Object selection

Final Year Project Final Report

Nguyen Van Khoi and Tran Hoang Nam

A thesis submitted in part fulfilment of the degree of BSc. in Computer Science with the supervision of Tu Trung Hieu.

Bachelor of Computer Science
Hoa Lac campus - FPT University

08 August 2016

Abstract

In this thesis, we introduce the Graph Cut solution of interactive segmentation and its derivative – the Lazy Snapping.

The interactive Graph Cut segmentation for the multi-dimensional image provided by Jolly and Boykov in 2001. The hard constraints are the user marked pixels as “object” and “background” while the soft constraints involving both boundary and region properties.

The fast implementation provided by Boykov and Kolmogorov of min-cut max-flow is used to find the optimal segmentation. This solution gives the balance of boundary and region properties among all segmentation satisfying the constraints. The experimental results in term of photo editing are presented. It works quite well with small and medium resolution multi-dimensional images which have various color distribution.

In 2004, Li, Tang, Sun, and Shum published the Lazy Snapping method basing on Jolly and Boykov research. Instead of working directly on the pixel, the pre-segmentation step is used to grouping pixel into many segments. Each one is considered as super-pixels the graph and energy function is constructed similarly to Jolly-Boykov’s one.

We also renewed the existing Lazy Snapping method [2] by using another algorithm in the pre-segmentation step. By replacing the traditional watershed [14] in the Lazy Snapping framework with the SEEDs [8], we can achieve a simpler implementation and usable result. The experimental results show that we can reach the same accuracy as Lazy Snapping without using second refining step. Finally, our method is quite fast with some simple steps.

Acknowledgements

First and foremost, our heartfelt gratitude goes to our supervisor, Mr. Tu Trung Hieu for his extremely valuable advice, comments, instructions, correction and encouragement for us during the research process. We do appreciate the effort and time he spent to guide us from the starting point when we learned the first definition of Image Processing like a teacher, and also a friend. It is an amazing time of hard-working, testing, failure and do-it-again. Thank you a lot, Henry!

We also want to share this moment with our family – who always believe and beside us. We will not go far this way without the unlimited love from them. And now, they can be proud of ourselves and what we did. It is best gift for our family. Thank you for always being by my side!

We cannot forget the truly friends who stayed, learned and tried in FPT University. We joined and shared valuable student time together. Thank you, guys! We will always remember CS0801.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 The interactive image segmentation problem

An image is a set of sub-images that sometimes referred to as regions-of-interest, ROIs, or simply regions. One of the problem in image processing is image selection – how to cut out or select an area for later specific image processing operations.

Interactive single-image segmentation (image cutout or foreground/background separation), one of the typical case in region selection problem, is to select the object (e.g. human, flower) from the background that based on samples provided by the user. The result can be used to photo collaging or identifying.

The purpose of image cutout is to determine which parts of the original image belong to "object" and which belong to the background. On one hand, it is quite easy for human perspective to specify. On the other hand, the computer is not smart enough to achieve this purpose without human assistance. It would take a long way such that the computer understands cognitive image at the human level. So that, the user has to provide samples in both foreground and background individually, with pixel accuracy.

Although this problem has a long history in computer vision, how to find an efficiency solution with high accuracy in pixel level is still a challenge. Boykov and Jolly had provided an approach in 2001 [1]. They provided a very general energy minimization framework that is at the heart of many modern techniques. A user gives foreground/background strokes or a coarse bounding box of the object in the image. The algorithm is based on this information to learn foreground/background color feature, and then add constraints to make energy minimization problem easier. It on region-based methods – assigns pixels to a given region based on the degree of mutual similarity. For the general F/B separation problem, based on the uses of samples, there are two primary methods: boundary-based and region-based. Both of them use the features of selected ones to automate foreground specification.

In the boundary-based method, the user surrounds and traces the object boundary while the system optimizes that curve. The user has to control the curve manually since there are many "minimum cost contour" for highly textured or un-textured image. The user should also have the ability to get progress overview in close control. For high-resolution images, enclosing entire boundary is very costly while zooming in or out at dragging borderline is not easy. Once it is finally computed, then the tool is no longer helpful, and errors must be cleared. Examples of this method include intelligent scissor [Mortensen and Barrett 1995; Mortensen and Barrett 1999], image snapping [Gleicher 1995] and Jetstream [Perez and Blake 2001].

In the region-based method, the user may specify some hints. The hint can be the region contain the object or some samples of object and background. Meanwhile, the underlying optimization extracts actual object boundary based on these. The user can operate at any scale they want. The process can show the partial results after each hint inputting. However, shadow, low-contrast edges or other ambiguous areas become troubles to hint or even must be manually made by hand. Some examples of this method are magic wand in Photoshop, intelligent paint [Reese and Barrett 2002; Barrett and Cheney 2002], marker drawing [Falcao et al. 2000], sketch-based interaction [Tan and Ahuja 2001], interactive graph cut image segmentation [Boykov and Jolly 2001], GrabCut [Rother et al. 2004]) and interactive image Photomontage [Agarwala et al. 2004].

In our thesis, we focus on interactive graph cut image segmentation [1], a typical example of the region-based approach for later solutions, and lazy snapping [2], a great combination of quick

hinting (of region-based approach) and pixel-accurate boundary editing (of boundary-based approach).

1.2 Image segmentation as graph cut

Interactive graph cut image segmentation [1] is a region-based method for segmenting N-dimensional image. It converts the image into network flow, which each internal node corresponds to a node, the source and sink are object and background terminal correspondingly. The edges of graph are weighting based on boundary and region properties of adjacency node. Then, max-flow is solved to find minimum cut correspond to optimal segmentation. A better max-flow algorithm is provided by Boykov and Kolmogorov.

1.3 Improving lazy snapping

Image segmentation can be known as the division of an image into meaningful structures is often an essential step in image analysis, object representation, visualization, and many other image processing tasks.

In other words, image segmentation focus on methods that find the particular pixels that make up an "object". There were many different segmentation approaches in the recently decades. In 2002, Li et al [23] divided the segmentation approaches into five categories that present in the following table:

Table 1. Segmentation approaches

Threshold-based segmentation	Histogram thresholding and slicing techniques are used to segment the image. They may be applied directly, also combined with pre- and post-processing techniques
Edge-based segmentation	With this technique, detected edges are assumed to represent object boundaries and used to identify these objects.
Region-based segmentation	Where an edge-based technique may attempt to find the object boundaries and then locate the object itself by filling them in, a region-based takes the opposite approach, by (e.g.) starting in the middle of an object and then "growing" outward until it meets the object boundaries.
Clustering techniques	Clustering is known as a synonym for (agglomerative) segmentation. Here it is used to denote techniques that are primarily used in exploratory data analysis of high-dimensional measurement patterns. In this context, clustering methods attempt to group together patterns that are similar in some sense. This goal is very similar to what we want do when we segment an image, and indeed some clustering techniques can readily be applied for image segmentation.
Matching	When we know what an object we wish to identify in an image (approximately) looks like, we can use this knowledge to locate the object. This approach to segmentation is called matching.

Lazy Snapping paper presents an interactive image cutout tool with two steps: a quick object marking step and a boundary editing step to separate objects from the background. With an improvement by combining graph cut with pre-computed segmentations, the results are accurate with less time than working on individual pixel level in object marking step. It also provides two simple events to increase the accuracy of the boundary. Moreover, Lazy Snapping provides instant visual feedback, based on the cutout contour to the actual object boundary despite the noise of ambiguous

or low contrast edges. That makes the different of Lazy Snapping – the combination of region-based segmentation and edge-based segmentation.

Based on the Lazy Snapping paper, we have simple method overview:

- Goal: separate object from background
- First step: foreground/background selection
    - Samples are given by users in each part of image
    - Solve minimum cut problem on segment level
- Second step: boundary correction
    - User provides some strokes on incorrect parts of object
    - Solve minimum cut problem on pixel level

The new thing in Lazy Snapping method is a novel graph cut formulation which is built on a pre-computed image over-segmentation, instead of pixel. By using watershed algorithm [14] to divide the image into many small regions, each node in graph cut problem present for a segmented region. By that, we have a new network, while the nodes V are the set of all small regions from pre-segmentation step, and the edges E are the set of all arcs connecting adjacent regions.

By reducing the number of nodes and edges in the new network, the speed of new algorithm increases significantly. When watershed segmentation provides good segments, we can get good object without boundary editing step.

With the same idea of the authors in Lazy Snapping paper about a pre-computed image segmentation before apply to Graph Cut solution, I built my own framework with an understanding about several pre-segmentation methods:

- Watershed by Vicent and Soille, 1991
- Watershed by Meyer, 1992
- Kmeans by Duda and others, 2000
- SEEDs superpixels by M. van den Bergh et al, 2012
- SEEDs Revised Mean Pixels by D. Stutz, A. Hermans, B. Leibe, 2014

Step by step, my framework achieved different results with each method. Finally, the combination between SEEDs Revised Mean Pixels and Graph Cut Solution give acceptable results. In my experiment, I focus on improvement of Object Marking Step to separate objects without using boundary editing step. In general, the results of my framework are closed with results in Lazy Snapping paper.

SEEDs Revised Mean Pixels method is very useful by allowing developer control and change important parameters such as the number of segments and iterations as well as level of the smooth boundary. By that, we can avoid the over-segmentation problem but still can get acceptable results with reducing the time of the framework.

1.4 Contributions

The main contribution of this thesis are:

- Testing implementations that replace traditional watershed [14] of Lazy Snapping with others segmentation algorithms such as K-means [13], Distanced Transform Watershed [15] and SEEDS [7].
- An implementation of **Silent Lazy Snapping** - an interactive image cutout framework which separate the foreground and background with some simple steps.
- A comparison between several segmentation algorithms which can be applied to pre-segmentation step that tested in Berkeley Segmentation Dataset and our own dataset.

The results of Berkeley Segmentation Dataset and my own dataset is made publicly available together with my implementation of Silent Lazy Snapping.

a. Original image *b. Lazy Snapping result **with** boundary editing* *c. Silent Lazy Snapping result **without** boundary editing*

Figure 1. Results of Lazy Snapping method and Silent Lazy Snapping method

Without boundary editing step, the result from Silent Lazy Snapping is usable. In this thesis, my contribution focuses on how to improve the efficiency and the speed of object marking step, not similar with original paper. There are many methods to the image-cutout problem so far, but it is not easy to solve this in one step. With the limitation of my knowledge in student level, I did some researches to create Silent Lazy Snapping method.

1.5 Organizations

We will review related works in chapter 2, and our contributions in chapter 3 and chapter 5. Finally, we will summarize our works and future directions in chapter 5.

This thesis is organized to review the image segmentation problem and introduce two solutions.

Chapter 2 will show the literature review of image segmentation problem and some approaches. Basic concepts and some typically related works will be shortly described for betting understanding.

Chapter 3 and 4 introduces and describe interactive graph cut and lazy snapping correspondingly. At the end of each chapter are our experiments and the conclusion.

Chapter 5 is for conclusion.

2 Literature Review

2.1 On the lazy snapping framework

Let us review the Lazy Snapping framework [2] which creates the foundation for our modified algorithm in chapter 4.

Based on the energy minimization method provided by Yuri Boykov and Marie-Pierre Jolly [1] in 2001, the authors of Lazy Snapping paper had a new idea that uses watershed algorithm [14] to reduce the complexity before apply graph cut labeling solution. In marking step, the user needs to draw some strokes to give samples for object and background. Instead of processing in pixel level, graph cut solution will label all regions in segment level. Even if object marking step processes the boundary as accurately as possible, the errors are still existed, especially in hard cases such as low contrast or high color similarity. So that, boundary editing step will refine the object boundary by using Direct vertex editing or Overriding brush tools.

There are several ways to select objects from the background other than lazy snapping, such as GrabCut [3] and Drap-and-Drop pasting [4]. In the same year 2004, Carsten Rother and others introduced GrabCut method. Unlike Lazy Snapping, the user provides the samples by using a bounding box (rectangle) or a lasso (polygon) to mark the region-of-interest. GrabCut is also based on energy minimization method like Lazy Snapping, but it is not good enough to specify details for objects with arms or external details.

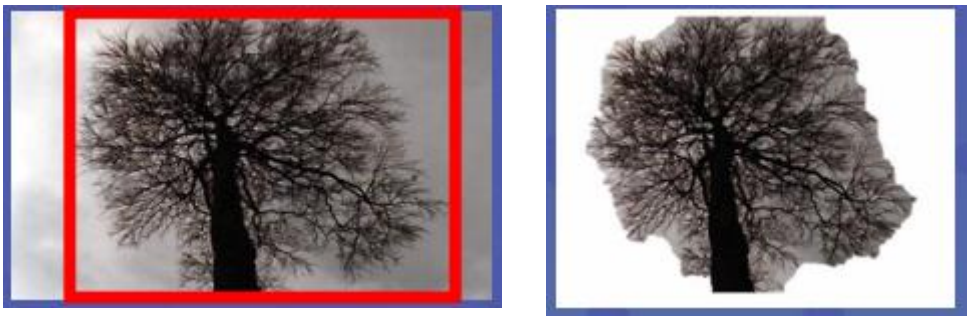


Figure 2. Result of GrabCut method (from [3])

In User Study session of [3], C. Rother shows some comparison of GrabCut method and Lazy Snapping method:

- Easy to use: Lazy Snapping is 20% fewer mistakes
- Speed: Lazy Snapping is 60% less time
- Accuracy: Lazy Snapping is 60% fewer wrong pixels

In 2006, the authors of Drag-and-Drop Pasting created a new objective function to compute an optimized boundary condition that based on Poisson image editing [Perez et al. 2003]. User only need to draw an outline of the object, then drag and drop it onto the output image. The result of this method depends on the way user draws the boundary because Poisson image editing may not always feedback good results. Therefore, with objects with external details, it is quite hard for the user to specify the boundary carefully. This method is a combination of GrabCut and image matting methods.

Figure 3. User define boundary and Drag-and-Drop Pasting results (from [4])

The improvement of Lazy Snapping method is pre-segmentation step – to pre-compute the source image by using watershed segmentation [14] to reduce the number of vertices in graph cut labelling solution, by that, reduce the run time of framework efficiently.

In 1991, Vincent and Soille presented Random access memories and waiting queue driven algorithms to procedure histogram of the image with purpose merge pixel with the same color similarity to a *super pixel*. It was an innovation that achieves new speed in digital image processing that time. However, the number of super pixels is big lead to the over-segmentation problem, so to get better segmentation is still an issue in image processing.

As presenting in [16], the watershed method has a long definitive history that linked with the technological development of the image processing devices. The first watershed transform started in the mid-seventies, and up to now, there are more and more approaches to solving image segmentation problem based on that starting point.

In 1992, Meyer presented different approach named *Watershed transformation* which is implemented in the openCV framework. It is also called *marker-based watershed* algorithm. But this approach gives us over-segmentation result because of noise in the image. The result of this algorithm existed regions with label 0 – not sure which region they belong to and the boundary of each region with label -1. So we have to duel with noises after apply graph cut labelling solution.

Up to now, 20 years later, there are many alternatives choices of the pre-segmentation algorithm. In 2000, K-means method [13] is a simple way to separate the image into small segments. However, this idea may not be a good way to apply to graph cut labelling solution because the information of color is not enough to give good segments, especially in cases the object and background have the similar color.

The SEEDS – Superpixels Extracted via Energy-Driven Sampling [7] is a giant step to achieve a new level of pre-segmentation that based on a simple hill-climbing optimization.

This method can handle over-segmentation problem by controlling the number of super pixels. based on that, the user can test and define how detail of segment needed for next step. The running time of this method depends on number of iterations, number of history bin and how smooth the boundary is. It can be setting by users to get necessary result. The weakness of this method is the border of each segment. It will be a great combination with boundary editing step of Lazy Snapping to have new object selection framework, but how can we get better a boundary from pre-segmentation step so that we do not need to refine it?

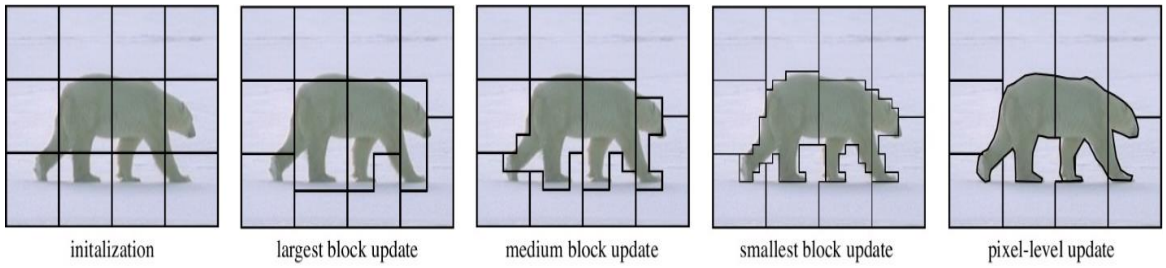


Figure 4. Process of SEEDS method (from [27])

In 2014, Stutz from RWTH Aachen University, Aachen, Germany defended his thesis for Bachelor Degree with Superpixel Segmentation using Depth Information. His research and experiments are the answer to the previous question. The method combines the advantages of SEEDS method and solves the existing problem about unsmooth boundary in the same case. For more detail, I will present in chapter 4.

The important graph-labelling method in Lazy Snapping framework was Graph Cut [11]. This new algorithm is presented by Yuri Boykov and Vladimir Kolmogorov in 2004. In most examples, this method worked 2-5 times faster than any other methods, including outperformace in typical vision graphs such as push-relabel or Dinic's algorithms.

In 2009, Frank R. Schmidt et al introduced *Planar Graph Cut* algorithm in Computer Vision. With this method, we can segment a given image of N pixels in $O(N\log(N))$ which is incredible complexity for image segmentation problem.

In our new method, we will adopt the Lazy Snapping framework but we will modify the pre-segmentation step in order to achieve fast segmentation with few modifications in the boundary by using the overriding brush. In fact, we do not need to use the refining step in lazy snapping to achieve the same result as their paper.

2.2 On max-flow and min-cut problem

As described in chapter 1, the image segmentation problem has two main approaches: region-based and boundary-based. However, both implementation focuses mainly on region-based approach. They are the applications of min-cut max-flow in computer graphics.

The Ford-Fulkerson method was described by L. R. Ford and D. R. Fulkerson on Canadian Journal of Mathematics in 1956 [9]. The method idea is finding shortest paths from source to sink that doesn't reach maximum capacity (called "augmenting path") and saturate the flow through that path until all augmenting paths are exhausted. In 1970, Yefim Dinitz published an algorithm that uses bread-first search to find augmenting path [14].

In 1988, Andrew V. Goldberg and Robert Tarjan described push-relabel algorithm [17]. The "pre-flow" is gradually converted to the maximum by moving flow between neighbor nodes by push operation in admissible network maintained by relabel operation.

In 2001, Boykov and Kolmogorov proposed a new max-flow algorithm [11] also based on finding augmenting path and comparing with some old algorithms in term of energy minimization in vision. Their main improvements are building two search trees and reusing these trees to increase actual performance.

Those papers are mainly written to how to solve the graph cut. In computer graphics, the remaining problem is how to construct the graph. In equivalent, they don't describe the objective energy function that deprived in some ways from image for minimizing in graph cut.
In 2001, Boykov and Jolly published the interactive graph cut [1]. It is a particular approach to the later solution. They provide a way for converting image segmentation to graph cut. In the detail level of pixels, the energy function is the balance of region and boundary properties that satisfies hard constraints provided by the user. The max-flow algorithm runs in constructed network and provides the result of labeling.

In 2004, Li, Tang, Sun and Shum described the lazy snapping that is the super-pixel graph cut with boundary editing [2]. Instead of detail level of pixels, the pixels were group into several segments by the watershed algorithm. Each of them is considered as a super-pixel to construct similar energy function. The boundary after running max-flow can be edited.

In the same year, Rother, Kolmogorov, and Blake also described the GrabCut [3]. That is an iterated graph cut involving both color and contrast information, with border matting for optimizing the boundary. From the inputted bounding box of the object, GMM estimates the color distribution of the target object. It is used to construct the Markov random fields with energy function of connected regions having the same label. Then running the graph cut based optimization to get their values. The procedure is repeated until convergence. Further additional user interaction points out misclassified regions and re-run the optimization. The preserve edges are corrected by border matting.

In 2008, S. Vicente, V. Kolmogorov, and C. Rother published the DijsktraGC [5]. This technique improves object segment by imposing the connectivity prior. After segmenting by graph cut, the user to input some parts belongs to object but are misclassified (thin details in this case). The Dijkstra algorithm as a heuristic way is used to extract these, produces a better result.

In 2009, Frank R. Schmidt, Eno Toppe and Daniel Cremers described a new algorithm [6] to solve minimum cut on planar graphs faster by always augmenting leftmost of all paths from source to sink.

3 Interactive Graph Cut

3.1 Problem statement

Technically, the object segmentation is a binary labelling problem that assigns the label of each pixel on image is either "object" or "background". However, the computer is not capable of recognizing which "object" user want in the image. In our case, user has to label some pixel, called "seeds", so that the technique can separate image in the interactive way.

3.2 Constructing the graph from image segmentation

In Boykov and Jolly' paper, the interactive object segmentation problem can be solved by using min-cut max-flow theorem to find global minimum of cost function among all segmentation that satisfy hard constraints imposed by user.

3.2.1 The min-cut max-flow theorem on network flow

A graph is set $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges connect these vertices.

A directed graph is a graph contains directed edges, that means edges (p, q) is different from (q, p).

A flow network is directed graph that satisfy:

- Each edge has a capacity and receives a flow. The flow on each edge doesn't exceed the capacity.
- The total flow into a vertex is equal to the total flow out of it, except the sink (has only ongoing flow) and source (has only outgoing flow).

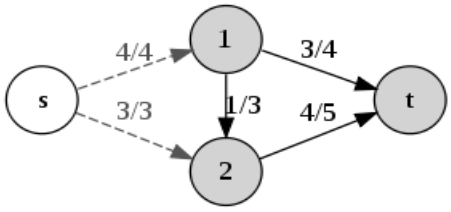


Figure 5. A network with the value of flow equals to the capacity of s-t cut

The min-cut max-flow duality theorem states that in a flow network, the maximum amount of flow from source to sink is also the minimum capacity, when removed in a specific way, such that no flow can pass from source to sink. The solution for min-cut problem is also the solution for max-flow problem and otherwise.

This theorem has been proof by Ford and Fulkerson in 1956 [9]. They also provide a method for solving the max-flow problem in 1962 [10]. It is described more in next section about new algorithm from Yuri Boykov and Vladimir Kolmogorov in 2004 [11].

3.2.2 Another max-flow algorithm

As mentioned, the object segmentation is interpreted as min-cut max-flow problem. Some solution, such as push-relabel algorithm and Ford-Fulkerson method, had been proposed.

- Ford-Fulkerson method's main idea is sending flow through all *augmenting paths* from source to sink. The augmenting path concept is the path from source to sink that have available capacity on all edges of the path (e.g. not max flow). However, they didn't provide a specific way to find augmenting path. In this thesis, the Dinic's algorithm is used to demonstrate Ford-Fulkerson method [9]. It uses the breadth-first search on residual graph to find the shortest path along non-saturated edges.

- Push-relabel algorithm, also called preflow-push algorithm, initializes a preflow and convert it to maximum flow by *push* operations (moving flow between neighbor vertices) and *relabel* operations (maintaining admissible network). The generic algorithm was also called Goldberg-Tarjan algorithm [17].

In their experiments in paper [1], Yuri Boykov and Jolly used the algorithm proposed by Yuri Boykov and Vladimir Kolmogorov. It's based on augmenting path. Their main ideas are:

- Building two search trees, one from source and the other from sink
- Reusing these trees and never start building trees from scratch

The algorithm is iterative of three stages until no search tree can grow anymore:

- Growth stage (Figure 6): the two search is grow until they are touch that a path from source to sink is found.
- Augmentation stage: the found path is augmented and the search trees break into forest.
- Adoption stage: the two search tree are restored.

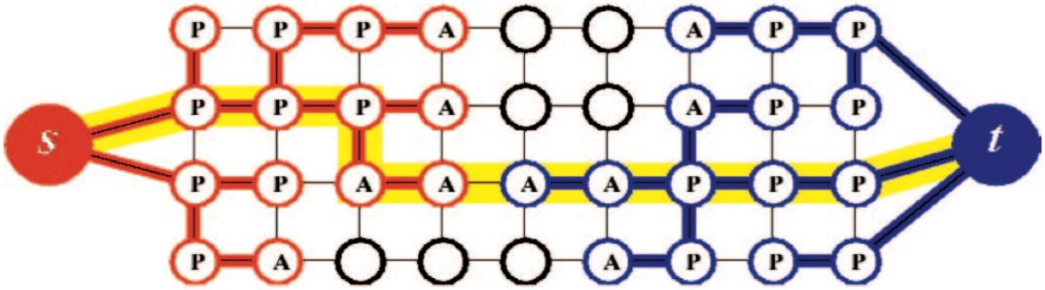


Figure 6. S and T tree at the end of growth stage (from [5])

3.2.3 Object segmentation to graph cut

As described above, the object segmentation can be solved by using min-cut max-flow theorem. To convert the binary labeling to graph (Figure 7), Boykov and Jolly consider:

- The two destination labels Object ($\mathcal{O}$) and Background ($\mathcal{B}$) of object segmentation correspondingly as the source (S) and sink (T) in the flow network.
- All pixels in image are internal nodes.
- Each internal node has exactly one flow from source and one flow to sink. It describes how the corresponding pixel associate to object and background class. These flows are called t-links.
- The other flows connect nodes together describe how adjacent pixel related. These flow are called n-links. Since Boykov and Jolly use 4-neighbor of P, each internal node has at most 4 n-links.

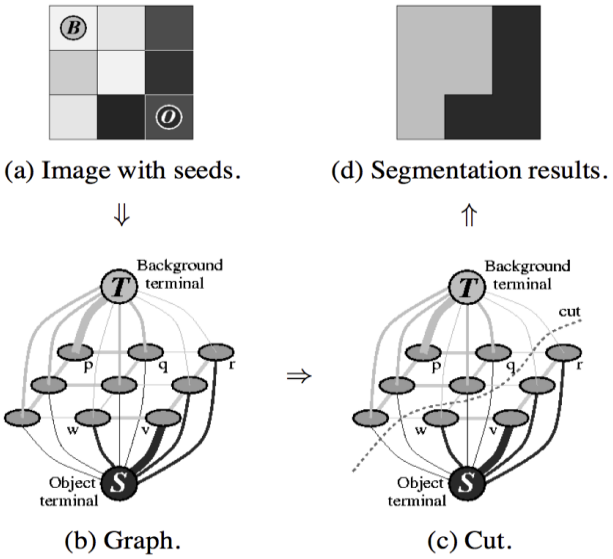


Figure 7. Image segmentation as a graph cut (from [1])

The labeling problem now is finding a proper cut through the graph. Each internal node has exactly one t-link join the cut. That also means it "cuts" the relationship of corresponding pixel to the class of another side of that t-link and belongs to the remaining one: t-link from source joins cut, the pixel is background and otherwise.

The graph cut separates vertices into two disjoint sets. Correspondingly, the image is separated into two segments. The minimum cost cut creates optimal segmentation, which is based on properties. This cost is built into edge weights.

3.2.4 Objective function

From section 2.3, we can see the object segmentation as a graph cut.

Boykov and Jolly define feasible cut:

- Each internal node serves exact one t-link
- Each edge belongs to feasible cut if and only if two connected nodes are t-linked different terminal
- Seed nodes belong to feasible cut

In equivalent, the feature cut is respectively the segmentation in the image.
Let A is a $\{\text{"object", "background"}\}$ binary vector of pixels that defines a segmentation. Boykov and Jolly see the soft constraints imposed boundary and region properties of A as an energy cost function $E(A)$:

$$E(A) = \lambda \cdot R(A) + B(A) \ ,$$

where

$$R(A) = \sum_{p \in P} R_p\left(A_p\right)$$

$$B(A) = \sum_{\{p,q\} \in \mathcal{N}} B_{\{p,q\}} \cdot \delta\left(A_p, A_q\right)$$

and

$$\delta\left(A_p, A_q\right) = \begin{cases} 1 \text{ if } A_p \neq A_q \\ 0 \text{ otherwise.} \end{cases}$$

- The regional term $R(A)$ assumes the penalties of assigning each pixel p to "object" or "background", respectively $R_p(\text{"object"})$ or $R_p(\text{"background"})$, are given. $R_p(\cdot)$ may just the intensity of pixel p fit intensity model of the object or background.
- $B(A)$ contains the boundary properties of A. Each coefficient $B_{\{p,q\}}$ is discontinuity penalty between two pixel p and q. The larger $B_{\{p,q\}}$ are, the more similar between p and q, otherwise when it approach zero.
- The $\lambda \geq 0$ term is the relative importance of region properties term $R(A)$ over the boundary term $B(A)$.

Meanwhile, there are some initial background and object pixel, called "seeds", defined by user. The boundary segmentation must separate background seeds and object seeds. Boykov and Jolly define these hard constraints be the combining of region and boundary properties of segments. The initial weight of edges in $\mathcal{E}$ is described as following table

Table 2. Weight of three types of edge

edge	weight (cost)	for
$\{p,q\}$	$B_{\{p,q\}}$	$\{p,q\} \in \mathcal{N}$
$\{p,S\}$	$\lambda \cdot R_p(\text{"background"})$	$p \in \mathcal{P}, p \notin \mathcal{O} \cup \mathcal{B}$
	K	$p \in \mathcal{O}$
	0	$p \in \mathcal{B}$
$\{p,T\}$	$\lambda \cdot R_p(\text{"object"})$	$p \in \mathcal{P}, p \notin \mathcal{O} \cup \mathcal{B}$
	0	$p \in \mathcal{O}$
	K	$p \in \mathcal{B}$

where

$$K = 1 + \max_{p \in \mathcal{P}} \sum_{q:\{p,q\} \in \mathcal{N}} B_{\{p,q\}}.$$

When the user interactively adds or removes the "object" or "background" seeds, the weight of corresponding edges of those seeds is also changed. However, if the weight is changed according to the previous table, we can't compute from initial flow due to reduction of some edge capacities. But if both t-links are increase a constant, it's not a problem anymore. For example, user adds a new "object" seeds on pixel p, the new weight can be described as below

Table 3. Cost updating for reusing

t-link	initial cost	add	new cost
$\{p,S\}$	$\lambda \cdot R_p(\text{"background"})$	$K + \lambda \cdot R_p(\text{"object"})$	$K + c_p$
$\{p,T\}$	$\lambda \cdot R_p(\text{"object"})$	$\lambda \cdot R_p(\text{"background"})$	c_p

Boykov and Jolly has proved that:

- The minimum cut is a feasible cut

- The minimization of this cost function corresponds to the resulting labelling of the min-cut on the graph. The solution of max-flow min-cut is also the solution of labelling A that minimize energy function $E(A)$.

So, the object segmentation problem is solving the max-flow problem on the converted flow network. We use the Boykov-Kolmogorov algorithm (described in previous section) instead of standard algorithms.

3.3 Experimental Results

In our implementation, the seeds are visualized as green "object" and blue "background". The seeds have double uses: the hard constraints and the histogram of "object" and "background" intensity distribution. The regional penalties can be negative log-likelihood

$$R_p(\text{"object"}) = -\ln \Pr(I_p|\mathcal{O})$$

$$R_p(\text{"background"}) = -\ln \Pr(I_p|\mathcal{B})$$

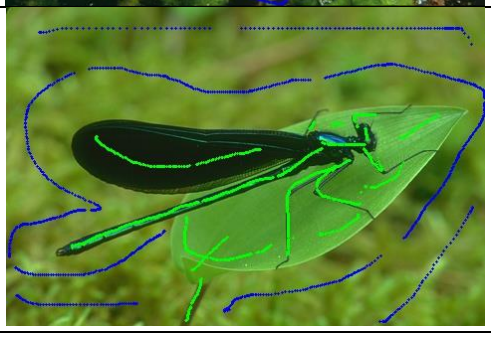
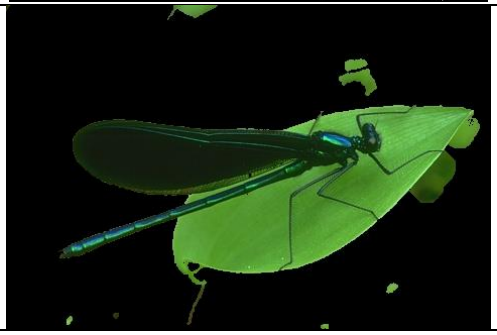
We use 8-neighbor for adjacency edges. The boundary penalties are the ad-hoc function with the intensity and pixel-to-pixel distance use Euclidean distance.

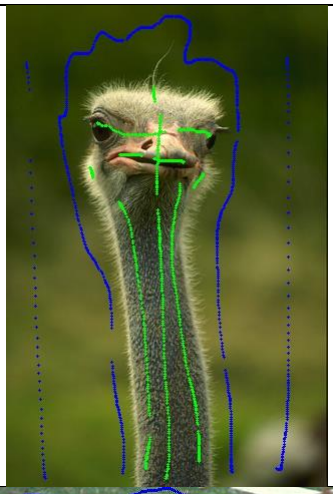
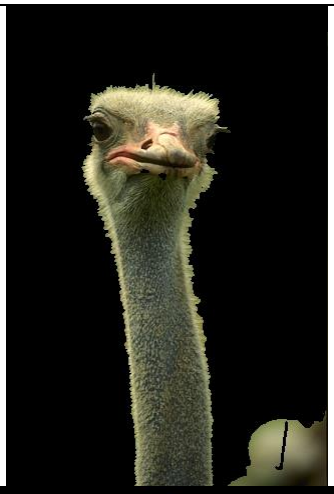
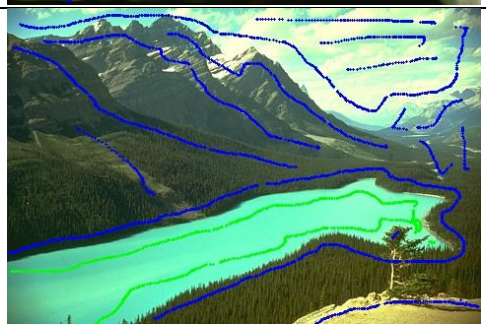
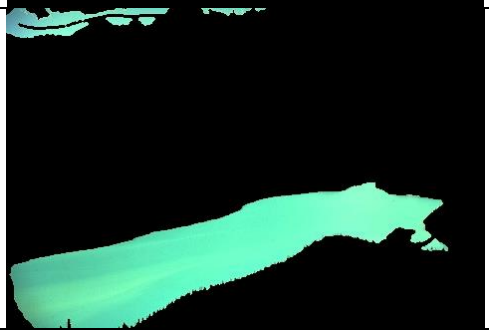
$$B_{\{p,q\}} \propto \exp\left(-\frac{(I_p - I_q)^2}{2\sigma^2}\right) \cdot \frac{1}{dist(p,q)}.$$

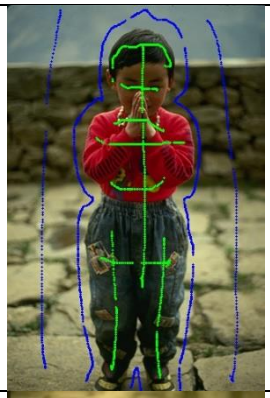
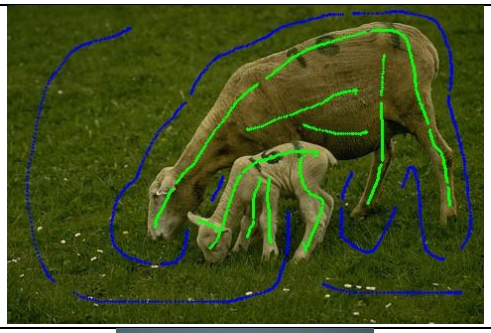
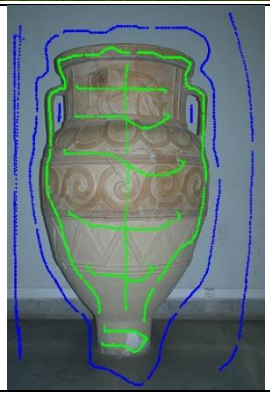
Note that the experiment run on Intel Core i5 2.2Ghz. Our implementation use max-flow algorithm described in section 3.2.

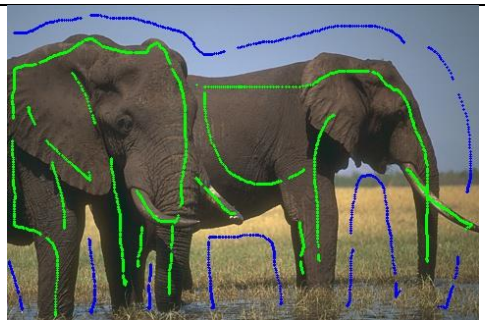
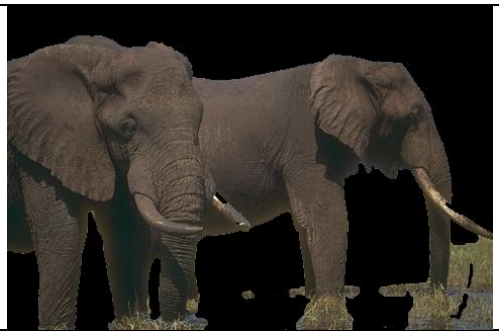
3.3.1 Berkeley segmentation dataset

Table 4. Experiments on Berkeley dataset

Image	Image with seeds	Final segmentation
12003		
35070		

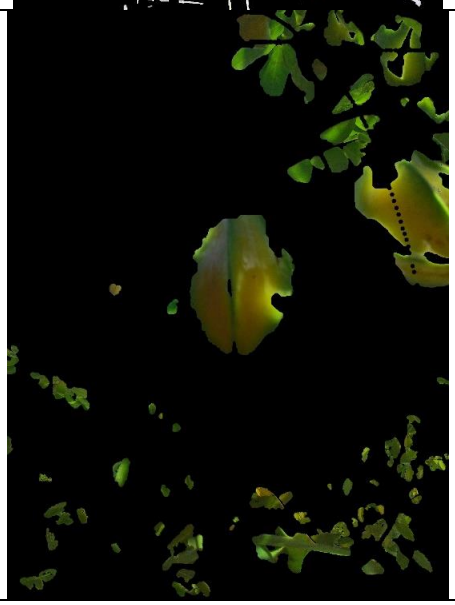
66075		
85048		
124084		
176035		

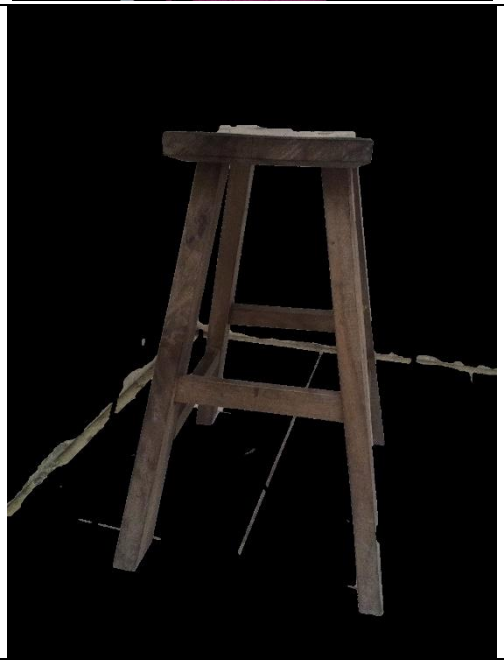
187029		
187083		
189011		
207056		
227092		

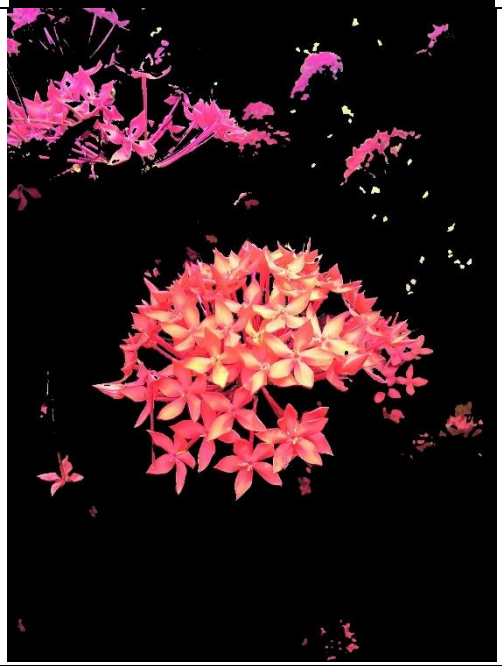
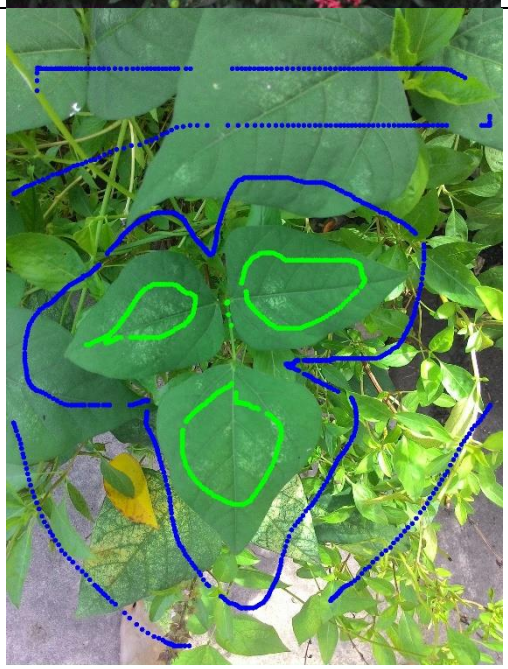
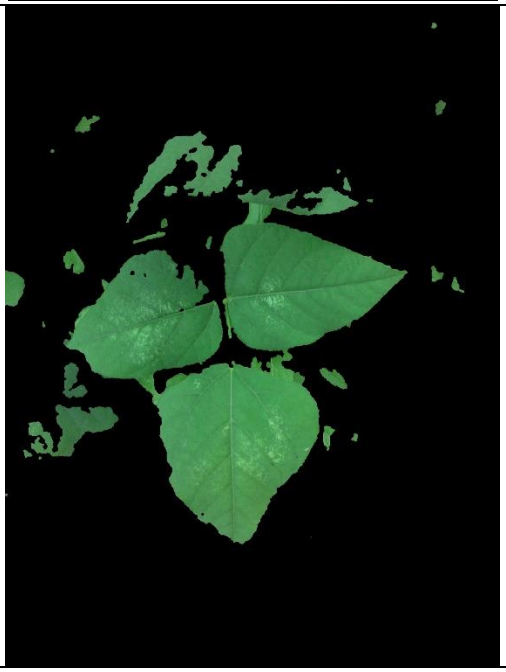
296059		

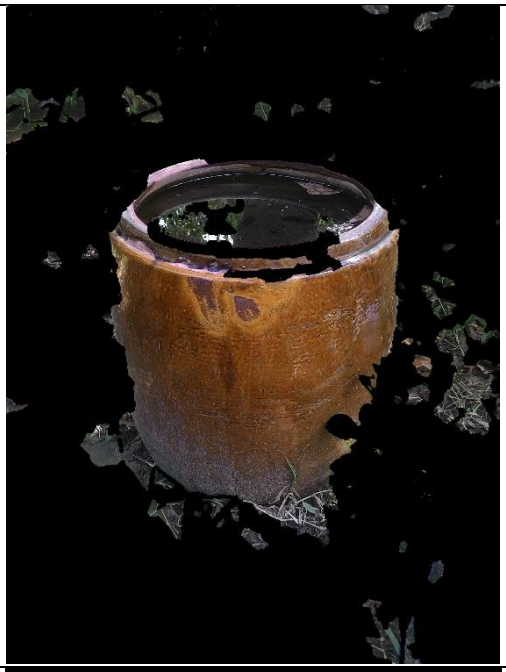
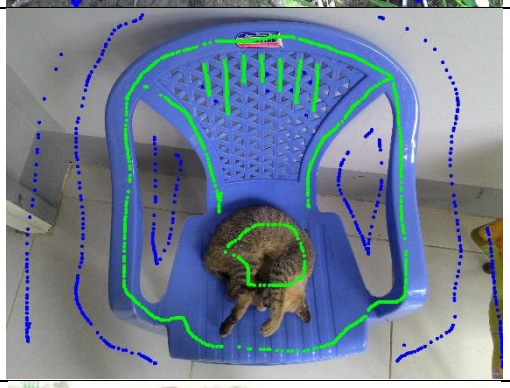
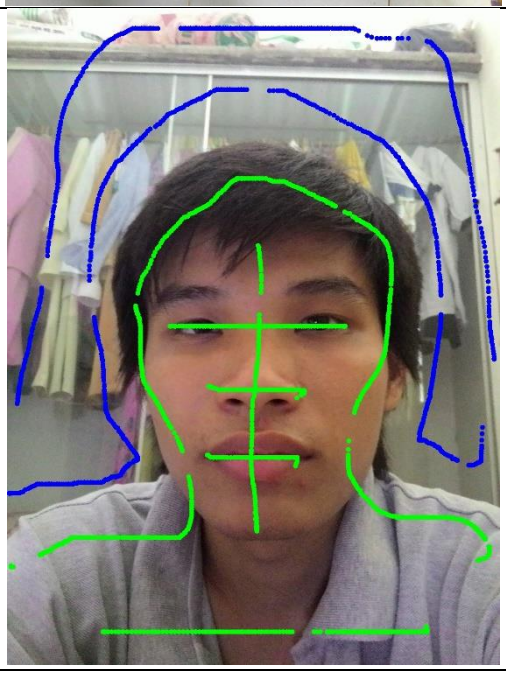
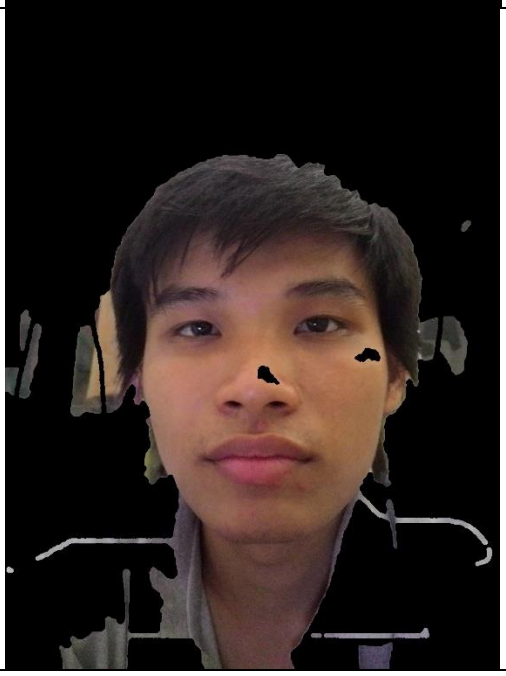
3.3.2 Our dataset

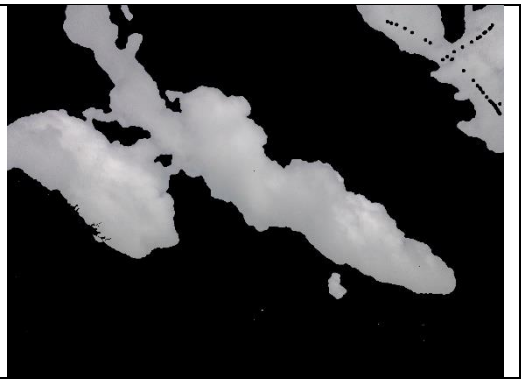
Table 5. Experiments on our dataset

Name	Image with seeds	Final segmentation
1		
2		

3		
4		
5		

6		
7		
8		

9		
10		
11		

12		

3.3.3 Discussion

The interactive graph cut has some goods and limitations. In this sections, we will discuss more about limitations.

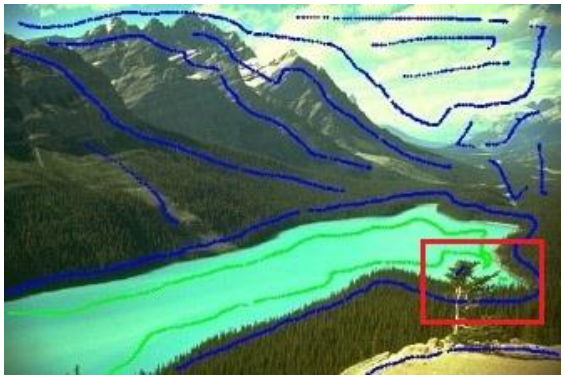
3.3.3.1 Performance

The most serious problem of interactive graph cut is its performance. For any image, it considers each pixel as a node and its relation to neighbours as edges. That make the graph size huge. For a 2M image, the graph has nearly 2 million nodes and 20 million edges. It surely takes a lot of time to compute the max-flow and consumes tremendous resources. This method is not a good solution for high resolution image which become more and more popular.

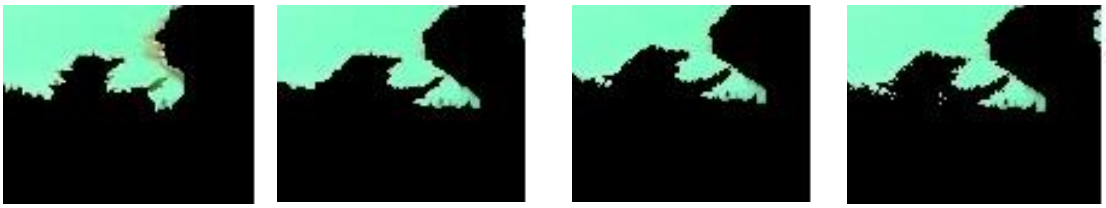
3.3.3.2 The effects of lambda

Figure 8 shows the result of segmentation of various lambda. In b, only boundary properties are used, only pixels that “close” (by both intensity difference and Euclidean distance meaning followed the ad-hoc function) to object seeds are shrunk. When region term has more affect (c-e), the color histogram makes the color distribution factor become more and more important over the

Euclidean distance, and results in not only the boundary more detailed, but also some outside pixels labelled wrongly.



a. Image with seeds and selected red regions for zooming in in results



b. $\lambda = 0$ *c.* $\lambda = 0.5$ *d.* $\lambda = 1$ *e.* $\lambda > 1$

Figure 8. Results of graph cut segmentation with various lambda

The trouble is the "right" value of lambda can't be automatically computed. The user should manually set until satisfied.

3.4 Conclusion

This project is a simple implementation of the interactive graph cut. It still has many limits. The pixel level of accuracy makes the computation process slow and resource consumption high for high resolution image. Moreover, the balance of boundary and region terms is not easily archived to get satisfied result. However, it is a good base for later solution in term of conversion between image components and graph cut, also the energy function construction. The Lazy Snapping in next chapter is a good instance for such solutions.

4 Lazy Snapping

4.1 Problem statement

Interactive single-image segmentation (or image cutout) is one of the popular problems in Computer Vision. A user typically provides samples of foreground (the part that you want to cut out) and background by drawing some strokes in each parts or a coarse bounding box of the object in the image. Then the algorithm bases on the information with the added hard constraints for some pixels to separate the object from the alternative background. By the time, there would have had many improvement and new method to solve the problem. Yuri Boykov and Marie-Pierre Jolly [1] provided a method using energy minimization to solve the problem and up to now, that is the heart of most recent technique.

Computer is still long way to recognize objects in picture, so that, the user has to give samples of each region with high accuracy. In common, a user gives the sample data for both foreground and background that defined as foreground seeds *F* and background seeds *B*. All the another pixel that is not marked defined as "uncertain region" *U* in the image, as show in Figure 9.

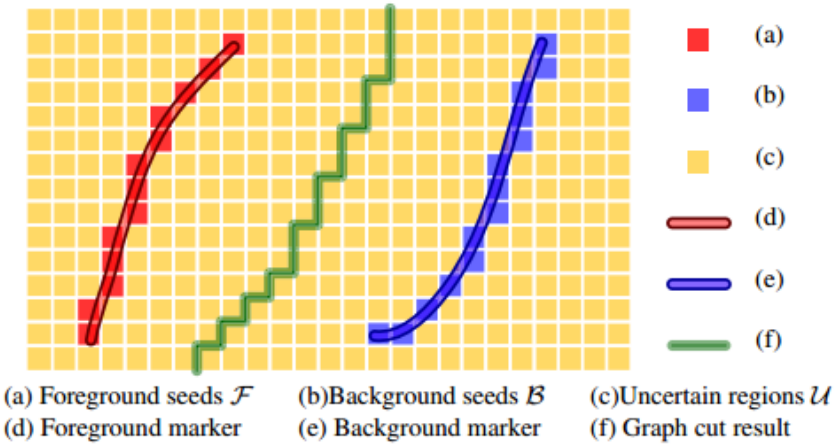


Figure 9. Sample pixels of background/foreground and uncertain region from paper [2]

To be more formal:

Suppose we have an input image $\mathcal{N}$ and some sets of pixels:

- Foreground Set $\mathcal{F} = \{f_1, f_2, f_3, ...\}$: set of all pixels that are sample for the object.
- Background Set $\mathcal{B} = \{b_1, b_2, b_3, ...\}$: set of all pixels that are sample for the background.
- Uncertain Set $\mathcal{U} = \mathcal{N} - (\mathcal{F} \cup \mathcal{B})$: set of all pixels that are not in the both $\mathcal{F}$ and $\mathcal{B}$.

How to figure out all the pixels belong to the object in the image and separate them from the background?

4.2 Casting the problem as binary labelling problem

4.2.1 Binary images: foreground, background and connectivity

A binary image is an image in which each pixel belongs to one of only two possible discrete, logical value: 1 or 0. In our problem, the logical value 1 is known as 'object', whilst the logical value 0 is described as 'not object'. Base on the logical value in the binary image, we define the set of pixels which the logical value is 1/0 as the temp *foreground* pixel/*background* pixel. An *object* is created by groups of *connected* pixels.
There are 2 common definition of connection between pixels in the image:

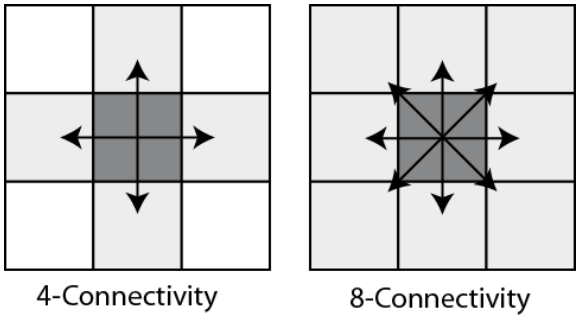


Figure 10. 4-connected pixel and 8-connected pixel examples from [22]

- If we required that a given foreground pixel must have at least on neighboring foreground pixel to 4 directions: north, south, east or west of itself to analyze them as parts of the object, then we are using 4-connection.
- However, in case we need 4 another direction: north-east, north-west, south-east or south-west to analyze, then we are using 8-connection.

4.2.2 Formulation, constraint and energy function

Based on the approach present in [24], we can formulate our image cutout problem as a binary labelling problem:

- For all image pixels, we need to assign a unique label $x_i \in \{ foreground\,(=1), background\,(=0) \}$
- If a pixel is in $\mathcal{F}$ / $\mathcal{B}$, the label of it is foreground/background.
- If a pixel is not in $\mathcal{F}$ and $\mathcal{B}$, the temporary label of it 'uncertain'.

How to find a way to label all pixels in the image so that all the pixels belong to the object we want to select have same label, and so do to all the another pixel?

We define the constraints:

Data constraint: If a pixel has its colour similar to at least one pixel belongs to foreground seeds *F*, it is more likely to get foreground label, and so do with background seeds *B*.

Prior constraint: Pixels that belong to the object tend to group together, and pixels that belong to background tend to group together.

To find a labelling that satisfy our constraints, we have to formulate and minimize an energy function of labelling L:

Let L_p be the label value of pixel p, so that $L_p \in \{0,1\}$.

Firstly, we model the data constraint by using Data Penalty Function $D_p(L_p)$ that is defined as follows:

$$D_p(x_i = 1) = 0 \quad D_p(x_i = 0) = \infty \quad \forall x_i \in \mathcal{F}$$
$$D_p(x_i = 1) = \infty \quad D_p(x_i = 0) = 0 \quad \forall x_i \in \mathcal{B}$$

With the pixels that belong to 'uncertain' region, we follow some steps such as [2]:

1. Use K-means method to cluster the color of foreground seeds $\mathcal{F}$ and background seeds $\mathcal{B}$. (In this experiment, we try with K = 64)
   The mean colors of each set are denote as $\{M_K^F\}$ and $\{M_K^B\}$
2. Suppose each node i has its color $C(i)$, we compute the minimum distance from its color to all clusters of each set, defined as d_i^F and d_i^B:

$$d_i^F = \min \left\| C(i) - M_n^F \right\|$$
$$d_i^B = \min \left\| C(i) - M_n^B \right\|$$

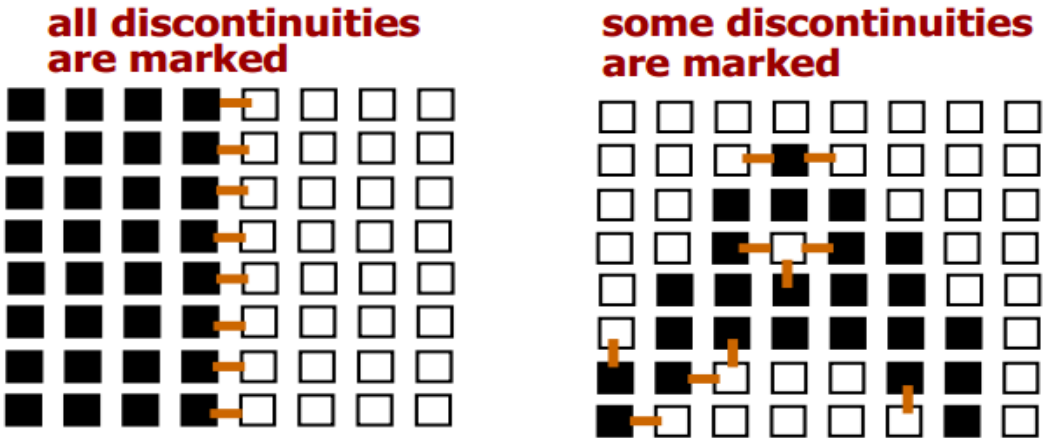
Therefore, we complete the data constraint with the following table after model the constraint for 'uncertain' region

$$D_p(x_i = 1) = \frac{d_i^F}{d_i^F + d_i^B} \quad D_p(x_i = 0) = \frac{d_i^B}{d_i^F + d_i^B} \quad \forall x_i \in \mathcal{U}$$

To figure out how image satisfy the data constraint, we sum up all pixels' penalty function:

$$\sum_p D_p(L_p)$$

Before modeling the prior constraint, we will see two examples about 'discontinuity' in labelling L in following figure.



a. 8 discontinuities *b. 31 discontinuities*

Figure 11. Example about discontinuities in binary image (from [25])

Pixel p and q exist a discontinuity if $L(p) \neq L(q)$. By that, each pixel belongs to the object boundary have some discontinuities pixels around. We model the prior constraint as a function of the color gradient for all discontinuities:

$$\sum_{p,q \in N} |L_P - L_q| * g\left(C_{i,j}\right)$$

Where:

- $g(\xi) = \dfrac{1}{1+\xi}$
- $C_{p,q} = \left\| C_p - C_q \right\|^2$ is the L2-Norm of RGB color difference between two pixels p and q.

More detailed, we use Euclidean distance to compute $C_{p,q}$. The more similar the colors or two neighbor pixels are, the less likely belong to different categories.

Finally, we have our final energy equation, which need to satisfy both data constraint and prior constraint:

$$E(L) = \sum_p D_p\left(L_p\right) + \lambda * \sum_{p,q \in N} |L_P - L_q| * g\left(C_{i,j}\right)$$

Where λ know as a balancing parameter between the data constraint and the prior constraint, from [24], here is the conclusion:

- The larger the λ, the less discontinuities in the optimal labelling L.
- The smaller the λ, the more the optimal labelling L.

4.3 Max flow/min cut problem

This section is short summarize that taken from [21].

4.3.1 Network

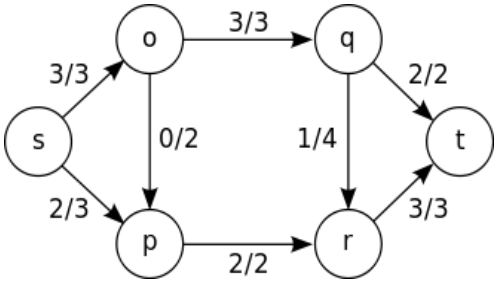


Figure 12. A network with an example of maximum flow from source s to sink t (from [26])

Network is a directed graph $G = (V, E)$ that have n vertices and m edges, include 2 separate vertices are "s" and "t", where s is the source and t is the sink of the network. With each edge $e = (u, v) \in E$ is assigned a non-negative number $c(e) = c[u, v]$ that is the capacity of that edge.

4.3.2 The concept of flow in network

Suppose we have network $G = (V, E)$. We have *flow* φ in network G is an operation that assign $e = (u, v)$ with a non-negative number $\varphi(e) = \varphi[u, v]$ that call a flow in the edge e, satisfy 3 constraints:

- ***Capacity Constraint***: Flow in each edge have not to over than the capacity of that edge: $0 \le \varphi[u, v] \le c[u, v]$ $(\forall u, v \in V)$
- ***Skew symmetry***: $\forall u, v \in V$, flow in edge (u, v) and flow in edge (v, u) have the same value but opposite sign: $f[u, v] = -f[v, u]$
- ***Flow Conservation***: Every vertices v that is not the source and the sink, sum of all in-flow that go to v equal to sum of all out-flow that go out v:

$$\sum_{u \in V} \varphi[u, v] = \sum_{w \in V} \varphi[v, w] \ (\forall v \in V \setminus \{s, t\})$$

The 'value' of a flow is defined as (sum of the edge that go out from the source) minus (sum of the edge that go in the source):

$$|\varphi| = \sum_{u \in V} \varphi[s, u] - \sum_{w \in V} \varphi[v, s]$$

Because the value of the flow φ in each edge is non-negative, we can also call φ is positive flow in the network G.

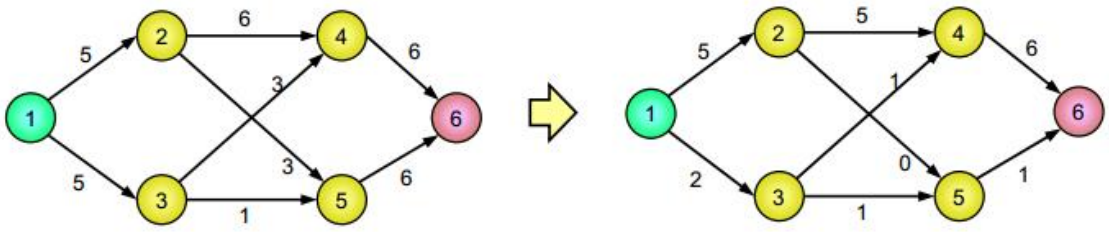


Figure 13. A sample of network with the capacity of each edge (1 is source, 6 is sink) and a flow of it with value is 7 (from [21]).

4.3.3 Max flow problem

Suppose we have a network G, find flow φ that have maximum value?

Without loss of generality, in the max flow problem, we can suppose that with every flow φ, flow in edge (u,v) and flow in edge (v,u) are simultaneously positive $(\forall u,v \in V)$. Because if we just decrease both $\varphi[u,v]$ and $\varphi[v,u]$ by a number that equal with $\min(\varphi[u,v],\varphi[v,u])$, we will get a new flow which have the same value with original flow, include $\varphi[u,v]=0$ or $\varphi[v,u]=0$.

4.3.4 Basic properties

We have X and Y are two subset of vertices set V, we call the capacity from X to Y is:

$$c(X,Y)=\sum_{u\in X,v\in Y} c[u,v]$$

and the value of flow from X to Y is:

$$f(X,Y)=\sum_{u\in X,v\in Y} f[u,v]$$

Theorem 1:

(1) $\forall X \subseteq V$, we have $f(X,X)=0$

(2) $\forall X,Y \subseteq V$, we have $f(X,Y)=-f(Y,X)$

(3) $\forall X,Y,Z \subseteq V, X\cap Y=\phi$, we have $f(X,Z)+f(Y,Z)=f(X\cup Y,Z)$

(4) $\forall X \subseteq V\setminus\{s,t\}$, we have $f(X,V)=0$

Proof:

- $\forall X \subseteq V$, we have $f(X,X)=\sum_{u,v\in X} f[u,v]$, both the value of $f[u,v]$ and $f[v,u]$ appear in the flow value. We also have $f[u,v]=-f[v,u]$ (skew symmetric constraint), so $f(X,X)=0$
- $\forall X,Y \subseteq V$, we have $f(X,Y)=\sum_{u\in X,v\in Y} f[u,v]$ and $f(Y,X)=\sum_{v\in Y,u\in X} f[v,u]$. $f[u,v]$ appears in $f(X,Y)$ and $f[v,u]$ appears in $f(Y,X)$, follow skew symmetric, we have $f(X,Y)=f(Y,X)$

- $f(X \cup Y, Z) = \sum_{u \in X \cup Y, v \in Z} f[u,v] = \sum_{u \in X, v \in Z} f[u,v] + \sum_{u \in Y, v \in Z} f[u,v] = f(X,Z) + f(Y,Z)$
- $\forall u \subseteq V \setminus \{s,t\}$, we have $f(\{u\}, V) = 0$ (flow conservation constraint), so we have $\forall X \subseteq V \setminus \{s,t\}$, then $f(X,V) = 0$ (also $f(V,X) = 0$).

4.3.5 The concepts of residual graph and augmenting path

4.3.5.1 Residual network

From a network G and a flow F in G, we create a network $G_f = (V, E_f)$ with the set of edges E_f is defined as the set of edges E without *fully saturated* edges (an edge is fully saturated if flow in the edge exactly equal to the capacity of that edge):

$$E_f = \{(u,v) | (u,v \in V) \wedge (f[u,v] < c[u,v])\}$$

The capacity of edge (u,v) in G_f is computed by:

$$c_f[u,v] = c[u,v] - f[u,v]$$

with s is the source and t is the sink in G_f.

That is definition of G_f is called residual network of the network G that create by flow f.

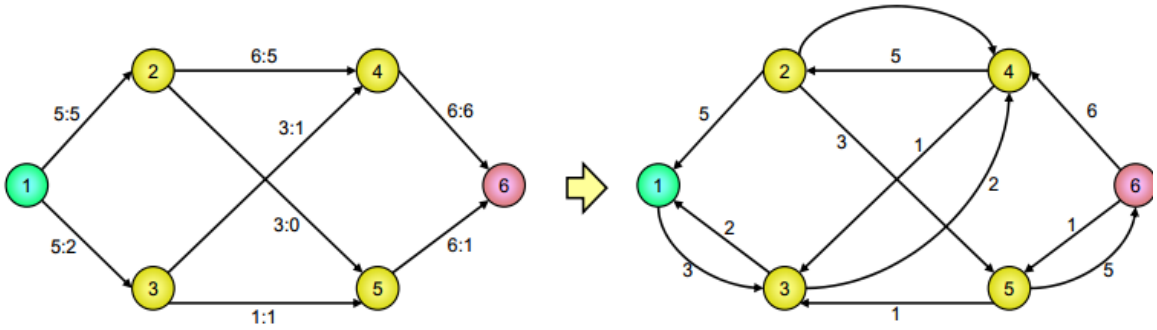


Figure 14. Network G and residual network G_f (in each edge have information about c[u, v]:f[u, v] show the capacity and the flow in edge (u, v) (from [21])

Theorem 2: Suppose we have network G and a flow f. Call f' is a flow in network G_f. Then $(f + f')$ is also a flow in network G with flow value is $|f| + |f'|$. The flow $(f + f')$ is defined as:

$$(f + f')[u,v] = f[u,v] + f'[u,v] \ (\forall u,v \in V)$$

Proof: We prove that flow $(f + f')$ satisfy 3 constraints of flow definition.

- ***Capacity Constraint***: $\forall u,v \in V$.

  Because $f'[u,v] \le c_f[u,v] = c[u,v] - f[u,v]$ so $f[u,v] + f'[u,v] \le c[u,v]$

- ***Skew symmetry***: $\forall u,v \in V$.

  $(f + f')[u,v] = f[u,v] + f'[u,v] = -f[v,u] - f'[v,u] = -(f + f')[v,u]$

- ***Flow Conservation***: $\forall u \in V \setminus \{s,t\}$.

$$\sum_{v \in V}(f + f^{'})[u,v] = \sum_{v \in V}(f[u,v] + f^{'}[u,v]) = \sum_{v \in V} f[u,v] + \sum_{v \in V} f^{'}[u,v] = 0$$

About the value of flow, we have:

$$|f + f^{'}| = \sum_{v \in V}(f + f^{'})[s,v] = \sum_{v \in V}(f[s,v] + f^{'}[s,v]) = \sum_{v \in V} f[s,v] + \sum_{v \in V} f^{'}[s,v] = |f| + |f^{'}|$$

Theorem 2 is proved.

4.3.5.2 Augmenting Path

We have a network G and a flow f, a path from A to B in network G_f is called an augmenting path.

With an augmenting path P, we defined $\Delta_P = \min\{c_f(u,v) \mid (u,v) \in P\}$ is minimum value of capacities in each edges of P. Δ_P is called residual capacity of P.

Theorem 3: With P is an augmenting path then the value $f_P[u,v]$ is defined as:

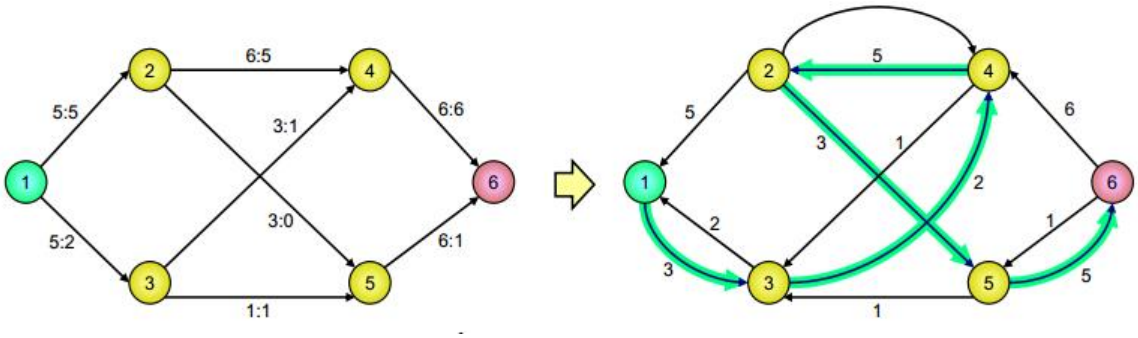


Figure 15. Residual network and an augmenting path (from [21])

$$f_P[u,v] = \begin{cases} \Delta_P, \text{if}(u,v) \in P \\ -\Delta_P, \text{if}(v,u) \in P \\ 0, otherwise \end{cases}$$

is a flow in G_f.

Proof: It is easy to prove f_P satisfy 3 constraints. Theorem 3 hold unconditionally.

Theorem 4: Suppose we have network G and flow f, if P is an augmenting path in residual network G_f then the operation $f = (f + f_P)$ is a new flow in f which have value equal to $|f| + \Delta_P$.

Proof: From theorem 2 and 3, we have theorem 4 is proved.

Operation $f = (f + f_P)$ is called increase flow follow augmenting path P.

4.3.6 Cut in network

We call a cut (X,Y) is a way to separate the vertices set V to 2 different and not null subset X and Y $(X \cap Y = \phi, X \cup Y = V)$. The cut that satisfy $s \in X$ and $t \in Y$ is called s-t cut. Capacity of the cut (X,Y) is defined as:

$$c(X,Y) = \sum_{u \in X, v \in Y} c[u,v]$$

and the flow value of the cut (X,Y) is defined as:

$$f(X,Y) = \sum_{u \in X, v \in Y} f[u,v]$$

Theorem 5: Suppose we have network G and flow f, then the flow pass through any s-t cut have value $|f|$.

Proof: (X,Y) is a s-t cut of the network.

$f(X,Y)$	$= f(X,V) - f(X,V \setminus Y)$	Theorem 1
	$= f(X,V) - f(X,X)$	$V \setminus Y = X$
	$= f(X,V)$	Theorem 1
	$= f(s,V) + f(X \setminus \{s\},V)$	Theorem 1
	$= f(s,V)$	Theorem 1, $t \notin X \setminus \{s\}$
	$= \|f\|$	

Theorem 6: Suppose we have network G and flow f, and (X,Y) is a s-t cut, then the flow pass through cut (X,Y) is not over than the capacity of the cut (X,Y):

$$f(X,Y) \le c(X,Y)$$

Proof: $f(X,Y) = \sum_{u \in X, v \in Y} f[u,v] \le \sum_{u \in X, v \in Y} c[u,v] = c(X,Y)$

Therefore, value of any flow f in network G is not over than the capacity of a s-t cut.

From theorem 2 and 3: $|f| \le c(X,Y)$

Theorem 7: If f is a flow in network $G = (V,E)$ with source s and sink t, then three following clauses are equivalent:

(1) f is maximum flow in G

(2) Residual network G_f does not have any augmenting paths.

(3) Exist a s-t cut (X,Y) with $|f| = c(X,Y)$

Proof:

$(1) \Rightarrow (2)$

We argue by contradiction that G_f has an augmenting path, then we have $(f + f_P)$ is also a flow in network G and the value of new flow $(f + f_P)$ is bigger than f, when f is maximum flow in network G, which mean that contradiction is wrong which proves the required claim.

$(2) \Rightarrow (3)$

If G_f does not have any augmenting paths, we have:

- X = {u | Exist path from s to u in network G_f }
- $Y = V / X$

which $X \cap Y = \phi, s \in X, t \notin X$ (because there is not existed path from s to t), we deduce (X,Y) is a s-t cut. $\forall u \in X$ and $\forall v \in Y$, we have $f[u,v] = c[u,v]$ because if $f[u,v] < c[u,v]$ then exist an edge (u,v) in G_f so we have a path from s to v, opposite to the definition of cut. By that, we have $f(X,Y) = c(X,Y)$

$(3) \Rightarrow (1)$

Follow theorem (3), the value of every flow in network G $|f| \leq c(X,Y)$, so if the value of the flow $|f|$ exactly equal to the capacity of the cut (X,Y), then f have to be maximum flow value in the network.

Definition: s-t cut have smallest capacity (equal to maximum value in the network) is called minimum s-t cut of the network G.

4.3.7 Minimum cut problem

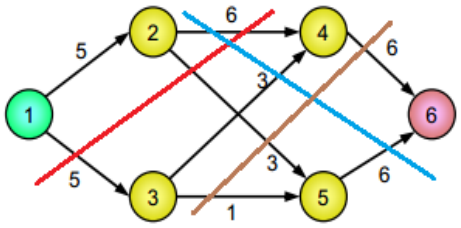


Figure 16. Which is minimum cut in network? (from [21])

Suppose we have network G, find a s-t cut that satisfy sum of all cost in edges set is minimum to disconnect s from t?

4.4 The graph-cut solution to the image cutout problem

Minimum cut problem gave us a new approach that is called ‘graph cut’. If we can represent the cut-out image problem as binary labelling problem, then model the problem to become a network G, the original problem will be solved. This section discusses details of the “**Silent Lazy Snapping**” approach that developed and implemented by Tran Hoang Nam, FPT University, Vietnam.

a. Input image *b. Object marking* *c. Presegmentation* *d. Final segmentation* *e. Output Composition*

Silent Lazy Snapping is an interactive image cutout framework, consisting of two steps: a quick object marking step and a simple segment editing step. In [b], the (blue) lines are drawn to indicate the foreground, and another (red) line to indicate the background. In [c], we apply several method to separate pixels in the input image into super pixels. Based on result of SEEDs Revised Mean Value method [c – image in the bottom], a good final segmentation with accurate boundary can be obtained by simply clicking to non-object super pixels in [d]. Finally, the cut out is composed on another painting in [e].

Figure 17. An example of Silent Lazy Snapping process

4.4.1 Sample define by users

Define action that user gives example to foreground/background. In this experiment, we use two following events:

- Press-button-f-event: allow user to change to foreground mode.
- Press-button-b-event: allow user to change to background mode.
- Left-mouse-click-event: allow user to visualize examples by drawing some strokes in the input image.
    - Every points in the strokes present for foreground will be display by blue point.
    - Every points in the strokes present for background will be display by red point.

Figure 18. Input image (left image) and image with examples given by user (right image)

- Reload-image-event: allow user to reload the input image in case their examples are not exactly what they want to draw.

- More detailed, in this experiment, user can reload image when click ‘r’ button in the keyboard.

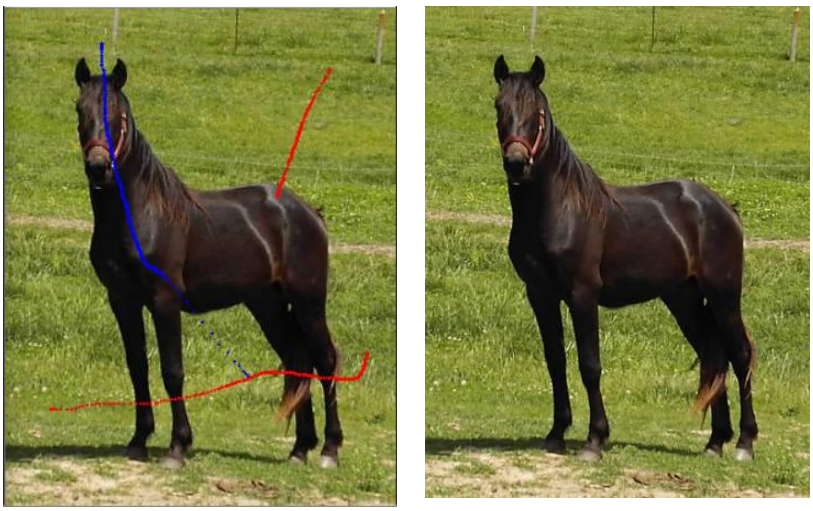


Figure 19. Image with wrong examples given by users (left image) and reload image (right image)

After finishing this step, we have pixels under markers as SEEDS.

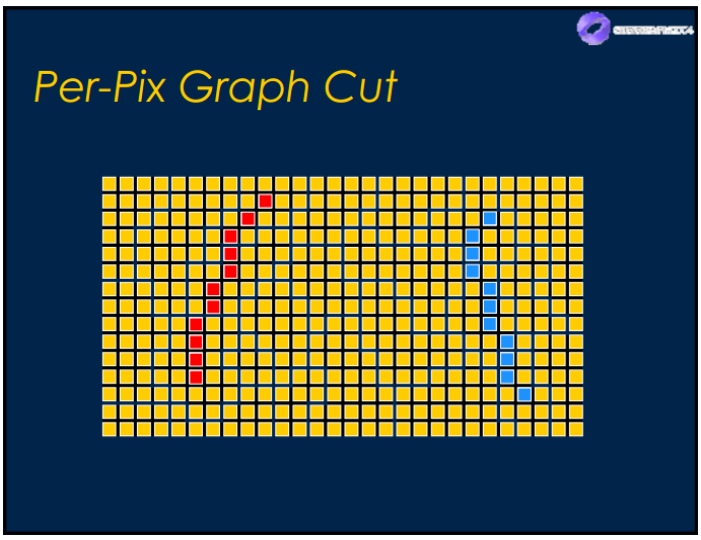


Figure 20. SEEDS pixels for foreground and background (from [24])

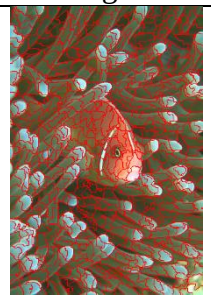
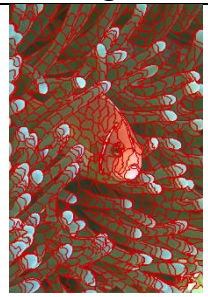
4.4.2 Pre-segmentation

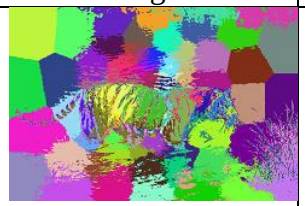
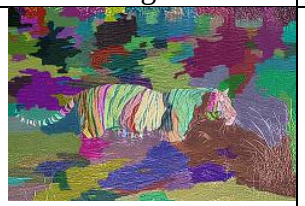
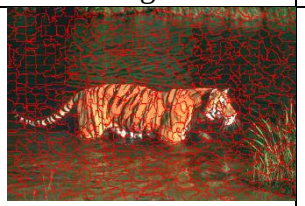
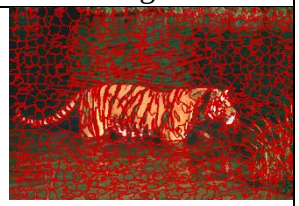
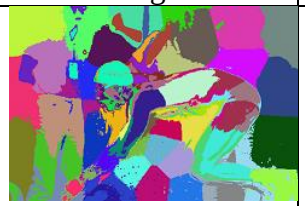
4.4.2.1 Experiment by using different methods

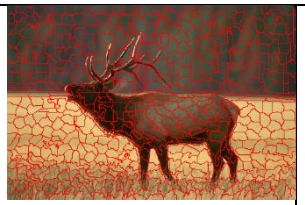
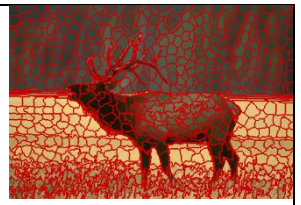
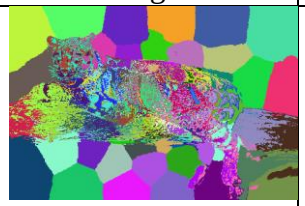
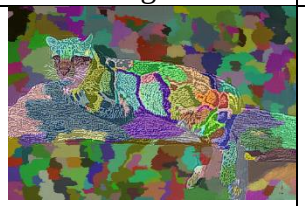
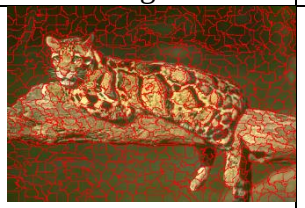
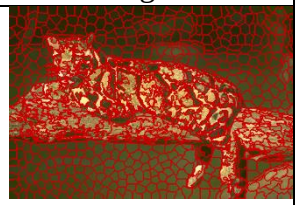
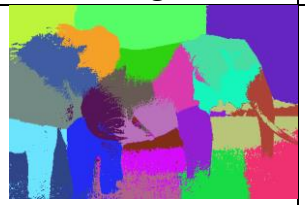
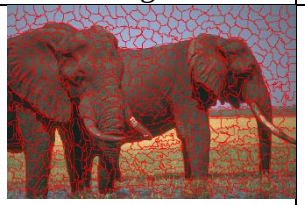
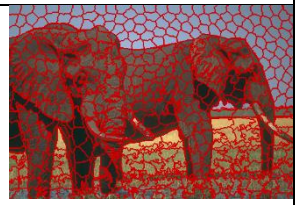
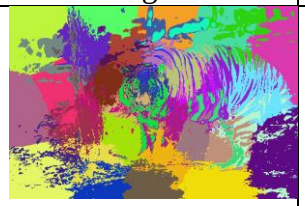
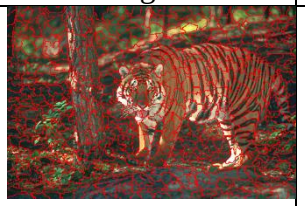
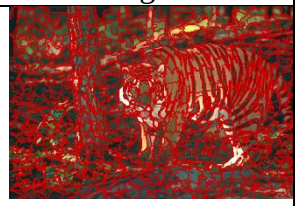
The original image will be segmented into small fragments using the watershed algorithm. Each fragment is represented by a *Super Pixel* with color is computed by mean color of the all pixels belong to the segment. The purpose of this step is to reduce the number or vertices need to add to graph. This initialization step increases the speed of algorithm.
In my experiment, I used 4 methods to pre-segment the image to compare the result that present in Table 6.

- K-means method (OpenCV 2.4.1.3 – Java Implementation)
- Watershed method (OpenCV 3.1.0 Library – C++ Implementation): This implementation is based on "Watersheds in digital spaces: an efficient algorithm based on immersion simulations" of [Vincent and Soille, 1991]
- SEEDs Superpixels method (OpenCV 3.1.0 – C++ Implementation): This implementation is based on a contributed module in Extened Image Processing – OpenCV 3.1.0
- SEEDs Revised Mean Pixels method (OpenCV 3.1.0 – C++ implementation) is based on Superpixel Segmentation using Depth Information, D. Stutz, A. Hermans, B. Leibe, 2014

Table 6. Results of different segmentation methods

Image Size	Original Image	K-means method	Watershed method	SEEDs Superpixels method	SEEDs Revised Mean Pixels
481 x 321 pixel	 124084	 1737 segments	 504 segments	 600 segments	 800 segments
481 x 321 pixel	 189011	 1194 segments	 351 segments	 800 segments	 800 segments
321 x 481 pixel	 210088	 1534 segments	 436 segments	 800 segments	 800 segments

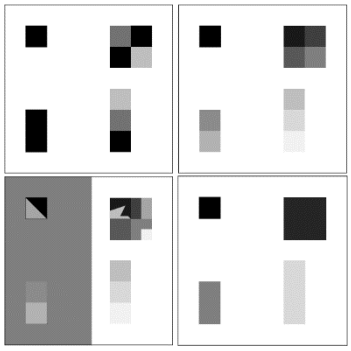
321 x 481 pixel	 388016	 3160 segments	 203 segments	 800 segments	 800 segments
481 x 321 pixel	 108073	 4124 segments	 212 segments	 800 segments	 800 segments
481 x 321 pixel	 153077	 1045 segments	 548 segments	 800 segments	 800 segments
481 x 321 pixel	 65019	 2260 segments	 742 segments	 600 segments	 800 segments

481 x 321 pixel	 41004	 1862 segments	 278 segments	 800 segments	 800 segments
481 x 321 pixel	 160068	 2784 segments	 328 segments	 800 segments	 800 segments
481 x 321 pixel	 296059	 859 segments	 82 segments	 800 segments	 800 segments
481 x 321 pixel	 108005	 4520 segments	 303 segments	 800 segments	 800 segments

4.4.2.2 Conclusion for pre-segmentation

The label of each segment base on pre-segmentation step is very important for initializing and processing graph cut algorithm. There are some problems in pre-segmentation step that need to define before go to conclusions:

- Over-segmentation: pixels belong to same object are classified as belonging to different segments. A single object may be represented by two or more segments.
- Under-segmentation: pixels belong to different objects are classified as belonging to the same object. A single segment may contain several objects.



(Top – Left) Original image

(Top – Right) Correct segmentation

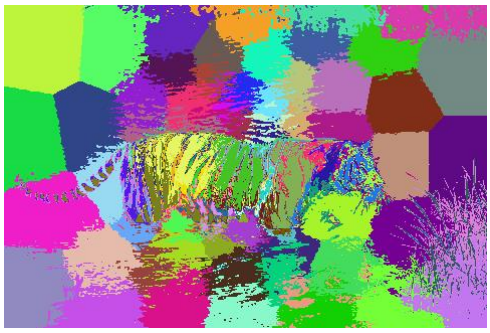
(Bottom – Left) Over-segmentation

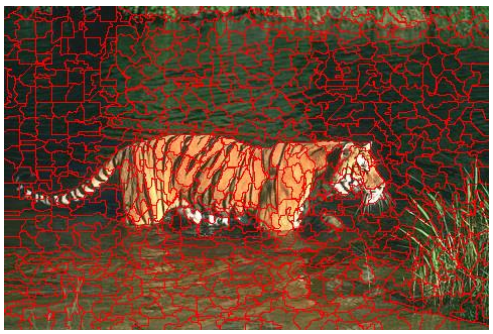
(Bottom – Right) Under-segmentation

Figure 21. Example about pre-segmentation (from [23])

In my experiment, after compare the results of each method, I have some following conclusions in Table 7.

Table 7. Comparison and conclusion of each method

Method	Problem
K-means	 4124 segments (**Over-segmentation problem**) Cannot control the number of segments. The number of segments usually is usually bigger than other methods. Each segment is a set of pixels have same color, not a region in the image. Easy to implement, but require a lot of time and memory spend for pre-segmentation step. Based on segment information given by K-means, Graph Cut algorithm run slow and inaccuracy.

Watershed	 212 segments (**Under-segmentation problem)** Cannot control the number of segments. The number of segment is usually smaller than other methods. There are many regions that include pixels belong to object and another pixel belong to background. Depend on the value of threshold parameter, so pre-segmentation step run unstably. In some cases, the number of segment is small enough with good segments that make Graph Cut algorithm get good results in small time. But with each image, we have to do experiment with threshold value to get best result make this method hard to use.
SEEDs Superpixels	 800 segments **(Under-fitting boundary problem)** Can control the number of segments by setting parameter. By that, the number of segment is small enough to make Graph Cut algorithm run in time. There are few regions, especially in the boundary of object that include pixels belong to object and another pixel belong to background make the boundary of the selected object is unsmooth and low accuracy. Depend on the number of iteration parameter: the bigger iteration parameter is, the higher accuracy in each region, and the slower pre-segmentation step is and vice versa. In most of cases, Graph Cut algorithm get exactly regions that belong to object based on segment information of this method. But boundary of the object is unsmooth even you increase iteration parameter as big as possible.

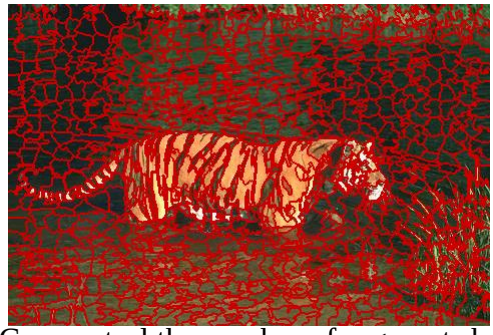
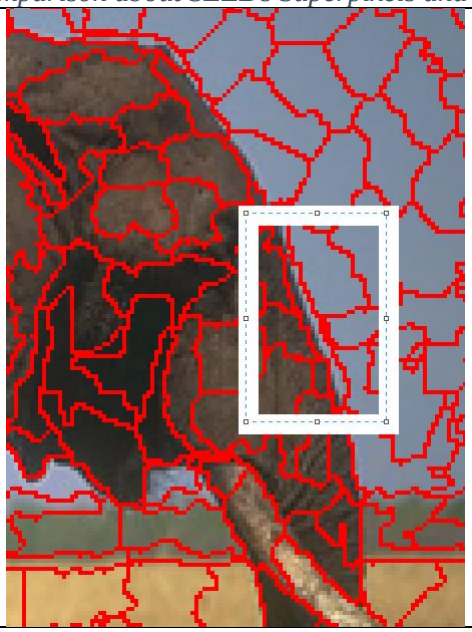
SEEDs Revised Mean Pixels	 800 segments Can control the number of segments by setting parameter. Not even make Graph Cut algorithm run in time, there are rarely regions that belong to both foreground and background. The boundary of each region is closed and smooth such as original image. Depend on the number of iteration parameter (same as SEEDs Superpixels method) In most of cases, Graph Cut algorithm get good final segmentation based on segment information of this method. Boundary of the object is smooth, but with low contract image or image that foreground and background have same color, it is still a hard problem.

Table 8 will show more detailed about the different in same image "296059" of Berkeley dataset after pre-segmentation step between SEEDs Superpixels method and SEEDs Revised Mean Pixels:

Table 8. Detailed comparison about SEEDs Superpixels and SEEDs Revised Mean Pixels

Parameter		
Number of iterations	2000	500
Number of super-pixel	800	800
Number of histogram bin	10	10

Perfect image segmentation - each pixel is assigned to the correct object segment is a goal that cannot usually be achieved. On one hand, with image that object and background have different color, all method return good label results. On another hand, when the color of background is the same with color in object, it is hard to get good segments to separate these regions.
Each method has advantages and disadvantages, so we need to base on image information and the expected value of user to choose appropriate method.

4.4.3 Transfer label data to vertices in graph

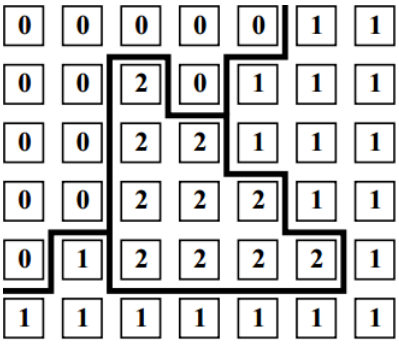


Figure 22. Marker matrix from pre-segmentation step (from [11])

Base on the segment result of pre-segmentation steps, we have marker matrix that present in Figure 12. Each segment has same label value.
To initialize the necessary information for Graph Cut algorithm to create a network, I do some following steps:

- Determine number of vertices in network by counting the number of segments
- Compute mean value of each segments – super pixel
- Determine number of edges in network by finding two pixels that have different label and create edge list.
- Based on samples given by user, we determine new label value of the corresponding sample pixel and mark vertices connect to terminal
    - If sample pixel p belongs to foreground, we mark vertex x (the new label value of pixel p is x) connect to source.
    - If sample pixel q belongs to background, we mark vertex y (the new label value of pixel q is y) connect to sink.
- Apply K-means function provided OpenCV to create information for “source” super pixel and “sink” super pixel.
- Compute edge in the edge list
    - Compute E_1 energy – the cost for connecting a pixel and a terminal.
    - Compute E_2 energy – the cost for every pairs of neighboring pixels.

Use information from marker matrix and SEEDS pixels, we finish this initialization step.

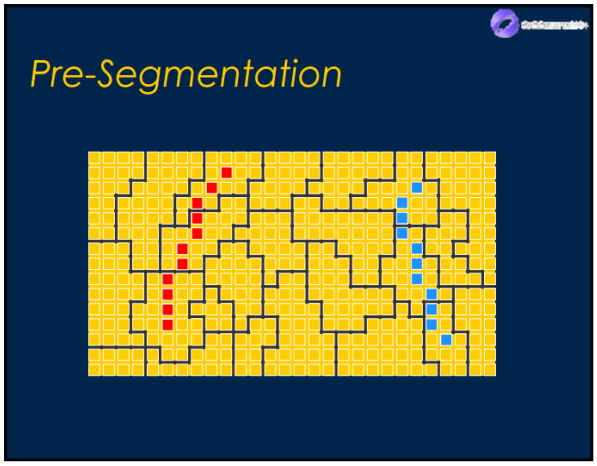


Figure 23. Combine information between marker matrix and SEEDS pixels (from [24])

4.4.4 Graph cut algorithm and final segmentation

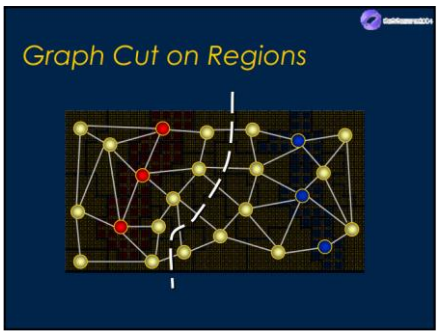


Figure 24. Apply Graph Cut solution on regions (from [24])

In this experiment, we use graph cut solution of Yuri Boykov and Vladimir Kolmogorov present in [7]. They provided a useful library to solve maximum flow problem, so that we just need to call max-flow() interface to get the result and labelling of each vertices.

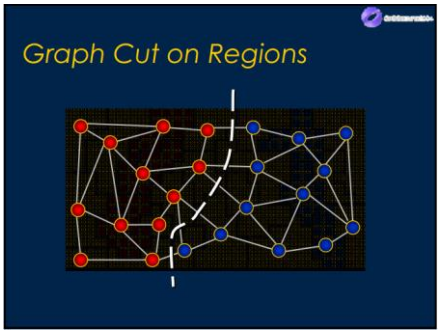


Figure 25. Vertices separate to 2 sets by minimum cut in network (from [24])

After running max flow algorithm, we have the image selection label of all vertices with the value defined as {GraphType::SOURCE, GraphType::SINK}.

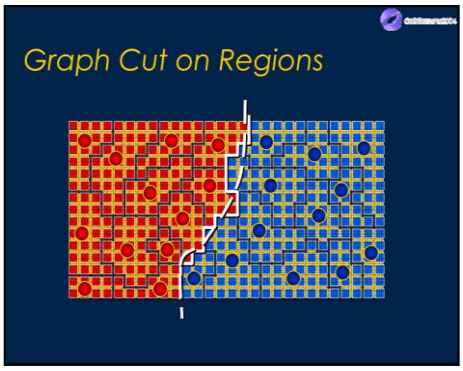


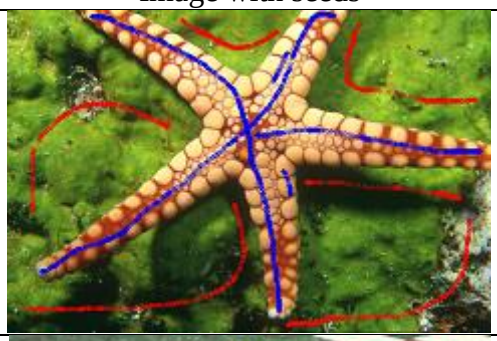
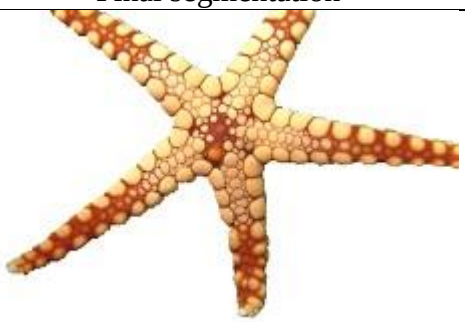
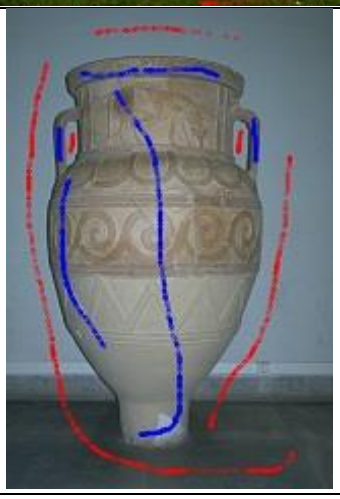
Figure 26. Transfer label back to original image (from [24])

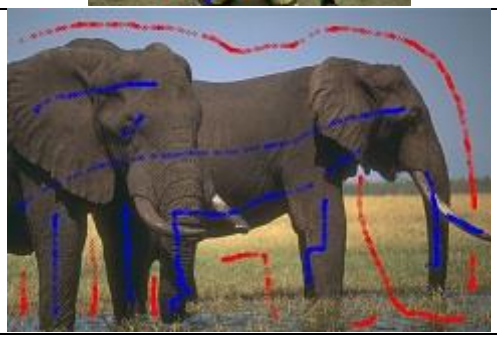
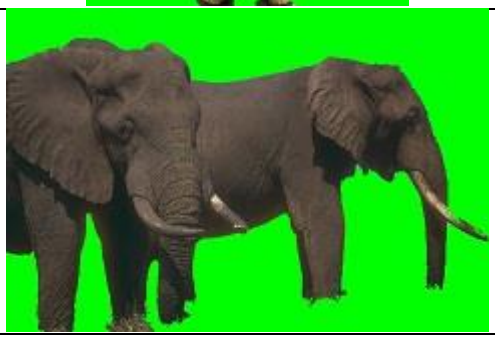
We transfer label value of vertices back to all regions in the original image, change color of background regions to get final segmentation – object selection.

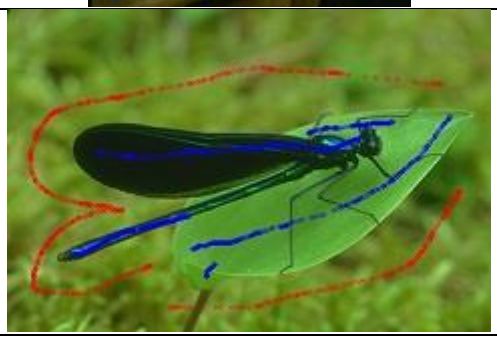
4.5 Experimental results

4.5.1 Berkeley segmentation dataset

Table 9. Experiment of Nam's solution in Berkeley segmentation dataset

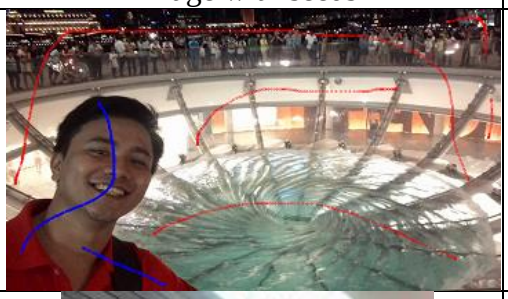
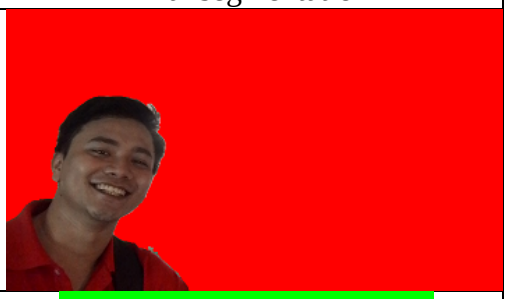
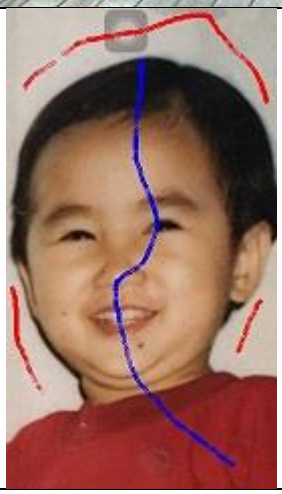
Image	Image with seeds	Final segmentation
12003		
85048		
207056		
227092		

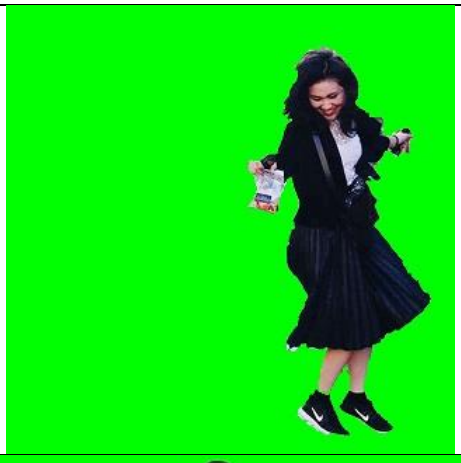
124084		
189011		
187029		
296059		

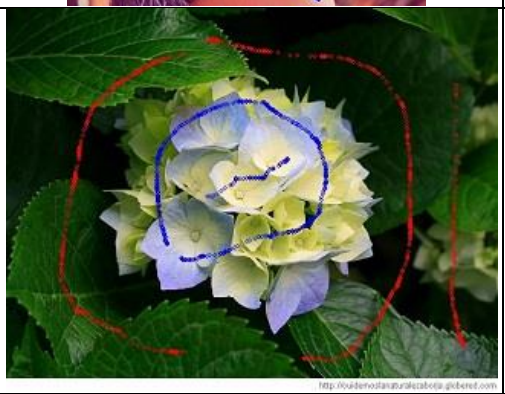
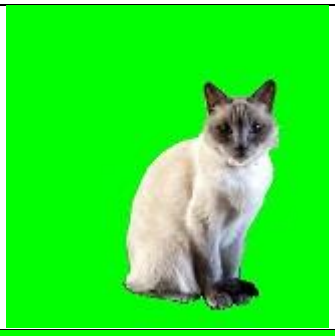
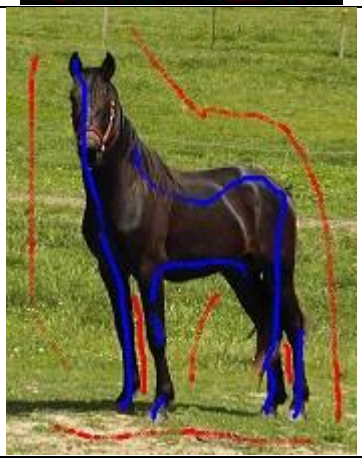
66075		
176035		
187083		
35070		

4.5.2 Our dataset

Table 10. Experiment of Nam's solution in his own dataset

Image	Image with seeds	Final segmentation
Nam_1		
Nam_2		
Emily_Child		

Emily_Dance		
Vango		
River_Safari		
LSPaper		

WSPaper		
Flower		
Cat		
Horse		

Sona		
MocChau		
TrangAn		

4.5.3 Results and Discussion

Some tests result that shown in two sections 5 and 6. There are some good segmentation that we can observe. However, there are 3 limitations I want to discuss in this section.

4.5.3.1 Pixel loss in thin area

We can see very clear in the test result of image “189080”. When the pixels belong to object are in very thin area, they can be segmented into fragments that include both background pixels. When the size of the area is too small, it will be merge into the Super Pixel around so that cannot be separated out.

Figure 27. Image with pixel loss in left and right shoulder of the old man (right picture)

4.5.3.2 Unsmooth boundary

The test result of image “Emily_Dance” shows unsmooth boundaries of the segmentation. The reason is the colour similarity at the edge of the object and background.

Figure 28. Image with unsmooth boundary in dress of the girl (right picture)

4.5.3.3 Poor performance when foreground and background have similar color distribution

When the foreground and the background have similar color distribution, the color difference between each segments has the same value, so the graph cut solution return inaccurate segmentation like we see in the test result of image “Vango”.

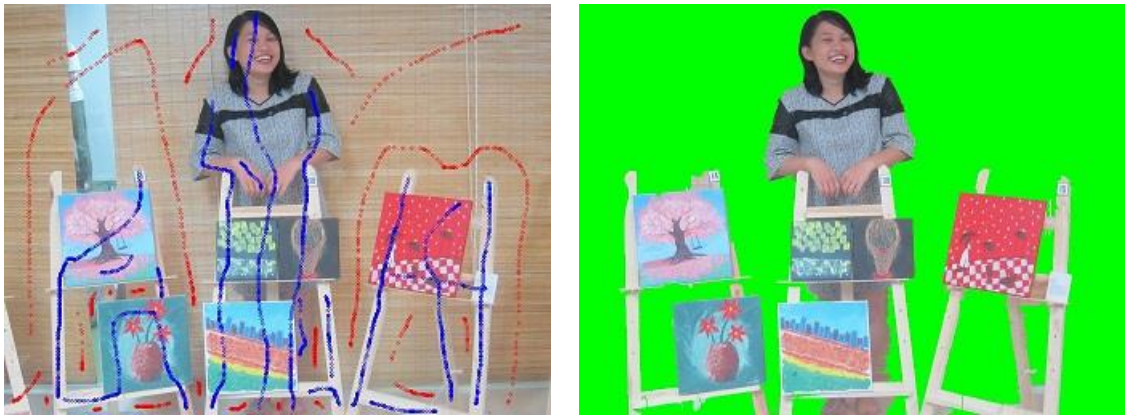


Figure 29. Image with poor performance when the color of foreground and background are similar in the leg of the girl (right picture)

4.6 Conclusion

This project is a simple implementation that based on Lazy Snapping paper to create an interactive segmentation tool. The algorithm and the implementation used openCV library (open source) and max-flow-v3.01 library (open source provided by Yuri Boykov research website). On my laptop (Processor Inter® Core™ i5-3210M 2.50GHz, 6.00 GB RAM), after initializing steps, the project returns final segmentation result quite fast (equal or less than 0.6 sec with Berkeley dataset). However, the boundaries sometimes are unsmooth. The higher accurate in pre-segmentation step when increase the iteration parameter, the better result return and the slower program run. Accept to get better result will pay by time and accept to run faster will pay by accuracy.

Unlike the original Lazy Snapping method [2], we replace the traditional watershed segmentation method [14] with super-pixel segmentation [8] method with the same purpose of dimensionality reduction. However, super-pixel segmentation shows its advantages, such as simpler implementation and better cutout result.

5 Conclusion

We have implemented the Graph Cut for Interactive Segmentation and Lazy Snapping. The key things are constructing the graph from the image and forming the energy function. In the Boykov-Jolly's solution, each pixel is directly mapped to a node, and the energy involves both boundary and region properties. In Lazy Snapping, a node maps to a segment with objective function combines likelihood and prior energy.

To be more specific, we have read more about the min-cut max-flow algorithm and focused on the suggestion algorithm of Boykov and Kolmogorov. We also read, get and implemented the idea to convert the image to the graph: a pixel to a node, the neighbor relationship to the edge. Also, some example suggestion for detailed implementation was used, including the ad-hoc function for boundary and color histogram distribution for region term. We chose the 8-connected neighbor instead of 4-connected that was suggested to use more region properties and the Manhattan distance for color intensity difference in the multi-dimensional image. Finally, we built an automatic testing to process dataset in batch mode.

For Lazy Snapping, the pre-segmentation step groups pixel into segments. These are considered as super-pixels. We read, did some experiments and compared five popular segmentation algorithms. Based on the comparisons and conclusions, we renewed Lazy Snapping method by replacing the traditional watershed with SEEDs algorithm. Silent Lazy Snapping – our improvement from original method have been implemented and tested with the image in Berkeley Segmentation Dataset combine with our own dataset, and some usable results are presented in Experimental results session.

Our future direction for interactive Graph Cut Segmentation is using other feature descriptors, such as Histogram of Oriented Gradients, to find out a better way of describing terms. Due to working on raw pixel, this method can't be used to process high resolution image without high hardware requirements, and other improvements (such as GrabCut, Lazy Snapping) are suggested to be alternatives. But the editing windows can be zooming freely for more accurate view. Additionally, some options will be added to GUI so that the end-user can control the lambda or add more seed until satisfied without touching the codes.

Based on the discussion about limitations of Silent Lazy Snapping framework, we will also investigate the boundary editing step in the Lazy Snapping method for more accuracy and simpler action from the user. We can combine other filters (Gaussian, Guided filter, etc.) with Silent Lazy Snapping method to get more details in the boundary to remove noises and JPEG block artifacts in the border of the selected subject.

References

[1] Yuri Y. Boykov, Marie-Pierre Jolly. *Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images*. ICCV, Vancouver, Canada, July 2001.

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

[3] C. Rother, V. Kolmogorov and A. Blake. *"GrabCut" – Interactive Foreground Extraction using Iterated Graph Cut*. SIGGRAPH, 2004.

[4] J. Jia, J. Sun, C. K. Tang, and H. Y. Shum, *Drag-and-Drop Pasting*. Proceedings of ACM SIGGRAPH', 2006

[5] S. Vicente, V. Kolmogorov and C. Rother. *Graph cut based image segmentation with connectivity priors*. Computer Vision and Pattern Recognition, 2008.

[6] Frank R. Schmidt, Eno Toppe and Daniel Cremers. *Efficient Planar Graph Cuts with Applications in Computer Vision*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Miami, Florida, 2009.

[7] M. van den Bergh, X. Boix, G. Roig, B. de Capitani, L. van Gool for SEEDS. *Superpixels extracted via energy-driver sampling*. Proceedings of the European Conference on Computer Vision, pages 13-26, 2012.

[8] Stutz, A. Hermans, B. Leibe. *Superpixel Segmentation using Depth Information*. Bachelor thesis, RWTH Aachen University, Aachen, Germany, 2014.

[9] L.R. Ford, Jr. and D.R. Fulkerson. *Maximal Flow Through a Network*. Canadian Journal of Mathematics, chapter 8: page 399-404, 1956.

[10] L.R. Ford, Jr. and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962

[11] Y. Boykov and V. Kolmogorov. *An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision*. In 3rd. Intnl. Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR). Springer-Verlag, September 2001, to appear.

[12] C. Fowlkes, D. Martin, J. Malik. *Local Figure/Ground Cues Are Valid for Natural Images*. Journal of Vision, 7(8):2, 1-9.

[13] Duda, R.O., Hart, P. E., Stork, D. G. for Pattern Classification (2nd Edition). Wiley Press.

[14] Vincent, L., Soille, P. *Watersheds in digital spaces: an efficient algorithm based on immersion simulations*. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-13(6): page 583-598, June 1991.

[15] Meyer, F. Color image segmentation. In *IEE International Conference on Image Processing and its Applications*, pages 303–306, May 1995. Maastricht, The Netherlands.

[16] Meyer, F. *The Watershed Concept and Its Use in Segmentation: A brief History*, Available online: http://arxiv.org/pdf/1202.0216.pdf (accessed on 18 July 2016).

[17] Andrew V. Goldberg and Robert E. Tarjan. *A new approach to the maximum flow problem*. Journal of the Association for Computing Machinery, 35(4): page 921-940, October 1988.

[18] Yefim Dinitz. *Algorithm for solution of a problem of maximum flow in a network with power estimation*. Doklady Akademii nauk SSSR 11: page 1277-1280, 1970.

[19] Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, Tsuhan Chen. Interactive Co-segmentation of Object in Image Collections, page 5, 2011.

[20] Chris Solomon and Toby Breckon. Fundamentals of Digital Image Processing, page 263-265, 2011.

[21] Le Minh Hoang. Giai thuat va lap trinh, page 257-263, 2006.

[22] (2013). Image Analysis: Morphological Operations. Available online: http://www.ualberta.ca/~ccwj/teaching/image/morph/ (accessed on 14 July 2016).

[23] Available online: www.cs.uu.nl/docs/vakken/ibv/reader/chapter10.**pdf** (accessed on 14 July 2016).

[24] Available online: http://www.cs.tau.ac.il/~dcor/Graphics/adv-slides/LazySnappingand-GrabCut.pdf (accessed on 15 July 2016).

[25] Available online: www.cs.cornell.edu/courses/cs664/2005fa/Lectures/lecture8.pdf (accessed on 16 July 2016).

[26] Maximum flow problem from Wikipedia: The Free Encyclopedia. Available online: https://en.wikipedia.org/wiki/Maximum_flow_problem (accessed on 17 July 2016).

[27] OpenCV Document - Extended Image Processing: SuperpixelSEEDS. Availabel online: http://docs.opencv.org/3.0-beta/modules/ximgproc/doc/superpixels.html (accessed on 20 July 2016).