Project 2 – Binary Image Segmentation

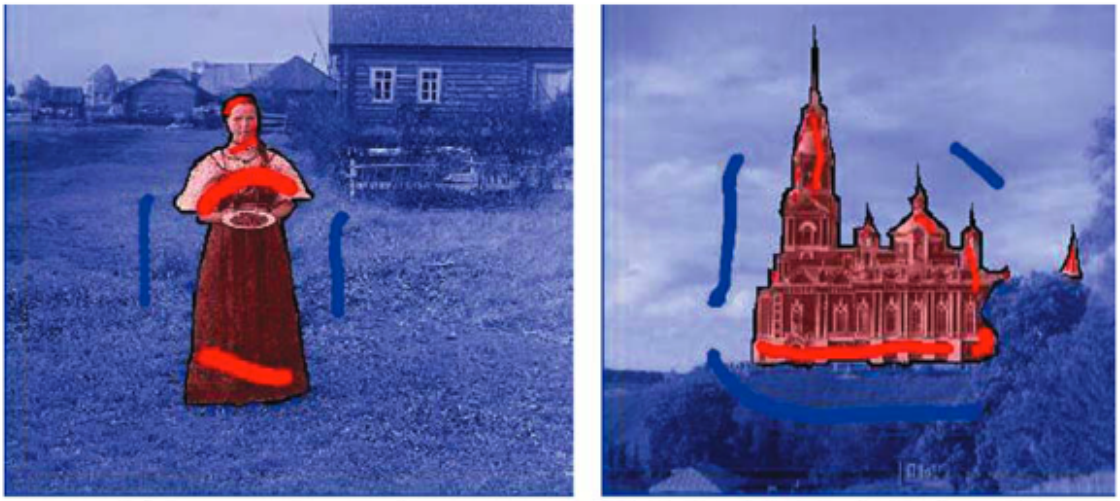
COMP 6651 – Winter 2018

Dr. Tiberiu Popa

Introduction

Binary Image segmentation is the process of classifying the pixels of an image into two categories: pixels belonging to the foreground objects of an image and pixels belonging to the background objects of an image. Figure 1 shows an example of image segmentation where the pixels belonging to the foreground objects are highlighted in red and the pixels for the background are highlighted in blue. (Image I taken from the web). Image segmentation is an important problem in image processing and computer vision with many application ranging from background substraction and removal to object tracking, etc.

While there are many ways to model this problem mathematically, the one we will use for this homework in as a min-cut finding problem with multiple sources and sinks [1] as described below. In this project, you are asked to implement a simple binary image segmentation technique using min-cut and the OpenCV library.



(a) A woman from a village (b) A church in Mozhaisk (near Moscow)

Figure 1. Image segmentation examples from eth web.

The Min Graph-cut problem

Given a connected graph G(V, E), and two vertices s (source vertex) and t (sink vertex), a cut is a subset of edges E' that disconnects any path from s to t. A minimum cut E'' is a cut where the sum of the weights of all its edges is not larger than any other cut E'. The problem of minimum cut can be generalized to the case where more than one source or sink exist as shown in the textbook. It is easy to observe that any cut of G classifies the vertices in V into two disjoint sets: vertices connected to s and vertices connected to t.

As illustrated in class the min-cut problem and max-flow are dual to each other, in other words, if we solve for one we also get the solution for the other. In class the Ford-Fulkerson algorithm for finding the maximum flow was presented. However, for this homework you can choose any algorithm you wish as long as you submit your

own original implementation that is not take in full or in parts from internet resources or other people either students in the class or anyone else.

Image Segmentation using Graph-cuts

The binary image segmentation problem can be reduced to finding a minimum cut in the graph induced by the image graph: the pixels are the vertices or nodes in the graph and we have edges between any neighboring pixels in the horizontal and vertical direction (i.e. any vertex has maximum 4 neighbours).

Given an initial set of user specified foreground and background pixels (displayed as red and blue strokes in Figure 1) you can use these as sources and targets (or sources and sinks depending on the terminology) and find a minimum cut in this graph. After removing the edges in this cut, the pixels in the image will be disconnected such that each pixel in the image will be connected to either the source the target. If a pixel is connected to the source than it is a foreground pixel and if it is connected to the target than it is a background pixel.

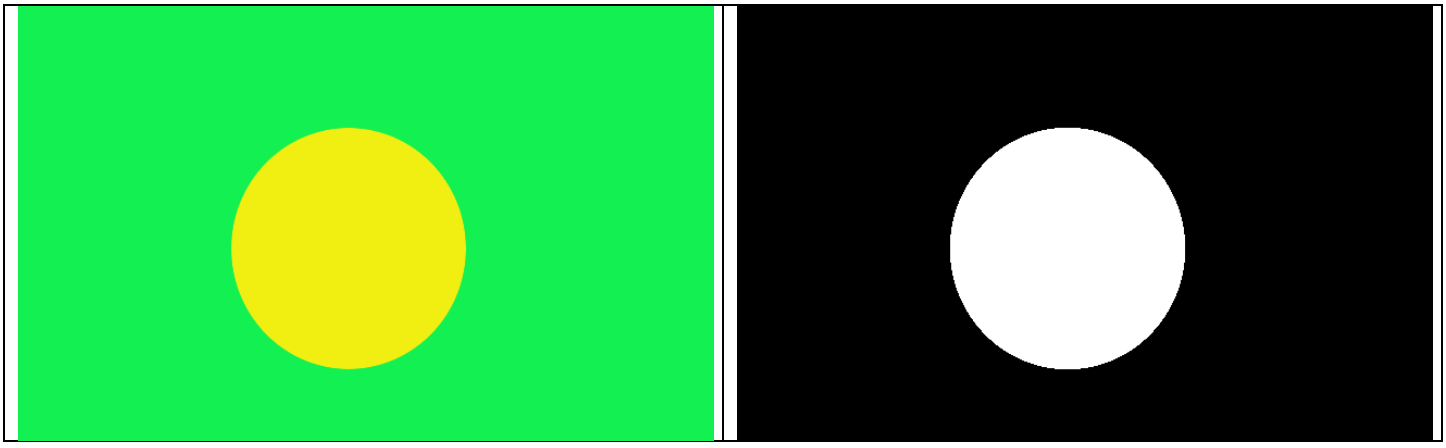
For further resources including how to set up the weights in such a graph we recommend the following resources: [1, 2,3];

Specifications

The program has 3 arguments: an input image, a configuration file that provides the initial set of foreground and background points and an output image. Your program has to compute the binary segmentation of the pixels in the input image based on the initial segmentation provided in the configuration file. The output has to be an image of the same size of the input image such that each pixel is either white (foreground) or black (background). You are allowed to save it as a 3-channel image for simplicity, but if the image has a pixel of any other color a grade of 0 will be awarded for that test case.

Example: *./seg simple.png config.txt result.png*

Your program has to be called *seg* for our automatic testing. An example input output is shown below where left is the input and right is the output. The input image and the configuration file is provided together with the source code.



What is given

You are given a codebase that simple reads the input image, it clones it as the output image and replaces the pixels read from the configuration file rendering them in red if they are foreground and in blue if they are background. Note that since we are changing just one pixel, the difference between the input and output image is pretty subtle.

This codebase is provided simply as a platform for your solution as well as a jump start with OpenCV.

The codebase has a src folder that contains one files: main.cpp. The *main.cpp* contains the main function where all the provided code resides. If you wish to add additional files (and we recommend that you do) you need to change the CMakefile.txt file appropriately.

The codebase provided is configured and compiled with cmake. The only other dependency is opencv. The project is configured to run on any machine that has a properly installed opencv, but it was tested only on the linux machines in the labs as well as on a Mac OS X Sierra or later.

In the linux lab you need to run first the command *module load opencv*. This command will mount the folder where OpenCV is installed. Read carefully the README file provided with the code to compile and run the project. After compilation, the executable will be called seg and that the executable is stored in the build folder and it is not mixed with the source files that are stored in the *src* folder.

Additionally in the asset folder (as well as in the main build folder) you have the image above as well as a configuration file (*config.txt*) that contains an sample of 8 points, 4 in the background and 4 in the foreground.

For testing you can use this image as well as any images from the web. Note that you will have to create your own initial segmentation in the configuration file. Note that if the image is complicated, you will need to add a lot of point in order for the algorithm to give a reasonable result.

Submission

You have to implement your program in C/C++ using OpenCV and the codebase provided. You are allowed to change the codebase, but the following steps cannot be changed: 1) the project must be configured using only the command *lab_config.sh*. 2) the project must be compiled using only the command make from the build folder. 3) the project must be run using the specifications from the previous section, including the name of the project. As the compilation and execution is automated, failure to meet these requirements will result in a grade of 0. **The project must compile and run as specified above on the lab machines.**

Unlike the programming assignments, the project you have to make a zip file of the entire project and submit it using the EAS system under "**Project 1**". The due-date is **March 8th, 5pm.** The deadline is soft in that we accept late submissions at a penalty of 20% (of the total marks) for every 24-hour period.

If you have problem compiling your code in the lab, first revert to the codebase and if that does not work as intended please send me an e-mail containing the relevant details.

This project is individual.

Unlike the exercises, the projects are fairly complex and require a lot of work, therefore please start working on it early.

Originality and Plagiarism

Binary Imager segmentation algorithm using min-cuts is a very well-known algorithm, whose code undoubtedly can be found on the internet. This is a reminder to everyone that everything you submit must be your original work and you are expected to be able to explain the code in detail if such a request is made by the teaching assistants or the instructor.

Evaluation and Testing

The code is compiled and ran automatically on a number of images. However, the evaluation will be done by a human visually. You will receive a code of 0 if the code does not build and execute as specified. You may be asked to explain your code to the teaching assistant or the instructor. This may lead to a decrease in your overall grade for this exercise.

References

[1] Eriksson, A. P., Barr, O., & Astrom, K. (2006). Image segmentation using minimal graph cuts.
[2] Boykov, Y., & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, *26*(9), 1124-1137.
[3] http://www.coe.utah.edu/~cs7640/readings/graph_cuts_intro.pdf