Computer Vision Mini Project 2:

GrabCut*Implementation in Processing† for interactive image segmentation

Karan Daei-Mojdehi
Course Instructor: Dr. Ulas Bagci
Department of Computer Science, University of Central Florida

December 11, 2015

Outline

In this second mini project, followed by a region growing segmentation as the first mini-project, a very well known and successful region growing segmentation algorithm named GrabCut[1] is implemented in Processing language. User can easily interact with the program by selecting a region around an object and the program will segment the area in image which believes is the object.All of the program is written in Java and can be transferred to an android phone using Processing's android mode.

Keywords:Computer Vision, Image Segmentation, GrabCut, Processing Language, Android

*[1]
† www.processing.org

Introduction

As asserted in the overview GrabCut has been one of the successful image segmentation algorithms in the past decade. For more information on description of image segmentation reader is referred to report of previous mini project. GrabCut algorithm exploits two novelties that its predecessor methods lacked: first one is that it incorporates an iterative estimation of segmentation and enhances results of previous iteration in each step, and second novelty is incomplete labeling which minimizes required user interaction: user simply needs to draw a rectangle which completely includes the object and will not suffer if parts of background are inside this rectangle. We will discuss the algorithm in detail in next section. Implementing this powerful algorithm in Processing allows use to easily upload it to android phone or even use it in web pages with new JavaScript mode of Processing. The only none-core java library that was used for this implementation was Boykov's Graphcut java implementation. I will go through details of this graphcut concept as well as creating its library for android mode in Boykov Grphcut section**??**.

Algorithm: GrabCut Segmentation

First, I will give a short and abstract explanation of Grabcut, then I will go through details of each step in the relevant subsections. GrabCut segmentation is an iterative segmentation algorithm that incorporates both Expectation Maximization (EM) and Graph based segmentation method in its procedure. At each iteration, EM step tries to fit different regions of image pixels (based on three channel color) to different multivariate Gaussian Mixture Models (GMMs). Technically there are two sets of GMMs: one for back ground and the other one for foreground. After the EM step, a Graphcut based method is used to segment the image. In the graphcut phase, unary terms of the edges in the graph (i.e. edges connecting pixels as graph nodes to the sink (foreground) and source(background) nodes of the graph), are assigned based on the parameters of the GMM that the relevant pixel most likely belongs to. After the graph cut, pixels mats are updated and pixel members of GMMs are altered as a result. The new members of GMMs are used in EM step of next iteration. Now let us go through details of these steps:

I Expectation Maximization (EM)

Expectation Maximization is a two step algorithm for fitting data to clusters. These two steps are called E-step (Expectation) and M-step (Maximization). In the E-step, each data point is assigned to the cluster to which it most likely belongs. In the M-step, parameters of clusters are updated based on the new members of data points that are assigned to them. These two steps follow each other until the algorithm converges, i.e. until no members are swapped between clusters and therefore cluster parameters stagnate. EM algorithm have shown to be very successful and time efficient in machine learning literature.

II Multivariate Gaussian Mixture Models (GMM)

Mixture models are used to model data distributions that is believed to be a combination of simpler models. Each data point will have a partial membership to these models based on how probable that data point is if it were to belong to that model. This membership is usually annotated by π_c and sums to 1 over all the Gaussian Mixture models. Multivariate Mixture Models are used to model distribution which have more that one dimension (in our case, each color channel of pixel will introduce a dimension, therefore dimension is 3). The probability of a mixture model (c) describing a data point is as follow:

$$P(x|c) = \pi_c \ \mathcal{N}(x; \mu_c, \Sigma_c) \tag{1}$$

Where π_c is probability of belonging to cluster c and is the proportion of model c members to sum of all data points and $\mathcal{N}(x; \mu_c, \Sigma_c)$ is a multivariate normal distribution with mean μ_c and co-variance matrix Σ_c and is calculated by the following equation:

$$\mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}} |\Sigma|^{\frac{-1}{2}} \exp(\frac{-1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)) \tag{2}$$

Together with the EM algorithm, this pair is used to find the separate Multivariate GMMs that best describe separate sets of foreground and background pixels.

III Graphcut Segmentation

Graphcut segmentation is one of the methods that minimizes an energy function by translating it to a graph problem. The energy function is defined on pixels' euclidean distance in color space to

foreground and background (unary term) as their distance to their neighbor pixels (pairwise term which is in close relation with image gradient in different color channels. This mapping is done by selecting image pixels as nodes of the graph. Foreground and background are also introduced as nodes (conventionally named source and sink nodes) in this graph. Unary terms are taken into account by connecting each pixel as a node to foreground and background nodes with an edge cost equal to the distance. Pairwise terms are considered as edges between pixels nodes. In the next step a minimum cut is executed on the graph which is equivalent to minimizing the energy function. This cut needs to separate source and sink nodes thoroughly. After the cut, pixels that are still connected to source are considered foreground and rest of them are labeled as background.

IV Boykov Mincut/Maxflow Graphcut

This is modification of Mincut/Maxflow algorithms that Boykov came up with when he was doing a survey of graphcut algorithms in 2004 [2]. The idea behind Mincut/Maxflow algorithms is that running maximum flow through edges of a graph (assuming weights are maximum available flow in an edge) is equivalent to achieving a minimum cost cut on the graph edges where no more flow is left. Details of this algorithm can be found in [2]. I have also implemented this paper in python (attached as Boykov.py) which is well commented for further demonstration of the algorithm. In order to use this algorithm in Processing, I downloaded its Java implementation from fiji's git and used android sdk along with processing android library template in eclipse to create an appropriate library for exploiting this library in Processing.

V Putting it All Together

GrabCut algorithm works in this order: First the user will select a region of image as the input, pixels outside this region are selected as background and pixels inside it are given an unknown status. Then, two sets of multivariate GMMs (each of them with k=5 components) are created: one for background and the other for background. A clustering algorithm is used to initiate these 2k mixture components with their relevant sets of pixels (foreground GMMs are seeded from pixels marked with unknown status) to achieve a minimum variance in each component. Then in an EM manner, after updating parameters of each component with its initiated set of members, each pixel is assigned to its most relevant Gaussian component (in the same region, i.e. pixels in the

foreground sets of Gaussian component are only assigned to one of foreground components) by considering the following parameter (r_{ic}:

$$r_{ic} = \frac{\pi_c . \mathcal{N}(x_i; \mu_c, \Sigma_c)}{\Sigma_{c'} \pi'_c . \mathcal{N}(x_i; \mu'_c, \Sigma'_c)} \tag{3}$$

where x_i is the color vector of the relevant pixel (E-step done). After the E-step, each GMMs parameters ($\mu_c$andΣ_c) are updated with respect to the new pixels assigned to them to complete the M-step. As the final step of current iteration, a graph with same number of nodes as number of image pixels plus two (sink and source nodes) is initialized. As for the unary terms, each pixels edge weight to source and sink is logarithm likelihood of that pixels membership to set of foreground (edge weight to source) or background (edge weight to sink) set of GMMs. This is calculated by the following equation:

$$EdgeWeight(set)_i = -log(\Sigma_{c \in set} p(x_i|c)) \tag{4}$$

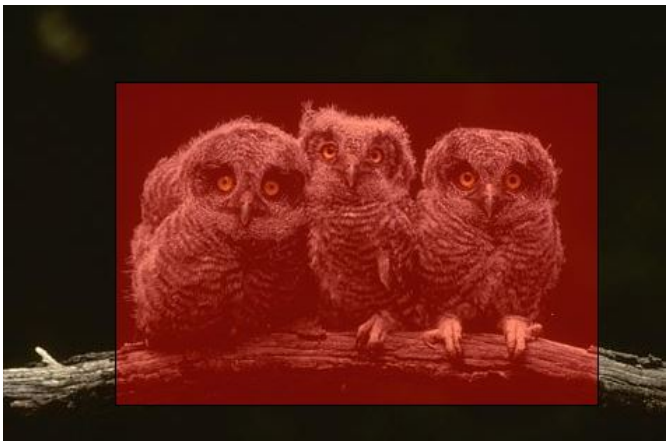
in which $p(x_i|c)$ is computed using equation 1. Pairwise terms are modeled by setting the edge weight between neighbor pixels as follow:

$$EdgeWeight(x_n, x_m) = \exp(\frac{|x_n - x_m|^2}{2\sigma^2}) \tag{5}$$

where $|x_n - x_m|^2$ is the euclidean distance in color space between pixels and σ is the expected value of deviation in neighbors of pixel. After setting unary and pairwise terms, Boykov Min-Cut/MaxFlow graph cut algorithm is executed to split pixels into background and foreground sets. This split is maintained in the next iteration to specify which pixles belong to which set(foreground or background) of GMMs. Iterations continue until algorithm converges which usually happens in less than five iterations. The attached codes are full of comment and go through micro details of each step.

Results

Here is a sample result from using the algorithm, on computer. the transparent red rectangle in left image shows user selected area as approximate region for object and right image shows segmentation results. Due to time limit no further assessments on the algorithm were possible.



(a) User Selection

(b) Segmentation Results

References

[1] Rother, Carsten, Vladimir Kolmogorov, and Andrew Blake(2004). "Grabcut: Interactive foreground extraction using iterated graph cuts", *ACM Transactions on Graphics* (TOG) 23, no. 3 (2004): 309-314.

[2] Boykov, Yuri, and Vladimir Kolmogorov(2004). "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision." *Pattern Analysis and Machine Intelligence, IEEE Transactions* on 26.9 (2004): 1124–1137.