Image Segmentation via Network Flow

Project Mid-term Progress Report

Li, Peixin	Nie, Yuchen	Wang, Chunxi	Wang, Sihan	Xiao, Lijun
pxli@bu.edu	ycnie@bu.edu	tracycxw@bu.edu	shwang95@bu.edu	ljxiao@bu.edu

April 8, 2018

1 Overview

1.1 Description

In computer vision, *image segmentation* is the process of partitioning a digital image into multiple segments (sets of pixels, also known as *super-pixels*). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics. We want to segment the foreground from the background in an image in this project. And network flow can be used as a way to solve the segmentation problem efficiently in the case of two segments.

1.2 Feature List

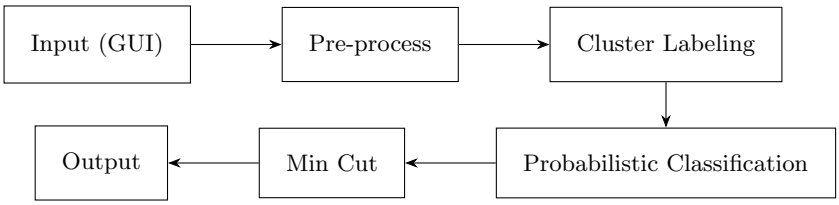


Figure 1: Flow Chart of the Project

- **Input**: The image is opened as an array of pixels, each pixel contains three value of R, G, B for representing the color. Currently there are two modes in the prototype: manual annotation and k-mean, the former one input two images and the latter one input one image. In the final program, the manual annotation mode will only input one image, the user then need to manually input the annotation through the GUI.
- **Pre-process**: The original image is too large for doing minimum cut fast, so it has to be down-sampled.
- **Cluster labeling**: Before building the flow, we need to first cluster the image, i.e., find the background and foreground part by using kinds of clustering algorithm. This step can be also achieved by input an annotation image by manually indicate the area of background and foreground.
- **Probabilistic classification**: After clustered, we need to build the flow and calculate the minimum cut. Probabilistic classification provides a way to create capacities for edges between neighborhood pixels.

- **Minimum Cut**: Calculate the maximum flow for the generated network flow, then find the minimum cut set which are foreground pixels and background pixels. Segmentation is done in this step. A mask image is generated for output.
- **Output**: Four images are outputted: the original image, the foreground, the background and the up-sampled mask.

2 Schedule

- Completed
  - **March 29**: Read image and implement down-sampling for clustering. Determine the features for k-mean clustering algorithm.
  - **April 4**: Clustering the data by k-mean. Deal with the output of k-mean and manual annotation for use in segmentation.
  - **April 10**: Segment background and foreground based on probabilistic classification.
- Not completed
  - **April 20**: Finish the GUI application.
  - **April 25**: Finish testing and improvement.
  - **April 30**: Ready for final presentation.
  - **May 4**: Final report due.

3 Prototype

3.1 Instruction

3.1.1 Requirements

The prototype is written in Python and is tested on Python 3.6.4. The following table shows necessary libraries required for running the prototype and a concise description of the libraries.

Name	Version	Description
OpenCV	3.4.0.12	Computer vision library
Matplotlib	2.1.2	2D plotting library
NumPy	1.12.0	Scientific computing library
SciPy	1.0.0	

Table 1: Requirements of the Prototype

3.1.2 Files

The prototype comes with 7 files: 4 core program, 1 requirement description, and 2 test images. The following table shows the description of these files.

Filename	Description
bayes.py	Naïve Bayes classifier class
ek.py	Edmonds-Karp algorithm class
main.py	Main program in annotation mode
main_kmean.py	Main program in k-mean mode
requirement.txt	Requirement packages list for `pip`
teagle.jpg	Test image
teagle_anno.jpg	Test image annotation

Table 2: Requirements of the Prototype

3.1.3 Usage

First, you need to clone or download the prototype from GitHub, the link for the prototype is `https://github.com/shwang95/EC504_Project`. Then you can use command `pip install -r requirement.txt` to install the required libraries. At this point, you are ready to run the prototype. The usage of the main programs are:

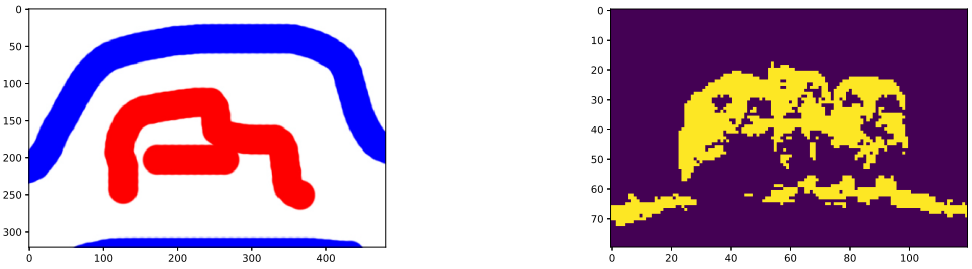
- **main.py**: `python main.py [image_name] [annotation_name]`
  - `image_name`: The name for the image to be segmented.
  - `annotation_name`: The name for the annotation image with blue lines indicate background and red lines indicate foreground, the size should be the same as the image to be segmented.
- **main_kmean.py**: `python main_kmean.py [image_name]`
  - `image_name`: The name for the image to be segmented.

3.2 Pre-process

We use bilinear interpolation to down-sample the image, this function is provided by existing Python scientific computing library `scipy`.

3.3 Cluster Labeling

K-mean clustering algorithm provide a sufficient way to find different clusters, in our project, we use k-mean for finding the approximate foreground and background. We can also use a manually indicated annotation image with the same size of the image need to be segmented as the basic material for cluster labeling.



(a) Manual Annotation

(b) K-mean

Figure 2: Different Clustering Method

3.4 Probabilistic Classification

We use the simplest *4-neighborhood*, which is where a pixel is connected to the pixels directly above, below, left, and right, to define the graph.
To determine the weights on these edges, we need a segmentation model that determines the edge weights between pixels and between pixels and the *source* and *sink*. Let w_{ij} be the weight between

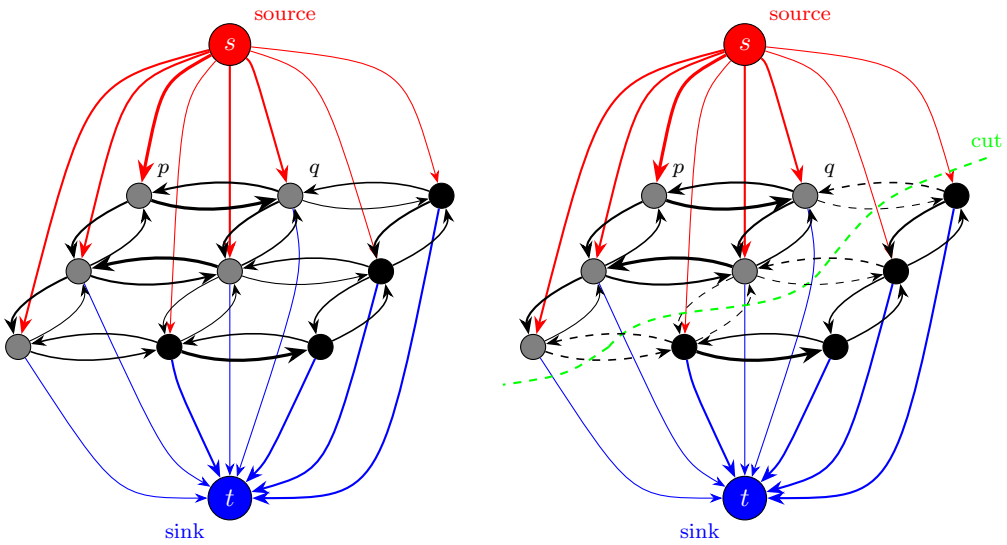
pixel i and j, and s and t represent the source and the sink, then

$$w_{si} = \frac{p_F(I_i)}{p_F I_i + p_B(I_i)},$$
$$w_{it} = \frac{p_B(I_i)}{p_F I_i + p_B(I_i)},$$
$$w_{ij} = \kappa e^{-|I_i - I_j|^2/\sigma},$$

where $p_F(I_i)$ and $p_B(I_i)$ indicate the probabilities for the pixel be in foreground and background, which we use the Gaussian naïve Bayes classifier (NBC) for calculating.
The w_{ij} describe the pixel similarity between neighbors, similar pixels have weight close to κ, dissimilar close to 0. The parameter σ determines how fast the values decay toward zero with increasing dissimilarity.

3.5 Minimum Cut

According to the max-flow min-cut theorem, finding the minimum cut is equivalent to finding the maximum flow between the source s and the sink t. Here we use the Edmonds-Karp (E-K) algorithm for finding the minimum cut. The running time for E-K algorithm in a directed graph $G = (V, E)$ is $O(VE^2)$. E-K algorithm is identical to Ford-Fulkerson (F-F) algorithm. The different of the E-K algorithm and the F-F algorithm is how they find the s-t path. The E-K algorithm uses the breadth-first search (BFS) wile the E-K algorithm uses the depth-first search (DFS).



(a) A graph G (b) A cut on G

Figure 3: Example of a directed capacitated graph, edges' capacities are reflected by their thickness.

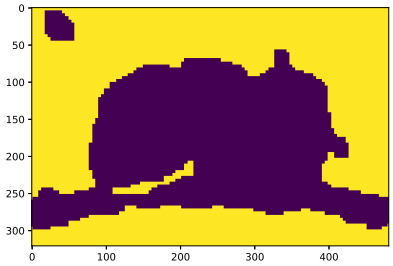
After finding the maximum flow, we need to determine the minimum cut set, i.e., the nodes that can reach the source and the nodes that can reach the sink. Since the BFS used in the E-K algorithm is a first in, first out ($FIFO$) method, the terminal point i.e., the source or the sink, that can reach/be reached by a node in the last iteration is the set where it should belong to.

3.6 Rebuild Image

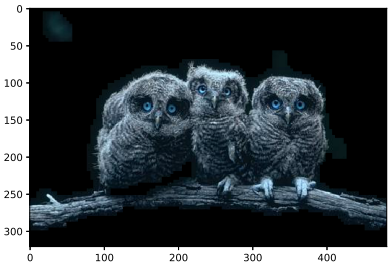
By the minimum cut, we obtained pixels of the foreground and the background, which is an array of 0 and 1 (the *mask*). Then we can then up-sampled the mask and apply it on the original image,

then print it out. The following figure shows the output of different cluster labeling method.

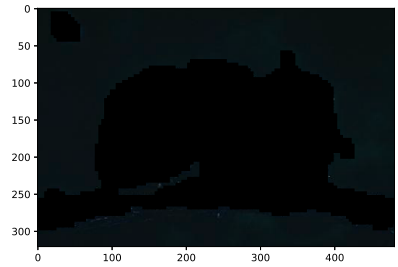
(a) Original Image



(b) Mask Image



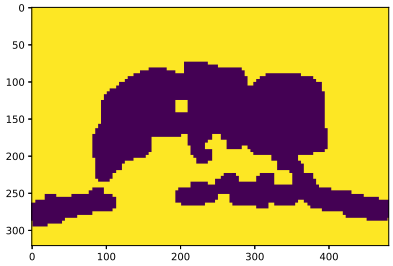
(c) Foreground Image



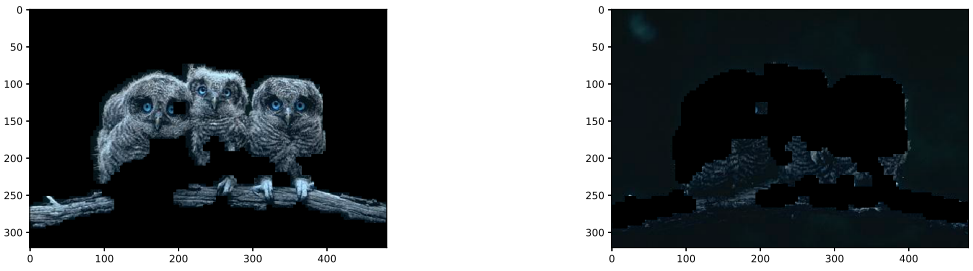
(d) Background Image

Figure 4: Output of Manual Annotation Method

(a) Original Image



(b) Mask Image



(a) Foreground Image

(b) Background Image

Figure 6: Output of K-mean Method

4 Challenges

As the extension to our project, we need to add GUI for users to directly indicates the foreground and the background area, which is a challenge. We intend to use the GUI function provided by python computer vision library `opencv` to achieve this. Another challenge is the training speed of the *NBC* need to be improved.

References

[1] Edmonds, Jack; Karp, Richard M. (1972). "Theoretical improvements in algorithmic efficiency for network flow problems". *Journal of the ACM.* Association for Computing Machinery. **19** (2): 248–264.

[2] Goodrich, Michael T.; Tamassia, Roberto. (2002). *Algorithm Design: Foundations, Analysis, and Internet Examples.* John Wiley & Sons, Inc. pp. 391–395. ISBN 0-471-38365-1.

[3] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. (2009). *Introduction to Algorithms* (Third Edition). MIT Press. pp. 727–730. ISBN 978-0-262-03384-8.

[4] Solem, Jan Erik. (2012). *Programming Computer Vision with Python.* O'Reilly Media, Inc. pp. 167–207. ISBN 978-1-449-31654-9.

[5] Boykov, Yuri; Kolmogorov, Vladimir. (2014). "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision". *IEEE Transactions on PAMI.* IEEE. **26** (9): 1124–1137.