BBM415
ASSIGMENT-3 REPORT

K-Means Clustering for Image Segmentation

Mehmet Akif Özgür
Instructor: Aydın Kaya
TA: Burçak Asal
5 Ocak 2024

1. Introduction

a. Overview of Image Segmentation

In the realm of computer vision and image processing, the concept of image segmentation stands as a critical technique. Image segmentation involves the partitioning of an image into distinct and meaningful regions, a process pivotal in extracting valuable information from visual data. It serves as a fundamental step, allowing for a more granular understanding and analysis of images, enabling various downstream applications in fields ranging from medical imaging to object recognition in autonomous vehicles.

The significance of image segmentation lies in its ability to break down complex visual data into comprehensible units. By segmenting images into constituent parts based on common properties, such as color, texture, or intensity, this technique facilitates the isolation of objects, boundaries, or regions of interest within an image. This process is pivotal in unlocking a myriad of possibilities in image analysis and interpretation.

b. Overview of Clustering and Its Relevance in Image Analysis.

One of the prevalent approaches to image segmentation involves the utilization of clustering algorithms, among which K-Means clustering stands out as a widely employed technique. K-Means clustering, a form of unsupervised learning, partitions data into distinct clusters based on similarity, making it an ideal candidate for segmenting images into meaningful components.

This report embarks on a comprehensive exploration of employing the K-Means clustering algorithm for image segmentation. The focus is two-fold, encompassing the utilization of pixel-level features as well as superpixel-level features for segmentation. Pixel-level features involve extracting information directly from individual pixels, while superpixel-level features involve grouping pixels into larger, more coherent units before feature extraction.

c. Objective

The primary objectives of this report are investigating the efficacy of various feature extraction methods, such as RGB color features, spatial location information, mean color values, color histograms, and Gabor filter responses.

Assessing the impact of different clustering parameters, particularly the number of clusters (k), on the segmentation results.

Analyzing the influence of superpixel sizes on the clustering outcomes and evaluating their relevance in achieving accurate segmentation.

Through experimentation and analysis, this report endeavors to provide insights into the effectiveness, limitations, and nuances of employing K-Means clustering for image segmentation. By exploring different feature representations and their impact on clustering results, this study aims to contribute to a deeper understanding of the role of clustering algorithms in the realm of image segmentation.

2. Methodology

a. Data Collection

The image collection process was one of the crucial parts of this project. I tried to collect as many different images as possible. I tried to add images with many objects and images with a single object into my dataset. In this way, I hoped that different k values would give more successful results in different images. I also noticed differences in color intensity in the images. While in some images there was not much difference in color intensity between the object and the background, in some images this difference was quite large.

b. Explanation of K-Means Algorithm

So, imagine I have a bunch of data points on a graph, and I want to group them into K clusters. The first thing I do is randomly pick K points on the graph as the starting centers for these clusters. We call these points centroids.

Then, I measure the distance between each data point and these centroids. I typically use the straight-line distance (Euclidean distance) to figure out how far each point is from each centroid.

After that, I assign each data point to the cluster whose centroid is closest. This step is all about grouping each data point with the centroid it's nearest to.

Once the points are grouped, I recalculate the centroids for these clusters. How? By finding the average position of all the points in each cluster. This move effectively shifts the centroids closer to the middle of their clusters.

Now, here's where the loop begins. I keep repeating these steps: measuring distances, reassigning points to clusters based on new centroids, and recalculating centroids. This loop goes on until either the centroids don't change much between iterations or until I reach a set number of iterations.

Finally, I end up with K clusters, each with its centroid as the center point. The data points are grouped together based on their similarity to these centroids.

The whole idea behind K-means is to minimize the difference between the points in a cluster and their centroid, making sure each point is as close as possible to the center of its cluster.

However, there's a catch: K-means might find a local best solution, meaning the final clusters could depend on where I initially placed those centroids. That's why running the algorithm multiple times and choosing the best result can help get a more accurate clustering. To avoid this problem, I used np.random.seed. Now, no matter how many times I run the algorithm, my starting values are always the same.

c. Description of Feature Extractions

RGB Color Feature

In this section, I gave the RGB values of each pixel in the image to the k-means algorithm.

RGB Color and Spatial Location Feature

In this section, I gave the k-means algorithm the RGB values of each pixel in the image and the x and y coordinates of that pixel in 2-dimensional space.

Mean of RGB Color Values

In this section, I gave the k-means algorithm the average RGB values of the pixels in each superpixel I created.

RGB Color Histogram

In this section, first of all, I gave the RGB values of each superpixel separately into the np.histogram function and received the output of how many times each color value was found in that superpixel. Then I combined these separate RGB values and gave this output to the k-means algorithm.

Mean of Gabor Filter Responses

In this section, I put the RGB channels of the image into Gabor filters separately. I used a total of 110 Gabor filters in this project. The Gabor filters I use consist of all combinations of [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0] values as frequency parameters and [30,60,90,120,150,180,210,240,270,300,330,360] values as theta parameters. Then, I averaged the real images returned from these filters and combined them in RGB values. I gave these values, which are the average of the outputs of the Gabor filters, to the k-means algorithm.

d. Details on parameter tuning for optimal segmentation.

In this experimental project, I gave [2,3,4,5,6] as k value to the kmeans algorithm. In the sections containing superpixels where I used the slic() function, I used [100,200,500] values for the n_segment parameter. Of course, as stated in the documentation, I could not always divide the image into the segments I gave, but approximately similar values were returned from the slic() function.

3. Result and Analysis

a. All Images at Their Best

I worked with a total of 13 images in this project. I tried to segment these 13 images with 5 different future extractors, 5 different k values and 3 different n_segment parameters. In total, I obtained 715 final images. These final images gave good results at some parameter values, but they showed poor segmentation at some parameter values. Since I cannot show all 715 final images and over 2000 images along with the input image, superpixel and image segments in this report, I will show the best results of some images for 5 feature extractors. You can access all images from the Google Drive link I shared.

<u>Soldier</u>

RGB color feature

Input Image

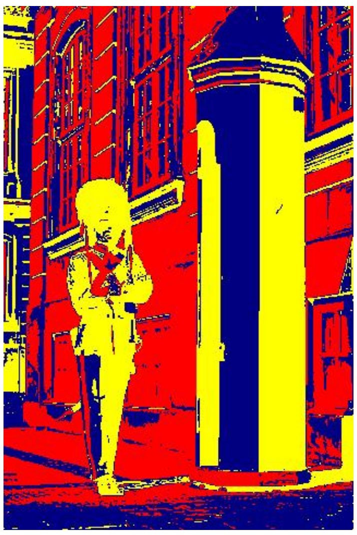


Image Labels (K=3)

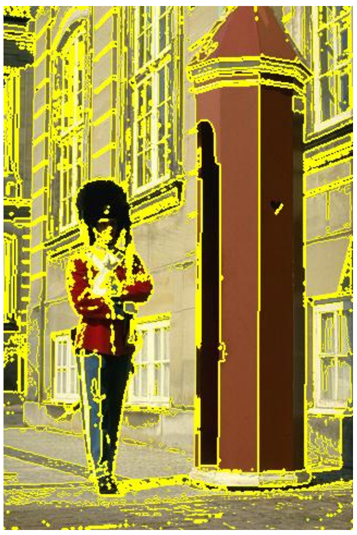


Image Segments (K=3)

RGB color and spatial location feature

Input Image

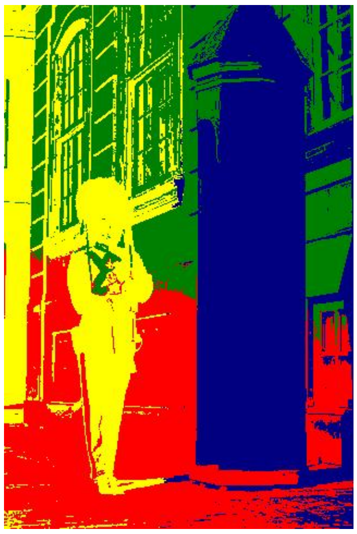


Image Labels (K=4)

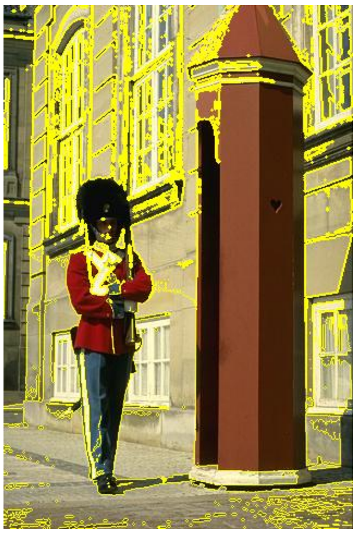
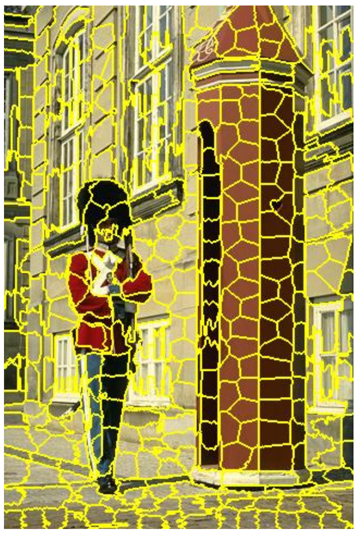


Image Segments (K=4)

Mean of RGB color values

Input Image


Superpixels (Seg=342)

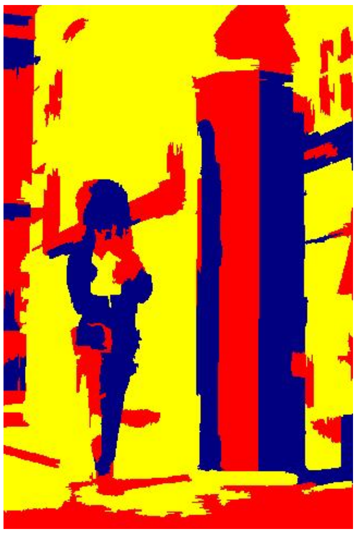


Image Labels (K=3)

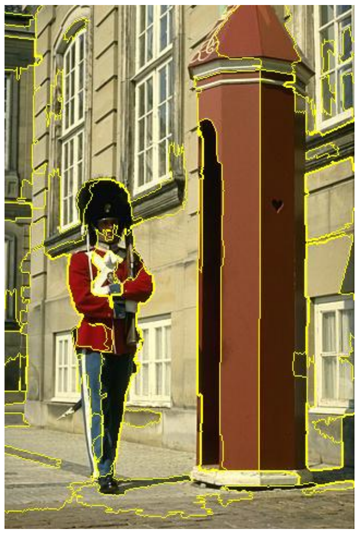
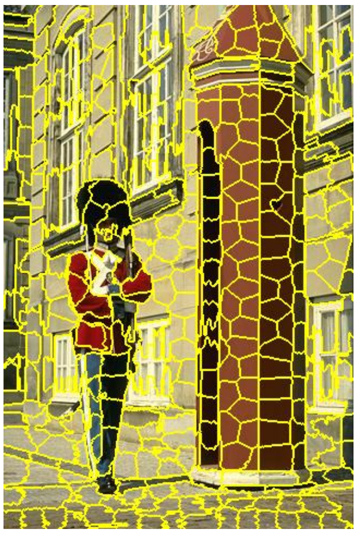


Image Segments (K=3)

RGB color histogram

Input Image


Superpixels (Seg=342)

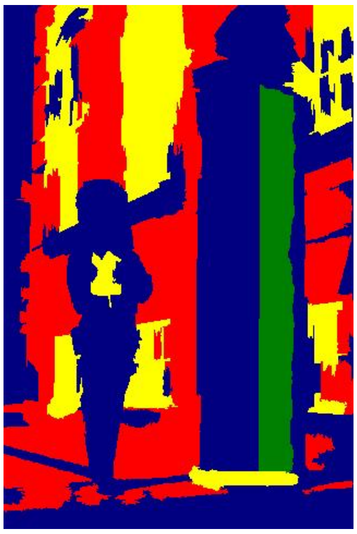


Image Labels (K=4)

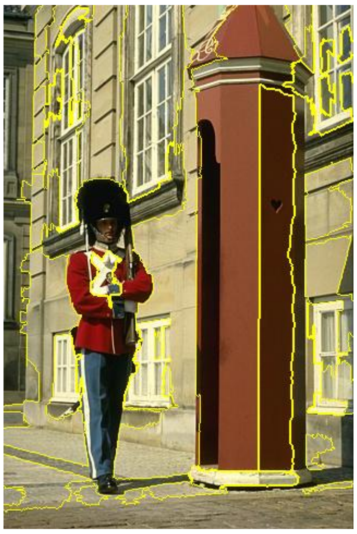
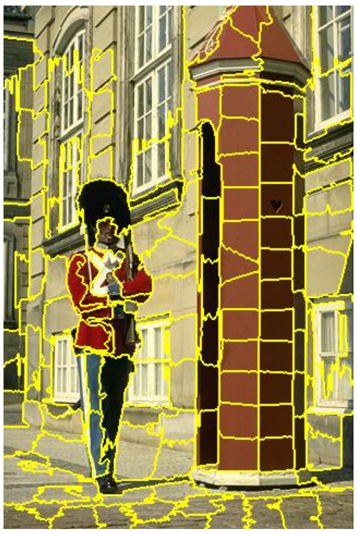


Image Segments (K=4)

Mean of Gabor filter responses

Input Image


Superpixels (Seg=122)

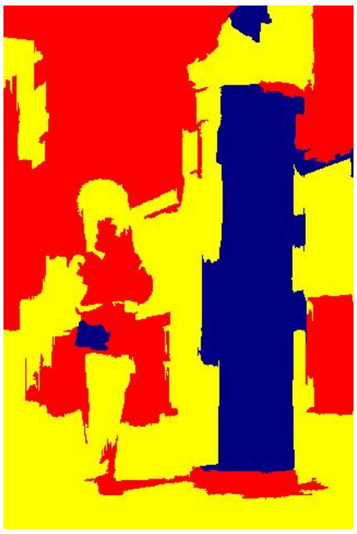


Image Labels (K=3)

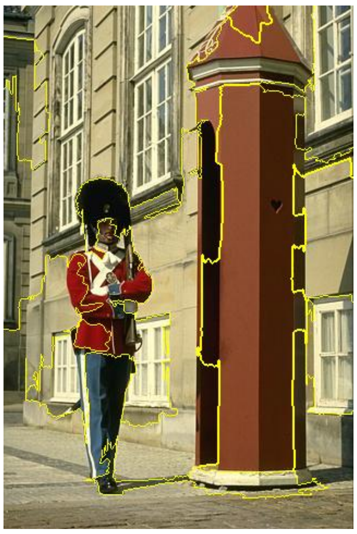


Image Segments (K=3)

Deer

RGB color feature

Input Image

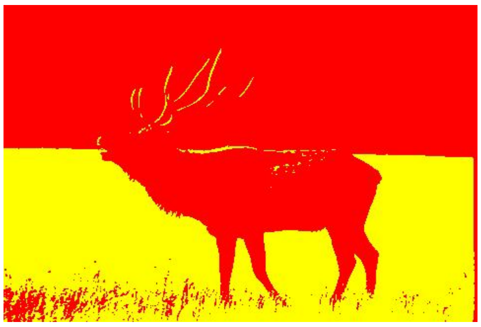


Image Labels (K=2)

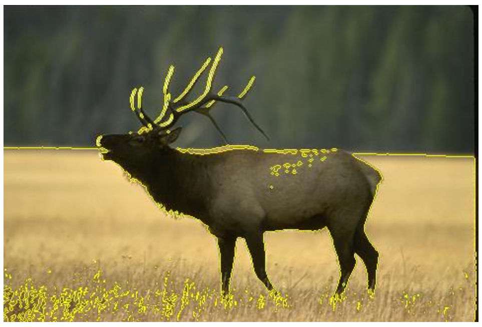


Image Segments (K=2)

RGB color and spatial location feature

Input Image

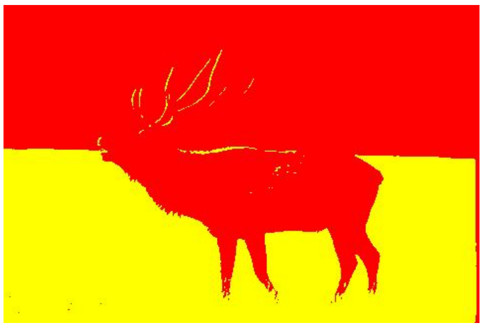


Image Labels (K=2)

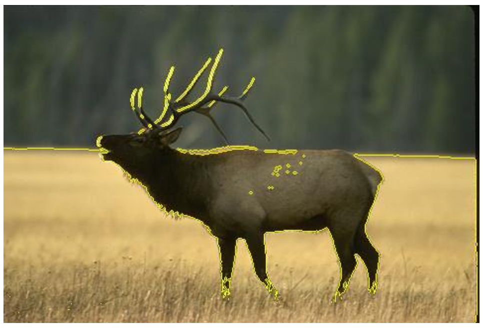
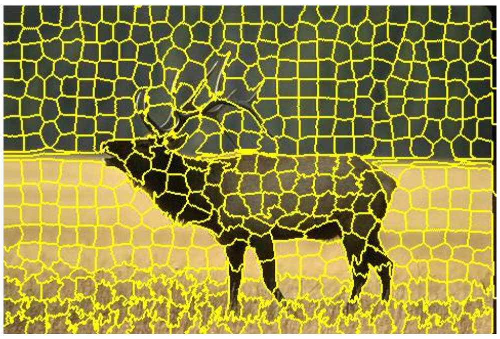


Image Segments (K=2)

Mean of RGB color values

Input Image


Superpixels (Seg=429)

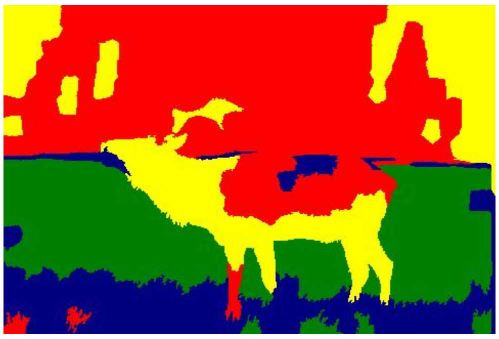


Image Labels (K=4)

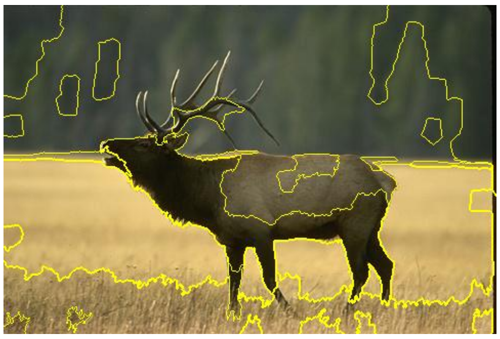
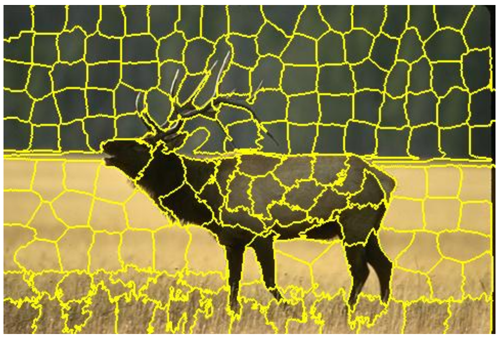


Image Segments (K=4)

RGB color histogram

Input Image


Superpixels (Seg=161)

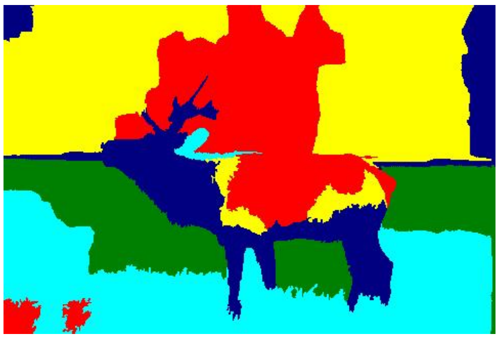


Image Labels (K=5)

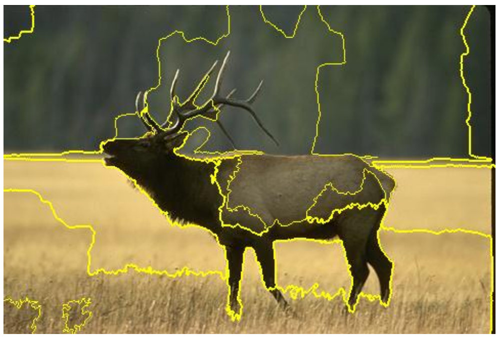
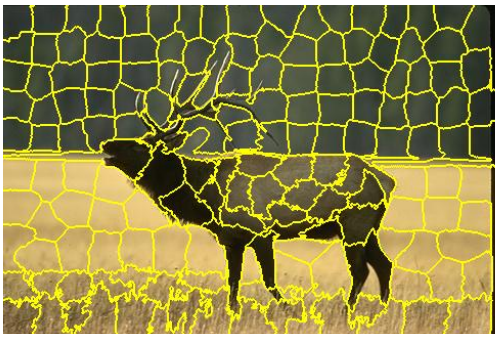


Image Segments (K=5)

Mean of Gabor filter responses

Input Image


Superpixels (Seg=161)

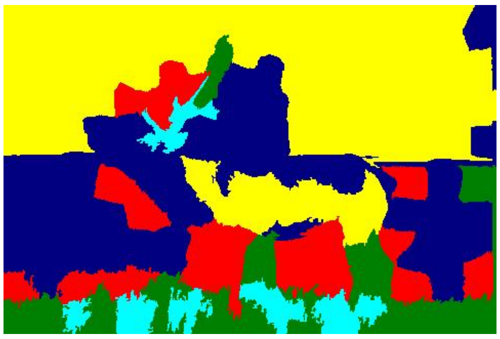


Image Labels (K=5)

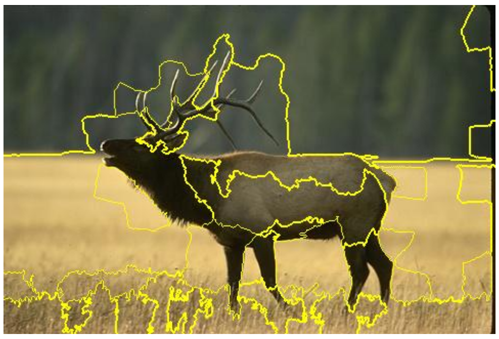


Image Segments (K=5)

<u>Plane</u>

RGB color feature

Input Image

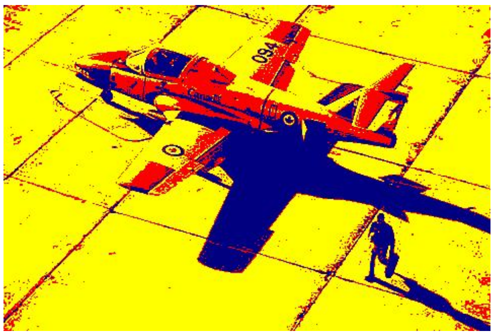


Image Labels (K=3)

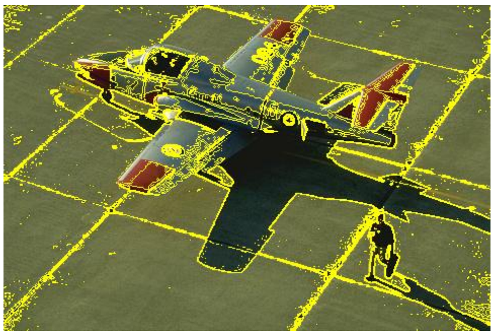


Image Segments (K=3)

RGB color and spatial location feature

Input Image

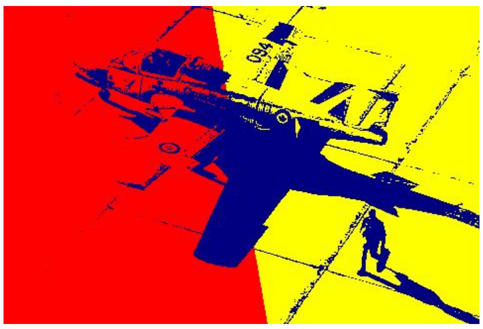


Image Labels (K=3)

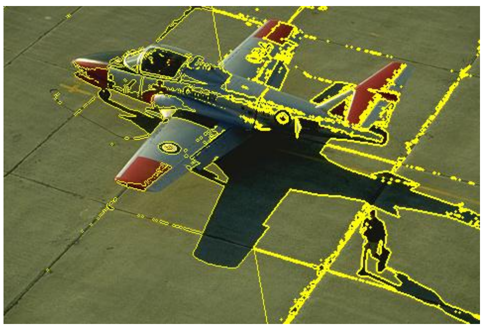
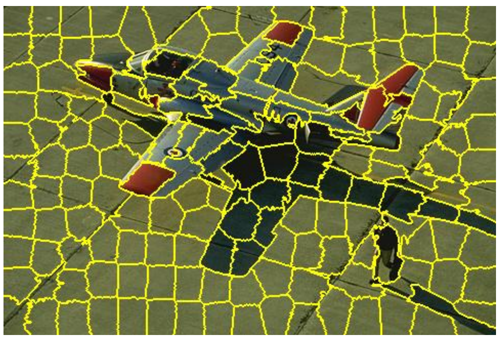


Image Segments (K=3)

Mean of RGB color values

Input Image


Superpixels (Seg=163)

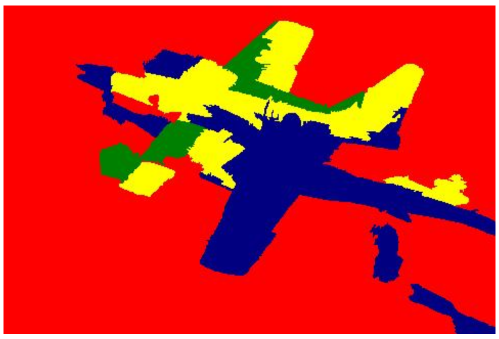


Image Labels (K=4)

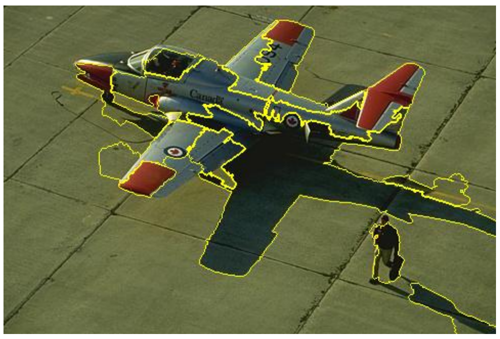
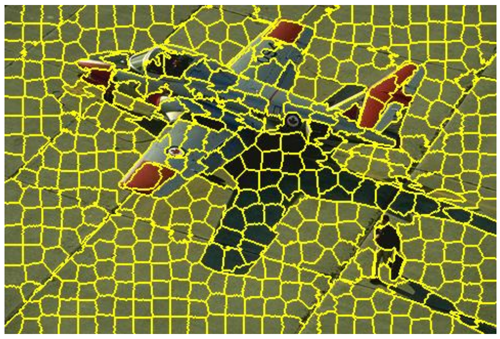


Image Segments (K=4)

RGB color histogram

Input Image


Superpixels (Seg=421)

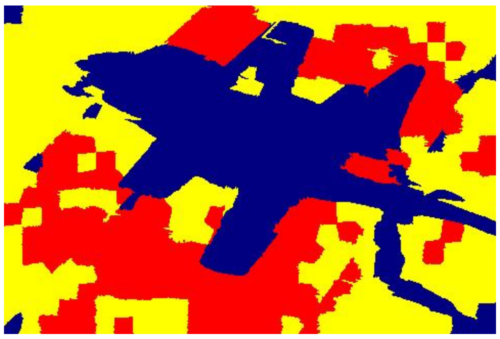


Image Labels (K=5)

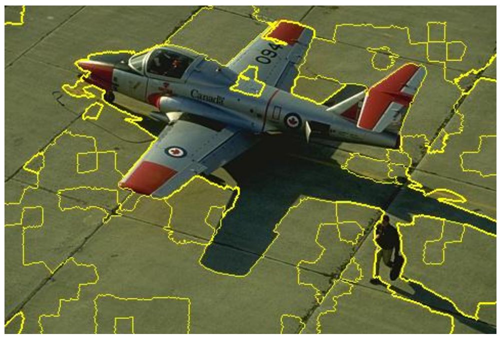
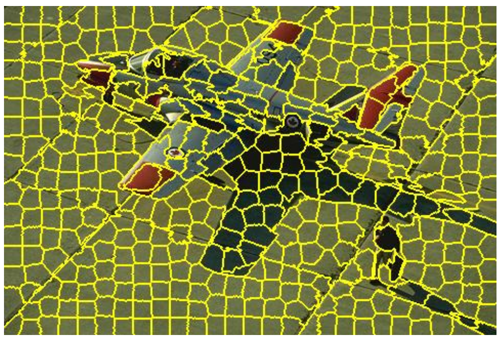


Image Segments (K=5)

Mean of Gabor filter responses

Input Image


Superpixels (Seg=421)

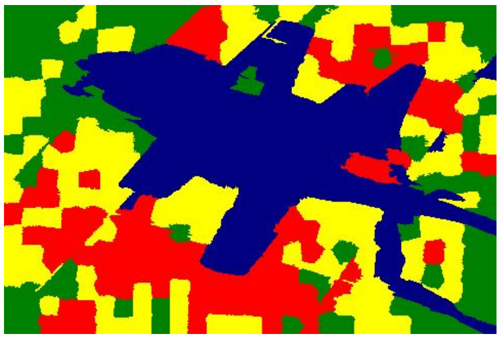


Image Labels (K=4)

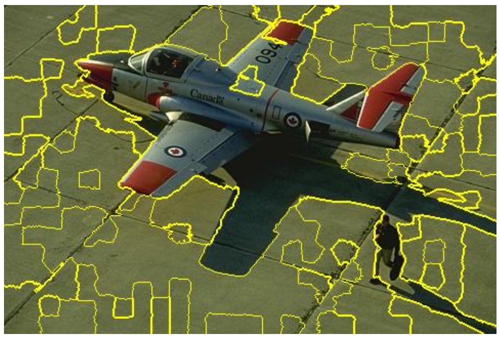


Image Segments (K=4)

Human

RGB color feature

Input Image

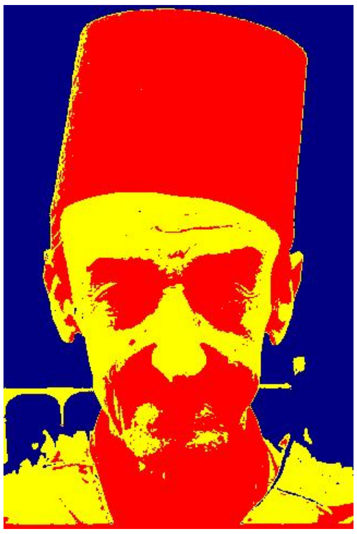


Image Labels (K=3)

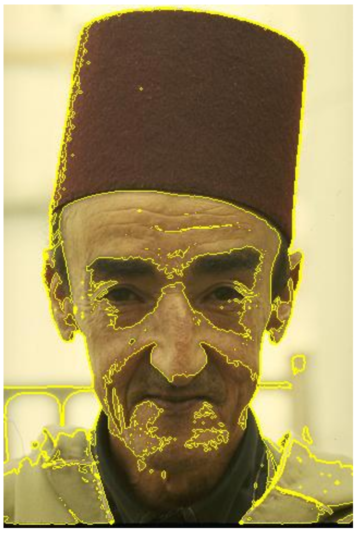


Image Segments (K=3)

RGB color and spatial location feature

Input Image

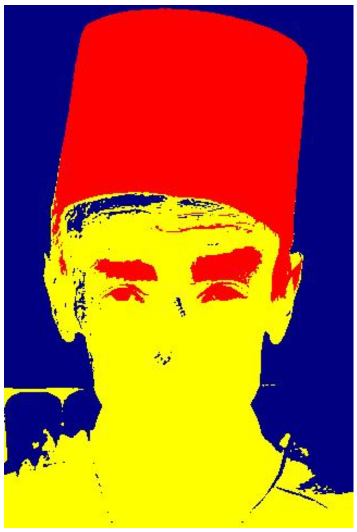


Image Labels (K=3)

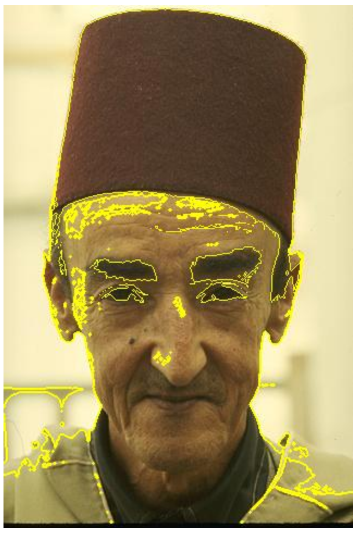
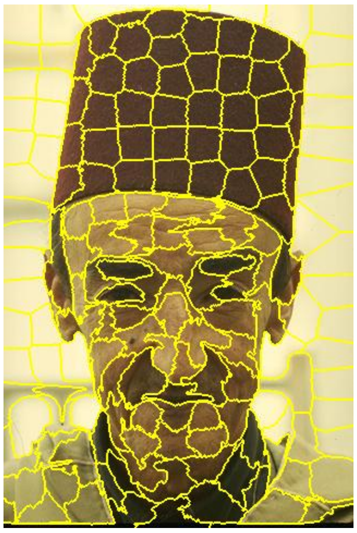


Image Segments (K=3)

Mean of RGB color values

Input Image


Superpixels (Seg=161)

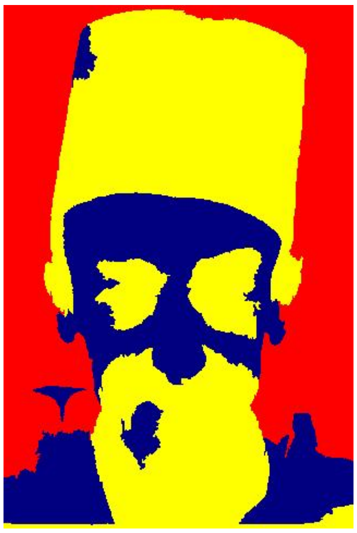


Image Labels (K=3)

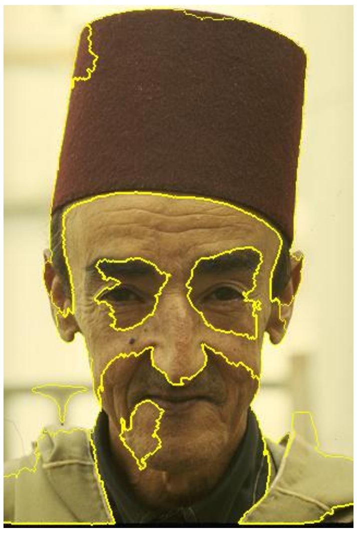
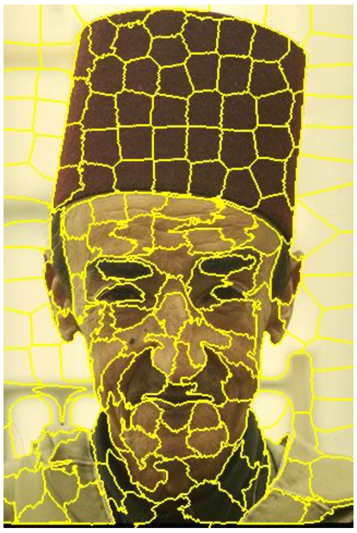


Image Segments (K=3)

RGB color histogram

Input Image


Superpixels (Seg=161)

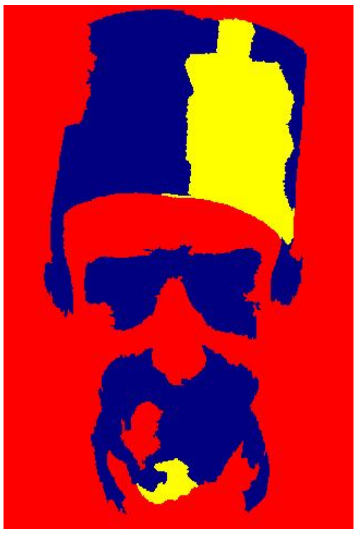


Image Labels (K=3)

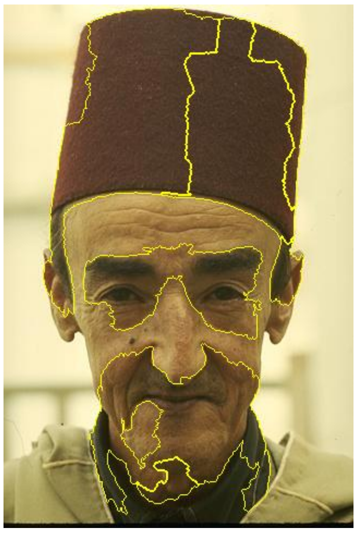
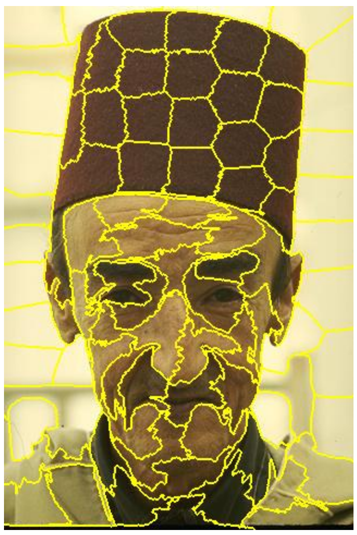


Image Segments (K=3)

Mean of Gabor filter responses

Input Image


Superpixels (Seg=82)

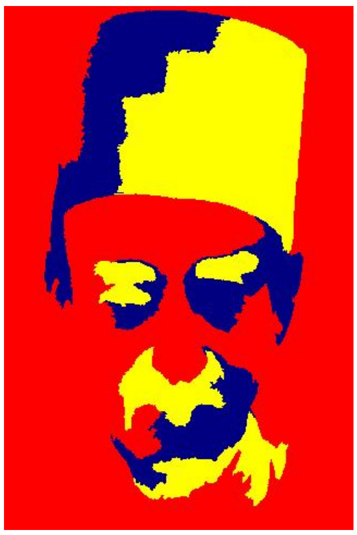


Image Labels (K=3)

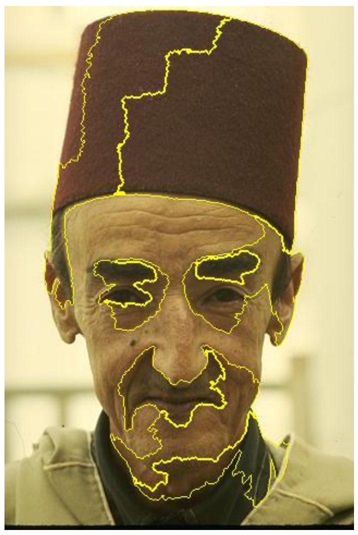


Image Segments (K=3)

Apple

RGB color feature

Input Image

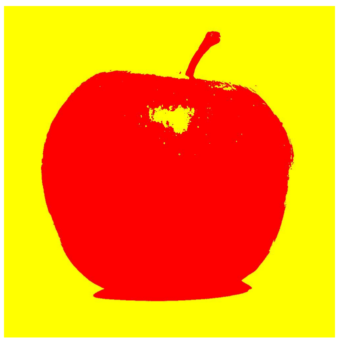


Image Labels (K=2)

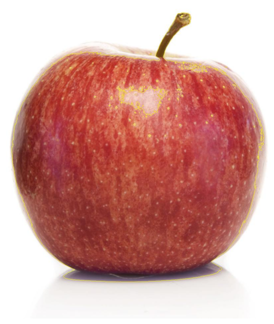


Image Segments (K=2)

RGB color and spatial location feature

Input Image

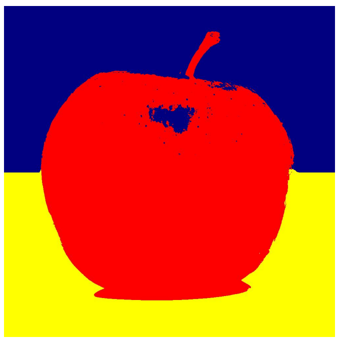


Image Labels (K=3)

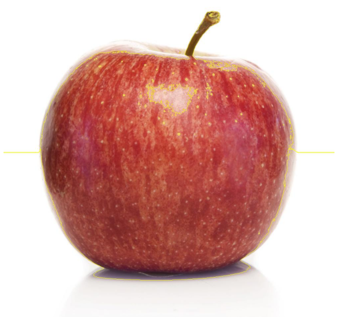
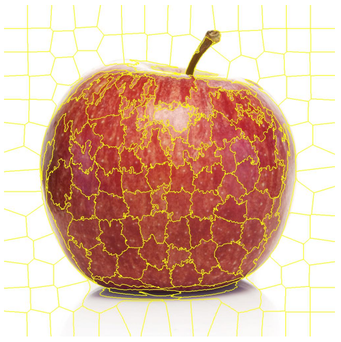


Image Segments (K=3)

Mean of RGB color values

Input Image


Superpixels (Seg=174)

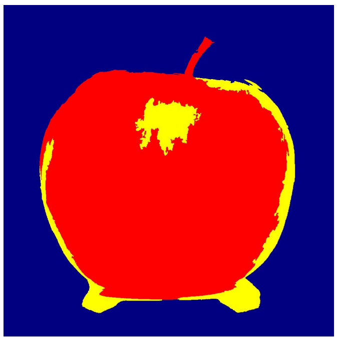


Image Labels (K=3)

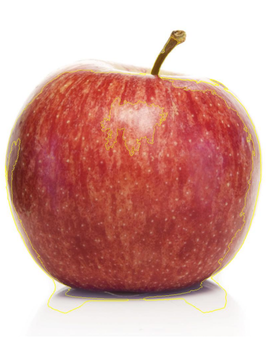
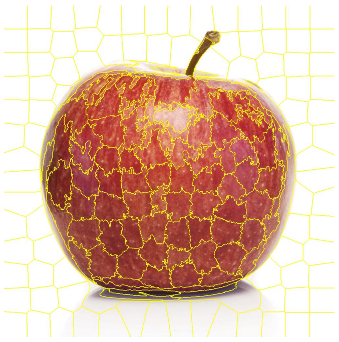


Image Segments (K=3)

RGB color histogram

Input Image


Superpixels (Seg=174)

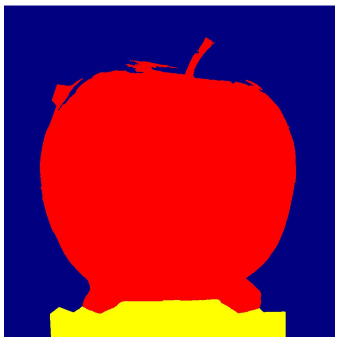


Image Labels (K=3)

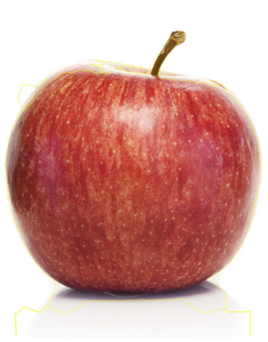
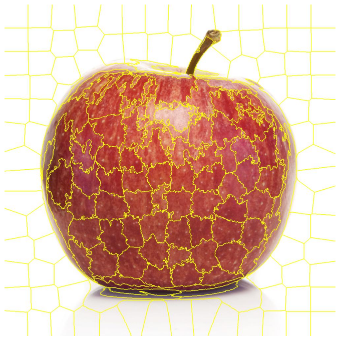


Image Segments (K=3)

Mean of Gabor filter responses

Input Image


Superpixels (Seg=174)

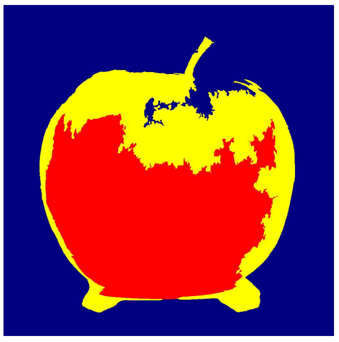


Image Labels (K=3)

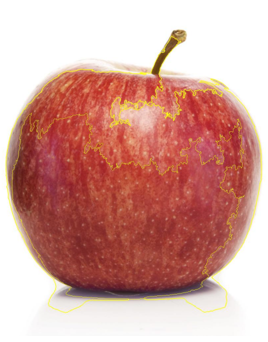
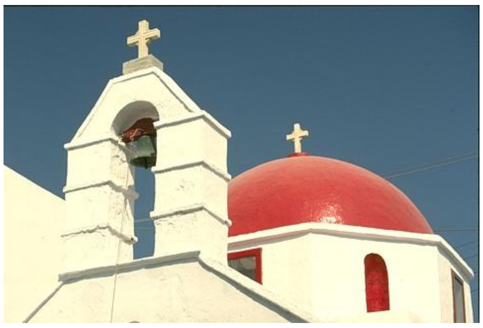


Image Segments (K=3)

b. Comparison in Terms of Number of n_segments

In this section, we will examine the effect of the number of n_segments on the change of features extracted using superpixels. Here we will try to explain the event only with visuals. In the "Conclusion and Discussion" section, we will talk about the effect of n_segment on features in more detail. I made this comparison for each image. In order not to enlarge the report too much, I will show 3 examples here. You can access all comparisons from the Google Drive link I shared.

Church (Mean of RGB color / K = 3)



Input Image

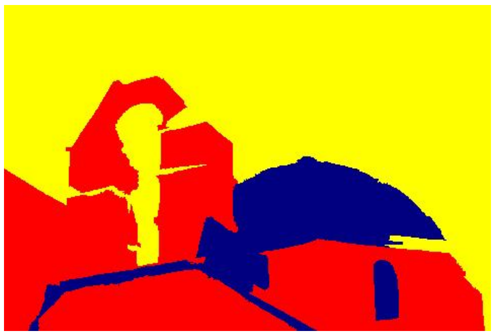


Image Labels (Seg=79)

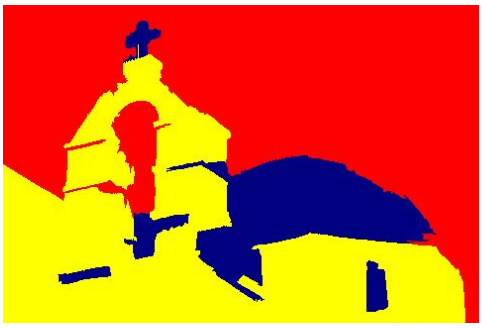


Image Labels (Seg=149)

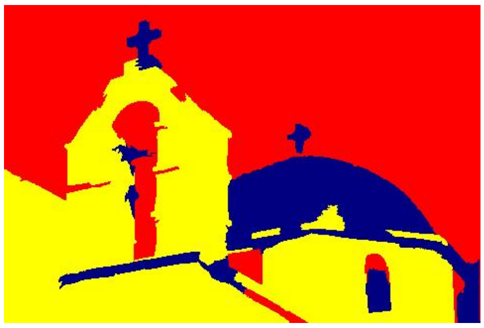


Image Labels (Seg=411)

Short Tree (Mean of Gabor filter responses / K = 4)

Input Image

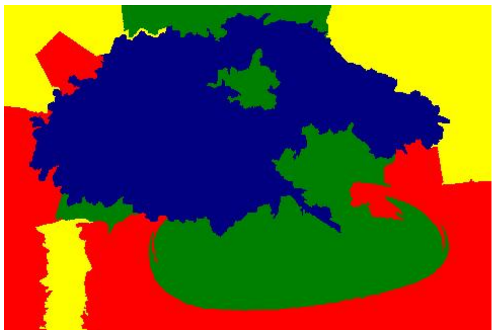


Image Labels (Seg=68)

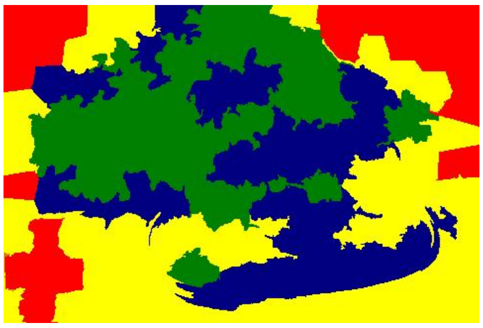


Image Labels (Seg=140)

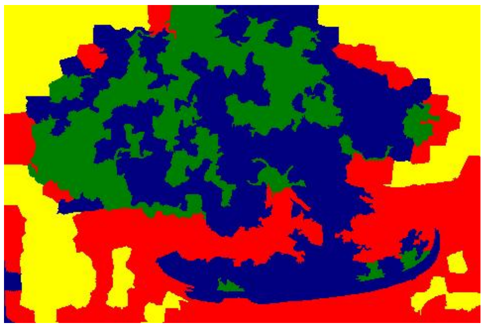


Image Labels (Seg=395)

Pyramid (Mean of RGB color / K = 3)

Input Image

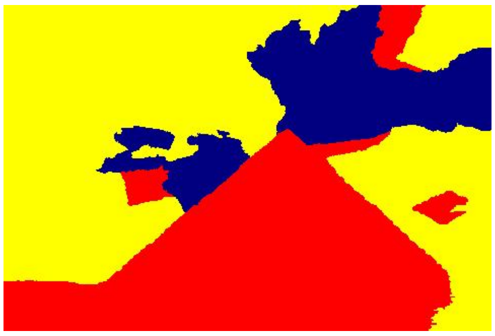


Image Labels (Seg=93)

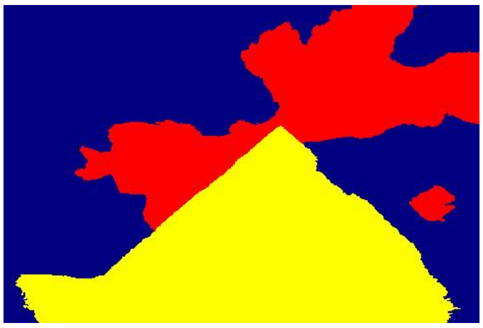


Image Labels (Seg=177)

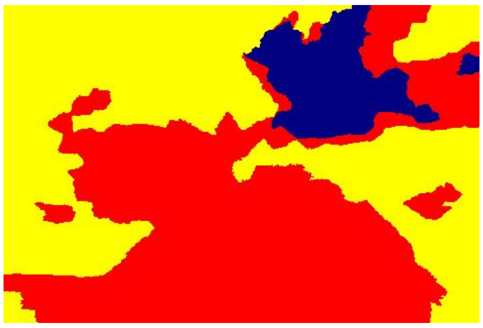


Image Labels (Seg=466)

4. Conclusion and Discussion

a. Evaluation of Obtained Experiments

Change of k-value

The change in the k value used for the k means algorithm varies from image to image. Because the optimum number of threads to separate each image may be different. Therefore, I tried 5 different k values for each image during this experiment. Different k values gave better results for optimal segmentation in each image.

Change of n_segment value

I used the 'n_segment' value when extracting superpixel-level features. In my experiments, I included three different 'n_segment' values for each image. Varying the 'n_segment' number depending on the complexity level of the image played a significant role in determining the optimal segmentation level. In the 'Comparison in Terms of Number of n_segments' section above, I demonstrated visually how changing the 'n_segment' value would result in different outcomes.

Generally, when increasing the 'n_segment' number, we obtain a more detailed result. This is because as the 'n_segment' value increases, the number of superpixels also increases, and each superpixel covers a smaller area. Therefore, superpixels dominating a smaller area enhance the details in the final image.

In conclusion, increasing the 'n_segment' number makes the output closer to a pixel-level feature approach, while decreasing the 'n_segment' number makes the output less detailed.

Comparing pixel-level and superpixel-level (detail and speed)

Pixel-level feature extraction methods yield a more detailed result compared to superpixel-level feature extraction methods. The reason behind this is that pixel-level feature extraction methods feed the characteristics of each pixel into the k-means algorithm, whereas superpixel-level feature extraction methods send pixels as groups to the k-means algorithm. The k-means algorithm influenced by each individual pixel returns a more detailed and intricate label array, whereas the k-means algorithm influenced by pixels grouped together yields a simpler and shallower label array.

Additionally, a significant difference between pixel-level feature extraction methods and superpixel-level feature extraction methods is speed. Superpixel-level feature extraction methods work faster with the k-means algorithm compared to pixel-level feature extraction methods because they send less data to the k-means algorithm. As I mentioned earlier, although superpixel-level feature extraction methods may lead to a loss of details, they provide a significant advantage in terms of speed.

Interpretation of pixel-level feature extraction methods

Pixel-level feature extraction methods are divided into two categories: 'RGB color future' and 'RGB color and spatial location feature.' 'RGB color future' performs segmentation solely based on RGB, while 'RGB color and spatial location feature' incorporates the spatial locations of pixels in addition to RGB. In most images, these two feature extraction methods yielded similar results. However, in some images, the spatial location feature of pixels played a significant role in segmentation, allowing for the differentiation of areas. For example, in the case of an apple, it managed to distinguish the background as walls and floor.

Interpretation of superpixel-level feature extraction methods

Superpixel-level feature extraction methods are divided into three categories: 'Mean of RGB color values,' 'RGB color histogram,' and 'Mean of Gabor filter responses.' Each method has its distinct characteristics, resulting in significantly different visual outcomes. For instance, 'Mean of RGB color values' focuses on averaging the colors of pixels within each superpixel, while 'RGB color histogram' counts the number of different colors within each superpixel. On the other hand, 'Mean of Gabor filter responses' differs from both; it first processes the image through Gabor filters and then matches the output with superpixels.

In my experience, I obtained the best results from the 'Mean of Gabor filter responses' feature extraction method. The results obtained from this method were visually more appealing compared to the other two methods. For example, in the case of the 'Short Tree,' the 'Mean of Gabor filter responses' feature extraction method segmented the overall structure of the plant first, followed by the colors of the leaves, and finally captured subtle color transitions beautifully as the number of superpixels increased.