Image compressor based on K-means and K-medoids

Kaiwen Gong

1 Implementing Detail of my k-medoids

1.1 Initial Medoids Selection

At the beginning of the K-medoids algorithm, k-medoids are randomly selected from the given image. This selection is crucial as it forms the initial clusters around which the algorithm iteratively refines the clustering.

1.2 Distance Calculation and Medoids Updating

For each iteration, the algorithm computes the Euclidean distance between the RGB values of each pixel and the medoids. Euclidean distance is a common choice for this kind of task due to its effectiveness in capturing the geometric differences between points (or pixels, in this case) in a multi-dimensional space like RGB color space.

After computing these distances, for each cluster, the algorithm identifies the pixel that is closest to the cluster's centroid (mean of the cluster points) and sets this pixel as the new medoid. This step is crucial for adjusting the clusters to more accurately reflect the data's structure.

1.3 Stopping Criteria

Two conditions are set for stopping the iteration to balance computational efficiency and clustering effectiveness:

1.4 Convergence Check

The algorithm checks if the newly generated image is the same as the previous one. If there is no change, it implies that the clusters have stabilized, and the algorithm can safely terminate.

1.5 Iteration Limit

A maximum of 10 iterations is set as a limit to prevent excessive computation. This limit ensures that the algorithm does not consume too much computing resource while still achieving a reasonably good clustering result.

1.6 The Result of my program

1.6.1 Result k = 32

k-means: 6025.05ms

k-medoid: 6330.39ms

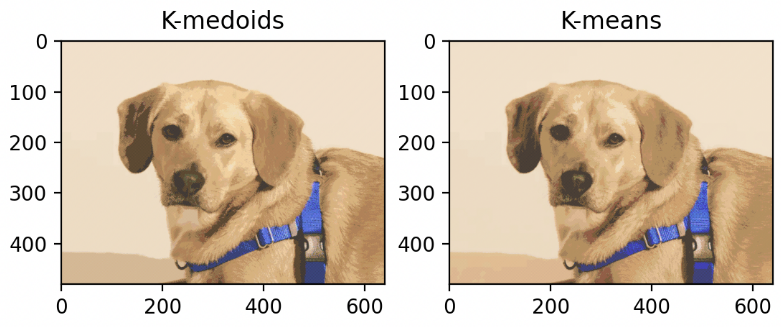

Figure 1: Result k = 32

1.6.2 Result k = 16

k-means: 3038.93ms

k-medoid: 3234.15ms

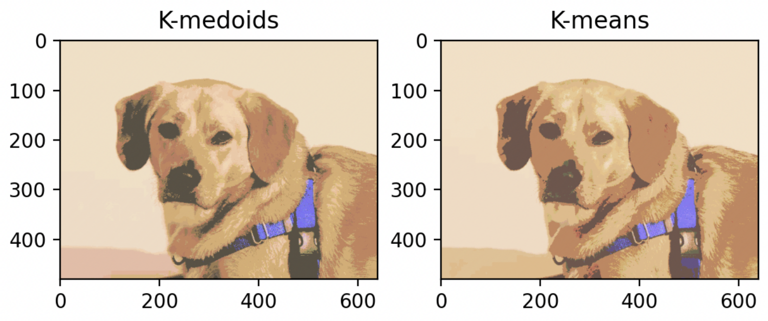

Figure 2: Result k = 16

1.6.3 Result k = 3

k-means: 577.45ms

k-medoid: 705.37ms

The k-means spend less time in those three cases.

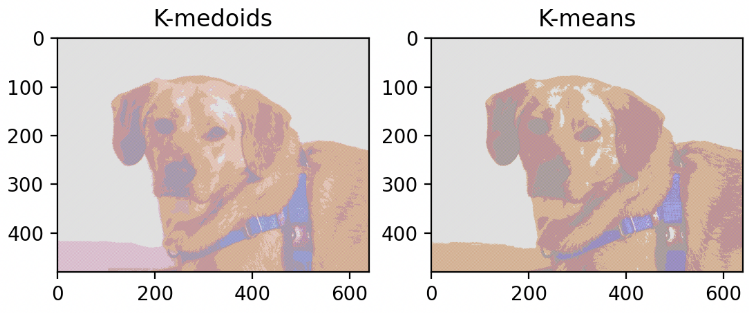

Figure 3: Result k = 3

2 Run the figure with different initial points

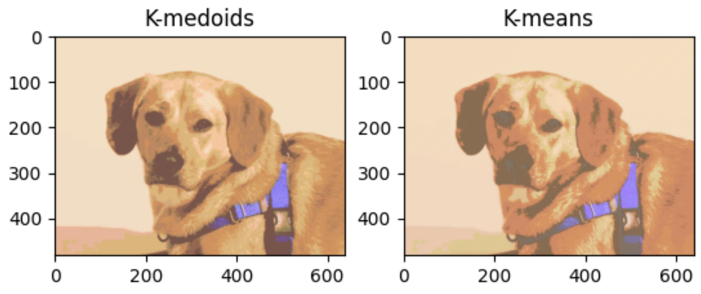

Figure 4: Initial centroids case1

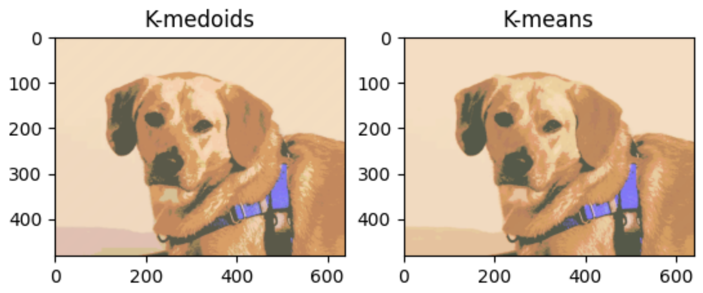

Figure 5: Initial centroids case2

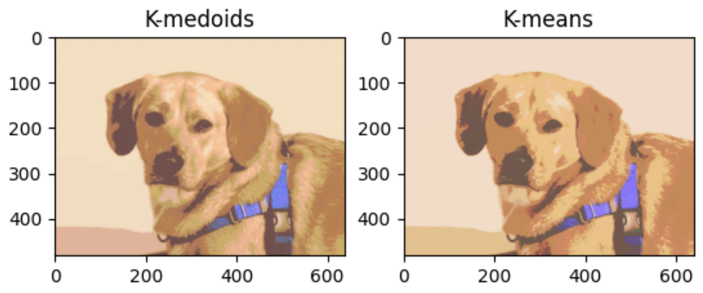

Figure 6: Initial centroids case1

These cases illustrate how varying initial points can significantly impact the outcomes of the test, yielding distinct results across different scenarios.

3 Conclusion

In k-means, the average of the clusters is used as the new center while k-medoids use the closest point to the average. We can observe that k-means takes less time. k-medoids reduce the effect of the outliner and are more robust. However, in my cases compressing a dog image without much noise and complexity, k-means performance is better in restoring the images. Also, the result of those functions will be influenced by the initial points.