Introduction

In your writeup, described the steps you completed for each problem and show the results. Readability will be part of your grade.

Please note that the instructor will be out of the country from Wednesday, October 27 to November 2. If you have questions during this time, please contact Paul Scovanner at pscovanner@cs.ucf.edu or Zain Massood at zain@cs.ucf.edu

1 Image Clustering with the \( k \)-means Algorithm

In class, we discussed how images could be segmented by grouping the pixels into clusters. In this problem, you will use the \( k \)-means algorithm to group pixels into segments.

1. Implement a function \( \text{im2featurevec} \) that takes an image with \( N_P \) pixels and returns a matrix (2D array) with \( N_P \) rows and \( M \) columns per row. Each of the \( M \) columns will denote a single feature response. Initially, there will be five features: \( R \), \( G \), \( B \), \( x \), and \( y \).

2. Implement the \( k \)-means algorithm. Assuming that there are \( N_C \) clusters, the basic steps of the algorithm are:
   - Initialize the algorithm by either initializing the cluster assignments or the cluster locations
   - Until convergence criterion is satisfied:
     - Assign each point to the nearest cluster center
     - Ensure that each center has at least one point assigned to it. One possible heuristic for filling in empty clusters is to choose a random point that is far from the cluster center.
     - Set each cluster center to the average of the points assigned to it. You will average each dimension of the vectors separately.

The input to your function should be an array of points and the number of cluster centers. The output should be the cluster to which each point is assigned and the cluster centers.

3. Write a function that takes an image and returns an image where each pixel contains the cluster number assigned to that cluster. Write a second function that takes this image and displays the boundaries. A simply method for finding the boundaries is to filter the image with vertical and horizontal derivative filters, compute the gradient magnitude, then threshold that value.

4. Test your function on the images provided.

5. Problem 1.1 does not ask you to normalize the feature vectors. To see the effect of normalizing the vectors, write a new function that creates the feature matrix as before, but now subtracts the mean
from each column, so that each column has a zero mean. In addition, the variance of each column should be divided by a constant so that the quantity:

\[ \frac{1}{N_p} \sum_{n=1}^{N_p} (F_n - \mu)^2 \]

is equal to 1. The term \( F_n \) denotes the feature at row \( n \) in a column. Compare the new results to the old results.

6. Experiment with different features. How do the results change?

2 SSD Stereo Matching

In class, we discussed how the depth of points in the scene could be obtained by finding corresponding points in stereo images. If the images are rectified, the epipolar lines correspond with the scanlines in the image. In this problem, you will use a simple method to find the diparity at each point in the image.

We’ll use the term disparity to denote the number of pixels between a point in the right image and the corresponding point in the left image. The disparity will always be a positive number. Using the Sum of Squared-Distances heuristic, the disparity with

\[
d(x, y) = \arg \min_i \sum_{<m,n> \in W(x,y)} (r(m, n) - l(m + i, n))^2
\]

(1)

where \( <m, n> \in W(x, y) \) denotes all the locations in a window around \((x, y)\).

1. Implement a function that takes a left image and right image, maximum disparity value, and windows size. This function should return the disparity at each point. If your results look extremely bad, it is likely that your maximum disparity is too small. You should manually verify that the maximum disparity is large enough

2. Experiment with different window sizes, including a one-pixel window. What are the tradeoffs?