ASSIGNMENT -4 COMPUTER VISION

1.SLIC SUPERPIXELS:

Method:

1. Initialized the grid interval S to $\sqrt{N/k}$ where N is the number of pixels and K is the number of segments.

2.The clusters are centered at intervals of S using linspace and at the lowest gradient in each window of size 2S*2S.

3. Initialize Dorig (the distances between center of each cluster and pixel to vary large value

```
Dorig = 10000000000000000000*ones(sz(1),sz(2));
```

4. Then loop through every cluster and every pixel. For each pixel consider a region of size 2S*2S, and calculate the Dc(coordinate distance) and Ds (Spatial Distance) by using the following formula:

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2},$$
$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2},$$

```
Ds=sum((grad-clust_im_new).^2,3);
Dc=(Xn-Xnew(i,j)).^2+(Yn-Ynew(i,j)).^2;
```

5. The new distance is calculated by:

$$D' = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2}.$$

```
D=Ds+compactness^2*Dc/S^2;
```

6. If this distance is less than the initialized Dorig, then the corresponding pixel index in cIndMap is assigned the label of that cluster. Then each cluster is averaged to its mean colour. This is done by taking the mean of all the pixels belonging to a cluster. Thus the new cluster centers are updated by taking the mean of the corresponding x and y values in that cluster.

```
[~,~,mr]=find(mn_img(:,:,1));
[~,~,mg]=find(mn_img(:,:,2));
[~,~,mb]=find(mn_img(:,:,3));
clust_im(i,j,1)=mean(mr);
clust_im(i,j,2)=mean(mg);
clust_im(i,j,3)=mean(mb);
```

7. The center of cluster corresponding to least distance is calculated by :

```
(xregion-Xnew(i,j)).^2+(yregion-Ynew(i,j)).^2;
```

The indices of those pixels which are at least distance from the centre of the cluster are saved in the corresponding index of the CIndMap

```
dy_new = new_cent_clusy(dy);
dx_new = new_cent_clusx(dx);
size(index)
% set all the pixels to the nearest cluster center
cIndMap(index)=dx_new;
cIndMap(index+size(cIndMap,1)*size(cIndMap,2))=dy_new;
```

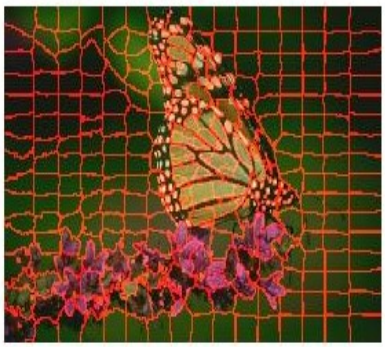
8. The boundary map is evaluated by finding the positive image gradient of CIndMap. Thus the segmented image is obtained by overlaying the boundary map onto the image.

```
time = toc;
[gx,gy]=imgradient(cIndMap);
bMap = (gx.^2 + gy.^2) > 0;
imgVis(bMap)=255;
```

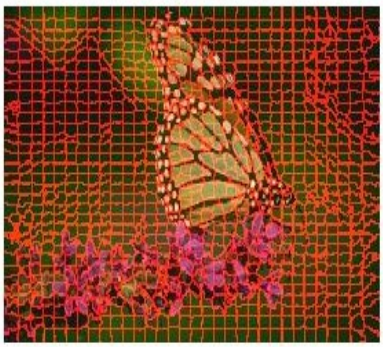
Original Image

Segmenation for K = 64

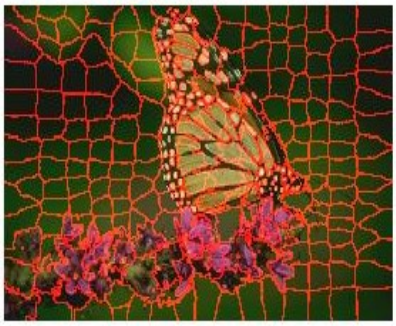
Segmenation for K = 256



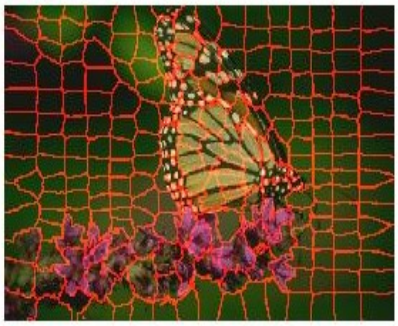
Segmenation for K = 1024



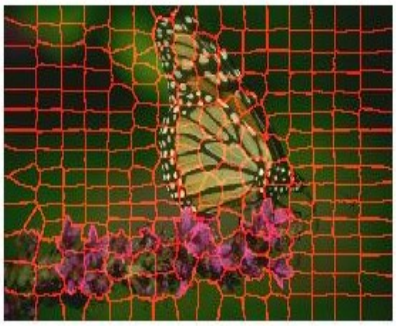
Segmenation for weight = 20



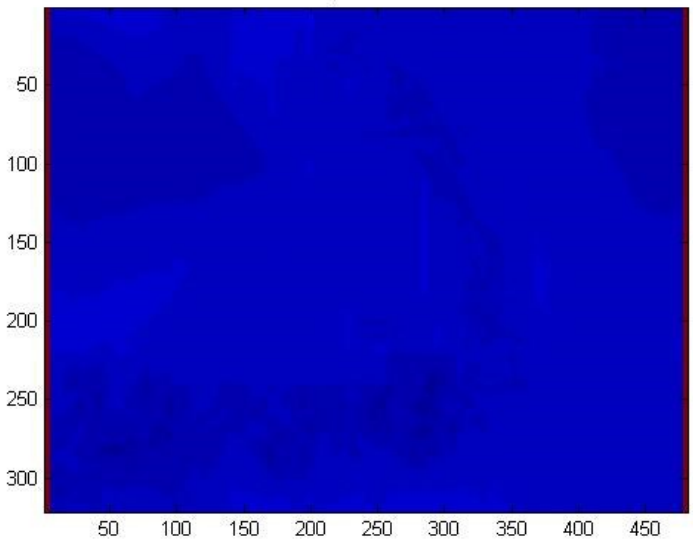
Segmenation for weight = 30

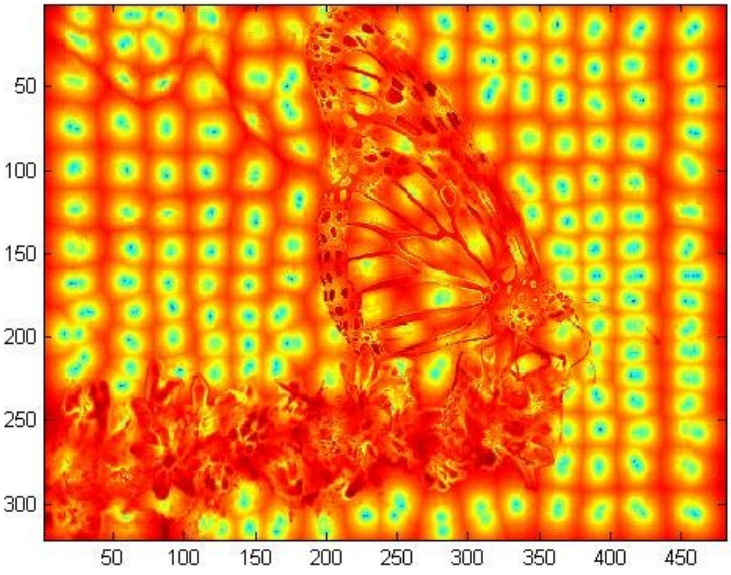


Segmenation for weight = 50



Error Map at initialization





Error map at Convergence

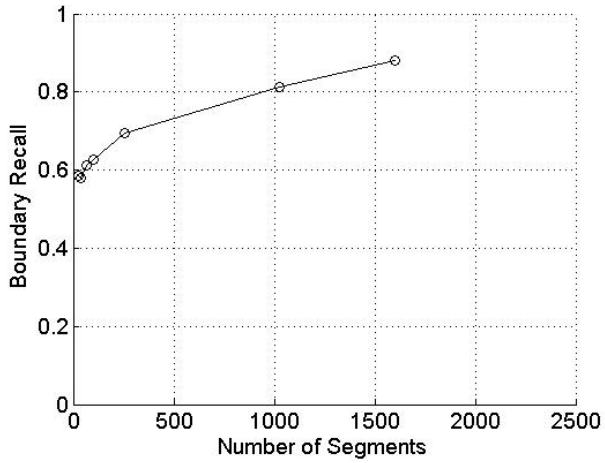
```
time =

    5.7492   21.3603   85.8774
```

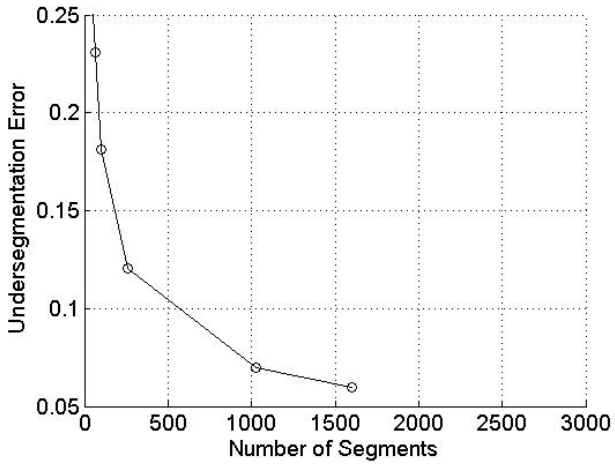
The times above gives the run times for 64,256 and 1024 segments . Thus it can be seen that the computation time increases as the number of segments increase.

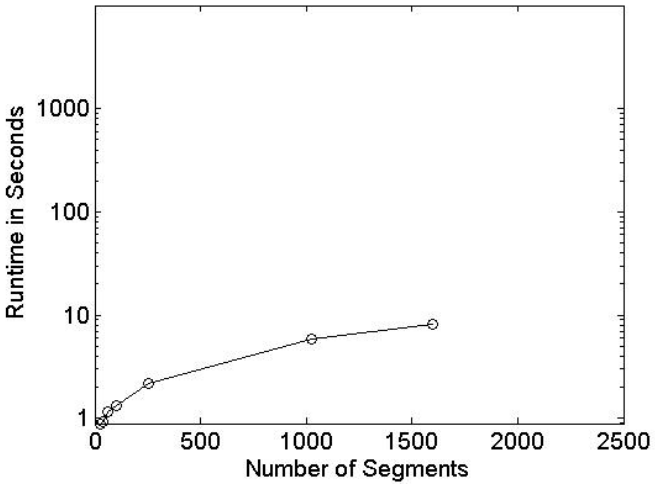
- Performing SLIC on Benchmark data set using eval SLIC :

  Average boundary recall = 0.694741 for K = 256
  Average underseg error = 0.420422 for K = 256

  The run time for execution on Benchmark was very high though for about 25 mins







Under segmentation error compares segment areas to measure to what extend superpixels flood over the ground truth segment borders. As it can be seen, as the number of segments are increased, the under segmentation error decreases. However more segmentation leads to extensive computation and hence increasing run time.

EM-ALGORITHM

1) Please find the attached pdf in the folder for the derivation.
2) Method:
   - The weight function was calculated for the different annotators.
   - For every image the mean value was calculated by the equation

$$\hat{\mu}_m^{(t+1)} = \frac{1}{\sum_n \alpha_{nm}} \sum_n \alpha_{nm} x_n$$

Where alpha is the weight function and x is the annotation scores by the 5 users.

```
for i=1:no_img
    mu(i) = sum(annotation_scores(image_ids==I(i)).*wf(image_ids==I(i))) /
sum(wf(image_ids==I(i)));
end
```

- Sigma is calculated by the equation :

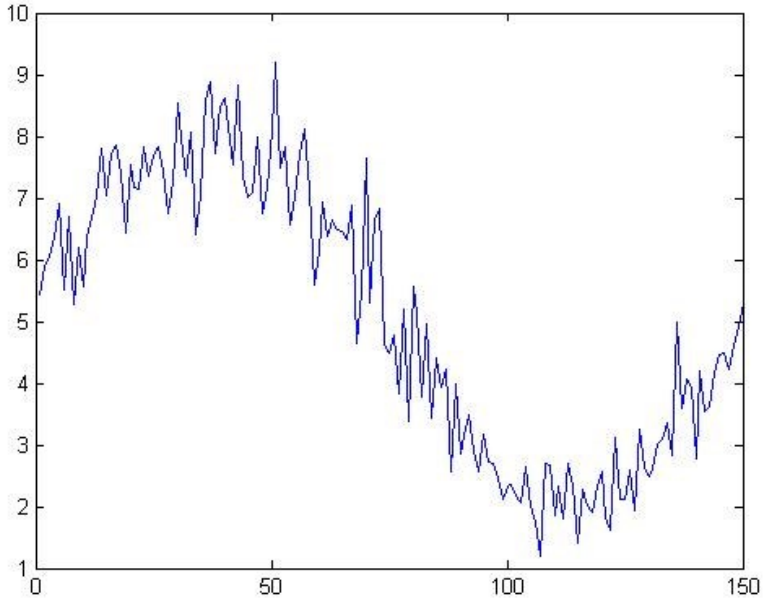
$$\hat{\sigma}_m^{2(t+1)} = \frac{1}{\sum_n \alpha_{nm}} \sum_n \alpha_{nm} \left(x_n - \hat{\mu}_m\right)^2$$

Where we subtract the mean corresponding to every image.

```
sigma = sqrt(sum((annotation_scores - mu(image_ids)).^2 .* wf) / sum(wf));
```

- The good annotators are those whose weight is greater than 0.5.
- The bad annotators are those whose weight is lesser than 0.5.

a)The plot of mean scores for 150 images



Plot of mean for all the images

b) Reporting sigma and the indices of bad annotators

```
bad =

     1
     2
     3
     5
     7
    11
    13
    17
    19
    23


sigma =

   0.868139710301756
```

The indices of bad annotators and the sigma value

3.GRAPH CUT ALGORITHM

Method:

1) Foreground and Background likelihood Estimation

Initially the GMM Model of foreground and background pixels were estimated.

```
fg_gmm = fitgmdist(fg, num_gmm);
bg_gmm = fitgmdist(bg, num_gmm);
```

Then the likelihood is evaluated by taking the log of probability of foreground and background models which can be obtained from the gmm distribution of both models correspondingly.

```
p_fg = fg_gmm.pdf(data);
p_fg = reshape(p_fg, sz(1), sz(2));
p_bg = bg_gmm.pdf(data);
llhBG = -log(p_fg);
llhFG = -log(p_bg);
```

2)The unary potential is defined as :

$$unary\_potential(x) = -\log\left(\frac{P(c(x);\theta_{foreground})}{P(c(x);\theta_{background})}\right)$$

It is the negative log of ratio of probability of foreground to background probability.

```
llhBG = -log(p_fg);
llhFG = -log(p_bg);
Dc = cat(3, llhBG, llhFG);
```

Pair wise potential terms:

$$edge\_potential(x,y) = k_1 + k_2 \exp\left\{\frac{-\|c(x)-c(y)\|^2}{2\sigma^2}\right\}$$

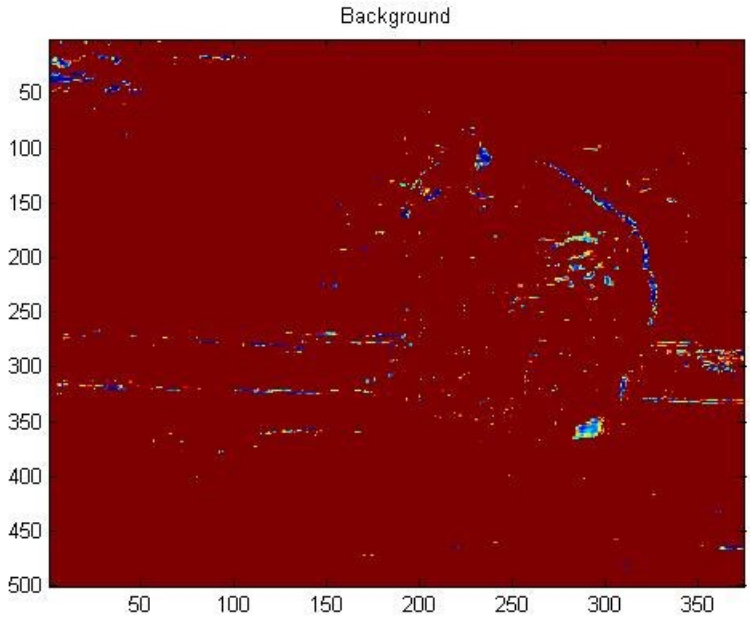
This was done by

```
for k=1:3
  [gx{k}, gy{k}] = gradient(im(:,:,k));
end
gx = sum(cat(3, gx{:}).^2, 3);
gy = sum(cat(3, gy{:}).^2, 3);
hV = 2 - exp(-gy/(2*sigma));
hC = 2 - exp(-gx/(2*sigma));
```

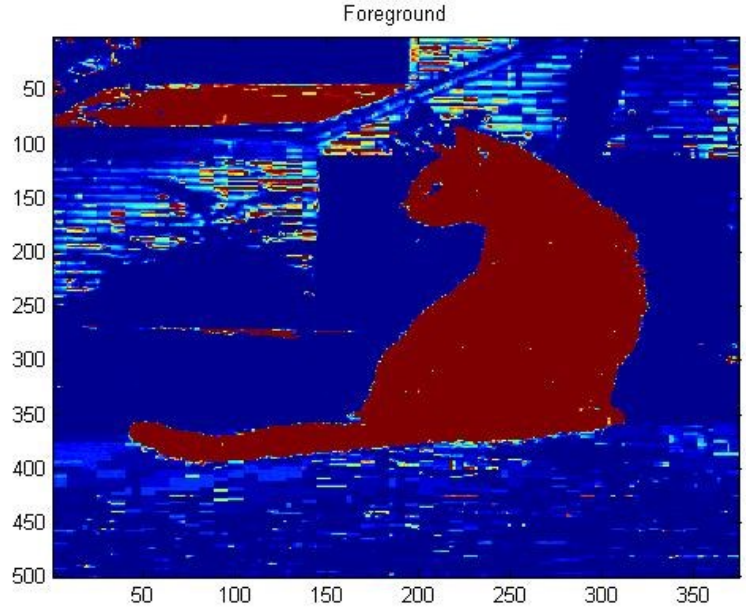
Energy Function :

$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

In the RHS of above expression , the first term corresponds to the unary potential and the second term corresponds to pair wise potential.



The background Intensity



The foreground Intensity

EXTRAS:

1.SUPER PIXEL

1) The segmentation for normal problem was performed with LAB colour space. When the segmentation was performed in RGB it gave:
   Average boundary recall = 0.268602 for K = 256
   Average underseg error = 0.565604 for K = 256

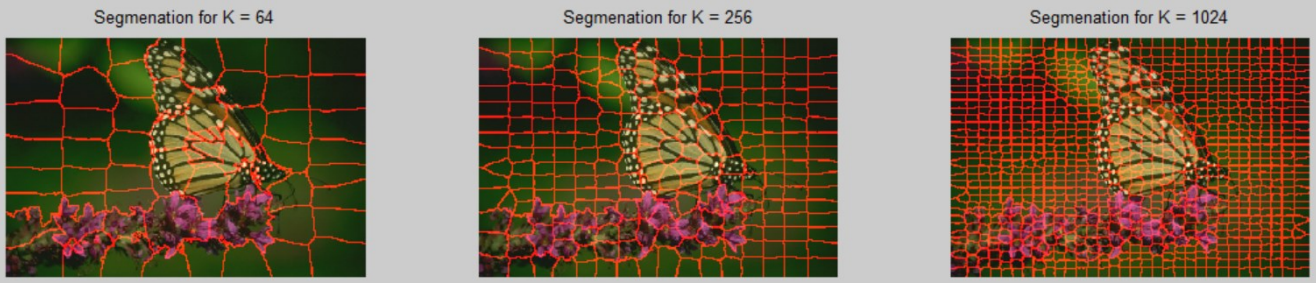
   This shows that the under segmentation error has reduced when changed to LAB colour space.
2) The Adaptive SLIC was performed by having a dynamic compactness and was changed to maximum special and coordinate distance corresponding to previous iteration. This follows from the equation mention in the SLIC reference paper section 4.5 :

$$D = \sqrt{\left(\frac{d_c}{m_c}\right)^2 + \left(\frac{d_s}{m_s}\right)^2}.$$

This was used to normalize the distance in the consecutive iterations. The segmentation as shown in the image below was more tight and precise than normal SLIC. The edges have become more smooth.

```
if (iter>1)
    ms=Ds_max;
    mc=Dc_max;
    D=sqrt((Ds./norm(ms)).^2+(Dc./norm(mc)).^2);
else
    D=Ds+compactness^2*Dc/S^2;
end

if norm(Ds)>Ds_max
    Ds_max=(Ds);
end
if norm(Dc)>Dc_max
    Dc_max=(Dc);
end
```

Output of Adaptive SLIC for different K



2. GRAPH CUT

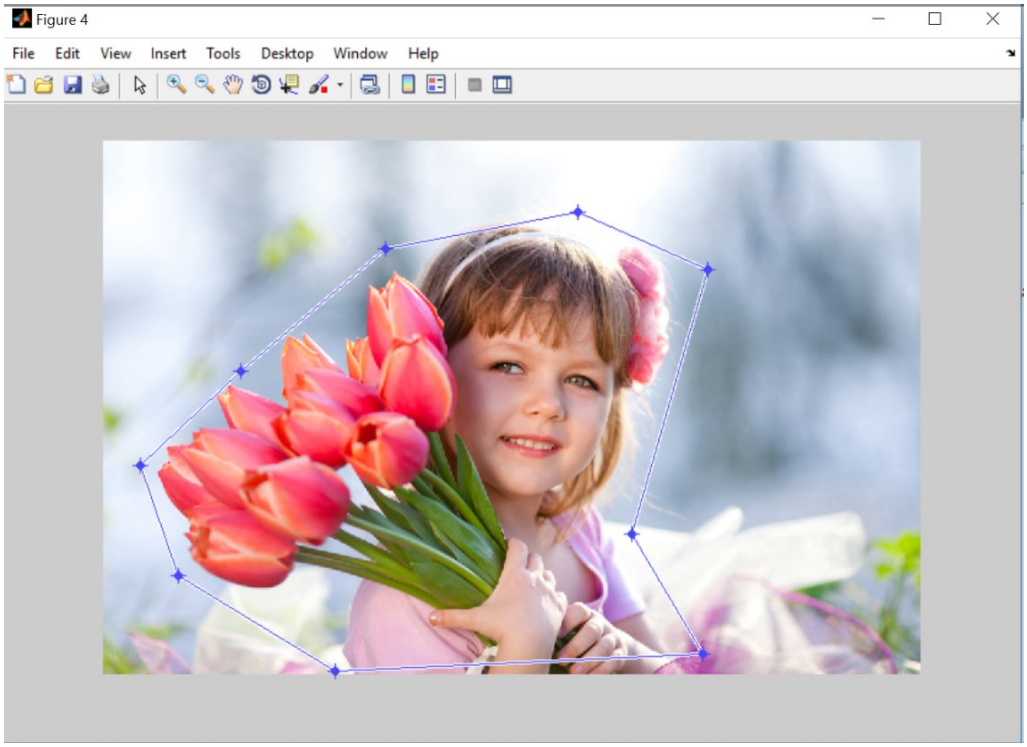
1. The function

   `[x,y,BW, xi, yi] = roipoly(img);`

   Was used to continuously draw polygon of our region of interest continuously. When the bounding box is drawn once the x and y vector corresponding to the boundary is passed to ploy2mask to form the foreground model.

   `mask = poly2mask(x,y, sz(1), sz(2));`

   This is then passed as foreground model and the usual graph cut is performed. The two sample outputs for 2 images with their corresponding drawn bounding boxes is shown below.



Graph Cut Output

Graph Cut Output

2. Pasting several masks onto the same image.

Method:
The mask formed from the to be pasted image was multiplied with the background image. This image was saved as a new overlap image and was used as the background image in the next iteration for pasting the grab cut image. If there is no corresponding pixel in the background image for pasting the grab cit mask , the corresponding pixel in background image is assigned 0. This code is given in bb_rect.m.

```
%im2 is background image and mask is to be overlapped image

g_im =im2;
% set background as blue
% g_im(:,:,3) = 1.0;
mask = zeros(sz);
for i = 1:sz(3)
    mask(:,:,i) = cut_img;
end
im2 = im2.*(1-mask) + img .* mask;
```

Graph Cut Output

Graph Cut Output

Background image

Overlap Image 1

Final overlap image