Grab Cut - Interactive Image Segmentation

Mehul Arora (2021701033)
Neha S (2021702012)
Sabyasachi Mukhopadhyay (2021801003)
Sai Amrit Patnaik(2020701026)

December 6, 2021

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Contents

1 Abstract

In this report we describe our final project, GrabCut, which augments graph cut by iteratively minimizing the energy in an image to find the local optimal binary segmentation. The method features a simple user interface which only requires the user to draw a rectangle around the object to be cut. This initialization defines only the background pixels, and then iteratively improves the segmentation by recalculating the probability distributions of the background and foreground colors and then uses these estimates in graph cut.

2 Problem Statement

The problem of efficient, interactive foreground/background segmentation in still images is of great practical importance in image editing. All previous approaches demand considerable user interaction and yet the quality of segmentation obtained is poor, mostly because most of these approaches use texture or edges information. These approaches also don't addresses the problem of extraction of a foreground object in a complex environment whose background cannot be trivially subtracted.

3 Objective

The objective of the project is to implement GrabCut which is a powerful and robust algorithm that builds over Graph Cut algorithm(proposed by Boykov and Jolly). The user is requested to mark a single rectangle around the object, defining the outer part of the rectangle as definite background, and the inner part of it as an unknown combination of the object (foreground) and some background. These constraints are used as initial solution to the problem, leading to an iterative method which in conclusion aims to assign each pixel in the image its label - Background or Foreground.

4 GrabCut

GrabCut [1] is an algorithm for image segmentation based on graph cuts [2]. It is used to extract the foreground objects in images with minimal user interaction. Initially, the user needs to draw a bounding-box (i.e., a rectangle) around the foreground object to be segmented. Based on these regions, the color distribution of the foreground and the background objects are estimated by the algorithm using Gaussian mixture model. This is then followed by the iterative energy minimization process to get the best estimates and segment the foreground object. However, in certain cases, the algorithm could misclassify the background and the foreground regions resulting in improper segmentation. This is rectified by taking additional inputs from the users (in terms of strokes) regarding the misclassified regions and rerunning the algorithm to obtain better segmentation results.

4.1 Problem Formulation

The image is defined in an RGB color space. Instead of using histograms of pixels in the original implementation of Graph Cut [Boykov and Jolly 2001] the algorithm of GrabCut uses Gaussian Mixture Models (GMM) to cluster the RGB pixel values from the image. There are two GMMs, one foreground and one for background pixels. Each GMM is defined to have 5 clusters (K = 5, set to be constant based on the original paper). There is a hard segmentation of the pixels between the foreground and background GMMs represented by α_n= 0 (background) or 1 (foreground).

To help with segmentation, a trimap is used. It is defined by $T = \{T_B, T_U, T_F\}$. T_B stores all background pixels, T_U stores unknown pixels, and T_F stores foreground pixels. At the beginning of GrabCut, the user draws a box around the foreground object. The background pixels in the trimap (T_B) are initialized with everything outside this rectangle. The unknown pixels are set to be $T_U = T_B^\top$ and $T_F = \phi$.

The overall objective is to minimize the energy function,

$$E(\alpha, k, \theta, Z) = U(\alpha, k, \theta, z) + V(\alpha, Z)$$

$U(\alpha, k, \theta, z)$ is the unary potential of each pixel, learnt using a GMM with K components. and $V(\alpha, Z)$ is the pairwise potential or smoothness energy.

4.2 Unary Potentials

The Unary potentials are learnt using GMMs based on color intensities. After initializing the trimap and the alpha values, two GMMs are created. The background GMM is created by T_B which has values $\alpha_n = 0$. The foreground GMM is created by T_F which has values $\alpha_n = 1$

The Unary potential is,

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

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

4.3 Pairwise Potentials

The pairwise potential is computed using a smoothness function,

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

$\beta = (2 < (z_m - z_n)^2 >)^{-1}$, <> denotes expectation over the original image.

4.4 Graph structure

The graph structure for GrabCut is defined by each pixel and its edges to its four neighboring pixels (North, East, South, West). Every pixel also connects to two terminal nodes defined as sink (background) and source (foreground). The terminal nodes are important because they are used when performing the min cut to segment foreground and background pixels. The neighborhood weights are set such that min

cut will find it expensive to separate pixels with similar intensity, and cheap to cut where they are different (along edges). These weights are static across iterations. Terminal weights must be recalculated based on the new GMMs.

5 Procedure

5.1 Initialization Step

1. he user initializes the trimap by only giving the background pixels. The foreground set of pixels is set to the empty set. The unknown pixels are set to the compliment of the background pixels.

2. Hard segmentation is performed to create a foreground and background pixel sets. The background pixels go into the background pixel set. The unknown pixels is the foreground set.

3. Create foreground and background GMMs based off the sets previously defined.

5.2 Iterative Step

1. Assign a foreground and background GMM cluster to every pixel in the unknown set based off the minimum distance to the respective clusters.

2. Learn GMM parameters based off the pixel data with the newly assigned foreground and background clusters. This step will get rid of the old GMM and create a new GMM with the assigned foreground and background clusters from every pixel.

3. A graph is constructed and Min Cut runs to estimate new foreground and background pixels.

4. Repeat steps 13 until the the label of foreground background pixels converges.

6 Results of GrabCut algorithm

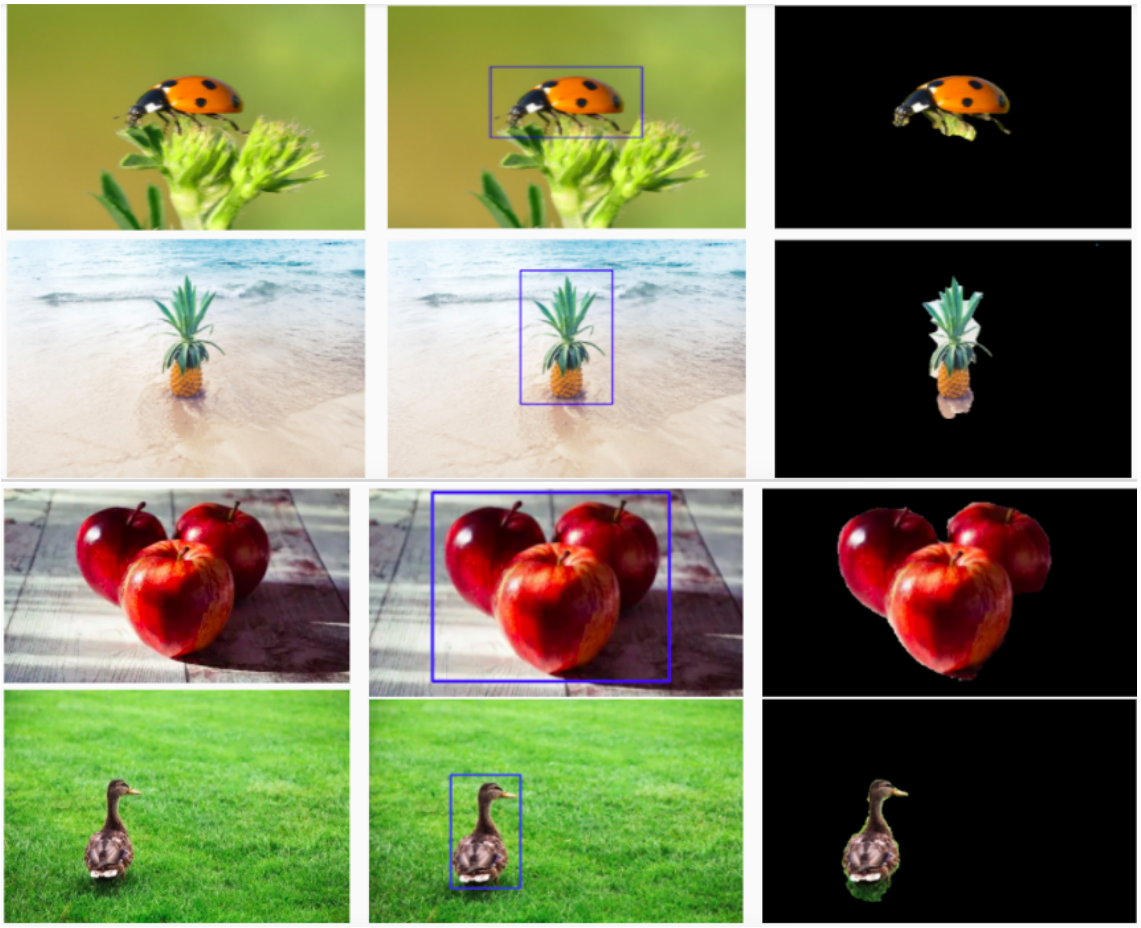

Figure 1: Examples of the input image (left), bounding box drawn by the user (middle) and the corresponding output image (right) obtained using GrabCut algorithm.

7 User Editing

In certain cases, the vanilla algorithm fails in appropriately estimating the background and foreground regions as shown in Figure 2.

7.1 Failed results

We can observe that the algorithm fails to segment out the vehicle and the forest from Figure 2.

7.2 Improvement

To enhance the performance in such cases, we can provide additional information about the background and foreground regions. The results of the algorithm after providing additional information is shown in Figure 3 and 4.

Figure 2: Examples of the input image (left), bounding box (middle) and the failed segmentation output image (right) obtained using GrabCut algorithm.

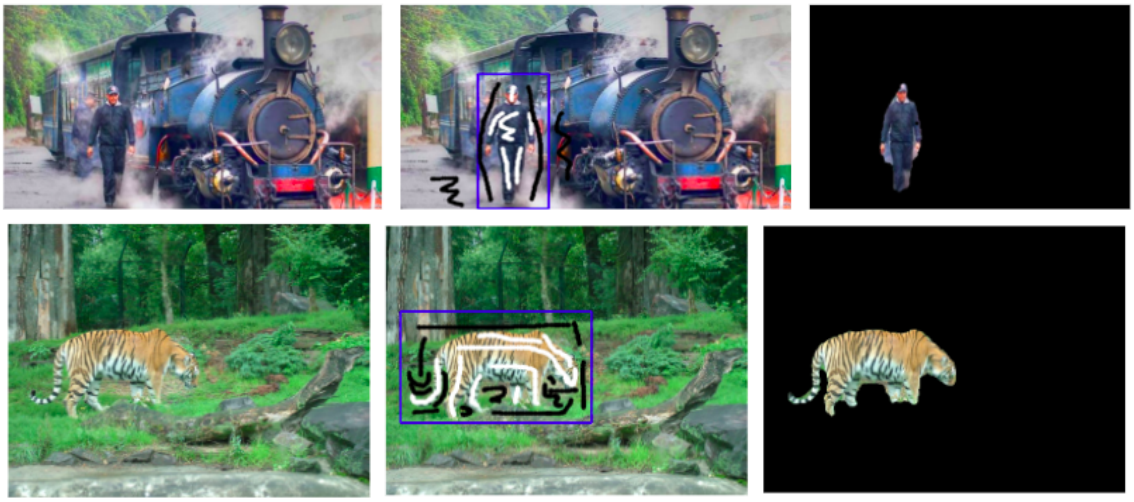

Figure 3: Improvement using user inputs

When the foreground and background information about the vehicle and the person are given, as shown in Fig.3, the person is properly segmented. Similarly, we can notice in Fig.4, that the tiger is properly segmented.

Thus, the performance of the algorithm improves on providing additional information.

8 Experiments

From the above figures that show the results of GrabCut algorithm, we can observe that the algorithm is able to extract the foreground objects appropriately. We demonstrate the results of GrabCut segmentation algorithm by varying different parameters in further Sections.

8.1 Number of iterations

In this section, performance of the implemented GrabCut algorithm is evaluated based on the number of iterations required to segment the foreground and back-

ground objects of the image. The result of varying the number of iterations is shown in Figure 4. We can observe that the performance of the algorithm increases when the number of iterations is increased from 1 to 5. However, no significant improvement is observed when the number of iterations is increased to 12. Thus, we set the optimal number of iterations as 5 based on the experimentation.

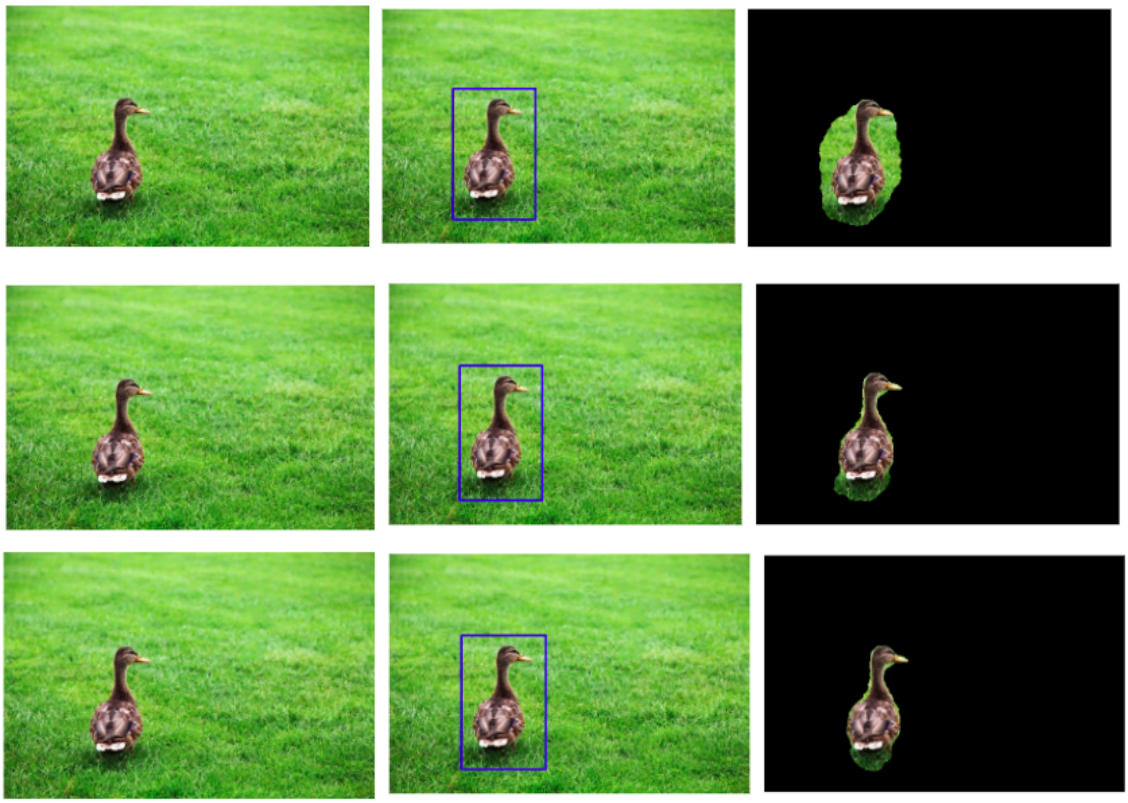

Figure 4: Outputs of grabcut algorithm with number of iterations 1, 5 and 12.

8.2 Number of GMM components

In this section, results of GrabCut for varying number of GMM components is presented as shown in Figure 5. We can observe that increasing the GMM components does not significantly improve the results. Thus, we set the optimal number of GMM components as 5 based on the experiments and also according to [1].

8.3 Choice of gamma

In this section, gamma is varied and the observations on the performance of the algorithm are noted. We experiment with different values of gamma - 5, 30 and 100 as shown in Figure 6. As we can see, best results are obtained when gamma is set to 30.

8.4 Neighborhoods in pairwise term

Algorithm is tested for 4-neighborhood and 8-neighborhood in pairwise term. The results are as shown in image 7. 8neighborhood performs better.

Figure 5: Outputs of grabcut algorithm with number of GMMs as 2, 5 and 10.

8.5 Size of bounding box

Performance based on different initial sizes of bounding box from a tight initial bounding box or a loose bounding box, is checked. Providing a proper bounding box is essential for good segmentation as seen from the results in Figure 8.

8.6 Different color spaces

In this section, effect of different color spaces on the performance of the GrabCut algorithm is analized. We experimented with HSV, LAB and RGB color spaces, however no major improvements were found on the tested images. Thus, we set the color space to be RGB.

References

[1] Y.Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images. In *Proceedings Eighth IEEE Inter-*

Figure 6: Outputs of grabcut algorithm with gamma values 5, 30, 100.

national Conference on Computer Vision. ICCV 2001, volume 1, pages 105–112 vol.1, 2001.

[2] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, page 309–314, New York, NY, USA, 2004. Association for Computing Machinery.

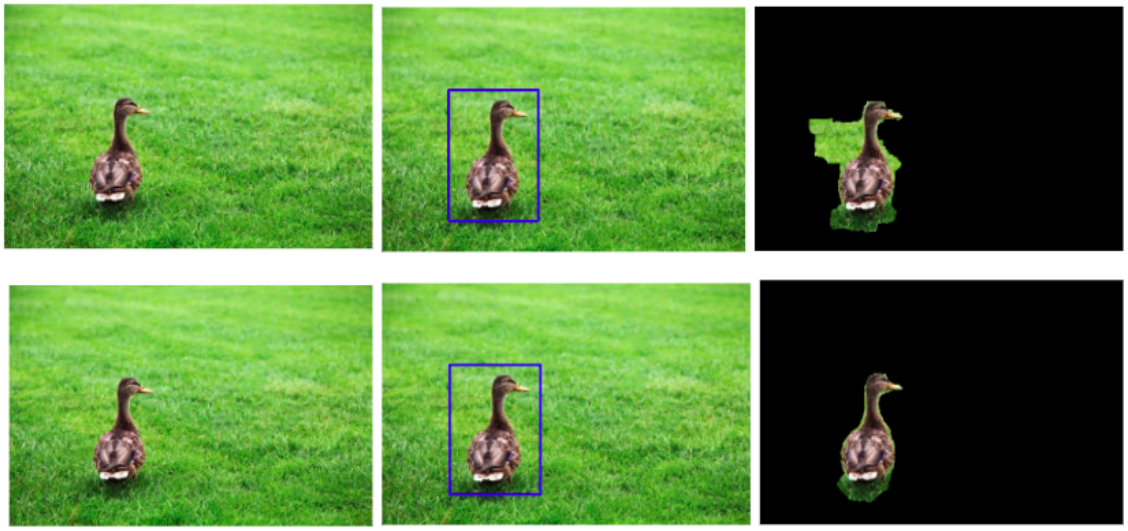

Figure 7: Outputs of grabcut algorithm for 4 and 8 neighborhood.

Figure 8: Outputs of grabcut algorithm for tight to loose bounding box.

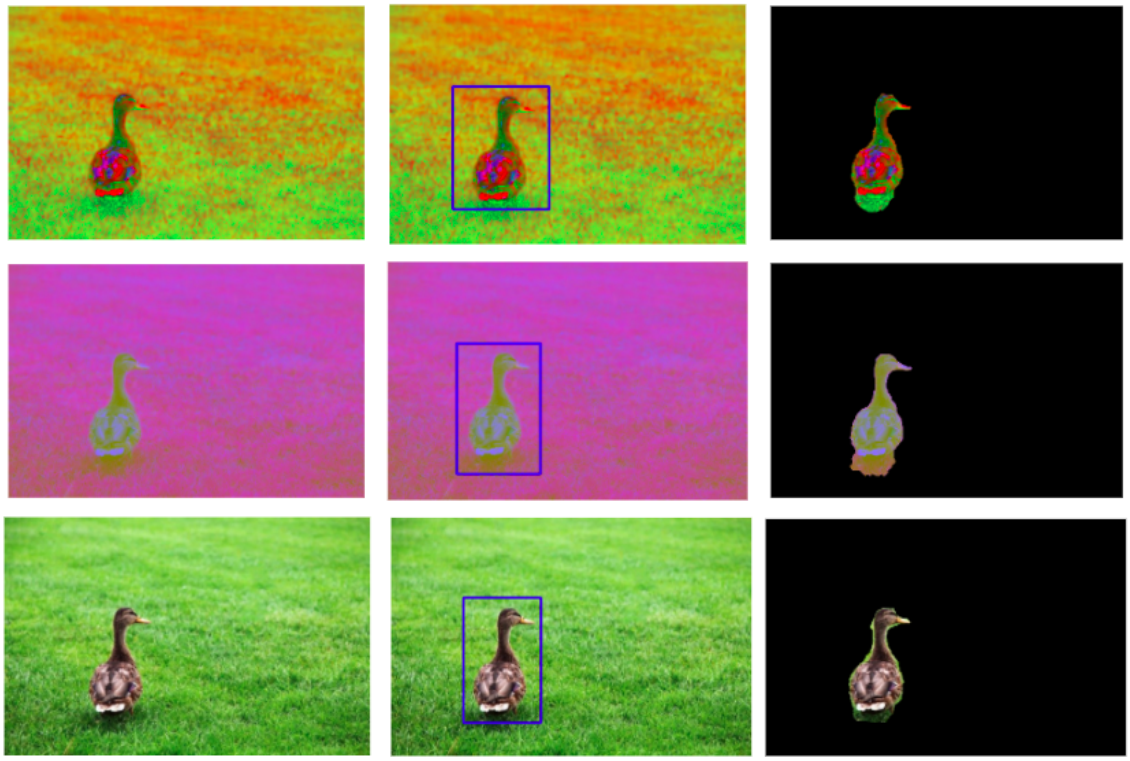

Figure 9: Outputs of grabcut algorithm for HSV, LAB and RGB color schemes.