EE5175 - Lab 9 Report

Gadha Premadasan Malayil - NS24Z155

25 April 2024

1 Introduction

This report presents the implementation and observations of the K-means clustering algorithm in Python. The algorithm is applied to two input images: a car image and a flower image.

In part a , we do k means with the mean values that were given in the question. In part b, we randomly take means from a uniform distribution and find the result for highest cost and lowest cost.

2 Functions

2.1 K-means Clustering

The function `k_means_clustering` implements the K-means algorithm for image segmentation. It assigns pixels to the nearest cluster mean based on their Euclidean distances. The output image pixels are filled with the newly calculated mean values from the cluster created.

2.2 K-means b

The function `k_means_b` performs K-means clustering using random initialization of cluster means. It repeats the clustering process with different initializations and selects the outputs with the lowest and highest costs.

2.3 euclidean distance

The function `euclidean_distance` finds the euclidean distance of the pixel and the mean that is passed to it.

3 Implementation

The K-means clustering algorithm is implemented using the numpy library for numerical operations and the Pillow library for image processing. The algorithm is applied to two input images, and the resulting clustered images are saved.

4 Observations

1. K-means clustering iteratively groups data points based on similarities, gradually forming clusters with similar features.
2. The algorithm requires prior knowledge of the number of clusters. In this implementation, we assumed 3 clusters and performed 5 iterations.
3. Various similarity measures, such as Euclidean distance and Mahalanobis distance, can be employed. Here, we utilized the Euclidean Distance metric.
4. The process of recomputing centroids and reassigning data is akin to the Greedy Descent algorithm, leading to convergence to a local minimum.
5. The final clustering results heavily depend on the initialization of centroids. Poor initialization may trap the algorithm in inferior local minima.
6. To mitigate the impact of initialization, we conducted 30 iterations with random centroid initialization, resulting in a range of cluster formations.

7. The randomization in centroid selection introduces initialization sensitivity, influencing the clustering outcome based on initial centroid placement.

8. Images corresponding to the maximum cost function exhibit sharper clustering compared to those of the minimum cost function.

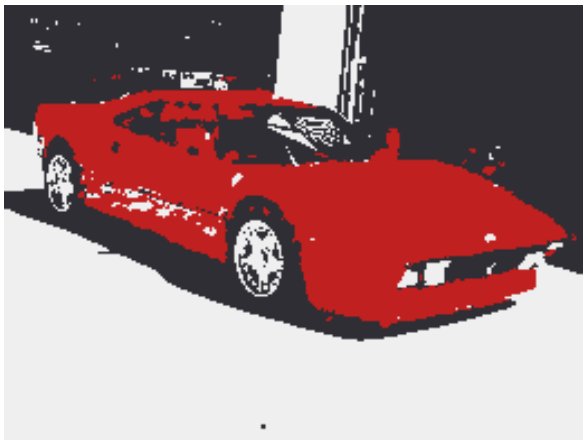
5 Results

The implementation is tested on the car and flower images using both random initialization and a given initial means. The resulting images demonstrate the segmentation of the input images into distinct clusters.

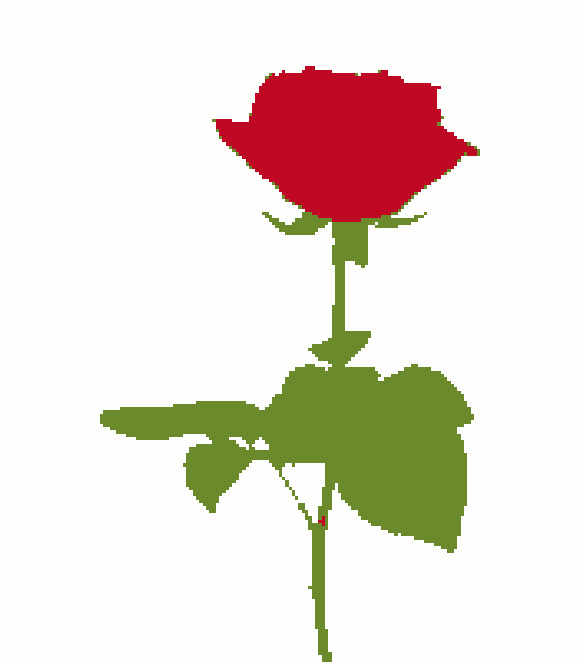
(a) Clustered Car Image

(b) Clustered Flower Image

Figure 1: Result of K-means Clustering

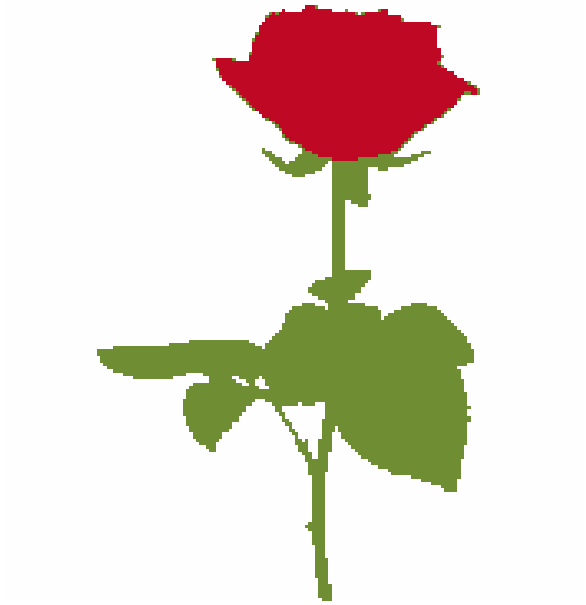


(a) Max Cost - Car Image



(b) Max Cost - Flower Image

(c) Min Cost - Car Image



(d) Min Cost - Flower Image

Figure 2: Output Images for Min and Max Costs

6 Conclusions

The implementation of the K-means clustering algorithm successfully segments the input images into distinct clusters. The cost function is a good way to find good means from random mean values.