CS 558: Homework Assignment 4 - Visual Recognition
Due: Oct 9, 11:59pm

Enrique Dunni
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 source code (excluding libraries), the output of Problem 1, the resulting images for Problem 2 and a brief explanation of the implementation.
- the code,
- the output images for Problem 2.

Problem 1: Image Classification. (50 points) Using the images in the ImClass directory, The objective of this problem is to classify each image of the **test** set into one of three classes: coast, forest or "insidecity". The representation will be in the form of three separate histograms of the R, G and B color channels. Each histogram will have 8 bins. Therefore, each image will be represented by 24 numbers. These representation should be computed for all images in the training set. The class labels of the 12 images in the training set will be considered known.

When computing the histograms make sure that all pixels are counted exactly 3 times, once in each color channel. Include a verification step that will be submitted with your code.

During testing, each image is classified independently of all other images in the test set. Use the same function to compute the representation and assign to the test image the label of the training image that has the "nearest" representation. The "nearest" representation should be computed using the Euclidean distance in the 24D histogram space. In other words, use the 1-nearest neighbor classifier. (You can use brute force search for the nearest neighbor due to the small size of the training set.)

Your code should print a sting like the following for each image:

```
Test image 1 of class 2 has been assigned to class 1.
```

Compute the accuracy of your classifier and include it in the report.

See if you can improve the accuracy of the classifier by changing the number of bins in the histograms.

Problem 2: Pixel Classification. (50 points) In this problem, the objective is to classify individual pixels as sky or non-sky.

1. Use gimp or a similar program to label the sky pixels in the first 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 *visual words* for each class.

4. For each pixel of the test image find the nearest word and classify it as sky or non-sky. (Brute force search is acceptable.)

5. Generate an output image in which sky pixels are painted with a distinctive color.

Test your code on all test images. Discuss why it works and where it fails. Suggest ideas for improving the accuracy of the classifier (no implementation required.)

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. You can convert the images to a different format for reading them.
- You are allowed to use a built-in k-means function without being penalized. In Matlab use the following syntax to avoid annoying errors:

  ```
  kmeans(sky, k, 'EmptyAction', 'singleton');
  ```

  This creates a new cluster in case empty clusters are generated during initialization.
- You are also allowed to use a k-Nearest Neighbors data structure. This will not affect your grade.

Acknowledgment: The data for these problems have been made available by Aude Oliva and Antonio Torralba, to whom we are grateful.