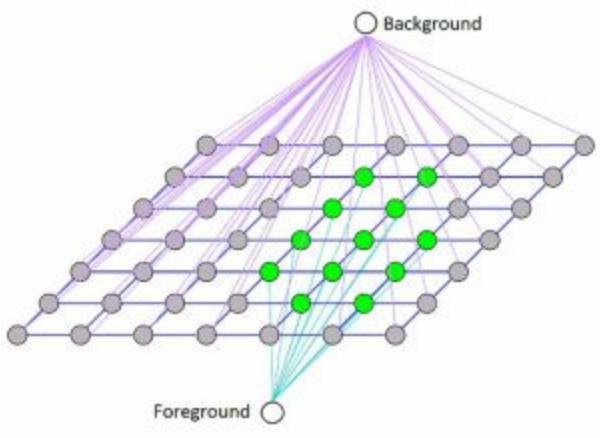
In this assignment I have implemented the GrabCut Algorithm used for Binary Image Segmentation. I have segmented the foreground and background parts of an image and also created a GUI which satisfy following requirements:

1. Selecting the bounding box for the object.
2. Selecting the specific foreground and background parts for corner and edge cases.

Algorithm details:

- The region of interest is decided by the amount of segmentation of foreground and background which is chosen by the user. Everything outside the ROI is considered as background while inside the ROI is still unknown.
- A Gaussian Mixture Model (GMM) is used for modeling the foreground and the background. The GMM learns and creates labels for the unknown pixels and each pixel is clustered in terms of color statistics. This is used to construct a Markov random field over the pixel labels, with an energy function that prefers connected regions having the same label, and running a graph cut based optimization to infer their values.
- A graph is generated from this pixel distribution where the pixels are considered as nodes and two additional nodes are added that is the Source node and Sink node. All the foreground pixels are connected to the Source node and every Background pixel is connected to the Sink node. The weights of edges connecting pixels to the Source

node and to the End node are defined by the probability of a pixel being in the foreground or in the background.



- If huge dissimilarity is found in pixel color, the low weight is assigned to that edge. Then the algorithm is applied to segment the graph. The algorithm segments the graph into two, separating the source node and the sink node with the help of a cost function which is the sum of all weights of the edges that are segmented.
- After the segmentation, the pixels that are connected to the Source node is labeled as foreground and those pixels which are connected to the Sink node is labeled as background. This process is done for multiple iterations as specified by the user. This gives us the extracted foreground.

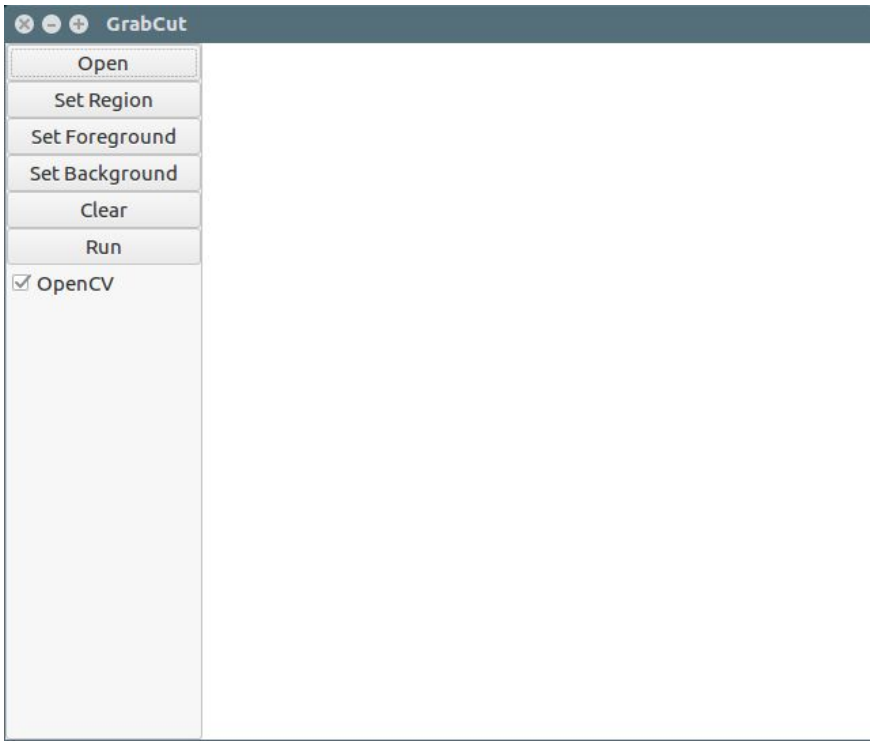
Implementation Details:

Following are the required libraries for running the code:
wx python, Scikit learn and igraph

GUI Details:
Once the function is invoked a window opens which allows for the following tasks:

Open: Allows user to browse for the image.
Set Region: Allows user to create a bounding box once image is loaded.
Set Foreground: Allows user to select specific regions which definitely lies in Foreground.
Set Background: Allows user to select specific regions which definitely lies in Background.
Clear: Allows user to reselect the bounding box.
Run: Runs the Grabcut Algorithm.

Here is a screenshot of the same.



Output:

Following are the output of few images on the given test case with the given bounding boxes. Output of all the images are given in the link. Number of Gamma Components = 5 and Gamma = 50.

Original Image vs Output

Original Image vs Output

Original Image vs Output

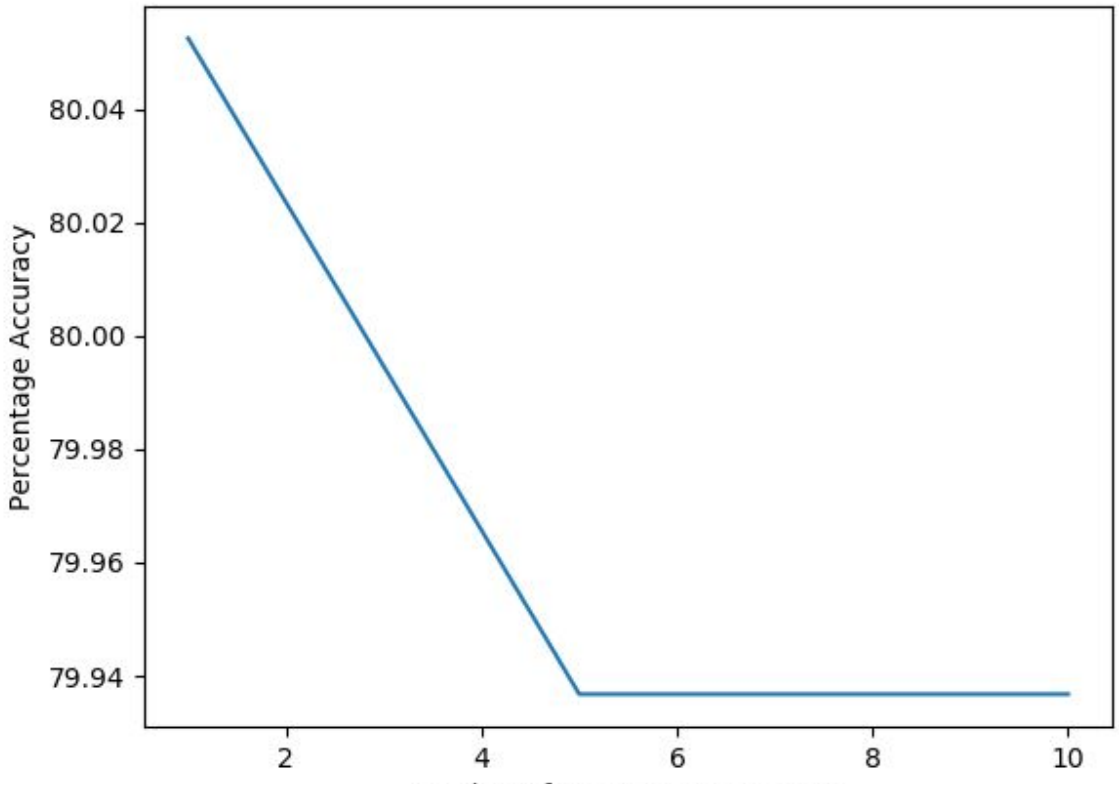
Original Image vs Output

Hyperparameter Variations:

Inbuilt implementation of OpenCV algorithm of the same is considered as the ground truth for the algorithm.

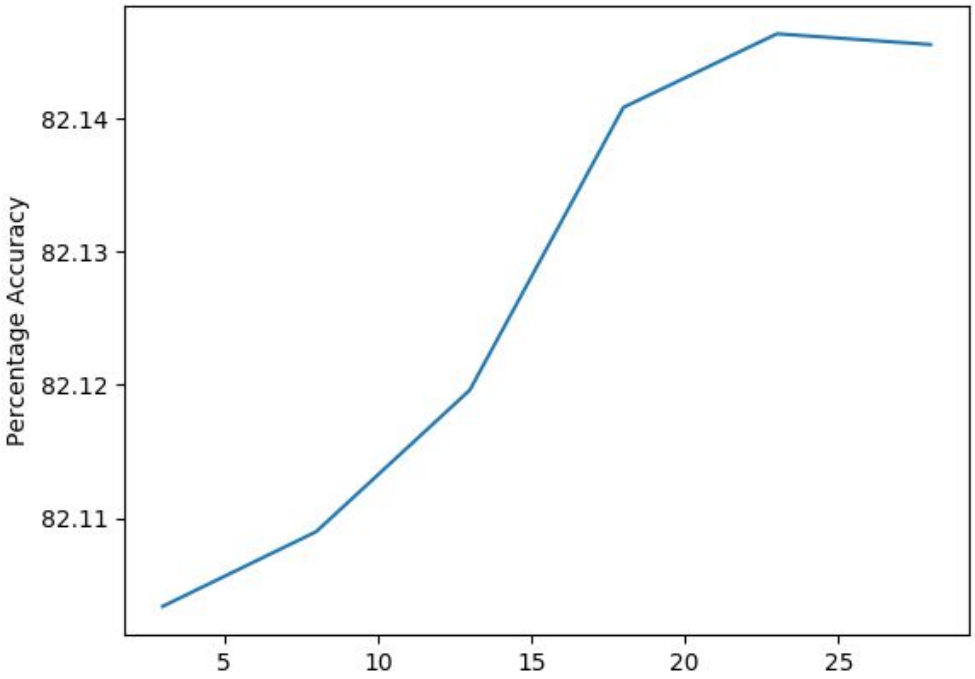
1. **Iterations**:

I varied the number of iterations from 1 to 10. The graph shows as the number of iterations increases, the accuracy decreases however it still remains very same.



2. **GMM Components:**

Let K denote the number of components in each Gaussian mixture model. The variation was done on K ranging from 1 to 28 with a step size of 5. Some images exhibited high accuracies when K is small because there is lesser variation in colors. However, when K > 5, performance decreases. In a sense, large K causes "overfitting" The implication of this experiment is that for images with large color skew, a small value of K should be used.



Variation of GMM components with percentage accuracy

GMM Components = 3

GMM Components = 28

3. **Gamma Optimization:**

I found that there is no linear correlation between entropy of spectral histogram of an image and an appropriate value of gamma.

4. **Color Space:**

Variations are shown in HSV and YCbCr output.

HSV Output

YCbCr Output

Outputs of all the test cases are uploaded to the google drive on this link.