Project 2. Color and Texture Features

Due Wed February 27, 2019

CMPE 685 Teaching Assistant: Nilesh Pandey np9207@rit.edu

Objectives: The objectives of this project are:
- Implement k-means clustering and use it to perform image segmentation
- Generate color histogram features and use them for image retrieval
- Generate texture features and use them for texture matching

Project Report:
A pdf version of the report (pdf and zipped overleaf project), images (zipped), and all code (zipped) should be uploaded on mycourses by noon on the due date. The project report in paper form (without the code) is due at the beginning of the next class. There is no page limit, but the report should include the following sections: Introduction, a Section for each Topic that includes Methods and Results as subsections, Discussion, References (include URLs of images and other references as appropriate) and Appendix with code listing. Use color figures in the pdf version of the report and grayscale images in the figures of the printed version. Make sure you maintain the image aspect ratio when displaying images in the figures. The report will be graded based on methods used, results obtained, interpretation of results, answering of questions, clarity of presentation, discussion and English. **(10 points)**
Hint: You can write your own latex format document on overleaf. For writing equations, you could use https://www.codecogs.com/latex/eqneditor.php

Important: This project should be done in Python.

1. Color image segmentation using k-means (25 points)

a) Color Histogram. Read a color image and create 3 histograms for the Red, Green and Blue channels. The image that you select should be natural (no cartoon, text or graphics images). Then create a 384 bin quantized color histogram, obtained by quantizing the bins on the separate R, G, B histograms and then concatenating all the bins into one histogram. Plot these histograms next to the image and discuss their representation of the image.

b) K-means Implementation. Write your own Python function that performs k-means clustering. For a description of the k-means clustering algorithms see http://en.wikipedia.org/wiki/K-means_clustering and the class notes. You may check your function by running a simple example and comparing the results with Sklearn's kmeans function https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
You may explore (but not copy) code from other sources as the basis for your Python function, as long as you reference these sources (e.g. URL or book reference).

c) K-means Segmentation based on Intensity. Perform segmentation of a grayscale image based on k-means clustering of the intensity code values. If you are using a color image convert it to grayscale using the intensity component I=(R+G+B)/3 and display the grayscale image. Select k based on the number of important regions in the image. Display the original color image, the gray scale intensity image and the segmented image where each segment has a different color or gray tone. Write your own Python code.

d) Repeat the above k-means segmentation procedure for a small value of k and a large value of k. Display the segmentation results.
e) K-means Segmentation based on Color. Perform segmentation of the color image based on k-means color clustering, where each color pixel is represented in RGB space. Use the same values for k as in the earlier part, including a small value of k and a large value of k. Display the original color image and the segmented images where each segment has a different color or gray tone. Write your own Python code.
f) Discuss the k-means segmentation results using the gray scale intensity image and the color image. Comment on how the choice of k affects the segmentation results.

2. Image retrieval using color histogram features (25 points)

a) Download the color image dataset from mycourses and compute the 96 bin quantized color histogram of each image. Histogram functions are allowed to be used.
b) Select an image from each class of the database and find the 4 closest matches based on the histogram intersection distance. Write your own code for all distance computations. Create a table that reports the class of the query image and the class of the 4 closest matches retrieved in each case.
c) Repeat the above experiment with a different image from each class. Create a table that reports the class of the query image and the class of the 4 matches retrieved in each case.

Table with class information of query and 4 best matches
(2 query images from each class)

Query	Match1	Match2	Match3	Match4
Rose	Rose	Rose	Foods	Bus
…	…	…	…	

d) Create a figure showing the query image and your top 3 retrieval results for 2 cases that work well and one case that does not work very well.

Table with images of query and 3 best matches
(3 examples: 2 that return good matches and 1 that does not)

Query	Match1	Match2	Match3
	(show image)	(show image)	(show image)
…	…	…	…

e) Discuss the results and comment on the advantages and limitations of color-based image retrieval.

3. Image retrieval using texture features (20 points)

a) Download the texture image dataset and for each image, use the intensity (gray scale) component to compute a 9-dimensional texture feature based on Laws filter masks. Laws filter masks are defined as follows.
   - 1D Masks:
     - Level: L5 = [1 4 6 4 1]
     - Edge: E5 = [-1 -2 0 2 1]
     - Spot: S5 = [-1 0 2 0 -1]
     - Ripple: R5 = [1 -4 6 -4 1]

- 2D Masks:
  - Obtained by separable combinations of 1D masks in the vertical and horizontal.
  - For example, E5L5 uses E5 in the vertical and L5 in the horizontal.

$$\text{E5L5} = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 1 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

- For each resolution obtain a feature vector with 9 dimensions
  - Filter each image point (except near the border) using the following masks and compute the average of the absolute value of the filter response.
    L5E5, E5L5: average abs of both filter responses to obtain a single result.
    L5R5, R5L5: average abs of both filter responses to obtain a single result.
    L5S5, S5L5: average abs of both filter responses to obtain a single result.
    E5R5, R5E5: average abs of both filter responses to obtain a single result.
    E5S5, S5E5: average abs of both filter responses to obtain a single result.
    S5R5, R5S5: average abs of both filter responses to obtain a single result.
    E5E5: average abs of filter responses across the image to obtain a single result.
    R5R5: average abs of filter responses across the image to obtain a single result.
    S5S5: average abs of filter responses across the image to obtain a single result.

b) Select an image from each texture class and find the 4 closest matches based on the Chi-Square distance of the feature vectors.
c) Repeat the above experiment with a different image from each class. Create a table that reports the class of the query image and the class of the 4 matches retrieved in each iteration. Create figures showing your query image and your retrieval results for 2 cases that work well and one case that does not work very well.
d) When comparing textures, you may be able to improve your retrieval results by changing the scale of the image by resizing to various resolutions. Experiment with resizing the image to find the best match across scales. Generate a table that shows the class of the matches retrieved in each case of resizing. Create figures with the best and worst retrieval results for each class.
e) Discuss the results and advantages/limitations of texture matching with this feature vector.

4. Texture Matching using Gabor texture features (20 points)

a) Download the texture image dataset and for each image, use the intensity (gray scale) component to compute texture features based on Gabor filters.
b) Experiment with the number of Gabor filters used to generate your texture features. Select between 7 and 38 filters at various frequencies, scales, and orientations. Display all of your Gabor filters and their frequency responses. You may use Python functions or libraries to generate your Gabor filters.
c) Select an image from each texture class and find the 4 closest matches based on the Chi-Square distance of the feature vectors.
d) Repeat the above experiment with a different image from each class. Create a table that reports the class of the query image and the class of the 4 matches retrieved in each iteration. Create figures showing your query image and your retrieval results for 2 cases that work well and one case that does not work very well.
e) Discuss the results and advantages/limitations of texture matching with Gabor filters.