Grouping images into semantically meaningful categories is an important problem in computer vision. This problem has been traditionally tackled by using holistically computed low-level visual features. One of the popular features involves edge orientation histograms that are computed in terms of the gradient orientations of the edge pixels detected in images. This homework assignment involves a basic attempt for organizing the content of a data set using edge orientation histograms. The motivation for this is that man-made structures generally have regular edge features oriented in only a few directions while natural image regions have randomly oriented edges as illustrated in Figures 1 and 2. Different mixtures of man-made and natural structures will result in a variety of descriptions.

The approach can be summarized in terms of the following steps:

1. Prepare data. Download the land use image data set from the course web site. It includes a subset of the data set at http://vision.ucmerced.edu/datasets/landuse.html with 10 scene categories and 50 images in each category for a total of 500 images (Y. Yang, S. Newsam, “Geographic Image Retrieval Using Local Invariant Features,” IEEE Transactions on Geoscience and Remote Sensing, vol. 51, no. 2, pages 818–832, 2013. You can see example images in Figure 3.
2. **Perform edge detection.** Run the Canny edge detector on each image after converting it to gray scale. The output of this step is a binary edge image that marks pixels that represent a significant change as well as the gradient orientation values for these pixels. You have to experiment with the parameters to obtain important edges that will be useful in the following steps. After evaluating different parameter values for a subset of the data set, you must fix them and use the same values for all images in the rest of the assignment.

3. **Compute edge orientation histograms.** The orientation values are typically in the range $[0, 2\pi]$. You can divide this range into uniform bins and compute a histogram. You can also normalize the histogram to make it sum to 1. You have to experiment with the number of bins to find a good representation.

4. **Cluster the data set using edge orientation histograms.** Use the $k$-means algorithm to cluster the data set. You have to experiment with the number of clusters and also try different initializations.

5. **Compute the performance criterion.** You should use the entropy based performance measure described at the end of this document, and evaluate the performance for different number of clusters.
6. *Discuss your results.* You should discuss the results of each step and compare how the performance is affected by different parameters (i.e., Canny parameters, number of bins, number of clusters) based on your expectations and your observations on the resulting clusters and entropy plots. Provide examples for cases where the edge orientation histograms worked well and for cases where they were not that effective in modeling the structure.

7. *Proposal for improvement.* Discuss how the models can be improved for more effective characterization of the content. *(Hint: You are still asked to use edges in this step but can process them differently for better characterization.)*

Figure 3: Example images from the assignment data set. From top to bottom, the rows belong to the following categories: agricultural, beach, buildings, denseresidential, forest, freeway, harbor, intersection, parkinglot, storagetanks.
Software:

- You can use the edge detector in Matlab’s Image Processing Toolbox, or use the C code provided at [http://www.cs.washington.edu/education/courses/cse455/07wi/software/index.html](http://www.cs.washington.edu/education/courses/cse455/07wi/software/index.html). The Canny edge detector code in the link above has an optional argument to output the orientation values as a separate image, but you have to compute them separately if you use Matlab’s implementation.

- Matlab’s Statistics and Machine Learning Toolbox has an implementation of $k$-means clustering. You can also use any other implementation.

- You are allowed to use either the Matlab function or the C code listed above for step 2. You are not allowed to use code from other sources without the permission of the instructor. Furthermore, you are required to write your own code for steps 3 and 5. You are allowed to use code from other sources for the $k$-means clustering algorithm in step 4 with proper citation. You are required to know how each function that you use works.

Submit:

1. Example results for edge detection on images with different characteristics. Results on at least 10 different images (at least one image per scene category) are required. You must provide results for different parameters. You must also cite the edge detector implementation that you used.

2. Well-documented code in ASCII format for computing the edge orientation histogram using the edge detection results and gradient orientations.

3. Example edge orientation histograms for the images you used in (1) above. You must provide results for different number of bins.

4. Example qualitative results for clustering. You must provide visual results for different number of clusters. You must also cite the $k$-means implementation that you used, and report the initialization procedure and stopping criterion you used in the implementation.

5. Quantitative performance evaluation in terms of entropy plots for different settings.

6. Detailed discussion of the results.

7. Discussion for improvement. You will get bonus points if you implement these ideas and provide results.

Notes:

This assignment is due by midnight on Monday, November 28, 2016. You should upload your solutions as a single archive file that contains your code, resulting images, descriptions of how you obtained them, and discussion of the results using the online submission form on the course web page before the deadline. Please see the course syllabus for a discussion of the late homework policy as well as academic integrity. If you have any questions about what is allowed and what is not allowed in a solution, please check the course syllabus on the course web page.
Quantitative evaluation of clustering:
To evaluate the accuracy of clustering, goodness measures for both individual clusters and the overall clustering must be defined. One way of defining these measures involves the use of ground truth data where the resulting groups are compared to the known labels. In other words, the measures should quantify how well the results of the unsupervised clustering algorithm reflect the groupings in the ground truth.

In an optimal result, the images of the same category must be assigned to the same cluster and the images corresponding to different categories must appear in different clusters. An information theoretic criterion that measures the homogeneity of the distribution of the clusters with respect to different categories is the entropy. Let \( h_{ck} \) denote the number of images assigned to cluster \( k \) with a ground truth category label \( c \), \( h_c = \sum_{k=1}^{K} h_{ck} \) denote the number of images with a ground truth category label \( c \), and \( h_{c,k} = \sum_{c=1}^{C} h_{ck} \) denote the number of images assigned to cluster \( k \), where \( K \) is the number of clusters given as input to the clustering algorithm and \( C \) is the number of categories. The quality of individual clusters is measured in terms of the homogeneity of the category labels within each cluster. For each cluster \( k \), the cluster entropy \( E_k \) is given by

\[
E_k = -\sum_{c=1}^{C} \frac{h_{ck}}{h_k} \log \frac{h_{ck}}{h_k}.
\]

(1)

Then, the overall cluster entropy \( E_{\text{cluster}} \) is given by a weighted sum of individual cluster entropies as

\[
E_{\text{cluster}} = \frac{1}{\sum_{k=1}^{K} h_k} \sum_{k=1}^{K} h_k E_k.
\]

(2)

A smaller cluster entropy value indicates a higher homogeneity. However, the cluster entropy continues to decrease as the number of clusters increases. To overcome this problem, another entropy criterion that measures how images of the same category are distributed among the clusters can be defined. For each category \( c \), the class entropy \( E_c \) is given by

\[
E_c = -\sum_{k=1}^{K} \frac{h_{ck}}{h_c} \log \frac{h_{ck}}{h_c}.
\]

(3)

Then, the overall class entropy \( E_{\text{class}} \) is given by a weighted sum of individual class entropies as

\[
E_{\text{class}} = \frac{1}{\sum_{c=1}^{C} h_c} \sum_{c=1}^{C} h_c E_c.
\]

(4)

Unlike the cluster entropy, the class entropy increases when the number of clusters increase. Therefore, the two measures can be combined for an overall entropy measure as

\[
E = \beta E_{\text{cluster}} + (1 - \beta) E_{\text{class}}
\]

(5)

where \( \beta \in [0, 1] \) is a weight that balances the two measures.

Use the measure defined in (5) to evaluate the consistency of the clusters found for each choice of the number of clusters (you can use \( \beta = 0.5 \)), and plot the overall entropy vs. the number of clusters.