Automatic Image Segmentation using Graph Cuts

CSC14801 Project (Mentor: Dr. S. Mukhopadhyay)

Madhur Chauhan

Indian Institute of Technology (Indian School of Mines), Dhanbad

April, 2018

Contents

Example

Let's start discussion by using an example which will make us walk through whole process.

Outline

By visual inspection we can see that there are 5 components in the image.

Components

- Hair
- Skin
- Background
- T-Shirt
- Frock/Gown

Sample image

Outline

What to choose? User Input?

It is clear that out of the 5 components we have to choose **Hair, Skin, T-Shirt, Frock** as the foreground and rest as background. But since user is not involved, computer doesn't know what to segment.

Idea

Since image can be broken down into components we can let computer cluster "similar" components. That way computer will segment the image without knowing what is desired area that we want. This is the concept of "**Automatic Segmentation**".

Example

Let's say there are thousands of images and will you still as a user want to select each and every component as foreground and background? Do you have time?

Outline

Need for Clustering

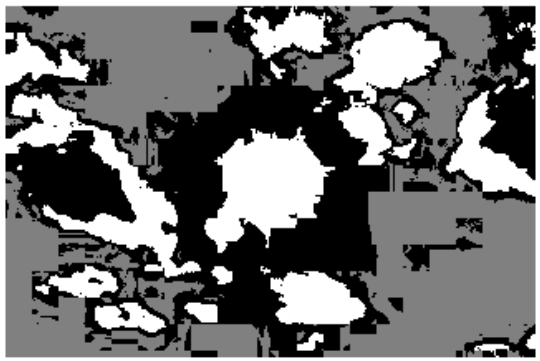
There is need for k-means clustering in finding the components which is discussed in previous slides. Clustering provides with centers of the coressponding components which is used for initialising the centres and weights of Gaussians of Gaussian Mixture Model.
Random initialisation of centres and weights will produce disastrous results and will lead to a very poor segmented image.
Comparison of results without using k-means is shown in next slide.

Without k-means

Original Image

2 Components

3 Components without kmeans



3 Components with kmeans

Outline

Need for GMM

Question

When clustering provides the necessary components why go for GMMs?

Answer

Refering to example picture on the slide, k-means provides with components but it doesn't take into picture the close relation between the components. Maybe the background components have similar shade and different from that of the foreground components. This gets taken care by 2 GMMs.

Example

Background in this picture has a shade with very high contrasting difference from that of foreground. It can be seen that contrast of components of foreground is similar to that of the background.

Outline

Estimation-maximization algorithm

Expectation-maximization is a well-founded statistical algorithm to get around this problem of finding which point belong to which latent component (i.e. gaussian) an iterative process. First one assumes random components (randomly centered on data points, learned from k-means, or even just normally distributed around the origin) and computes for each point a probability of being generated by each component of the model. Then, one tweaks the parameters to maximize the likelihood of the data given those assignments. Repeating this process is guaranteed to always converge to a local optimum. Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters comprising the means and covariances of the components and the mixing coefficients).

Steps

1. Initialize the means μ_j, covariances Σ_j and mixing coefficients π_j, and evaluate the initial value of the log likelihood.
2. E step. Evaluate the responsibilities using the current parameter values.

$$\gamma_j(x) = \frac{\pi_k N(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x|\mu_j, \Sigma_k)}$$

Steps (continued)

3. M step: Re-estimate the parameters using the current responsibilities.

$$\mu_j = \frac{\sum_{n=1}^{N} \gamma(x_n) X_n}{\sum_{n=1}^{N} \gamma_j(x_n)} \quad , \quad \pi_j = \frac{1}{N} \sum_{n=1}^{N} \gamma_j(x_n)$$

$$\Sigma_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n)(x_n - \mu_j)(x_n - \mu_j)^T}{\sum_{n=1}^{N} \gamma_j(x_n)}$$

4. Evaluate log likelihood.

$$ln(p(X|\mu, \Sigma, \pi)) = \sum_{n=1}^{N} ln(\sum_{k=1}^{K} \pi_k N(x_n|\mu k, \Sigma_k))$$

5. If there is no convergence then repeat step 2. (Take convergence limit as 10^{-6})

Outline

Outcome of EM algorithm

After the convergence of the EM algorithm. We can hopefully get probabilities of the pixel fitting either of the GMM. The result with be a matrix with 2 colums and length equal to number of pixels in the image. This way we can use these probabilities for graph cut.

Outline

Why graph cuts?

The probabilities calculated from the GMM were a measure for how pixels are associated with the foreground or background components.
This doesn't give the idea about neightbourhood points from the image. If the pixel belong to the foreground then its neighbours (atleast one) must belong to the foreground. This way using graph cuts the **local optimal** value of segmentation can be achieved as the neighbourhood gets penalized if no neighbour match.
Thus we can minimize errors of segmentation. This also gives a **continous boundary** of image segments

Example

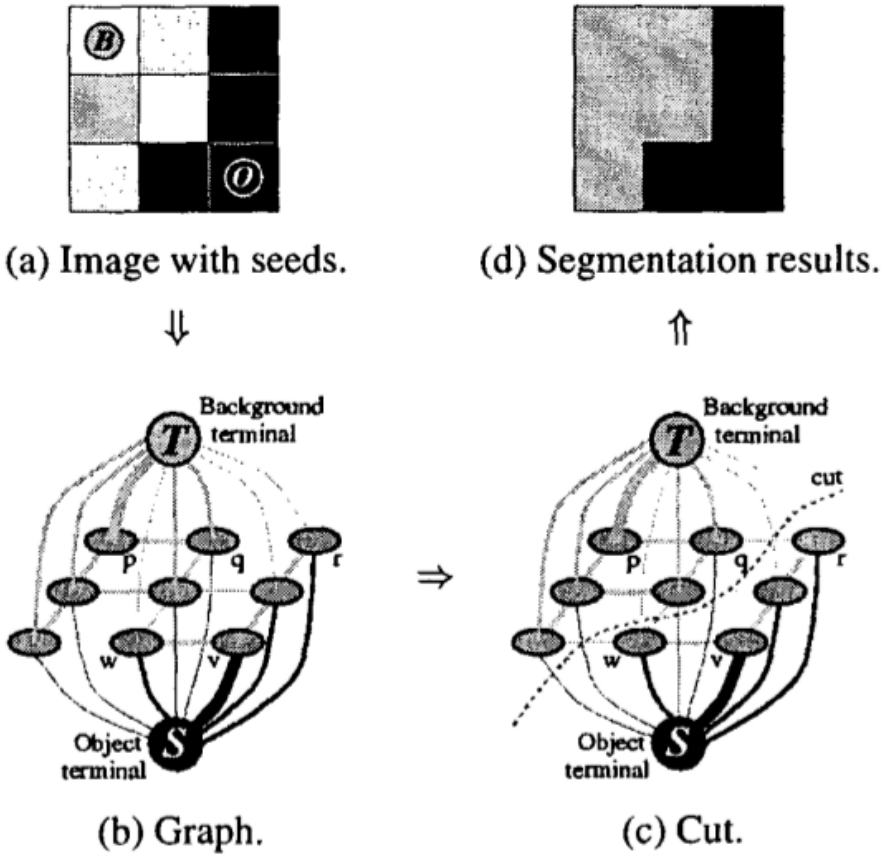


Figure: Segmentation using graph cuts (Boykov & Jolly, 2001)[1]

Outline

Representing graph

An image of dimension ($height$, $width$) can be represented as a graph. For color images 3 color channels RGB can be represented as tuple which can be made to work with graph attributes. These tuple carry the node attributes. For grayscale image intensity channel can be used as a node attribute. The graph is undirected i.e. the edges can be traversed in both directions.
The edges can be represented as adjacency matrix because every pixel except borders will have 8 neighbours. Value of bordering pixel can be a sentinel value denoting absence.

1. **Nodes**: Total number of pixels in the graph contribute to the total number of nodes in the graph. Therefore there will be total of $height * width$ nodes in the graph.
2. **Edges**: These can be unweighted or weighted based on the application. In this project we have used weighted edges representing the capacity of the flow across it. The 8 - neighbourhood system of the nodes adds 8 edges to the graph between the node and its 8 neighbour. Though it can reduced to 4 neighbourhood system also but loses accuracy.

Outline

Push-relabel for Graph Cuts

Think of the push-relabel algorithm as source s impatiently sending flow to nodes downstream from it, which in turn try to send flow to nodes downstream from them, until some of the nodes cannot send any more. They re-evaluate the situation. In particular, they re-evaluate what downstream means. Eventually they send the flow back to s. Lets see how. We are given a directed capacitated graph $G = (V, E)$ with source s and sink t, and want to find a max $s - t$ flow. Lets imagine all nodes are reachable from s, and can reach t. (All other nodes can be deleted.) For brevity, we denote the directed edges $(u, v) \in E$ merely as uv.

Asymptotics

Time Complexity is : $O(V^2E)$

Steps

1. **Initialization**
   We start off by saturating all the edges sv coming out the source s. In other words, we send $c(sv)$ flow on each such edge $sv \in E$. This gives a valid preflow f_0. Moreover, the residual graph with respect to this preflow f_0 is almost the same as the original flow network G, except all the arcs $sv \in E$ have been replaced by arcs vs. For each node v with $sv \in E$ (and hence $vs \in G_{f_0}$), we have excess $e_{f_0}(v) = c(sv)$.

Steps (continued)

2. **Main Loop**
   While there is an active node u, do one of the following for u.
   1. **Push**(u): If $\exists v$ with admissible arc $uv \in E_f$, then send flow $\delta := min(c_f(uv), e_f(u))$ from u to v. Note that this causes excess $e_f(u)$ to fall by δ, and excess $e_f(v)$ to increase by δ. If $\delta = c_f(uv)$, this is called a saturating push, else it is an non-saturating push.
   2. **Relabel** (u): If for all arcs uv out of u, either uv is not in E_f or $d(v) \geq d(u)$, then set

$$d(u) = 1 + \min_{v:uv \in E_f} d(v)$$

Outline

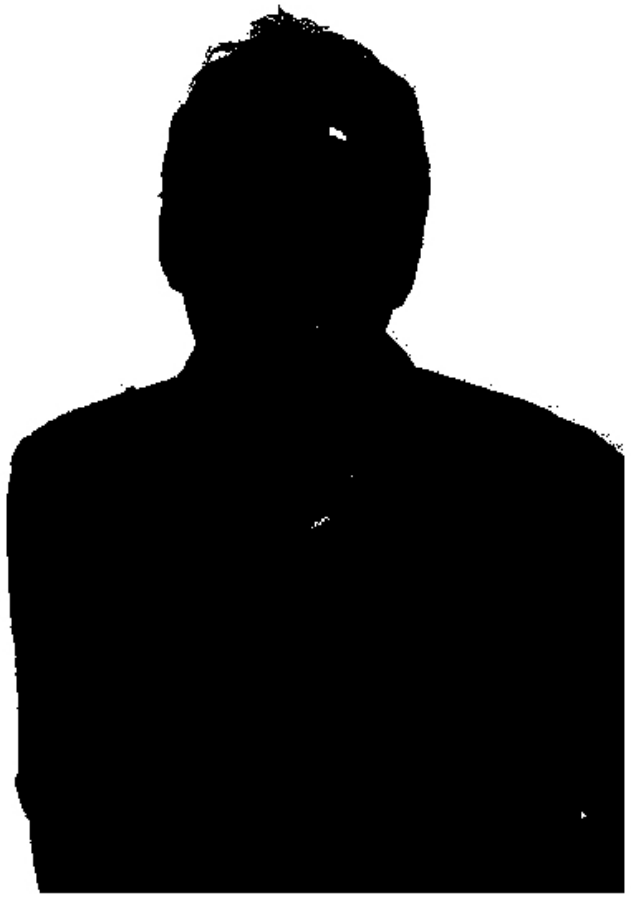
Outcome

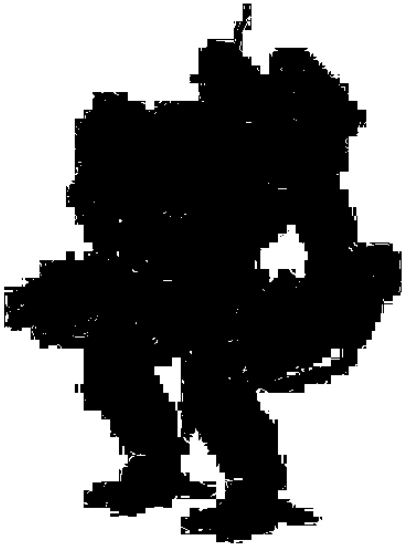
We have to find min cut of the graph after running max flow algorithm accoring to max flow-min cut theorum.

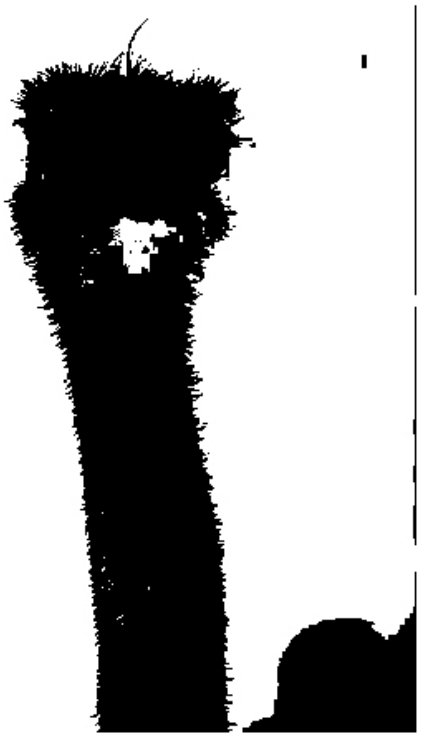
Max Flow - Min Cut theorum

In any network, the value of max flow equals capacity of min cut.

After running min cut, the partition of the nodes (pixels) belonging to either foreground or background is done. This gives final result.







References

1. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images, Boykov & Jolly, 2001
2. Texture aware image segmentation using graph cuts and active contours by Hailing Zhou, Jianmin Zheng, Lei Wei from ELSEVIER, Pattern Recognition 2013.
3. Report on "Automatic Image Segmentation using Graph Cuts" by Madhur Chauhan, 2018 (github.com/madhur4127)