EE 569: Homework #3

Issued: 2/23/2018 Due: 11:59PM, 3/25/2018

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%)

In this problem, you will implement texture analysis and segmentation algorithms based on the 3x3 Laws filters discussed in the lecture or the 5x5 Laws filters constructed by the tensor product of the five 1D kernels 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]

1(a) Texture Classification (Basic: 10%)

In this part, please construct nine 5x5 Laws Filter using the following three filters:

$$E5 = \frac{1}{6}[-1 \;\; -2 \;\; 0 \;\; 2 \;\; 1], \qquad S5 = \frac{1}{4}[-1 \;\; 0 \;\; 2 \;\; 0 \;\; -1], \qquad W5 = \frac{1}{6}[-1 \;\; 2 \;\; 0 \;\; -2 \;\; 1],$$

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.

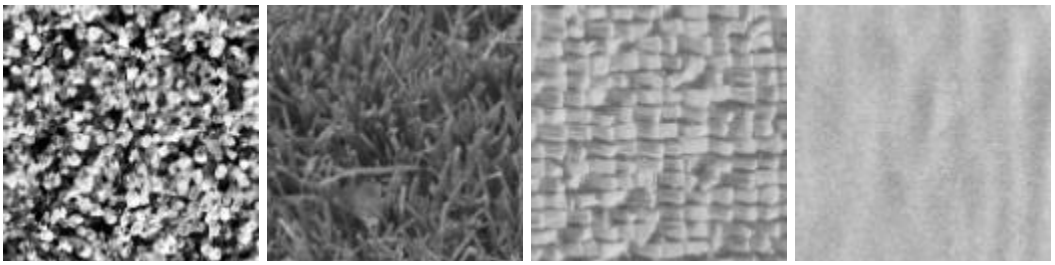


Figure 1: Four types of textures: rock, grass, weave, and sand

Please cluster them into four texture types with the following steps below to complete this problem.

1. Feature Extraction: Use the nine 5x5 Laws Filters to extract feature vectors from each pixel in the image (use appropriate boundary extensions).

2. Feature Averaging: Average the feature vectors of all image pixels, leading to a 9-D feature vector for each image. Which feature dimension has the strongest discriminant power? Which has the weakest? Please justify your answer.
3. Clustering: Use the K-means algorithm for image clustering based on the 9-D feature obtained in step 2.

Report your results and compare them with the reality (by checking the texture images by eyes). Discuss any discrepancy.

(b) Texture Segmentation (Basic: 20%)

In this part, apply the 3x3 Laws Filters to texture segmentation for the image shown in Figure 2 with the following steps.

1. **Laws feature extraction:** Apply all 3x3=9 Laws filters to the input image and get 9 gray–scale images.
2. **Energy computation:** Use a windowed 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 9-D energy feature vector for each pixel.
3. **Normalization:** All kernels have a zero-mean except for $L3^T L3$. Note that $L3^T L3$ is typically not a useful feature, and can be used for the normalization of all other features.
4. **Segmentation:** Use the k–means algorithm to perform segmentation on the composite texture images given in Figure 2. 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, you can use six gray levels (0, 51, 102, 153, 204, 255) to denote six segmented regions in the output image.

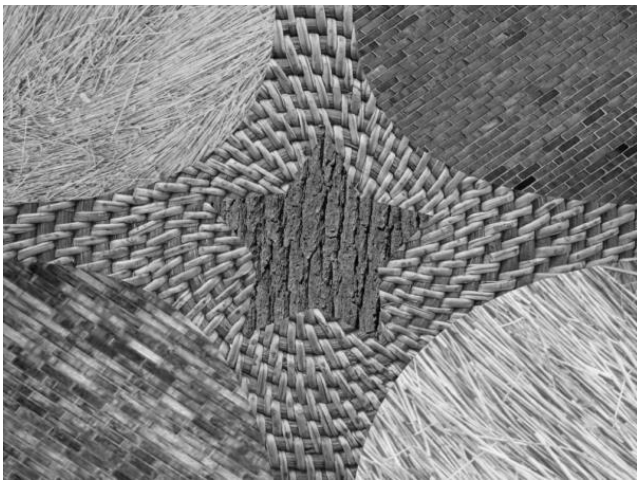


Figure 2: Composite texture image

(c) Advanced (Bonus: 10%)

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

1. Post-processing technique to improve your segmentation results based on your observation.
2. Adopt the PCA for feature reduction. You may adopt all twenty-five 5x5 Laws Filters and use the Principal Component Analysis (PCA) method to reduce the dimensions of the feature vectors. Use dimension reduced features to do texture segmentation of Figure 2. You may use built-in C/Matlab functions of PCA.

Problem 2: Edge Detection (40 %)

a) Basic Edge Detector (Basic: 10%)

Implement two edge detectors and apply them to clean and noisy *Boat* images as shown in Fig. 3 (a) and (b). Note that you need to convert RGB images to grey-level image first.

1) Sobel Detector
   - Normalize the x-gradient and the y-gradient values to 0-255 and show the results.
   - Tune the thresholds (in terms of percentage) to obtain your best edge map. An edge map is a binary image whose pixel values are either 0 (edge) or 255 (background)
2) Zero crossing Detector
   - Normalize the LoG response to 0-255 and show the results.
   - The LoG histogram is in form of a sharp symmetric spike. Develop an algorithm to determine the two knee locations of the LoG histogram. Then, you can use them as the threshold values to partition the LoG histogram into three values -1, 0 and 1. In order to show the corresponding ternary map, you can map -1, 0, 1 to gray-level 64, 128, 192.
   - Apply zero crossing to obtain your best edge map

Compare the results obtained in (1) and (2)

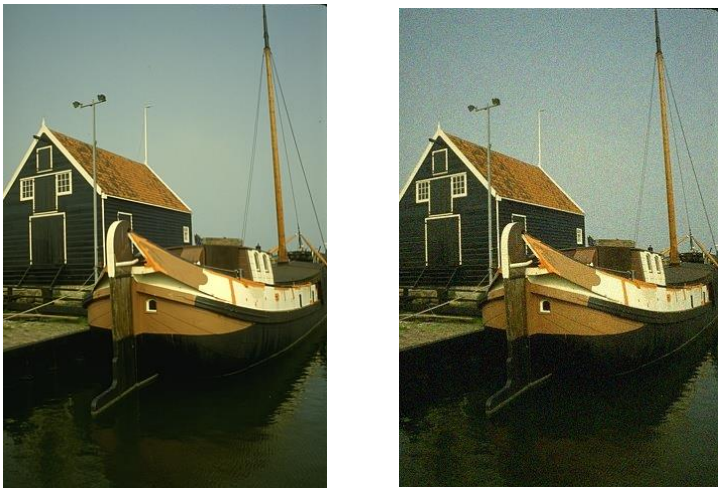


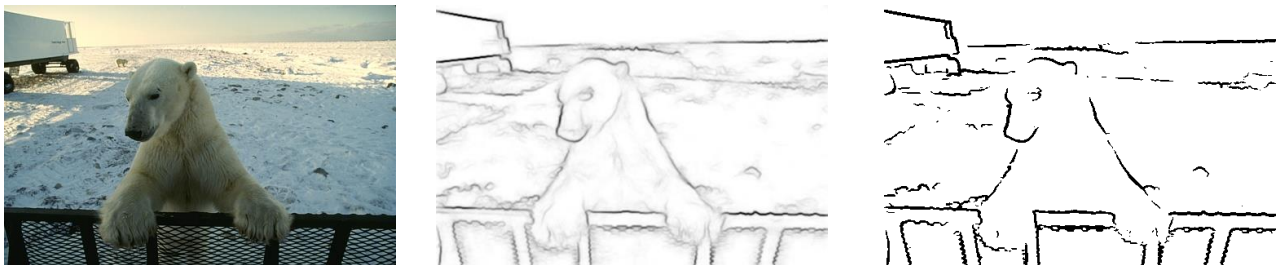
Fig. 3 (a) Original *Boat* image and (b) noisy *Boat* image

Figure 4: Animal and house images

b) Structured Edge (Advanced: 10%)

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 *Bear* image are shown in Figure 5. 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.

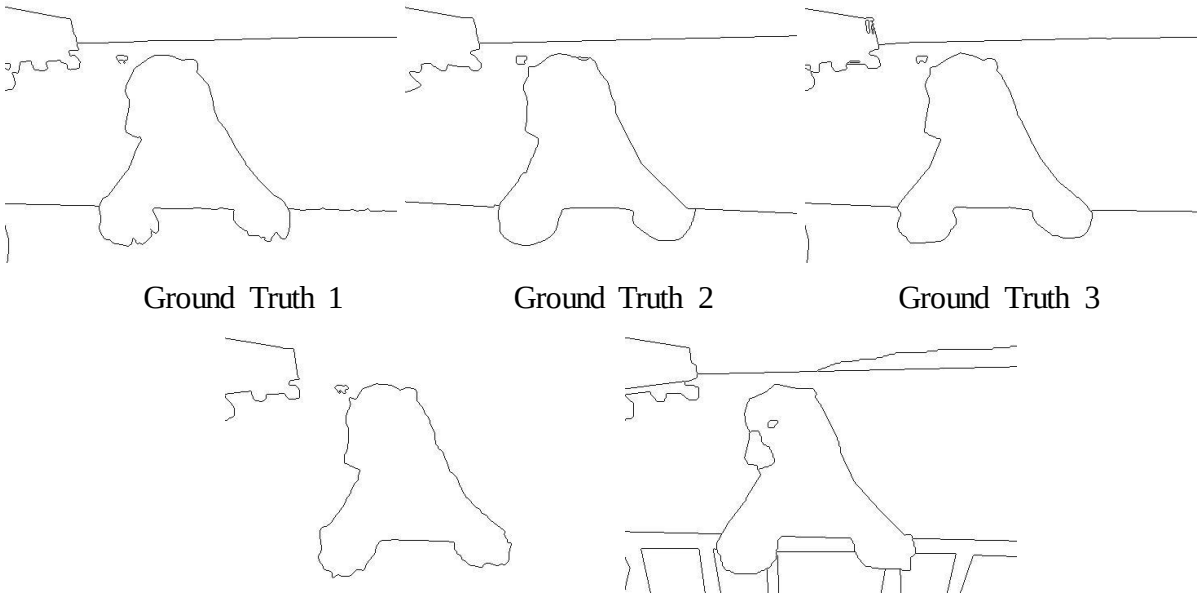
1. Please digest the SE detection algorithm. Summarize it with a flow chart and explain it in your own words (no more than 1 page, including both the flow chart and your explanation).
2. The Random Forest (RF) classifier is used in the SE detector. The RF classifier consists of multiple decision trees 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 *Animal* and *House* 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.



Bear | Probability edge map | Binary edge map (with p>0.2)

Figure 5: The *Bear* image and its probability and binary edge maps obtained by the SE detector

c) Performance Evaluation (Advanced: 20%)



Ground Truth 1 | Ground Truth 2 | Ground Truth 3

Ground Truth 4 | Ground Truth 5

Figure 6: Five ground truth edge maps for the *Bear* image

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 6 shows 5 ground truth edge maps for the *Bear* image from the Berkeley Segmentation Dataset and Benchmarks 500 (BSDS 500) [5]. 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.
(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.
(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.

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 *Animal* and *House* images (both 5 images), evaluate the quality of edge maps obtained in Parts (a)-(c) with the following:

1. Calculate the precision and recall for each ground truth (saved in .mat format) 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 - *Animal* or *House*? 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.

Problem 3: 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 7. 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) Optimus prime truck (b) Bumblebee car

Figure 7: Vehicle images

(b) Image Matching (Basic: 10%)

You can apply SIFT and SURF to object matching. Extract and show SIFT features from the two racing car images in Figure 8. Then, show the corresponding SIFT pairs between the two images. Repeat the same task for the SURF feature.

(a) Ferrari_1 (b) Ferrari_2

Figure 8: Ferrari racing car 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) *Ferrari _1* and *Optimus prime truck*, and 2) *Ferrari _1* and *Bumblebee car*. 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 (*Optimus prime truck*, *Bumblebee car*, and *Ferrari_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 (*Optimus prime truck, Bumblebee car, Ferrari_1 and Ferrari_2*), and match Ferrari_2's codewords with other images. Show the results and discuss your observation.

Appendix:

Problem 1: Texture Analysis and Segmentation

Texture1 to 12.raw	128x128	8-bit	Gray
Composite.raw	600x450	8-bit	Gray

Problem 2: Edge Detection

Boat.raw	321x481	24-bit	Color(RGB)
Boat_noisy.raw	321x481	24-bit	Color(RGB)
Bear.raw	481x321	24-bit	Color(RGB)
Bear_prob.raw	481x321	8-bit	Gray
Bear_binary.raw	481x321	8-bit	Gray
Animal.raw	481x321	24-bit	Color(RGB)
House.raw	481x321	24-bit	Color(RGB)
Bear_GT_1 to 5.jpeg	481x321	8-bit	Gray

Problem 3: Salient Point Descriptors and Image Matching

Optimus_prime.jpg	592x424	24-bit	Color(RGB)
Bumblebee.jpg	450x248	24-bit	Color(RGB)
Ferrari_1.jpg	470x220	24-bit	Color(RGB)
Ferrari_2.jpg	470x220	24-bit	Color(RGB)

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/