K-Means Clustering for Image Segmentation

Elif Gürkan

Introduction

This report delves into the realm of image segmentation using the K-Means clustering algorithm, a cornerstone technique in unsupervised learning. The objective is to partition an image into distinct segments, enhancing our understanding and analysis of its content. This assignment offers an in-depth exploration of image segmentation, emphasizing the versatility and efficacy of K-Means in diverse scenarios.

My approach integrates two advanced feature extraction techniques:

> Pixel-Level Features: Incorporating both color (RGB) and spatial information, I developed a comprehensive pixel representation. This dual feature set enables a nuanced segmentation, capturing both color intensity and spatial relationships.
>
> Superpixel-Level Features: Leveraging the SLIC Superpixel method, I created clusters of pixels, or superpixels, as foundational units for segmentation. This level includes:
>
> - Mean RGB Color Values
> - RGB Color Histograms
> - Mean Gabor Filter Responses

The fusion of these features with K-Means clustering unveils intricate details within images, facilitating a deeper analysis. The assignment's outcome is a set of segmented images, each highlighting the unique characteristics brought forth by different feature sets. I evaluated the efficiency and representativeness of these features, contributing to the broader understanding of image segmentation techniques.

My implementation is a testament to the power of combining traditional algorithms with innovative feature extraction methods, setting the stage for advanced image processing applications.

1. Features

Pixel-Level Features

- Normalization and Reshaping: The code normalizes RGB values by dividing by 255.0, ensuring that the color data fits within a 0-1 range. This normalization is crucial as it prevents features with larger numeric ranges from dominating those with smaller ranges during the clustering process.
- Feature Vector: RGB values are reshaped into a 2D array of shape (num_pixels, 3), where num_pixels is the total number of pixels in the image. This reshaping is important for processing the image data in a format suitable for machine learning algorithms.
- Spatial Features: The code also incorporates spatial information by normalizing the x and y coordinates of each pixel and appending them to the RGB features, creating a 5-dimensional feature vector per pixel ([R, G, B, x, y]). This addition of spatial data is significant as it allows the K-Means algorithm to segment the image based on both color and spatial proximity, which can lead to more coherent segments.

Superpixel-Level Features

In this part of the assignment, superpixels are generated using the SLIC algorithm from the scikit-image library. Each superpixel aggregates pixels into perceptually meaningful, non-overlapping regions. For each superpixel, I extracted three types of features:

Mean of RGB Color Values: This feature vector represents the average color of all pixels within a superpixel.

RGB Color Histogram: It quantifies the color distribution within a superpixel by calculating histograms for each RGB channel.

Mean of Gabor Filter Responses: Gabor filters at various scales and orientations are applied to the image. The feature vector for each superpixel is then computed as the mean response of these filters within the superpixel area.

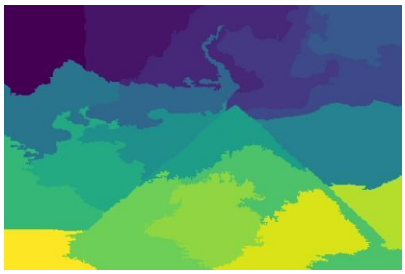
At first, I implemented an algorithm which includes;Superpixel Segmentation with SLIC: The slic function from skimage.segmentation is used to segment the image into superpixels. Parameters such as num_segments and compactness control the segmentation process.Mean RGB Values: For each superpixel, the mean of the RGB values is calculated, providing a color representation.

RGB Histograms: RGB color histograms for each superpixel are computed to capture the color distribution.

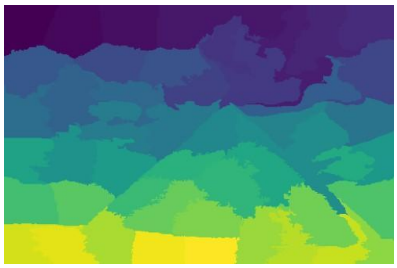
Gabor Filter Responses: The image is first converted to grayscale, then a set of Gabor filters with varying frequencies and orientations is applied to extract texture features. The mean response of these filters within each superpixel is computed. Andü Gabor Filter Bank Creation is a filter bank is generated using gabor_kernel from skimage.filters, and each filter is applied to the grayscale image using convolve from scipy.ndimage.These features provide a rich representation of the image's content at a higher level than individual pixels, which is likely to yield more meaningful segments when used with K-Means clustering.

```
def extract_superpixel_features(image, num_segments=20, compactness=10, num_bins=256):
```

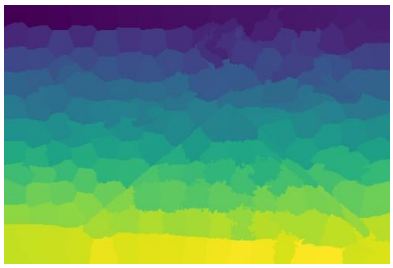
- num_segments variable



num_segments = 20



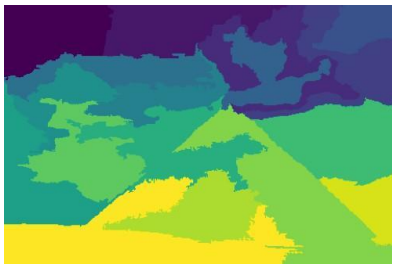
num_segments = 50



num_segments = 200

When I increase the num_segments, the image is divided into more parts and the image with a more obvious outline at 20 becomes a more fragmented and more complex image when it is 200.

- compactness variable



compactness = 5

compactness = 20

compactness = 100

When I increase compactness, the outline becomes unclear and as the density increases, I see only a color table at compactness = 100.

```
num_segments = 10

segments_slic = slic(img, n_segments=num_segments, compactness=10)
```

- Outline the segments with SLIC


```
num_segments = 10
```


```
num_segments = 50
```

2. K-Means Clustering

K-Means is a popular clustering algorithm used in unsupervised machine learning. In image processing, it's often used for segmentation tasks.

Initialization of Centroids: My algorithm starts by initializing k centroids. These centroids are crucial as they represent the initial grouping of the data points. In the context of image segmentation, these centroids can be thought of as representing the initial 'average' color or texture of k segments.

Assignment of Data Points to Nearest Centroids: In the assignment step, calculating the distance between each data point and every centroid is a key operation. The code should efficiently handle this computation, as it can be resource-intensive, especially for large images or a high number of clusters.

Each data point (pixel or superpixel feature) is assigned to the nearest centroid. The 'nearest' here is usually determined by the Euclidean distance between the feature vector of the data point and the centroid.

Updating Centroids: After all data points have been assigned to centroids, the centroids are recalculated as the mean of all data points assigned to them. This step refines the segmentation by updating the centroids to better represent the current clusters.Updating centroids by calculating the mean of assigned points is straightforward but critical. The code needs to accurately aggregate and average the features of all points assigned to each centroid.

Convergence Check: The algorithm checks if the centroids have stabilized (i.e., the shift in centroids' positions is below a certain threshold). If the centroids are stable, the algorithm has converged, and the process stops. Otherwise, steps 2 and 3 are repeated. My code uses a tolerance level to check the convergence of the algorithm. This is a crucial parameter as setting it too high can lead to premature convergence, while setting it too low can result in unnecessary computations.

Variables Affecting Outputs

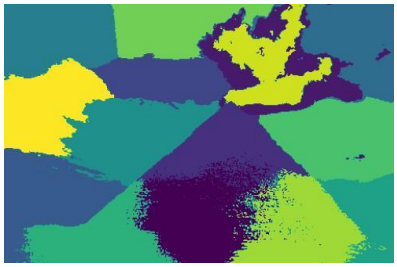
- Number of Clusters (k): Deciding the value of k is a critical choice. In the context of image segmentation, this translates to how many distinct segments you want in the image. The right choice of k can greatly affect the quality of segmentation.
- Feature Space: The dimensions and nature of the feature space (e.g., RGB values, texture, spatial coordinates) directly affect the clustering. Different features can lead to different segmentation results.

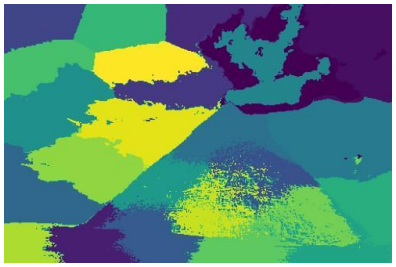
Algorithmic Considerations

- Sensitivity to Initialization: K-Means is sensitive to the initial placement of centroids. Different initializations can lead to different clustering results.
- Handling of Outliers: K-Means can be sensitive to outliers in the data since centroids are calculated as means of assigned points.
- Assumption of Cluster Shape: K-Means assumes that clusters are spherical and of similar size. This assumption might not always hold true for image data, affecting segmentation quality.

- Number of Clusters (k):

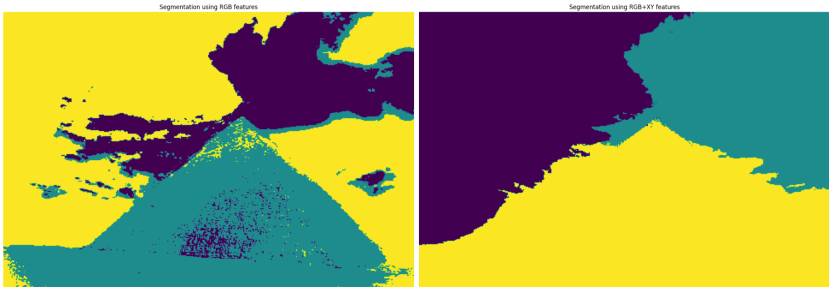

k=5


k=15

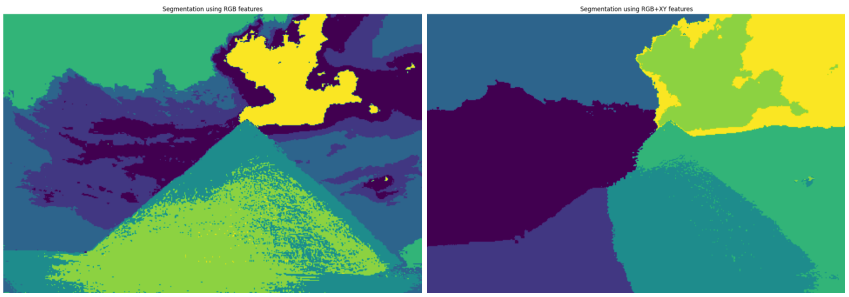

k=25

As we increase the number of clusters, the image segments get more detailed and the outlines are more noticeable. But we should look up the number of clusters and consider memory usage. We should reduce k for less memory usage. That's why I go with the k= 15.

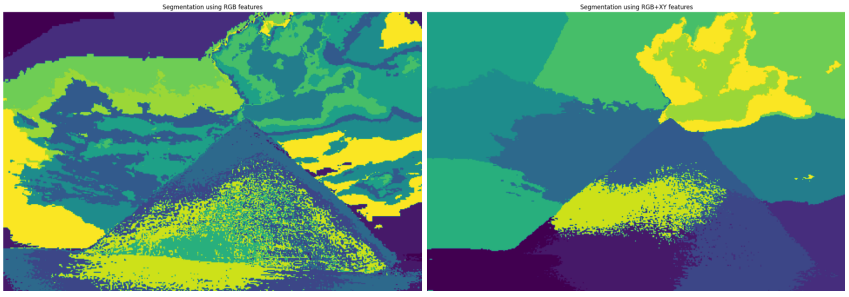
K-means clustering on pixel-level RGB and RGB+XY features:



k = 3

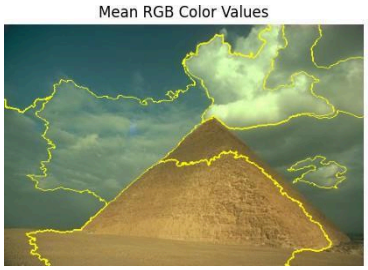


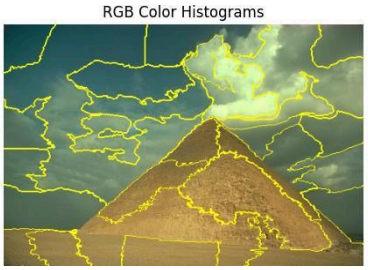
k=7

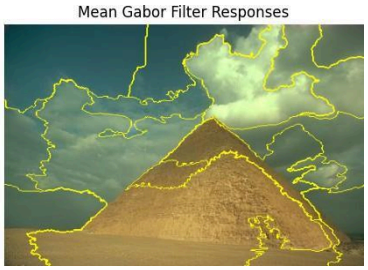


k=15

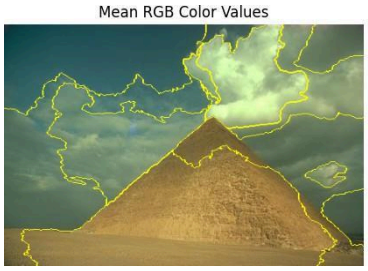
In the following parts of the assignment, I continued by combining the k-means algorithm and superpixel level features.I examined the Mean of RGB color values, RGB color histogram and Mean of Gabor filter responses features separately and observed their effects and differences on the outputs.

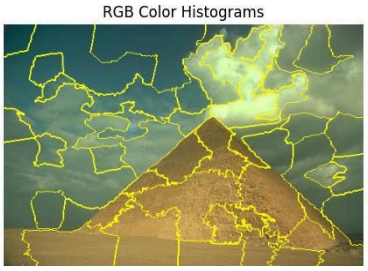


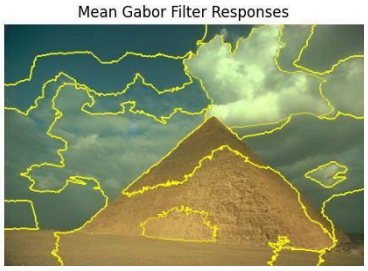




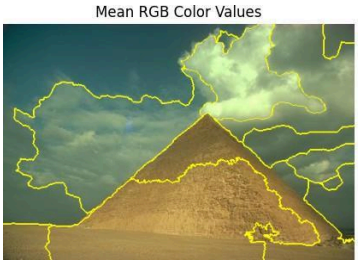
num_segments= 50

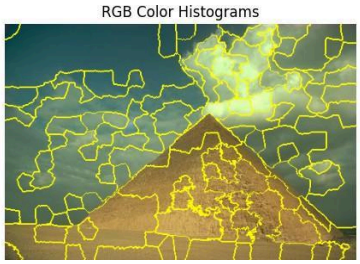


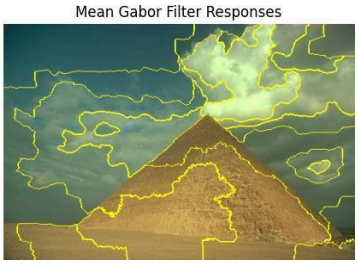




num_segments= 100

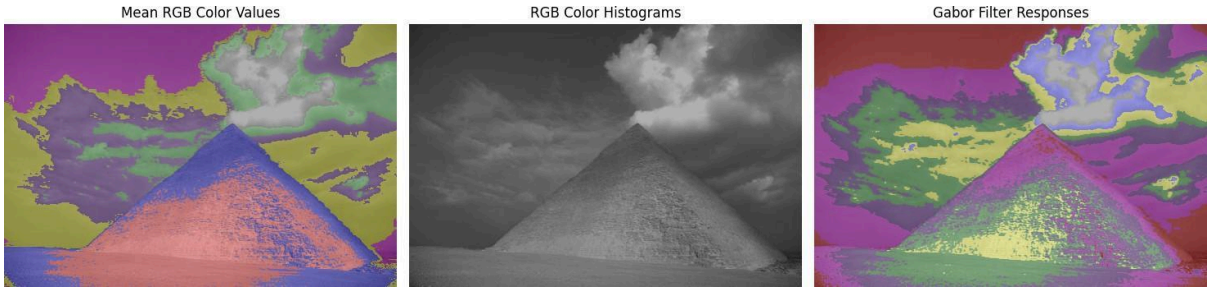






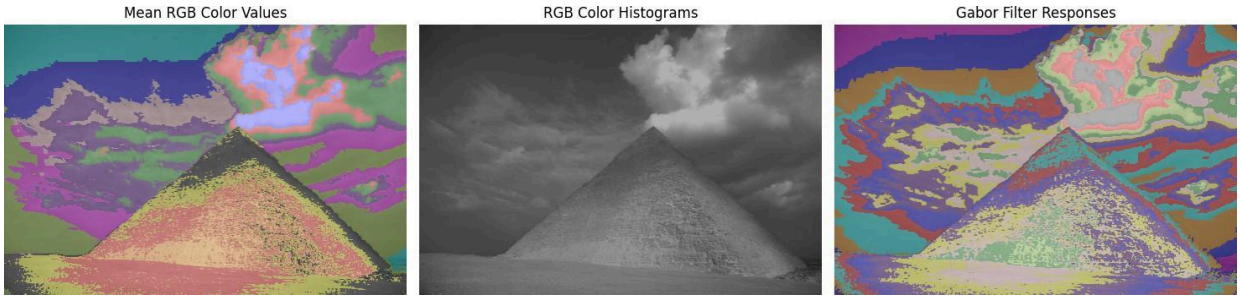
num_segments= 350

I couldn’t fix the RGB Color Histogram part because when I tried to increase the k value , my Google Colab environment crashed because of the RAM restriction.So I tried a different approach and I got this outputs:



num_segments= 300 and k =7

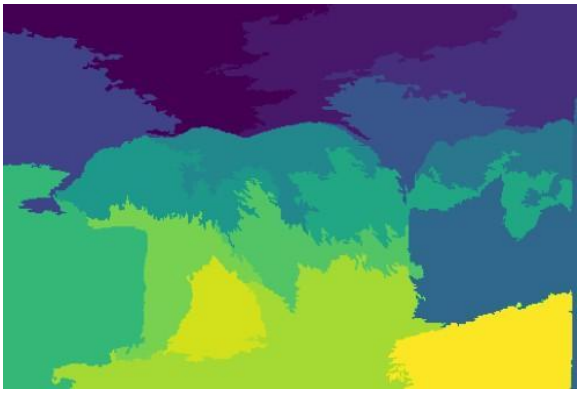
num_segments = 300 and k =3

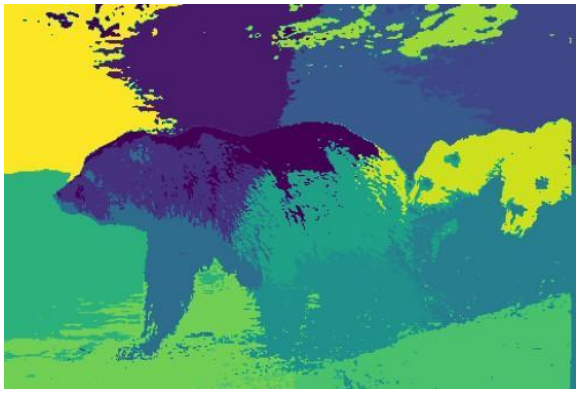


num_segments = 100 and k = 13

After all these iterations, I finally got the output as in the assignment's "d) Image Segments " part.

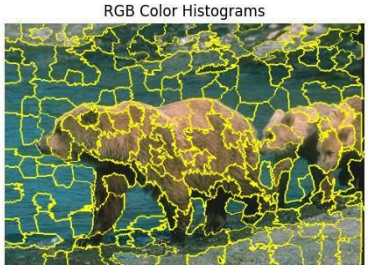
The outputs that I get from my second input:



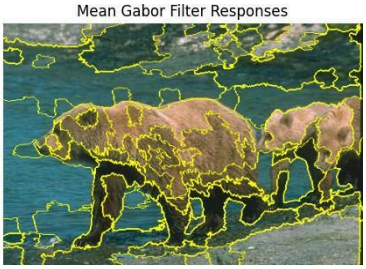


Mean RGB Color Values

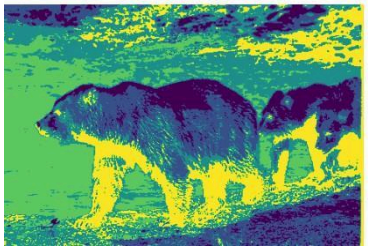
RGB Color Histograms



Mean Gabor Filter Responses

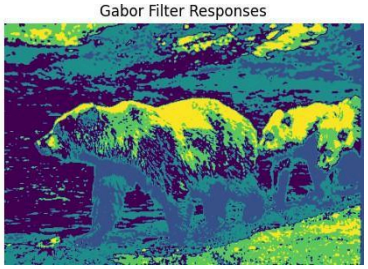


RGB Color Features



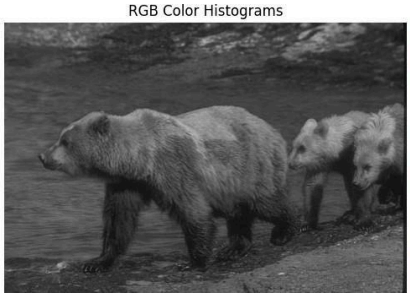
RGB Color Histograms

Gabor Filter Responses

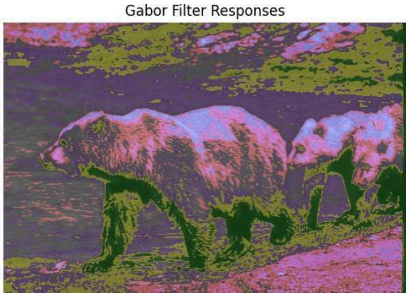


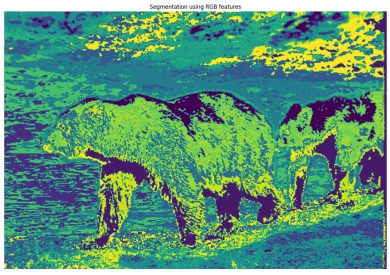
Mean RGB Color Values

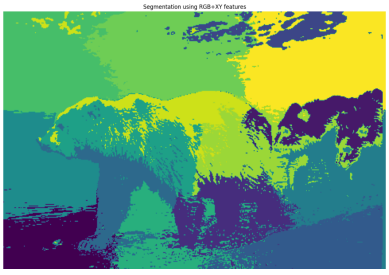
RGB Color Histograms



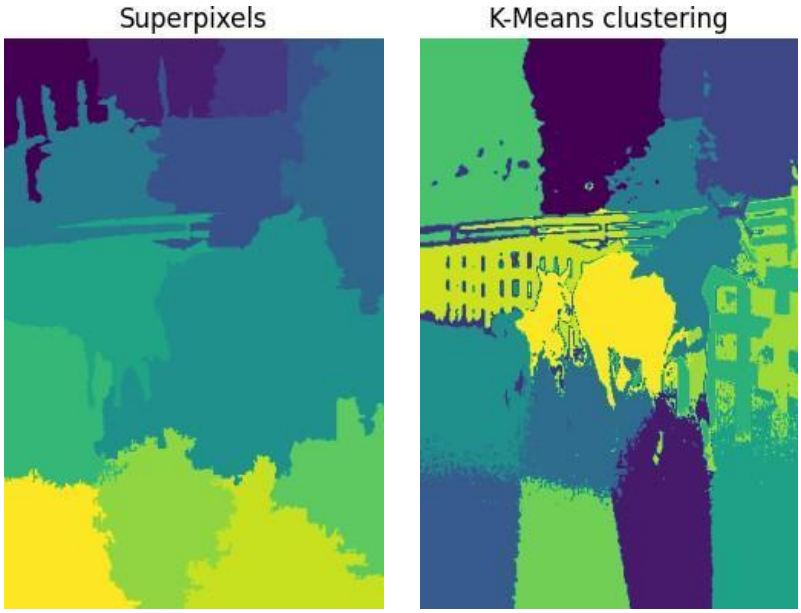
Gabor Filter Responses

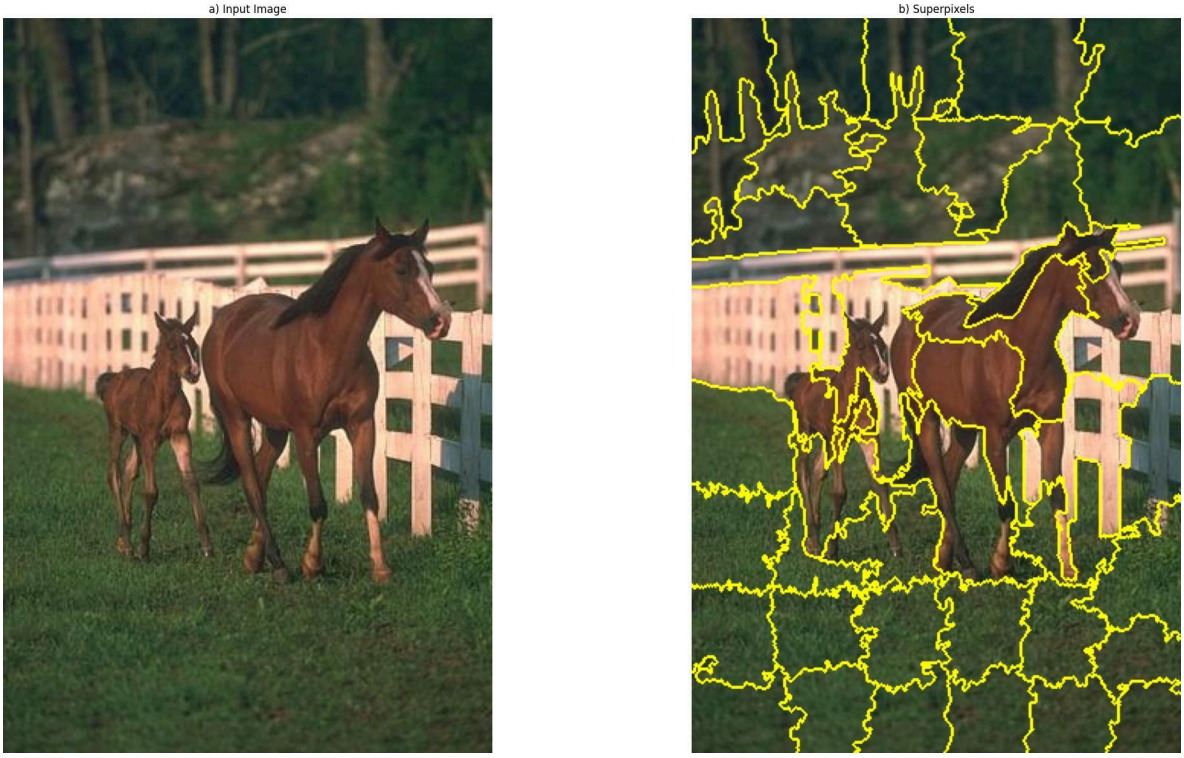




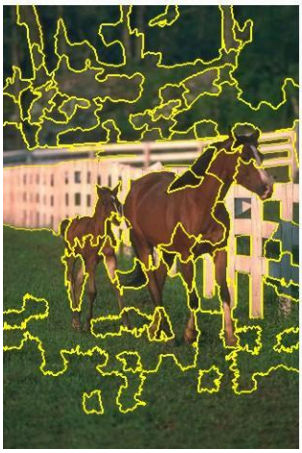


Input 3:

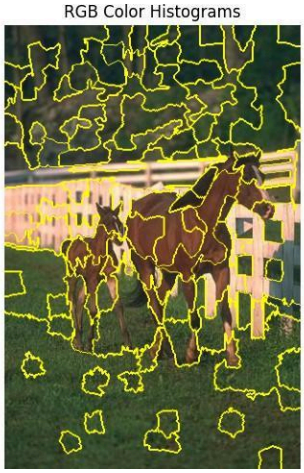

Superpixels
K-Means clustering


a) Input Image
b) Superpixels

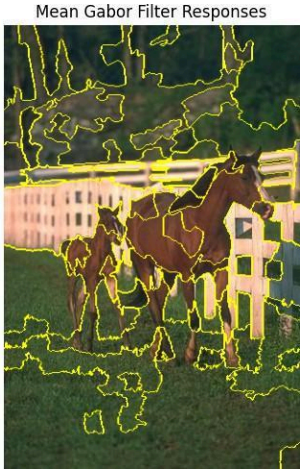
Mean RGB Color Values



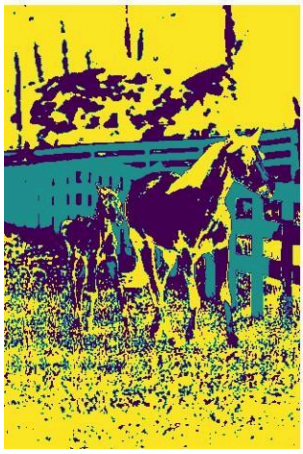
RGB Color Histograms



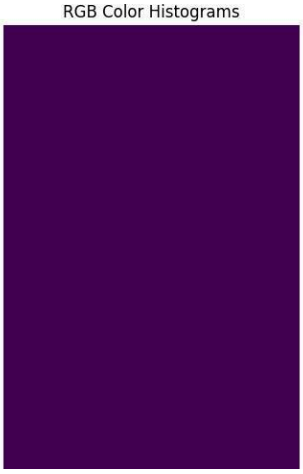
Mean Gabor Filter Responses



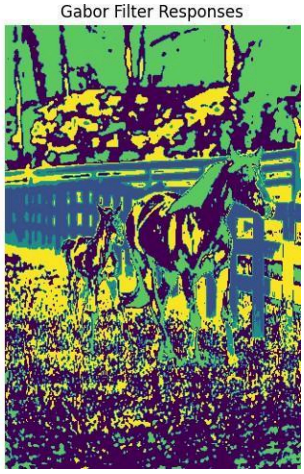
RGB Color Features



RGB Color Histograms



Gabor Filter Responses



Mean RGB Color Values

RGB Color Histograms

Gabor Filter Responses

Segmentation using RGB features

Segmentation using RGB+XY features

d) Image Segments

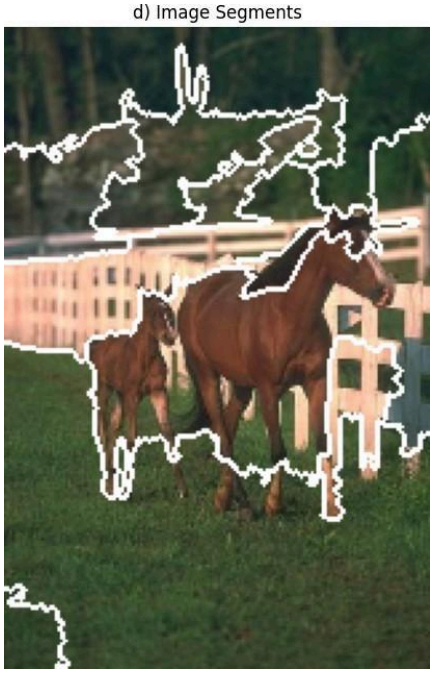
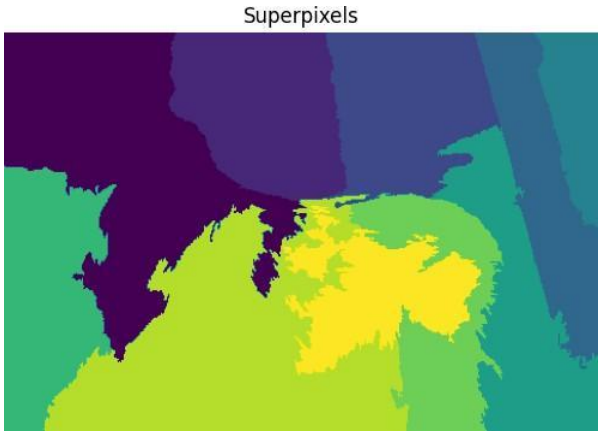
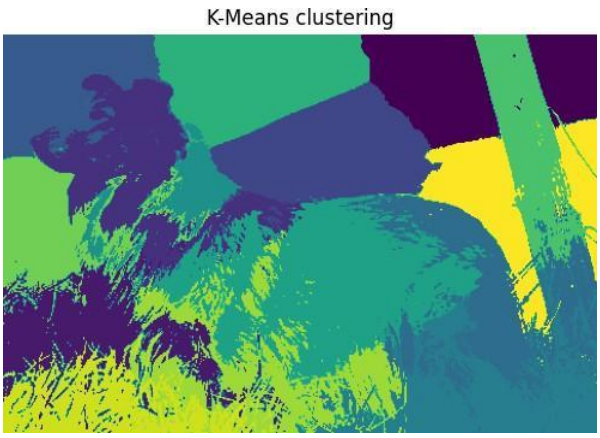
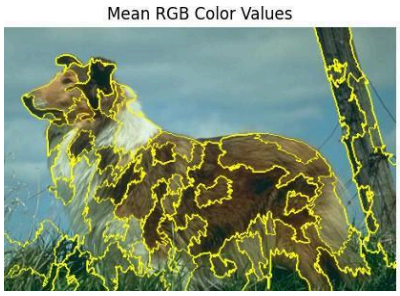


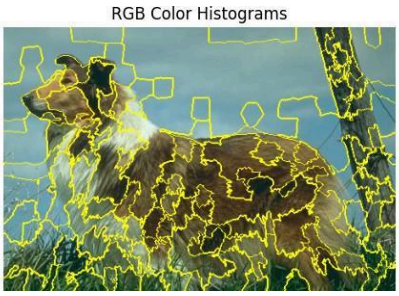
Image 4:

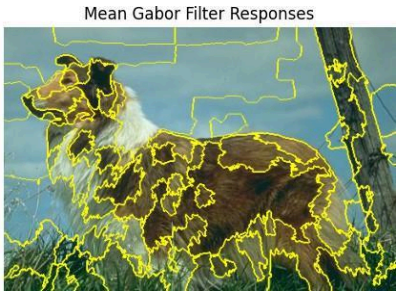

Superpixels


K-Means clustering


a) Input Image
b) Superpixels


Mean RGB Color Values

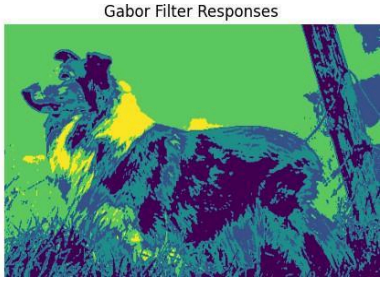

RGB Color Histograms


Mean Gabor Filter Responses

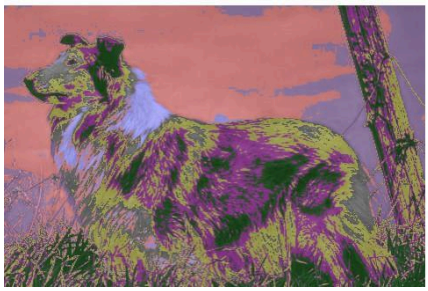
RGB Color Features

RGB Color Histograms

Gabor Filter Responses

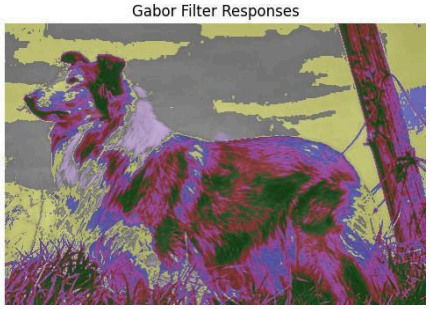


Mean RGB Color Values

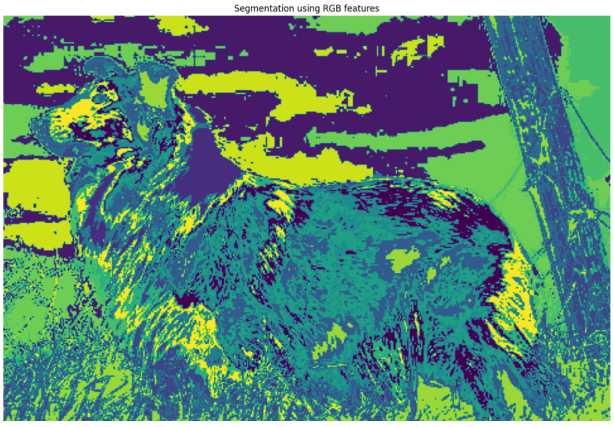


RGB Color Histograms

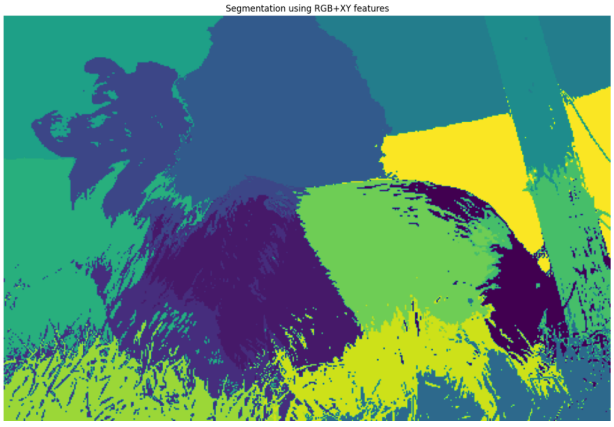
Gabor Filter Responses



Segmentation using RGB features



Segmentation using RGB+XY features



d) Image Segments

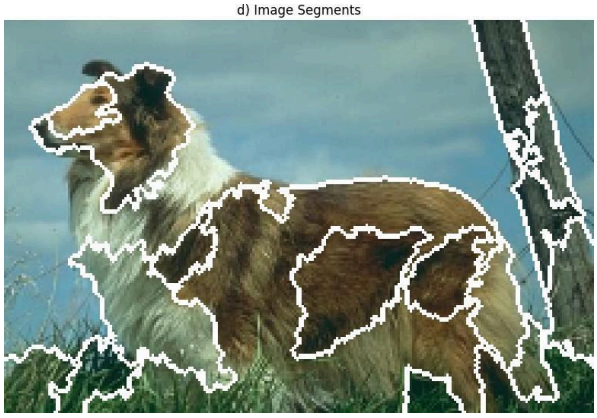
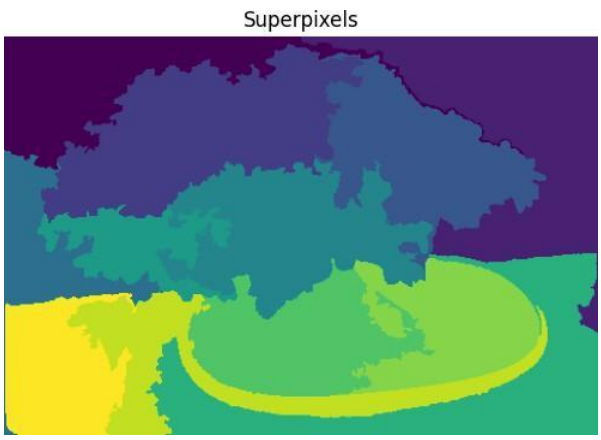
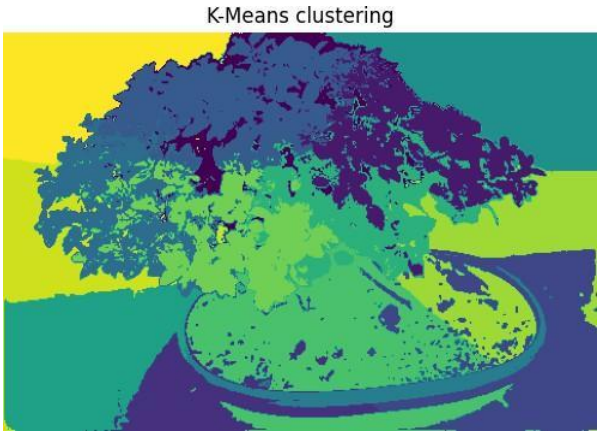
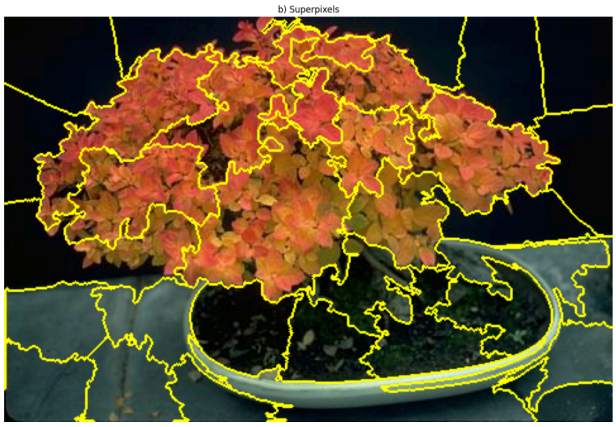


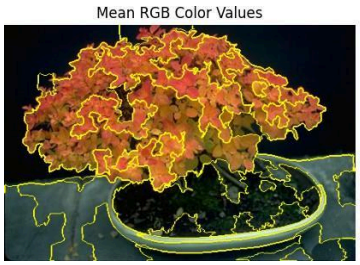
Image 5:

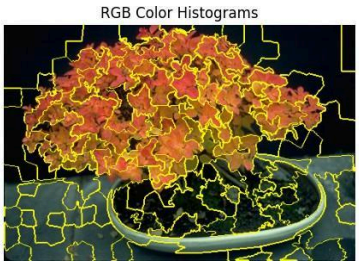

Superpixels

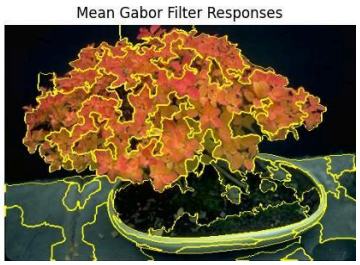

K-Means clustering


a) Input Image

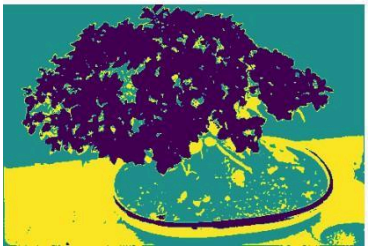

b) Superpixels


Mean RGB Color Values


RGB Color Histograms

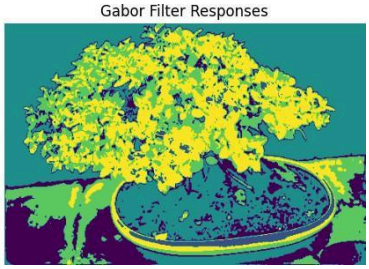

Mean Gabor Filter Responses

RGB Color Features



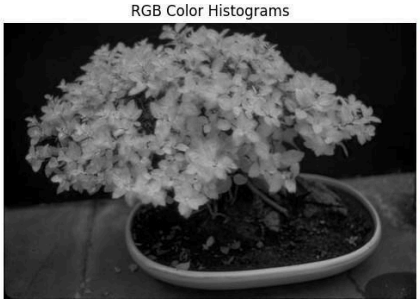
RGB Color Histograms

Gabor Filter Responses

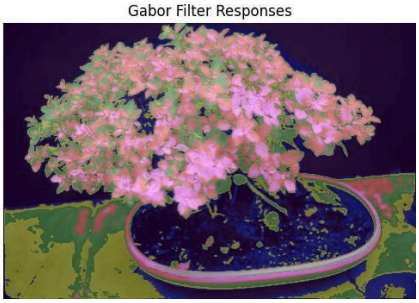


Mean RGB Color Values

RGB Color Histograms

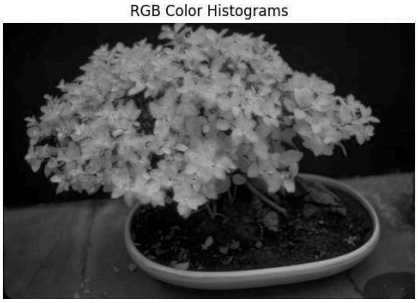


Gabor Filter Responses

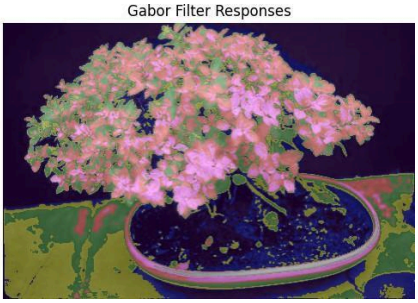


Mean RGB Color Values

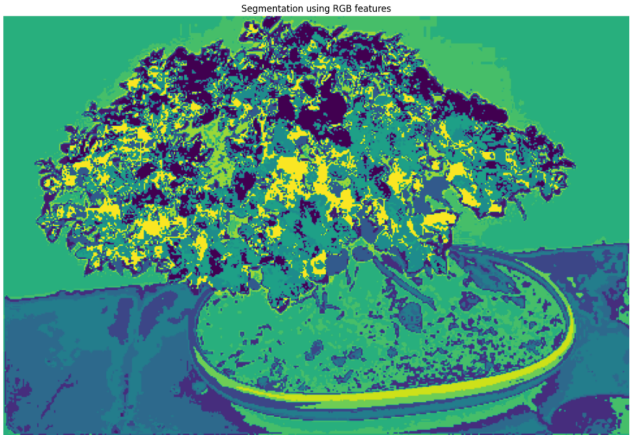
RGB Color Histograms



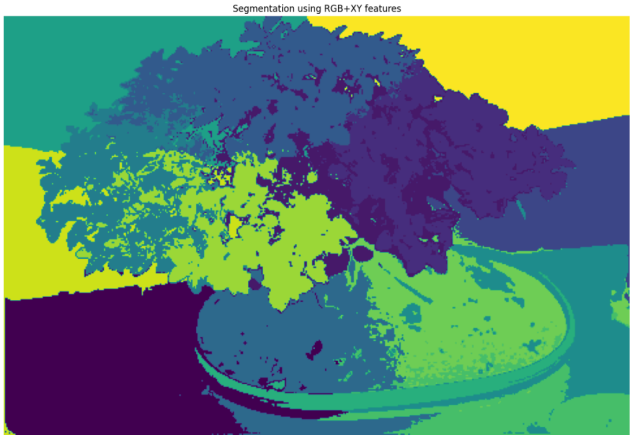
Gabor Filter Responses

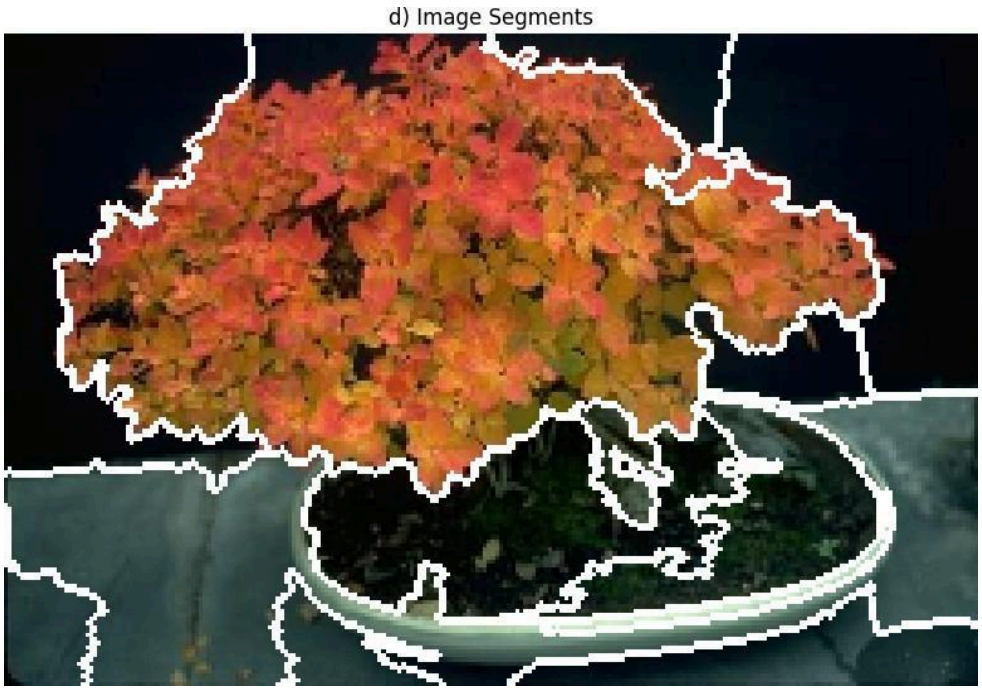


Segmentation using RGB features



Segmentation using RGB+XY features





As we see from the outputs from the 5 images, the model depends on the input edges, rgb values, etc.

So it's important to choose the input picture properly.