# Interactive Image Segmentation – Implementation and Testing on PC and Smartphone

Yu Lu

*Final Project Report for CS6241: Advanced Topics in Computer Graphics*

(a) User-marked Image (b) Segmented Foreground

Fig.1 The interactive image segmentation implemented in this project – (a) user-marked pixels are labelled in red (foreground) and blue (background) (b) the result after segmentation

***Abstract* — an interactive image segmentation algorithm is investigated and implemented in this project. The approach follows the well-known image cut-out method "Lazy Snapping" [1]. Lazy snapping solves the binary-segmentation (foreground-background) problem by making use of some intuitive user inputs, where small sets of foreground and background pixels are annotated through simple scribbling. After introducing a pre-computed over-segmentation procedure, feedback from user input can be received instantly. Implementation on a touch screen smartphone is conducted. Performance and usability of the method are analysed and discussed.**

***Keywords***: Lazy Snapping, Interactive Image Segmentation, Graph Cut, Smartphone

## I. INTRODUCTION

Image segmentation refers to the technique partitioning an image into multiple non-overlapping segments. The goal of segmentation is to simplify or change the representation of an image into something that is more meaningful and easier to use. Researchers have investigated in various approaches to this problem on computers. This project intends to explore interactive image segmentation techniques on both PC and Smartphone. Recent years, smartphone has gained its popularity as image capture devices in daily life, mainly due to its portability and ease of information-sharing. Simple image editing applications are also popular among users. It is observed that touch-screen of the smartphone exhibits an excellent user interface for region-based segmentation application utilizing loose indications from users.

## II. RELATED WORK

For general image segmentation, there are two main approaches: boundary-based and region-based. Widely-known applications of these techniques are Photoshop's magnetic lasso tool and magic wand tool. More advanced region-based segmentation techniques achieve efficient result by implementing some underlying optimization algorithms. Lazy Snapping is a famous image cut-out tool combining boundary-based and region-based segmentation techniques with an intuitive UI [1].

Early work on image cut-out was mainly boundary-based approach. Intelligent Scissors allow user to choose an optimal contour by roughly tracing the object's boundary with the cursor [2]. Image Snapping moves the cursor location to nearby features in the image to obtain neat snapping edges [3]. Graph Cut was introduced as a powerful optimization technique which can be used to achieve robust segmentation [4]. Based on a modification on the Graph Cut, GrabCut proposed an iterative energy minimization for segmentation and a border matting algorithm for better boundary selection, which achieves accurate segmentation result with very clean cut-lines [5]. A more recent work, Paint Selection presents a progressive painting-based tool for local selection which can efficiently handle multi-megapixel images [6]. Beyond the still image processing, Video SnapCut is presented as a robust interactive image cut-out tool for video input [7].

## III. ALGORITHM AND IMPLEMENTATION

This section discusses the details of the approach. An overview of the system is first described (*Section A*), followed by a thorough investigation on the main algorithms used to solve the problem (*Section B).*

### *A. General Approach*

The approach basically involves four steps. First a mask is applied to the operation image (Fig.2(b)). The user scribbled pixels are highlighted in red (foreground) and blue (background) on the mask. The pixels in the original image that are under the scribbled lines on the mask are classified as sets of *Foreground Seeds* and *Background Seeds* respectively.

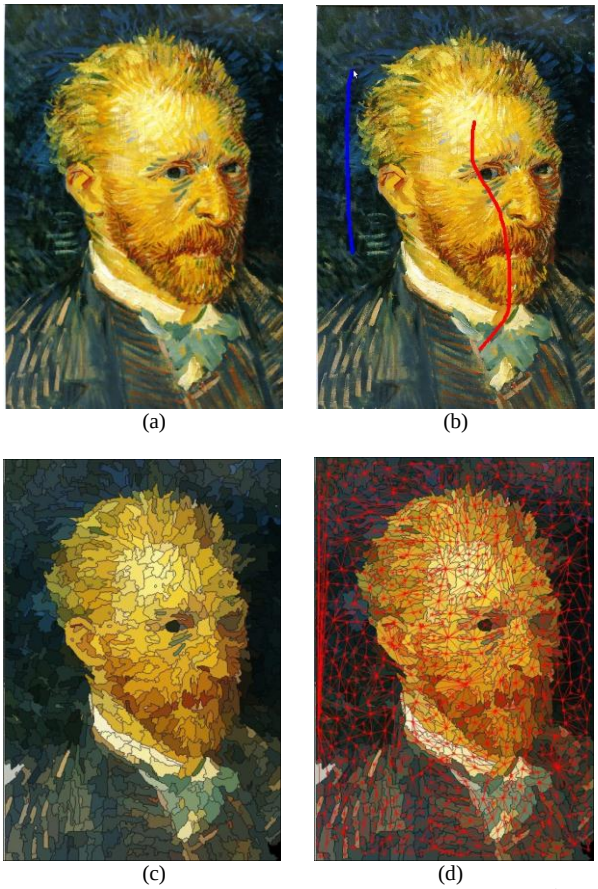
The original image is then segmented into small fragments using the watershed segmentation algorithm [8]. Each fragment is considered as a *Super Pixel* whose colour is assigned with the mean colour of the all pixels enclosed in the region as shown in Fig.2(c). The watershed algorithm groups similar pixels in a local area while preserving the boundary information. This over-segmentation step accelerates the optimization process comparing to the use of single pixels.

The segmentation of the pre-segmented image is then formulated as a binary labelling graph cuts optimization

approach. The resulted labelling image contains only either foreground labels or background labels. A visualization of graph model is shown in fig.2(d).

At the last stage, a connected-component analysis based on blob extraction algorithm is apply to the image so that only the most significant foreground area – the one with the largest area is segmented out as, while other smaller areas are discarded. This stage filtered the noise and faults resulted from the last step.





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

Fig.2 (a) Original image (b) User-marked mask (c) Visualization of the over-segment image (d) Graph generated by linking the neighbouring Super Pixels

### B. *Model Formulation – Graph Cuts Optimization*

Image segmentation problem is formulated in terms of energy optimization in this approach. The image can be represented as a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where the pixels in an image are considered as nodes $\mathcal{V}$ which are linked with their 4-connected neighbours with edges $\mathcal{E}$. Segmentation is considered as a binary labelling problem, in which either a foreground label or background label is to be assigned to each pixel. Following the definition by Boykov and et al in [9], the energy function considered in this case consists of the two terms – a data term and a smooth term. The labelling f assigns each pixels $p \in P$ a label $f_p \in L$:

$$E(f) = E_{data}(f) + E_{smooth}(f)$$

Data term $E_{\text{data}}$ measures the disagreement between the label $f$ and the observed data. The smooth term $E_{\text{smooth}}$ measures the extent to which $f$ is not piecewise smooth. Here how to define these two energy terms for each pixel in the image is discussed below.

#### 1) *Data Energy* $E_{data}(f)$

The data term can be formulated as below by considering each pixels in the image.

$$E_{data}(f) = \sum_{i \in \mathcal{V}} E_i(x_i)$$

The energy is defined as the cost of assigning the label to the pixel. Therefore a false assignment is expected to result in a higher cost. The criteria for calculating the cost of assigning a label to the unclassified pixel is determined derived from the colour similarity between the pixel and the *Seeds*. The *Seeds* are first clustered by performing a k-means clustering in RGB colour space. When an unclassified pixels is processed, the two distances from it to the nearest foreground cluster $d_i^{\mathcal{F}}$ and background cluster $d_i^{\mathcal{B}}$ are used to calculated the cost function (shown in Fig.3).

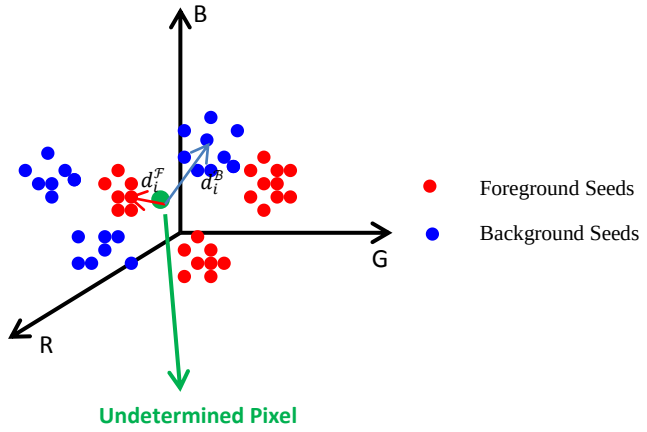



Fig.3 Foreground clusters and background clusters in colour space. The distances between the input pixel and its nearest foreground cluster and background cluster is calculated.

The complete formulation of data term following that proposed by Lazy Snapping [1] as shown:

$$\begin{cases} E_i(x_i = 1) = 0 & E_i(x_i = 0) = \infty & \forall i \in \mathcal{F} \\ E_i(x_i = 1) = \infty & E_i(x_i = 0) = 0 & \forall i \in \mathcal{B} \\ E_i(x_i = 1) = \dfrac{d_i^{\mathcal{F}}}{d_i^{\mathcal{F}} + d_i^{\mathcal{B}}} & E_i(x_i = 0) = \dfrac{d_i^{\mathcal{B}}}{d_i^{\mathcal{F}} + d_i^{\mathcal{B}}} & \forall i \in \mathcal{U} \end{cases}$$

Here $\mathcal{U}$ is the undetermined region. This formulation gives infinite cost when an incorrect assignment for the marked pixel is made. And it gives higher cost when distance in colour space between the pixel and the nearest cluster centre is large.

#### 2) *Smooth Energy* $E_{smooth}(f)$

Smooth term can be formulated for each pixel as follows:

$$E_{smooth}(f) = \sum_{i,j \in \mathcal{E}} E_{i,j}(x_i, x_j)$$

Smooth term enclose the information of the colour similarity between neighbouring pixels. Large difference between adjacent pixels should indicate a smaller cost since it

should be more likely to be a boundary. Therefore the definition for the smooth term is:

$$E_{i,j}(x_i, x_j) = |x_i - x_j| g(C_{i,j})$$
$$\text{Where } g(\xi) = \frac{1}{1+\xi} \text{ and } C_{i,j} = \|C(i) - C(j)\|^2$$



The distance C between pixels is also calculated in RGB colour space as the one used in data term.

The total Energy term can then be formulated:

$$E = \sum_{i \in \mathcal{V}} E_i(x_i) + \lambda \sum_{i,j \in \mathcal{E}} E_{i,j}(x_i, x_j)$$

It is noted that there is a weighting term $\lambda$ which needs to be carefully selected to balance the effects of two energy terms. A too small $\lambda$ eliminates the smoothness cost, resulted that the object boundary to be incorrectly determined. Too large of the $\lambda$ eliminates the user-identify information for foreground and background classifying while gives the segmented areas based purely on the edge information detected in the original image.

Solving the graph cuts optimization problem can be refer to the work of Boykov [9]. Here the open source provided by the author is used.

### C. Accelerated Model

It has been observed that using graph cuts optimization in solving the problem fails to achieve an interactive processing time. As described in *section A*, a pre-computed segmentation process is first performed, and the graph is generated from the over-segmented *Super Pixels*. Since the number of nodes is significantly reduced in this case, a much faster performance is achieved. A comparison of the two models is summarized in table.1 for images in different dimensions.

| Image | Dimension | Time (Pixel Model) | Time (Super Pixel Model) |
|---|---|---|---|
| Fruits | (512, 480) | 0.851s | 0.053s |
| Painting | (554, 800) | 1.95s | 0.054s |
| Flower | (800, 600) | 1.34s | 0.054s |
| Fish | (1024, 768) | 1.65s | 0.047s |
| Lady | (1024, 683) | 4.46s | 0.098s |

*Number of Super Pixel at 10000 (100 x 100)
*Weighting parameter $\lambda$ at 1000
*Test on a MacBook Pro laptop with 2.8 GHz dual-core CPU and 4GB Memory

Table.1 Comparison of Processing Time of Two Models

## IV. Results and Discussion

Some tested result is shown in Fig.4. A relatively good segmentation result can be observed. However unsmooth boundaries sometimes are obvious. And loss of information on foreground segments is also noticed. The limitations are discussed below. At last an evaluation of the algorithm on smartphone is presented and the possibility of fully implementing the system on the smartphone is discussed.

### A. Limitations

#### 1) Pixel loss in thin areas

As shown in Fig.5(e), the pixels of the foreground in the very thin areas are segmented into the background. The reason is that the minimum size of the *Super Pixels* is limited by the number we can handle. When the size of the area to segment is smaller than the one of the *Super Pixel*, it will be merge into the surround area so that cannot be separated out. Similar visual flaws is found in Fig.4(d), the segmentation is not perfect because the foreground image is small in size.

#### 2) Unsmooth boundaries

Fig.5(c) shows unsmooth boundaries of the segmentation. This is due to the color similarity at the edge of foreground and the background. This is an unavoidable issue of auto-segmentation tool, the original algorithm of Lazy Snapping introduced a refine tool to adjust the boundaries for better result.

#### 3) Poor performance when foreground and background have similar colour distribution

The performance becomes poorer when foreground and background have similar colour distribution since the algorithm depends mainly on the colour difference for classification (Fig.5 (f)). To supply more labeled pixels to the algorithm by selecting large areas can significantly improve the results. Since the tool is working in an interactive rate, it is efficient for the user to constantly improve the pixel selection to guide the segmentation.

### B. Test on Smartphone

The proposed interactive image segmentation system is implemented on a touch screen smartphone (Fig.5). OpenCV Library on Android is used in this implementation. Test of the performance indicates that further optimization is required to achieve interactive rate.

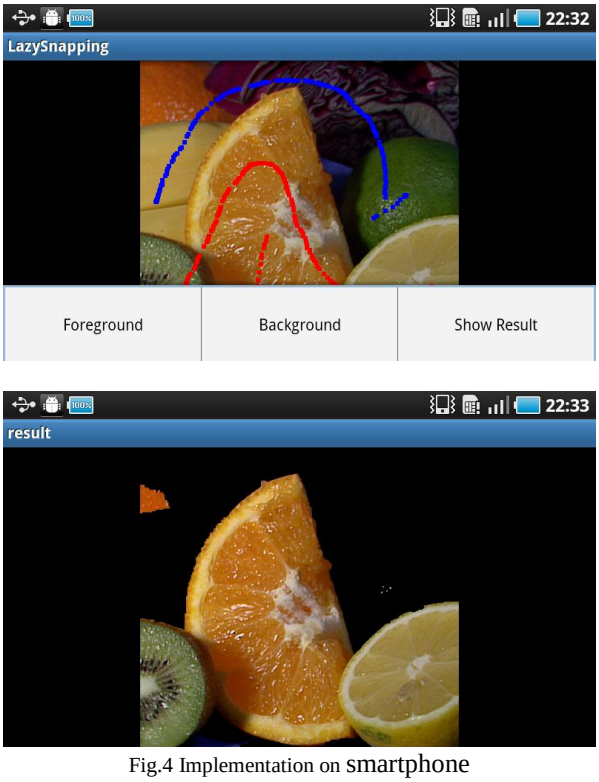
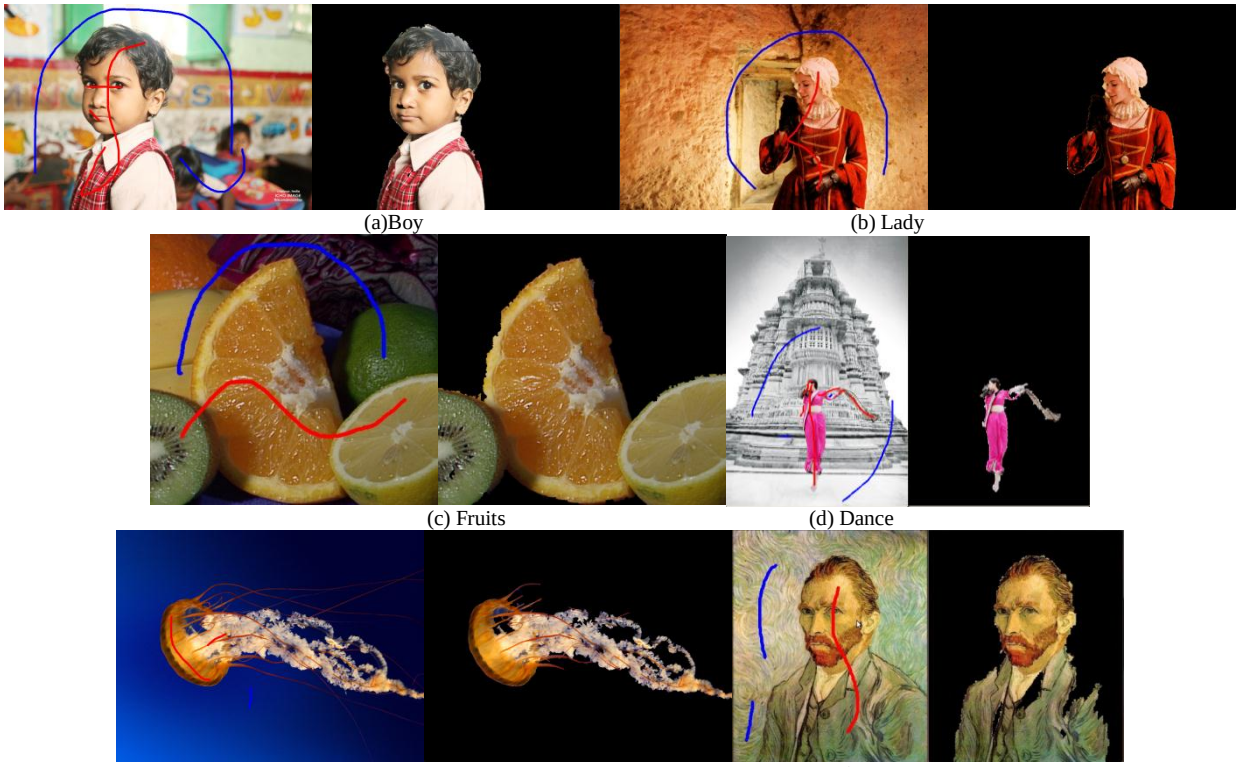



Fig.4 Implementation on smartphone



(a)Boy (b) Lady

(c) Fruits (d) Dance

(e) Sea (f) Paint

Fig.5 Some Test Results

## V. Conclusions

This project is a simple implementation of the interactive segmentation tool Lazy Snapping. The algorithm and the implementation is discussed in detail. On the PC, an interactive rate can be achieved with a relatively good performance. However it performs slower on the smartphone due to the low computational power of embedded system.

## References


[1] Y. Li, J. Sun, C. K. Tang and H.Y. Shum, "Lazy snapping." In *Proceedings of ACM SIGGRAPH' 04*. 2004.

[2] E. N. Mortensen and W. A. Barrett, "Intelligent scissors for image composition." In *Proceedings of ACM SIGGRAPH' 95*. 1995.

[3] M. Gleicher, "Image snapping." In *Proceedings of ACM SIGGRAPH' 95*. 1995.

[4] Y. Boykov and M.P. Jolly, "Interaction graph cuts for optimal boundary & region segmentation of objects in n-d images." In *Proceedings of ICCV 2001*. 2001.

[5] C. Rother, V. Kolmogorov and A. Blak, "GrabCut: interactive foreground extraction using iterated graph cuts." ACM Transactions on *Graphics (TOG) - Proceedings of ACM SIGGRAPH 2004. 23.*2004.

[6] J. Liu, J. Sun and H.Y. Shum, "Paint Selection." In Proceedings of ACM *SIGGRAPH' 09.* 2009.

[7] X. Bai, J. Wang, D. Simonsand and G. Sapiro, "Video SnapCut: Robust Video Object Cutout Using Localized Classifiers." In *Proceedings of ACM SIGGRAPH' 09.*2009.

[8] F. Meyer, "Color Image Segmentation." In Proceeding of the *International Conference on Image Processing and Its Processing and Its Application* pp 303-306, 1992.

[9] Y. Boykov, O. Veksler and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts." IEEE Transactions on *Pattern Analysis and Machine Intelligence, vol.23, no.11, 2001.*