Project 2: Local and Global Segmentation Algorithms

1 Introduction

This project is concerned with a simple example of image segmentation, where we are given an image with a bright object against a dark background and wish to automatically determine, for each pixel, whether it is part of the object or the background. This seemingly simple task is actually quite challenging. You will experiment both with sythetic and real data to get a sense of the challenges and trade-offs involved.

This assignment should be done individually. You can use any language you want **as long as it allows you to run the graph cuts library (see below)**. We suggest that you use C++ or Matlab, otherwise you will need to speak with Devin. For the purposes of your experiments, you may convert the images we provide into any format that you find convenient to read (PNM, for example, is a simple file format).

However, unlike the previous assignment you do not need to hand in any code, just the results of your experiments (including images and charts), and a description of your algorithms and findings.

2 Assignment

We will focus on a synthetic image we have provided for you, because this image has a known "ground truth" segmentation, which we have also provided. The image was created by making the object pixels intensity 200 and the background pixels intensity 100, and then adding noise. You will write code to compute a segmentation and compare it against the ground truth.

2.1 Local algorithms

A local algorithm has three stages: a spatial averaging stage, where the intensities of neighboring pixels are used to compute a new intensity value (for example, by weighted averaging); a thresholding phase, where this value is thresholded to determine whether the pixel is foreground or background; and a cleanup phase, where the resulting binary image is post-processed to obtain better accuracy.

For the spatial averaging phase, you should implement (1) smoothing by convolution with a Gaussian, and (2) median filtering, where we compute the median local value. Note that both of these methods have as a parameter the size of the local region.

For the thresholding phase, you should analyze the histogram of image intensities, with the knowledge that there are two main peaks. As an example, you could fit the histogram with two Gaussians, or you could do something more simplistic.

For the cleanup phase, you will rely on morphological filters, which are local operations on binary images. The fundamental operations are *dilation* and *erosion*. In dilation, if any neighborhood pixel is foreground, it becomes foreground. In erosion, if any neighborhood pixel is background, it becomes background. You can choose the neighborhood, though in practice usually relatively small neighborhoods are used. It is common to use dilation followed by erosion.

2.1.1 What do turn in

Your goal is to create the best local thresholding algorithm that you can. You should create a version that uses gaussian smoothing and one that uses median filters. For each of these two versions, you should run it with and without a cleanup step.

Thus you will create 4 different algorithms (gaussian versus median, and cleanup versus no cleanup). Note that we have not specified the parameters to use (such the size of the local region), nor the thresholding method, nor the precise morphological cleanup operation. You should experiment with these and get the best results that you can.

2.2 Global algorithm

You should also solve this problem using graph cuts to minimize the global energy. You will be using the Ising model, which can be solved exactly. You will need to design the appropriate data cost function, which specifies the cost for an individual pixel to be labeled as foreground versus background, based on its observed intensity.

2.2.1 Graph cuts code

For graph cuts energy minimization, we recommend you use Olga Veksler's C++ code, which is available at http://www.csd.uwo.ca/ olga/code.html. If you are not familiar with either C or Matlab you will need to speak to Devin about how to best run the code.

The latest release of Olga's library includes an example program which demonstrates how to set up an energy minimization problem on a grid graph.

2.3 Comparison against ground truth

Using the ground truth, find the setting of parameters for each algorithm that gives you the best results in terms of overall accuracy. The obvious way to measure accuracy is to count agreement with the ground truth; however, as you will discover, pixels on the border of the object are particularly difficult. In addition, these pixels are vital for most applications. You should therefore ignore pixels where all 4 neighbors have the same value in the ground truth.

For your most accurate algorithm, begin with the parameters that give you the best answer and pick one important parameter. Plot the accuracy as a function of the value of that parameter, centered at its optimal value, while holding all other parameters constant.

2.4 Experimentation with real data

We have also provided you with several real images. For these images, try your favorite local segmentation algorithm, and also run graph cuts. Hand in the results that seem most accuate to you, both for local methods and for graph cuts. Inlude a description of the exact algorithms and parameters you used to generate these results.

2.5 Submission format

You should not submit code, just a writeup containing the description of your exact algorithms, the resulting images and plots, and a brief summary of your conclusions.