IMAGE SEGMENTATION USING GRAB CUT

REPORT

ADITYA GOLATKAR 14B030009
RUDRAJIT DAS 140020012

INTRODUCTION:

This project addresses the problem of efficient, interactive extraction of a foreground object in a complex environment whose background cannot be trivially subtracted. **The aim is to achieve high performance at the cost of only modest interactive effort on the part of the user**. The novelty of this approach lies in the handling of segmentation. There are two enhancements to the usual graph cuts mechanism: "**iterative estimation**" and "**incomplete labelling**" which together allow a considerably reduced degree of user interaction for a given quality of result. This allows Grab Cut to put a light load on the user, whose interaction consists simply of dragging a rectangle around the desired object. In doing so, **the user is indicating only a region of the background,** and is free of any need to mark a foreground region. Note that **Grab Cut produces hard segmentation** of the images.

WHY USE GRAB CUT INSTEAD OF THE STANDARD GRAPH CUT?

First, the **monochrome image model is replaced for colour by a Gaussian Mixture Model (GMM)** in place of histograms. Secondly, the one-shot minimum cut estimation algorithm is replaced by a more powerful, **iterative procedure that alternates between estimation and parameter learning**. This iterative procedure results in better segmentation than the plain old graph cut algorithm. Thirdly, the demands on the interactive user are relaxed by **allowing incomplete**

labelling - **the user specifies only the background**(TB) for the trimap(comprises of the background region TB, the foreground region TF and the unknown region TU), and this can be done simply by placing a rectangle around the object.

ALGORITHM:

Initialisation

- User initialises trimap T by supplying only T_B. The foreground is set to $T_F = \emptyset$; $T_U = \overline{T}_B$, complement of the background.
- Initialise $\alpha_n = 0$ for $n \in T_B$ and $\alpha_n = 1$ for $n \in T_U$.
- Background and foreground GMMs initialised from sets $\alpha_n = 0$ and $\alpha_n = 1$ respectively.

Iterative minimisation

1. *Assign GMM components to pixels:* for each n in T_U,
$$k_n := \arg\min_{k_n} D_n(\alpha_n, k_n, \theta, z_n).$$
2. *Learn GMM parameters* from data $\mathbf{z}$:
$$\underline{\theta} := \arg\min_{\underline{\theta}} U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z})$$
3. *Estimate segmentation:* use min cut to solve:
$$\min_{\{\alpha_n:\, n \in T_U\}} \min_{\mathbf{k}} \mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}).$$
4. Repeat from step 1, until convergence.

Some Theory and Implementation Details -

Each GMM, **one for the background and one for the foreground**, is taken to be a full-covariance Gaussian mixture with **5** components. In order to deal with the GMM tractably, an additional vector **k = {k1,...,kn,...,kN}** is introduced, with kn ∈ {1,...5}, assigning, to each pixel, a unique GMM component, one component either from the background or the foreground model, according as αn = 0 or 1. The **Gibbs energy E** for segmentation now becomes:

$$\mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) + V(\underline{\alpha}, \mathbf{z})$$

The **data term U** is defined as:

$$U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = \sum_n D(\alpha_n, k_n, \underline{\theta}, z_n)$$

$$D(\alpha_n, k_n, \underline{\theta}, z_n) = -\log \pi(\alpha_n, k_n) + \frac{1}{2} \log \det \Sigma(\alpha_n, k_n) + \frac{1}{2}[z_n - \mu(\alpha_n, k_n)]^\top \Sigma(\alpha_n, k_n)^{-1} [z_n - \mu(\alpha_n, k_n)].$$

Here, **π (·)** are the mixture weighting coefficients and **μ(α,k), Σ(α,k),** α = 0,1, k = 1...5 are the mean vectors and the covariance matrices for the various Gaussians of the background and foreground region.

The **smoothness term V** is:

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n) \in \mathbf{C}} [\alpha_n \neq \alpha_m] \exp -\beta \|z_m - z_n\|^2 .$$

Set the constant β as:

$$\beta = \left(2 \left\langle (z_m - z_n)^2 \right\rangle \right)^{-1}$$

where <.> denotes expectation over the image sample.

Step 1 of the iterative minimization is done by **enumeration of the kn values for each pixel n**, based on it's label α. We set kn = K ∈ {1,...5} , such that the **likelihood is maximum for the Kth Gaussian** (or the negative log-likelihood is minimum for the Kth Gaussian as stated in the algorithm).
Step 2 involves **Gaussian parameter estimation**, as follows : for a given GMM component k in, say, the foreground model, the subset of pixels F(k) = {zn : kn = k and αn = 1} is defined. The mean **μ(α,k)** and covariance **Σ(α,k)** are estimated as the **sample mean** and **covariance** of pixel values in F(k) (**maximum likelihood estimation**) and the weights as **π(α,k) = |F(k)| / ∑ |F(k)|**, where |S| denotes the size of a set S.
Finally **Step 3** is a global optimization, using **minimum cut**. We construct a **Graph(Random Field)** using the parameters obtained

previously as follows : the nodes/vertices of the graph consist of all the pixels in the image as well as 2 auxiliary nodes say, '**s'** for the **foreground** and '**t'** for the **background**. The **edges between 2 nodes formed by 2 pixels** is given by **the pairwise potential term V** as defined earlier. We have used a **four neighbourhood system** in our implementation. Also, we add an **edge between a node formed by a pixel with 's' and 't' with weight equal to the unary potential, i.e. the likelihood term D** for that node as defined earlier (similar to what we do for the standard graph cut algorithm). This completes the construction of our weighted undirected graph on which we apply the min-cut algorithm to obtain the updated α's, i.e. the new labelling of the pixels. The min-cut algorithm used is **Boykov-Kolmogorov max-flow algorithm**.

The proposed algorithm guarantees proper convergence. **E decreases monotonically**, as is illustrated in the plots below. Thus, **the algorithm is guaranteed to converge at least to a local minimum of E**. When E ceases to decrease significantly, we terminate the iteration.

RESULTS:

NON MEDICAL IMAGES-

INPUT:

gamma = 20

F = 0

Gmm_comps = 5

SEGMENTED IMAGE:

INPUT:

gamma = 20

F = 0

Gmm_comps = 5

SEGMENTED IMAGE:

MEDICAL IMAGES-

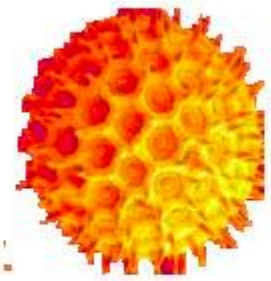
INPUT:1

gamma = 10

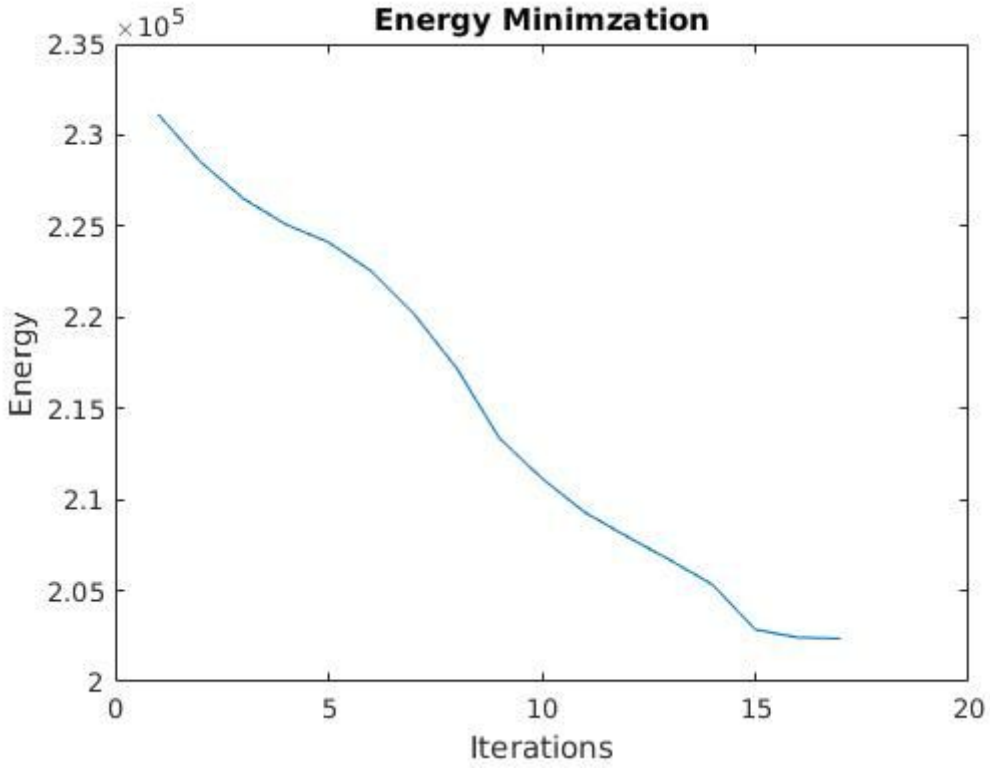
F = 0.1

Gmm_comps = 5

SEGMENTED IMAGE:



ENERGY MINIMIZATION:

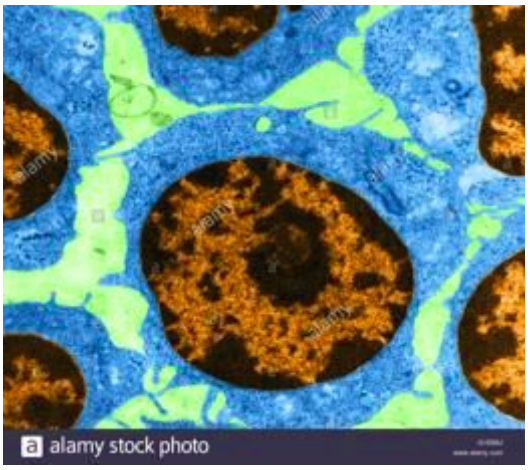


INPUT:2

gamma = 20

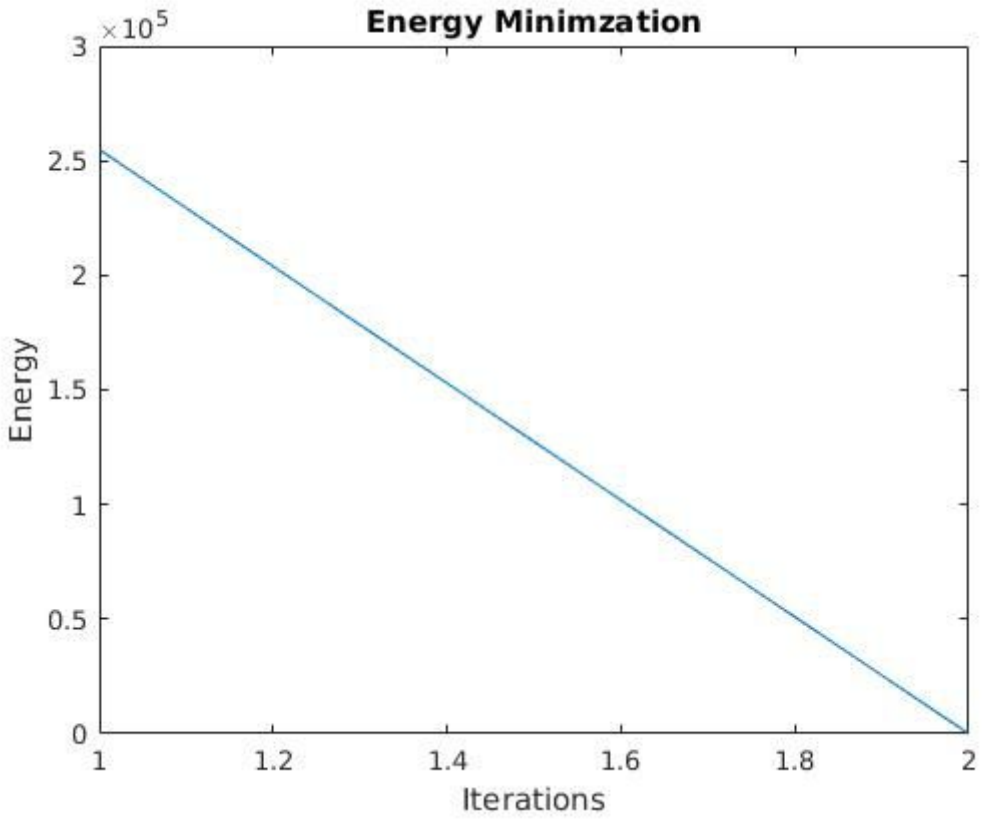
F = 0.1

Gmm_comps = 5



SEGMENTED IMAGE:

ENERGY MINIMIZATION:

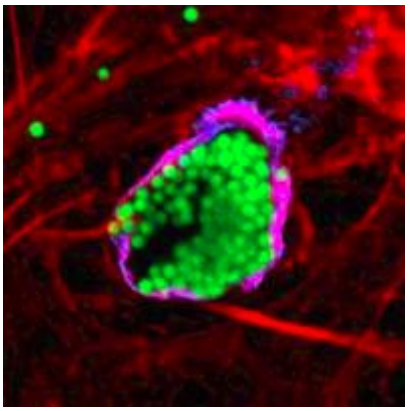


INPUT:3

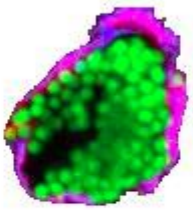
gamma = 20

F = 0.1

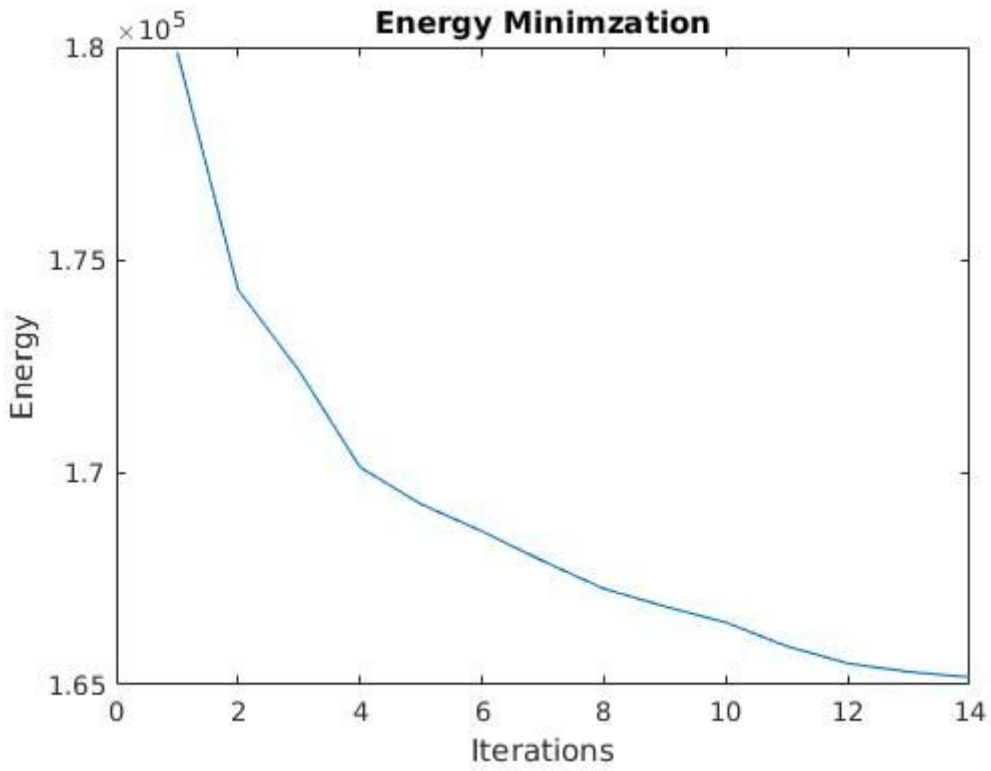
Gmm_comps = 5



SEGMENTED IMAGE:



ENERGY MINIMIZATION:

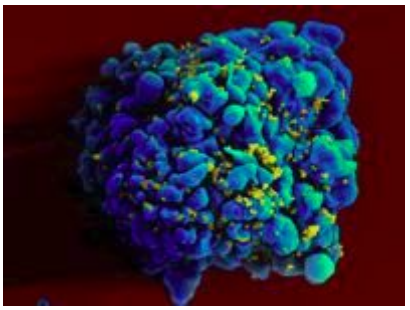


INPUT:4

gamma = 20

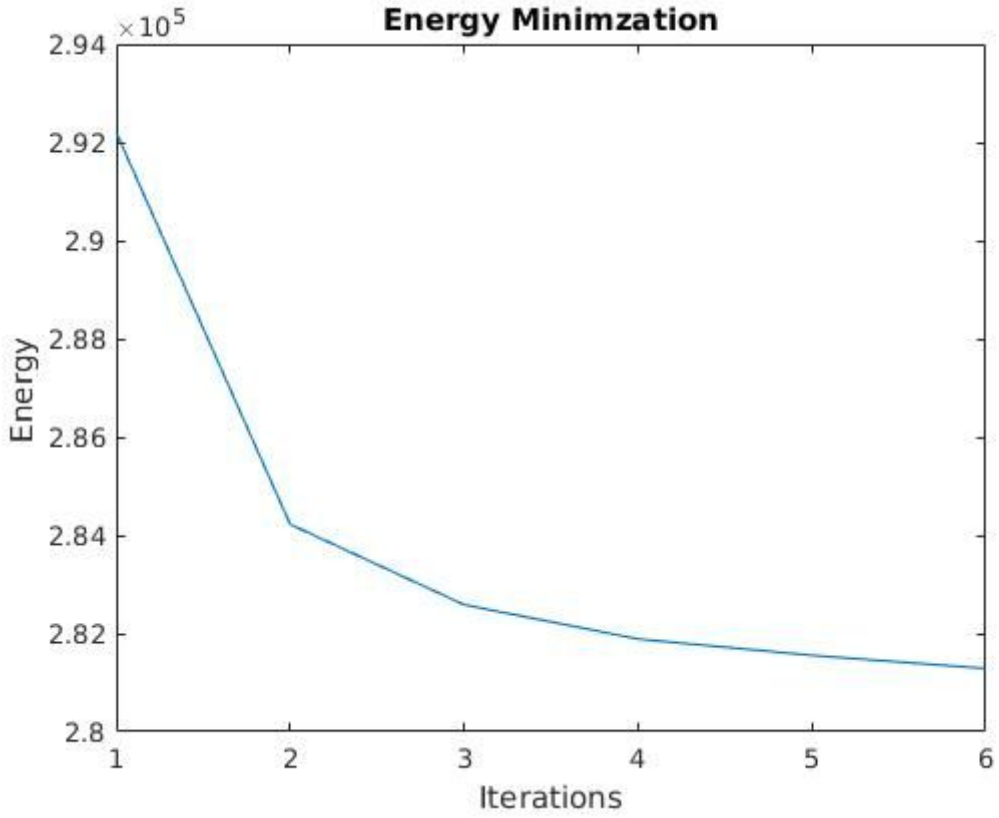
F = 0.1

Gmm_comps = 5



SEGMENTED IMAGE:

ENERGY MINIMIZATION:

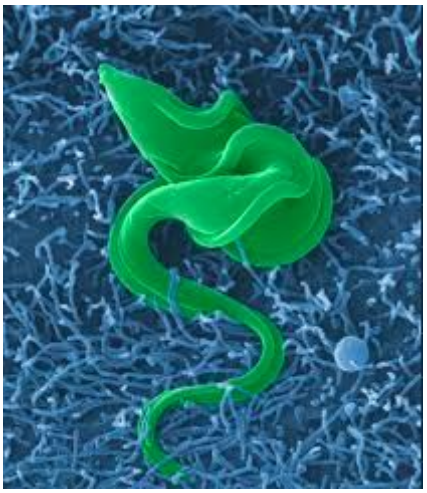


INPUT:5

gamma = 10

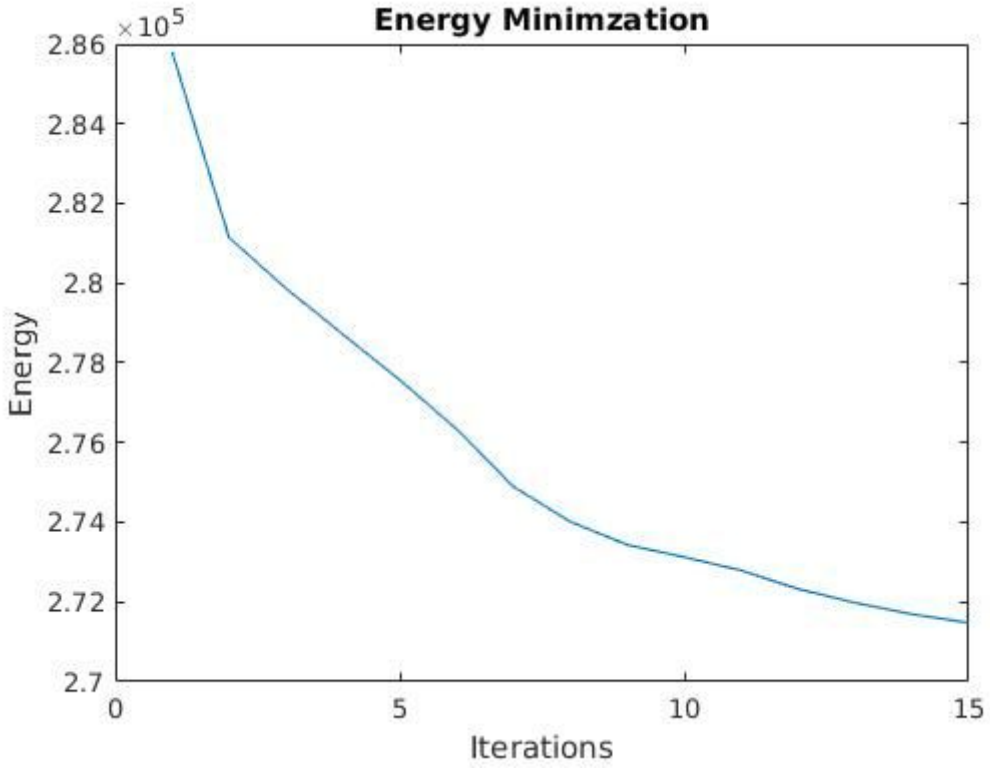
F = 0

Gmm_comps = 5



SEGMENTED IMAGE:

ENERGY MINIMIZATION:

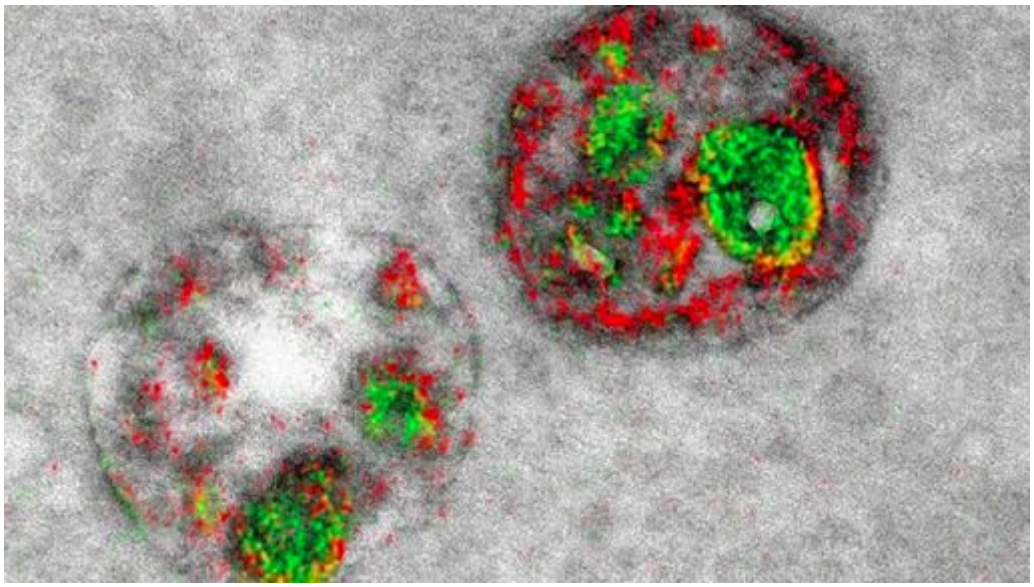


INPUT:6

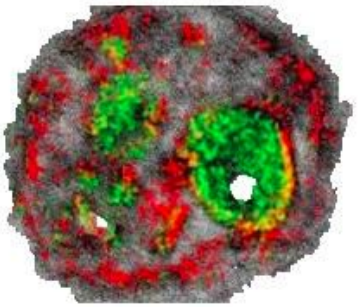
gamma = 18

F = 0.1

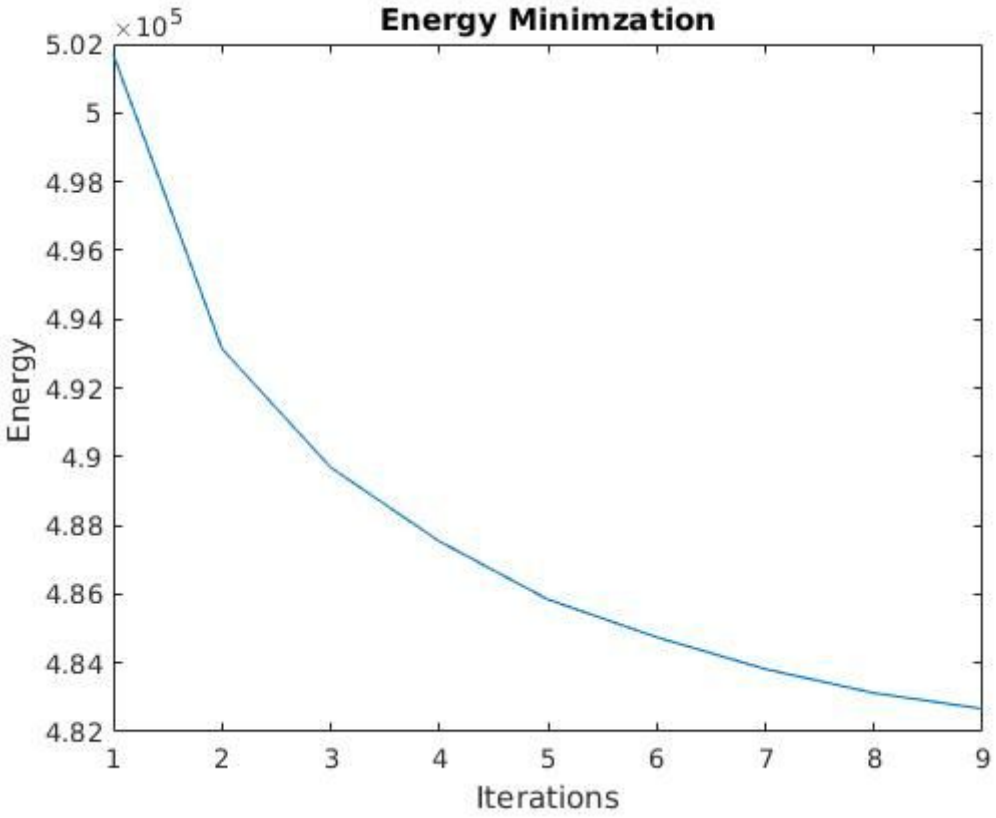
Gmm_comps = 6



SEGMENTED IMAGE:



ENERGY MINIMIZATION:



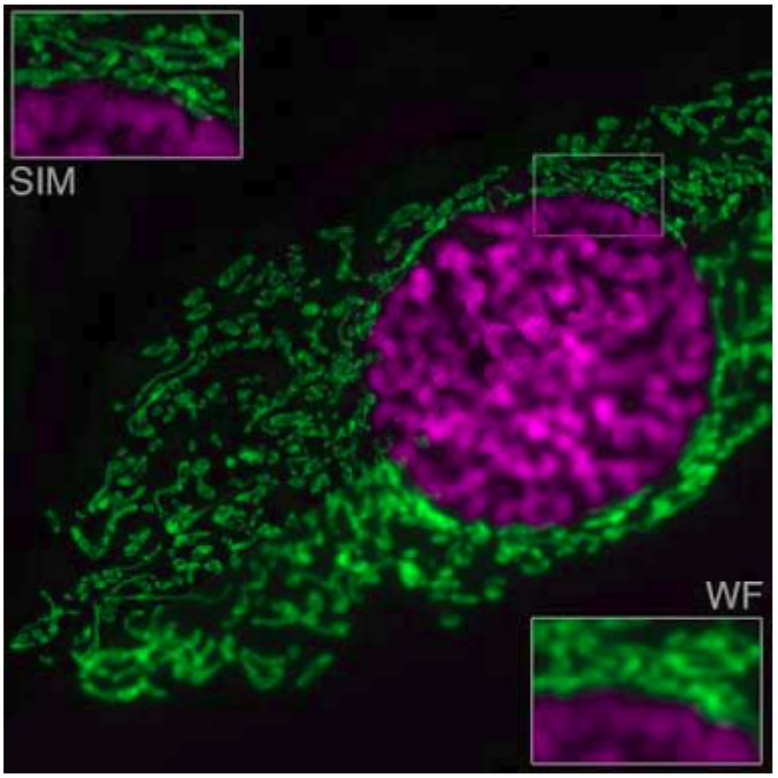
INPUT:7

gamma = 10

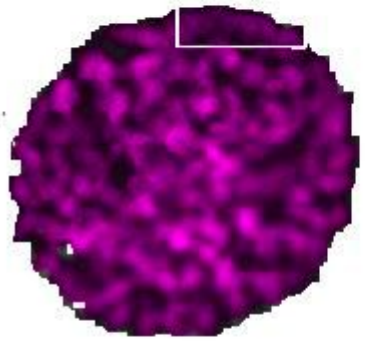
F = 0.1

Gmm_comps = 7

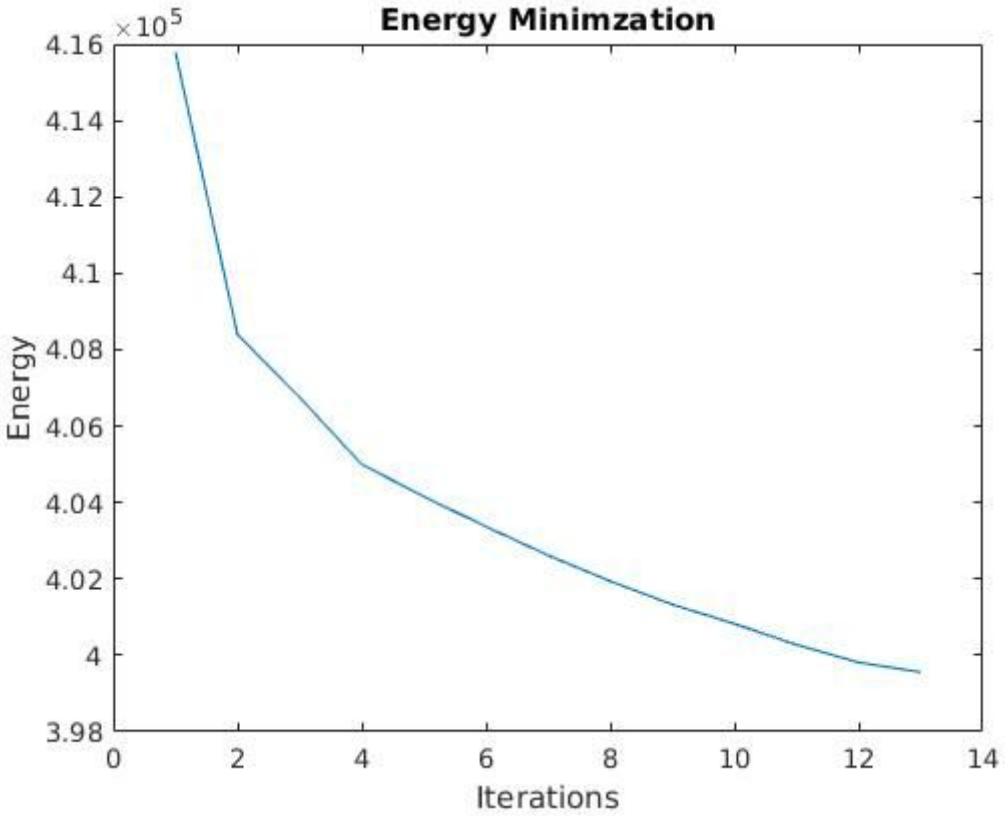
O = 13



SEGMENTED IMAGE:



ENERGY MINIMIZATION:



RUNNING THE CODE:

To run the code just call the function grab_cut with correct parameters.
First parameter is the image.
Second parameter is gamma.(Ideally in the range 10-20)
Third parameter is f which is user to prevent the covariance matrix from becoming rank deficient.(0 when the background is textured, 0.1 when the background is plain colored)
Fourth parameter is the gaussian mixture model components(Ideally in the range 5-7)

To run the code you will need the mex file. So if you are using a linux machine you can use the mex file which I am using. But for a non linux user the mex file needs to be generated from cpp file.
Mex requires a gcc version of 4.9.xx or lesser . So make sure you have that version.
For linux user use this to compile the mex file:

```
mex -v GCC='path/to/g++/bin/g++-4.7' path/to/somefile.cpp
```

CHALLENGES:

1. The images which the paper tests upon has a good texture in the background while the medical images generally have a constant color in the background as a result of which the covariance matrix becomes rank 1 matrix or NaN matrix. Hence to prevent this from happening corrective measures have been taken. To convert the rank 1 matrix to a full rank matrix, we just added a scaled identity matrix. In order to tackle the NaN problem we just stop the code when we encounter it, as the image is already segmented when the NaN error appears.

2.Also for the initial segmentation of the background and foreground we have used k-means algorithm. In the assignment since we had only 3 component of the GMM we used mean and variance to initially segment the image in 3 parts. But here since the minimum GMM components are 5 we thought it was better to use k-means.

ACKNOWLEDGEMENT:

We would like to thank University of Western Ontario for making the code for min-cut algorithm in C++ publically available and Andrew Delong for creating a Matlab wrapper so that the C++ code can be used. The link for the code : http://vision.csd.uwo.ca/code/
Note that all the files with their name beginning with bk_ are the files provided by the University. The main files are cpp files while bk_….m files are wrapper used to access the cpp files.