EE 569 HOMEWORK 2

Issued: 9/18/2015 Due: 10/11/2015

Name: Meiyi Yang

Email: meiyiyan@usc.edu

USC ID: 6761-0405-85

Problem 1: Texture Analysis and Classification

1.1 Motivation

Image texture is a set metrics to classify the different class image. In this problem, we will use Laws' filter to extract the feature of the image, and use Minimum Mean Distance Classifier and SVM (Support Vector Machine) Classifier to determine the image class. To reduce the dimension of feature and improve the performance, PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis) are used to the extracted feature vector.

1.2 Approach and procedure

1.2.1 Two Classes + Minimum Mean Distance Classifier

Feature extraction. Intuitively, using the whole image pixels as the information of feature can give the classifier the best data to determine the class. However, it will spend to much time, memory. [1] Here we use feature vector to represent the feature of one class. A n-dimension feature vector is a point in n-dimension space. And the classification is to find how to classify this n-dimension space. In this problem, we use Laws' Filter to extract the feature of each image.

The implementation of feature extraction is shown in Figure 1.

1) Subtract local mean from original image

2) Apply 2-D Laws' filter to subtracted image. The 1-D Law's Filter and 2-D 5 * 5 Laws' Filter are shown below in Table 1 and Table 2.

3) Get the average of energy of each pixel

4) Normalize the feature vector of the training set. This normalization applies for each feature separately. When extract the testing set, we use the training set result calculated before.

$$X' = \frac{X - min}{max - min}$$

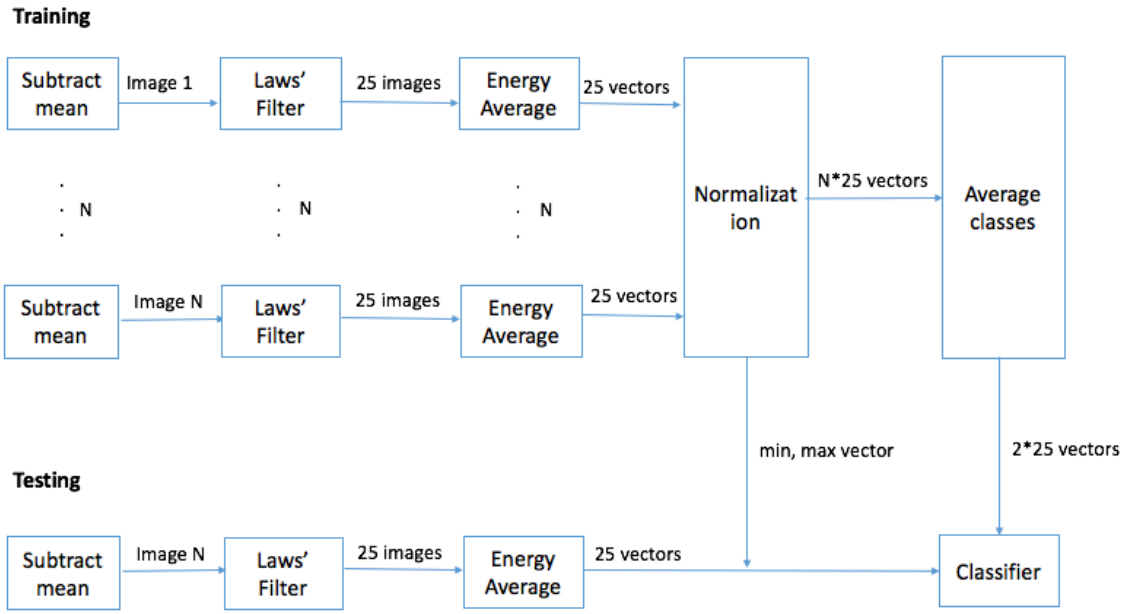

Figure 1. Flow chart of feature extraction

1 D Laws' Filter	Filter coefficients
L5	[1, 4, 6, 4, 1]
E5	[-1, -2, 0, 2, 1]
S5	[-1, 0, 2, 0, -1]
W5	[-1, 2, 0, -2, 1]
R5	[1, -4, 6, -4, 1]

Table 1. 1-D Laws' Filter

2-D Filter	0 (L5)	1 (E5)	2 (S5)	3 (W5)	4 (R5)
0 (L5)	$L5 * L5^T$	$L5 * E5^T$	$L5 * S5^T$	$L5 * W5^T$	$L5 * R5^T$
1 (E5)	$E5 * L5^T$	$E5 * E5^T$	$E5 * S5^T$	$E5 * W5^T$	$E5 * R5^T$
2 (S5)	$S5 * L5^T$	$S5 * E5^T$	$S5 * S5^T$	$S5 * W5^T$	$S5 * R5^T$
3 (W5)	$W5 * L5^T$	$W5 * E5^T$	$W5 * S5^T$	$W5 * W5^T$	$W5 * R5^T$

Table 2. 2-D Laws' Filter

Minimum mean distance classifier. To classify the class of each image, we can calculate the minimum distance between the feature vector of testing image and the feature vector of the mean of training image. The algorithm to get the minimum distance here is using Mahalanobis distance. [2]

$$d(x,y) = \sqrt{(x - y)^T S^{-1}(x - y)}$$

PCA (Principal Component Analysis). PCA is really a early algorithm which was invented by Karl Pearson in 1901[3]. The function of PCA here is to project the dimension of feature vector to lower dimension space using linear transformation [4]. Our goal is to find the eigenvectors in the n-dimension space and project other vectors to the space these eigenvectors formed, so that we preserve most of the information of the n-dimension feature vector. The implementation of PCA here is using OpenCV library cv::PCA. Flow of PCA is shown in Figure 2.

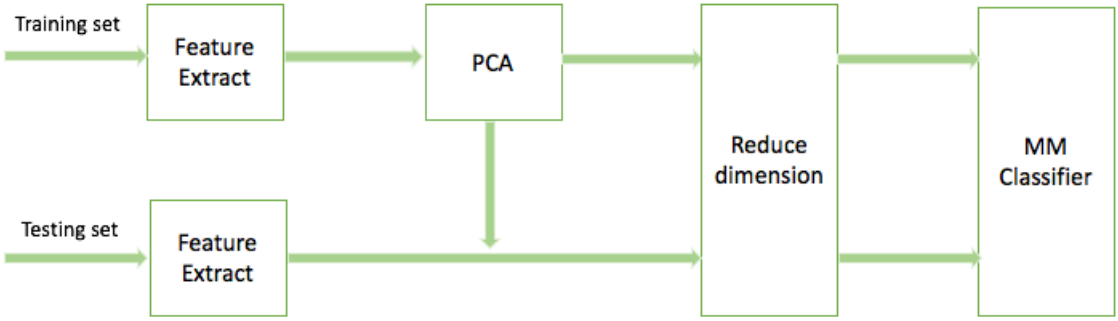

Figure 2. Flow chart of PCA

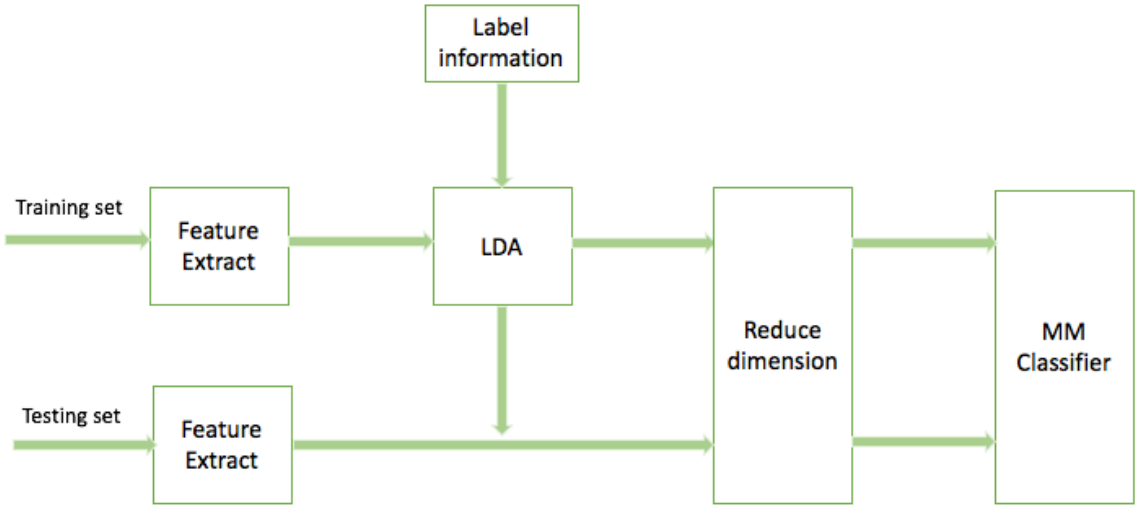

Figure 3. Flow chart of LDA

LDA (Linear Discriminant Analysis) is developed from Fisher's Linear Discriminant. This method is used to find the features that separated two or more classes points [5]. The implementation of LDA here is using OpenCV library cv::LDA. Flow of LDA is shown in Figure 3.

1.2.2 Multi-Classes + SVM

SVM (Support Vector Machine). SVM is a supervised learning models in machine learning [6]. The goal of SVM is not only finding a separating hyper-plane but also maximize the margin around the hyper-plane [7]. In the problem, we use OpenCV library to implement the SVM. The parameter and implementation are shown in Figure 4, 5.

```
// SVM train
Ptr<SVM> svm = SVM::create();
svm->setType(SVM::C_SVC);
svm->setKernel(SVM::LINEAR);
svm->setTermCriteria(TermCriteria(TermCriteria::MAX_ITER, 100, 1e-6));
svm->train(mat_train, ROW_SAMPLE, labelsMat);
```

Figure 4. Parameter used in SVM classifier

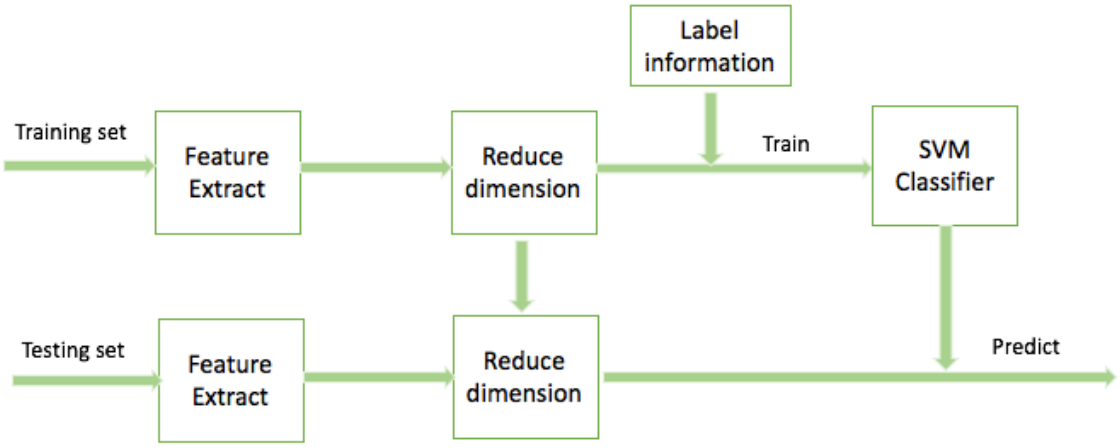

Figure 5. Flow chart of SVM classifier

1.3 Results

1.3.1 Two Classes + Minimum Mean Distance Classifier

1) Feature extraction: Here we use the training set of *grass 1-36* and *straw 1-36*. By subtracting the local mean of the image, we can get an extracted image. Figure 6 shows one the subtracted image. After apply Laws' Filter, we can get the mean of 72 * 25 feature vector shown in Figure 7.

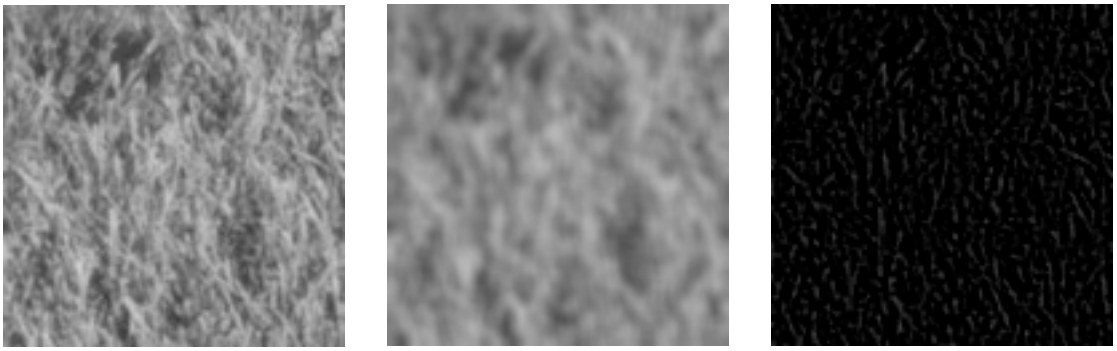

Figure 6. Left: original image. Middle: mean of original image. Right: subtracted image.

```
[0.03848205192194842;
 0.04220159784635971;
 0.05365060428134978;
 0.08098380690692716;
 0.1366680918790449;
 0.02657939646538538;
 0.04559830283862562;
 0.05511427522755147;
 0.07505553212163041;
 0.1326299996458987;
 0.04536888506303546;
 0.04084907050128248;
 0.05229326011501036;
 0.07233495032317311;
 0.1374549630130937;
 0.05452657015456856;
 0.04262895297462142;
 0.05909542746763417;
 0.08035559758950366;
 0.1813171788797291;
 0.07803154076795031;
 0.07561096858563127;
 0.08912915247593965;
 0.146675032064268;
 0.2691393276794511]
```

```
[0.8353507428583575;
 0.7909408823865719;
 0.7414769124228408;
 0.7007736196290062;
 0.6793457131766513;
 0.564962158815996;
 0.6795425706824549;
 0.6980653026402233;
 0.7054668690587753;
 0.6890481852753415;
 0.6055963437208294;
 0.5519789176907326;
 0.5832473419619072;
 0.5933804535224201;
 0.6046466253885836;
 0.6254063834685745;
 0.4906626488599173;
 0.5549258814403137;
 0.5317057574221178;
 0.5241357715492889;
 0.5062951504993722;
 0.5651469391143281;
 0.615143106610675;
 0.5790072091107065;
 0.5233747768753533]
```

Figure 7: Average of feature for two classes

2) Minimum mean distance classification for the unlabeled data. Apply MM classifier to the extracted image. The distance of each image and the error rate is shown in Fig8. Training set: *grass 1-36*, *straw 1-36*. Testing set: *unknown 1-24*.

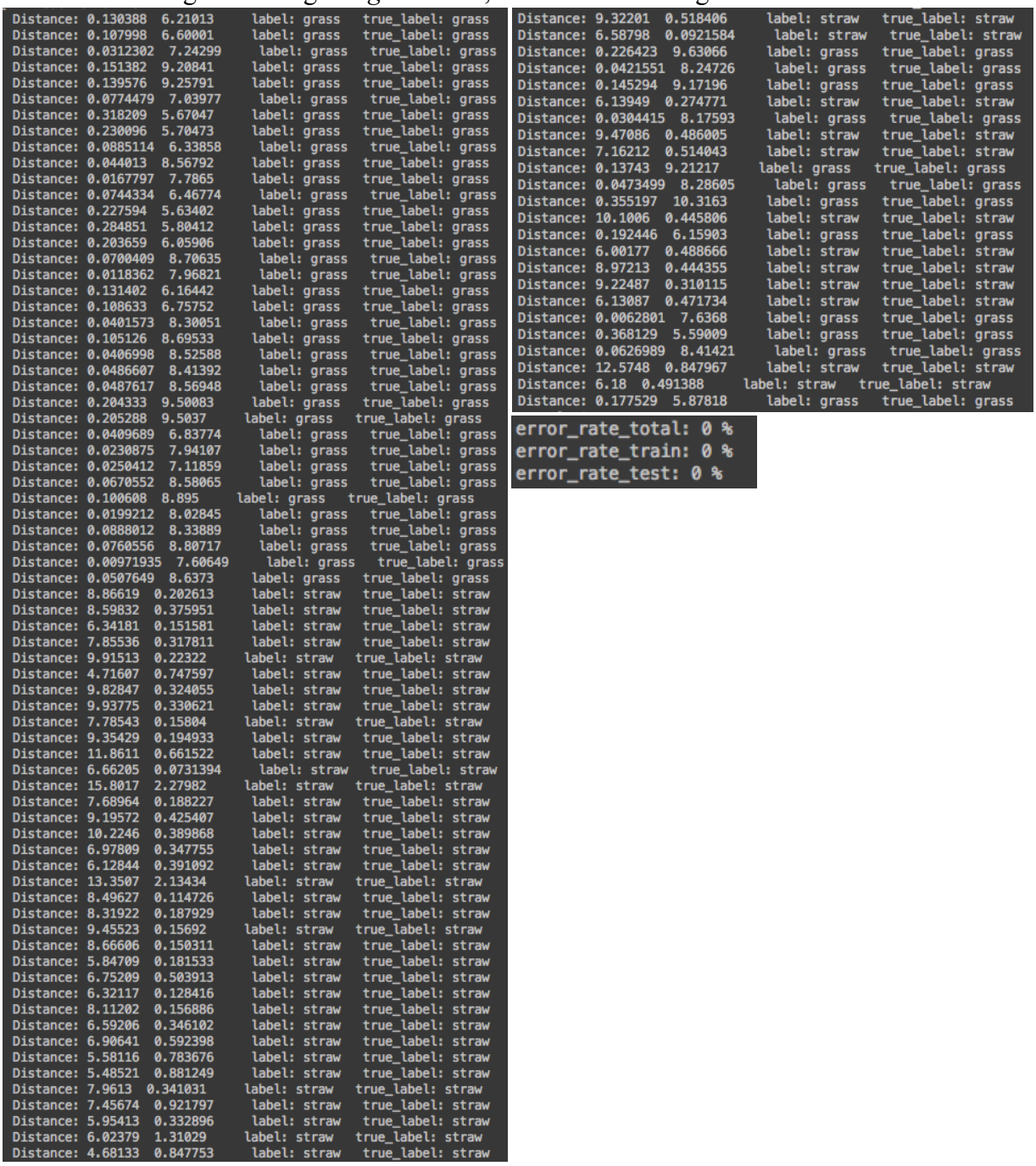

Figure 8. MM classifier, no feature reduction. Left: training data for *grass 1-36* and *straw 1-36*. Top-right: testing data for *unknown 1-24*. Button-right: Error rate of Minimum-mean distance classifier

3) Feature dimension reduction using PCA. Different features have different importance (1.4.1). Fig 9 is the result of PCA dimension reduction from 25 to 1.

```
----------mat_feature_average_pca----------
[-1.365780679872124]
[1.306259104714434]
```

Figure 9. PCA, dimension number = 1.

4) Feature dimension reduction using LDA (Figure 10)

```
----------mat_feature_average_lda----------
[-0.007088085703369104]
[-0.1276033155356752]
```

Figure 10. LDA, dimension number = 1

5) Classification with two reduced feature sets. PCA in Fig11, LDA in Fig12.

```
_MM  Mode: 1 num: 1
_Dimension_PCA
  Distance: 0.0700299  5.79561     label: grass   true_label: grass
  Distance: 0.0363682  6.15702     label: grass   true_label: grass
  Distance: 0.00311724  6.84454     label: grass   true_label: grass
  Distance: 0.0836169  8.76874     label: grass   true_label: grass
  Distance: 0.0894645  8.82771     label: grass   true_label: grass
  Distance: 0.0102907  6.60797     label: grass   true_label: grass
  Distance: 0.16254  5.1478     label: grass   true_label: grass
  Distance: 0.141264  5.27248     label: grass   true_label: grass
  Distance: 0.0541518  5.95035     label: grass   true_label: grass
  Distance: 0.0347913  8.17139     label: grass   true_label: grass
  Distance: 0.00227116  7.39675     label: grass   true_label: grass
  Distance: 0.0430924  6.07353     label: grass   true_label: grass
  Distance: 0.150515  5.21701     label: grass   true_label: grass
  Distance: 0.139626  5.28252     label: grass   true_label: grass
  Distance: 0.0960755  5.57942     label: grass   true_label: grass
  Distance: 0.0436893  8.3005     label: grass   true_label: grass
  Distance: 0.00664452  7.58206     label: grass   true_label: grass
  Distance: 0.0757138  5.74503     label: grass   true_label: grass
  Distance: 0.0272271  6.28522     label: grass   true_label: grass
  Distance: 0.0196734  7.90904     label: grass   true_label: grass
  Distance: 0.0408684  8.26102     label: grass   true_label: grass
  Distance: 0.0314248  8.11857     label: grass   true_label: grass
  Distance: 0.0253381  8.0158     label: grass   true_label: grass
  Distance: 0.0347524  8.17079     label: grass   true_label: grass
  Distance: 0.11028  9.02476     label: grass   true_label: grass
  Distance: 0.110398  9.02583     label: grass   true_label: grass
  Distance: 0.0170951  6.45816     label: grass   true_label: grass
  Distance: 0.00576237  7.55123     label: grass   true_label: grass
  Distance: 0.00618675  6.72564     label: grass   true_label: grass
  Distance: 0.0351445  8.17679     label: grass   true_label: grass
  Distance: 0.0571656  8.4747     label: grass   true_label: grass
  Distance: 0.00819563  7.63179     label: grass   true_label: grass
  Distance: 0.0197194  7.90996     label: grass   true_label: grass
  Distance: 0.0515724  8.40499     label: grass   true_label: grass
  Distance: 0.00025463  7.22533     label: grass   true_label: grass
  Distance: 0.0394444  8.24061     label: grass   true_label: grass
  Distance: 8.6701  0.0742362     label: straw   true_label: straw
  Distance: 8.29235  0.0430996     label: straw   true_label: straw
  Distance: 6.21222  0.0322588     label: straw   true_label: straw
  Distance: 7.47044  0.00374194     label: straw   true_label: straw
  Distance: 9.79674  0.209704     label: straw   true_label: straw
  Distance: 4.28525  0.36235     label: straw   true_label: straw
  Distance: 9.60817  0.182897     label: straw   true_label: straw
  Distance: 9.72543  0.199382     label: straw   true_label: straw
  Distance: 7.64735  0.00871335     label: straw   true_label: straw
  Distance: 9.23469  0.13456     label: straw   true_label: straw
  Distance: 11.5779  0.533774     label: straw   true_label: straw
  Distance: 6.55487  0.0124975     label: straw   true_label: straw
  Distance: 14.8952  1.40989     label: straw   true_label: straw
  Distance: 7.41128  0.0025327     label: straw   true_label: straw
  Distance: 8.84541  0.0912543     label: straw   true_label: straw
  Distance: 10.0425  0.246953     label: straw   true_label: straw
  Distance: 6.53486  0.0133872     label: straw   true_label: straw
  Distance: 5.77949  0.0718143     label: straw   true_label: straw
  Distance: 11.7296  0.566725     label: straw   true_label: straw
  Distance: 8.32219  0.0452754     label: straw   true_label: straw
  Distance: 8.15939  0.0340123     label: straw   true_label: straw
  Distance: 9.3672  0.150969     label: straw   true_label: straw
  Distance: 8.50976  0.060079     label: straw   true_label: straw
  Distance: 5.72542  0.0779836     label: straw   true_label: straw
  Distance: 6.2778  0.0277175     label: straw   true_label: straw
  Distance: 6.18679  0.0341193     label: straw   true_label: straw
  Distance: 7.88371  0.0184291     label: straw   true_label: straw
  Distance: 6.22832  0.0311097     label: straw   true_label: straw
  Distance: 6.25285  0.0294022     label: straw   true_label: straw
  Distance: 4.91314  0.207461     label: straw   true_label: straw
  Distance: 4.85826  0.218925     label: straw   true_label: straw
  Distance: 7.5676  0.00622325     label: straw   true_label: straw
  Distance: 6.50626  0.0147143     label: straw   true_label: straw
  Distance: 5.64968  0.0871046     label: straw   true_label: straw
  Distance: 4.82058  0.227012     label: straw   true_label: straw
  Distance: 4.21019  0.384605     label: straw   true_label: straw
```

```
Distance: 8.79315  0.0860161     label: straw   true_label: straw
Distance: 6.45537  0.0172392     label: straw   true_label: straw
Distance: 0.124907  9.15342     label: grass   true_label: grass
Distance: 0.0167862  7.84897     label: grass   true_label: grass
Distance: 0.0798061  8.7293     label: grass   true_label: grass
Distance: 5.85679  0.0634832     label: straw   true_label: straw
Distance: 0.0141468  7.78957     label: grass   true_label: grass
Distance: 8.94935  0.102085     label: straw   true_label: straw
Distance: 6.57241  0.0117436     label: straw   true_label: straw
Distance: 0.0844064  8.77681     label: grass   true_label: grass
Distance: 0.0189124  7.89364     label: grass   true_label: grass
Distance: 0.207831  9.78391     label: grass   true_label: grass
Distance: 9.8126  0.212029     label: straw   true_label: straw
Distance: 0.0826398  5.68617     label: grass   true_label: grass
Distance: 5.54772  0.100286     label: straw   true_label: straw
Distance: 8.45257  0.0553616     label: straw   true_label: straw
Distance: 8.93474  0.100529     label: straw   true_label: straw
Distance: 5.67668  0.0837884     label: straw   true_label: straw
Distance: 0.000555044  7.26625     label: grass   true_label: grass
Distance: 0.183395  5.03461     label: grass   true_label: grass
Distance: 0.0250884  8.01135     label: grass   true_label: grass
Distance: 12.2893  0.694841     label: straw   true_label: straw
Distance: 5.69992  0.0809911     label: straw   true_label: straw
Distance: 0.112593  5.45919     label: grass   true_label: grass
```

```
error_rate_total: 0 %
error_rate_train: 0 %
error_rate_test: 0 %
```

Figure 11. MM classifier, PCA = 1. Left: training data for *grass 1-36* and *straw 1-36*. Top-right: testing data for *unknown 1-24.* Button-right: Error rate of Minimum-mean distance classifier

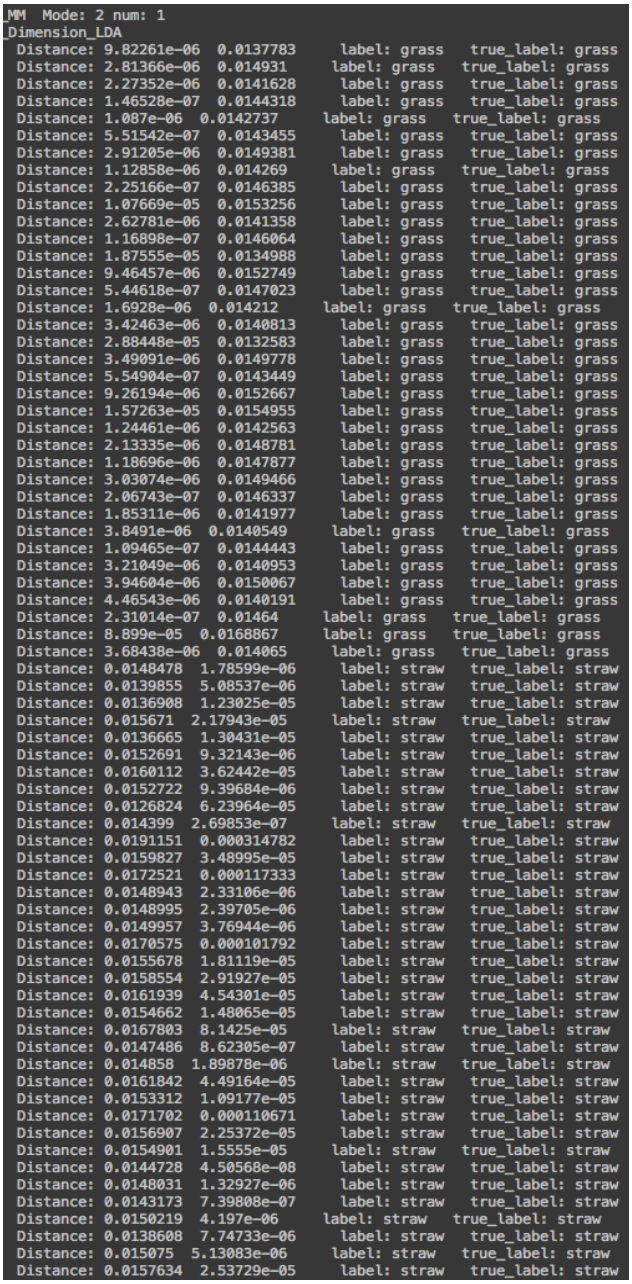

Figure 12. MM classifier, LDA = 1. Left: training data for *grass 1-36* and *straw 1-36*. Top-right: testing data for *unknown 1-24.* Button-right: Error rate of Minimum-mean distance classifier

6) Performance comparison and discussion. Please see discussion 1.4.1

1.3.2 Multi-Classes + SVM

1) Repeat the feature extraction and feature dimension reduction to 3-dimension. Figure 13 is the results. The training set and testing set of four cases are in Table 3.

2) Use four minimum-distance-to-class-means classifiers. The error rate is also shown in Figure 13. The training set and testing set of four cases are in Table 3.

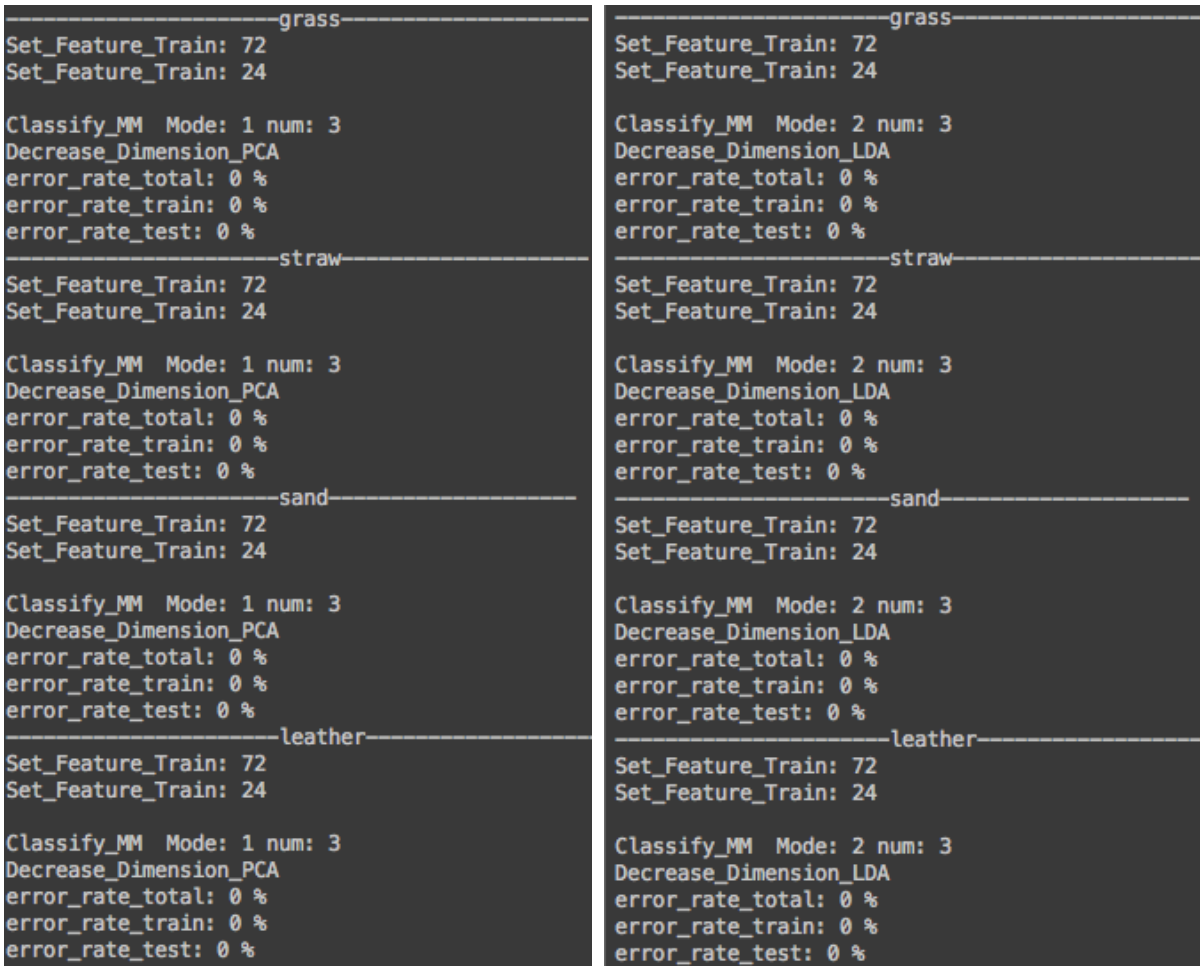

Figure 13. MM classifier. Left: PCA feature reduction, number from 1 to 3. Right: LDA feature reduction, number from 1 to 3.

Label	Training 72	Testing 24
Grass	Grass 36, Straw 12, Sand 12, Leather 12	Grass 12, Straw 4, Sand 4, Leather 4
Straw	Straw 36, Grass 12, Sand 12, Leather 12	Straw 12, Grass 4, Sand 4, Leather 4
Sand	Sand 36, Straw 12, Grass 12, Leather 12	Sand 12, Straw 4,Grass 4, Leather 4
Leather	Leather 36, Straw 12, Sand 12, Grass 12	Leather 12, Straw 4, Sand 4, Grass 4

Table 3. Training and testing data

```
----------grass----------
Set_Feature_Train: 72
Set_Feature_Train: 24

Classify_SVM  Mode: 1 num: 3
Decrease_Dimension_PCA
error_rate_total: 0 %
error_rate_train: 0 %
error_rate_test: 0 %
----------straw----------
Set_Feature_Train: 72
Set_Feature_Train: 24

Classify_SVM  Mode: 1 num: 3
Decrease_Dimension_PCA
error_rate_total: 0 %
error_rate_train: 0 %
error_rate_test: 0 %
----------sand----------
Set_Feature_Train: 72
Set_Feature_Train: 24

Classify_SVM  Mode: 1 num: 3
Decrease_Dimension_PCA
error_rate_total: 0 %
error_rate_train: 0 %
error_rate_test: 0 %
----------leather----------
Set_Feature_Train: 72
Set_Feature_Train: 24

Classify_SVM  Mode: 1 num: 3
Decrease_Dimension_PCA
error_rate_total: 0 %
error_rate_train: 0 %
error_rate_test: 0 %
```

Figure 14. SVM Classifier. PCA number = 3

1.4 Discussion

1.4.1 Two Classes + Minimum Mean Distance Classifier

Different feature vectors have different importance for the classification. What we want is to enhance the feature that have a large difference between two classes, and suppress the feature similar. Too many dimension of the feature vector is a disaster that increases the time and memory complexity of the computation. When we apply a dimension reduction to the high-dimension feature vector, in this case, from 25 to 1 or 2 or 3, we enhance the major feature and suppress the feature less useful, which compensates the information we lose during the dimension reduction.

Here is one test case that using 1, 2, 3 three-dimension reduction parameter to PCA, LDA. We can see there is no large difference between three dimension.

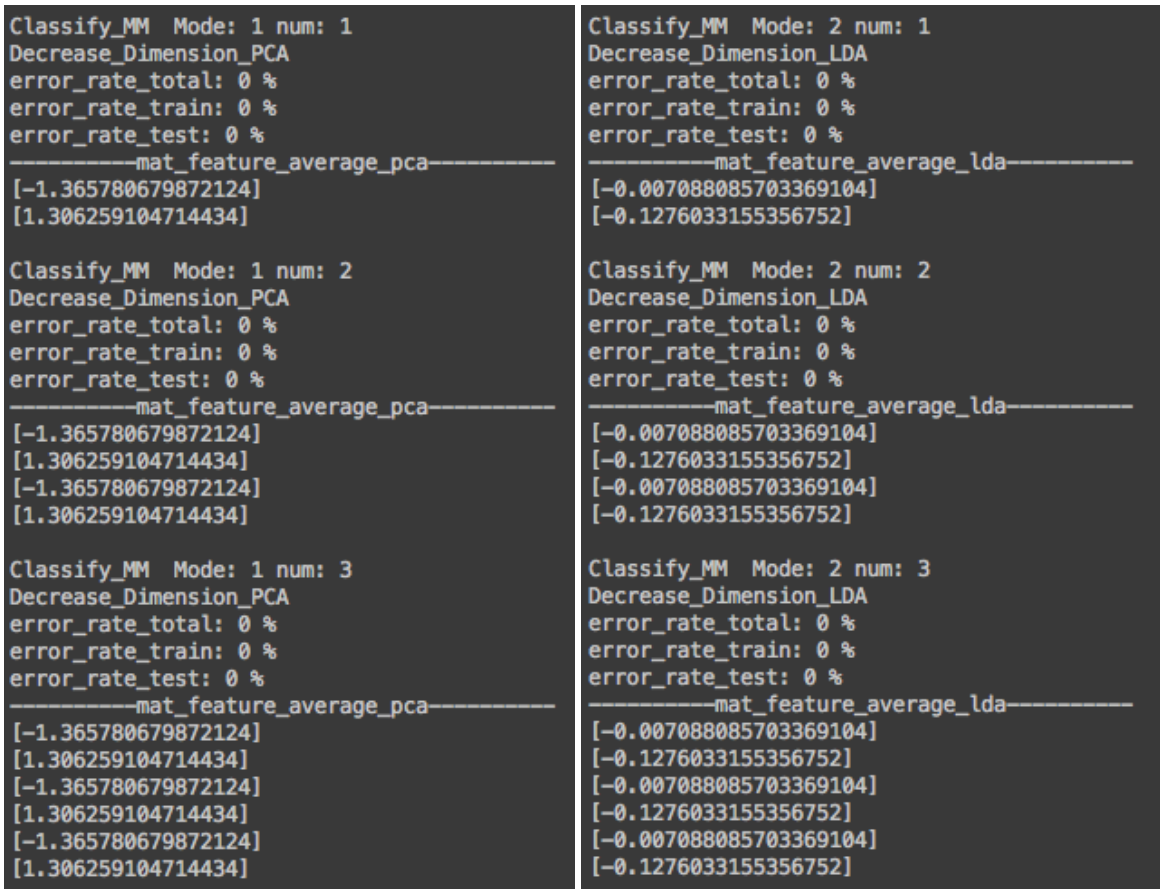

Figure 15 Special case for discussion. Classifier MM. Left: PCA, number 1, 2, 3. Right: LDA, number 1, 2, 3 (Mode 1 is PCA, Mode 2 is LDA)

Compared PCA and LDA dimension reduction algorithm, LDA is better than PCA in this case. By taking label information into account, LDA get a more accurate result. Though no feature reduction and and feature reduction get the same result, apparently feature reduction is better. Because PCA and LDA reduce the time and memory of computation. Also, Figure 16 is a result that I use classifier MM and PCA, LDA WITHOUT feature normalization. We can see in this case, LDA apparently performs better than PCA and no-feature reduction. No-feature reduction classification performance slightly better than classification with PCA. Also there is a more clearly result in Figure 19 and Figure 20. LDA in the four-class MM classifiers performs better than PCA.

```
Problem1a
Set_Feature_Train: 72
Set_Feature_Test: 24

Classify_MM  Mode: 0 num: 0
error_rate_total: 2.08333 %
error_rate_train: 2.77778 %
error_rate_test: 0 %

Classify_MM  Mode: 1 num: 1
Decrease_Dimension_PCA
error_rate_total: 3.125 %
error_rate_train: 2.77778 %
error_rate_test: 4.16667 %

Classify_MM  Mode: 1 num: 2
Decrease_Dimension_PCA
error_rate_total: 3.125 %
error_rate_train: 2.77778 %
error_rate_test: 4.16667 %

Classify_MM  Mode: 2 num: 1
Decrease_Dimension_LDA
error_rate_total: 0 %
error_rate_train: 0 %
error_rate_test: 0 %
```

Figure 16. Special case for discussion to compare PCA and LDA. (Mode 0 is no-feature reduction, Mode 1 is PCA, Mode 2 is LDA)

1.4.2 Multi-Classes + SVM

Because the error rates of SVM are all 0. I change PCA dimension number to 1 , 2, and 3 to see if there is any change (Figure 17). We can see when number of dimension in PCA decrease from 3 to 1, the performance of classification become

inaccurate. And the performance of each texture classification is texture-dependent. Leather class, in this case, is the hardest predicting class.

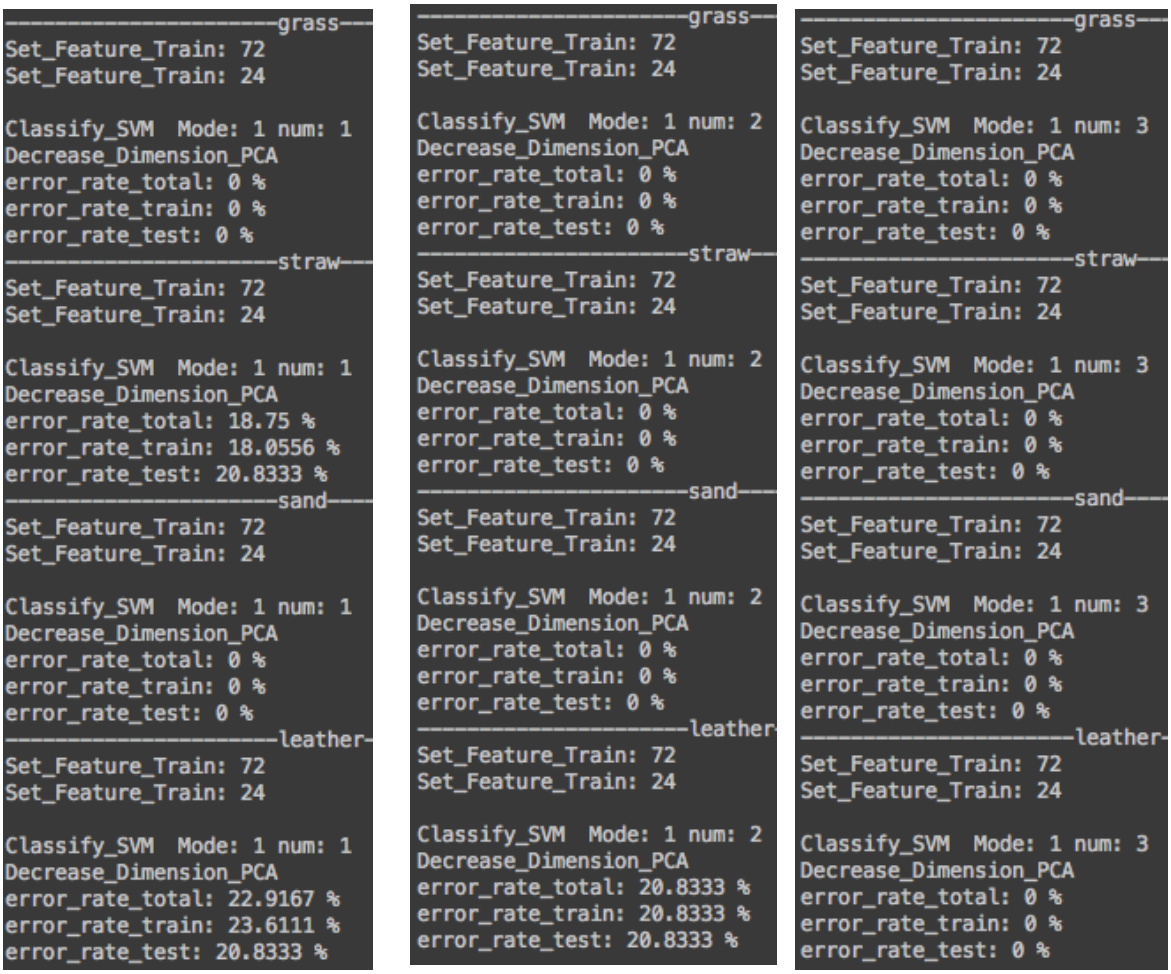

Figure 17. Classifier SVM, Dimension reduction PCA. Left: number of dimension is 1. Middle number of dimension is 2. Right: number of dimension is 3.

To compare with the Minimum-Mean distance classifier, I also set dimension reduction number to 1, 2, and 3 to run the four MM classifiers again and get the results in Figure 19, Figure 20. We will see that these texture classification performances a little better than SVM. The results are not as much texture-dependent as SVM results. Another interesting thing is, in MM classifier case, the error rates of training set and testing set (PCA number = 1) are not similar. In my opinion, if we want to get a "looks good" result in this case, we should use MM classifier. However, if we want to get a more reasonable result and have more training data, we can try to use SVM classifier.

Here is a deep explanation using an example in 2-D in Figure 18. Assume we only train "blue" points, to optimize the minimum-mean distance, MM classifier may get a line which has a "good" result but is not as useful as SVM classification. In other words, minimum-mean distance may lead to "over fit" result.

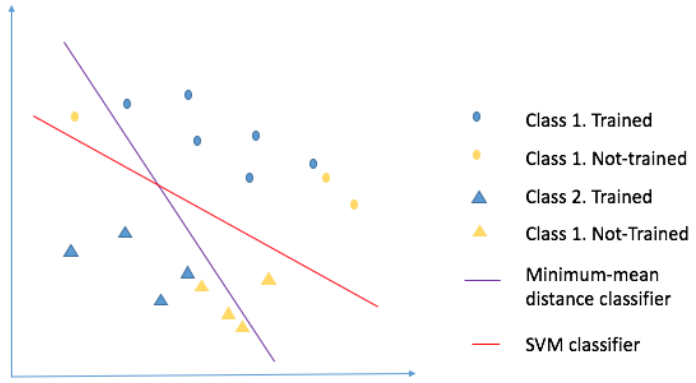

Figure 18. Explanation for Minimum-mean distance classifier and SVM classifier

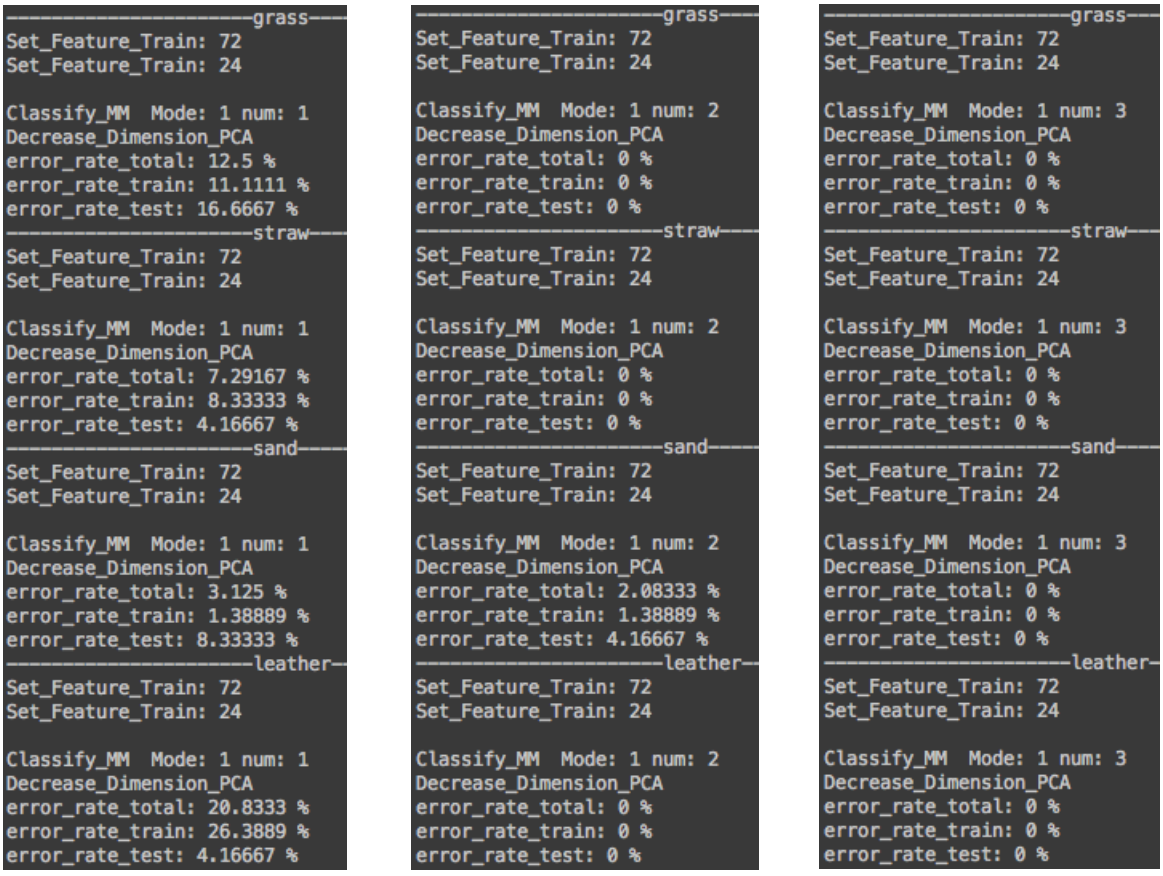

Figure 19. Classifier MM, Dimension reduction PCA. Left to right: numbers of dimension are 1, 2, 3.

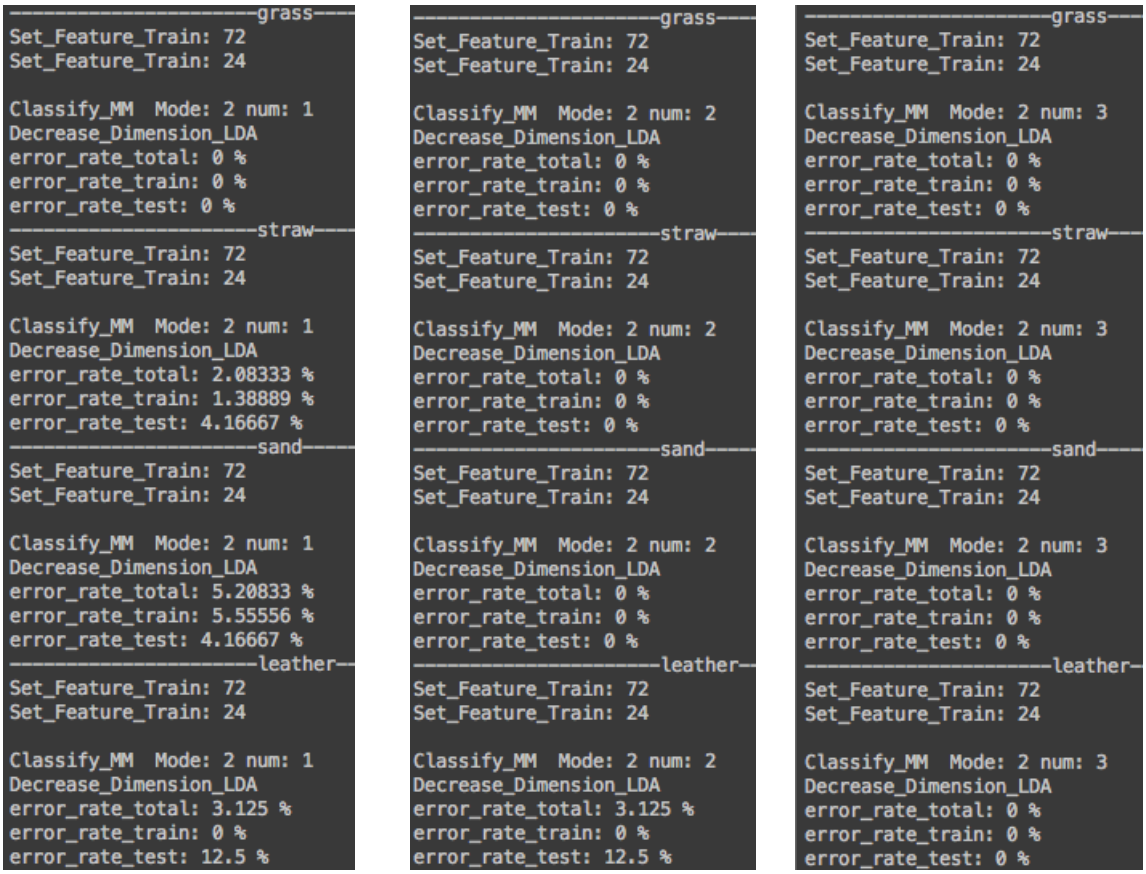

Figure 20. Classifier MM, Dimension reduction LDA. Left to right: numbers of dimension are 1, 2, 3.

Problem 2: Edge Detection

2.1 Motivation

The edge of image is one of the basic feature of the digital image. The edge points here mean that the intensity scale of certain field in the image has a sharp change or discontinuities [8]. Edge detection is one of the basic problem in the computer vision and image processing. The aim of edge detection is to find the edge points which has a large brightness changing in the digital image.

In this problem, I will implement three methods of edge detection using: Sobel edge detector and non-maximal suppression, Canny edge detector, and Structured edge detector. F-measure is used to measure the performance of these three methods.

2.2 Approach and procedure

2.2.1 Sobel edge detector and non-maximal suppression

The Sobel edge detector algorithm is mentioned by Sobel I. and Feldman G., in a talk at Stanford Artificial Intelligence Project (SAIL) in 1968 [9]. This is a discrete differentiation operator, which searches the edge by computing the gradient of each x direction and y direction.

Non-maximum suppression algorithm is always used in thin the edge in edge detection in image processing field. The idea of NMS is to suppress the intensity value of image to 0 except the local maximal point [10].

The implementation for above algorithm is following by these steps:

(1) Convert the color image to gray image.

$$Gray\ intensiy\ =\ R \times 0.21\ +\ G \times 0.72\ +\ B \times 0.07$$

(2) Apply Sobel Filter to gradient x, gradient y. Generate Sobel mask:

$$\frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad and \quad \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

A Sobel mask to x direction of the image, and get gradient x image, and apply Sobel mask to y direction of the image, and get gradient y image. Get magnitude image by

$$Magnitude\ =\ \sqrt{gradient\ x^2\ +\ gradient\ y^2}$$

(3) Normalize the magnitude matrix to 0 ~ 255.

$$X' = \frac{(X - min) \times 255}{max - min}$$

(4) Use threshold to binary the edge and non-edge. First sort the normalized matrix and get the intensity value by threshold. Then set black to edge, white to non-edge in the normalized image.

(5) Apply non-maximum suppression to get thin edge. Find the major direction by comparing gradient x and gradient y calculated before. Then set the edge point which is larger than both its neighbor in major direction.

2.2.2 Canny edge detector

Canny algorithm is first raised by Canny in 1986. He mentioned that there have three performance criteria to edge detection: 1) Good detection. The probability of error in the edge detection should be low. 2) Good localization. The edge detected should be as close as the true edge. 3) Only one response to a single edge. [11]

To find the best solution to match the above three criteria, the basic idea of Canny edge detector is first using Gaussian filter to smooth the image, and then use non-maximum suppression to get the final edge detection image. The details of Canny edge detector are following:

1) Use Gaussian filter to smooth the noise image.

2) Use first-derivative differentiation to get the magnitude and orientation gradient of the image.

3) Apply non-maximum suppression to the magnitude image.

4) Use double threshold algorithm to get rid of the false response.

5) Edge tracking by hysteresis to connect the edge [10].

I implement the Canny edge detector by OpenCV library. The Gaussian filter and gradient filters are [12]:

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \qquad G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

For performance evaluation, I convert all the result images to black edge and white non-edge images.

2.2.3 Structured edge

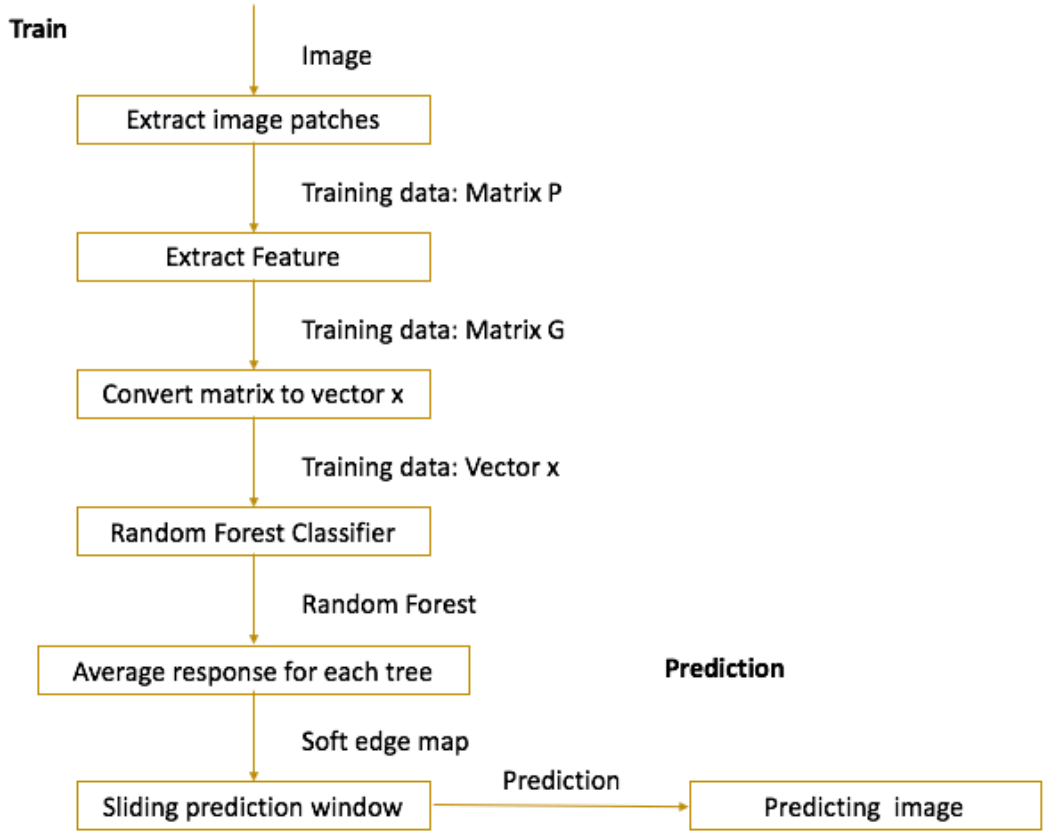

Figure 21. Flow chart of Structured Edge detector

Structured edge detector. Structured edge detection is first mentioned by Dollár, Piotr, and C. Lawrence Zitnick 2013. This is a algorithm developed from Structured Token Detection. This algorithm can be divided to two parts: Train and Predict.

Train: 1) Extract all image patches to get Matrix P; 2) Extract image feature from Matrix P to get Matrix G; 3) Convert each Matrix G to a vector x; 4) Use Random Forest Classifier to classify these vector x [13][15].

Predict: after apply RF classifier, we get a training tree result to predict the image. To predict the image, we need average response for each tree to get a soft edge map, and then sliding it to 16 * 16 prediction window [14].

Random Forest classifier. Compared to Structure Token using predefined labels, Structured Edge use edge structure directly, which preserves more information, but larger dimensions (2^{256}). To reduce the dimension, we should

apply the following strategies: 1) Reduced by space Y. Y is not random because patch is segmented by close contours. We can use Z to encode the information that measure the similarity of every pair patch in Y belong to the same segment and split them into two classes. Then the dimension reduced to 32640. 2) Reduced by PCA. Now we sample 256 dimensions of 32640 dimensions in Z and reduce the data to 5 dimensions using PCA. 3). K-mean to binary classify into two classes. The dimensions are 2 now [14].

The implementation of SE is provided by Structured Edge Detection Toolbox V3.0, Piotr Dollar [17].

2.2.4 Performance evaluation

Ground Truth. To evaluate the performance of the detection above (Sobel, Canny, SE), we need a "correct" answer for each image. Because we goal to develop an edge detection algorithm is making machine predict the edge like a human being. So for edge detection, we use the edge maps provided by human to evaluate the performance of edge detection. The data here we use is from the Berkeley Segmentation Dataset 500 (BSD500) [16].

F-measure	Edge map	Ground Truth
True Positive (TP)	+	+
True Negative (TN)	-	-
False Positive (FP)	+	-
False Negative (FN)	-	+

Table 4. F-measure coefficients

F-measure. We use F-measure here to evaluation the edge detection performance. First we can divide the edge map result to 4 classes: TP, TN, FP, FN, shown in Table 4. "+" means edge pixels are shown in the map, and "-" means edge pixels are not in the map. After counting the number of pixels in four classes, we calculate P and R and then get F-measure:

$$Precision: P = \frac{number\ of\ True\ Positive}{number\ of\ True\ Positive + number\ of\ False\ Positive}$$

$$Recall: R = \frac{number\ of\ True\ Positive}{number\ of\ True\ Positive + number\ of\ False\ Negetive}$$

$$F - measure = 2 \times \frac{P \times R}{P + R}$$

The implementation of F-measure is also provided by Structured Edge Detection Toolbox V3.0, Piotr Dollar [17].

2.3 Results

2.3.1 Sobel edge detector and non-maximal suppression

First convert RGB image of "*Farm*" and "*Cougar*" to gray image. The apply Sobel Filter to both images. The figure of the gradient image of two directions and magnitude map of "*Farm* " and "*Cougar*" images are shown below in Figure 22, 23.

Apply 10% and 15% threshold to "*Farm*" and "*Cougar*" magnitude images. For displaying, set black color to edge, and white color to non-edge. Then apply non-maximal-suppression to the above images. The results are shown in Figure 24, 25.

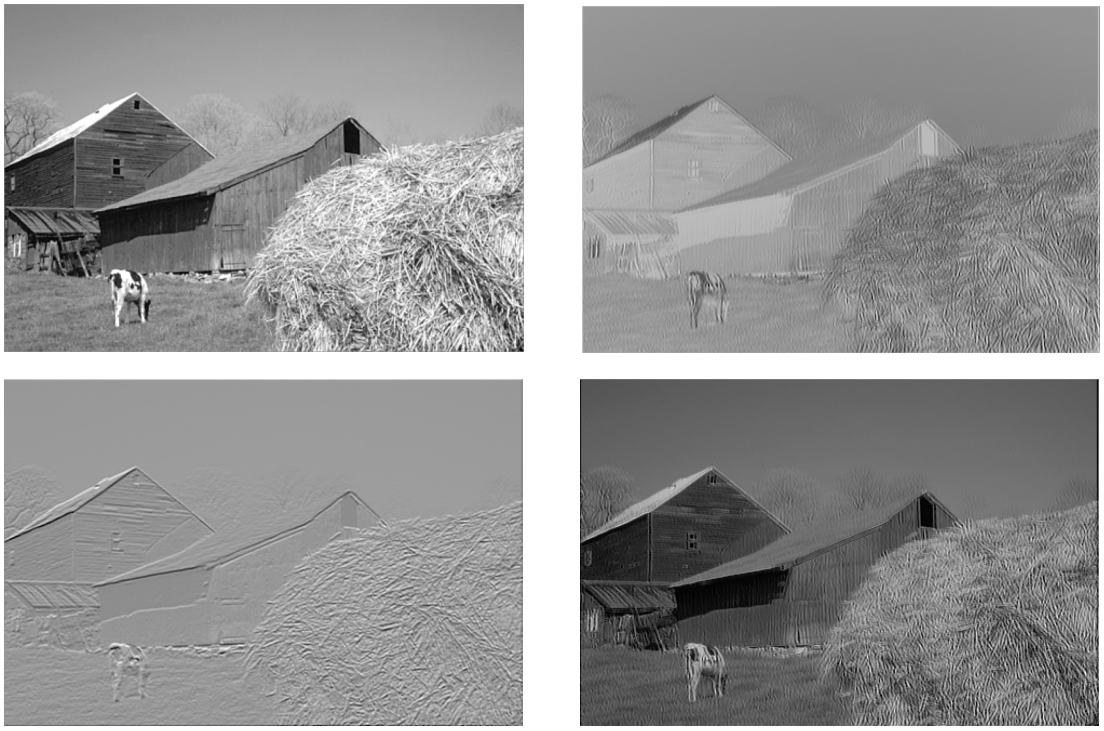

Figure 22. Top-left: The gray scale of *Farm*. Top-right: the gradient x image of *Farm*. Button-left: the gradient y image of *Farm*. Button-right: the magnitude image of *Farm*.

Figure 23. Top-left: The gray scale of *Cougar*. Top-right: the gradient x image of *Cougar*. Button-left: the gradient y image of *Cougar*. Button-right: the magnitude image of *Cougar*.

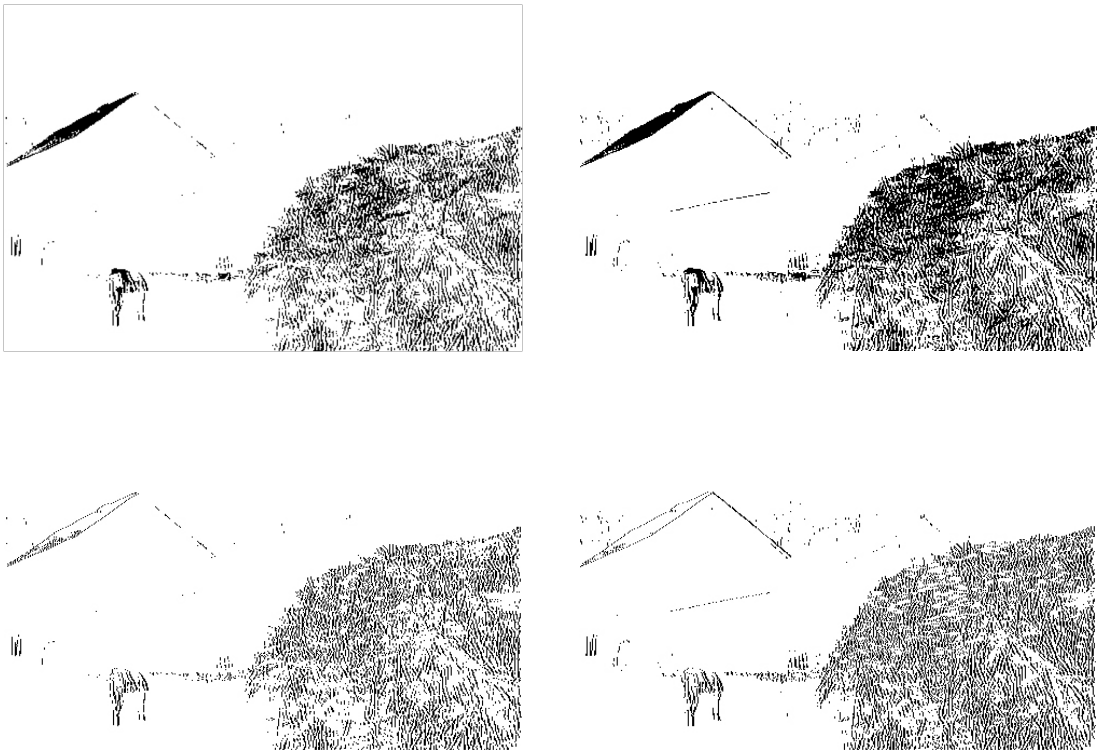

Figure 24. Top-left: the binary-threshold 10% image of *Farm*. Top-right: the binary-threshold 15% image of *Farm*. Button-left: the non-maximum suppression result of threshold 10% image of *Farm*. Button-right: the non-maximum suppression result of threshold 10% image of *Farm*.

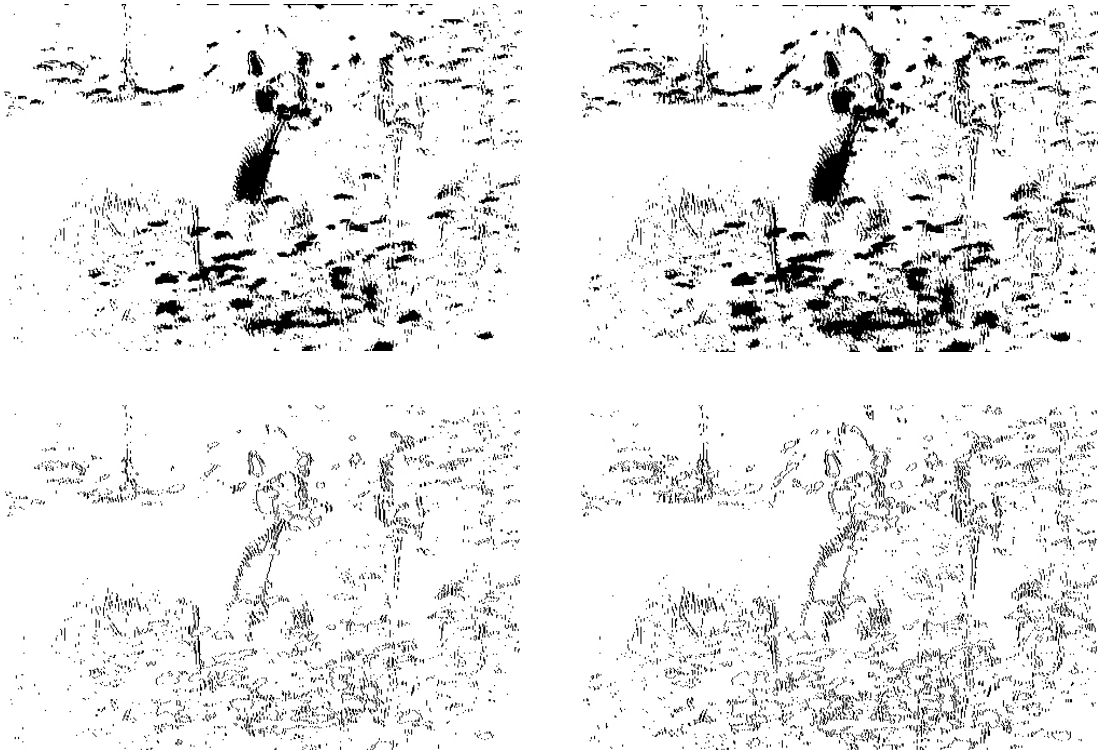

Figure 25. Top-left: the binary-threshold 10% image of *Cougar*. Top-right: the binary-threshold 15% image of *Cougar*. Button-left: the non-maximum suppression result of threshold 10% image of *Cougar*. Button-right: the non-maximum suppression result of threshold 10% image of *Cougar*.

2.3.2 Canny edge detector

Generate edge images by Canny edge detector using five different thresholds: Parameters table is below(Table 5). The results are Figure 26 - 30.

1	A: 0.3	B: 0.6	Default	Default
2	A: 0.2	B: 0.7	Default A - Δ	Default B + Δ
3	A: 0.2	B: 0.5	Default A - Δ	Default B - Δ
4	A: 0.4	B: 0.7	Default A + Δ	Default A + Δ
5	A: 0.4	B: 0.5	Default A + Δ	Default A - Δ

Table 5. Two threshold for Canny Edge detector

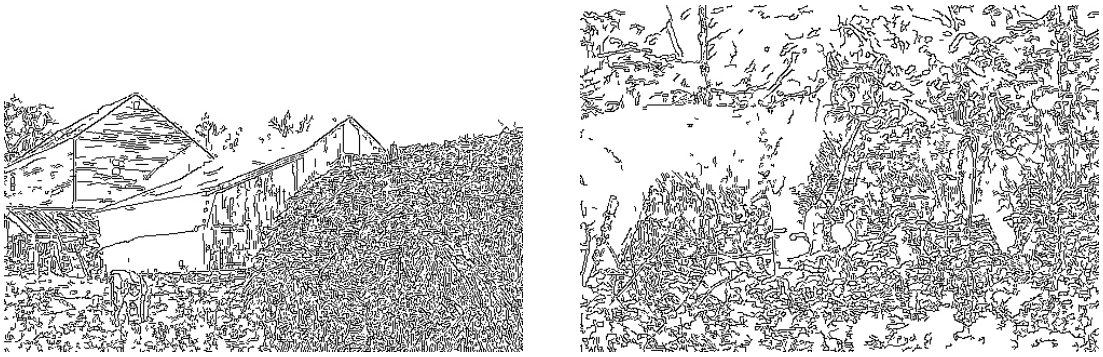

Figure 26. Canny edge detector, Threshold = (0.3, 0.6)

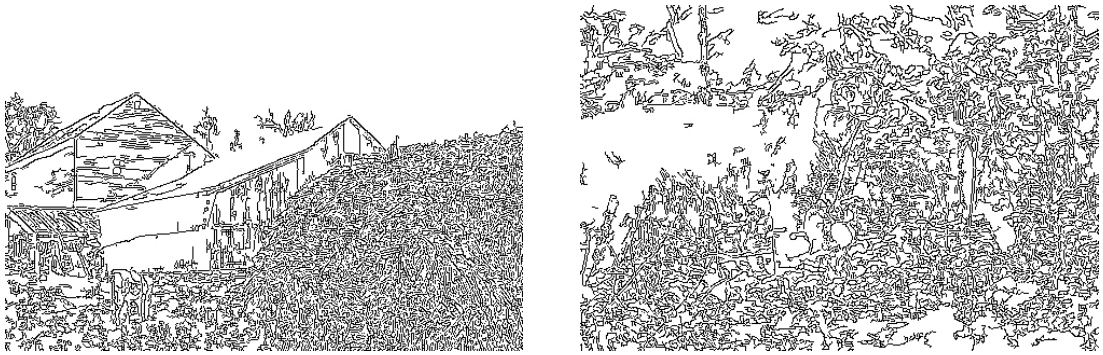

Figure 27. Canny edge detector, Threshold = (0.2, 0.7)

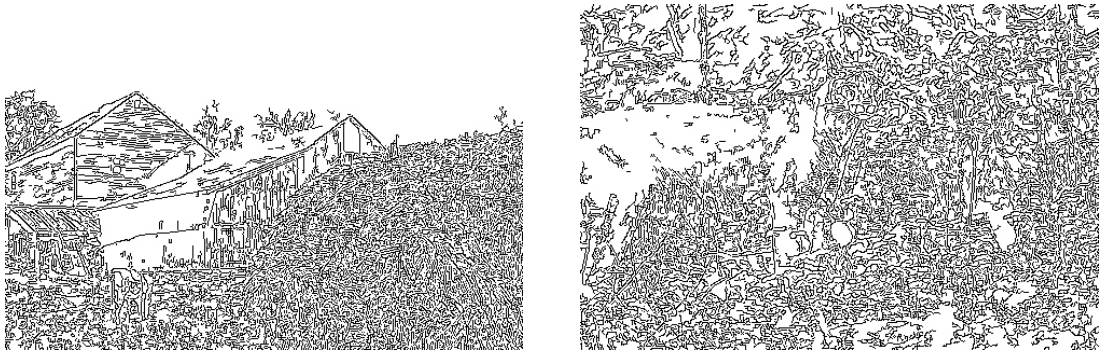

Figure 28. Canny edge detector, Threshold = (0.2, 0.5)

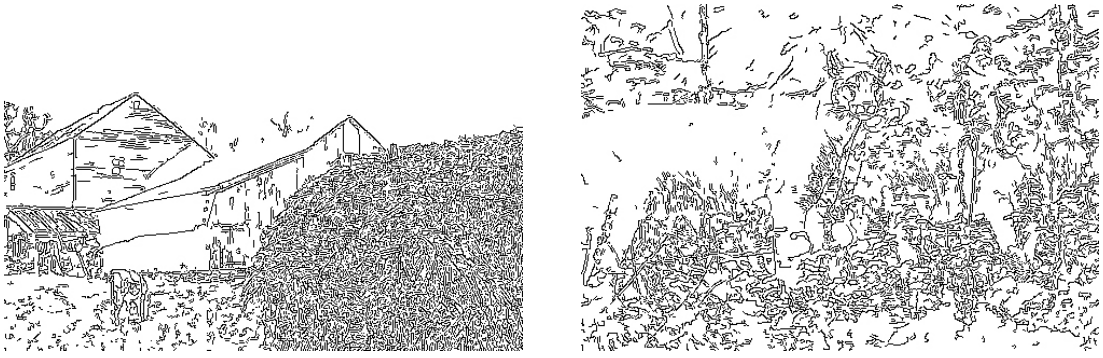

Figure 29. Canny edge detector, Threshold = (0.4, 0.7)

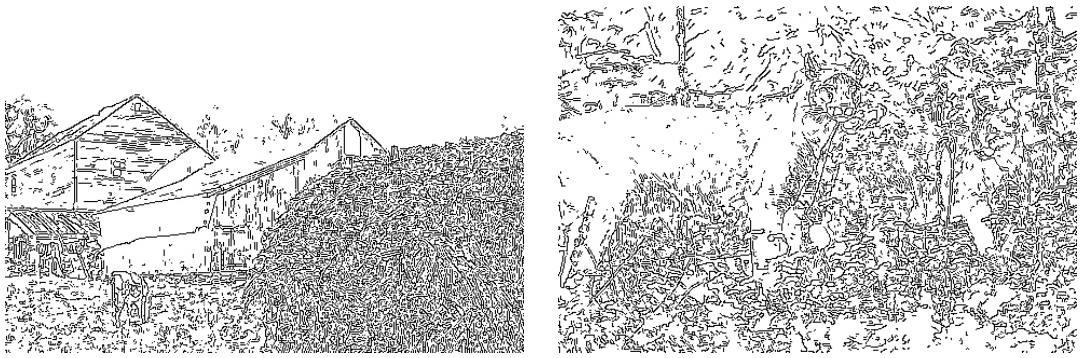

Figure 30. Canny edge detector, Threshold = (0.4, 0.5)

2.3.3 Structured edge

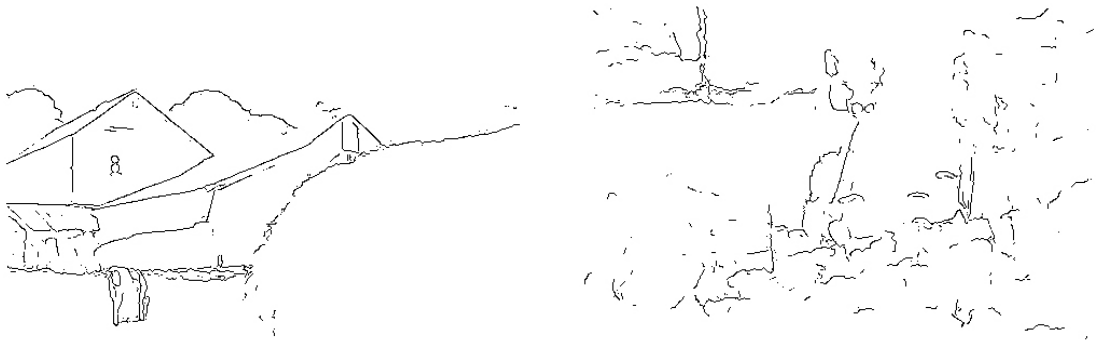

Figure 31. Structured edge detector results, Threshold = 0.02

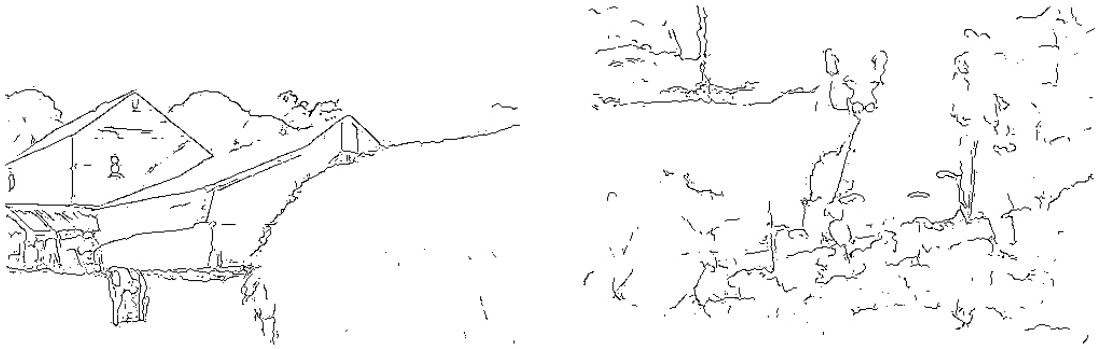

Figure 32. Structured edge detector results, Threshold = 0.03

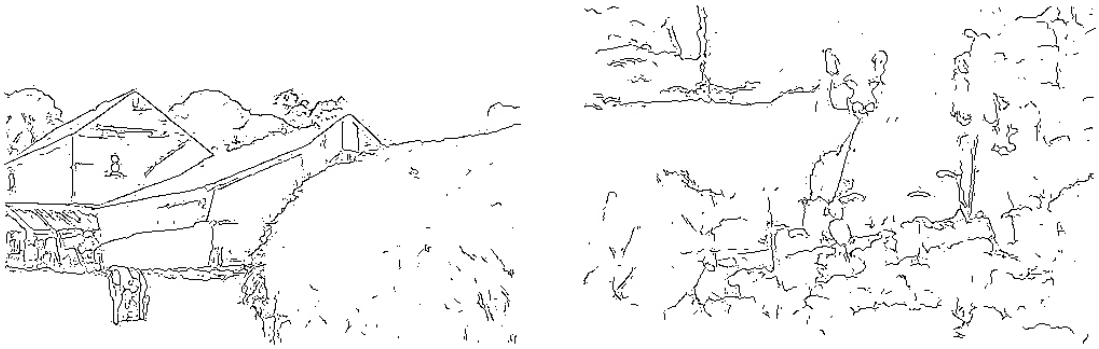

Figure 33. Structured edge detector results, Threshold = 0.04

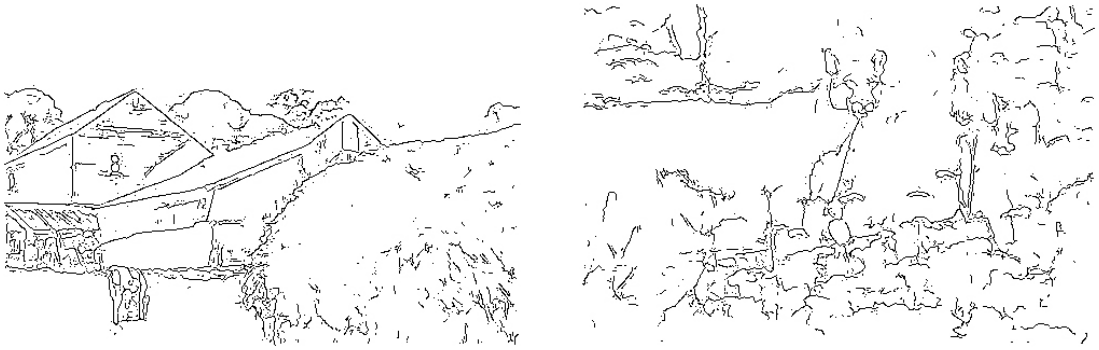

Figure 34. Structured edge detector results, Threshold = 0.05

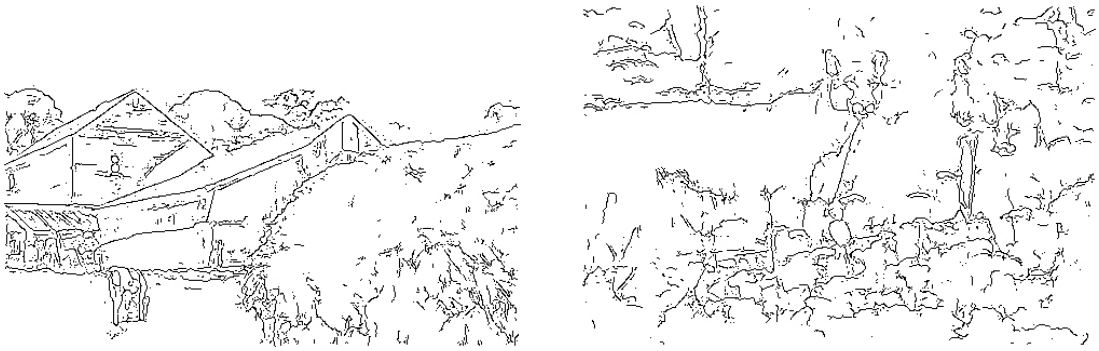

Figure 35. Structured edge detector results, Threshold = 0.06

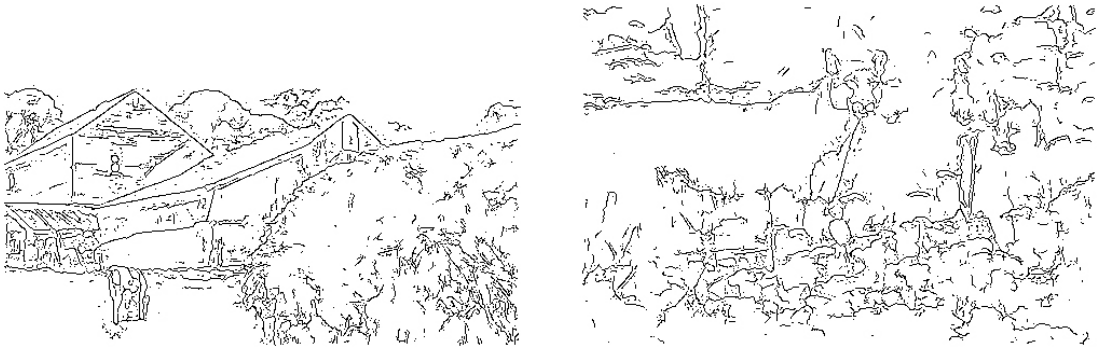

Figure 36. Structured edge detector results, Threshold = 0.07

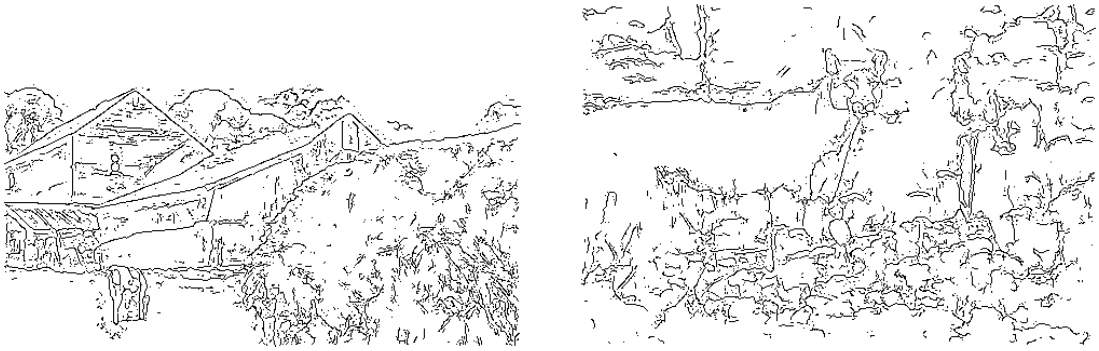

Figure 37. Structured edge detector results, Threshold = 0.08

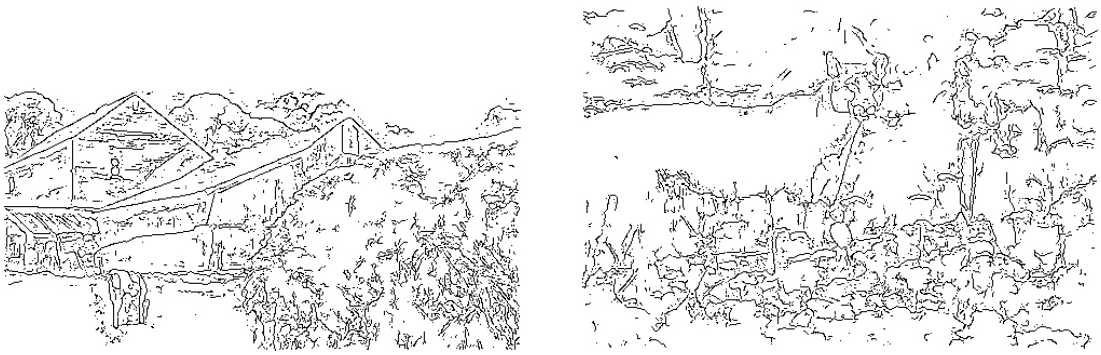

Figure 38. Structured edge detector results, Threshold = 0.09

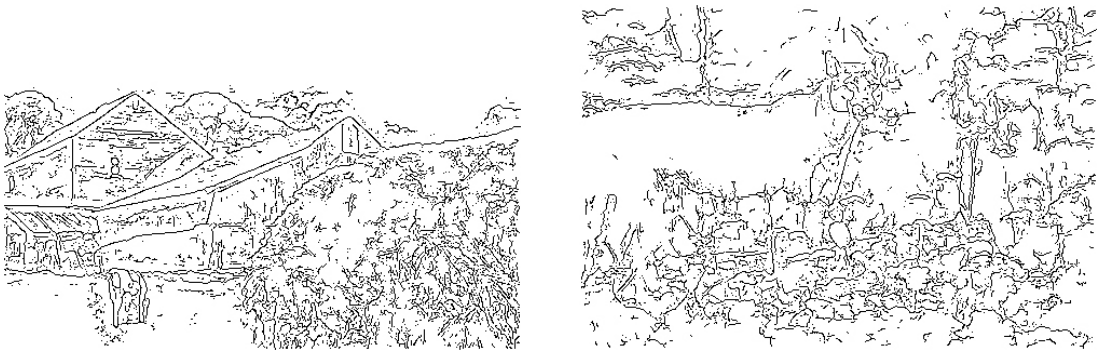

Figure 39. Structured edge detector results, Threshold = 0.10

To analysis the influence of threshold in Structured edge detector, here I choose 9 thresholds for Structure edge detector to see. These 9 thresholds are (0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10). The results are from Figure 31 to Figure 39.

2.3.4 Performance evaluation

To evaluate the performance of above results, I use F-measure to measure each image. Figure 40 is the F-measure value for *Farm* image. First two suppressed images are using Sobel, threshold is 10% and 15%. Then in "p2b: Canny", Farm_3_6 means Canny Edge Map for threshold (0.3, 0.6). In "p2c: SE", Farm_SE_02 means Structure Edge Map for threshold (0.02) of *Farm* image. Figure 41 represent the result of *Cougar* image.

```
Farm

p2a: Sobel
p2a_result/Farm_suppression_010.raw     P: 0.870841%  R: 4.97846%  F: 1.48238%
p2a_result/Farm_suppression_015.raw     P: 1.23254%  R: 8.89844%  F: 2.16518%

p2b: Canny
p2b_result/Farm_3_6.raw     P: 2.15417%  R: 27.4086%  F: 3.9944%
p2b_result/Farm_2_7.raw     P: 2.09968%  R: 28.0373%  F: 3.90678%
p2b_result/Farm_2_5.raw     P: 1.9994%  R: 28.7185%  F: 3.73852%
p2b_result/Farm_4_7.raw     P: 2.32816%  R: 26.0039%  F: 4.27369%
p2b_result/Farm_4_5.raw     P: 2.18582%  R: 26.4114%  F: 4.03749%

p2c: SE
p2c_result/Farm_SE_02.raw       P: 14.8139%  R: 19.5327%  F: 16.8491%
p2c_result/Farm_SE_03.raw       P: 11.1116%  R: 22.0384%  F: 14.7742%
p2c_result/Farm_SE_04.raw       P: 8.69354%  R: 23.0587%  F: 12.6266%
p2c_result/Farm_SE_05.raw       P: 7.01723%  R: 23.2712%  F: 10.7829%
p2c_result/Farm_SE_06.raw       P: 5.87156%  R: 23.3769%  F: 9.38571%
p2c_result/Farm_SE_07.raw       P: 5.0458%  R: 23.4419%  F: 8.30415%
p2c_result/Farm_SE_08.raw       P: 4.41836%  R: 23.4625%  F: 7.43634%
p2c_result/Farm_SE_09.raw       P: 3.93178%  R: 23.4913%  F: 6.73613%
p2c_result/Farm_SE_10.raw       P: 3.54511%  R: 23.5343%  F: 6.162%
```

Figure 40. F-measure for Farm edge map

```
Cougar

p2a: Sobel
p2a_result/Cougar_suppression_010.raw       P: 3.85445%  R: 8.9006%  F: 5.37934%
p2a_result/Cougar_suppression_015.raw       P: 3.71168%  R: 12.5372%  F: 5.72767%

p2b: Canny
p2b_result/Cougar_3_6.raw       P: 3.06524%  R: 29.8389%  F: 5.55938%
p2b_result/Cougar_2_7.raw       P: 2.85325%  R: 31.0242%  F: 5.22588%
p2b_result/Cougar_2_5.raw       P: 2.6609%  R: 32.4649%  F: 4.91866%
p2b_result/Cougar_4_7.raw       P: 3.34247%  R: 25.3973%  F: 5.90747%
p2b_result/Cougar_4_5.raw       P: 3.12919%  R: 27.5605%  F: 5.62026%

p2c: SE
p2c_result/Cougar_SE_02.raw     P: 8.83134%  R: 7.79869%  F: 8.28295%
p2c_result/Cougar_SE_03.raw     P: 8.08547%  R: 10.6099%  F: 9.17724%
p2c_result/Cougar_SE_04.raw     P: 7.26566%  R: 12.6428%  F: 9.22807%
p2c_result/Cougar_SE_05.raw     P: 6.60795%  R: 14.2821%  F: 9.03545%
p2c_result/Cougar_SE_06.raw     P: 6.1047%  R: 15.7496%  F: 8.79886%
p2c_result/Cougar_SE_07.raw     P: 5.60274%  R: 16.8391%  F: 8.40796%
p2c_result/Cougar_SE_08.raw     P: 5.21655%  R: 17.8503%  F: 8.07366%
p2c_result/Cougar_SE_09.raw     P: 4.87299%  R: 18.7357%  F: 7.73435%
p2c_result/Cougar_SE_10.raw     P: 4.57095%  R: 19.519%  F: 7.40726%
```

Figure 41. F-measure for Cougar edge map

2.4 Discussion

2.4.1 Sobel edge detector and non-maximal suppression

Sobel edge detector is a first-derivative algorithm. It takes account to the neighbor information and applies a weighted smooth effect to each x and y direction. This smooth effect reduces the noise but we can see there are a lots of true edge missing (Figure 42). Also Sobel edge detector is too sensitive to texture like barn in the *Farm* image (Figure 43).

Figure 42. Roof in Farm image. Left: Ground truth image. Right: Edge map image.

Figure 43. Barn in Farm image. Left: Ground truth image. Right: Edge map image.

2.4.2 Canny edge detector

Compared to Sobel edge detector, Canny detector has two thresholds to detect the edges. By using two thresholds T1 and T2, T1 < T2, we can get two edge images Image1 and Image2. Because Image2 is calculated by a higher threshold, it has less false edges, but more discontinuities than Image1. So we can use Image2 to find the true edge and use Image1 to get the edge points which can connect the edge points in Image2.

Compared Figure 26 to Figure 30 fives image. The most clearly image is Figure 29, Threshold A = 0.4, B = 0.7. Compared Figure 29, Figure 27, which both B threshold is 0.7 (Details in Figure 44), we can see, though the correct edge in two figures are similar, Figure 27 (threshold A = 0.2, B = 0.7) have too many false positive points which have a connection to the true positive points. This is because A threshold is used to binary a map for link discontinuities in B threshold map.

Also Figure 45 is a comparison of Figure 29 (A = 0.4, B = 0.7) and Figure 30 (A = 0.4, B = 0.5). Both two images have the same A threshold, but the results here are really different. Figure 30 (A = 0.4, B = 0.5) have too many noise that we even cannot recognize what it is. I think it is because B threshold is used for reduced the false edges.

For all the five results, we can see that Canny Edge detector is also sensitive to the texture and has too many noise.

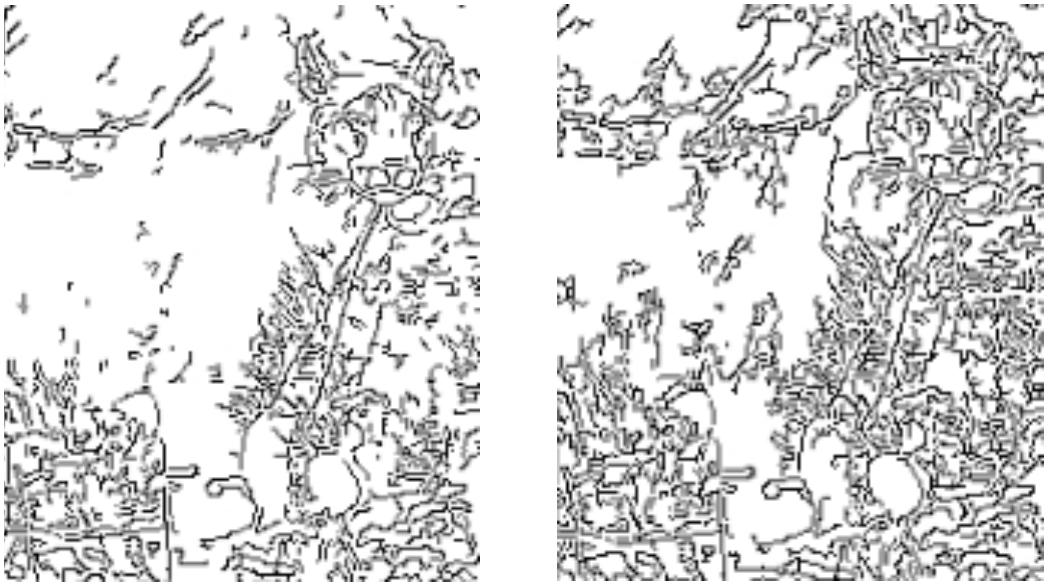

Figure 44. Cougar image. Left: threshold (0.4, 0.7). Right: threshold (0.2, 0.6)

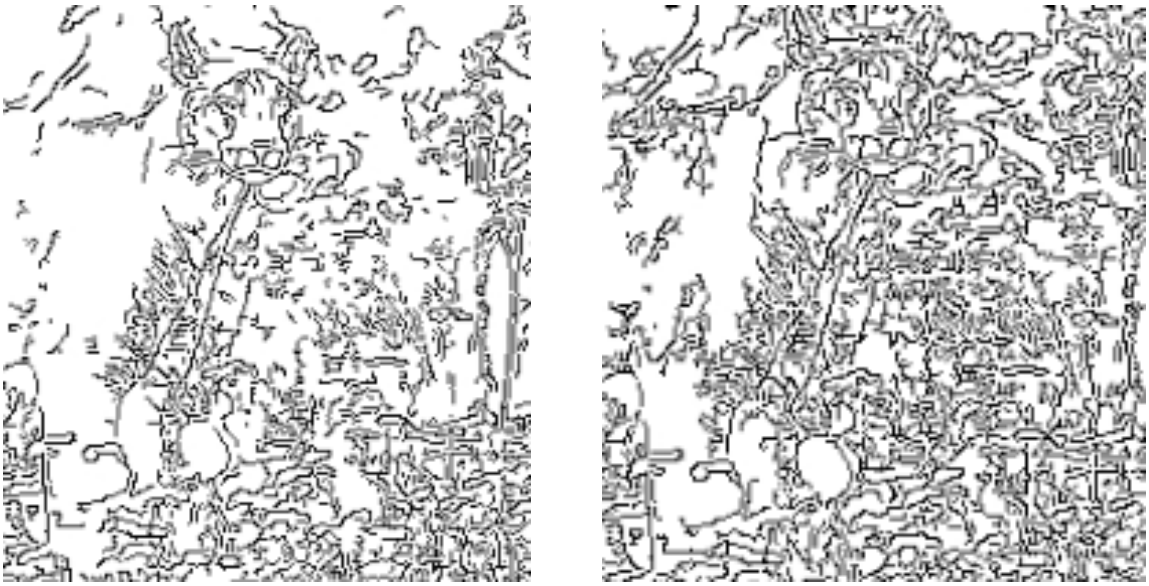

Figure 45. Cougar image. Left: threshold (0.4, 0.7). Right: threshold (0.2, 0.7)

2.4.3 Structured edge

Compared to Sobel and Canny edge detector, I find Structured edge detector perform much better than I thought. In all my 9 results, the best performance of detector is threshold 0.02 in *Farm* and threshold 0.03 in *Coug*ar.

Texture. First, it reduces the false detection of texture. We can clearly see this in Figure 46. The texture reduction effect of SE is close to human being.

Figure 46. Barn in Farm image. Left: SE (threshold = 0.02). Right: Ground truth

Continuity for above three structure edge. Compared Figure 24, 29, and 31, Sobel detector and SE detector are all have a efficient effect to increase the continuity of the image. And Sobel detector did a bad job here in detect the edge line of Roof in Farm image.

Noise. SE reduce most of false edges and perform the best in these three detector. Sobel detector has a less noise.

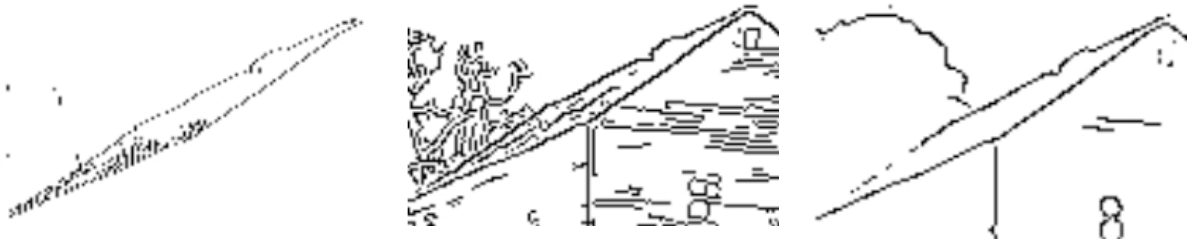

Figure 47. Roof in Farm image. Left: Sobel detector. Middle: Canny detector. Right: SE detector

2.4.4 Performance evaluation

Let us go back to the formula of P, R, F in the 2.2.4.

$$Precision: P = \frac{number\ of\ True\ Positive}{number\ of\ True\ Positive + number\ of\ False\ Positive}$$

$$Recall: R = \frac{number\ of\ True\ Positive}{number\ of\ True\ Positive + number\ of\ False\ Negetive}$$

$$F - measure = 2 \times \frac{P \times R}{P + R}$$

P value means the probability of correct pixels in the whole edge map. And R value means the probability of the correct pixels compared with the ground truth. Sometimes, we make a mistake that we think Precision P or Recall R is enough to measure the detection performance. However, here is a special case, if we only have one pixel in the edge map, and it is correct, then the precision P for the edge map is 100%, though Recall R in this case is really small. In contrast, if we fill all the pixels in the edge map, then the recall R here is 100%, but Precision P is low. To make a effective measure of edge detection, F-measure combines Precision P and Recall R to one parameter and get a optimal of P and R.

Performance of three edge detector. As the discussion above in 2.4.3, I think the best edge detector is SE detector, the worst edge detector is Sobel detector, which is shown in F-measure result in Figure 40, Figure 41. We can see that F-measure is image-dependent. *Farm* image detection always get a higher value of F-measure than *Cougar* image detection.

Reference

[1] https://en.wikipedia.org/wiki/Feature_extraction

[2] https://en.wikipedia.org/wiki/Mahalanobis_distance

[3] https://en.wikipedia.org/wiki/Principal_component_analysis

[4] EE 569 Discussion #4- Part 1, Junting Zhang.

[5] https://en.wikipedia.org/wiki/Linear_discriminant_analysis

[6] https://en.wikipedia.org/wiki/Support_vector_machine

[7] EE 569 Discussion #4, Chun-Ting Huang

[8] https://en.wikipedia.org/wiki/Edge_detection

[9] https://en.wikipedia.org/wiki/Sobel_operator

[10] https://en.wikipedia.org/wiki/Canny_edge_detector

[11] Canny, J., A Computational Approach to Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[12] http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/canny_detector

[13] Edge Detection State of The Art, P. Dollar and C. Zitnick

[14] Coutour Extraction, Chien-Yi Wang and C.-C. Jay Kuo

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

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

[17] https://github.com/pdollar/edges