K Means Segmentation Presentation

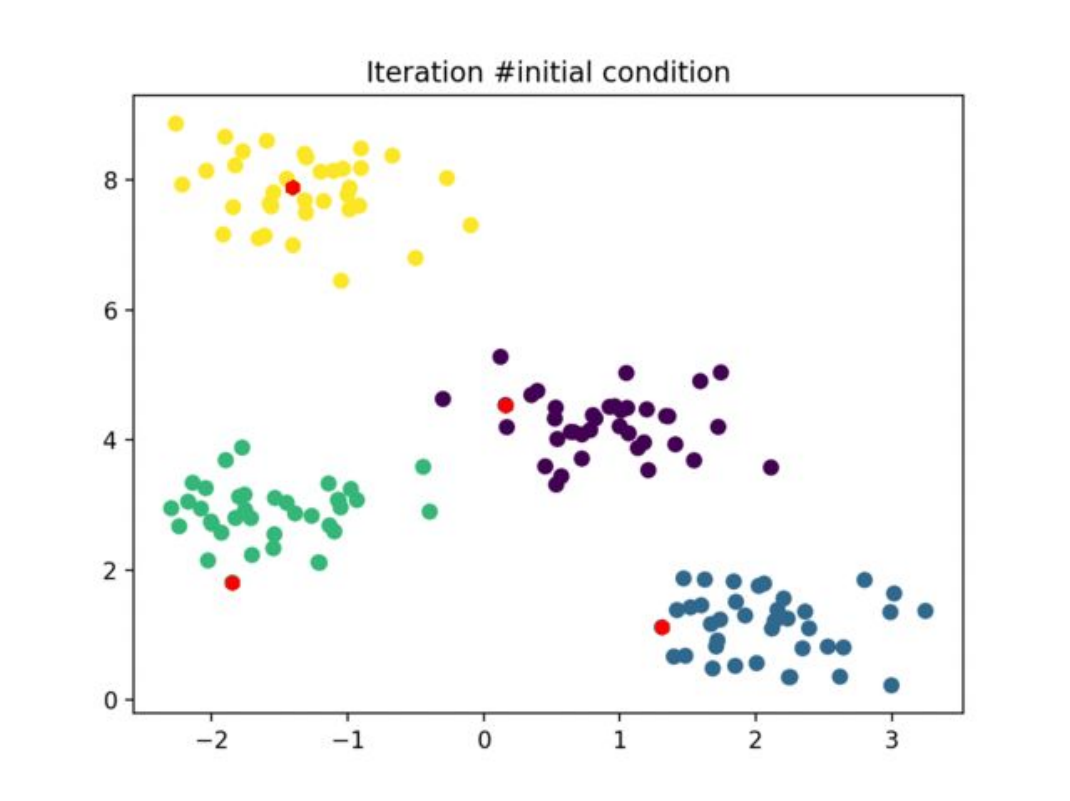
Monica Chan

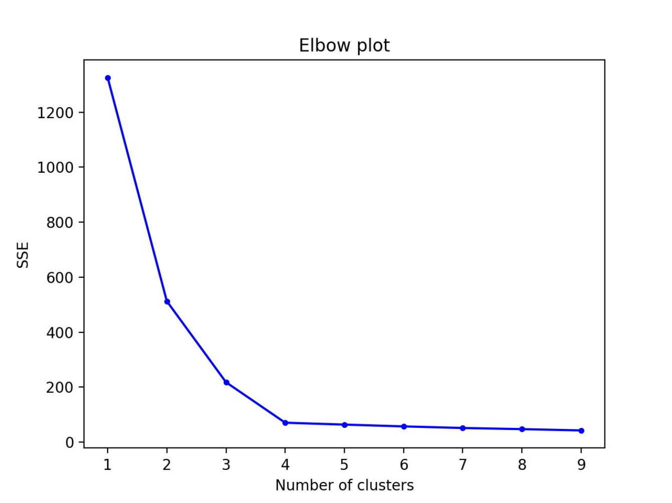
Basic Premise of How I structured things

I essentially unzipped each image into a single image before I fed it to the Kmeans algorithm:

- If my image was m pixels long and n pixels tall, I'd unzip the image into a numpy array m*n rows long.
  - If we are interested in RGB features for example, and my image was 481×321 pixels, the shape of my numpy array would turn from (481,321,3) to (154401, 3).
  - The show_segmentation function provided to us automatically reshapes it

Performing on Blobs



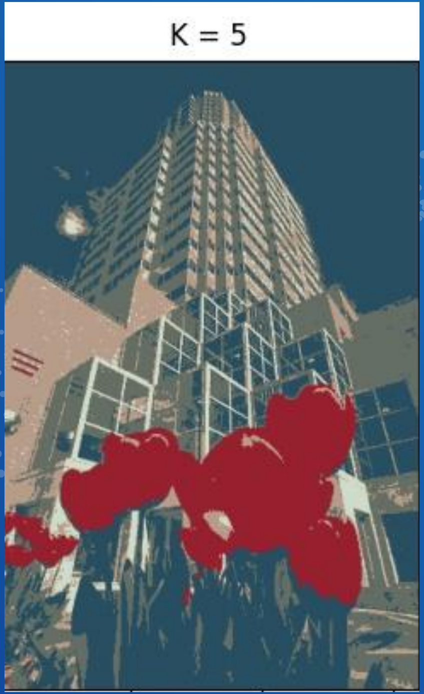


How I structured my experiments

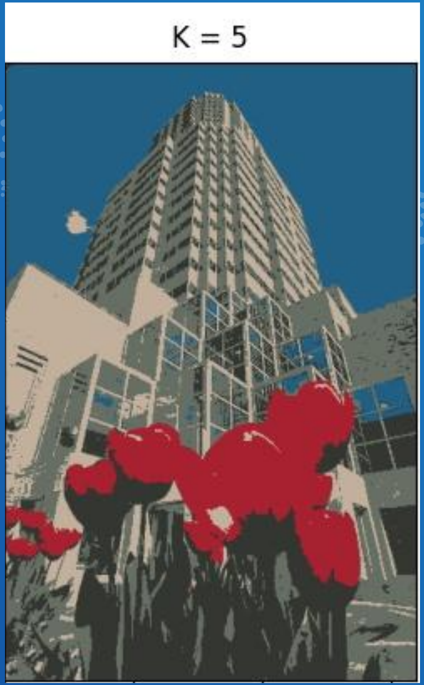
- Analyzed an elbow plot of HSV and RGB
  - Did not really find intensities helpful for what I could see these images needing segmentation for.
- If image had large color regions, weight in pixel location.
- Ran Kmeans++ on most tests

The Four Images I chose

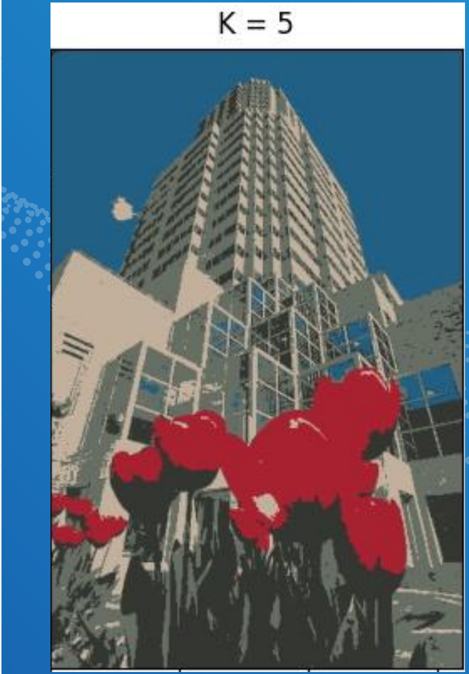
Random Cluster vs Best of 10 vs Kmeans++



Random initial

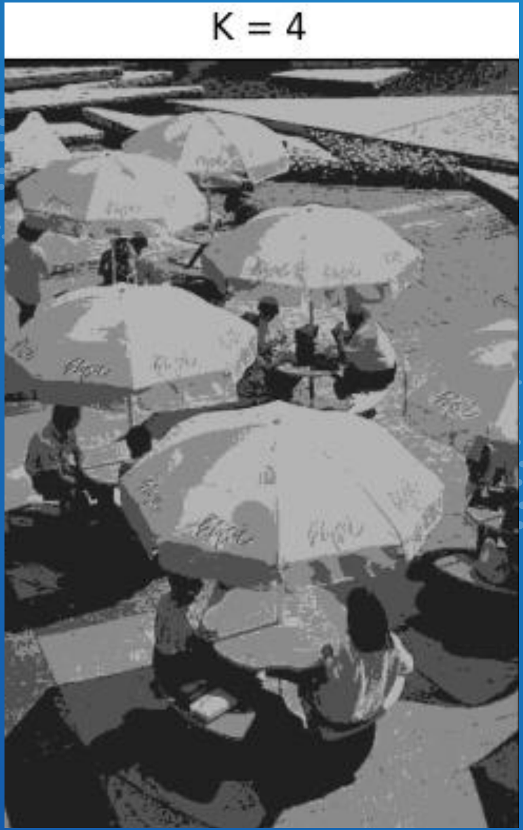


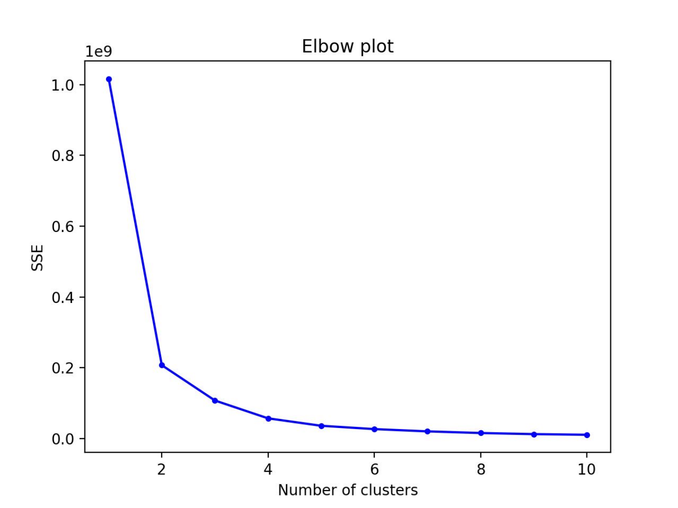
Best of 10



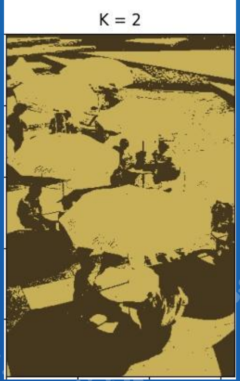
KMeans++

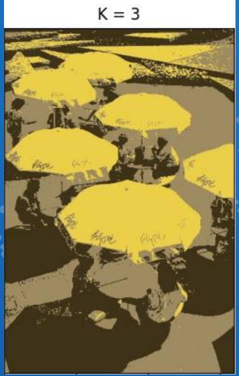
Umbrella image Black and White

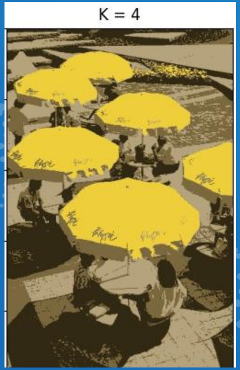


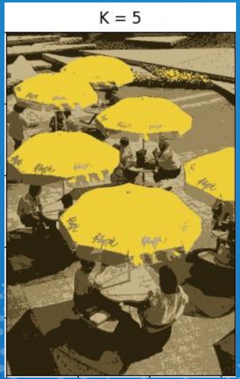


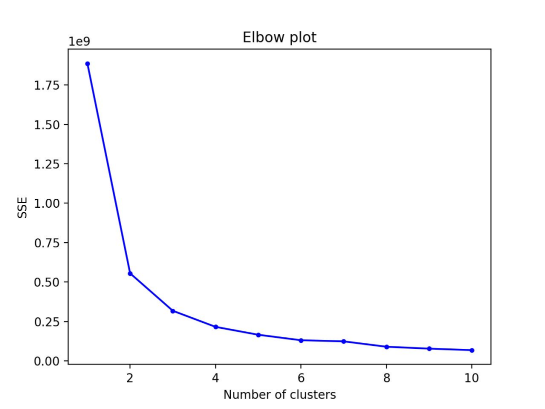
RGB Evolution

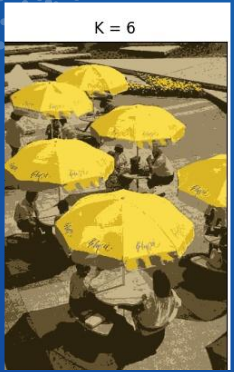


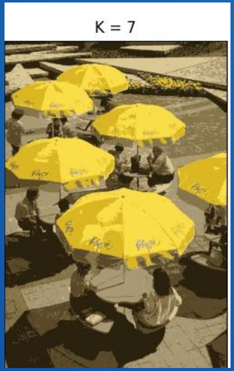


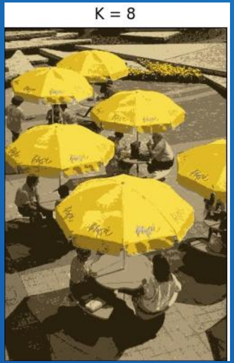


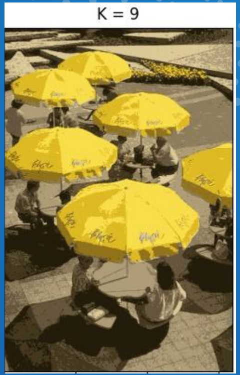


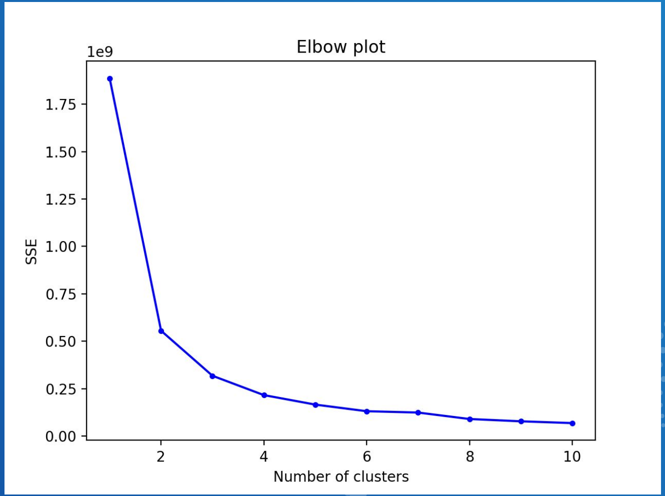




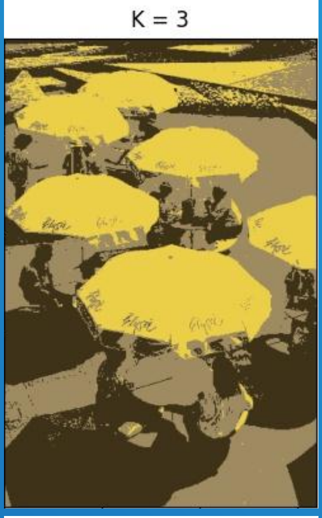


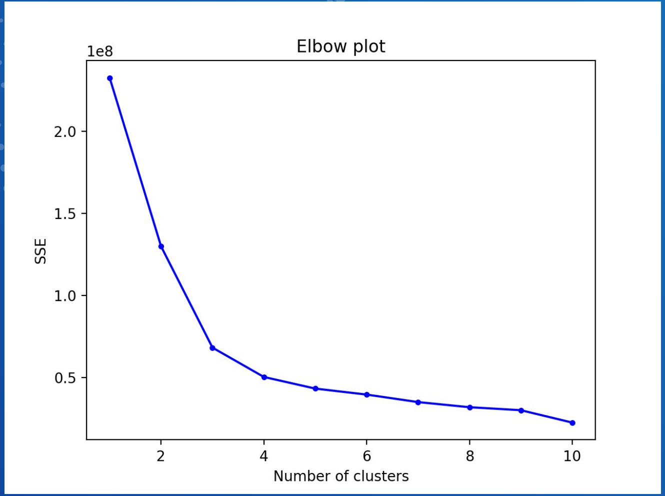




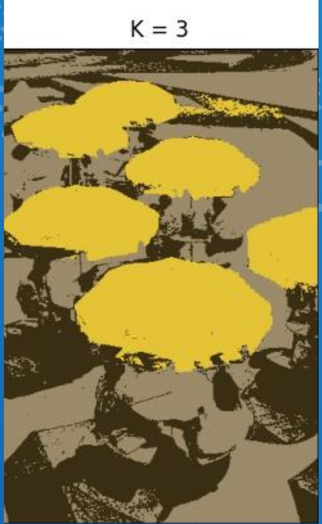

Elbow plot
1e9
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
SSE
2
4
6
8
10
Number of clusters

RGB

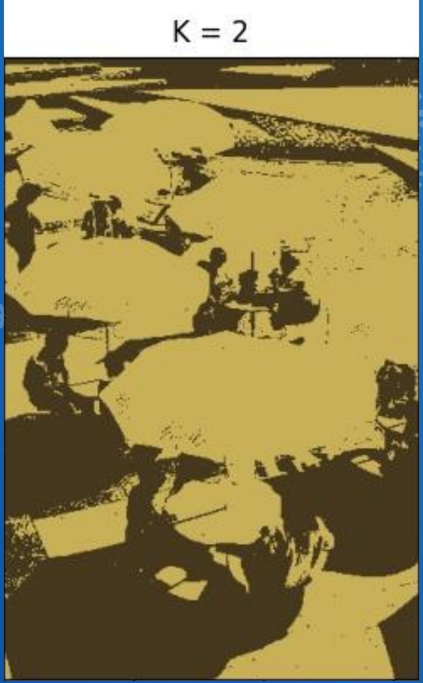

K = 3

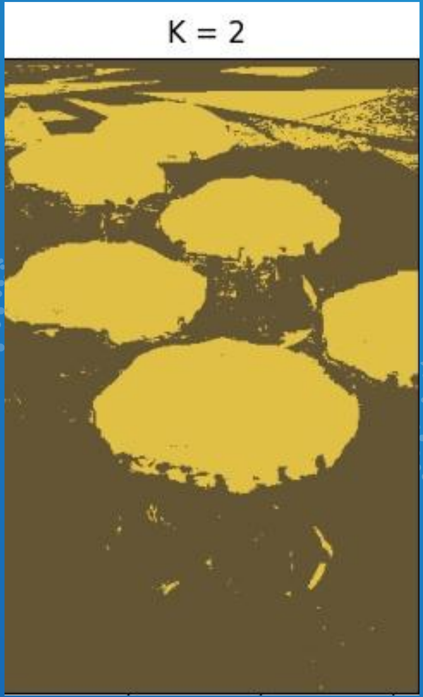

Elbow plot
1e8
2.0
1.5
1.0
0.5
SSE
2
4
6
8
10
Number of clusters

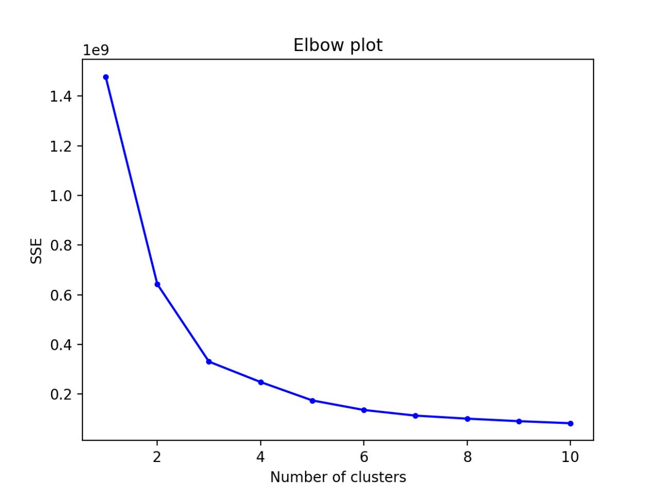
HSV

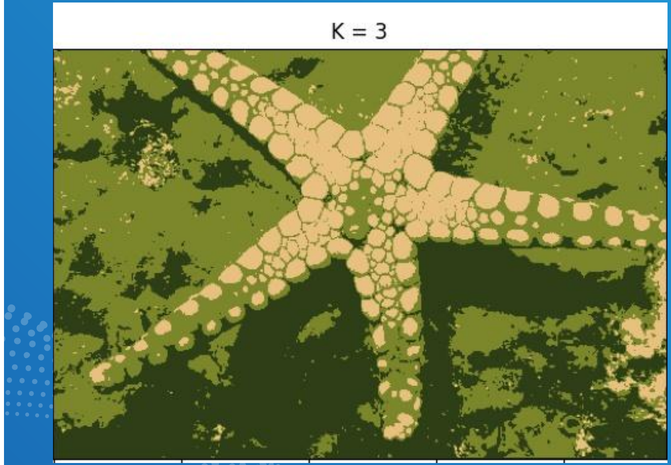

K = 3

Another interesting HSV vs RGB

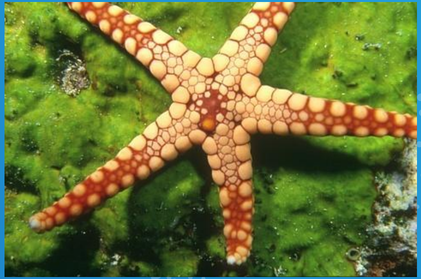




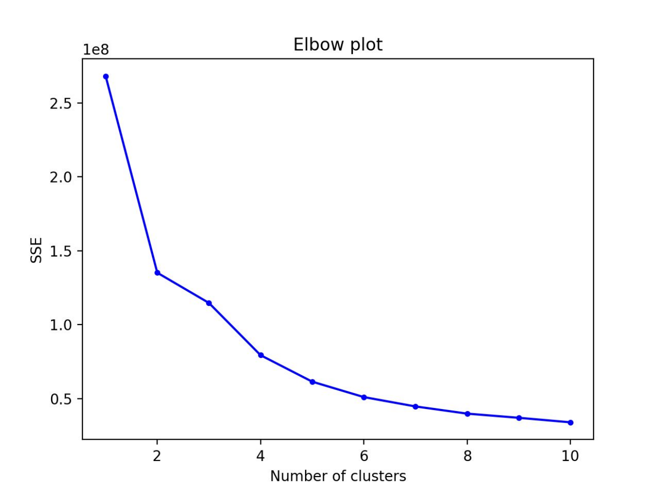

Elbow plot
1e9
SSE
1.4
1.2
1.0
0.8
0.6
0.4
0.2
2
4
6
8
10
Number of clusters

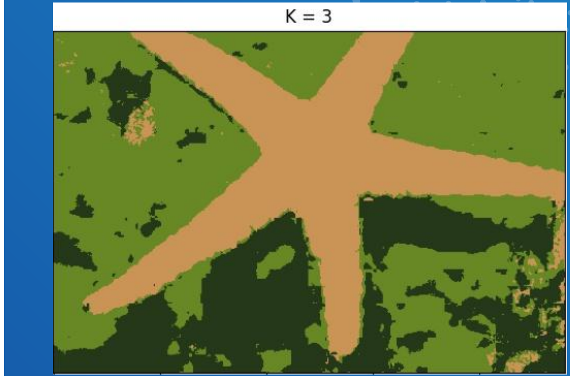

K = 3

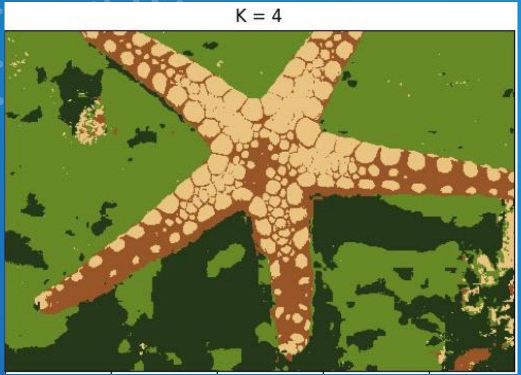
RGB

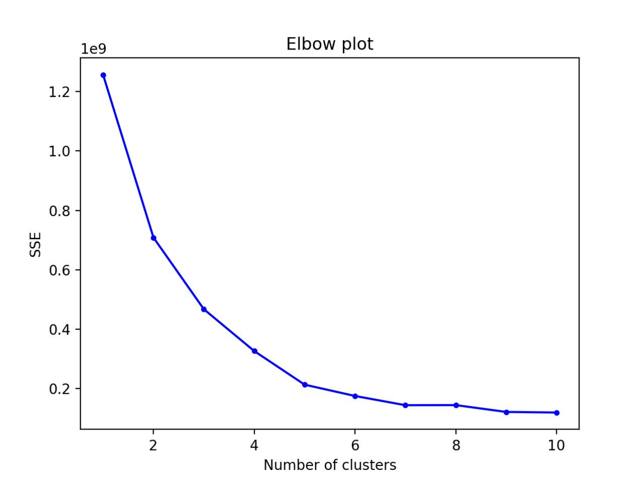


HSV

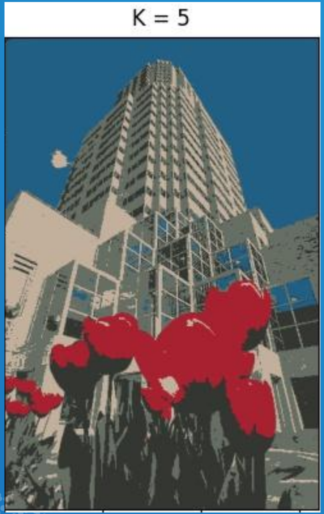

Elbow plot
1e8
SSE
2.5
2.0
1.5
1.0
0.5
2
4
6
8
10
Number of clusters

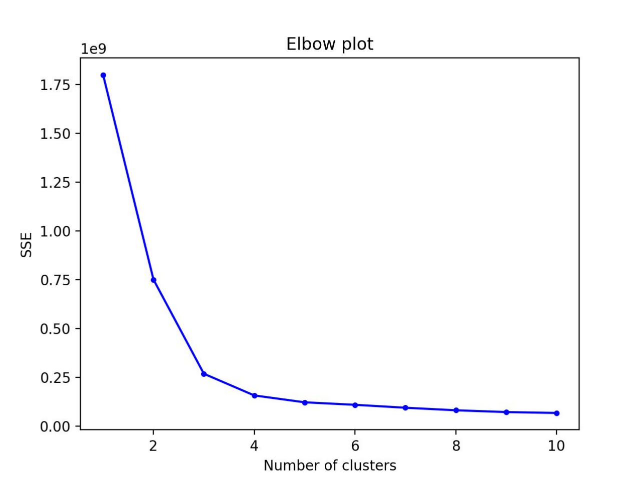

K = 3


K = 4

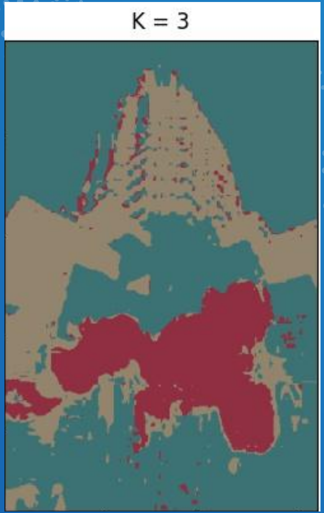

Elbow plot
1e9
1.2
1.0
0.8
0.6
0.4
0.2
SSE
2
4
6
8
10
Number of clusters

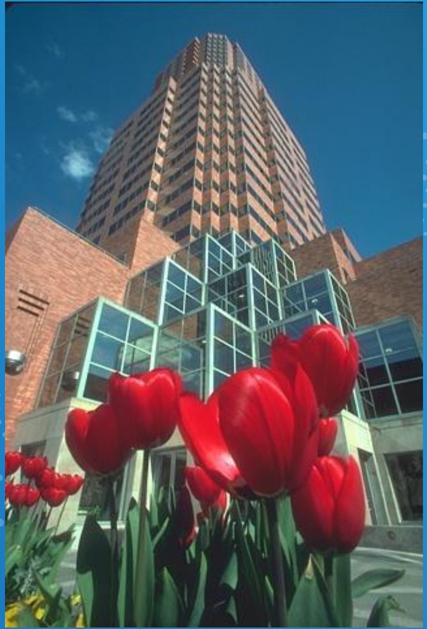
RGB


K = 5

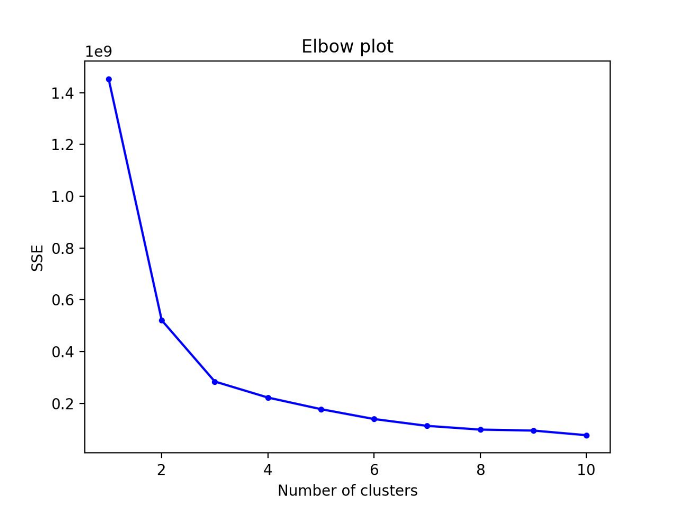

Elbow plot
1e9
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
SSE
2
4
6
8
10
Number of clusters

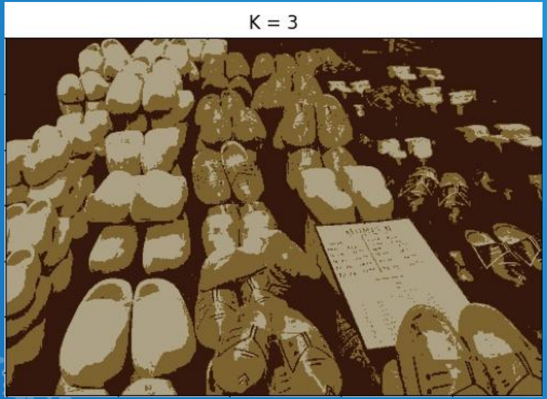
HSV


K = 3

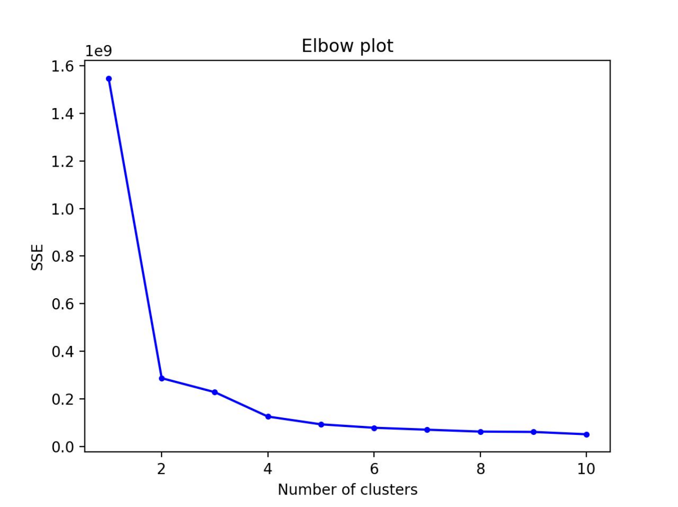


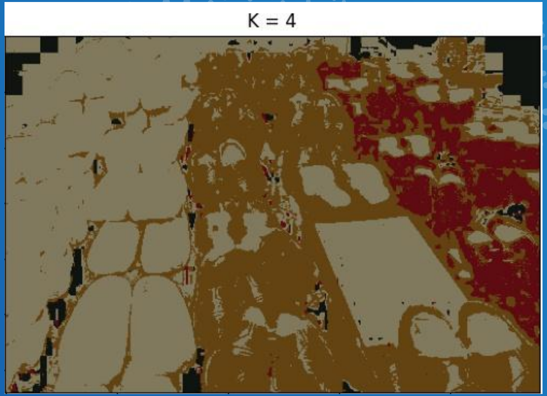
RGB


Elbow plot
1e9
1.4
1.2
1.0
0.8
0.6
0.4
0.2
SSE
2
4
6
8
10
Number of clusters

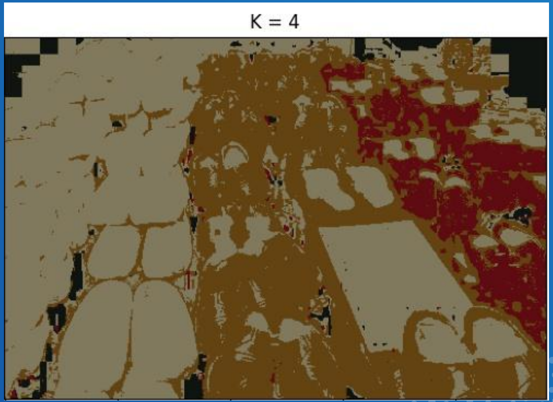

K = 3

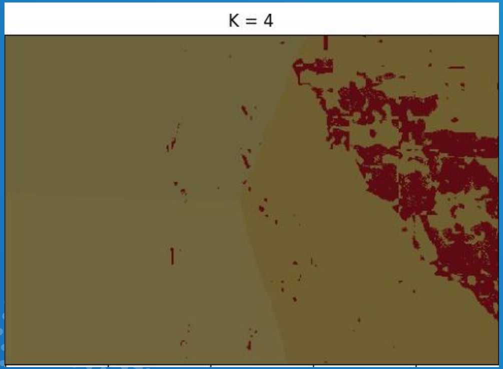
HSV


Elbow plot
1e9
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
SSE
2
4
6
8
10
Number of clusters

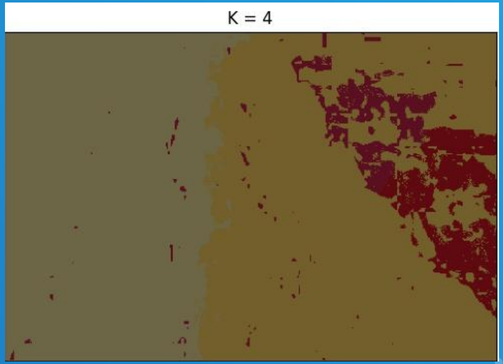

K = 4

KLOMPEN

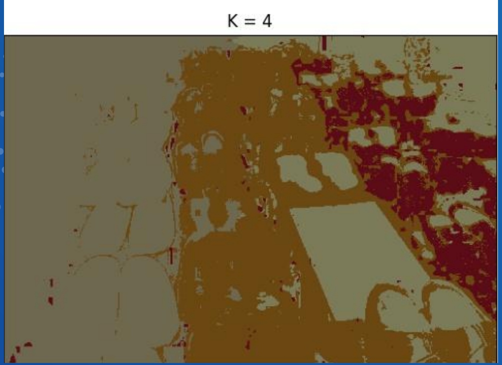




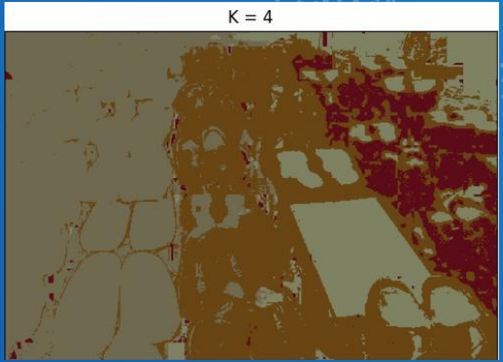
Weighted equal



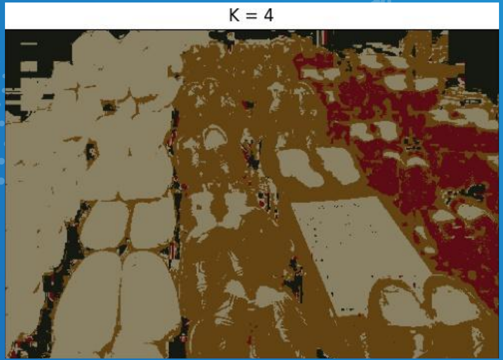
Divided Pixel Position by 2



Divided Pixel Positions by
3

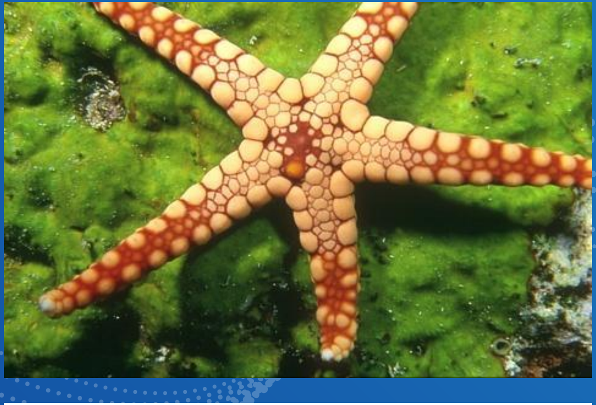


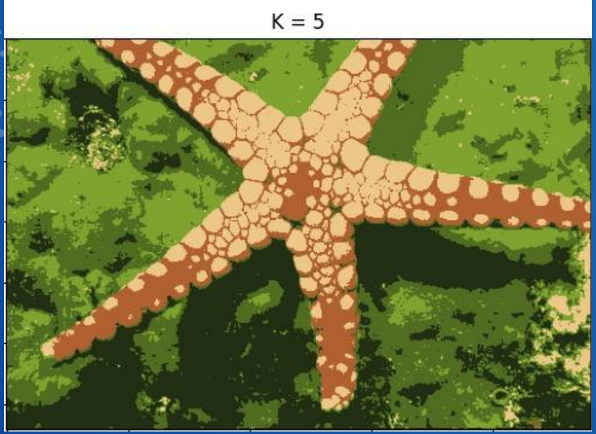
Divide by 5

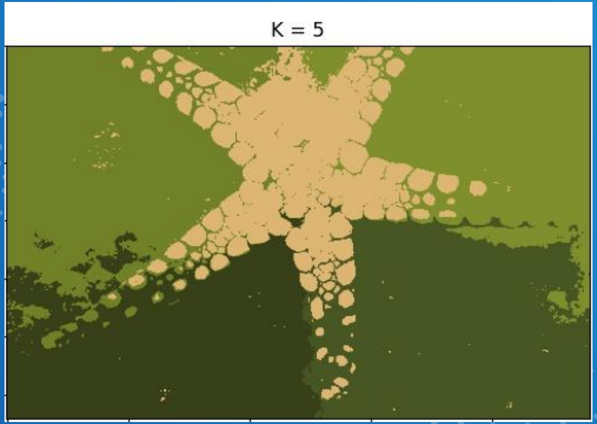


Divided by 10

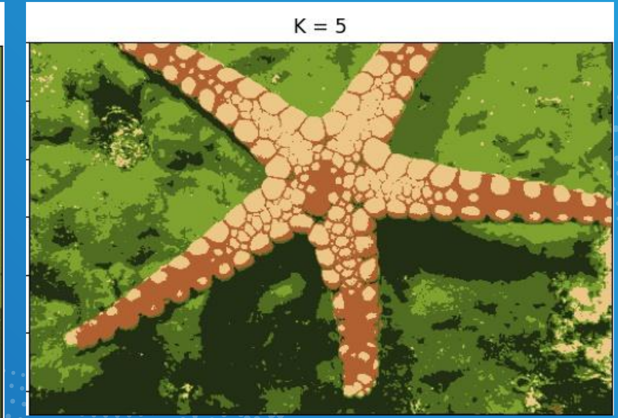
Using Pixel Location and Color channels







Pixel Location weighed equal to color channels



Dividing pixel positions by 10

Conclusions

- Honestly, I did not notice a huge performance difference between the best of 10 vs Kmeans++ method for choosing initial clusters
- For the images I used, most of them were around the same size and most of them took less than 100 iterations to get the correct image.
- HSV was able to show more contours of big regions in images at a lower k value (Peep slide 9)
- Pixel Regioning was only helpful on images with large color blocking
- Note: I feel like the best segmentation depends on the purpose!