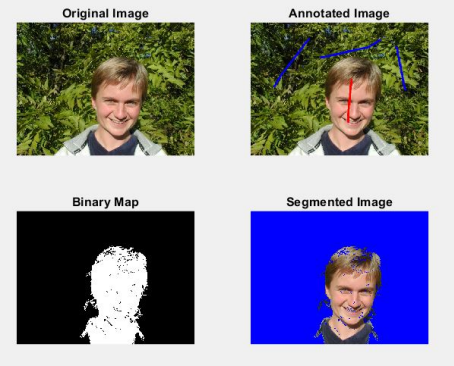

Fig. 6: Category 2 Output

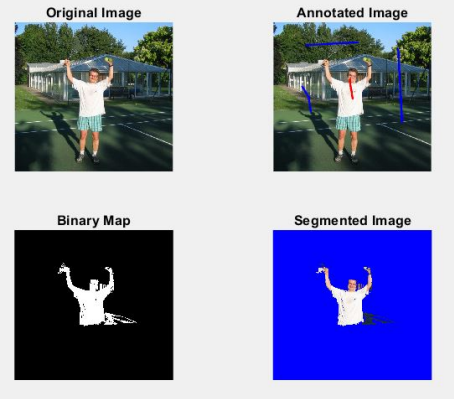

Fig. 7: Category 3 Output

Method	Error Rate
GMMRF [14]	7.9%
SIOX [7]	9.1%
Random walks [8]	5.4%
Proposed	11.1%

Table 5: Error Rate Comparison using the MSRC Grabcut dataset

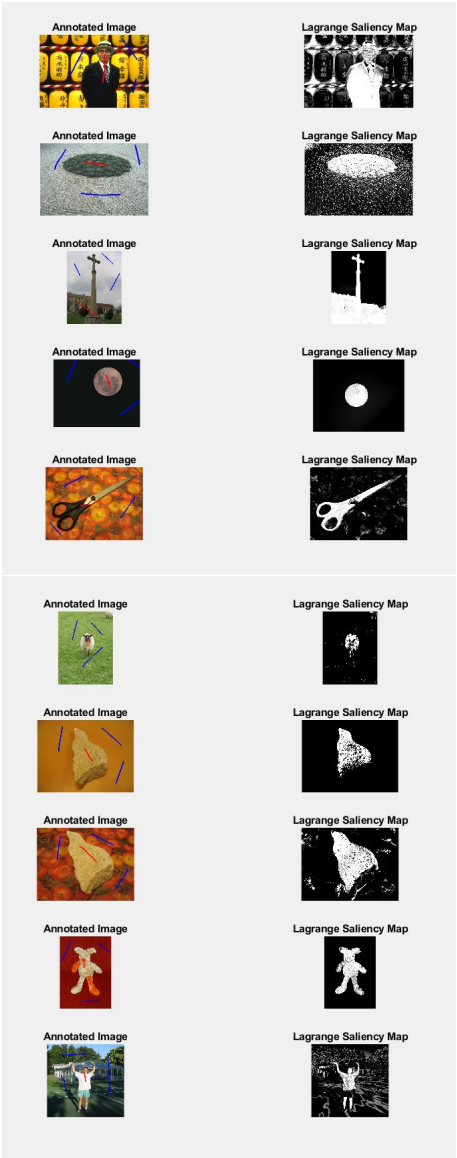

Fig. 8: Annotated image and Lagrange saliency map

Technique	Otsu	Thresholding(=0.8)
k=1	0.2317	0.1988
k=2	0.2325	0.1939
k=3	0.217	0.185
k=4	0.1886	0.1759
k=5	0.1977	0.175
k=6	0.1981	0.1992
k=7	0.2128	0.2029
k=8	0.2121	0.2153
k=9	0.2141	0.2139
k=10	0.1917	0.2217
Average Error Rate	0.2096	0.1982

Table 6: Error rate for KMeans based technique on Grabcut Dataset using scribble set of [13]

Technique	Otsu	Thresholding(=0.8)
k=1	0.4284	0.4217
k=2	0.4306	0.4175
k=3	0.4433	0.4199
k=4	0.4684	0.4357
k=5	0.461	0.4293
k=6	0.4651	0.4052
k=7	0.4449	0.3973
k=8	0.428	0.3908
k=9	0.4298	0.3857
k=10	0.4487	0.3796
Average Jaccard coefficient	0.4448	0.4083

Table 7: Jaccard Coefficient for KMeans based technique on Grabcut Dataset using scribble set of [13]

We compared the error rate and Jaccard Coefficient of scribble set 1 of the GrabCut dataset[6] and scribble set-2 [13] for different values of k as shown in Fig. 9-12. We see a slight improvement in the overall error rate and Jaccard coefficient as we added more foreground seeds from scribble set-2. However, the scribbles is not sufficient to handle category-1 images specified in Section 3.5 since most of the foreground and background scribbles are focused in the bounded box foreground object region of interest. The pixels with similar R,G,B values on the extremities have a very good chance of getting miss-classified since there is no representation of them in the input seeds. Also, the challenge remains for the images with multiple objects where the background scribbles do not cover the other objects which are similar to the primary object of focus.

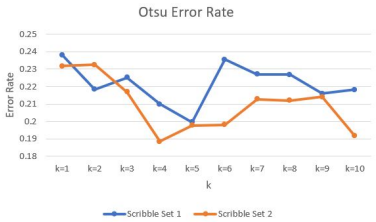

Fig. 9: Otsu Error rate comparision

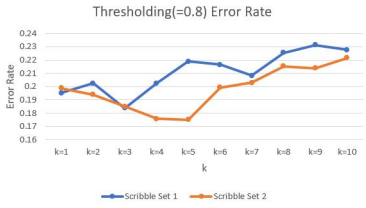

Fig. 10: Thresholding(=0.8) Error Rate Comparision

5. CONCLUSION

In this paper, we have proposed a novel interactive segmentation algorithm which takes spatial coordinates along with the RGB pixel coordinates into consideration while generating saliency map using Lagrange interpolation. We have

used KMeans[1], SLIC superpixels[2] algorithm to improve the seeds from user input, Otsu thresholding and morphological operations to improve the final saliency map. We have divided the Grabcut dataset[11] into 3 categories for the purpose of targeted seed selection and thresholding. However, there are still limitations with images which have similar foreground and background pixel value distribution. These images would require more foreground and background seeds to avoid fragmentation the saliency map. The images with multiple objects would need more seeds to cleanly segment out the focus of interest of the user.

We have compared our technique with other state of the art algorithms like SIOX[7], Random Walks[8], GMMRF[14] in Table 5 and the reason we have higher error rate is because we have not optimized the resultant salient map. In our future work, we will use graph based algorithms such as Grabcut to optimize the saliency map in order to reduce the error rate.

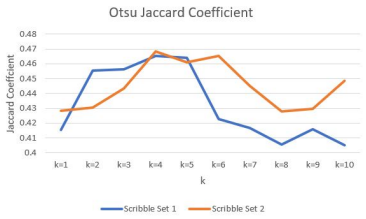

Fig. 11: Otsu Jaccard Coefficient comparision

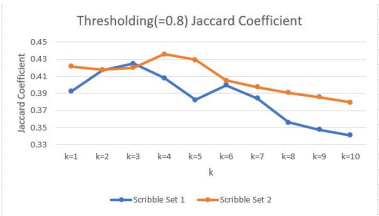

Fig. 12: Thresholding(=0.8) Jaccard Coefficient Comparision

6. REFERENCES

[1] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 881-892, July 2002, doi: 10.1109/TPAMI.2002.1017616.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süsstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 11, pp. 2274-2282, Nov. 2012, doi: 10.1109/TPAMI.2012.120.

[3] M. A. B. Siddique, R. B. Arif and M. M. R. Khan, "Digital Image Segmentation in Matlab: A Brief Study on OTSU's Image Thresholding," 2018 International Conference on Innovation in Engineering and Technology (ICIET), 2018, pp. 1-5, doi: 10.1109/CIET.2018.8660942.

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

[5] E. N. Mortensen and W. A. Barrett, "Interactive segmentation with in- telligent scissors," Graph. Models Image Process., vol. 60, no. 5, pp. 349–384, 1998

[6] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," ACM Siggraph, pp. 309–314, 2004.

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

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

[9] C. J. Moses, D. Selvathi and G. S. E. Queen, "Area efficient lagrange image up scaling architecture for multimedia applications," 2017 International Conference on Signal Processing and Communication (ICSPC), Coimbatore, 2017, pp. 201-205, doi: 10.1109/CSPC.2017.8305839.

[10] Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, Philip H. S. Torr, Shi-Min Hu, "Global Contrast based Salient Region Detection", vol. 37, no. 3, pp. 569-582, 2015, doi: 10.1109/TPAMI.2014.2345401.

[11] V. Gulshan, C. Rother, A. Criminisi, A. Blake and A. Zisserman, "Geodesic star convexity for interactive image segmentation," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 3129-3136, doi: 10.1109/CVPR.2010.5540073.

[12] W. Yang, J. Cai, J. Zheng and J. Luo, "User-Friendly Interactive Image Segmentation Through Unified Combinatorial User Inputs," in IEEE Transactions on Image Processing, vol. 19, no. 9, pp. 2470-2479, Sept. 2010, doi: 10.1109/TIP.2010.2048611.

[13] Andrade F., Carrera E. V., "Supervised evaluation of seed-based interactive image segmentation algorithms", In Proceedings of the 20th Symposium on Image, Signal Processing, and Artificial Vision, ISBN 978-1-4673-9461-1, Bogota, Colombia, pp. 225-231, September 2015

[14] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr, "Interactive image segmentation using an adaptive GMMRF model," in Proc. Eur. Conf. Computer Vision, 2004, pp. 428–441.

A. SOURCE CODE FOR LAGRANGE SALIENCY ALGORITHM

```
clear;
clc;
close all;

% folder="Lagrange Segmentation Output - SLIC Combined - 4";
%% folder="Lagrange Segmentation Output - SLIC Oxford GrabCut - Scribble Set";
% if ~exist(folder, 'dir')
%         mkdir(folder);
% end

%Read images
path="dataset-interactive-algorithms-flandrade\images";
warning('off');

files=dir(path);
files_count=length(files);

overall_error_rate_otsu=0;
overall_error_rate_80=0;
overall_jaccard_otsu=0;
overall_jaccard_80=0;
image_count=0;

for i=3:files_count
    image_count=image_count+1;
    file=files(i).name;
    disp(file);
    if contains(file,"png")
        img=imread(path+"\"+file,"png");
    end
    if contains(file,"bmp")
        img=imread(path+"\"+file,"bmp");
        file=strrep(file,".bmp",".png");
    end
    if contains(file,"jpg")
        img=imread(path+"\"+file,"jpg");
        file=strrep(file,".jpg",".png");
    end

%     output_file=folder+"/"+file;
%     imwrite(img,output_file);

    gt_path=strrep(path,"images","ground-truth");

    GT_img=imread(gt_path+"\"+file);

    scribbles_path=strrep(path,"images","scribbles-set-1");
    f=strrep(file,".png","-anno.png");
    img_scribbles_2=imread(scribbles_path+"\"+f);

    f_x=[];
    b_x=[];
```

```
f_y=[];
b_y=[];

[r,c]=find(img_scribbles_2==1);
f_x=[f_x',c'];
f_y=[f_y',r'];

[r,c]=find(img_scribbles_2==2);
b_x=[b_x',c'];
b_y=[b_y',r'];

I=double(img);
I1=I(:,:,1);
I2=I(:,:,2);
I3=I(:,:,3);

[P,Q,~]=size(img);

f_s=size(f_x,1);
b_s=size(b_x,1);
final_color_distance=0;

rmax=0;gmax=0;bmax=0;
fxmax=max(f_x);fymax=max(f_y);
bxmax=max(b_x);bymax=max(b_y);

%Compute color distance

for k=1:f_s
    for l=1:b_s
        if k~=l
            rk=double(img(f_y(k),f_x(k),1));
            rl=double(img(b_y(l),b_x(l),1));
            gk=double(img(f_y(k),f_x(k),2));
            gl=double(img(b_y(l),b_x(l),2));
            bk=double(img(f_y(k),f_x(k),3));
            bl=double(img(b_y(l),b_x(l),3));
            color_distance=round(sqrt((rk-rl)^2+(gk-gl)^2+(bk-bl)^2));
            final_color_distance=final_color_distance+(color_distance/(f_s*b_s))
                ;
        end
    end
end

final_color_distance=final_color_distance/(sqrt((255^2)*3));

disp(round(final_color_distance,2));


color_distance_limit=0.25;
color_distance_limit_2=0.5;

%add additional background seeds
```

```
if final_color_distance<color_distance_limit_2
    z=5;
    by=[z,z,P-z,P-z,round(P/4),round(P/2),round(3*P/4),round(P/4),round(P/2),
        round(3*P/4),z,z,z,P-z,P-z,P-z];
    bx=[z,Q-z,z,Q-z,z,z,z,Q-z,Q-z,Q-z,round(Q/4),round(Q/2),round(3*Q/4),round(Q
        /4),round(Q/2),round(3*Q/4)];
    b_x=[b_x',bx]';
    b_y=[b_y',by]';
end

%Choosing number of superpixels

if final_color_distance<color_distance_limit
    [L,NumLabels] = superpixels(img,150,'Method','slic');
else
    if final_color_distance<color_distance_limit_2
        [L,NumLabels] = superpixels(img,125);
    else
        [L,NumLabels] = superpixels(img,300);
    end
end

%Fetch all seeds from each of the mapped superpixel region

f_s=size(f_y,1);
Xval=zeros(size(f_s,1));

for j=1:f_s
   Xval(j)=L(round(f_y(j)),round(f_x(j)));
end

b_s=size(b_y,1);
Yval=zeros(size(b_s,1));
for j=1:b_s
   Yval(j)=L(round(b_y(j)),round(b_x(j)));
end

Xval=unique(Xval);
Yval=unique(Yval);

K=1;

%Superpixels seeds

error_rate_otsu_sum=0;
error_rate_80_sum=0;

jaccard_otsu_sum=0;
jaccard_80_sum=0;

prev_otsu=0;
prev_80=0;

best_otsu=0;
```

```
best_80=0;

t=1;

f=figure();
f.Name = file;

for index=1:t
    f2_x=[];f2_y=[];
    for j=1:size(Xval,2)
        [r,c]=find(L==Xval(j));
        if final_color_distance<color_distance_limit
            [~,y]=kmeans(r,K);
            [~,x]=kmeans(c,K);
            f2_x=[f2_x',x']';
            f2_y=[f2_y',y']';
        else
            f2_x=[f2_x',c']';
            f2_y=[f2_y',r']';
        end
    end

    b2_x=[];b2_y=[];
    for j=1:size(Yval,2)
        [r,c]=find(L==Yval(j));
        if final_color_distance<color_distance_limit
            [~,y]=kmeans(r,K);
            [~,x]=kmeans(c,K);
            b2_x=[b2_x',x']';
            b2_y=[b2_y',y']';
        else
            b2_x=[b2_x',c']';
            b2_y=[b2_y',r']';
        end
    end

    N2=round(double([f2_x;b2_x]));
    M2=round(double([f2_y;b2_y]));

    M=M2;N=N2;

    br=max(size(f2_x));
    [P,Q,~]=size(I);
    N(N2<1|N2>Q|M2<1|M2>P)=[];
    M(N2<1|N2>Q|M2<1|M2>P)=[];
    list=sub2ind([P,Q],M,N);

    if final_color_distance<=color_distance_limit_2
        [r,c,~]=size(img);

        alpha=0.6;
        if final_color_distance>=color_distance_limit
            alpha=0.5;
        end
```

```
    %Normalizing spatial coordinates

    f3_x=bsxfun(@times,f2_x,(alpha*255)/hypot(c,r));
    f3_y=bsxfun(@times,f2_y,(alpha*255)/hypot(c,r));
    b3_x=bsxfun(@times,b2_x,(alpha*255)/hypot(c,r));
    b3_y=bsxfun(@times,b2_y,(alpha*255)/hypot(c,r));

    M3=round([f3_x;b3_x]);
    N3=round([f3_y;b3_y]);

    ox=[M3,N3,I1(list),I2(list),I3(list)];
    [N1,M1]=ind2sub([P,Q],1:numel(I1));
    M1=bsxfun(@times,M1,(255*alpha)/hypot(c,r));
    N1=bsxfun(@times,N1,(255*alpha)/hypot(c,r));
    tt=[M1',N1',I1(:),I2(:),I3(:)];
else
    ox=[I1(list),I2(list),I3(list)];
    tt=[I1(:),I2(:),I3(:)];
end

%Setting final pixel values to be passed to Lagrange Interpolation
%algorithm

if final_color_distance<=color_distance_limit
    ff=ox(1:br,:);
    bb=ox(br+1:end,:);
    x=[ff;bb];
    y=[ones(size(ff,1),1);zeros(size(bb,1),1)];
else
    K=5;
    [~,ff]=kmeans(ox(1:br,:),K);
    [~,bb]=kmeans(ox(br+1:end,:),K);
    x=[round(ff);round(bb)];
    y=[ones(K,1);zeros(K,1)];
end

%Lagrange interpolation algorithm

fx=zeros(P*Q,1);
xs=size(x,1);
for k=1:xs
    tx=ones(P*Q,1);
    for l=1:xs
        if k~=l
            tx=tx.*(bsxfun(@minus,tt,x(l,:))/(x(k,:)-x(l,:)));
        end
    end
    fx=fx+tx*y(k);
end
fx(fx>1)=1;
fx(fx<0)=0;
fxi=reshape(fx,[P,Q]);
salient_img=mat2gray(fxi);
```

```
error_rate_otsu=0;
error_rate_80=0;


if final_color_distance>=color_distance_limit
    [counts,~] = imhist(fxi);
    T = otsuthresh(counts);
    salient_img_otsu = imbinarize(fxi,T);
    CC = bwconncomp(salient_img_otsu);
    numOfPixels = cellfun(@numel,CC.PixelIdxList);
    [~,indexOfMax] = max(numOfPixels);
    salient_img_otsu_largest_component = zeros(size(salient_img_otsu));
    salient_img_otsu_largest_component (CC.PixelIdxList{indexOfMax}) = 1;

    A=sub2ind([P Q],f_y,f_x);
    B=find(salient_img_otsu_largest_component==1);
    %If largest component doesn't have an overlap with foreground seeds
    %provided.
    while numel(intersect(A,B))==0
        salient_img_otsu(CC.PixelIdxList{indexOfMax}) = 0;
        CC = bwconncomp(salient_img_otsu);
        numOfPixels = cellfun(@numel,CC.PixelIdxList);
        [~,indexOfMax] = max(numOfPixels);
        salient_img_otsu_largest_component = zeros(size(salient_img_otsu));
        salient_img_otsu_largest_component (CC.PixelIdxList{indexOfMax}) =
            1;
        A=sub2ind([P Q],f_y,f_x);
        B=find(salient_img_otsu_largest_component==1);
    end
    error_rate_otsu=error_metric(salient_img_otsu_largest_component,GT_img);
    error_rate_otsu_sum=error_rate_otsu_sum+round(error_rate_otsu,3);

    jaccard_otsu=jaccard_metric(imbinarize(
        salient_img_otsu_largest_component),imbinarize(GT_img));
    jaccard_otsu_sum=jaccard_otsu_sum+round(jaccard_otsu,3);

    if ((error_rate_otsu<prev_otsu)||(index==1))
        best_otsu=salient_img_otsu_largest_component;
    end

    prev_otsu=error_rate_otsu;
else
    T2=0.8;
    salient_img_final = imbinarize(fxi,T2);
    salient_img_final=imfill(double(salient_img_final),'holes');

    error_rate_80=error_metric(salient_img_final,GT_img);
    error_rate_80_sum=error_rate_80_sum+round(error_rate_80,3);

    jaccard_80=jaccard_metric(imbinarize(salient_img_final),imbinarize(
        GT_img));
    jaccard_80_sum=jaccard_80_sum+round(jaccard_80,3);
```

```
            if ((error_rate_80<prev_80)||(index==1))
                best_80=salient_img_final;
            end

            prev_80=error_rate_80;
        end
    end

    if final_color_distance>=color_distance_limit
        error_rate=round(error_rate_otsu_sum/t,3);
        jaccard=round(jaccard_otsu_sum/t,3);
        overall_error_rate_otsu=overall_error_rate_otsu+error_rate;
        overall_jaccard_otsu=overall_jaccard_otsu+jaccard;
        k="Error rate~"+round(error_rate,3)+"["+final_color_distance+"] Jaccard~"+
            round(jaccard,3);
        ax = subplot(1, 1, 1);
        imshow(best_otsu,"Parent",ax),title(k);
        disp(error_rate);
        disp(jaccard);
    else
        error_rate=round(error_rate_80_sum/t,3);
        jaccard=round(jaccard_80_sum/t,3);
        overall_error_rate_80=overall_error_rate_80+error_rate;
        overall_jaccard_80=overall_jaccard_80+jaccard;
        k="Error rate~"+round(error_rate,3)+"["+final_color_distance+"] Jaccard~"+
            round(jaccard,3);
        ax = subplot(1, 1, 1);
        imshow(best_80,"Parent",ax),title(k);
        disp(error_rate);
        disp(jaccard);
    end

%       output_file=strrep(output_file,".png","");
%       saliency_map=strrep(output_file,".jpg","")+"_saliency_map.jpg";
%       saveas(f,saliency_map);
    disp("");
end
overall_error_rate=overall_error_rate_otsu+overall_error_rate_80;
overall_jaccard=overall_jaccard_otsu+overall_jaccard_80;
disp("Overall Error Rate");
disp(overall_error_rate/image_count);
disp("Overall Jaccard Coefficient");
disp(overall_jaccard/image_count);
close all;

function [error_rate] = error_metric(salient_img,GT_img)
error_rate=sum( sum( salient_img ~= imbinarize(GT_img) ) ) / numel(salient_img);

function [jaccard] = jaccard_metric(salient_img,GT_img)
jaccard= sum(salient_img & GT_img)/sum(salient_img | GT_img);
```