University of Southern California

Viterbi School of Engineering, Ming Hsieh Department of Electrical Engineering

Texture Analysis and Classification

Aman Vora USC ID : 4057-79-6804

MS in Electrical Engineering

Email ID : amanvora@usc.edu

9th Oct 2016

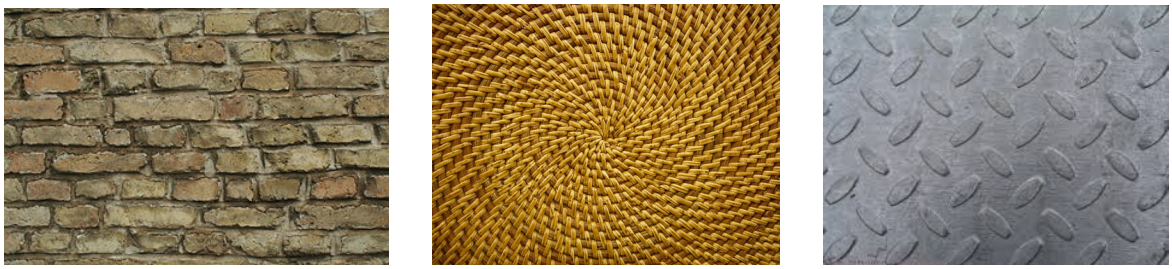
1 Texture Analysis and Classification

1.1 Motivation

Texture is a property of an image that is a characteristic visual aspect that is seen to be quite uniform and well spread out in the image. Images can have several textures. Classifying these images into several textured images is an important step in several industrial applications. This can be automated by developing a system that analyses the texture in several images and classifies them into several classes.

1.2 Approach

Texture can be thought of as a uniformly varying pattern of patches of the image. This hence has several unique statistical parameters for that texture. Consider Fig. 1.



(a) (b) (c)

Figure 1: Texture images of (a) brick, (b) wood basket and (c) metal. Each of the images has approximately uniform varying patterns and hence nearly constant and unique statistical parameters.

The statistical parameters that are studied are the variance and hence the energy distribution of the texture of the image. As different textures will result in different energy components, we can classify the different textures into different classes. Classifying these textures depends on the dimension size of the features of the image, the inter-class separation and intra-class separation.
The general flow diagram of texture analysis and classification is as shown in Fig. 2

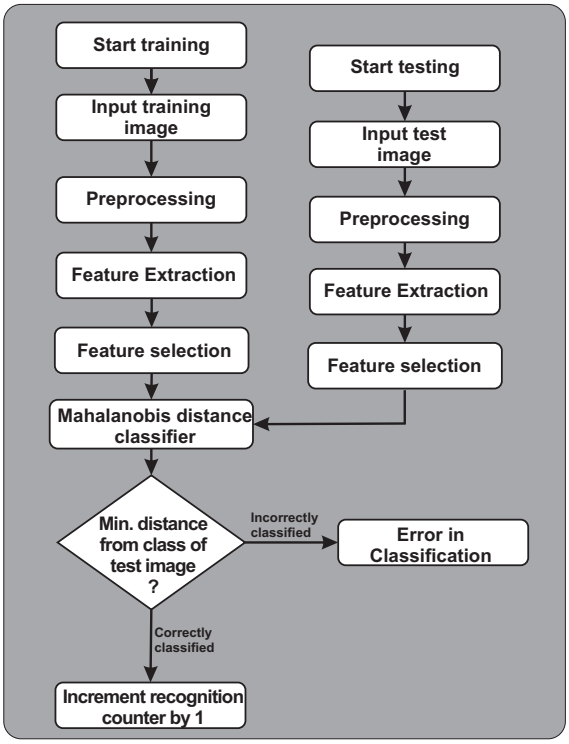


Figure 2: Various steps that are required for texture analysis and classification.

The steps involved are:

- Reading the image. They are stored in various formats such as .jpg, .png, .raw, .bmp, .tif, etc.
- Once read, preprocessing is performed. This can include removal of noise, image sharpening, histogram equalization, etc.
- Features are extracted and there are several methods to extract the features [1]. Some of these include feature extraction by using filter banks, frequency domain feature extraction such as DCT, FFT and DWT [2].
- Features extracted can be reduced using feature reduction/selection algorithms [3]. This ensures the most optimum features are retained. This removes computation complexity.
- The features of a test image can be classified using several classifiers. Some of these include Euclidean classifier, Mahalanobis minimum distance to mean classifier and SVM classifiers.

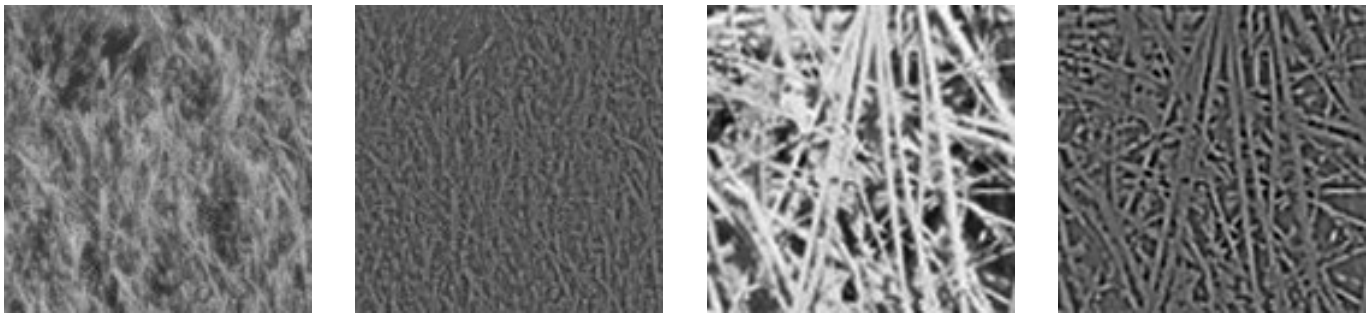
Preprocessing is favorable for good feature extraction. When noise is present in images of several images, the energy components start to resemble. This reduces inter-class separation. Hence, preprocessing is a very vital step. In the following case, the preprocessing step is to remove the dc components in the image textures. This enables to analyze the image textures by extracting their statistical parameters.

1.3 Texture Classification : Two Classes + Minimum Mean Distance Classifier

The presence of only two classes implies a binary classification of the texture images. In this case, *'grass'* and *'straw'* are the two classes considered. There are 36 grass images, 36 straw images which are labelled. These are taken as the training images. There are 24 images which are unlabelled and marked as *'unknown'*. The system is trained about the two classes using the training images and uses these classes to classify the unknown images.

1.3.1 Preprocessing

As mentioned, the statistical parameters that are made use of are the variance and hence the energy components of the image. This removes the dc level and removes the energy induced by the redundant dc levels in the features of the textures. This can be done by subtracting every pixel by the local mean. This gives us the zero mean array. With this, the energy of the texture is dependent majorally on the texture variation as shown in Fig. 3. Note that in Fig. 3, the images are scaled up to a higher gray level simply for visual display. As such the dc components are absent.



(a) (b) (c) (d)

Figure 3: Preprocessing of (a) grass texture images and the (b) preprocessed grass image. Preprocessing of (c) straw texture images and the (a) preprocessed straw image.

1.3.2 Feature extraction using filter Laws' filter bank

Once the Zero mean array is obtained, it is used to obtain the features for local window patches of the texture of the image. The Laws' filter bank is a set of filter bank that is used to obtain the energy components for a local window of the image. This filter bank depends on the 1-D kernels defined. The Laws 1-D kernels are given as shown.

```
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]
```

Tensor product of these 1-Dimensional kernels will result into a 5×5 matrix which are used to filter the texture images. As we have five 1-Dimensional kernels, we can obtain 25 Laws filter matrices. This will give features for the local window at which they are applied. For example, when the tensor product of $L5$ is taken with itself, we get the 5×5 filter matrix with co-efficients as shown in Eq. 1.

$$L5 \otimes L5 = \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{1}$$

The Laws kernels gives us 25 filter matrices which can be used to obtain the features of the local window to which it is applied. Hence, we obtain 25 filtered Laws' matrices. When the variance of the elements of the filtered matrices are calculated, nothing but the energy component is calculated because the local mean is already subtracted. This gives the 25 feature components for the particular window centered at a pixel in the zero mean array as shown in Eq. 2 and Eq. 3.

$$f_q = Var(X_q) = \frac{1}{25} \sum_{i=1}^{5} \sum_{j=1}^{5} X_q(i,j)^2 \tag{2}$$

where, q = 1, 2, ..., 25.

$$M(i,j) = \begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ f_{25} \end{bmatrix} \tag{3}$$

where, $i \in [1,M]$ and $j \in [1,N]$. The process so far can be summarized in Fig. 4

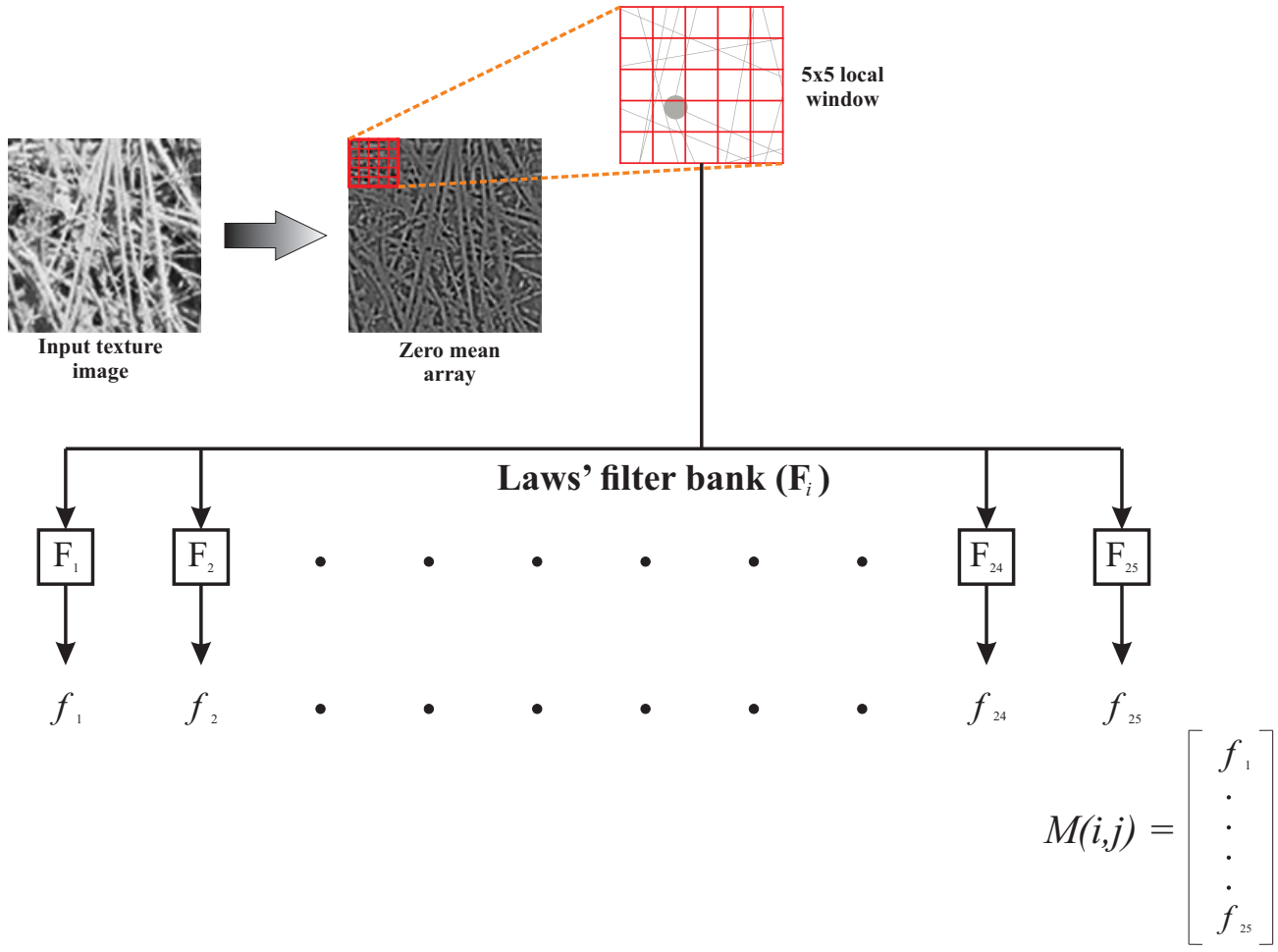


Figure 4: Various steps that are required for feature extraction.

If $M \times N$ is the size of the image, then we have $M \times N$ feature vectors for the entire texture image. When the average of all the feature vectors of the image is obtained we get the energy vector (E_I) of the texture image as shown in Eq. 4.

$$E_I = \sum_{i=1}^{M} \sum_{j=1}^{N} M(i,j) \tag{4}$$

The feature vector of all the 36 grass and 36 straw images are found. Each energy component is normalized by the maximum and minimum of that energy component of all the textured images as shown in Eq. 5.

$$E_i = \frac{E_i - E_{\min}}{E_{\max} - E_{\min}} \tag{5}$$

where, $i \in [1,25]$ The class mean is defined as the mean of all the feature vectors of all the texture images belonging to that class as given by Eq. 8

$$E_{I-mean} = \sum_{i=1}^{36} E_{I_i} \tag{6}$$

Therefore there are two class mean. One for the grass and the other for the straw. Though these vectors are 25 dimensional and cannot be visualized an intuitive idea about the class scatter and the mean can be understood from Fig. 5

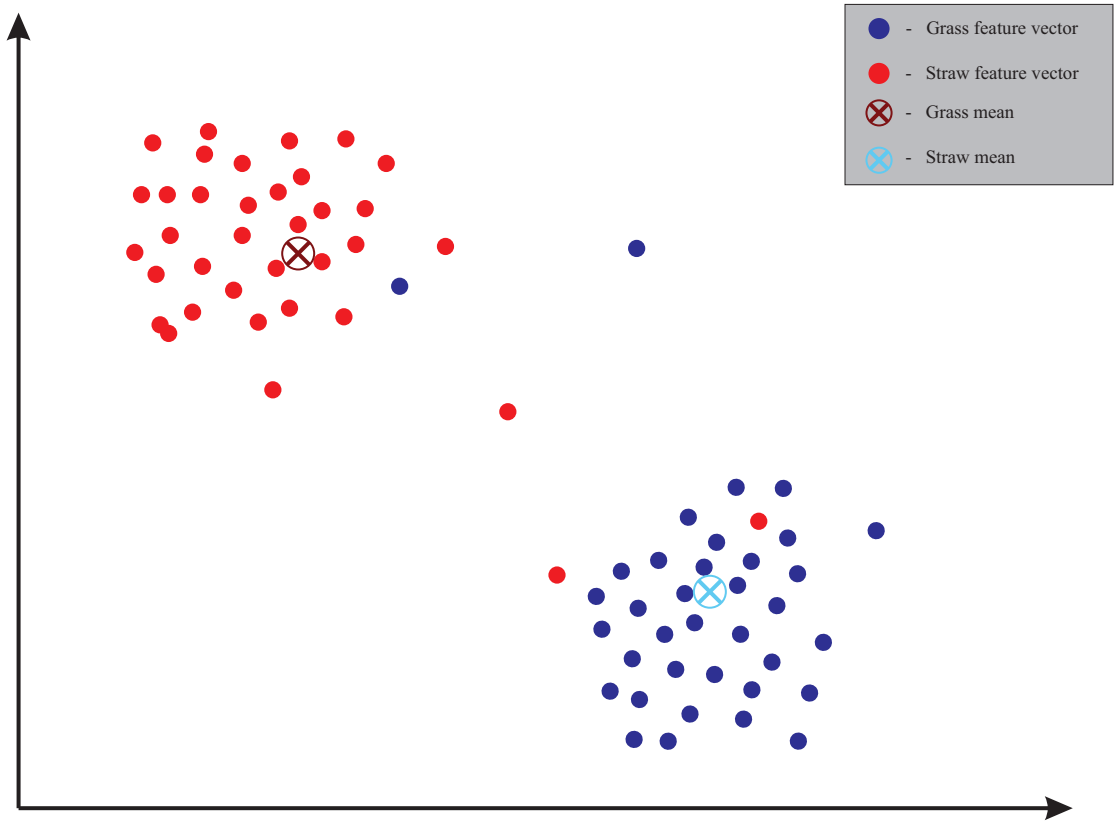


Figure 5: Feature vector of the two classes along with their mean.

1.3.3 Feature reduction using PCA

Feature extraction can often have features that are redundant. The entire set of features may not be required and the most optimum features from that can be computed. These are sufficient for texture classification. So far, the number of features that were obtained from each texture image was 25. This can reduce to any value till 1 using the procedure called **Principal Component Analysis** [4]. This process essentially transforms the feature vector $E_I \in \Re^n$ to another subspace $\Re^k$, where $1 \leq k < m$. The vectors used in the new subspace are orthogonal to each other and exhibit very low to no correlation amongst them. In PCA, the axes of principal components maximizes the variance of the dataset.
Let E_I be the feature vector of a texture image. It is represented as shown in Eq. 7

$$.E_I = [e_1, e_2, ..., e_n]^T, n \in \Re, E_I \in \Re^n \tag{7}$$

Here, e_i is a feature component of the feature vector. In PCA, all the classes are combined together and are transformed to another subspace. If there are totally N texture images or members from all the classes, then the class mean can be computed as in Eq. 8.

$$\overline{E_I} = \frac{1}{N}\sum_{i=1}^{N} E_{I_i} \tag{8}$$

This is used to compute the covariance matrix of this class as shown in Eq. 9.

(9)

This computes the eigenvectors given by $(\widehat{e_1}, \widehat{e_2}, ..., \widehat{e_n})^T$ and the corresponding eigenvalues given as $(\lambda_1, \lambda_2, ..., \lambda_n)$.
The **principal components** are those eigenvectors whose eigenvalues are the largest.
If the eigenvalues are sorted in descending order, they can be given as $\lambda_\mathrm{i} > \lambda_\mathrm{j} > ... > \lambda_\mathrm{k}$.
The corresponding k eigenvectors can be written in matrix form as $W = (\widehat{e_i}, \widehat{e_j}, ..., \widehat{e_k})^T$.
Given a feature vector of x dimension 25 (25×1), the transformed feature vector is computed as y with a reduced feature size k ($k \times 1$) as shown in Eq. 10

$$y = W^T x \tag{10}$$

The entire dataset is transformed to the other subspace with reduced feature components as shown in Eq. 11.

$$\widehat{E_I} = W^T E_I \tag{11}$$

The transformation to another space is as shown in Fig. 6. The features are projected to another space which have e_i and e_j as the eigenvectors with greatest corresponding eigenvalues. These are nothing but the principal components.

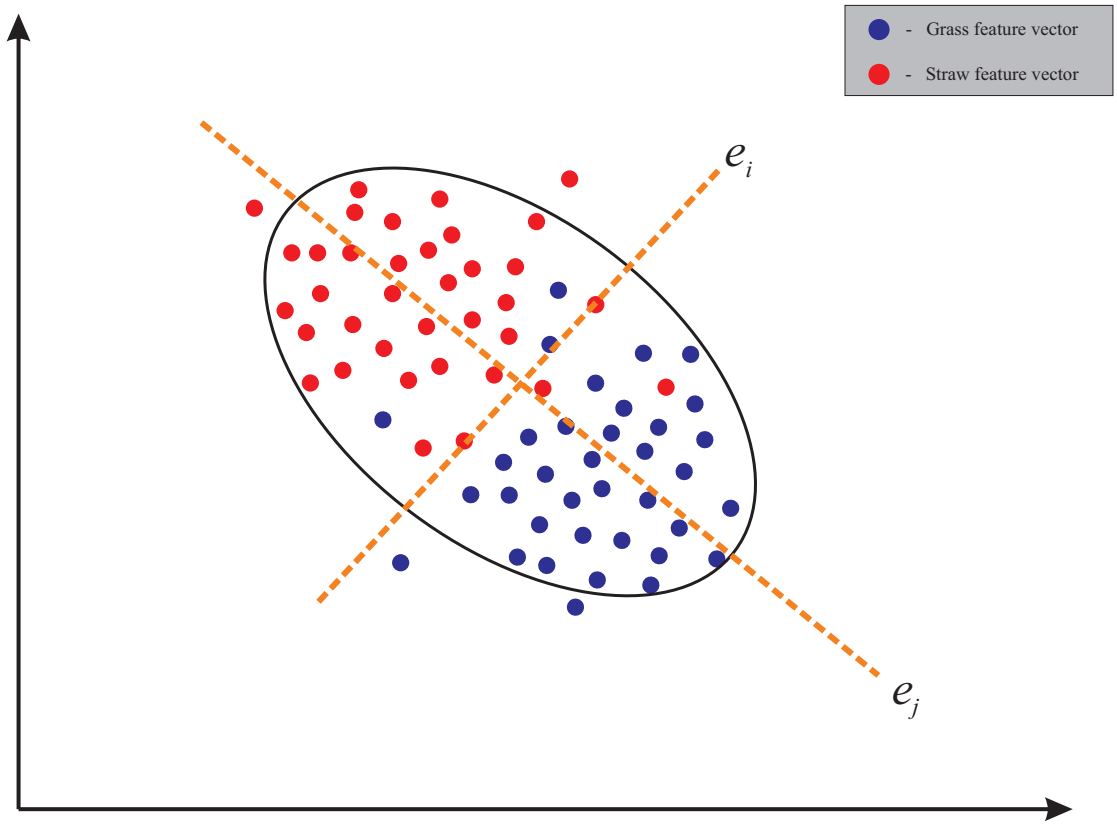


Figure 6: Computation of the principal components.

In the binary classification problem of grass and straw images, PCA is used to reduce the feature vector from 25 dimensions to 1 dimension. The entire dataset is taken as a whole to compute the principal components. In other words, it makes no use of the class labels.

1.3.4 Feature reduction using LDA

Unlike PCA, Linear Discrimant Analysis [5] can be termed as a 'supervised learning'. In PCA, the class labels are ignored and the entire dataset is taken as a whole to compute the principal components. But in LDA, the class labels are given as an input information. The procedure of LDA is :

- **Compute the n-dimensional mean of each class**
  For the class c, the mean of the class with a population of N is obtained as shown in Eq. 12.

$$\overline{E_c} = \frac{1}{N}\sum_{i=1}^{N} E_{c_i} \tag{12}$$

  Here E_{c_i} is the n-dimensional feature vector belonging to the class c.

- **Compute the scatter/covariance matrices**
  There are 2 scatter matrices that are computed :

  1. Within-class scatter matrix S_w The scatter matrix is computed for every class c as shown in Eq. 13.

$$S_c = \sum_{i=1}^{N} (E_{c_i} - \overline{E_c})(E_{c_i} - \overline{E_c})^T \tag{13}$$

     For a total of C classes, the within-class scatter matrix is given as shown in Eq. 14.

$$S_W = \sum_{i=1}^{C} S_c \tag{14}$$

  2. Between-class scatter matrix S_B After computing the mean of each class, i.e., $\overline{E_c}$, the between-class scatter matrix is given as shown in Eq. 15.

$$S_B = \sum_{i=1}^{C} N(\overline{E_c} - \overline{E})(\overline{E_c} - \overline{E})^T \tag{15}$$

- **Obtain eigenvectors and eigenvalues of the matrix** ${S_W}^{-1}S_B$ These are eigenvectors are computed by $(\widehat{e_1}, \widehat{e_2}, ..., \widehat{e_n})^T$ and the corresponding eigenvalues are given by $(\lambda_1, \lambda_2, ..., \lambda_n)$.

- Just like in PCA, if the **eigenvalues are sorted in descending order**, they can be given as $\lambda_\mathrm{i} > \lambda_\mathrm{j} > ... > \lambda_\mathrm{k}$. The corresponding k eigenvectors can be written in matrix form as $W = (\widehat{e_i}, \widehat{e_j}, ..., \widehat{e_k})^T$.

- **Obtaining linear discriminants to transform the feature vectors to a new subspace** With the $n \times d$ dataset and the transformation matrix W which is $d \times k$, the transformed dataset Y is given in Eq. 16

$$Y = W \times X \tag{16}$$

  The transformed dataset is not only reduced using LDA, but the linear discriminants separate the classes to the maximum extent possible.

The transformation to another space is as shown in Fig. 7. The features are projected to another space which have e_i and e_j as the linear discriminant vectors.

Both PCA and LDA are feature dimension reduction techniques. In PCA, the axes that obtains the maximum variance in the entire dataset is computed. These axes or principal components form the transformation matrix to reduce the dataset from n-dimension to k-dimension. In LDA, the axes that obtains the maximum separation between each class is computed. This is where the class labels are important to identify the class means, scatter matrices and hence the within-class scatter matrix. These axes or linear discrimant components form the transformation matrix to reduce the dataset from n-dimension to k-dimension.
In PCA, the dimension reduced is such that $1 \leq k < d$. However, in LDA, there is a constrain. The dimension reduced is such that $1 \leq k \leq C - 1$.

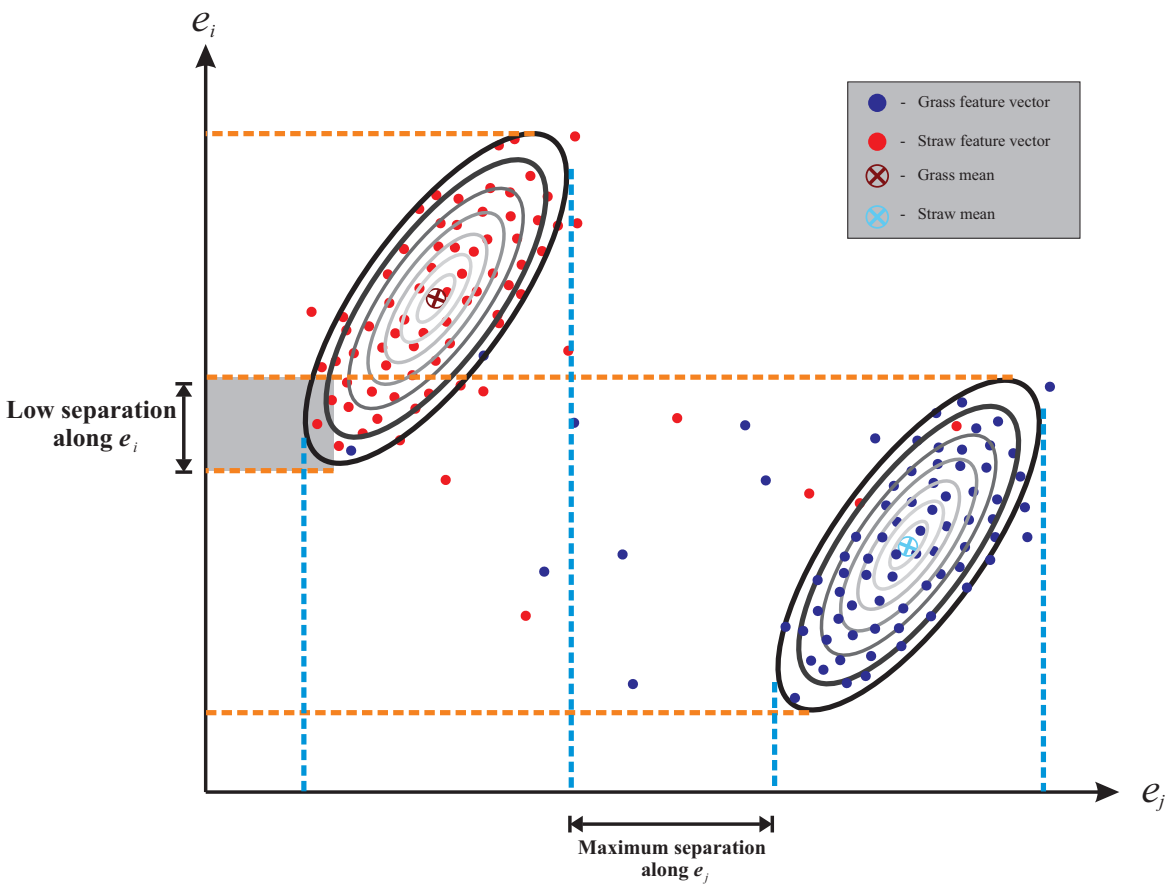


Figure 7: Computation of the linear discrimant components.

In LDA, the feature dimension can be reduced to *C-1* for C classes.

1.3.5 Mahalanobis distance classifier

Once the features are available, either optimized/reduced or not, classification is performed to determine as to which class the vector and hence the image belongs to. The Mahalanobis distance between a point and a distribution as a whole. The distribution looks like an ellipse with major and minor axes. The major axis is orthogonal to the minor axis. In the mahalanobis distance computation, the feature vectors are transformed to another subspace and the axes of the ellipse are the axes along which measurements are made. If X and Y are the coordinate axes of the ellipse with major and minor axes X' and Y', then X' and Y' are the axes of measurements in the new subspace.
The reason to transform to a new subspace is because the variation is different along different dimension in the space R^{n}. Hence, the distance traversed in one axis should not be given the same importance as the distance traversed in the other. The mahalanobis distance makes use of the covariance matrix (C) calculated from all the texture images, inverts it. Also the class mean (μ) is made use of.

$$d = \sqrt{(X-\mu)^T C^{-1}(X-\mu)} \tag{17}$$

The pseudo code to calculate the mahalanobis distance for a sample vector X and the mean of the class is as shown below.

```
CalcCovarMatrix(texture_gr, cov_gr, mean_gr)
cov_gr = (1/(N-1))cov_gr
invert(cov_gr, inv_cov_gr)
dist_gr = Mahalanobis(X, mean_gr, inv_cov_gr)

CalcCovarMatrix(texture_st, cov_st, mean_st)
cov_st = (1/(N-1))cov_st
invert(cov_st, inv_cov_st)
dist_st = Mahalanobis(X, mean_st, inv_cov_st)
```

```
if(dist_gr < dist_st)
    // Classify sample X as grass
else
    // Classify sample X as straw

if X wrongly classified
    err_count += 1
```

The sample can be a trained texture image or an unlabelled texture images. The error rate can be determined for the training images and the unlabelled texture images. Hence the training error rate and the unlabelled texture images cam be computed. The classification is as shown in Fig. 8. The red grass image has two distances computed. One is the mahalanobis distance with the grass images and the other with the straw images. It is classified to that class to which it is closest.

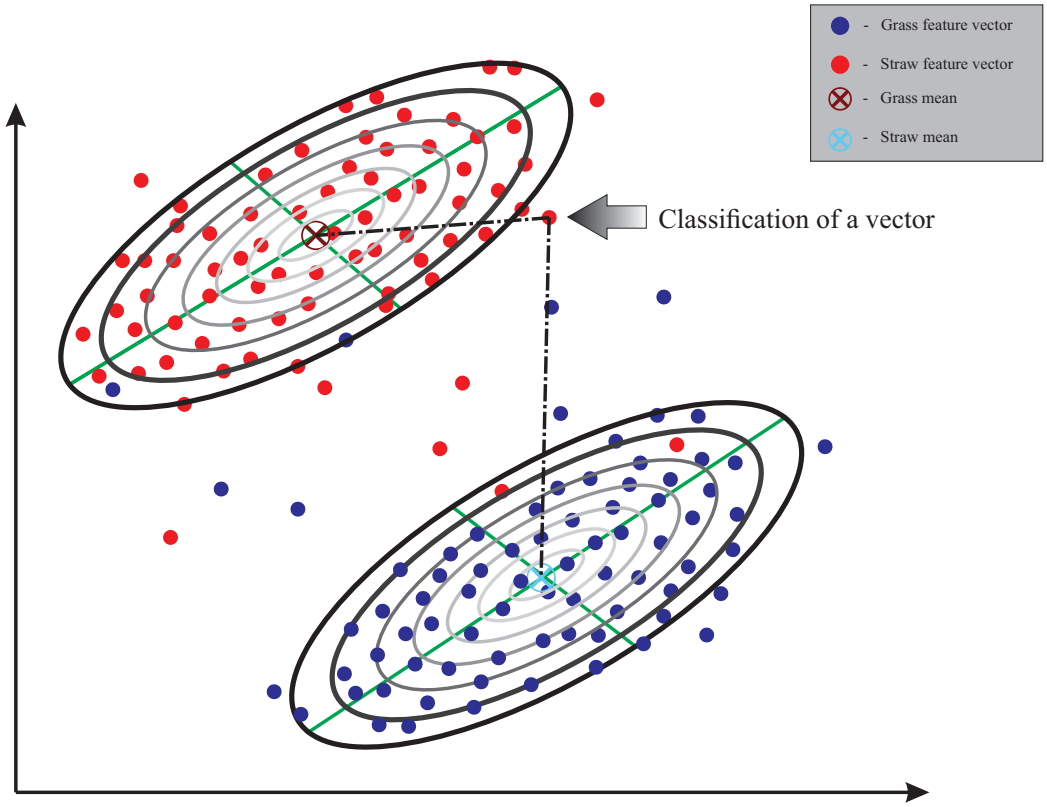


Figure 8: Comparison of the mahalanobis distance between the vector and the two class mean.

1.4 Advanced Texture Classification: Multi-Classes + SVM

In the multi-class texture classification, there are more than 2 classes. Earlier only *'grass'* and *'straw'* were the binary classes. In this case, there are 2 more classes called *'leather'* and *'sand'*. However the classifiers remain binary classifiers. They are :

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

There are essentially two groups of images. The first group contains 36 images of one class. The other contains 12 images each of the other three classes.

1.4.1 Feature extraction

As with the case with binary texture classification, the texture images are also read here in preprocessed. The image is preprocessed by removing the dc component and hence converting it to a zero mean array. The Laws' filter banks are applied to obtain 25 feature components of the texture image.

1.4.2 Feature reduction

The features are reduced using PCA or LDA.
PCA is used and we reduce the 25 dimensional feature vectors to 3 dimensional feature vectors.
In case of LDA, the labels represents the 2 groups, not the 4 classes. Hence, LDA uses binary labels and can be used to reduce the features from 25 to 1.

1.4.3 SVM classifier

SVM or support vector machine is a modern classification technique. It is more robust than simple Euclidean and Mahalanobis minimum distance to mean classifiers. Feature vectors can be multi-dimensional. Hence, the classification step to divide two classes is to partition them using hyperplanes. These are linear planes that use support vectors each belonging to a class. The width between the support vectors is called the margin (ρ). When this width is maximized, the classification gives best performance.
Support vector machines solves a major issue with many classifiers. It has soft margin classification. This means that a feature vector that is present in the cluster of the other class can still be classified to its correct class as a soft margin error is allowed as shown in Fig. 9.

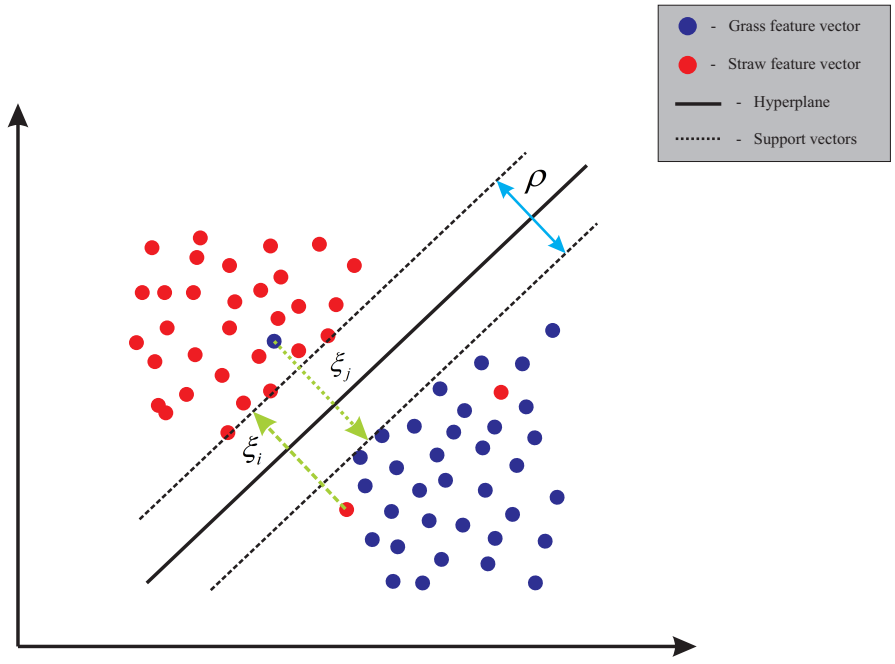


Figure 9: SVM classfication with soft margin error allowed.

If w is the normal to the decision hyperplane, x is a data point and b is a scaling factor, then we have the hyperplane given by Eq. 18.

$$w^T x + b = 0 \tag{18}$$

SVM uses a set of optimization methods to obtain the best result of classification. The class is represented by y which takes the value either 1 or -1. Depending on where the data point of the test vector lies, a score is evaluated. This is compared to the confidence threshold t.

```
if score > t
    label = 1
else if score < -t
    label = -1
```

```
else
    classification fails
```

1.5 Results

The results of the texture classification is presented for the two parts as given below.

1.5.1 Texture Classification : Two Classes + Minimum Mean Distance Classifier

The error rate was defined for both the training data and the testing data. There are two types of computation time obtained.

- Computation time in sec to extract the features of the texture image and perform feature reduction if any.
- Computation time in ms to classify a test/train vector.

These were average time obtained.

Table 1: Error rate and computation time for binary texture classification

Method	Error rate			Computation time (in s)	
	Training		Test	Training	Test
	Grass	Straw			
Mahalanobis	0	0	0	0.573	1.947m
PCA + Mahalanobis	0	0	0	0.647	1.458m
LDA + Mahalanobis	0	0	0	0.629	1.468m

1.5.2 Advanced Texture Classification: Multi-Classes + SVM

The error rate and the computation time is obtained for multi-class texture classification as well.

Table 2: Error rate and computation time for multi-class texture classification

Method		Grass Vs non-grass classifier		Leather Vs non-leather classifier		Sand Vs non-sand classifier		Straw Vs non-straw classifier	
		Training	Test	Training	Test	Training	Test	Training	Test
Mahalanobis	Error Rate	1.389	4.167	22.22	20.84	22.22	33.33	11.11	16.66
	Computation time (in s)	0.745	8.25m	0.749	7.916m	0.782	8.875m	6	8m
SVM	Error Rate	0	0	22.22	16.66	19.45	16.66	13.89	12.5
	Computation time (in s)	0.63	6.5m	0.657	7.125m	0.776	5.883m	0.752	9.583m
PCA + SVM	Error Rate	48.89	50	47.34	50	33.33	50	43.056	45.834
	Computation time (in s)	0.682	6.791m	0.689	7.083m	0.693	6.25m	0.759	7.133m
LDA + SVM	Error Rate	0	0	27.77	20.834	22.22	16.66	16.66	16.6
	Computation time (in s)	0.652	5.833m	0.634	6.5m	0.64	5.833m	0.649	6.5m

1.6 Discussion

1.6.1 Texture Classification : Two Classes + Minimum Mean Distance Classifier

The texture classification resulted in no error with or without the use of feature reduction. It is seen clearly that all the 25 feature components are not required. In both, PCA and LDA feature reduction techniques, there was no error built by reducing the feature vectors. A main reason for this could be because the features are already well separated in the original subspace. When the principal components were obtained, the axes of the new subspace was such that the variance of the dataset high. The linear discriminant vector obtained also finds good inter-class separation in the transformed subspace. Hence, obtaining optimum features is highly suitable. When the population of such classes increases, it can aid in storage of the features by a margin of 25.

Timing performance:
It is seen that the time to reduce the feature size from 25 to 1 component increases the computation time in comparison to simply extracting feature vectors. However there is a very nominal increase of just 11.43% increase in computation time. The classification of an image is much more easier than before due to the reduced feature dimension. The time taken to classify has also decreased.

1.6.2 Advanced Texture Classification: Multi-Classes + SVM

In multi-class classification, the performance depended on the classifier. It was seen that leather and non-leather classifier for leather images as one entire class has a higher error rate. This is because the intra-class separation of the leather class is comparatively high. Even the sand class has a lot of intra-class separation. Since these classes are scattered a lot, they affect those classifiers where these images are present in a large quantity.
It was clearly seen that classification using minimum distance to mean classifier is clearly outperformed by SVM classifier both in obtaining lower error rates and lower computation time to classify a given texture image.

It is seen that as the number of classes have increased, the performance of LDA is better than PCA. LDA uses the supervised learning of two binary groups rather than 4 separate classes. PCA fails to obtain the axes of high variation amongst the various classes. However linear discriminant eigenvectors are easily able to find adequate inter-class separation amongst each other in the new subspace.

2 References

[1] Todd R. Reed and JM Hans Dubuf, "A review of recent texture segmentation and feature extraction techniques" CVGIP: Image understanding 57.3 (1993): 359-372.

[2] Rabab M. Ramadan, and Rehab F. Abdel-Kaderx "Face recognition using particle swarm optimization-based selected features", International Journal of Signal Processing, Image Processing and Pattern Recognition 2.2 (2009): 51-65.

[3] Imola K. Fodor, "A survey of dimension reduction techniques", (2002).

[4] `http://sebastianraschka.com/Articles/2014_pca_step_by_step.html`

[5] `http://sebastianraschka.com/Articles/2014_python_lda.html#lda-in-5-steps`

[6] For documentation of the report, `http://tex.stackexchange.com`