EE 569: Homework #3

Issued: 10/7/2016 Due: 11:59PM, 11/6/2016

General Instructions:

1. Read Homework Guidelines and MATLAB Function Guidelines for the information about homework programming, write-up and submission. If you make any assumptions about a problem, please clearly state them in your report.
2. Do not copy sentences directly from any listed reference or online source. Written reports and source codes are subject to verification for any plagiarism. You need to understand the USC policy on academic integrity and penalties for cheating and plagiarism. These rules will be strictly enforced.

Problem 1: Texture Analysis and Segmentation (30 % + 10%)

Construct twenty-five 5x5 Laws Filter using the filters in Table 1.

Table 1: 1D Kernel for 5x5 Laws Filters

Name	Kernel
L5 (Level)	[1 4 6 4 1]
E5 (Edge)	[-1 -2 0 2 1]
S5 (Spot)	[-1 0 2 0 -1]
W5(Wave)	[-1 2 0 -2 1]
R5 (Ripple)	[1 -4 6 -4 1]

(a) Texture Classification (Basic: 10%)

Twelve texture images, from *texture1* to *texture12* (size 128x128) are provided for the texture classification task. Samples of these images are shown in Figure 1. Categorize them into four texture types (rock, grass, weave and sand) with three images for each type. Please follow steps below to complete this problem.

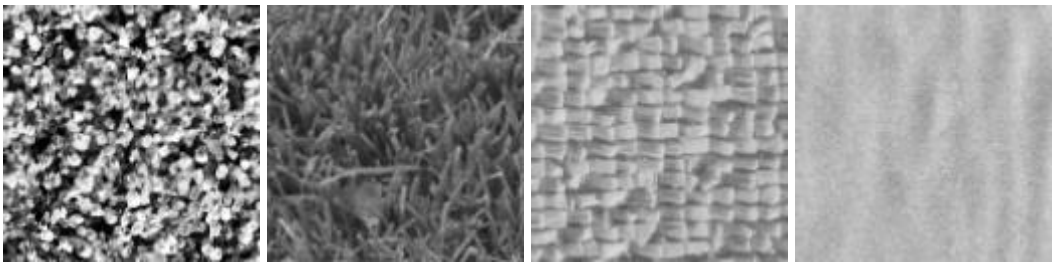


Figure 1: Four types of textures: rock, grass, weave, and sand

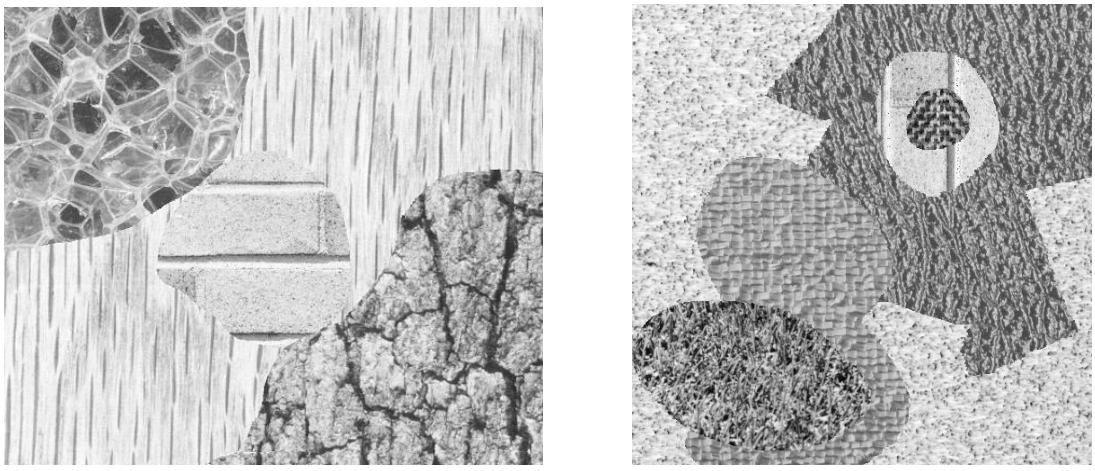
1. Use the twenty-five 5x5 Laws Filters to extract feature vectors from each pixel in the image (use appropriate boundary extensions).
2. Average the feature vectors of all image pixels, leading to a 25-D feature vector for each image. Which feature dimension has the strongest discriminant power? Which has the weakest? Please justify your answer.

3. Reduce the feature dimension from 25 to 3 using the principal component analysis (PCA). Plot the reduced 3-D feature vector in the feature space.
4. Use the K-means algorithm for image clustering based on the 25-D and 3-D obtained in Steps 2 and 3, respectively. Discuss the effectiveness of feature dimension reduction over K-means.

You may use built-in C/Matlab functions of PCA. Report and analyze your results.

(b) Texture Segmentation (Basic: 20%)
In this part, apply the twenty-five 5x5 Laws Filters to texture segmentation for the image shown in Figure 2 with the following steps.

1. **Laws feature extraction:** Apply all 25 Laws filters to the input image and get 25 gray–scale images.
2. **Energy feature computation:** Use a window approach to computer the energy measure for each pixel based on the results from Step 1. You may try a couple of different window sizes. After this step, you will obtain 25-D energy feature vector for each pixel.
3. **Energy feature normalization:** All kernels have a zero-mean except for $L5^T L5$. Actually, the feature extracted by the filter $L5^T L5$ is not a useful feature for texture classification and segmentation. Use its energy to normal all other features at each pixel.
4. **Segmentation:** Use the k–means algorithm to perform segmentation on the composite texture images given in Figure 2 based on the 25-D energy feature vectors.



(a) comb_1 image (b)comb_2 image

Figure 2: Two texture mosaic images for segmentation

If there are K textures in the image, your output image will be of K gray levels, with each level represents one type of texture. For example, there are six textures in Figure 2(b) you can use six gray levels (0, 51, 102, 153, 204, 255) to denote six segmented regions in the output image.

(c) Advanced Texture Segmentation Techniques (Bonus: 10%)

You may not get good segmentation results for the complicated texture mosaic image in Figure 2(b). Please develop various techniques to enhance your segmentation result. Several ideas are sketched below.

1. Consider the segmentation task in two spirals – the first spiral gives the coarse segmentation result while the second spiral offers fine-tuned segmentation results.
2. Develop a post-processing technique to merge small holes.
3. Enhance the boundary of two adjacent regions by focusing on the texture properties in these two regions only.
4. Adopt the PCA for feature reduction and, thus, cleaning.

Problem 2: Salient Point Descriptors and Image Matching (30 %)

(a) Extraction and Description of Salient Points (Basic: 10%)

SIFT [1] and SURF [2] are effective tools to extract salient points in an image. Extract and show both SIFT and SURF features from the two vehicle images in Figure 3. Compare their results in terms of performance and efficiency according to their strength and weakness. You are allowed to use open source library (OpenCV or VLFeat) to extract features.

(a) Jeep

(b) Bus

Figure 3: Vehicle images

(b) Image Matching (Basic: 10%)

You can apply SIFT and SURF to object matching. Extract and show SIFT features from the two rav4 images in Figure 4. Then, show the corresponding SIFT pairs between the two images. Repeat the same task for the SURF feature.

(a) Rav4_1

(b) Rav4_2

Figure 4: Rav4 images

The matching may not work well between different objects and against the same object but with a large viewing angel difference. Perform the same job with the following two image pairs:1) *Rav4_1* and *Jeep*, and 2) *Rav4_1* and *Bus*. Show and comment on the matching results. Explain why it works or fails in some cases.

(c) Bag of Words (Advanced: 10%)

Apply the k-means clustering to extracted SIFT features from three images (Jeep, Bus, and Rav4_1) to form a codebook. The codebook contains K=8 bins, where each bin is characterized by the centroid of the SIFT feature vector. In other words, each image can be represented as a histogram of SIFT feature vectors. This representation is called the Bag of Words (BoW). Create codewords for all four images (Jeep, Bus,

Rav4_1 and Rav4_2), and match Rav4_2's codewords with other images. Show the results and discuss your observation.

Problem 3: Edge Detection (40 %)

(a) Canny Edge Detector (Basic: 15%)

The Canny edge detector is an edge detection technique utilizing image's intensity gradients and non-maximum suppression with double thresholding. In this part, apply the Canny edge detector [3] to both *Zebra* and *Jaguar* images as shown in Figure 5. You are allowed to use any online source code such as the Canny edge detector in the Matlab image processing toolbox or the OpenCV (i.e. Open Source Computer Vision Library). Generate edge maps by trying different low and high thresholds.

Figure 5: Zebra and jaguar images

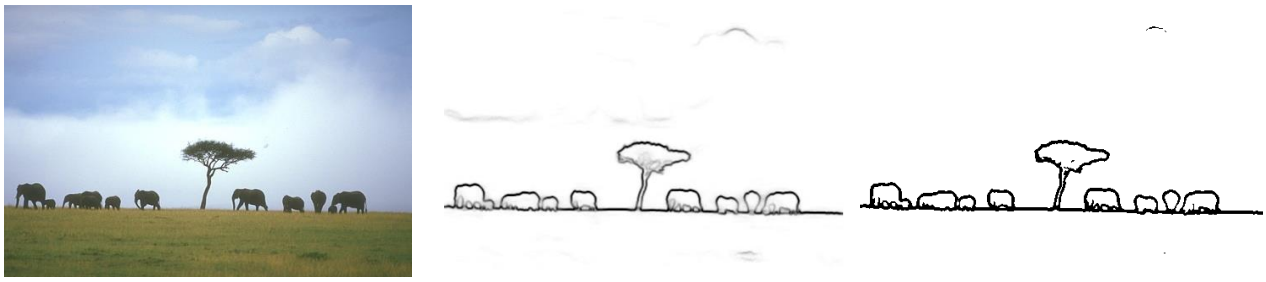
Answer following questions:

1. Compare and show the resulting edge maps obtained by different thresholding inputs. Which image performs best using the Canny edge detector? Can you tell object's location from the edge map?
2. Can you tune the parameter to show only the edges on the object? Why or why not?
3. Explain the relationship between the threshold values and image content (e.g., texture, smooth, weak object boundary regions).

(b) Structured Edge (Advanced: 15%)

Apply the Structured Edge (SE) detector [4] to extract edge segments from a color image with online source codes. Exemplary edge maps generated by the SE method for the *Elephant* image are shown in Figure 6. You can apply the SE detector to the RGB image directly without converting it into a grayscale image. Also, the SE detector will generate a probability edge map. To obtain a binary edge map, you need to binarize the probability edge map with a threshold.

1. Please digest the SE detection algorithm. Summarize it with a flow chart and explain it in your own words (no more than 1 page, including both the flow chart and your explanation).
2. The Random Forest (RF) classifier is used in the SE detector. The RF classifier consists of multiple decision trees and integrate the results of these decision trees into one final probability function. Explain the process of decision tree construction and the principle of the RF classifier.
3. Apply the SE detector to *Zebra* and *Jaguar* images. State the chosen parameters clearly and justify your selection. Compare and comment on the visual results of the Canny detector and the SE detector.

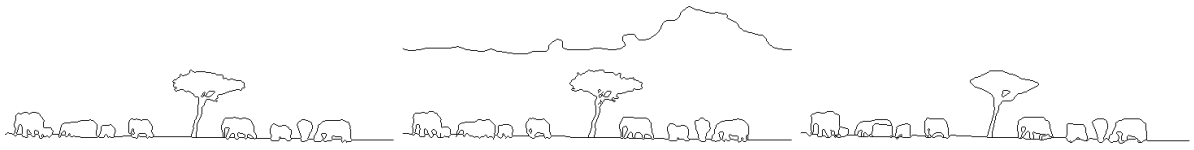


Elephant Probability edge map Binary edge map (with p>0.25)

Figure 6: The Elephant image and its probability and binary edge maps obtained by the SE detector

(c) Performance Evaluation (Advanced: 10%)

Perform quantitative comparison between different edge maps obtained by different edge detectors. The ultimate goal of edge detection is to enable the machine to generate contours of priority to human being. For this reason, we need the edge map provided by human (called the ground truth) to evaluate the quality of a machine-generated edge map. However, different people may have different opinions about important edge in an image. To handle the opinion diversity, it is typical to take the mean of a certain performance measure with respect to each ground truth, e.g. the mean precision, the mean recall, etc. Figure 7 shows 6 ground truth edge maps for the *Elephant* image from the Berkeley Segmentation Dataset and Benchmarks 500 (BSDS 500) [5].



Ground Truth 1 Ground Truth 2 Ground Truth 3

Ground Truth 4 Ground Truth 5 Ground Truth 6

Figure 7: Six ground truth edge maps for the Elephant image

To evaluate the performance of an edge map, we need to identify the error. All pixels in an edge map belong to one of the following four classes:

1. True positive: Edge pixels in the edge map coincide with edge pixels in the ground truth. These are edge pixels the algorithm successfully identifies. (Green pixels in Figure 8)

2. True negative: Non-edge pixels in the edge map coincide with non-edge pixels in the ground truth. These are non-edge pixels the algorithm successfully identifies.
3. False positive: Edge pixels in the edge map correspond to the non-edge pixels in the ground truth. These are fake edge pixels the algorithm wrongly identifies. (Blue pixels in Figure 8)
4. False negative: Non-edge pixels in the edge map correspond to the true edge pixels in the ground truth. These are edge pixels the algorithm misses. (Red pixels in Figure 8)

Figure 8: The error map of the SE edge map in Figure 7

Clearly, pixels in (1) and (2) are correct ones while those in (3) and (4) are error pixels of two different types to be evaluated. The performance of an edge detection algorithm can be measured using the F measure, which is a function of the precision and the recall.

$$\begin{aligned} \text{Precision}: P &= \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Positive}} \\ \text{Recall}: R &= \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Negative}} \\ F &= 2 \times \frac{P \times R}{P + R} \end{aligned} \tag{1}$$

One can make the precision higher by decreasing the threshold in deriving the binary edge map. However, this will result in a lower recall. Generally, we need to consider both precision and recall at the same time and a metric called the F measure is developed for this purpose. A higher F measure implies a better edge detector.

For the ground truth edge maps of *Zebra* and *Jaguar* images (5 and 6 images respectively), evaluate the quality of edge maps obtained in Parts (a)-(c) with the following:

1. Calculate the precision and recall for each ground truth separately using the function provided by the SE software package and, then, compute the mean precision and the mean recall. Finally, calculate the F measure for each generated edge map based on the mean precision and the mean recall. Please use a table to show the precision and recall for each ground truth, their means and

the final F measure. Comment on the performance of different edge detectors (i.e. their pros and cons.)

2. The F measure is image dependent. Which image is easier to a get high F measure - Zebra or Jaguar? Please provide an intuitive explanation to your answer.
3. Discuss the rationale behind the F measure definition. Is it possible to get a high F measure if precision is significantly higher than recall, or vice versa? If the sum of precision and recall is a constant, show that the F measure reaches the maximum when precision is equal to recall.

Appendix:

Problem 1: Texture Analysis and Segmentation

Texture1 to 12.raw	128x128	8-bit	Gray
Comb_1.raw	500x425	8-bit	Gray
Comb_2.raw	512x512	8-bit	Gray

Problem 2: Edge Detection

Jeep.raw(.jpg)	500x380	24-bit	Color(RGB)
Bus.raw(.jpg)	450x248	24-bit	Color(RGB)
Rav4_1.raw(.jpg)	500x256	24-bit	Color(RGB)
Rav4_2.raw(.jpg)	500x256	24-bit	Color(RGB)

Problem 3: Image Segmentation

Zebra.raw	481x321	24-bit	Color(RGB)
Jaguar.raw	481x321	24-bit	Color(RGB)
Elephant.raw	481x321	24-bit	Color(RGB)
Elephant_prob.raw	481x321	8-bit	Gray
Elephant_binary.raw	481x321	8-bit	Gray
Elephant_error.raw	481x321	24-bit	Color(RGB)
Elephant_gt1 to 6.raw	481x321	8-bit	Gray
Zebra_gt1 to 5.raw	481x321	8-bit	Gray
Jaguar_gt1 to 6.raw	481x321	8-bit	Gray

Reference Images

Images in this homework are taken from Google images [6], the USC-SIPI image database [7] or the BSDS 500 [5].

References

[1] David G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, 60(2), 91-110, 2004.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, "SURF: Speeded Up Robust Features", *Computer Vision and Image Understanding (CVIU)*, Vol. 110, No. 3, pp. 346--359, 2008

[3] Canny, John. "A computational approach to edge detection." Pattern Analysis and Machine Intelligence, IEEE Transactions on 6 (1986): 679-698.

[4] Dollár, Piotr, and C. Lawrence Zitnick. "Structured forests for fast edge detection." Computer Vision (ICCV), 2013 IEEE International Conference on. IEEE, 2013.

[5] Arbelaez, Pablo, et al. "Contour detection and hierarchical image segmentation." Pattern Analysis and Machine Intelligence, IEEE Transactions on 33.5 (2011): 898-916.

[6] [Online] http://images.google.com/

[7] [Online] http://sipi.usc.edu/database/