EE 569: Homework #2

Issued: 9/18/2015 Due: 11:59PM, 10/11/2015

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 Classification (30 %)

One effective way to extract features from texture is to filter an input texture image with a set of filters (called a filter bank) and compute the local energy from their output image. This is known as the "multichannel feature extraction" method. Implement a filter bank that consists of 25 Laws' filters of size 5x5 using the tensor product of 1D kernels as shown in Table 1:

Table 1: Construction of 2D Laws' Filters Using 1D Kernels

1D Kernel Name	Filter Coefficients
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: Two Classes + Minimum Mean Distance Classifier (Basic: 15%)

Tom, Mary and you are given 96 texture images of two types, grass and straw. They are all of size 128x128. Four exemplary images are shown in Figure 1. Tom helped label 36 grass images (called *grass_01, grass_02, ... , grass_36*) and Mary helped label 36 straw images (called *straw_01, straw_02, ... , straw_36*). There are still 24 unlabeled grass/straw images remaining (with file names *unknown_01, unknown_02, ... , unknown_24*). You are asked to implement a pattern recognition algorithm to classify these unlabeled images automatically.

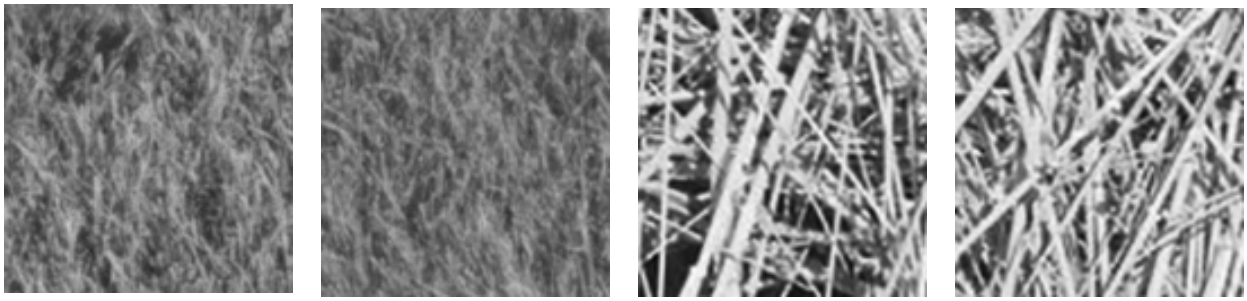


Figure 1: Samples of grass and straw texture images

You can accomplish the task by extracting Laws' features and classifying textures in the original or reduced feature spaces.

1) Feature Extraction
   a) Pre-processing: For each pixel of an input image, compute its local mean using the 5x5 window and subtract the local mean from its original intensity value. (Note: This pre-processing step helps remove the dc component that has no discriminant power for textures yet tend to mask the energy of the useful ac components.)
   b) Filter Bank Processing: Apply twenty-five 5x5 Laws' filters centered at each pixel of the input image to get 25 output images. Use the pixel energy of each output image to form a 25-D feature vector for the underlying pixel.
   c) Feature Averaging: Since each texture image consists of one texture type, you can compute the mean of all feature vectors to obtain a more robust feature vector to represent the whole input image.
2) Minimum Mean Distance Classification for the Unlabeled Data
   Implement a minimum-mean-distance classifier based on the Mahalanobis distance in the 25-D feature space for the 24 unlabeled texture images, where the class means of grass and straw are calculated from Tom's and Mary's labeled images, respectively. Calculate the error rate for both training and test data sets, which is defined as the number of misclassified samples divided by the total number of samples, usually expressed as the percentage.
3) Feature Dimension Reduction using PCA
   If you want to classify unlabeled 24 textured images into 2 classes, are the 25 features equally important? Justify your answer. Also, use the Principal Component Analysis (PCA) to reduce the feature dimension from 25 to 1.
   Note: Here, you can assume that the inter-class variance is larger than the intra-class variances.
4) Feature Dimension Reduction using LDA
   PCA does not take the class label into account. However, when you have labeled texture images from Tom and Mary, you can do a better job in selecting features that have a higher discriminant power. Again, you are asked reduce the feature dimension from 25 to 1. What is the feature? Justify your answer.
   Hint: Conduct the linear discriminant analysis (LDA) to the two labeled classes.
5) Classification with Two Reduced Feature Sets
   Apply the minimum mean distance classifier to classify *grass* and *straw* texture samples based on 1-D feature vectors selected in Parts (3) and (4), and calculate the error rate for both training and test data sets, respectively.
6) Performance Comparison and Discussion
   Compare your results obtained in Parts (2) and (5) and comment on the performance of three methods – i) no feature reduction, ii) feature reduction with no class label, iii) feature reduction with labeled training data. Which one is the best? Which one is the worst? Why?

For Parts (2)-(6), C/C++ users are allowed to use the Mat data structure and related functions in OpenCV to handle feature vector operations.

(b) Advanced Texture Classification: Multi-Classes + SVM (Advanced: 15%)

Four texture classes with 48 images each (*grass_01-48, straw_01-48, sand_01-48, and leather_01-48*, size 128x128) are provided and samples of these images are shown in Figure 2. You will conduct multi-class texture classification using the support vector machine (SVM) classifier in this part.

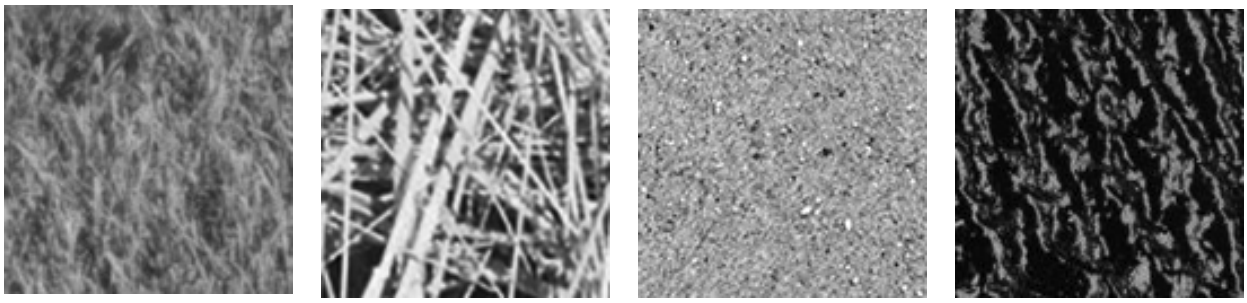


Figure 2: Samples of grass, straw, sand, and leather texture images

Implement the following four binary classifiers with two methods: the minimum-distance-to-class-means method and the Support Vector Machine (SVM) method:

- Grass VS Non-grass classifier
- Straw VS Non-straw classifier
- Sand VS Non-sand classifier
- Leather VS Non-leather classifier

Partition samples in each class randomly into a training set (36 samples) and a test set (12 samples). To train each classifier, use all 36 samples from the class of interest as positive ones and randomly select 12 training samples from each of the other three classes as negative ones. For example, to train the Grass VS Non-grass classifier, your training set should consist of 36 grass samples as positive ones and 36 non-grass samples (e.g., 12 straw samples, 12 sand samples and 12 leather samples) as negative ones.

Then, conduct the following four tasks:

1) Repeat the feature extraction and the feature dimension reduction steps as you did in Part (a). Instead of reducing to 1-D, you should reduce the dimension to 3-D. Show the results.
2) Classify texture images in the training and test data sets with the four minimum-distance-to-class-means classifiers and report the error rate for the two data sets separately.
3) Classify texture images in the training and testing data sets with the four SVM classifiers and report the error rate for the two data sets separately.

Hint: SVM is a widely used classification tool. You can find brief introductions in following links:
http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html;
https://www.csie.ntu.edu.tw/~cjlin/libsvm/.
Using the training data, SVM learns a parametric model, which describes the decision boundaries in corresponding feature space with support vectors and corresponding coefficients with the following steps.

- Step 1: Collect features of your training data and use them to form a design matrix X of dimension M x N, where M is the number of data samples (both positive and negative ones) and N is the feature dimension.
- Step 2: Collect labels (positive->1 and negative->-1) of your training data to form vector Lx of dimension M. (Here, we use x and y to denote the training and testing data, respectively.)

- Step 3: Set SVM training parameters. There are many important parameters in SVM implementations. Please read introductions of corresponding implementations carefully. In this task, you are asked to set the SVM type to "C-SVC", and the kernel type to "linear".
- Step 4: Run training functions with X, Lx and training parameters to obtain a trained SVM model.
- Step 5: Use the trained model to predict the training labels with the prediction function. Thus, you can get the error rates for the training data.
- Step 6: Build design matrix Y and labels Ly of testing data with the same steps. Then, you can predict the testing labels with the trained SVM model. You will obtain the predicted labels and classification accuracy accordingly.

4) Compare the results of in Parts 2) and 3). Which one is better? Why? Please give an intuitive explanation.

Note: You are allowed to use any SVM libraries in the public domain (e.g., libSVM or OpenCV).

Problem 2: Edge Detection (40 %)

(a) Sobel Edge Detector and Non Maximal Suppression (Basic: 10%)

Apply the Sobel filter to *Farm* and *Cougar* images in Figure 3. Note that you need to convert the RGB image to the grayscale image before applying the Sobel filter.

1) Normalize the gradient magnitude value to the range of 0-255 and show the gradient magnitude map for each image.
2) Choose two proper thresholds so that 10% and 15% of pixels are edge points. Show the image map for each image. Use the black and white colors to indicate the edge and the non-edge points, respectively.
3) Implement and apply the non-maximal-suppression technique to the two edge maps obtained in 2) and show their enhanced edge maps.

Farm **Cougar**

Figure 3: Farm and Cougar images

(b) Canny Edge Detector (Basic: 10%)

In this part, you are required to apply the Canny edge detector [1] to both *Farm* and *Cougar* images. 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 five edge maps using different low (A) and high (B) thresholds in the hysteresis thresholding step.

1) A: 0.3; B: 0.6 (Default values)
2) A: 0.2; B: 0.7 (Default A-Δ, Default B+Δ, with Δ=1)
3) A: 0.2; B: 0.5 (Default A-Δ, Default B-Δ)

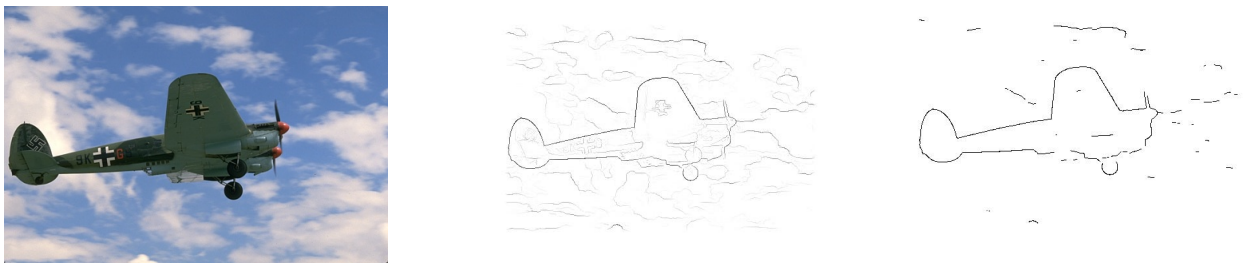
4) A: 0.4; B: 0.7 (Default A+Δ, Default B+Δ)
5) A: 0.4; B: 0.5 (Default A+Δ, Default B-Δ)

Compare the resulting edge maps obtained by the above five cases. Explain the relationship between the threshold values and image content (e.g., texture, smooth, weak object boundary regions).

(c) Structured Edge (Advanced: 10%)

In this problem, you are required to apply the Structured Edge (SE) detector [2] to extract edge segments from a color image with online source codes. Exemplary edge maps generated by the SE method for the *Airplane* image are shown in Figure 4. 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 *Farm* and *Cougar* images. Please state the chosen parameters clearly and justify your selection. Compare and comment on the visual results of the Sobel detector, the Canny detector and the SE detector.

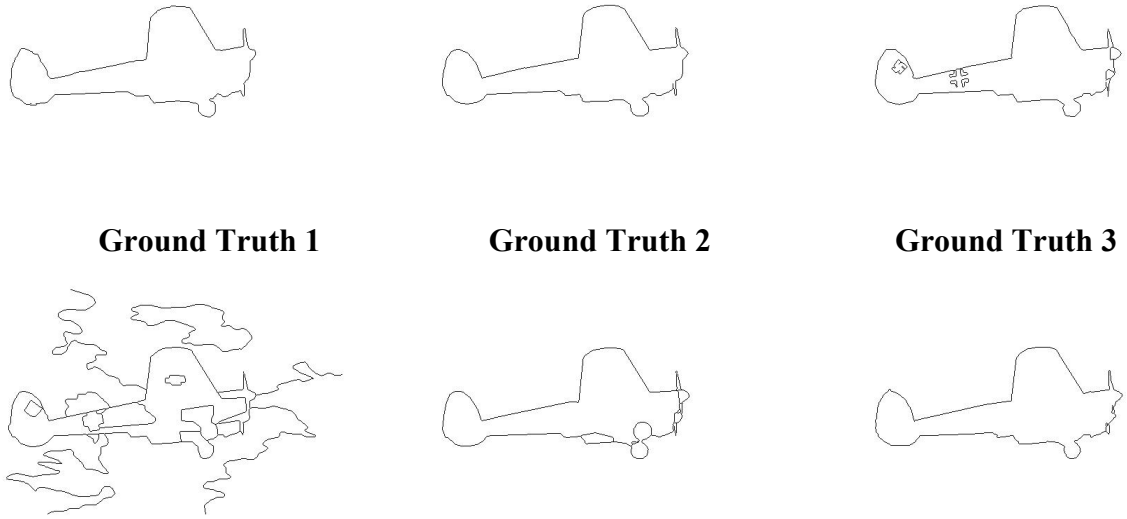


Airplane **Probability edge map** **Binary edge map (with p>0.38)**

Figure 4: The Airplane image and its probability and binary edge maps obtained by the SE detector

(d) Performance Evaluation (Advanced: 10%)

In this part, you are required to 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 5 shows 6 ground truth edge maps for the *Airplane* image from the Berkeley Segmentation Dataset 500 (BSD500) [3].



Ground Truth 1 **Ground Truth 2** **Ground Truth 3**

Ground Truth 4 **Ground Truth 5** **Ground Truth 6**

Figure 5: Six ground truth edge maps for the Airplane 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 6)
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 6)
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 6)

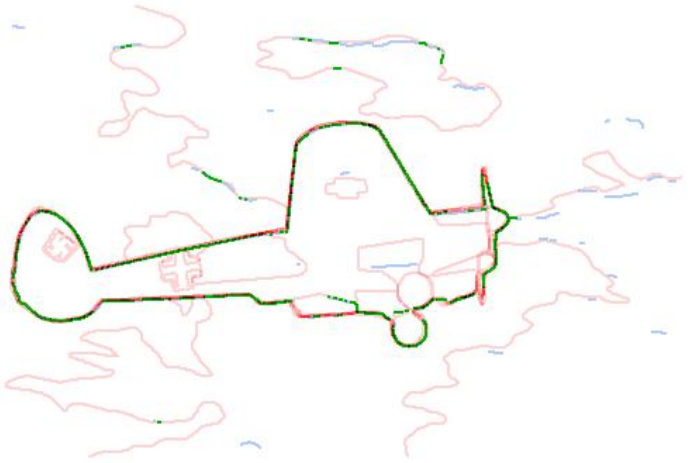


Figure 6: The error map of the SE edge map in Figure 4

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.

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

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 six ground truth edge maps of *Farm* and *Cougar* 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 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 - Farm or Cougar? 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: Image Segmentation (40 %)

Image segmentation partitions an image into multiple segments so that it can be easily analyzed in high level vision tasks such as image retrieval and objection detection. It is a process of labeling each pixel with a segment ID. In this problem, you are asked to conduct image segmentation using two methods: 1) Mean-Shift (MS) [4] and 2) Contour-guided Color Palettes (CCP) [5]. The CCP is one of the state-of-the-art segmentation technique developed at USC in 2015. You will be provided online source codes of both MS and CCP.

(a) MS+Superpixel Segmentation (Basic: 15 %)

MS is one popular image segmentation technique proposed by Comaniciu and Meer in 2002 [4]. It is often integrated with the superpixel representation. In this part, you will perform image segmentation on both Man and Rhinos images with the MS+Superpixel algorithm. You are allowed to use online source codes for superpixel representation and MS segmentation.

1) Plot the flow chart of the MS algorithm and describe the chart using you own words. What are key parameters needed in implementing the MS algorithm?
2) Try different values for the key parameters and discuss their effect on the segmentation results.

Man

Rhinos

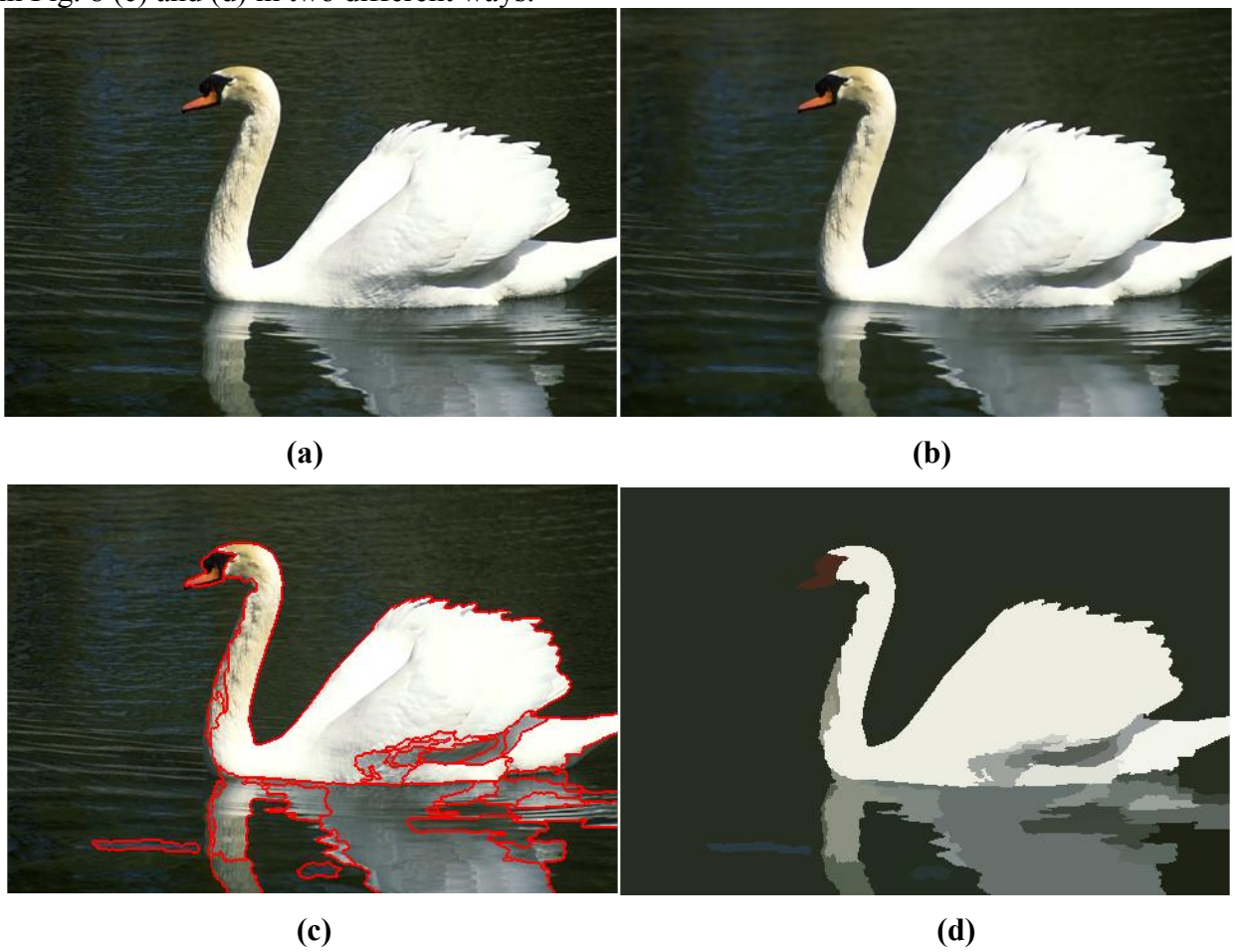
Figure 7: The Man and the Rhinos images

(b) Color Palettes Generation (Basic: 15 %)

CCP [5] takes the color information from sampled pixels to yield an image-dependent color palette and quantize the input image with the color palette. This is done by the following steps.

1) CCP applies the bilateral denoising filter in the LAB color space. An exemplary *Swan* image is shown in Fig. 8. Show the denoised *Man* and *Rhino* images using the CCP codes. Compare the original and denoised images and discuss the purpose of the denoising step.
2) Apply the SE algorithm to obtain contours and select long contours. Show the results of the *Man* and *Rhino* image using the CCP codes. Discuss why edge extension is required.
3) Select color samples from pixels along two sides of long contours. Comment on the purpose of this step. Why not use the uniform color sampling but contour-guided color sample?
4) Generate the color palette by clustering sampled colors. This is achieved by MS clustering with different spectral radius. Discuss the effect of the spectral radius parameter. Represent the Man and *Rhino* images using their corresponding color palettes. What happens if you change the MS algorithm to the K-means color clustering algorithm?

5) Improve the final segmentation result with post-processing. Explain the necessity of the three post-processing steps (namely, leakage avoidance by contours, fake boundary removal, and small region mergence) using examples from the *Man* and *Rhino* images. The final segmentation result is shown in Fig. 8 (c) and (d) in two different ways.



(a) **(b)**

(c) **(d)**

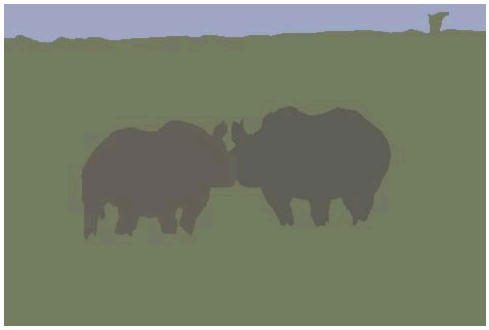
Figure 8: (a) The original Swan image, (b) the denoised Swan image, (c) the overlay of segmented regions on the input image and (d) the segmented Swan image represented with quantized colors

(c) Segmentation Result Evaluation (Advanced: 10%)

CCP outperforms other state-of-the-art algorithms in several metrics commonly used in segmentation method evaluation. They are: Cov, PRI VoI, GCE and BDE. Discuss the purposes of these measures. Explain what these metrics attempt to measure. Report the metrics for the *Man* and *Rhino* images with their ground truths given in Fig. 9.

Note: DO NOT quote statements directly from any listed references or online sources. You need to write your report in your own words. Written reports and source codes are subject to verification for any plagiarism.

(a)



(b)

Figure 9: The segmentation ground truths for (a) the Man image and (b) the Rhino image.

Appendix:

Problem 1: Texture Analysis and Segmentation

Part a/grass_01-36.raw	128x128	8-bit	Gray scale
Part a/straw_01-36.raw	128x128	8-bit	Gray scale
Part a/unknown_01-24.raw	128x128	8-bit	Gray scale
Part b/grass_01-48.raw	128x128	8-bit	Gray scale
Part b/straw_01-48.raw	128x128	8-bit	Gray scale
Part b/sand_01-48.raw	128x128	8-bit	Gray scale
Part b/leather_01-48.raw	128x128	8-bit	Gray scale

Problem 2: Edge Detection

Farm.raw	481x321	24-bit	Color (RGB)
Cougar.raw	481x321	24-bit	Color (RGB)
Airplane.raw	481x321	24-bit	Color (RGB)
Airplane_SE_prob_map.raw	481x321	8-bit	Gray scale
Airplane_SE_map.raw	481x321	8-bit	Gray scale
Airplane_GT(1~6).raw	481x321	8-bit	Gray scale
Farm_GT(1~5).raw	481x321	8-bit	Gray scale
Cougar_GT(1~5).raw	481x321	8-bit	Gray scale

Problem 3: Image Segmentation

Man.raw	321x481	24-bit	Color (RGB)
Rhinos.raw	481x321	24-bit	Color (RGB)
Swan.raw	481x321	24-bit	Color (RGB)
Swan_denoised.raw	481x321	24-bit	Color (RGB)
Swan_bound_segments.raw	481x321	24-bit	Color (RGB)
Swan_color_segments.raw	481x321	24-bit	Color (RGB)
Man_bound_GT.raw	321x481	24-bit	Color (RGB)
Man_color_GT.raw	321x481	24-bit	Color (RGB)
Rhinos_bound_GT.raw	481x321	24-bit	Color (RGB)
Rhinos_color_GT.raw	481x321	24-bit	Color (RGB)

Reference Images

Images in this homework are taken from Google images [6], the USC-SIPI image database [7] or the BSD500 [3].

References

[1] Canny, John. "A computational approach to edge detection." Pattern Analysis and Machine Intelligence, IEEE Transactions on 6 (1986): 679-698.
[2] Dollár, Piotr, and C. Lawrence Zitnick. "Structured forests for fast edge detection." Computer Vision (ICCV), 2013 IEEE International Conference on. IEEE, 2013.
[3] Arbelaez, Pablo, et al. "Contour detection and hierarchical image segmentation." Pattern Analysis and Machine Intelligence, IEEE Transactions on 33.5 (2011): 898-916.
[4] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 5, pp. 603–619, 2002.
[5] Xiang Fu, Chien-Yi Wang, Chen Chen, Changhu Wang and C.-C. Jay Kuo, "Robust image segmentation using contour-guided color palettes," IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, December 13-16, 2015.
[6] [Online] http://images.google.com/
[7] [Online] http://sipi.usc.edu/database/