Lab 2: Segmentation

Course: CS195 - Computer Vision (Spring 2022)
Instructor: Md Alimoor Reza, Assistant Professor of Computer Science, Drake University
Due: Tuesday March 03, 11:50PM

We have discussed different segmentation methods. First, you need to implement clustering-based segmentation. Then you need to explore existing implementation for graph-based segmentation.

Task 1: Clustering Based Segmentation

As you were shown in class k-means clustering is an iterative algorithm that assigns each data point to one of k possible centroids. The algorithm is simple to describe and takes two inputs, a set of points to be clustered X and the number of clusters k. Each iteration of k-means is a two step process. First each point in X is assigned to the cluster center nearest to it. Secondly the cluster centers are updated by setting them equal to the average of all points in X assigned to the corresponding cluster.

In this lab, we will interpret each of our pixels in an image as part of the input data X. The algorithm below spells out exactly how we will perform the segmentation. This includes how we will initialize the locations of the cluster centers. Bit of warning, this approach can take several seconds for a big image.

Algorithm 1: k-means Image Segmentation

Data: Input image I and number of desired clusters k
Result: A matrix L, the size of I containing a corresponding cluster id for each pixel.

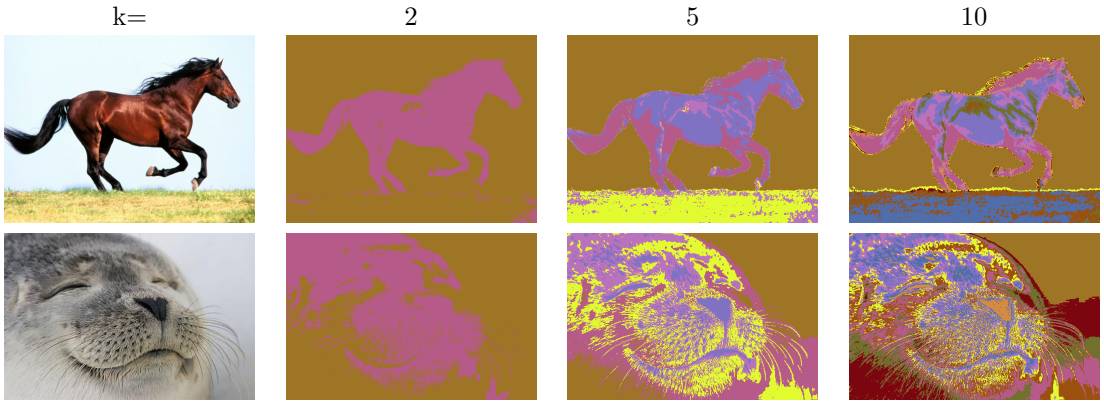
//Pick k random points in the image and initialize the k centers by copying over the RGB values.
for $j = 0$ **to** k **do**
  $x_{random} \leftarrow random(0, width - 1)$, $y_{random} \leftarrow random(0, height - 1)$
  $C(j) \leftarrow I(x_{random}, y_{random})$

//Repeat the k-means procedure for some iterations, lets say 100
for $i = 0$ **to** 100 **do**

  //Find the closest cluster in RGB distance to each pixel and record the ID
  for *each pixel I(x,y)* **do**
    $L(x, y) \leftarrow argmin_j ||I(x, y), C(j)||$

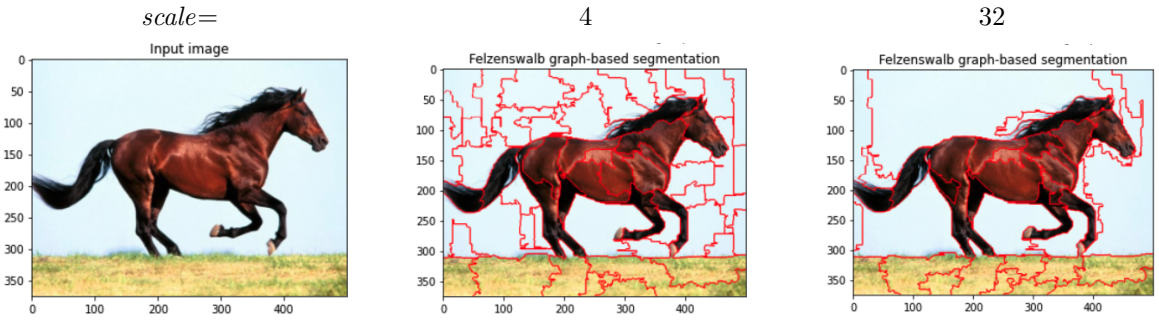
  //Update the centers by replacing the old values with the average of all assigned pixels
  for $j = 0$ **to** k **do**
    $C(j) \leftarrow \frac{1}{n_j} \sum I(x, y)$ for x, y such that $L(x, y) == j$
    and n_j is the number of pixels assigned to center j

After you have the matrix of labels L so that you know the assigned cluster for each pixel, you can assign a unique color to each centroid and paint corresponding pixels in the image that color to display the different segments. Please, do not import any existing k-means implementation from another library for this task. You will get a chance to do so for the next task.



Task 2: Graph Based Segmentation

Felsenszwalb's algorithm viewed an image as a graph for partitioning. The vertices in this graph represent image pixels. There are edges between adjacent pixels and edge weights give difference in color between pixels. The goal is to find a small number of homogeneous regions. It can be viewed as finding a set of connected components in the graph such that the sum of edge weights in each component is low. These connected components can be obtained by during the formation of a minimum spanning tree of the graph.Apply an existing implementation of **Felsenszwalb's** segmentation from ***skimage.segmentation.felzenszwalb***. You can vary the following three parameters: (1) *minsize*, (2) *scale*, and (3) *sigma*. *minsize* parameter determines the minimum size of a component (in pixel units). High value of *scale* will produce larger sized segments. *sigma* is the diameter of a Gaussian kernel, used for smoothing the image prior to segmentation. Vary the parameters and produce different segmentation outputs of an image. Observe how different parameter values affect the segmentation outcomes.



Two different segmentation outcomes for different scale parameters. Red boundaries overlaid on the image denote individual segments.

What to turn in:

Please submit a zip file containing your source code and the image(s) that you used for this lab. If you have any questions, please contact me.