COMP 3009 Assignment 1

Part II:

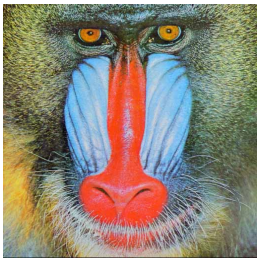
Question 8)

a. K = 50: The segmentation is relatively rough, which makes the regions larger and makes the images more difficult to identify. However this results in the fastest computational time.
b. K = 150: A little more refined, you can kind of start to tell what the image is. The regions are smaller and the colors are more separated. Still ok computationally but its slower than 50.
c. K = 500: This is significantly more visual, the output actually starts to become a lot more like the original image. Regions are smaller and the colors are segmented a lot better. But this makes it by far significantly slower than the previous 2 K values. This is where you will probably need to start thinking/worrying about to make it more efficient or about getting a more powerful machine for computations.

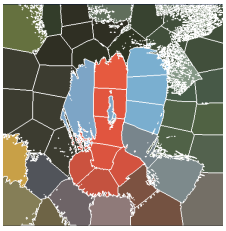
For further speed optimizations, you might want to think about techniques to reduce redundant calculations or parallelize the computation of pixel-to-site distances.

Question 9)

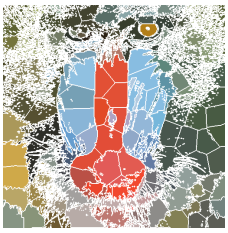
1. **Images with a lot of contrast:** the image works a lot better with straightforward images, where the colors are more blocky and the color regions are more easily identifiable. But once the colors start to become more complex, the algorithm starts to struggle a lot more. I also noticed that having a complex image with a very high k value actually results in a way worse output than lower k values.



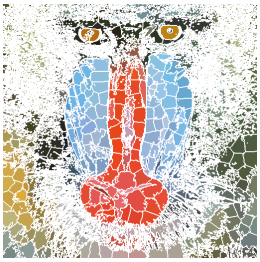
Baboon



Baboon with k=50

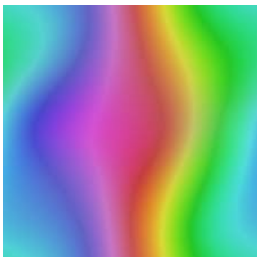


Baboon with K=150

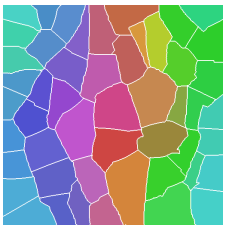


Baboon with k=500

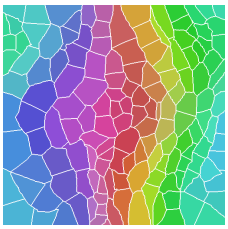
2. **Images with gradients or subtle changes in color**: The segmentation might struggle in these cases, creating regions that don't correspond to any meaningful features in the image. Subtle color changes might not be reflected accurately.



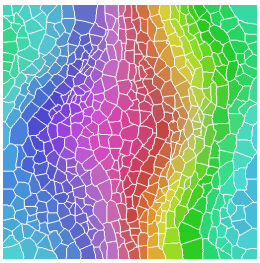
Gradient



Gradient with K=50



Gradient with K=150



Gradient with K=500

3. **Images with many small details**: As the number of sites (k) increases, it can capture more of these fine details, but at a computational cost.

Question 10)

In my implementation of SLIC, there were a few decisions I had to make, and those decisions may have affected the performance of this method. First, I initialized k sites with random values; thus, each run gave slightly different results. I considered using a more structured approach to place

sites, such as on a grid, but I couldn't quite get it working so I stuck with the random approach instead.

One question I curious about was what number of sites resulted in the most ideal output clarity and computational speed balance/ratio. For k = 50, these regions were a little on the bigger side and didn’t really give me a satisfactory output. In contrast, when k = 500, the output is far finer, but the increase in computational time is immense. However, since my approach was a brute force method, I believe that there could be potentially a different approach that would actually result in K=500 to be on the opposite end of the spectrum, wherein now it is not taking up nearly as much time.

Strong dependencies were also noted concerning the content of the images. Most segmentations obtained from images with high contrast and well-marked features provided extremely good segmentations that were clear and meaningful in their regions. In cases with subtle gradients or less-well-defined boundaries, the results were less useful since the regions did not correspond to any meaningful part of the image.

This, in general, works for images with good contrast and well-defined features but cannot work when the transitions are smooth or when there is too many fine details. Further improvements could be sought by investigating structured site placement or parallel computing that would lighten the computational load for higher values of k.