CS 558: Homework Assignment 4 - Image Segmentation & Pixel Classification

Due: Dec 02, 5:59pm

Enrique Dunn
Department of Computer Science Stevens Institute of Technology
edunn@stevens.edu

Collaboration Policy. Homeworks will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems. Use of the Internet is allowed, but should not include searching for previous solutions or answers to the specific questions of the assignment. I will assume that, as participants in a graduate course, you will be taking the responsibility of making sure that you personally understand the solution to any work arising from collaboration.

Submission Format. Electronic submission on Canvas is mandatory. Submit a zip file containing:

- a pdf file with the resulting images and a brief explanation of the implementation.
- the code,
- the output images.

Problem 1: k-means Segmentation. (40 points) Apply k-means segmentation on white-tower.png with k=10. The distance function should only consider the RGB color channels and ignore pixel coordinates. Randomly pick 10 RGB triplets from the existing pixels as initial seeds and run to convergence. After k-means has converged, represent each cluster with the average RGB value of its members

Problem 2: SLIC. (40 points) Apply an approximate SLIC algorithm to white-tower.png and wt_slic.png, implementing the following steps:

1. Initialization: Divide the image in blocks of 50x50 pixels and initialize a centroid at the center of each block.
2. Local Shift: Compute the magnitude of the gradient in each of the RGB channels and use the square root of the sum of squares of the three magnitudes as the combined gradient magnitude. Move the centroids to the position with the smallest gradient magnitude in 3x3 windows centered on the initial centroids.

3. Centroid Update: Assign each pixel to its nearest centroid in the 5D space of x, y, R, G, B and recompute centroids. Use the Euclidean distance in this space, but divide x and y by proper scaling factor to account for differences in ranges between color and coordinate values (i.e. scaling = 2 is an initial/reasonable choice).
4. Optional (This makes estimation much more stable!): only compare pixels to centroids within a distance of 100 pixels (twice the block size) during the updates.
5. IF (not converged) and (iterations < max_iter) THEN go to step 2 . Use max_iter=3

6. Display the output image as in the SLIC slide: color pixels that touch two different clusters black and the remaining pixels by the average RGB value of their cluster.

Problem 3: Pixel Classification. (40 points) In this problem, the objective is to classify individual pixels as sky or non-sky.

1. Pick one of the sky images as training. Use gimp or a similar program to label the sky pixels in the that training image. I used white in the example below, but feel free to use any color. Use the "free select tool" to select the entire sky in the image and the "bucket fill tool" to color the selected region. Make sure that "quick mask" is off and that in the bucket fill tool options "fill whole selection" has been selected. Use "export" to save the image.
2. Use the image you created as a mask to separate sky from non-sky pixels during training. (Load both the original input image and the one you created above which is used as to guide the formation of the training set.)
3. Run k-means separately on the sky and non-sky sets with k = 10 to obtain 10 color *visual words* for each class.
4. For each pixel of the test images find the nearest word and classify it as sky or non-sky. (Brute force search is acceptable.)
5. Generate output images in which sky pixels are painted with a distinctive color.

Requirements and notes.

- You can use any programming language. You may have to explain the homework to me in person, if I am not familiar with your choice.
- You are allowed to use image reading and writing functions, as well as plotting functions, but you are not allowed to use filtering, edge detection or other image processing functions. You can convert the images to a different format for reading them.

- The complexity of k-means is linear with respect to the number of pixels and k. Start developing on small (i.e. scaled) images with small values of k.