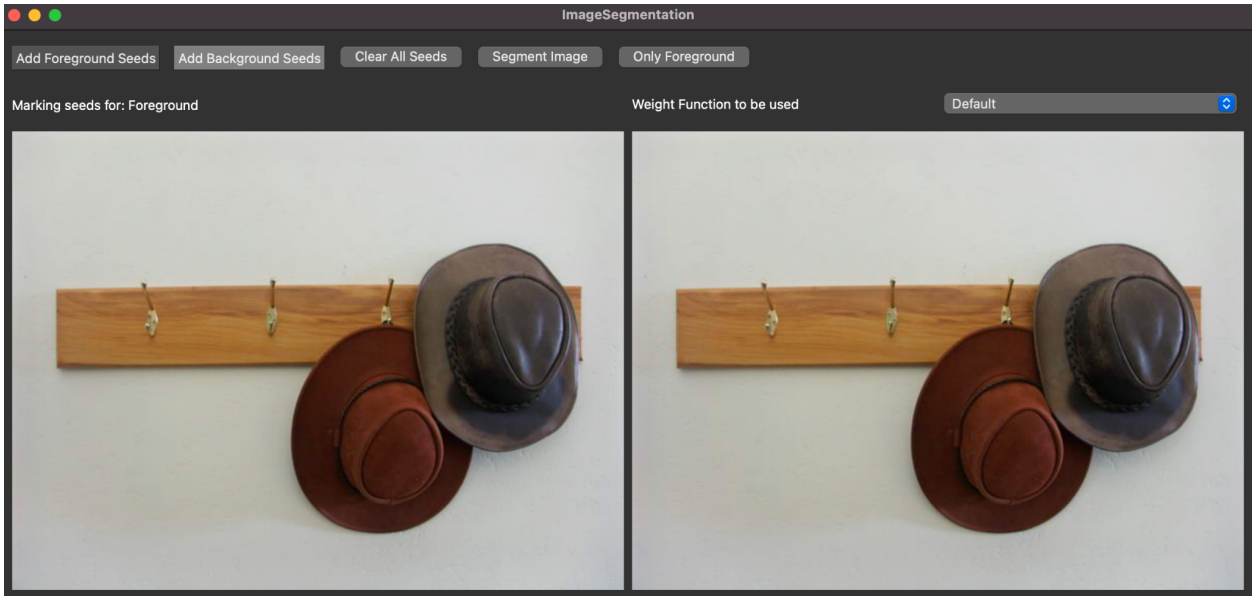
Design Lab

Image Segmentation Tool

Name: Shashvat Gupta

Roll Number: 19CS30042



Introduction

Graph algorithms have been successfully applied to a number of computer vision and image processing problems. Our interest is in the application of graph cut algorithms to the problem of image segmentation. We design a desktop application to upload images from the computer and apply the graph-cut algorithm to segment foreground and background from the image.

Implementation

Framework

Main language used to develop the application is **Python**. We use **PyQt6** to design the desktop application user interface. This module is used to have liquid and fluent design

elements. To support image operations, we decided to use **Python-OpenCV**. This library provides a wide range of operations for image processing including adding overlays and masking etc. which were needed while importing and showing segmentation of images. We converted mapped image pixels to graph nodes and constructed a representation of a graph. In this graph, we implemented the Min-Cut image segmentation algorithm using the Python module **PyMaxflow** to solve the min-cut algorithm and segment the entire image into foreground and background.

Working

User Interface

The user interface is an easy to use desktop application designed using **PyQt6**. We used file path dialogue boxes and simple button widgets to design the interface. We also used form elements like Edit Text boxes and dropdown menus to ease interface interaction for users. Apart from that, all states stored by the interface are visible to the users.

The interface allows the user to:

- Open an image file (Ctrl + O)
- Save an image file (Ctrl + S)
- Change method used for segmentation
- Change parameters for segmentation
- Put Seeds in the image (foreground and background)
- See segmented image using mask
- See Background removed image

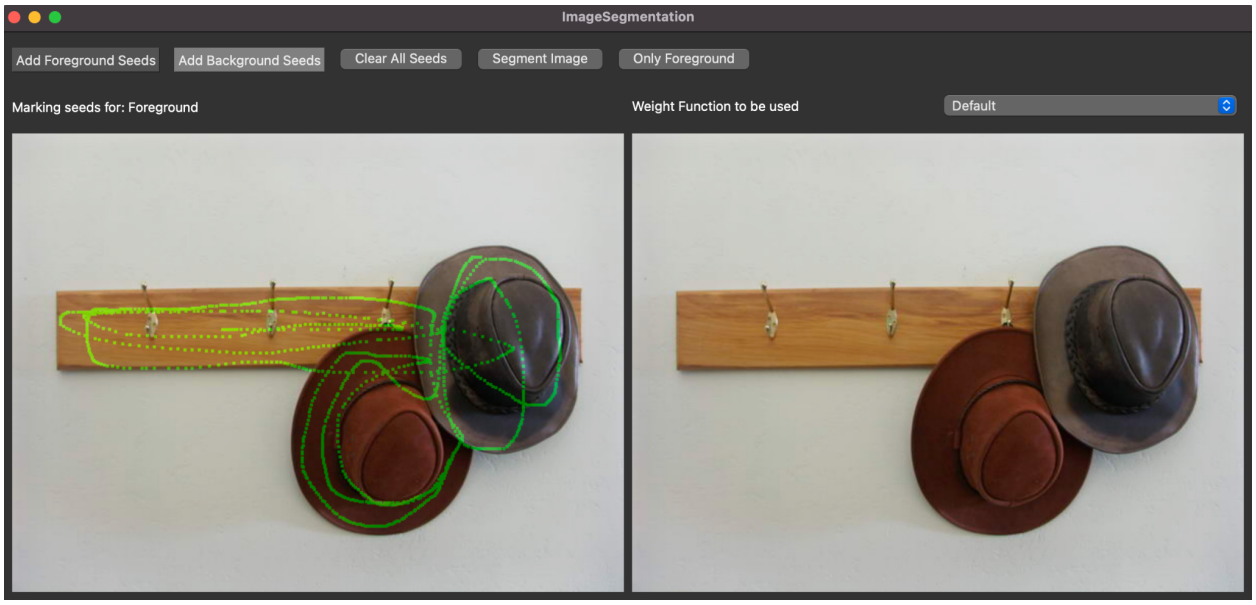


Fig 1: Seeding Foreground

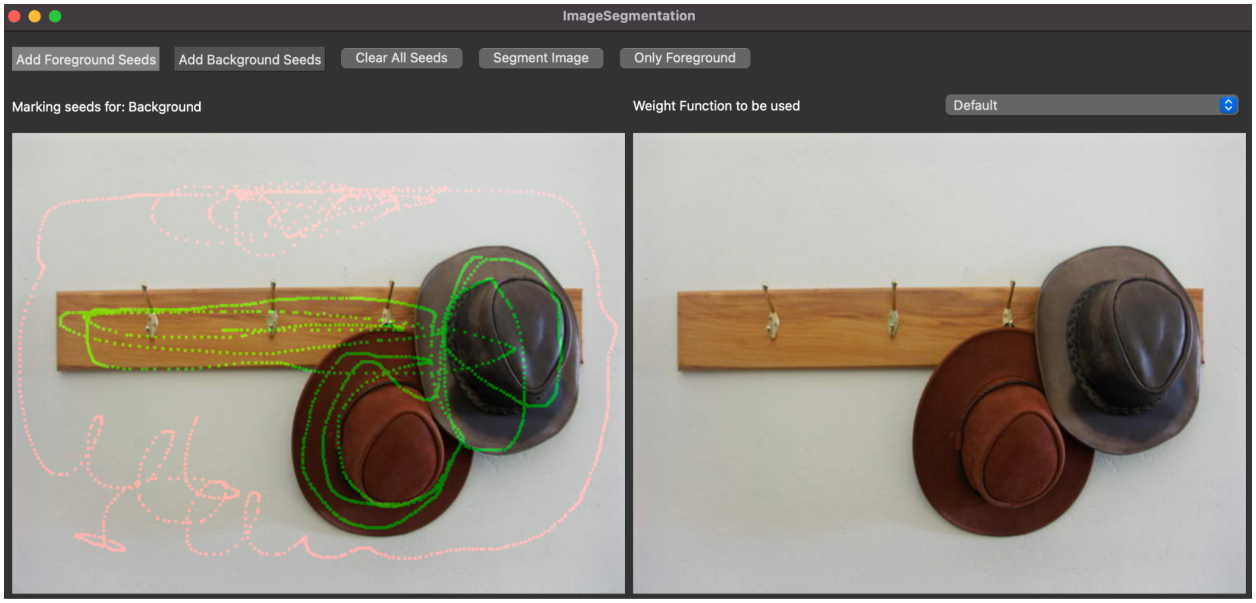


Fig 2: Seeding Background

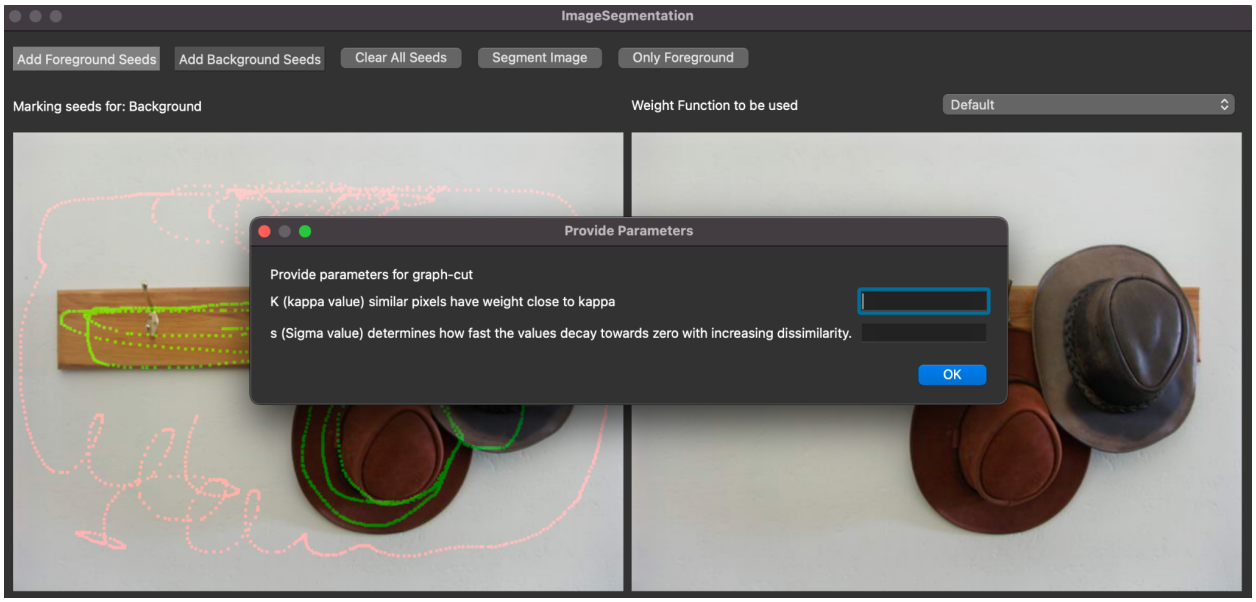


Fig 3: Providing parameters for Default weight Selection

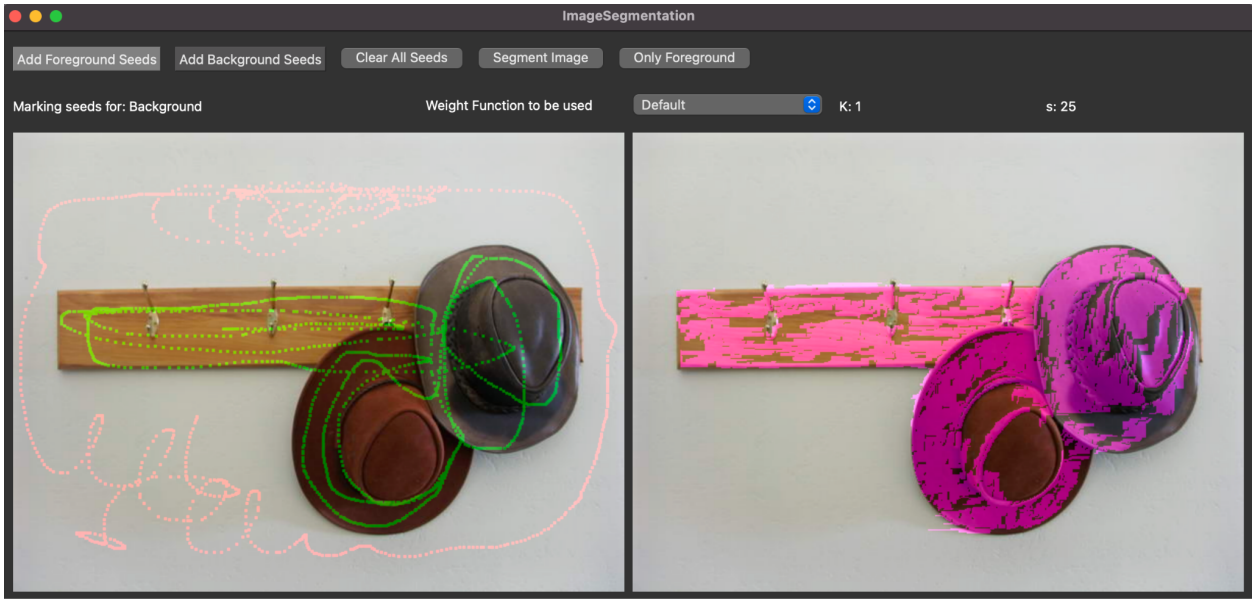


Fig 4: Segmentation Result using Default Weight Selection

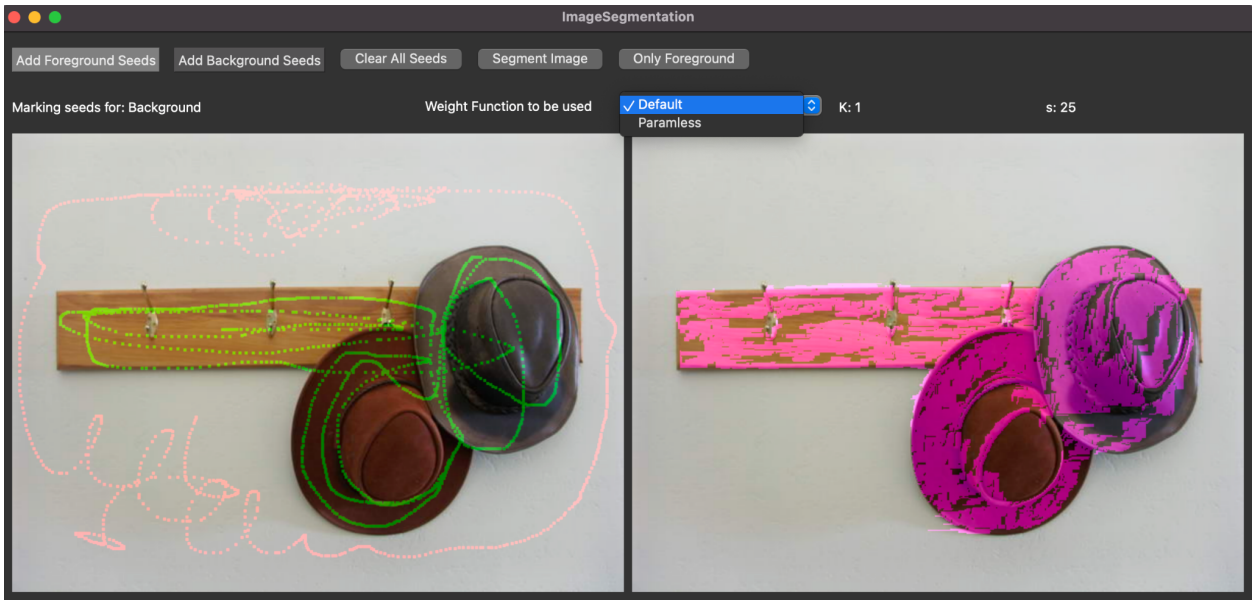


Fig 5: Changing Weight Selection Method

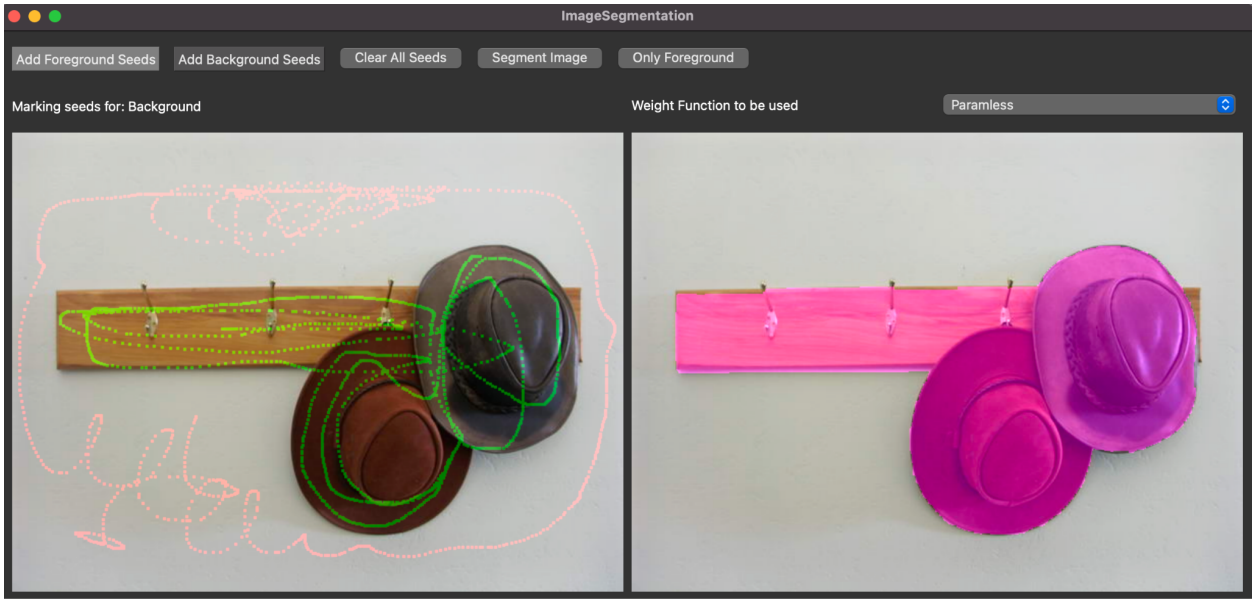


Fig 6: Segmentation Result using Paramless Weight Selection

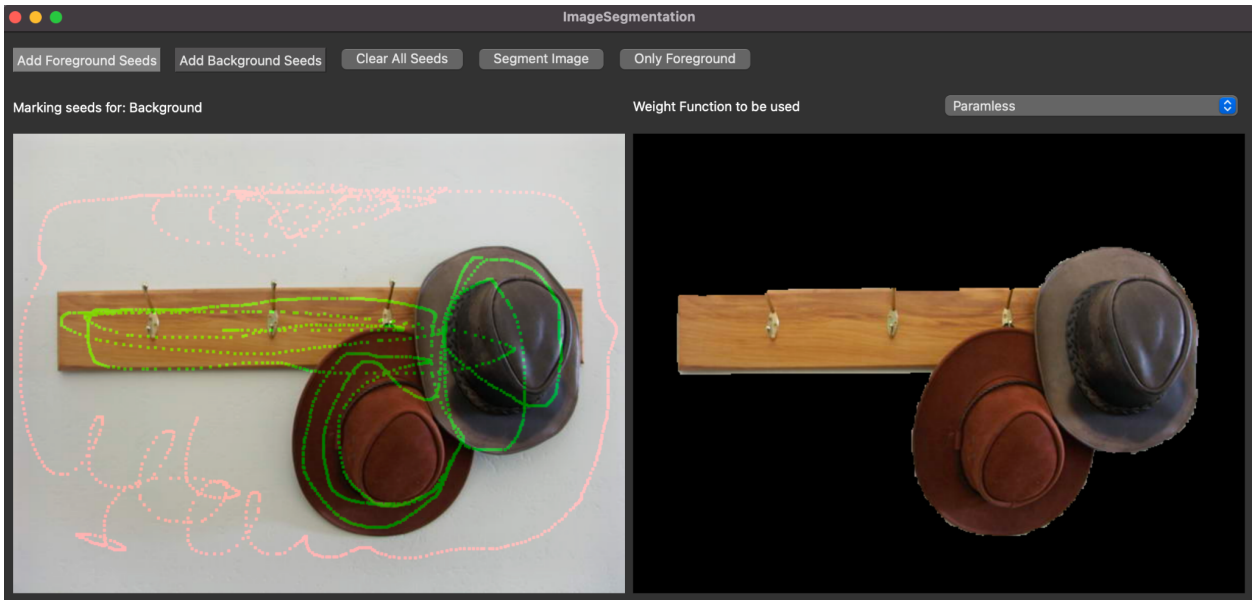


Fig 7: Result of removing background

Min-Cut Algorithm

We used **Python-OpenCV** to open and read image files. It is also used to design overlays and masks, for example segmentation masks. By reading the image, we map the image to a **Maxflow** graph. This graph is later used to solve the min-cut algorithm and find segmentation labels for each pixel.

The mapping between image pixels and graph is done as follows:

Nodes: Each pixel of the image is a node. Apart from these pixels, two extra terminal nodes are added which are source and sink.

Edges: An edge exists between each pixel and its four neighbors. Also, an edge exists from source node to the node corresponding to a pixel and from the pixel's node to the sink node.

Edge Weights: This is the most important part as the min-cut / max-flow algorithm works according to edge weights. In our interface we have provided two ways to decide edge weights:

1. *Default*: In this method, we provide the **k** *(kappa)* and **s** *(sigma)* tobe used to decide edge weights. The edge weight between two neighbors is decided as:

$$w_{i,j} = ke^{-|I_i - I_j|^2/\sigma}$$

2. *Paramless*: In this method, no user parameters are needed. The edge weights are decided as:

$$w_{i,j} = \frac{1}{1 + |I_i - I_j|^2}$$

$w_{i,j}$ is the weight between node i and j
I_i is the intesity of the node i
k is the kappa constant
σ is the sigma constant

On the mapped graph, we apply a max-flow algorithm to characterize foreground and background based on intensity change on pixels and the initial seed pixels. The segregation of these nodes is mapped back to pixels which are then showed in the interface.