Segmentation algorithms

Ananth Bharathwaj T A (21BCE1079)
Sankari Karthik (21BCE1396)

Segmentation: It is the assignment of labels to pixels. The input is an image and the output is a matrix with various elements specifying the object instance to which each pixel belongs. The segmentation is done based on color, intensity, contrast, boundary, texture, and similar factors.

The two segmentation algorithms chosen are:

(i) K-means
(ii) Mean shift

Dataset: The original dataset was sourced from google - it consists of 30 images. Since one of the algorithms chosen takes a while to run, the images were made smaller. The images were resized to have a height or width of 150 pixels while maintaining aspect ratio.
The benchmark dataset was made using Meta's Segment Anything (https://segment-anything.com/demo) which enables zero-shot generalization to unfamiliar objects and images. It can "cut out" any object, in any image, with a single click. This benchmark was used to compare our algorithms with each other. Link to our dataset: https://github.com/Sankari-K/segmentation-algos/tree/main/datasets

Advantages and Disadvantages:

K-means:

Advantages:

1) Speed: The execution time of the algorithm is fast.
   Proof: The time complexity of this algorithm is O(k*n*d) where k is the number of clusters, n is the number of d-dimensional points. For the entire dataset (of 30 resized images), the algorithm takes 10.422 seconds with k = 5. When k = 12, the execution time is 34.583 seconds.

```
PS C:\Users\Shankari\Desktop\VIT- Chennai\6th sem\BCSE306L - ARTIFICIAL INTELLIGENCE\segmentation-algos\kmeans> python .\k_means.py
Time for image 1: 0.3920736312866211
Time for image 2: 0.24068331718444824
Time for image 3: 0.19720196723937988
Time for image 4: 0.28899693489074707
Time for image 5: 0.22666215896606445
Time for image 6: 0.13773083686828613
Time for image 7: 0.24541020393371582
Time for image 8: 0.19876384735107422
Time for image 9: 0.2171485424041748
Time for image 10: 0.08855748176574707
Time for image 11: 0.0990447998046875
Time for image 12: 0.19474196434020996
Time for image 13: 0.23551511764526367
Time for image 14: 0.377779006958078
Time for image 15: 2.355336904525757
Time for image 16: 0.6395823955535889
Time for image 17: 0.13144493103027344
Time for image 18: 0.36463379859924316
Time for image 19: 0.248857021331871
Time for image 20: 0.20169448852539062
Time for image 21: 0.4456779956817627
Time for image 22: 0.36580944061279297
Time for image 23: 0.11424899101257324
Time for image 24: 0.293506383895874
Time for image 25: 0.07396340370178223
Time for image 26: 0.20676922798156738
Time for image 27: 0.3041269779205322
Time for image 28: 0.18462228775024414
Time for image 29: 0.4804835319519043
Time for image 30: 0.8571879863739014
Total time (k = 5): 10.422316312789917
PS C:\Users\Shankari\Desktop\VIT- Chennai\6th sem\BCSE306L - ARTIFICIAL INTELLIGENCE\segmentation-algos\kmeans>
```

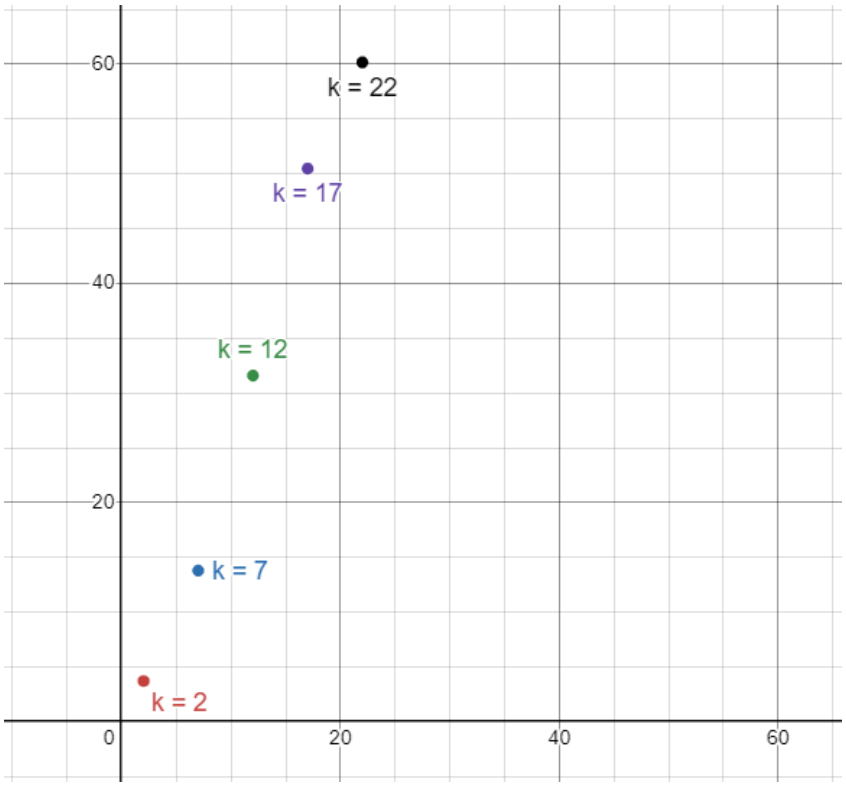
```
PS C:\Users\Shankari\Desktop\VIT- Chennai\6th sem\BCSE306L - ARTIFICIAL INTELLIGENCE\segmentation-algos\kmeans> python .\k_means.py
Time for image 1: 0.5938668251037598
Time for image 2: 0.4886946678161621
Time for image 3: 0.7125442028045654
Time for image 4: 1.069474458694458
Time for image 5: 0.5231878757476807
Time for image 6: 1.0484797954559326
Time for image 7: 0.6825032234191895
Time for image 8: 1.6863176822662354
Time for image 9: 0.2806735038757324
Time for image 10: 0.27469968795776367
Time for image 11: 0.6552472114562988
Time for image 12: 0.6336700916290283
Time for image 13: 0.3529379367828369
Time for image 14: 1.7386281490325928
Time for image 15: 1.1877832412719727
Time for image 16: 1.2351515293121338
Time for image 17: 0.7663452625274658
Time for image 18: 1.4456355571746826
Time for image 19: 1.094437599182129
Time for image 20: 1.4787108898162842
Time for image 21: 0.6684021949768066
Time for image 22: 0.6480593681335449
Time for image 23: 1.3950340747833252
Time for image 24: 2.0911426544189453
Time for image 25: 0.9690144062042236
Time for image 26: 2.056257963180542
Time for image 27: 4.191622972488403
Time for image 28: 0.7479979991912842
Time for image 29: 2.969540596008301
Time for image 30: 0.8860945701599121
Total time (k = 12): 34.58323335647583
PS C:\Users\Shankari\Desktop\VIT- Chennai\6th sem\BCSE306L - ARTIFICIAL INTELLIGENCE\segmentation-algos\kmeans>
```

2) Scalable: K-means can handle large datasets efficiently since its time complexity linearly depends on the number of pixels per image and the dimensions of the image.

Proof:

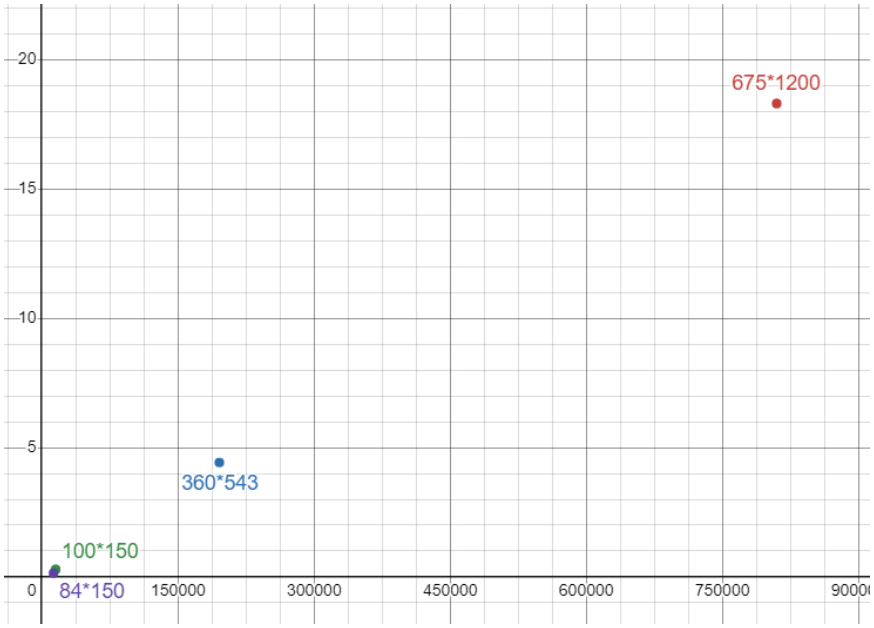
Varying k:

```
PS C:\Users\Shankari\Desktop\VIT- Chennai\6th sem\BCSE306L - ARTIFICIAL INTELLIGENCE\segmentation-algos\kmeans> python .\k_means.py
Total time (k = 2): 3.7109553813934326
Total time (k = 7): 13.79735517501831
Total time (k = 12): 31.58192276954651
Total time (k = 17): 50.47742676734924
Total time (k = 22): 60.17731738090515
PS C:\Users\Shankari\Desktop\VIT- Chennai\6th sem\BCSE306L - ARTIFICIAL INTELLIGENCE\segmentation-algos\kmeans>
```



Varying number of data points:

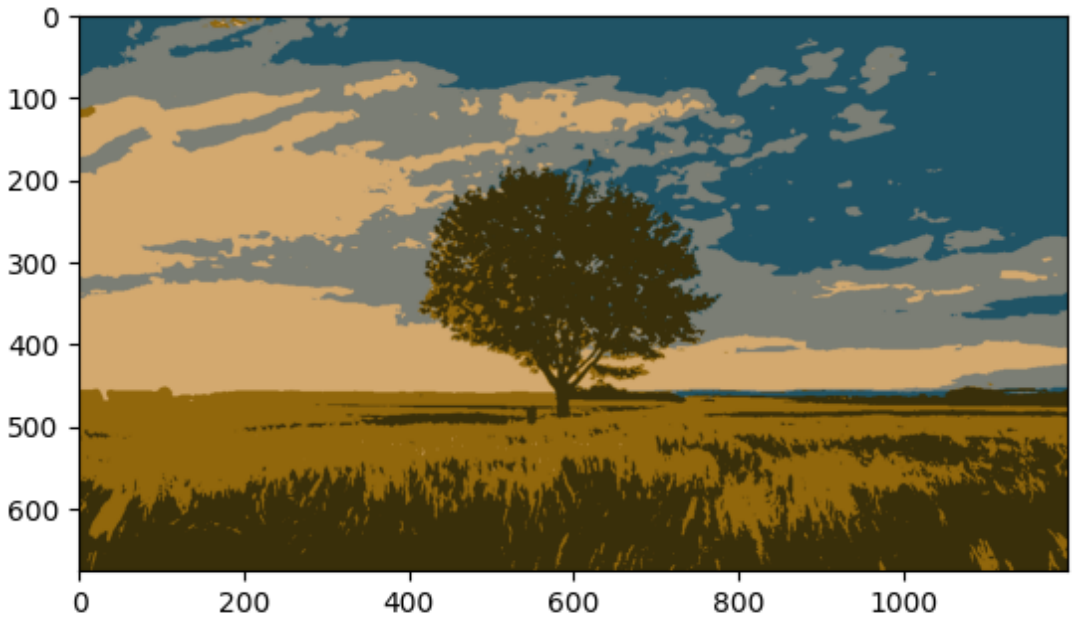
```
PS C:\Users\Shankari\Desktop\VIT- Chennai\6th sem\BCSE306L - ARTIFICIAL INTELLIGENCE\segmentation-algos\kmeans> python .\k_means.py

Dimensions:  675 1200
Total time:  18.307300329208374

Dimensions:  360 543
Total time:  4.423935890197754

Dimensions:  100 150
Total time:  0.2830193042755127

Dimensions:  84 150
Total time:  0.14314532279968262
PS C:\Users\Shankari\Desktop\VIT- Chennai\6th sem\BCSE306L - ARTIFICIAL INTELLIGENCE\segmentation-algos\kmeans>
```



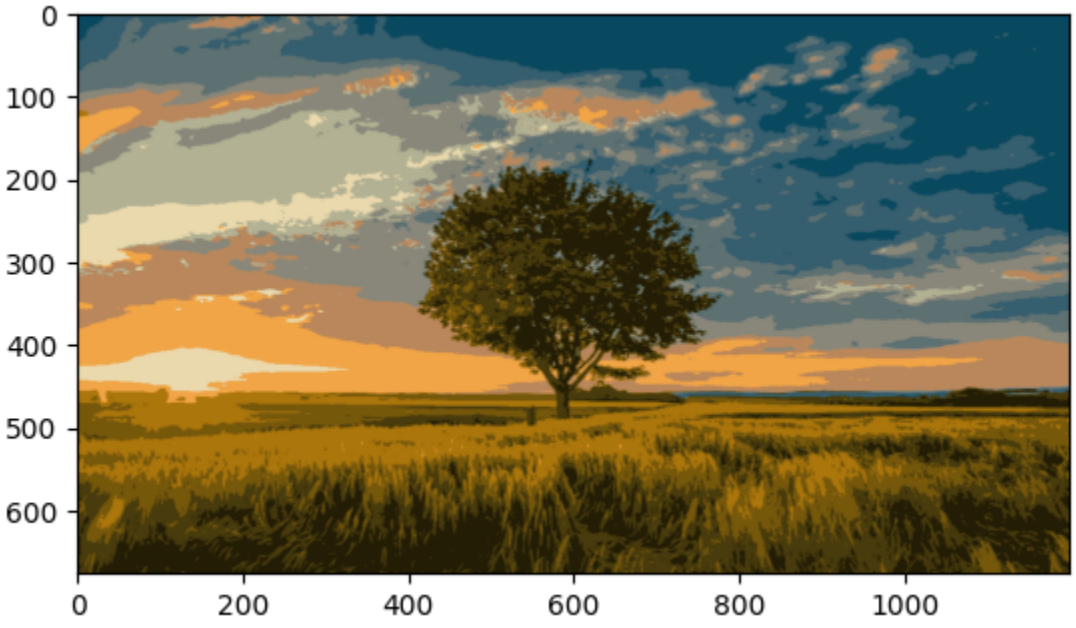
It can be seen that the relationship between these is linear - so, this algorithm is scalable.

3) Number of clusters can be chosen as required.
Proof: For the same image, the number of clusters as required can be customized. Here is an example for a demo picture.

Original



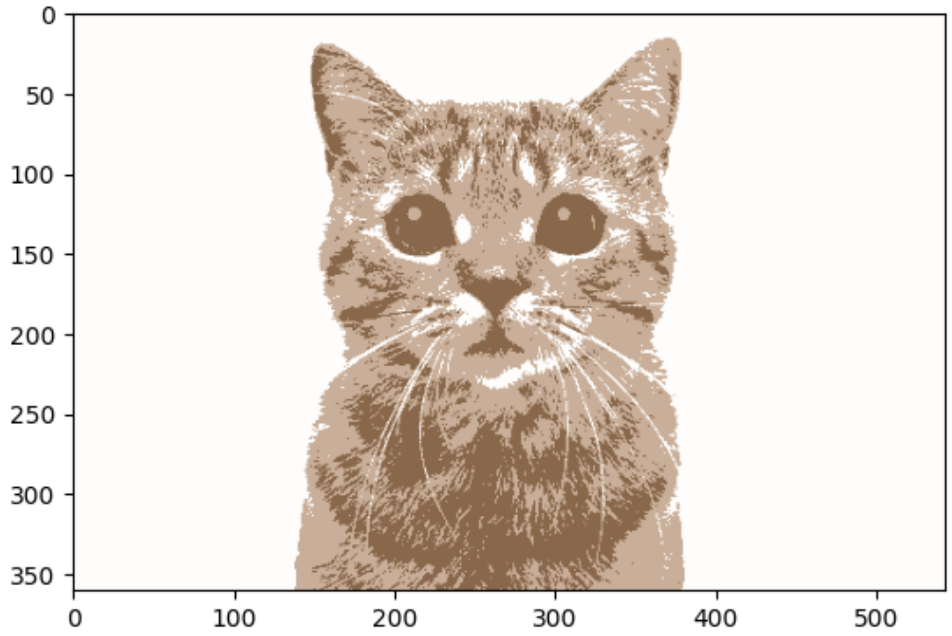
k = 5

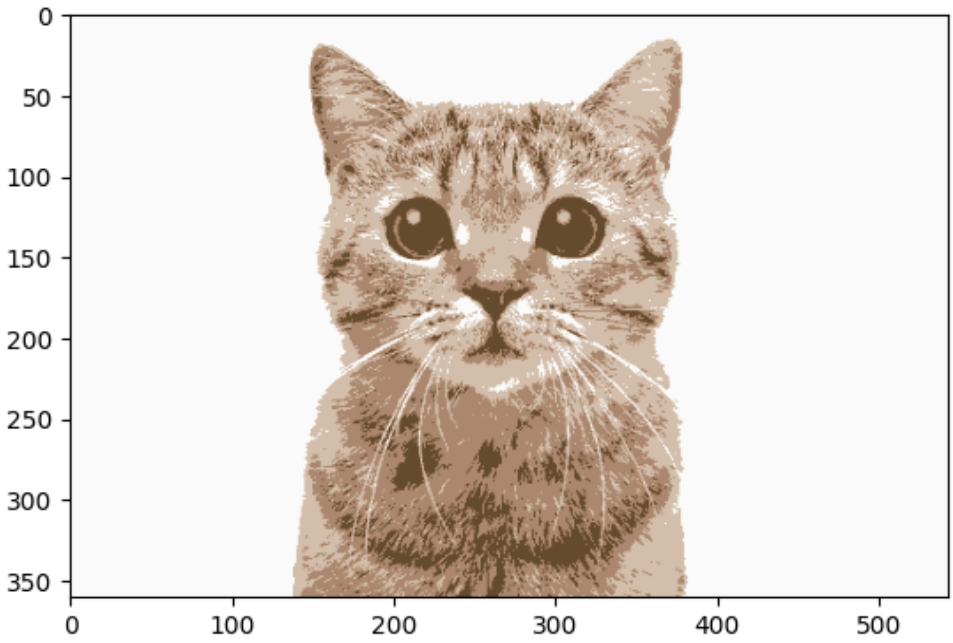


k = 12

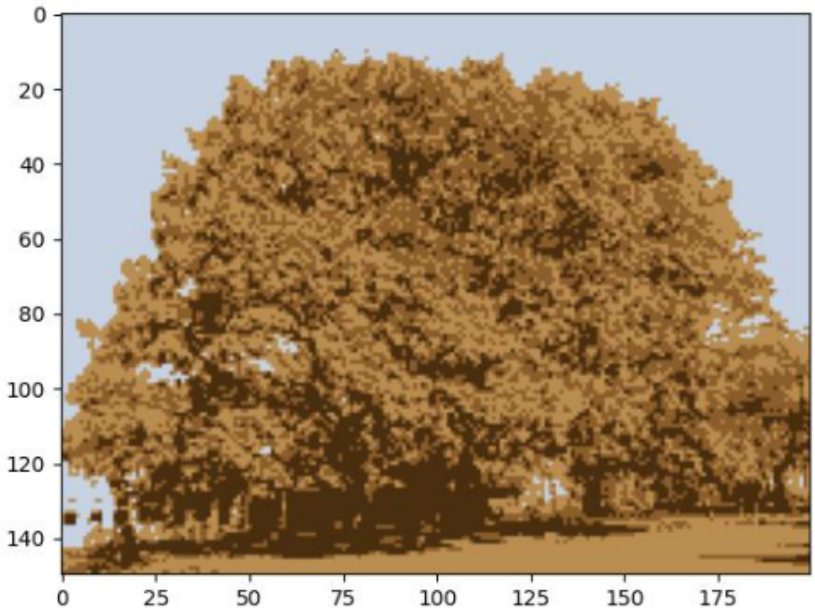
Disadvantages:

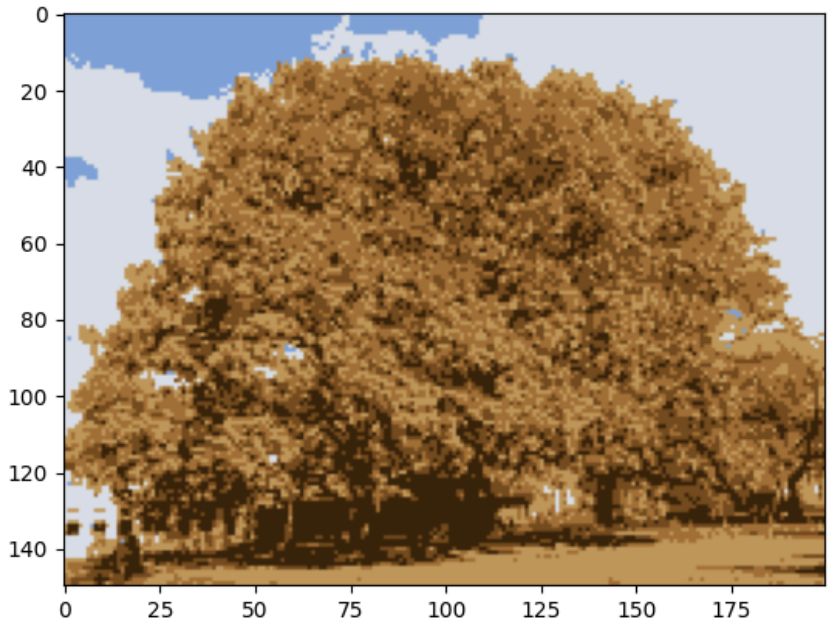
1) Sensitivity to initial centroid placement
   Proof: Since the initial centroids are randomly chosen, the segmented results differ slightly each time the algorithm is executed.





2) Tendency to converge to local optima
   Proof: The initial placement of centroids determines where the algorithm starts its iterative optimization process, and different initializations can lead to different local optima.
   For example, one of these images shows more details in the tree and another shows more detail in the sky, which were determined by the centroids.





Mean shift:

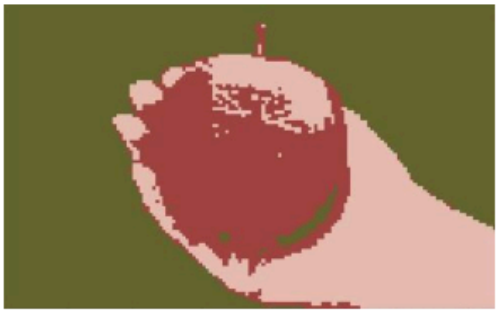
Advantages:

1. Flexibility: Unlike K-Means, Mean Shift does not require specifying the number of clusters beforehand. It dynamically determines the optimal cluster count based on the data distribution. This means that Mean shift can handle geometrically

complex images better than K-means clustering can.

Original

Benchmark Segmentation

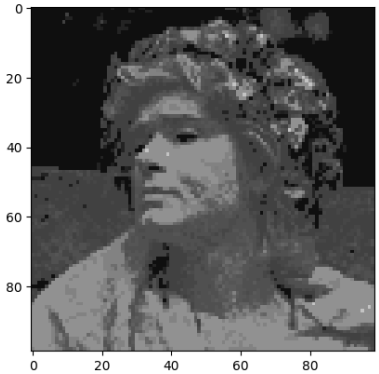


K-Means Clustering

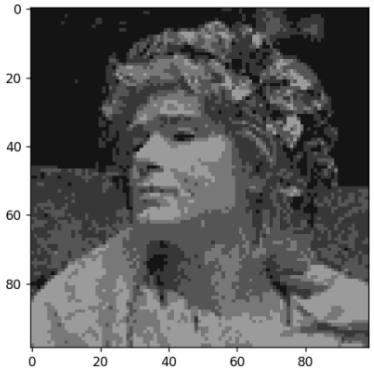
Mean Shift Clustering

2. Robust to Noisy Data: Mean-Shift is less sensitive to noisy data points. It tends to focus on dense regions, effectively ignoring outliers.

Original Image



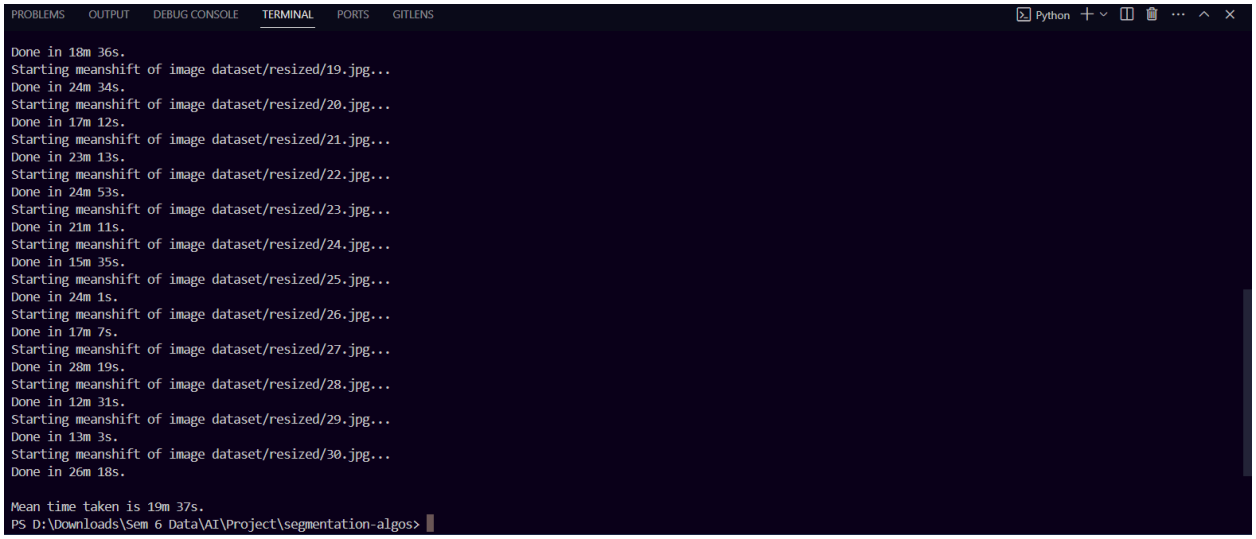
Mean Shift Result



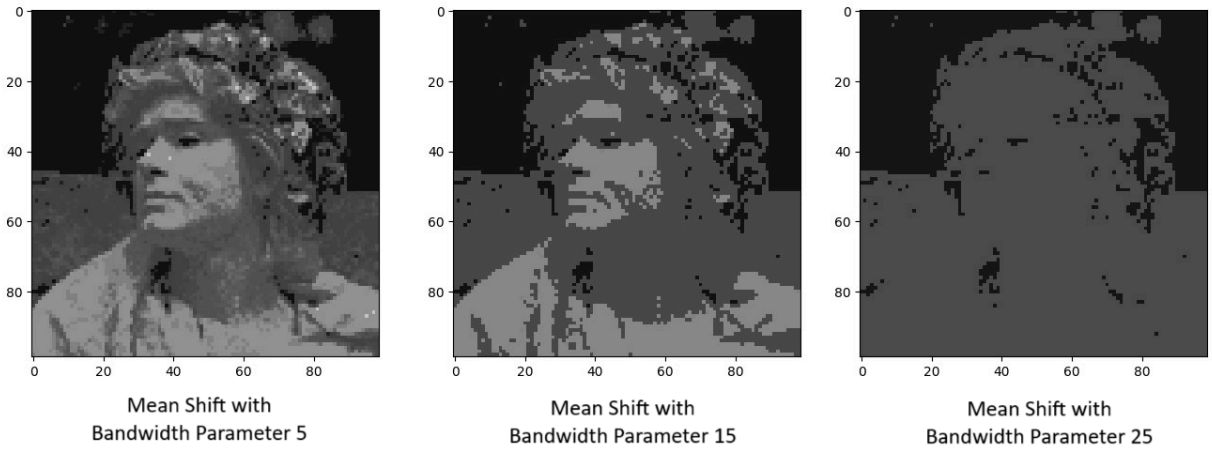
K-Means Result

Disadvantages:

1. Computationally expensive/Slow: Mean-Shift can be computationally expensive, especially for large datasets. Its time complexity is $O(n^2)$, impacting performance.



2. Sensitive to Bandwidth parameter: The choice of kernel (the shape of the neighborhood) and bandwidth (the radius of the kernel) significantly affects the resulting clusters. Poorly chosen parameters may lead to suboptimal results.



Results:

Image	Total Pixels	Benchamark Pixels	K-Means Object Pixels	Mean Shift Object Pixels
01.jpg	135850	94924	94912	94912
02.jpg	162032	101463	101434	101400
03.jpg	149188	106114	89562	100589
04.jpg	133590	102714	98702	90702
05.jpg	180072	137555	127539	107532
06.jpg	161704	128067	98040	118050
07.jpg	178729	80818	75807	65809
08.jpg	134096	80970	80458	79458
09.jpg	121278	61972	60958	58954
10.jpg	126464	107782	91264	102268
11.jpg	161865	66962	59950	45950
12.jpg	160884	93064	87562	76582
13.jpg	135356	80903	65391	34391
14.jpg	161538	60974	49462	26462
15.jpg	134808	93079	79567	52567
16.jpg	177996	149607	147571	143524
17.jpg	130152	78948	78636	78037
18.jpg	134689	156344	107751	10589
19.jpg	88695	67659	67594	67489
20.jpg	74501	30476	29638	27986
21.jpg	116090	35571	34861	33465
22.jpg	161048	32109	30209	31468
23.jpg	119804	58334	50713	55786
24.jpg	133956	32019	24790	10356
25.jpg	150060	95449	67017	85964
26.jpg	161376	132354	112938	125874
27.jpg	158253	41485	32737	38561
28.jpg	160884	43396	43384	43384
29.jpg	180310	100421	94742	98520
30.jpg	178466	65102	47115	11165

Performance metrics:

1) With respect to Time:
   K-means: 10s (mean, when k =5) Mean shift: 19m 37s (mean, bandwidth = 25)
2) With respect to the benchmark:
   Efficiency % = (Classified pixels / Benchmark pixels) * 100
   K-Means: 89.29% Mean Shift: 80.75%

Inferences:

In this study, we compared the K-Means and Mean Shift clustering algorithms in terms of efficiency and accuracy. K-Means demonstrated faster execution but lacked flexibility in handling irregular cluster shapes. Mean Shift, while slower, adapted well to complex geometries. According to our custom performance metric, K-Means seems to be doing better than Mean Shift. Practitioners should consider the trade-offs between speed and accuracy when choosing an algorithm for specific use cases.