

COMPUTER VISION

Programming Assignment - II

- Marri Bharadwaj
B21CS045

1.

SEGMENTATION

Image segmentation is a fundamental task in Computer Vision aimed at partitioning an image into multiple regions or segments to simplify its representation and facilitate analysis. The goal is to group pixels with similar characteristics, such as colour, intensity, texture, or motion, into coherent regions while maintaining discontinuities between different regions. This process enables various higher-level tasks like object detection, recognition, and tracking, making it a crucial step in many computer vision applications.

In this assignment, I have implemented *Ratio-cut* clustering algorithm based segmentation as well as *K-Means* clustering based segmentation. I have built the algorithms from scratch.

I began by importing the images folder from Google Drive, and imported the necessary libraries such as *OpenCV*, *OS*, *KMeans*, and *Matplotlib*. I displayed the images as below:



K-Means Clustering based Segmentation

K-means clustering-based image segmentation is a popular technique in computer vision used to partition an image into distinct regions or segments. It works by grouping pixels into clusters based on their feature similarity, with each cluster representing a segment in the image. K-means clustering aims to minimise the intra-cluster variance, meaning that pixels within the same cluster are as similar as possible, while pixels in different clusters are dissimilar. By adjusting the number of clusters, k , the algorithm can produce segmented images with different levels of granularity.

At first, I built the *KMeans* algorithm and implemented it on the above two images, without reducing the image for better clarity. I iterated through the list of these images and performed *K-means* clustering on each image using two different numbers of clusters: 3 and 6. Firstly, I converted each image to a numpy array and then flattened the array into a one-dimensional array. For each value of k (number of clusters), I initialised a *KMeans* model with the specified number of clusters and fit the model to the flattened image data. I then retrieved the labels assigned to each pixel and the cluster centres. These labels were reshaped back to the original image shape, resulting in a segmented image where each pixel is assigned a label corresponding to its cluster. Finally, the segmented image was displayed using *matplotlib.pyplot*. The result of the algorithm on the unaltered images is displayed below:

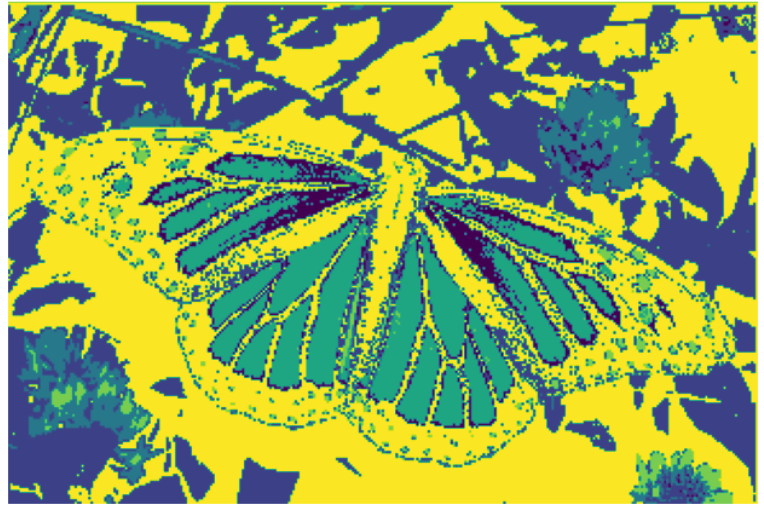
3 clusters

K-means clustering segmented image with 3 Clusters



6 clusters

K-means clustering segmented image with 6 Clusters



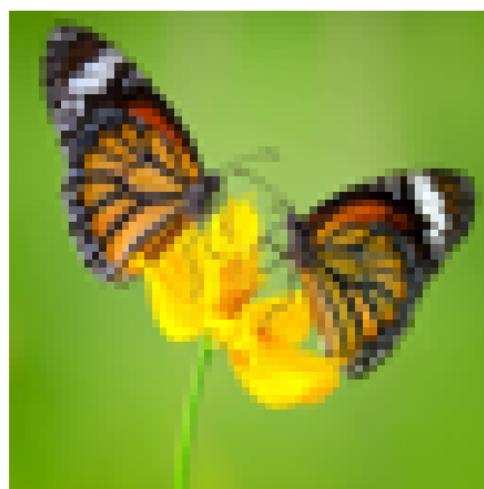
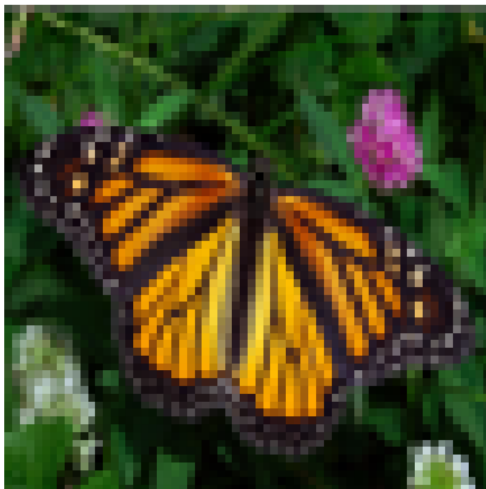
K-means clustering segmented image with 3 Clusters



K-means clustering segmented image with 6 Clusters



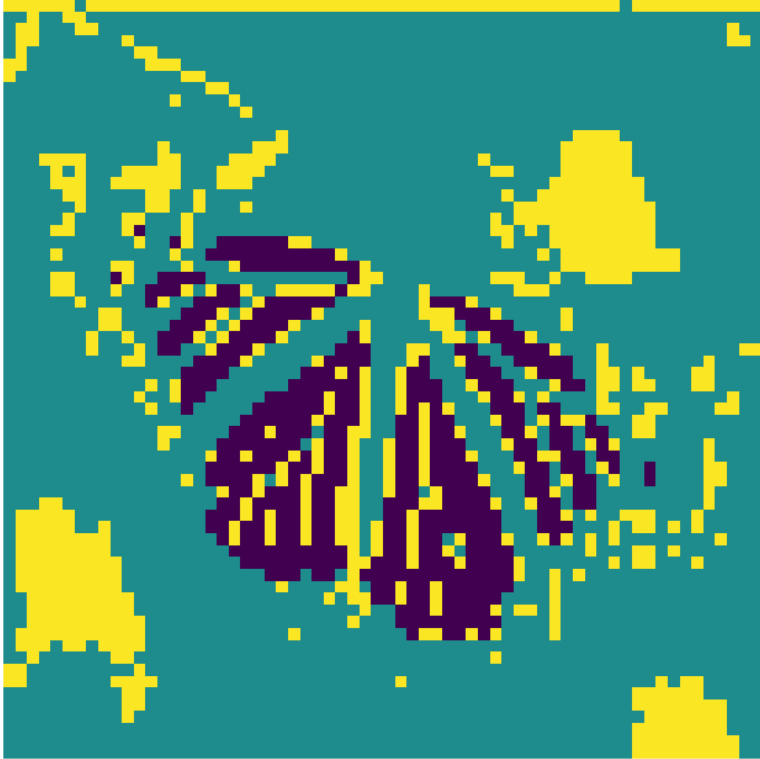
Now, as per the suggestion, I reduced the size of the images to 64x64 to reduce computation cost as below:



Now, I implemented the KMeans algorithm on these processed images and displayed them below:

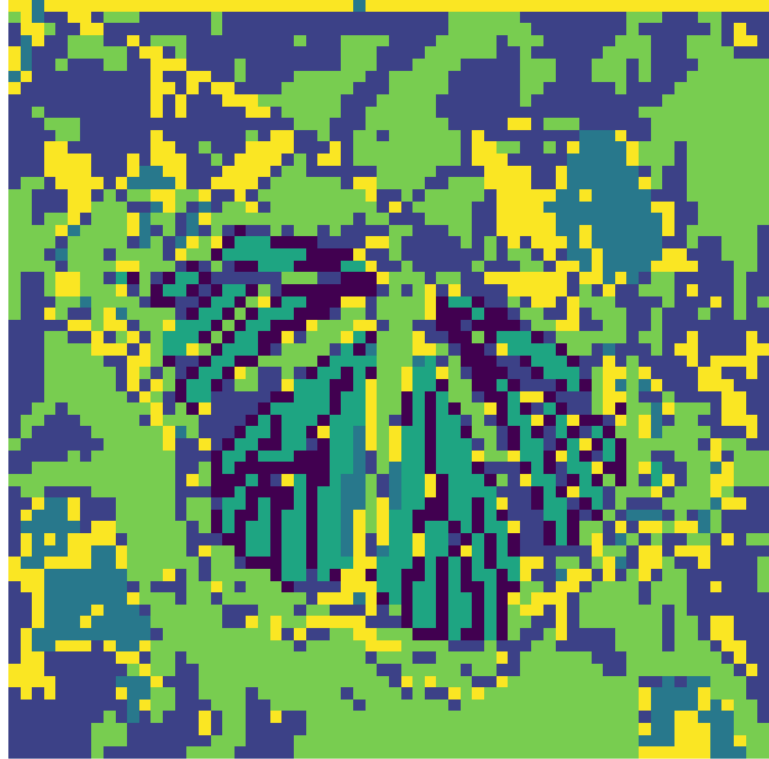
3 clusters

K-means clustering segmented image with 3 Clusters

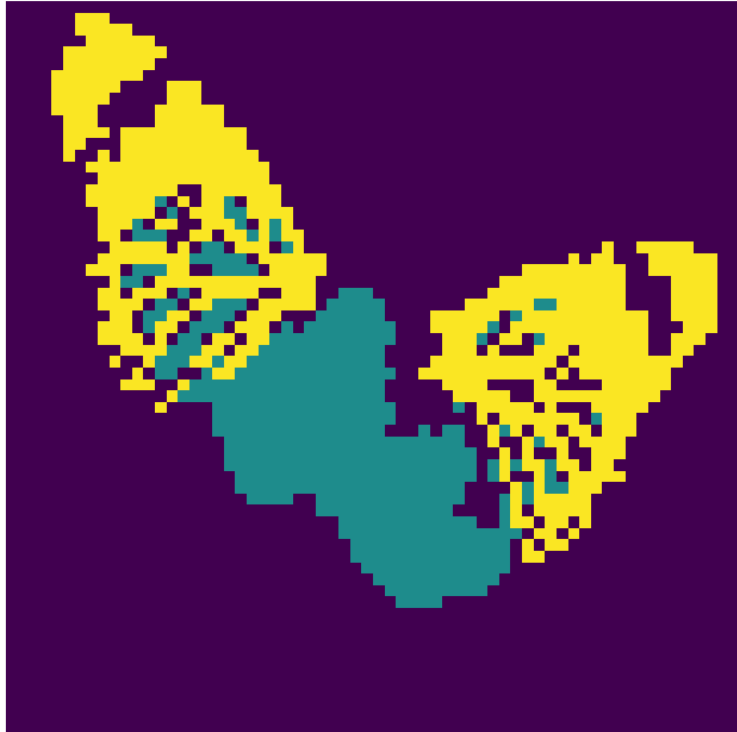


6 clusters

K-means clustering segmented image with 6 Clusters



K-means clustering segmented image with 3 Clusters



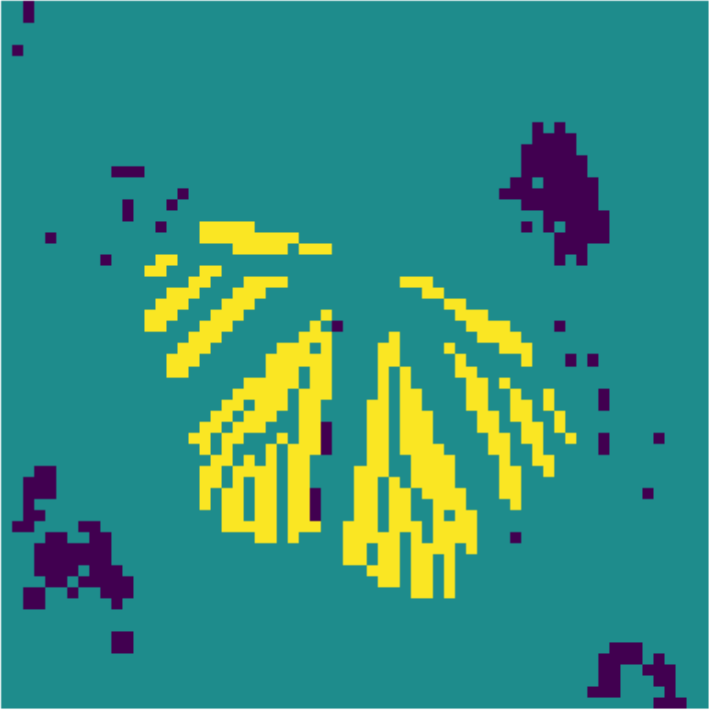
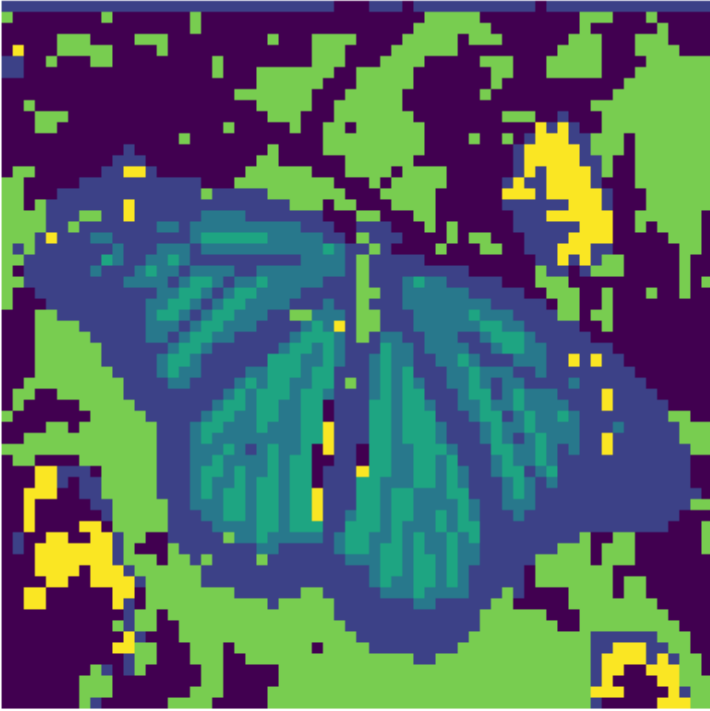

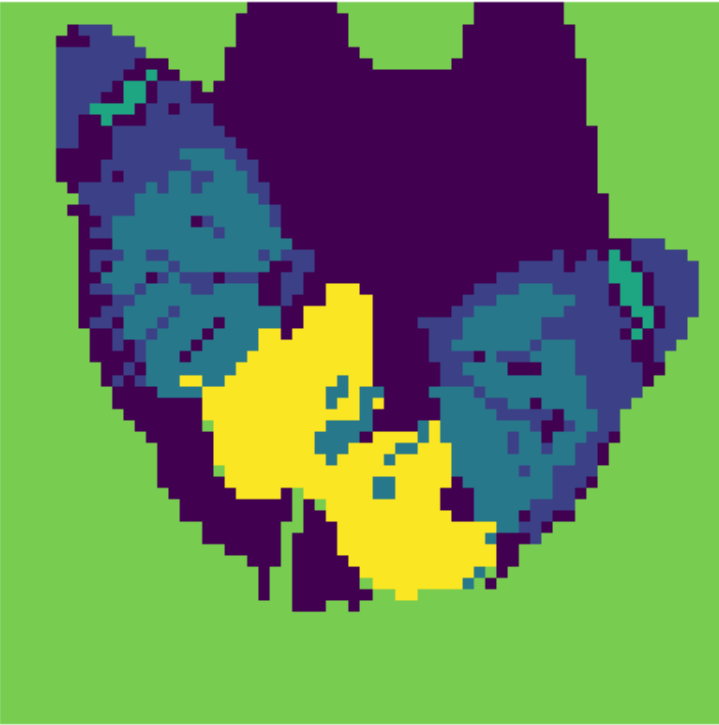
K-means clustering segmented image with 6 Clusters



Ratio-Cut Clustering based Segmentation

Ratio-cut based clustering is a popular technique in image segmentation, a fundamental task in computer vision. Unlike traditional clustering methods like K-means, which aim to minimise intra-cluster variance, ratio-cut clustering focuses on dividing the image into disjoint regions based on minimising the ratio of the cut cost to the total dissimilarity within each region. This approach aims to achieve segmentation by finding the optimal partition of the image that maximises inter-cluster dissimilarity while minimising intra-cluster dissimilarity. Ratio-cut clustering offers advantages in preserving boundary details and handling complex image structures

I built the algorithm from scratch using only the basic functions from the OpenCV library. I began by loading and preprocessing an image, resizing it to a specified shape. I then flattened the image into a feature vector while retaining its RGB channels. Next, I calculated the affinity matrix A , which represents pairwise similarities between pixels using a Gaussian kernel with a specified sigma value. Using this affinity matrix, I constructed the Laplacian matrix L , which captures the structure of pixel relationships in the image. By computing the eigenvalues and eigenvectors of L , I got a set of transformed features that encode meaningful information about the image's structure. I selected and standardised a subset of these transformed features, and then clustered using *K-means* into the specific required number of clusters. Finally, I displayed the segmented image as below:

3 clusters	6 clusters
<p data-bbox="228 201 659 237">Ratio-Cut segmented Image</p>  <p>This image shows a butterfly segmented into three clusters using the Ratio-Cut algorithm. The background is a solid teal color. The butterfly's body and wings are segmented into three distinct regions: a yellow region for the head and upper wings, a dark purple region for the lower wings and tail, and a small dark purple region on the left side of the body.</p>	<p data-bbox="976 201 1406 237">Ratio-Cut segmented Image</p>  <p>This image shows a butterfly segmented into six clusters using the Ratio-Cut algorithm. The background is a solid green color. The butterfly's body and wings are segmented into six distinct regions: a yellow region for the head and upper wings, a dark purple region for the lower wings and tail, a blue region for the central part of the wings, a teal region for the outer edges of the wings, a dark purple region for the left side of the body, and a small dark purple region on the right side of the body.</p>
<p data-bbox="224 1029 664 1064">Ratio-Cut segmented Image</p>  <p>This image shows a butterfly segmented into three clusters using the Ratio-Cut algorithm. The background is a solid teal color. The butterfly's body and wings are segmented into three distinct regions: a yellow region for the head and upper wings, a dark purple region for the lower wings and tail, and a small dark purple region on the left side of the body.</p>	<p data-bbox="979 1029 1419 1064">Ratio-Cut segmented Image</p>  <p>This image shows a butterfly segmented into six clusters using the Ratio-Cut algorithm. The background is a solid green color. The butterfly's body and wings are segmented into six distinct regions: a yellow region for the head and upper wings, a dark purple region for the lower wings and tail, a blue region for the central part of the wings, a teal region for the outer edges of the wings, a dark purple region for the left side of the body, and a small dark purple region on the right side of the body.</p>

It can be thus observed that *Ratio-Cut* based clustering performed as well, if not better than, as the *KMeans* clustering algorithm for image segmentation.