Assignment 1: Color Quantization and Connected Component Analysis

Alperen Gezer – 2017405081

Color Quantization

For the purpose of color quantization, k-means algorithm is implemented. In the algorithm, steps below are taken:

1- Assign random or manually selected K cluster centers.
2- Calculate the Euclidian distance of each data point to the cluster centers and assign the nearest center's label accordingly.
3- Calculate the mean of points in same cluster and assign as new cluster centers.
4- Repeat 2 and 3 until convergence.

However, the method is for n datapoints with m values. Meaning, the function accepts 2D tensors. In the other hand, an image is a 3D tensor. Thus, we need to flatten the image.

Following applying k-means, every datapoint is updated to be same value as its cluster center, and then reshaped back to a 3D tensor, an image.

We get the results below with the given dataset.

As the algorithm is greedy, we may observe it getting stuck at local optimum points.

Connected Component Analysis

In certain applications, we need to apply a mask to detect shapes, patterns or objects. In a single image, there may be many separate objects. In order to count those objects, we apply a method called "Connected Component Analysis". The algorithm runs as explained below:

1- Take mask as input and create a segmentation array with the same dimensions.
2- Iterate over each pixel in the mask
3- if you encounter 1, check if it is also unlabeled in segmentation array. If it is already labeled, skip; if it is not labeled, label it as a new object.

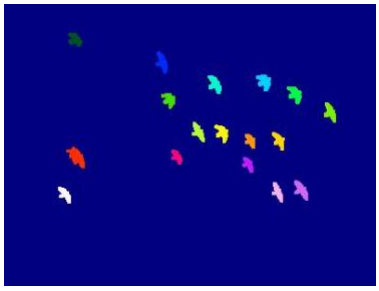
4- *If you declare a new object, check for its surroundings. If they are also 1 in the mask label as the same.*
5- *If you labeled a surrounding pixel, then check for its surroundings. If they are also 1 in the mask label as the same.*
6- *Repeat 5 recursively.*

This recursive algorithm finds separate pixel groups. However, sometimes mask has salt & pepper noises. In those cases, separation in a single object or little mask segments that are so small to be counted as objects are also considered. In order to prevent that, we apply erosion or dilation.

The results are given below. Algorithm also return object count as well as segmentation matrix.

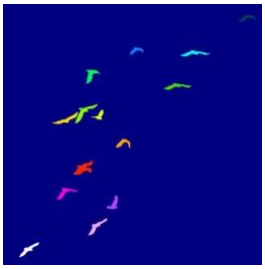
In special case of dice, I added a black padding to image. Because background is black and if there would be any separation that causes background to be detected as 2 objects, I wanted to connect it. It was not necessary for this case, but it can be for the future applications. Then, I counted background as an object and subtracted 1 from the resulting count.

In the case of okey tiles, I used HSV masking. I did not want to detect just tile count, I also wanted to detect red and blue tiles separately. I used saturation value to separate the tiles, then used hue value to separate between colors.



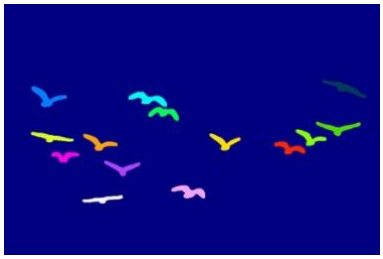
17 Birds

5 Points



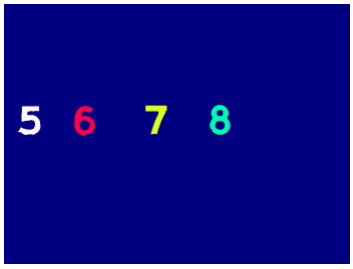
14 Birds

6 Points



14 Birds

4 Blue Tiles



4 Red Tiles