CS445 Computational Photography Final Project
Graph cut-based Image Segmentation
Final Report

Zhiyan Jiang, Shikong Chen

{zjiang2, shikong2}@illinois.edu

1. Introduction

The final project implemented by this team is graph cut-based image segmentation. The goal is to enhance understanding of the fundamentals of graph cut theory. This topic was briefly introduced in Week 4 video without many details. It was chosen as the final project because image segmentation is a fundamental and important aspect in computational photography and computer vision. Many other techniques, such as image compositing, image warping, image stitching, require high quality imagery patches as input. These patches can be the outcome of graph cut based on various sources. A deep understanding of this topic helps gain more insights into computational photography and understand more recent research on graph segmentation.

2. Fundamentals of Graph cut-based segmentation

The basis idea of graph cut-based segmentation is modeling an image as bidirectional graph, in which each pixel is a node. The similarity between adjacent nodes, defined as edge affinity, is assigned as edge weight. The problem is defined as segmenting the graph in two parts through cutting edges whose total weights are minimized. This problem can be solved by its dual problem – Max flow problem. In this problem, an additional source node and another sink node (notated as terminals) are introduced. Edge affinity is assigned as the edge capacity that indicates the maximum possible flow through a given edge. The problem is defined as finding the maximum flow from the source node to the sink node. Two constraints are applied: 1) for each node, the inflow equals the outflow; 2) the actual flow through an edge must not exceed the edge capacity. Finally, nodes with inflow from the source node are classified as the foreground object pixels, and the remaining nodes with outflow to the sink are classified as background pixels.

There are two key components in the problem – terminal affinity and edge affinity. Terminal affinity is defined as:

$$\text{source-foreground affinity} = -\lambda \log P_B(p)$$

$$\text{sink-background affinity} = -\lambda \log P_F(p)$$

where $P_F(p)$ and $P_B(p)$ denote the probability of node belonging to foreground and background, respectively; λ is a scaling factor.

Edge affinity is defined as:

$$\text{edge affinity} = k_1 + k_2 e^{-\frac{\|c(x)-c(y)\|^2}{2\sigma^2}}$$

where k_1 specifies the preference of short and roundish boundary or long squiggly boundary; k_2 is a scaling factor; σ^2 is the averaged standard deviation of edge weights for a given node; $c(x)$ and $c(y)$ denote measurement of affinity, which can be intensity, distance, or texture.

Higher affinity denoted higher similarity between the nodes and higher cost associated with cutting the edge. In other words, similar pixels are less likely to be separated.

3. Implementation

The implementation was completed using Jupyter Notebook on local laptops. A few external packages were used, including *OpenCV*, *PyMaxflow*, *scikit-learn*, *scipy*, *numpy*, and *matplotlib*. Detailed package list can be found in the submitted code.

The image data are downloaded from a tutorial on image segmentation with graph cut (Reference 1). The reason to use these data is that the tutorial also provides final results that can be used as benchmark solutions to verify this project.

The implementation has three parts – Preparation, Terminal and non-terminal affinity, Build graph.

In the "Preparation" part, necessary modules, raw image, and scribed images are loaded. Two options are provided for specify scribbles.

- Option 1: Two sample points are manually specified to help identify the foreground and the background scribbles.
- Option 2: Interactively draw foreground and background scribbles.

In the "Terminal and non-terminal affinity" part, the foreground and background scribble masks are generated. To compute the terminal affinity, two methods are implemented – histogram of intensity and Gaussian Mixture Model (GMM). The former is suitable for processing black & white images and the latter handles colored images well. In the histogram approach, the histogram curves can be fitted with either normal distribution or can be simply filtered by low-pass filter. In the GMM approach, the colored pixels are first clustered using KMeans method (K=2). Then, the prior, mean and covariance can be calculated for each latent cluster. Finally, a Gaussian Mixture Model is applied to calculate the probability of belonging to foreground and background for each node. Based on the tests, the GMM approach yields much better results than the histogram approach, but cannot be applied to black & white images.

In the 'Build graph' part, a graph is established using the package *PyMaxflow*. The bidirectional edges are assigned of edge affinity. Each node is assigned of terminal affinity. Then the maximum flow is calculated and all the nodes are classified as either foreground or background. The final foreground segment is visualized.

The challenging part in the project is implementing the GMM model, and figuring out the parameters for affinity, both of which are not elaborated in the lecture. The parameters were determined by trial-and-error.

4. Results

The major intermediate results are presented below by using a bunny image as an example to demonstrate the implementation.

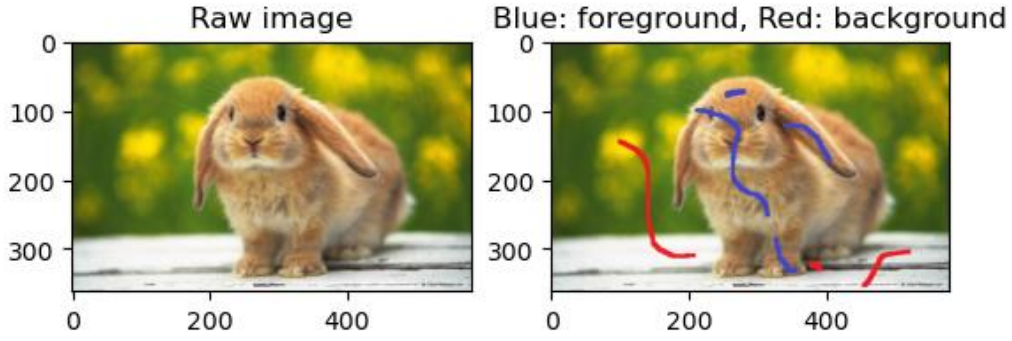


Figure 1 Raw image and image with scribbles

Figure 2 Interactive interface for drawing foreground and background scribbles

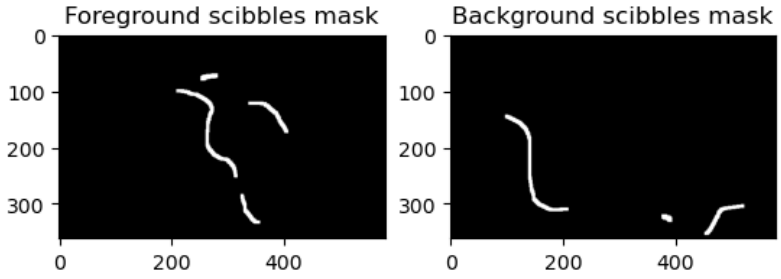
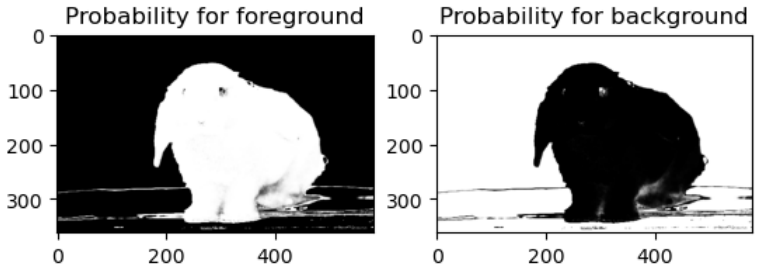
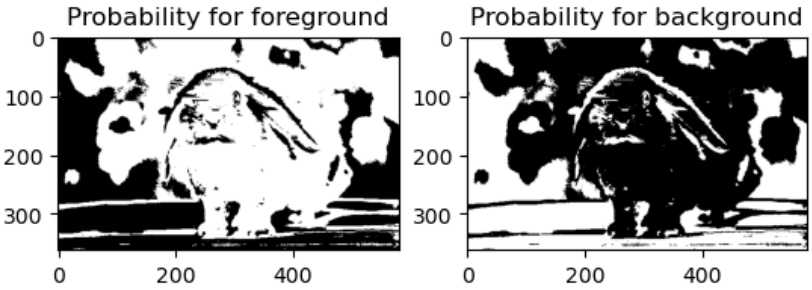


Figure 3 Masks of foreground and background scribbles



(a) by GMM approach (recommended)



(b) by histogram – smooth approach

Figure 4 Probability of belonging to foreground and background.

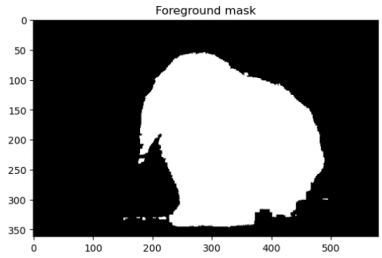
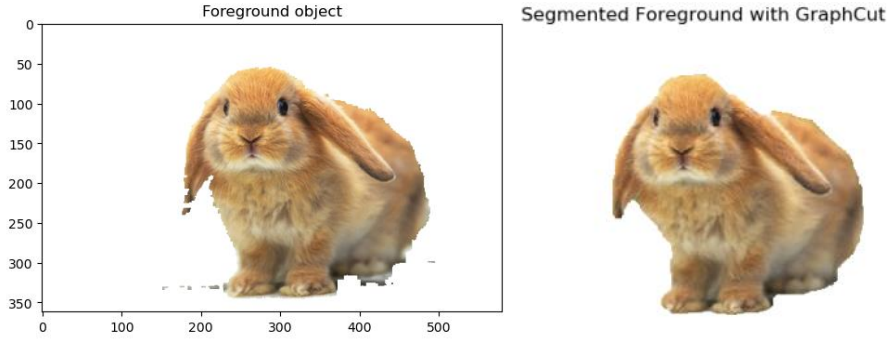


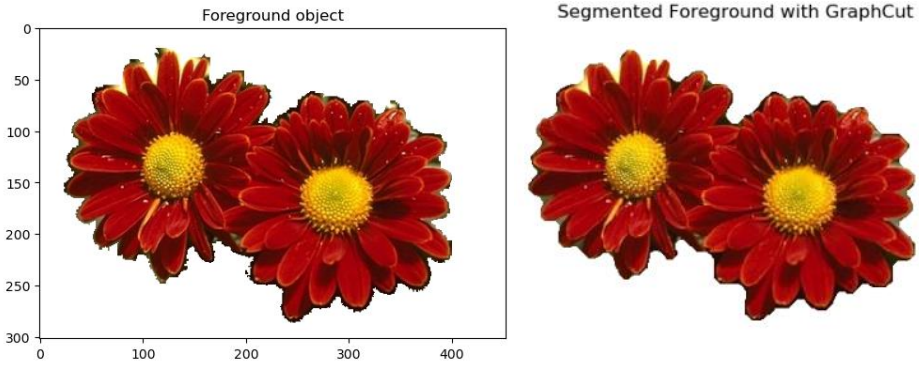
Figure 5 Final foreground object mask

Example results from this project are compared with the benchmark solutions as below:



(a) This project (b) Benchmark

Figure 6 Comparison of results on bunny image



(a) This project (b) Benchmark

Figure 7 Comparison of results on flower image

5. Conclusions

Graph cut-based image segmentation with heuristic foreground and background scribbles is implemented as the final project. The Gaussian Mixture Model produces the best terminal affinity. The final foreground object is decent and meets the expectation. Future work includes parametric studies for helping tune the affinity parameters.

6. Contributions

Zhiyan worked on the implementation of graph cut segmentation algorithm, proposal and final report drafts. Shikong worked on implementation of interactive interface for drawing scribbles, review of proposal and final report.

We expect to receive 10 points for the challenge/innovation component of grading.

7. References

1. Tutorial of graph cut segmentation (https://sandipanweb.wordpress.com/2018/02/11/interactive-image-segmentation-with-graph-cut/)

2. Boykov, Y.Y., and Jolly, M-P. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. Proceedings of "International Conference on Computer Vision", Vancouver, Canada, July.

3. Rother, C., Kolmogorov, V., and Blake, A. "GrabCut" – Interactive Foreground Extraction using Iterated Graph Cuts.

4. PyFlowmax tutorial, (http://pmneila.github.io/PyMaxflow/tutorial.html), accessed on Apr.23.

5. A tutorial on implementing graph cut, (https://notebook.community/long0612/randProbs/graphcut/graphcut), accessed on Apr.23.

6. An assignment on implementing graph cut, (https://cw.fel.cvut.cz/old/courses/b4m33dzo/labs/4_segmentation), accessed on Apr.23.

7. OpenCV, Interactive Foreground Extraction using GrabCut Algorithm, (https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html), accessed on Apr.23.

8. Scovanner, P. An Introduction to Graph-Cut, (https://www.cs.ucf.edu/courses/cap6411/cap6411/spring2006/Lecture11.pdf), accessed on Apr.23.

9. George Mason University, Image Segmentation continued Graph Based Methods, (https://cs.gmu.edu/~kosecka/cs482/lect-segmentation-part2.pdf), accessed on Apr.23.