INDIAN INSTITUTE OF TECHNOLOGY

KHARAGPUR

PATTERN RECOGNITION AND MACHINE INTELLIGENCE IN MEDICINE (MM61504)

And

MEDICAL IMAGE ANALYSIS (EE61008)

A Report on

Breast Cancer Detection and classification from Mammogram images based on GMM Segmentation and GLCM-DWT feature extraction and PNN Classification

Under the Guidance of

DR. NIRMALYA GHOSH

Assistant Professor, Department of Electrical Engineering, IIT Kharagpur, India

PRABIR KUMAR DAS

Roll – 20MM62R02

M-Tech in Medical Imaging and Informatics

SHIVANI SINGH

Roll - 20MM61R10

M-Tech in Biomedical Engineering

SCHOOL OF MEDICAL SCIENCE AND TECHNOLOGY

Introduction

In India and over the world, Cancer has become a deadly disease and more and more people are suffering from Cancer and a survey says one in every 30 women suffer from this disease in their lifetime and so basically the project was first thought of because of the increase in cases of breast cancer and one thing which is very important that if we can detect the Cancer at an early stage then there is an increased chance of it getting cured. So, this project lays a foundation in making the detection of the cancer automated so that more and more people can get it diagnosed early so as get cured.

Breast cancer originates when cells grow uncontrollably in the breast resulting in a tumour that can be felt as a lump or observed on x-ray. The tumour is malignant that is cancerous if the cells invade surrounding tissues or spread to distant areas of the body. Breast Cancer can be observed both in women as well as men. Image processing aims at divide one picture into different types of classes or regions, recognition of objects and detecting of edges, etc that is done after the image is segmented. The main aim of this paper is to detect and separate background and foreground by using Gaussians Mixture Model, the parameters of the model and training data are learned by EM-algorithm. Pixel labelling corresponding to each pixel of the true image is done by Bayesian rule. This labelled image is constructed during of running EM-algorithm. Numerical method of finding maximum a posterior estimation is done by using EM-algorithm and Gaussian mixture model which we called EM-MAP algorithm.

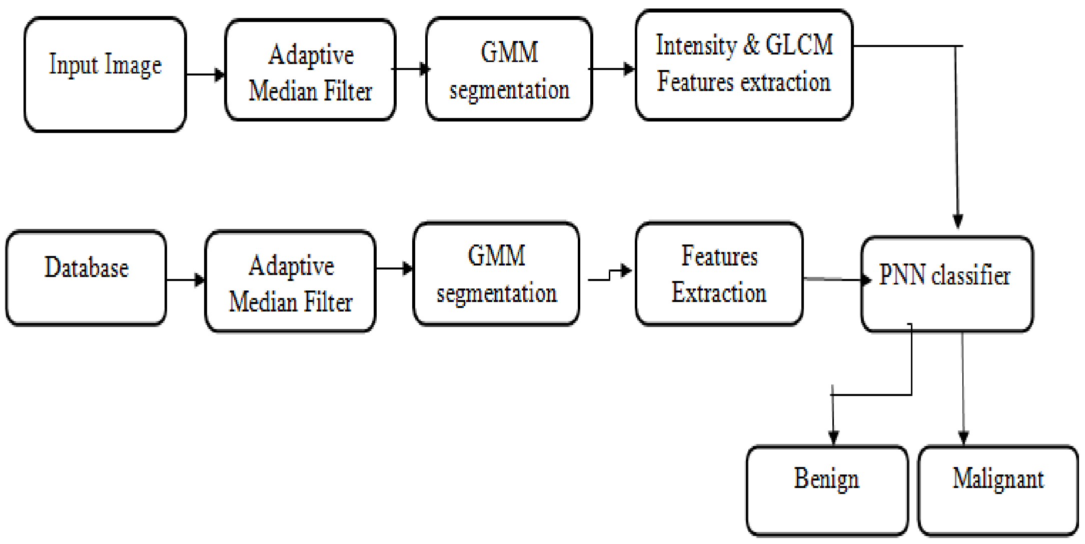
The signs of detection are Masses and micro calcification clusters which are important in early detection of breast cancer. Micro-calcifications are nothing but tiny mineral deposits within the breast tissue. They look similar to small white-coloured spots. They may or may not be caused by cancer. Masses can be many things, including cysts (fluid-filled sacs) and non-cancerous solid tumours, but they could also be cancerous. The difficulty in cancer detection is that the abnormalities from normal breast tissues are hard to read because of their subtle appearance and ambiguous margins. Automated tools which can help radiologist in early detection of breast cancer.

Facing the condition that the inefficient training of traditional classifiers in the classification process of mammography, a classification method is proposed combining image processing and supervised learning. Firstly, the improved adaptive median filter enhances the image contrast. Then, according to the result of breast segmentation based on Gauss Mixture Model (GMM), this project proposed a classification model based on Probabilistic Neural Network optimized (PNN) optimized by Gray Level Co-occurrence Matrix (GLCM) and Discrete Wavelet Transform (DWT).

Objective

Our objectives in this project are to – pre-processing and noise removal of breast mammogram images, segment the images to identify Region of Interests (ROI) using Machine Learning techniques, extract useful feature information using GLCM, DWT and finally classify the images into three categories after its detection - Normal, Malignant, Benign.

Block Diagram of the Project



DATASET USED

For testing and analysis of the proposed algorithm, randomly selected 100 images of Mammographic Image Analysis Society (mini-MIAS) database, organized by J Suckling et al.in 1994 [13]. The team developed a database of digital mammograms. Films taken in the UK National Breast Screening Programme (NBSP). Images were digitized to 50 micron pixel and represented with an 8-bit word of each pixel. It reduced to a 200 micron pixel and padded, all the images are in 1024×1024 size. The database consists of 322 digitized mammograms (Among which it consists 202 normal and 120 abnormal images). It also includes radiologist’s markings on the locations of abnormalities if present. Mammographic images are available online in Pilot European Image Processing Archive (PEIPA) at the University of Essex.

Dataset 2 : University of South Florida :Digital Mammography /Home Page

http://www.eng.usf.edu/cvprg/Mammography/DDSM/thumbnails/normals/normal_02/overview.html

Images containing suspicious areas have associated pixel-level "ground truth" information about the locations and types of suspicious regions. Also provided are software both for accessing the mammogram and truth images and for calculating performance figures for automated image analysis algorithms.

Methodology

IMAGE PROCESSING : Contrast enhancement

Mammography is the basic screening test for breast cancer. It consists many artefacts, which negatively influences in detection of the breast cancer. Therefore, removing artefacts and enhancing the image quality is a required process in Computer Aided Diagnosis (CAD) system. The accuracy and efficiency of the CAD is increased by providing exact Region of Interest (ROI). Extracting ROI is a challenging task in pre-processing because the presence of pectoral muscle influences the detection of abnormality. Here, the proposed show that the wiener filter and Contrast Limited Adaptive Histogram Equalization (CLAHE) techniques efficiently aids for enhancing the quality of the image, thereby it also removes the unwanted background and the pectoral muscle by using thresholding and modified region growing technique respectively.

The types of noise observed in the mammogram image are marked in the Figure 1. In the proposed algorithm, it abolishes all these unwanted and surplus noises from the mammogram image.

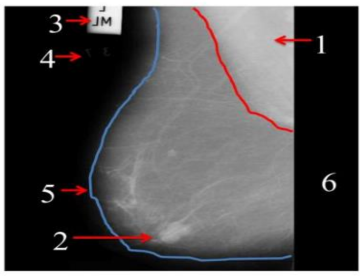


Fig 1. Types of noises observed in original image and marked with numbers as 1. Pectoral Muscle, 2.Tumor, 3.High Intensity, 4.Low Intensity, 5. Breast Part and 6. Background.

The main purpose of image pre-processing is to help the radiologist and doctors to take decision swiftly. By providing exact ROI will help to identify abnormality. The proposed method works in three stages as explained in figure 2. The first step is to remove the back ground artefacts identified in the figure 1. The second step is to reduce the pectoral muscles identified in figure 1 and the digital mammography enhanced by using wiener filter and Contrast Limited Adaptive Histogram Equalization (CLAHE).

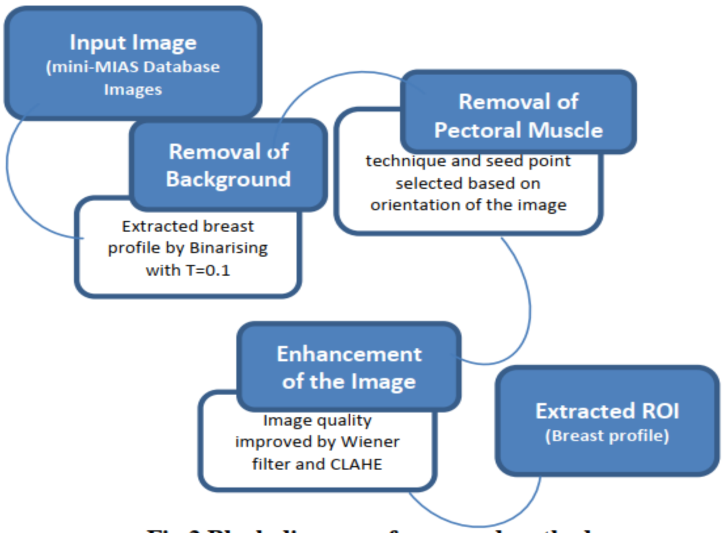

Input Image
(mini-MIAS Database Images
Removal of Background
Extracted breast profile by Binarising with T=0.1
Removal of Pectoral Muscle
technique and seed point selected based on orientation of the image
Enhancement of the Image
Image quality improved by Wiener filter and CLAHE
Extracted ROI
(Breast profile)

Fig 2 Block diagram of proposed method

3.1 Background Removal

Initially image was binarized with threshold value 0.1 then the connected component organized in descending order to extract the largest blob which is the breast profile but consists of pectoral muscle.

3.2 Suppression of pectoral muscle

The second stage was used to reduce the pectoral muscle part by using modified region growing technique. The seeded region growing is one of the image segmentation methods, it works in two ways based on selected pixel locational value and other is selection of seed point. The seed point may be selected adaptively or manually. In the proposed method, seed point is selected automatically by considering the orientation of the mammography. This approach determines the neighbouring pixels of seed point and examines whether the next pixels should be added to the region or not. The process is iterated till to extract the complete ROI.

3.3 Image enhancement

Third stage was used to enhance the quality of the image using wiener filter and CLAHE.

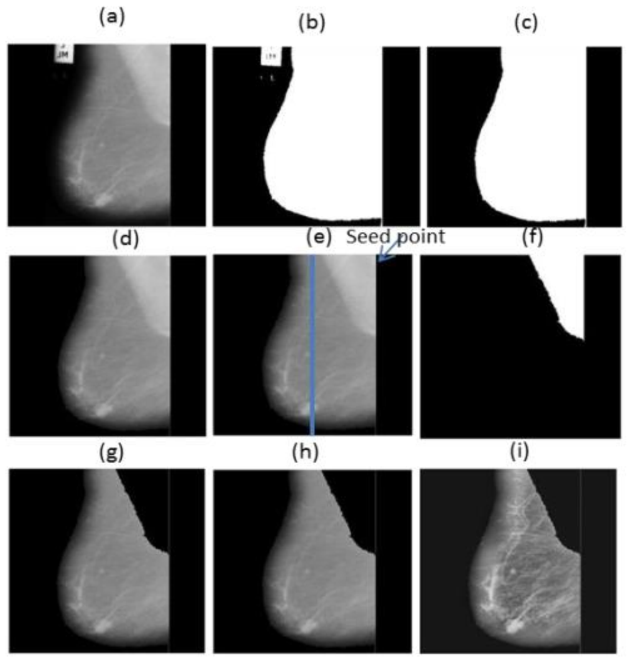


Fig 3 Experimental results proposed method(a) Original Image (b) Binary Image with threshold value 0.1 (c) Breast Part extracted (d) Multiplication of (a) and (c) which consist only breast part without background (e) seed point marked for region growing, (f) pectoral muscle segmented, (g) suppressed from original image, (h) wiener filter, (i) result of CLAHE

The background artefacts are removed by binarization with threshold value 0.1 (fig 3b) and all the connected components are organized in largest to smallest in size to extract the largest blob (fig 3c). Then that blob is multiplied with original image to get original breast profile (fig 3d).

The region growing method used to reduce the pectoral muscle part. The proposed method helps to select the seed point automatically (fig 3e) to remove the pectoral muscle (fig 3f) using modified region growing technique. The conventional selection of seed point is modified based on the orientation of the image. The mini-MIAS dataset consist either left oriented or righted oriented images. Hence, the seed point is either left topmost or right topmost first nonzero pixel. Orientation of the image found by dividing image into half and counting the non-zero pixels if left oriented, left part consist more pixels else right part consist more pixels.

Adaptive Median Filter

We have used adaptive mean filter to remove noise from image. since it is better among all the spatial filters and distinguish fine details from noise.The Adaptive Median Filter performs spatial processing to determine which pixels in an image have been affected by impulse noise. The Adaptive Median Filter classifies pixels as noise by comparing each pixel in the image to its surrounding neighbour pixels.

The size of the neighbourhood is adjustable, as well as the threshold for the comparison. A pixel that is different from a majority of its neighbours, as well as being not structurally aligned with those pixels to which it is similar, is labelled as impulse noise.

These noise pixels are then replaced by the median pixel value of the pixels in the neighbourhoods that have passed the noise labelling test. We are initially converting the image into grayscale image using rgb2gray() function then applying adaptive mean filtering to the resulting image and then converted the image into unsigned integer 8 using unit8() function.

Basic Operation

The application of median filter has been investigated. As an advanced method compared with standard median filtering, the Adaptive Median Filter performs spatial processing to preserve detail and smooth non-impulsive noise. A prime benefit to this adaptive approach to median filtering is that repeated applications of this Adaptive Median Filter do not erode away edges or other small structure in the image.

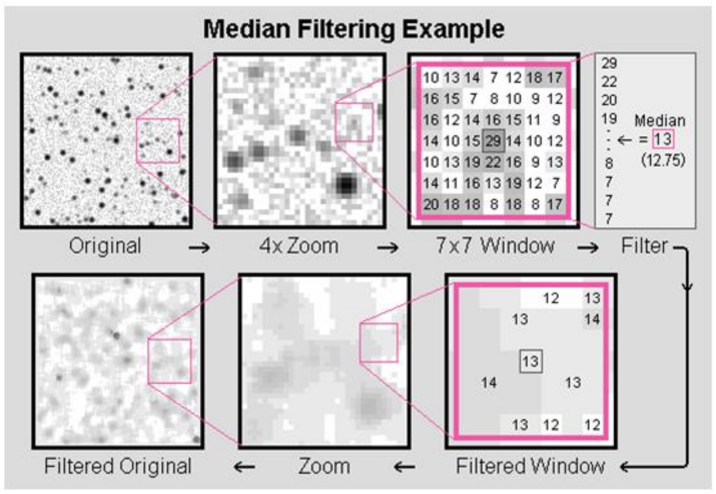
To understand what adaptive median filtering is all about, one first needs to understand what a median filter is and what it does. In many different kinds of digital image processing, the basic operation is as follows: at each pixel in a digital image, we place a neighbourhood around that point, analyse the values of all the pixels in the neighbourhood according to some algorithm, and then replace the original pixel's value with one based on the analysis performed on the pixels in the neighbourhood. The neighbourhood then moves successively over every pixel in the image, repeating the process.

What a median filter is and what it does?

Median filtering follows this basic prescription. The median filter is normally used to reduce noise in an image, somewhat like the mean filter. However, it often does a better job than the mean filter of preserving useful detail in the image. This class of filter belongs to the class of edge preserving smoothing filters which are non-linear filters. This means that for two images *A(x)* and *B(x)*:

$$median[A(x) + B(x)] \neq median[A(x)] + median[B(x)]$$

These filters smooth the data while keeping the small and sharp details. The median is just the middle value of all the values of the pixels in the neighbourhood. Note that this is not the same as the average (or mean); instead, the median has half the values in the neighbourhood larger and half smaller. The median is a stronger "central indicator" than the average. In particular, the median is hardly affected by a small number of discrepant values among the pixels in the neighbourhood. Consequently, median filtering is very effective at removing various kinds of noise. Below example illustrates an example of median filtering.



10	13	14	7	12	18	17
16	15	7	8	10	9	12
16	12	14	16	15	11	9
14	10	15	29	14	10	12
10	13	19	22	16	9	13
14	11	16	13	19	12	7
20	18	18	8	18	8	17

Image Segmentation

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images.

Segmentation is an important stage of medical image processing, because it extracts the objects of our interest, for further processing such as description or recognition. Segmentation of an image is in practice for the classification of image pixels. Segmentation techniques are used to isolate the desired object from the image in order to perform analysis of the object. For example, a tumour, cancer or a block in the blood flow can be easily isolated from its background with the help of image segmentation.

In this project we used machine learning techniques for image segmentation. Now, we can classify machine learning techniques into –

Machine Learning Technique: Clustering

Image segmentation is the classification of an image into different groups. Many kinds of research have been done in the area of image segmentation using clustering. There are different methods and one of the most popular methods is **K-Means clustering algorithm**.

Image segmentation is the process of partitioning a digital image into multiple distinct regions containing each pixel (sets of pixels, also known as super pixels) with similar attributes.

The goal of Image segmentation is to change the representation of an image into something that is more meaningful and easier to analyse.

Image Segmentation involves converting an image into a collection of regions of pixels that are represented by a mask or a labelled image. By dividing an image into segments, you can process only the important segments of the image instead of processing the entire image.

A common technique is to look for abrupt discontinuities in pixel values, which typically indicate edges that define a region.

Another common approach is to detect similarities in the regions of an image. Some techniques that follow this approach are region growing, clustering, and thresholding.

K-Means clustering

K-Means clustering algorithm is an unsupervised algorithm and it is used to segment the interest area from the background. It clusters, or partitions the given data into K-clusters or parts based on the K-centroids.

The algorithm will categorize the items into k groups of similarity. To calculate that similarity, we will use the Euclidean distance as measurement.

The algorithm works as follows:

1. First, we initialize k points, called means, randomly.
2. We categorize each item to its closest mean and we update the mean’s coordinates, which are the averages of the items categorized in that mean so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

$$\text{objective function} \leftarrow J = \sum_{j=1}^{\overset{\text{number of clusters}}{k}} \sum_{i=1}^{\overset{\text{number of cases}}{n}} \underbrace{\left\| \overset{\text{case } i}{x_i^{(j)}} - \overset{\text{centroid for cluster } j}{c_j} \right\|^2}_{\text{Distance function}}$$

The K-means clustering is used as a pre-processing step for the GMM segmentation where the breast images are partitioned into three clusters as the background region, breast tissues and the lumpy part which is the main area of focus.

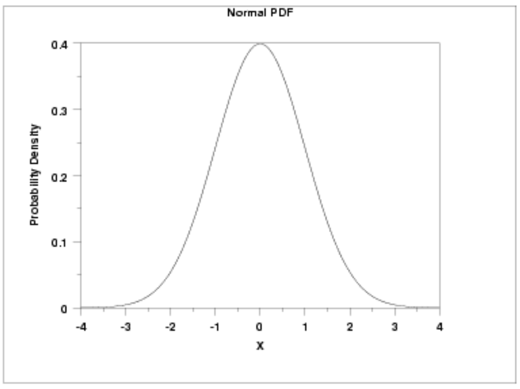
GAUSSIAN MIXTURE MODELS

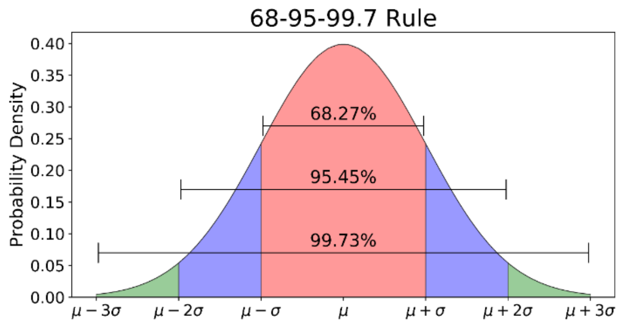
As K-Means clustering is a hard clustering, it will not able to segment image pixels efficiently. To increase the efficiency in the segmentation process, we will use Gaussian Mixture model (GMM). First, we have to know about the Gaussian function.

It is a continuous probability distribution function of random variables.

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

where σ is the standard deviation and μ the mean



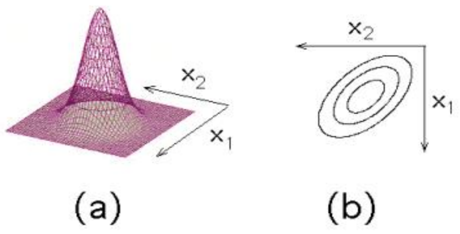


2-D Gaussian function

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Here, x and y both are independent variables.

Its graphical representation



(a) Two–dimensional Gaussian distribution.
(b) Constant probability contour plot representing 2–*D* Gaussian.

A vector-valued random variable X = X1 · · · XnT is said to have a multivariate
normal (or Gaussian) distribution with mean μ ∈ Rn and covariance matrix Σ
if its probability density function2 is given by
p(x; μ, Σ) = 1

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$

We write this as X ∼ N (μ, Σ).

We consider, **Gaussian mixture model** as a simple linear superposition of Gaussian components, aimed at providing a richer class of density models than the single Gaussian. We now turn to a formulation of Gaussian mixtures in terms of discrete *latent* variables.

Gaussian mixture distribution can be written as a linear superposition of Gaussians in the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

Let us introduce a *K*-dimensional binary random variable **z** having a 1-of-*K* representation in
which a particular element *zk* is equal to 1 and all other elements are
equal to 0. The values of *zk* therefore satisfy *zk* ∈{0, 1} and *k zk* = 1, and we
see that there are *K* possible states for the vector **z** according to which element is
nonzero. We shall define the joint distribution *p*(**x**, **z**) in terms of a marginal distribution *p*(**z**)
and a conditional distribution *p*(**x**|**z**), corresponding to the graphical
model in Figure 9.4. The marginal distribution over **z** is specified in terms of the
mixing coefficients *πk*, such that

$p(z_k = 1) = \pi_k$

where the parameters $\{\pi_k\}$ must satisfy

$$0 \leqslant \pi_k \leqslant 1$$

together with

$$\sum_{k=1}^{K} \pi_k = 1$$

in order to be valid probabilities. Because $\mathbf{z}$ uses a 1-of-K representation, we can also write this distribution in the form

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}.$$

Similarly, the conditional distribution of $\mathbf{x}$ given a particular value for $\mathbf{z}$ is a Gaussian

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

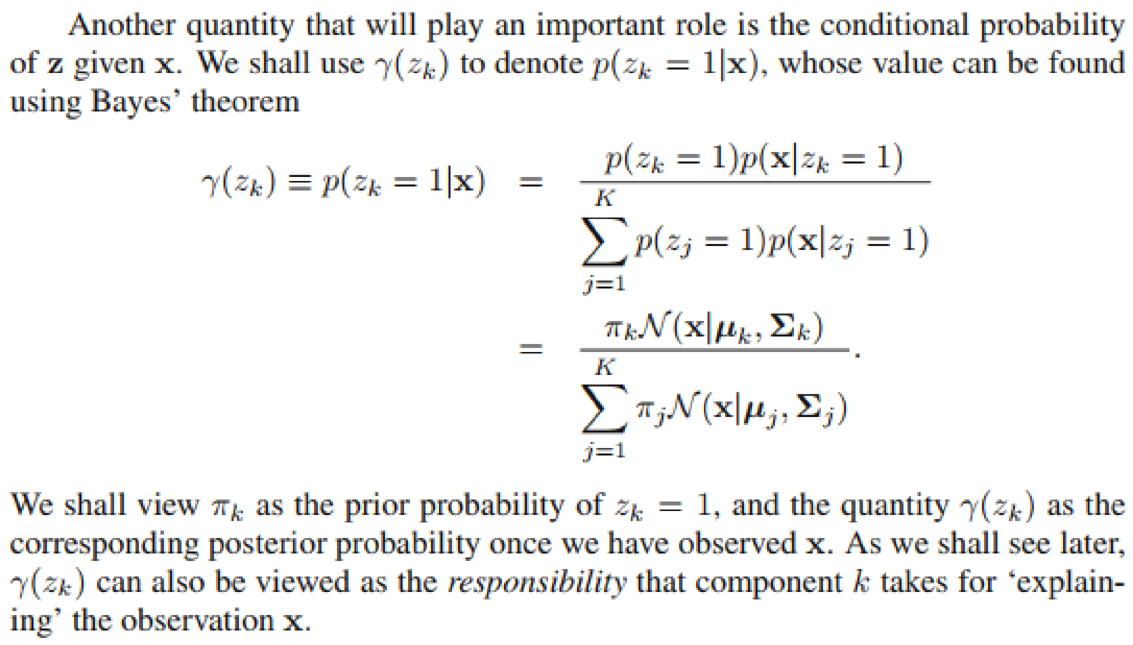
which can also be written in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}.$$

The joint distribution is given by $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, and the marginal distribution of $\mathbf{x}$ is then obtained by summing the joint distribution over all possible states of $\mathbf{z}$ to give

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

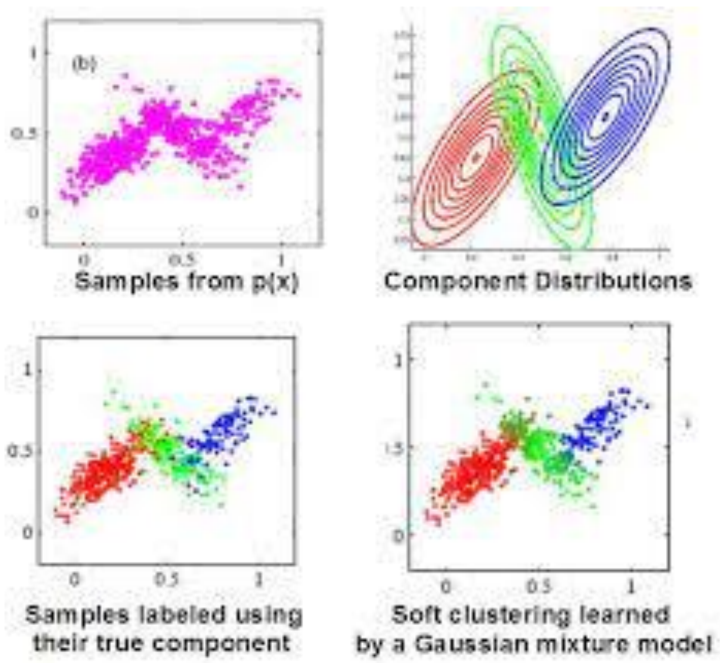
We have therefore found an equivalent formulation of the Gaussian mixture involving an explicit latent variable. It might seem that we have not gained much by doing so. However, we are now able to work with the joint distribution $p(\mathbf{x}, \mathbf{z})$ instead of the marginal distribution $p(\mathbf{x})$, and this will lead to significant simplifications, most notably through the introduction of the expectation-maximization (EM) algorithm.



Another quantity that will play an important role is the conditional probability of $\mathbf{z}$ given $\mathbf{x}$. We shall use $\gamma(z_k)$ to denote $p(z_k = 1|\mathbf{x})$, whose value can be found using Bayes' theorem

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

We shall view π_k as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ as the corresponding posterior probability once we have observed $\mathbf{x}$. As we shall see later, $\gamma(z_k)$ can also be viewed as the *responsibility* that component k takes for 'explaining' the observation $\mathbf{x}$.



Feature Extraction

In image processing feature is a piece of information about the content of an image. And the main idea behind feature extraction is to reduce the number of features in a dataset by creating new features from the existing ones and these new reduced set of features summarizes the important information contained in the original set of features. Feature vectors from masses are assumed to be considered different from normal tissue, and based on a collection of their examples from several subjects, a system can be trained to differentiate between them.
In this project, feature extraction algorithm focuses on texture features based on underlying texture of the region using gray level co-occurrence matrix(GLCM) and two dimensional Discrete wavelet transform(2dDWT).

Discrete wavelet transforms (DWT)

Two-dimensional DWT (2D-DWT) decomposes the mammographic ROI into a number of sub-images preserving the high and low frequency information in different resolution levels and hence leads to extract better texture information from the mammographic ROIs.
Given a continuous, square integrable function f (x), its wavelet transform is calculated as the inner product of f(.) and a real valued wavelet function (ψ (x)) given by,

$$W\left[f\left(s,\tau\right)\right] = \left\langle f, \psi_{s,\tau}^{k} \right\rangle = \int_{-\infty}^{\infty} f\left(x\right) \psi_{s,\tau}^{k}\left(x\right) dx$$

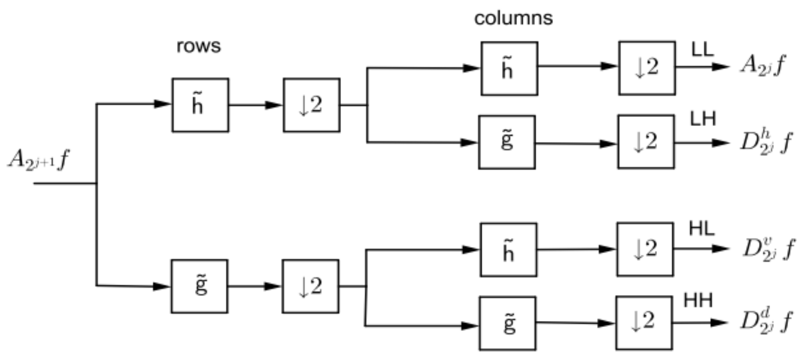
Here, S(scale) ∈ **Z** , T(translation) and k(orientation) ∈**{h,v,d}**

$$\psi_{s,\tau}^{k}\left(x\right) = \frac{1}{\sqrt{s}} \psi^{k}\left(\frac{x-\tau}{s}\right)$$

Here the orientation parameters h, v and d represent to horizontal, vertical and diagonal directions respectively.
Now the dyadic wavelet decomposition is achieved when s = 2^j and τ = 2^j.n, j, n ∈ Z.
A Two-Dimensional Discrete Wavelet Transform is implemented using the combination of digital filter banks and down-samplers. The digital filter banks consist of high-pass (g) and low-pass (h) filters and the number of banks is set as per the desired resolution. As the image is a 2D signal, separable wavelet functions compute the Discrete Wavelet Transform (DWT). The rows and columns of the image are separately undergone through the 1D wavelet transform to establish the 2D-DWT. If the original image A2j+1 f at resolution 2^(j+1) is decomposed into four sub-band images in the frequency domain. Among them, three sub-band images, $D_{2^j}^{h} f,\ D_{2^j}^{v} f,\ D_{2^j}^{d} f$ are the detail images at resolution 2^j in horizontal, vertical, and diagonal directions respectively. The Fourth one is the approximation image, A2j f found at coarse resolution. Then the whole image A2j+1 f is represented as,

$$A_{2^{j+1}} f = D_{2^j}^{h} f + D_{2^j}^{v} f + D_{2^j}^{d} f + A_{2^j} f.$$

The decomposed sub-images are the representation of 2D orthogonal wavelet. Thus, the output of a wavelet decomposition of an image results into four orthogonal sub-band components like Low-Low (LL), Low-High (LH), High-Low (HL) and High-High (HH), that correspond to sub-images $D_{2^j}^{h} f,\ D_{2^j}^{v} f,\ D_{2^j}^{d} f$ and $A_{2^j}\ f$ respectively.

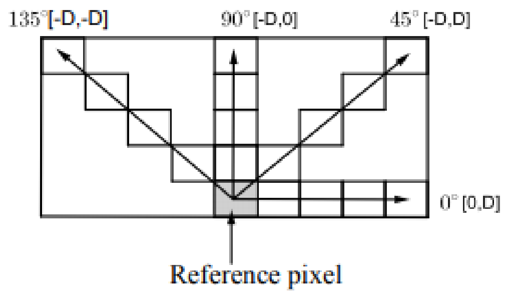


Wavelet decomposition using analysis filter banks filter banks.

Gray level co-occurrence matrix (GLCM)

GLCM is the most classical second-order statistical method for texture analysis. The GLCM is a tabulation of how often different combinations of grey level co-occur in an image section, also referred as co-occurrence distribution. Texture features use the contents of the GLCM to give a measure of the variation in intensity at the point of interest.
Every element q(i, j) in GLCM represents the sum of the frequency of occurrence of that pixel i in a certain association with the pixel j. It can be denoted as q(i, j | d, theta), where d is the distance of separation between the pixel pair i and j. The parameter theta stands for the orientation of the pixel i with respect to the pixel j. GLCM matrix is a square matrix and can be calculated at any angle(direction). This symmetric matrix is normalized to convert it into probabilities. Each element of the normalized GLCM (NGLCM) is represented as:

$$p(i,j) = q(i,j) \Big/ \sum_{i=0}^{G-1}\sum_{j=0}^{G-1} q(i,j).$$



The feature descriptors to be derived from NGLCM for mammographic ROI can be mathematically represented as:

Feature Descriptor	Name	Computation
FD_1	Energy	$\sum_{i=1}^{G}\sum_{j=1}^{G}\{p(i,j)\}^2$
FD_2	Correlation	$\frac{\sum_{i=1}^{G}\sum_{j=1}^{G}(i,j)p(i,j)-\mu_x\mu_y}{\sigma_x\sigma_y}$
FD_3	Entropy	$-\sum_{i=1}^{G}\sum_{j=1}^{G}p(i,j)\log(p(i,j))$
FD_4	Sum variance	$\sum_{i=2}^{2G}(i-sum\,entropy)^2 p_{x+y}(i)$
FD_5	Sum average	$\sum_{i=2}^{2G} i p_{x+y}(i)$

where, μ_x, μ_y, σ_x and σ_y are the means and standard deviations of p_x and p_y, and $sum\,entropy = -\sum_{i=2}^{2G} p_{x+y}(i)\log\{p_{x+y}(i)\}$.

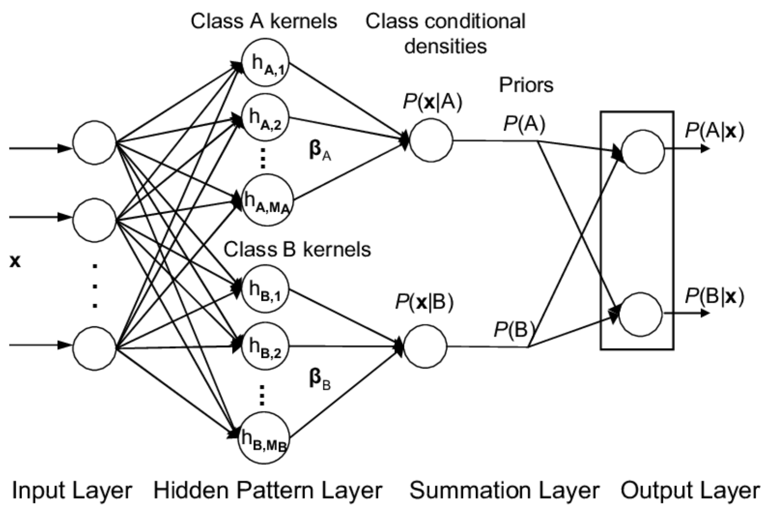
Feature Extraction using 2D-DWT and GLCM

On each of the mammographic ROIs the 2D-DWT is applied to capture the local information. Using the GLCM approach, the statistical measures or feature descriptor values are determined from the sub-bands of each of the blocks. at a set distance D. Then all the feature descriptors (F D) mentioned in Table are computed from each NGLCM and combined to form a feature descriptor matrix (F DM). Thus, a feature matrix is generated by concatenating all the FDMs from all NGLCMs for K number of ROIs.

Classification with Probabilistic Neural Networks

A probabilistic neural network (PNN) is a feedforward neural network, which is widely used in classification and pattern recognition problems. In the PNN algorithm, the parent probability distribution function (PDF) of each class is approximated by a Parzen window and a non-parametric function.

A probabilistic neural network (PNN) has 3 layers of nodes. The figure below displays the architecture for a PNN that recognizes K = 2 classes, but it can be extended to any number K of classes. The input layer (on the left) contains N nodes: one for each of the N input features of a feature vector. These are fan-out nodes that branch at each feature input node to all nodes in the hidden (or middle) layer so that each hidden node receives the complete input feature vector x. The hidden nodes are collected into groups: one group for each of the K classes as shown in the figure.

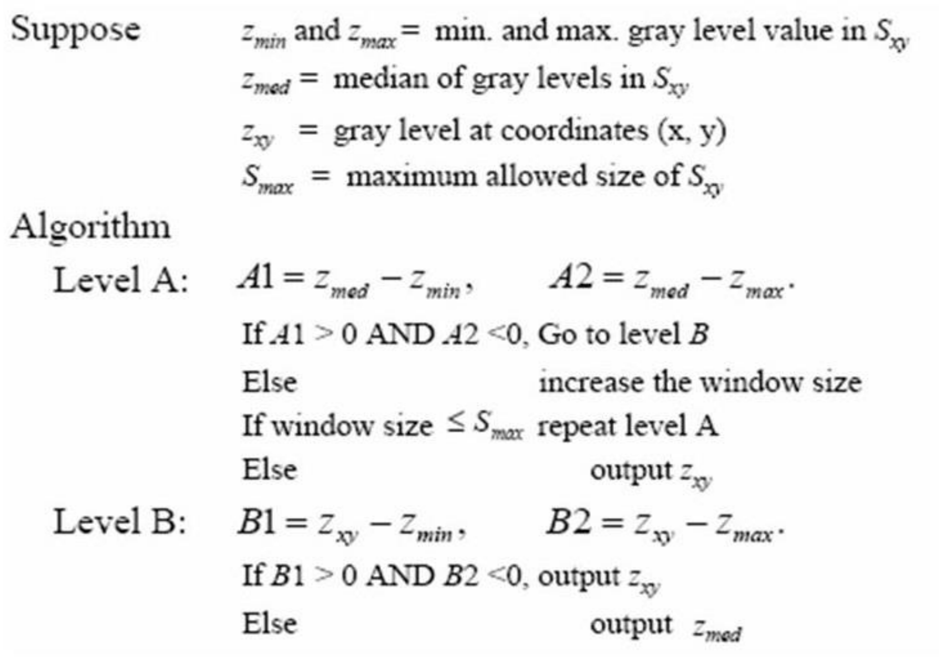


Each hidden node in the group for Class k corresponds to a Gaussian function centered on its associated feature vector in the k class (there is a Gaussian for each exemplar feature vector). th All of the Gaussians in a class group feed their functional values to the same output layer node for that class, so there are K output nodes.

ALGORITHMS

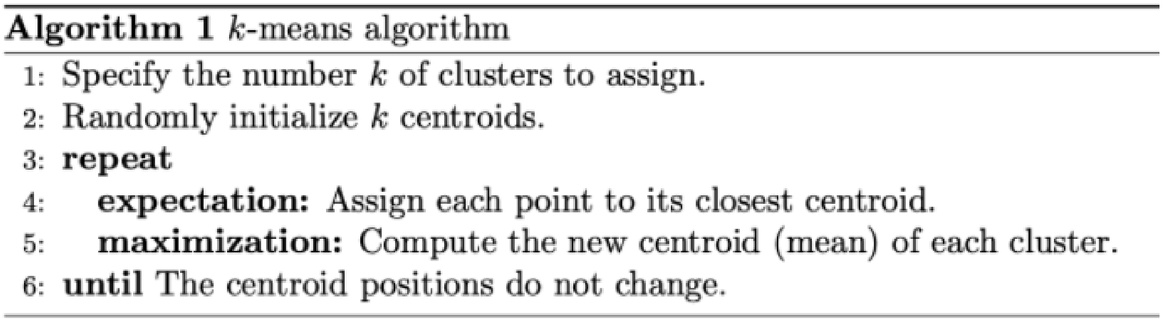
We used several image processing and Machine Learning methods in our projects. Now we will discuss the algorithms used in those techniques.

Adaptive Median Filter



```
Suppose     $z_{min}$ and $z_{max}$ = min. and max. gray level value in $S_{xy}$
            $z_{med}$ = median of gray levels in $S_{xy}$
            $z_{xy}$ = gray level at coordinates (x, y)
            $S_{max}$ = maximum allowed size of $S_{xy}$
Algorithm
  Level A:  $A1 = z_{med} - z_{min}$,        $A2 = z_{med} - z_{max}$.
            If $A1 > 0$ AND $A2 < 0$, Go to level $B$
            Else                      increase the window size
            If window size $\leq S_{max}$ repeat level A
            Else                         output $z_{xy}$
  Level B:  $B1 = z_{xy} - z_{min}$,        $B2 = z_{xy} - z_{max}$.
            If $B1 > 0$ AND $B2 < 0$, output $z_{xy}$
            Else                         output $z_{med}$
```

K-Means Clustering



Algorithm 1 *k*-means algorithm

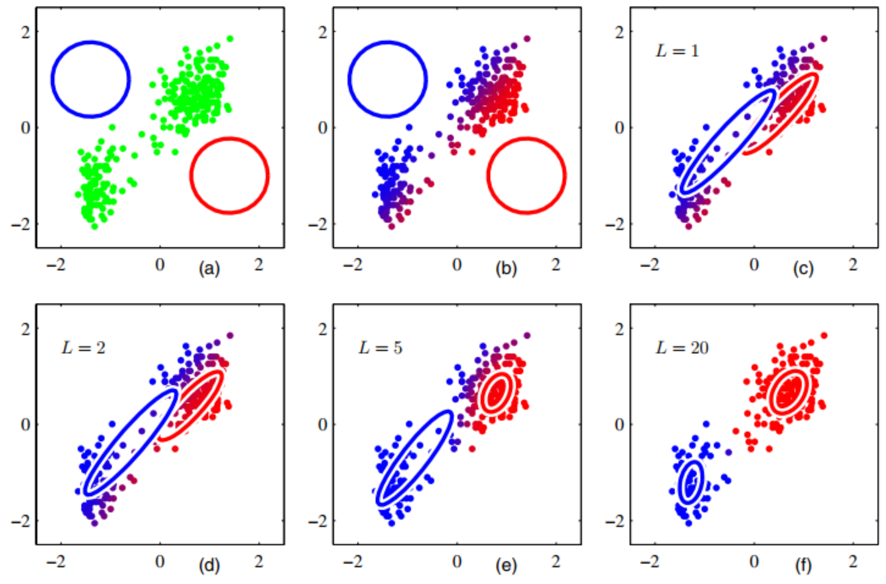
```
1: Specify the number $k$ of clusters to assign.
2: Randomly initialize $k$ centroids.
3: repeat
4:    expectation: Assign each point to its closest centroid.
5:    maximization: Compute the new centroid (mean) of each cluster.
6: until The centroid positions do not change.
```

Expectation- Maximization Algorithm for GMM

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$



Example of EM algorithm

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

where

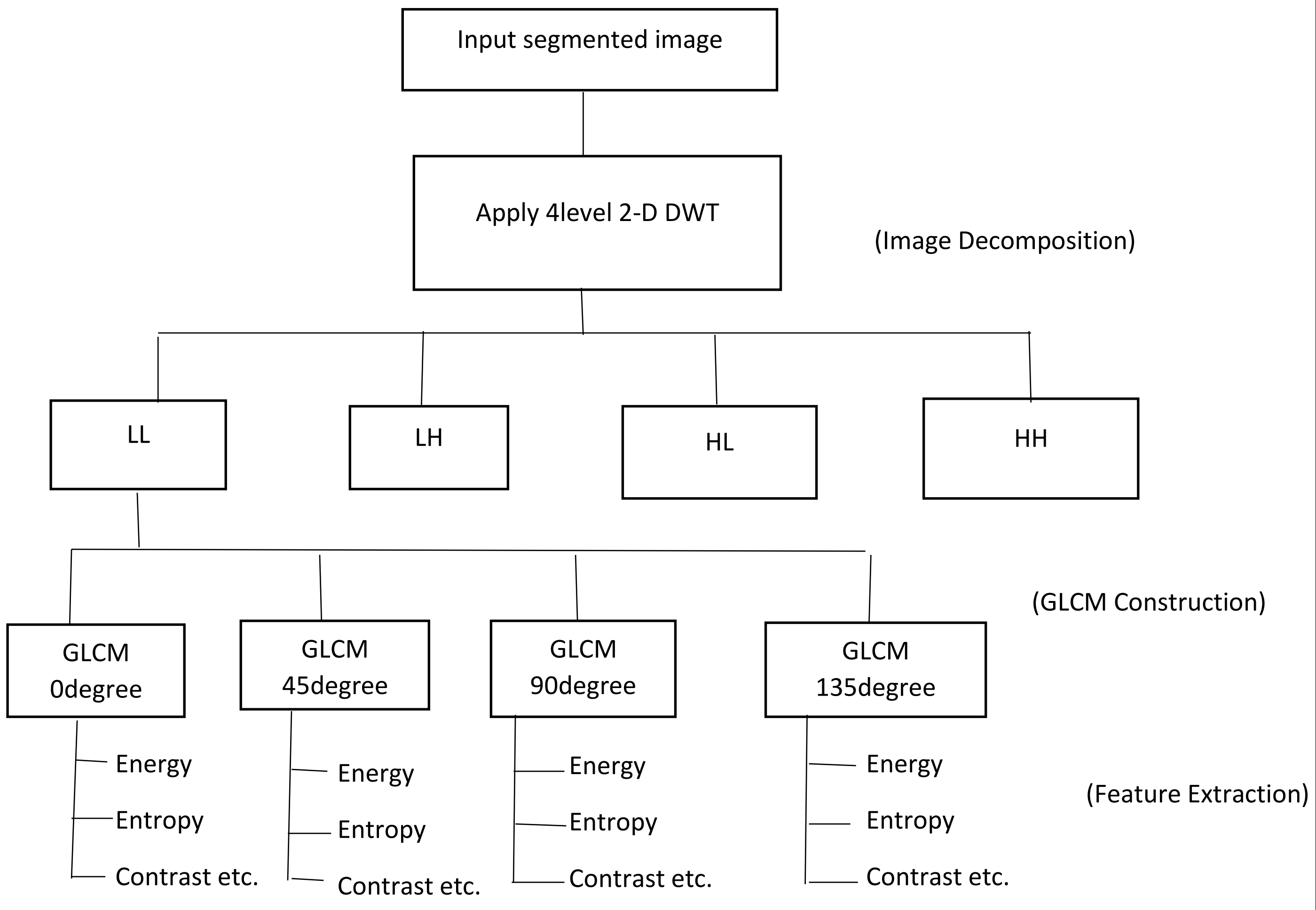
$$N_k = \sum_{n=1}^{N}\gamma(z_{nk}).$$

4. Evaluate the log likelihood

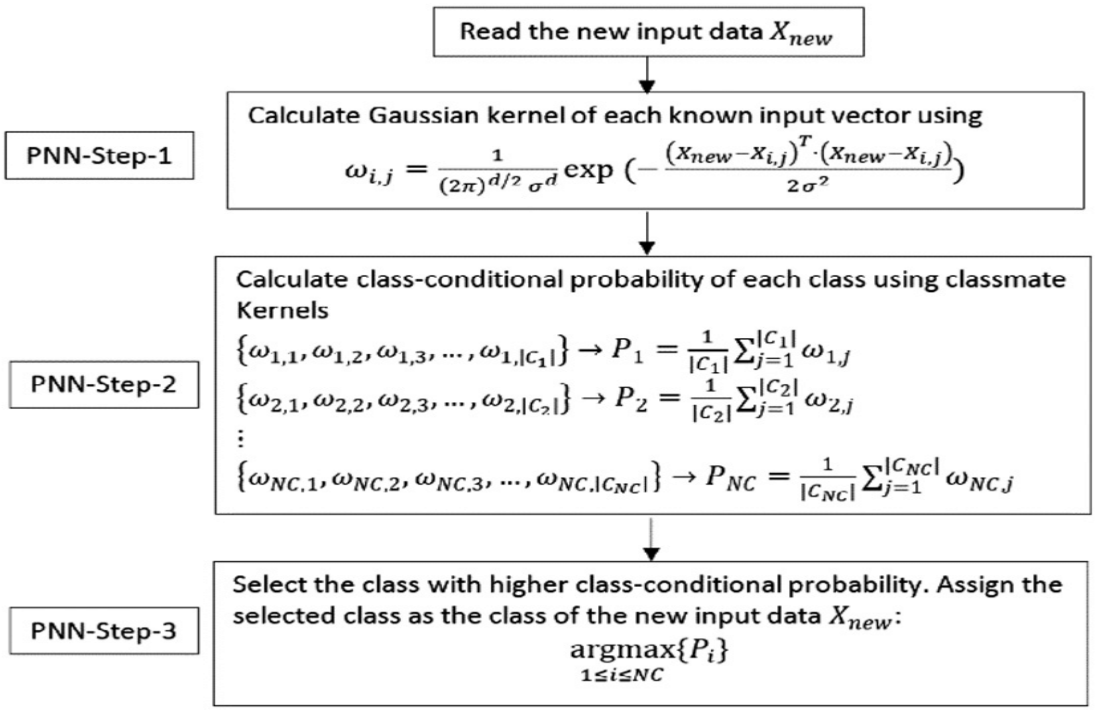
$$\ln p(\mathbf{X}|\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}) = \sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)\right\}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

Feature extraction algorithm



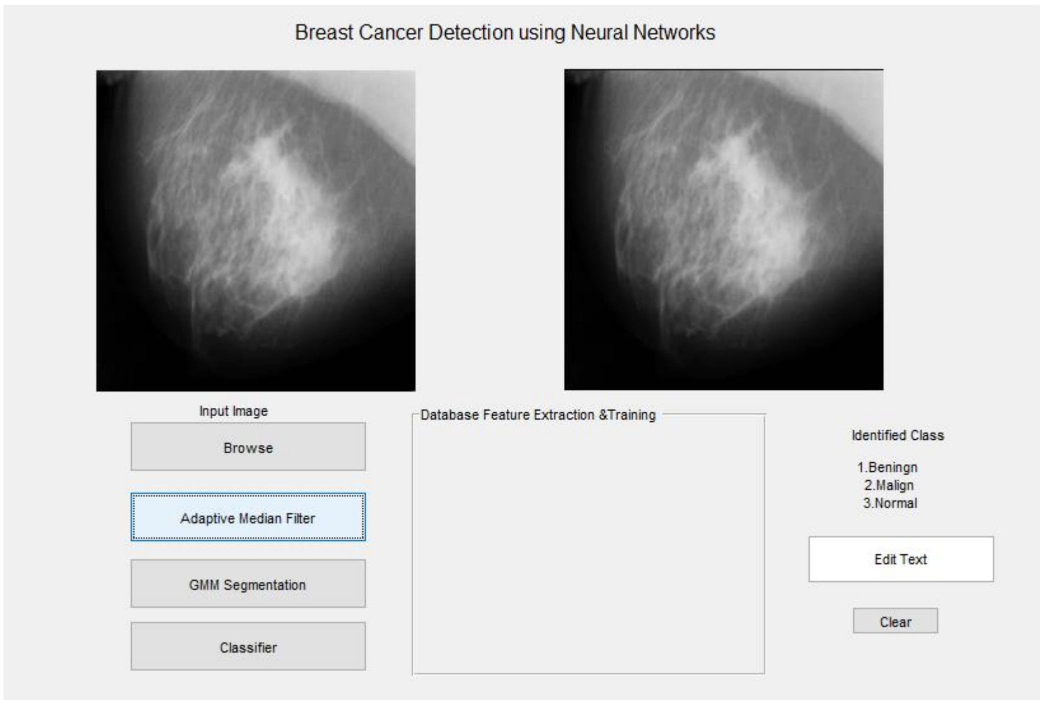
Probabilistic Neural Networks


Read the new input data X_{new}
PNN-Step-1
Calculate Gaussian kernel of each known input vector using
$\omega_{i,j} = \frac{1}{(2\pi)^{d/2}\,\sigma^d} \exp\left(-\frac{(x_{new}-x_{i,j})^T \cdot (x_{new}-x_{i,j})}{2\sigma^2}\right)$
PNN-Step-2
Calculate class-conditional probability of each class using classmate Kernels
$\{\omega_{1,1}, \omega_{1,2}, \omega_{1,3}, \ldots, \omega_{1,|C_1|}\} \to P_1 = \frac{1}{|C_1|}\sum_{j=1}^{|C_1|} \omega_{1,j}$
$\{\omega_{2,1}, \omega_{2,2}, \omega_{2,3}, \ldots, \omega_{2,|C_2|}\} \to P_2 = \frac{1}{|C_2|}\sum_{j=1}^{|C_2|} \omega_{2,j}$
$\vdots$
$\{\omega_{NC,1}, \omega_{NC,2}, \omega_{NC,3}, \ldots, \omega_{NC,|C_{NC}|}\} \to P_{NC} = \frac{1}{|C_{NC}|}\sum_{j=1}^{|C_{NC}|} \omega_{NC,j}$
PNN-Step-3
Select the class with higher class-conditional probability. Assign the selected class as the class of the new input data X_{new}:
$\underset{1 \le i \le NC}{\operatorname{argmax}}\{P_i\}$

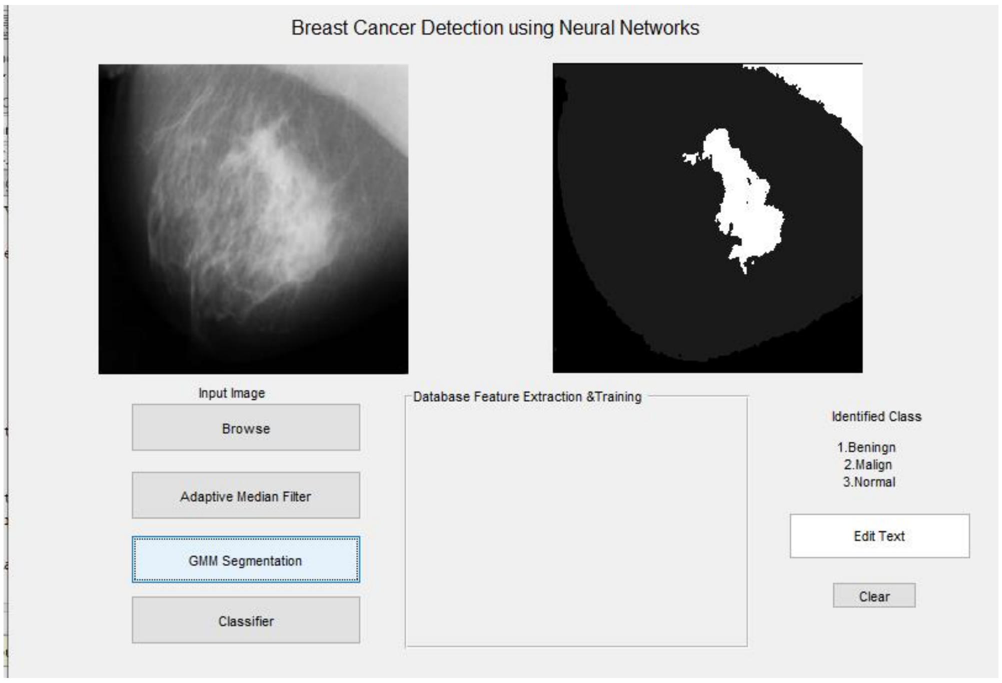
Result and Discussion

Result we got after each step:

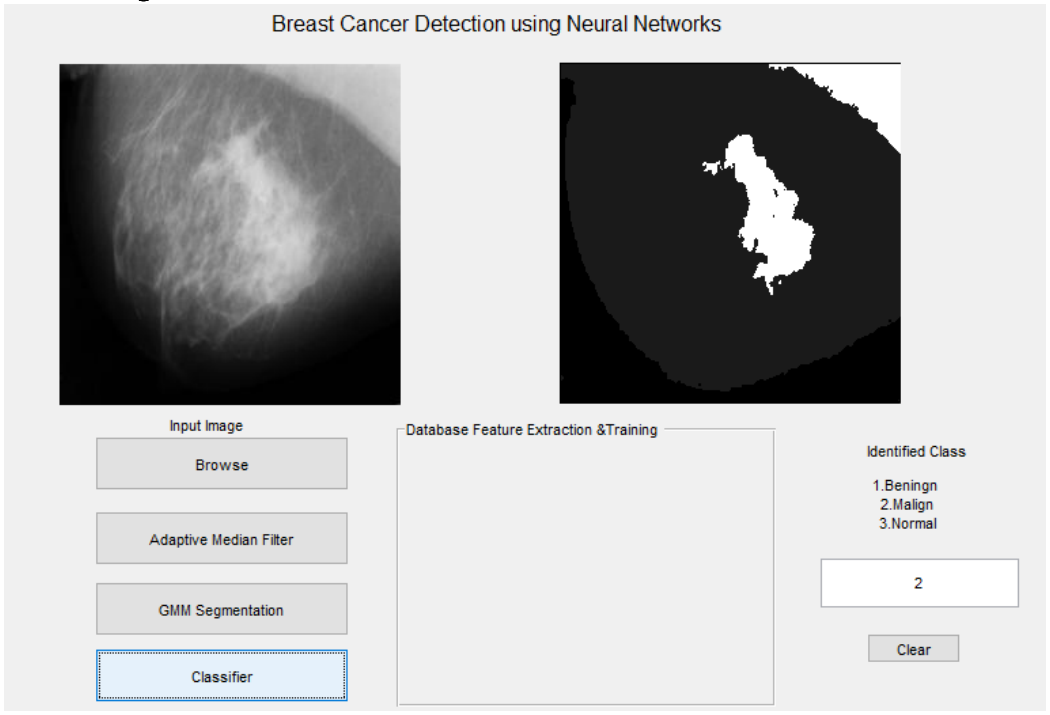
1.After applying adaptive median filter-



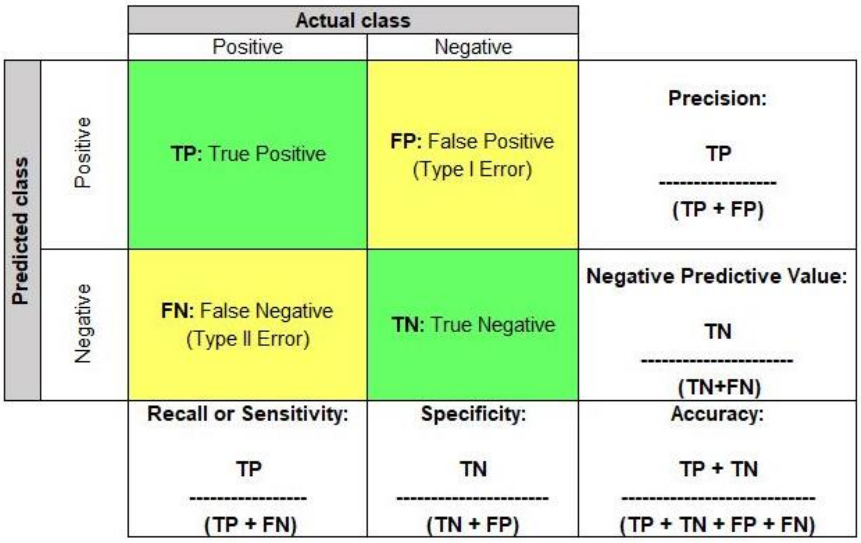
2. After performing GMM segmentation-



4. After doing classification-



Now the performance of the proposed system is evaluated by considering the actual and predicted classification. Accuracy of the system is calculated by using the confusion matrix obtained for the classifier used. Table 2 shows the confusion matrix for a two-class classifier. Classification accuracy, sensitivity, specificity, positive predictive value and negative can be defined by using the elements of the confusion matrix as. Classification accuracy: Accuracy of the classification is obtained by using the given equation:



		Actual class		
		Positive	Negative	
Predicted class	Positive	**TP:** True Positive	**FP:** False Positive (Type I Error)	**Precision:** TP ----------------- (TP + FP)
	Negative	**FN:** False Negative (Type II Error)	**TN:** True Negative	**Negative Predictive Value:** TN ---------------------- (TN+FN)
		Recall or Sensitivity: TP ----------------- (TP + FN)	**Specificity:** TN ---------------------- (TN + FP)	**Accuracy:** TP + TN ---------------------------- (TP + TN + FP + FN)

TP: Correctly classified as having breast cancer

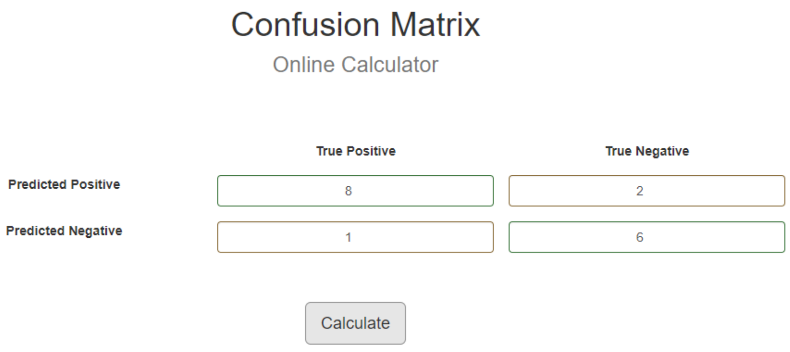
TN: Correctly classified as not having breast cancer.

FP: Classified as having breast cancer but actually they don't have (Error of type I)

FN: Classified as not having breast cancer but actually they have cancer. (Error of type II)

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100$$

This term tells us how many right classifications were made out of all the classifications. In other words, how many TPs and TNs were done out of TP + TN + FP + FNs. It tells the ratio of "True"s to the sum of "True"s and "False"s.



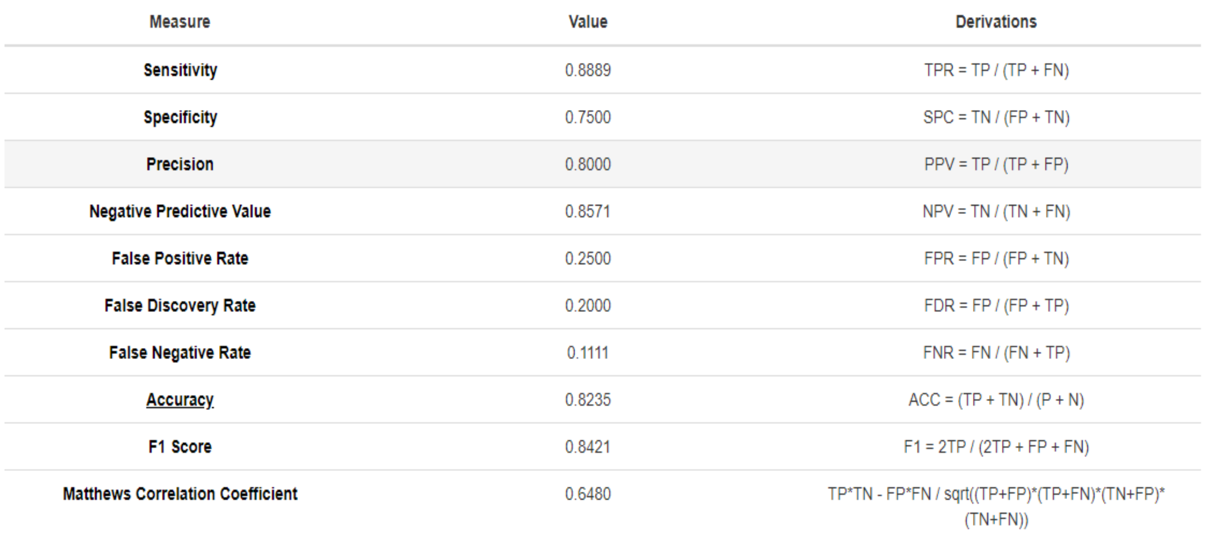
Confusion Matrix

Online Calculator

	True Positive	True Negative
Predicted Positive	8	2
Predicted Negative	1	6

We used online calculator for calculation of confusion matrix.

Our model performance:



Measure	Value	Derivations
Sensitivity	0.8889	TPR = TP / (TP + FN)
Specificity	0.7500	SPC = TN / (FP + TN)
Precision	0.8000	PPV = TP / (TP + FP)
Negative Predictive Value	0.8571	NPV = TN / (TN + FN)
False Positive Rate	0.2500	FPR = FP / (FP + TN)
False Discovery Rate	0.2000	FDR = FP / (FP + TP)
False Negative Rate	0.1111	FNR = FN / (FN + TP)
Accuracy	0.8235	ACC = (TP + TN) / (P + N)
F1 Score	0.8421	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.6480	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Conclusion and Future Scopes

In this project, we did detection and classification along by segmentation and feature extraction of breast mammogram images. We built machine learning model and incorporate image processing techniques with it. We used 60 images for training our model and 17 images for testing. The accuracy calculated from confusion matrix is 82.35 % with F- score

84.21 %. The reason for low accuracy is less training and test data. Due to time constraint and less computing power, other resources we were not able to use large dataset. That is why our model got low accuracy.

In Future, we want to use more data and train our model with Convolutional Neural Networks (CNN) instead of PNN by incorporating new technology Deep Learning in our project.

Also, we can use more sophisticated image processing techniques for pre-processing and feature extraction.

References and Bibliography

1. Digital Image Processing Using MATLAB by Gonzalez and Woods , 2nd edition
2. Bishop - Pattern Recognition And Machine Learning - Springer 2006
3. PATTERN. CLASSIFICATION. Second Edition. Richard O. Duda. Peter E. Hart.
David G. Stork. A Wiley-Interscience Publication. JOHN WILEY & SONS, INC.
4. Mammograms Classification using Gray-level Co-occurrence Matrix and Radial Basis
Function Neural Network Mellisa Pratiwia , Alexandera , Jeklin Harefaa , Sakka
Nandaa a Industrial Engineering Department, Faculty of Engineering, Bina Nusantara
University Jl KH Syahdan 9, Jakarta 11480, Indonesia, mpratiwi@binus.edu
5. Mammogram classification method based on GMM and GLCM-PSO-PNN by
Xiaojian Zhanga, Chengjian Weib and Xili Wanc
School of computer science and technology, Nanjing Tech University, Nanjing
211816, China
6. Khoulqi, Ichrak & Idrissi, Najlae & Sarfraz, Muhammad. (2020). Segmentation of
Pectoral Muscle in Mammogram Images Using Gaussian Mixture Model-Expectation
Maximization. 10.4018/978-1-7998-4444-0.ch009.

7. A GLCM based Feature Extraction in Mammogram
Images using Machine Learning Algorithms
BN Jagadesh, L Kanya Kumari
Professor, Department of Computer Science & Engineering, Srinivasa Institute of Engineering and Technology, Amalapuram, Andhra
Pradesh, India; 2Assistant Professor, Department of Information Technology, Andhra Loyola Institute of Engineering and Technology,
Vijayawada, Andhra Pradesh, India.
8. Abirami C, Harikumar R, Chakravarthy SRS. Performance analysis and detection of microcalcification in digital mammograms
using wavelet features. International Conference on Wireless
Communications, Signal Processing and Networking.2016;
2327-2331.
9. Nagarajan V, Britto EC, Veeraputhiran SM. Feature extraction
based on empirical mode decomposition for automatic mass
classification of mammogram images. Med Novel Tech Devic
2019;1:100004
10. Rangayyan M, El-Faramawy NM, Desautels JEL Alim OA.
Measures of acutance and shape for classification of breast tumours. IEEE
Transactions Med Imaging.1997;16:799-810.
11. Damiati S, Peacock M, Mhanna R, Sopstad S, Sleytr UB, Schuster B. Bioinspired detection sensor based on functional nanostructures of S-proteins to target the folate receptors in breast
cancer cells. Sens Actuators B Chem. 2018;267:224-230.
12. Margolies LR, Salvatore M, Yip R, Tam K, Bertolini A, Henschke C, et al. The chest radiologist's role in invasive breast cancer
detection. Clin Imaging 2018;50:13–19.
13. Liu J, Shi Y. Image Feature Extraction Method Based on Shape
Characteristics and Its Application in Medical Image Analysis.
Appl Inform Commun Comp Inform Sci 2011; 224:172-178.
14. Pradeep S, Malliga L. Content-based image retrieval and segmentation of medical image database with fuzzy values. International Conference on Information Communication and Embedded Systems. 2014;1-7.
15. Debelee TG, Gebreselasie A, Schwenker F, Amirian M, Yohannes D. Classification of mammograms using texture and
CNN based extracted features. J Biomim Biomater Biomed Eng
2019;42:79–97.
16. Sonar U. Bhosle F, Choudhury C. Mammography classification
using modified hybrid SVM-KNN. In: International Conference on Signal Processing and Communication (ICSPC), Coimbatore, pp. 305-311(2017).
17. https://wiki.cancerimagingarchive.net/display/Public/CPTAC-HNSCC
18. https://onlineconfusionmatrix.com/

Code:

```
function varargout = guidemo(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @guidemo_OpeningFcn, ...
                   'gui_OutputFcn',  @guidemo_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before guidemo is made visible.
function guidemo_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to guidemo (see VARARGIN)

% Choose default command line output for guidemo
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guidemo wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = guidemo_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Browse.
function Browse_Callback(hObject, eventdata, handles)
% hObject    handle to Browse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


    [filename, pathname] = uigetfile('*.jpg', 'Pick a Image');
    if isequal(filename,0) || isequal(pathname,0)
       warndlg('User pressed cancel')
    else
    filename=strcat(pathname,filename);

    InputImage=imread(filename);

    axes(handles.axes1);
    imshow(InputImage);

    handles.InputImage=InputImage;
    end
    % Update handles structure
guidata(hObject, handles);


% --- Executes on button press in AdaptiveMedianFilter.
function AdaptiveMedianFilter_Callback(hObject, eventdata, handles)

        InputImage=handles.InputImage;
        GrayScaleImage=rgb2gray(InputImage);

        NoisyImage=GrayScaleImage;
        NoisyImage=double(GrayScaleImage);
        [R C P]=size(NoisyImage);
        OutImage=zeros(R,C);
        Zmin=[];
        Zmax=[];
        Zmed=[];


        for i=1:R

            for j=1:C
                       if (i==1 & j==1)


                  % for right top corner[8,7,6]
                        elseif (i==1 & j==C)


                    % for bottom left corner[2,3,4]
                        elseif (i==R & j==1)


                             % for bottom right corner[8,1,2]

                        elseif (i==R & j==C)


                            %for top edge[8,7,6,5,4]

                        elseif (i==1)
```

```
                                                    % for right
edge[2,1,8,7,6]

                        elseif (i==R)

                                % // for bottom edge[8,1,2,3,4]

                        elseif (j==C)

                        elseif (j==1)

                       else


                                     SR1 = NoisyImage((i-1),(j-1));
                                     SR2 = NoisyImage((i-1),(j));
                                     SR3 = NoisyImage((i-1),(j+1));
                                     SR4 = NoisyImage((i),(j-1));
                                     SR5 = NoisyImage(i,j);
                                     SR6 = NoisyImage((i),(j+1));
                                     SR7 = NoisyImage((i+1),(j-1));
                                     SR8 = NoisyImage((i+1),(j));
                                     SR9 = NoisyImage((i+1)),((j+1));

TempPixel=[SR1,SR2,SR3,SR4,SR5,SR6,SR7,SR8,SR9];
                                     Zxy=NoisyImage(i,j);
                                     Zmin=min(TempPixel);
                                     Zmax=max(TempPixel);
                                     Zmed=median(TempPixel);
                                     A1 = Zmed - Zmin;
                                     A2 = Zmed - Zmax;

                                     if A1 > 0 && A2 < 0

                                          %   go to level B
                                          B1 = Zxy - Zmin;
                                          B2 = Zxy - Zmax;
                                          if B1 > 0 && B2 < 0
                                              PreProcessedImage(i,j)= Zxy;
                                          else
                                              PreProcessedImage(i,j)= Zmed;

                                          end
                                     else

                                         if ((R > 4 && R < R-5) && (C > 4
&& C < C-5))


                                         S1 = NoisyImage((i-1),(j-1));
                                         S2 = NoisyImage((i-2),(j-2));
                                         S3 = NoisyImage((i-1),(j));
                                         S4 = NoisyImage((i-2),(j));
                                         S5 = NoisyImage((i-1),(j+1));
                                         S6 = NoisyImage((i-2),(j+2));
                                         S7 = NoisyImage((i),(j-1));
                                         S8 = NoisyImage((i),(j-2));
```

```
                                          S9 = NoisyImage(i,j);
                                          S10 = NoisyImage((i),(j+1));
                                          S11 = NoisyImage((i),(j+2));
                                          S12 = NoisyImage((i+1),(j-1));
                                          S13 = NoisyImage((i+2),(j-2));
                                          S14 = NoisyImage((i+1),(j));
                                          S15 = NoisyImage((i+2),(j));
                                          S16 = NoisyImage((i+1)),((j+1));
                                          S17 = NoisyImage((i+2)),((j+2));

TempPixel2=[S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12,S13,S14,S15,S16,S17];
                                          Zmed2=median(TempPixel2);
                                          PreProcessedImage(i,j)= Zmed2;
                                          else

                                          PreProcessedImage(i,j)= Zmed;

                                          end

                                      end

                  end

            end
        end




        PreProcessedImage3=[]
        PreProcessedImage3(:,:,1)=PreProcessedImage;
        PreProcessedImage3(:,:,2)=PreProcessedImage;
        PreProcessedImage3(:,:,3)=PreProcessedImage;

        PreProcessedImage=PreProcessedImage3;
        PreProcessedImage=uint8(PreProcessedImage);
        axes(handles.axes2);
        imshow(PreProcessedImage,[]);
        handles.PreProcessedImage=PreProcessedImage;

    % Update handles structure
guidata(hObject, handles);

warndlg('Process completed');





% --- Executes on button press in GMMSegmentation.
function GMMSegmentation_Callback(hObject, eventdata, handles)
        PreProcessedImage=  handles.PreProcessedImage;


        Y=double(PreProcessedImage);
        k=2; % k: number of regions

        g=2; % g: number of GMM components

        beta=1; % beta: unitary vs. pairwise

        EM_iter=10; % max num of iterations

        MAP_iter=10; % max num of iterations

       % fprintf('Performing k-means segmentation\n');

        [X,GMM,ShapeTexture]=image_kmeans(Y,k,g);
        [X,Y,GMM]=HMRF_EM(X,Y,GMM,k,g,EM_iter,MAP_iter,beta);

        Y=Y*80;

        Y=uint8(Y);
 %OutImage=Y;



        Y=rgb2gray(Y);
    Y=double(Y);

    statsa = glcm(Y,0,ShapeTexture);
    ExtractedFeatures1=statsa;
    axes(handles.axes2);

    imshow(Y,[]);
Y=uint8(Y);

    handles.ExtractedFeatures=ExtractedFeatures1;
    disp('exit');
    handles.gmm=1;

    % Update handles structure
    guidata(hObject, handles);
warndlg('Process completed');

% --- Executes on button press in Classifier.
function Classifier_Callback(hObject, eventdata, handles)
% hObject    handle to Classifier (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
gmm=0;
gmm=handles.gmm;


load ExtractedFeatures

A=1:20;
B=21:40;
C=41:60;

P = [A B C];
```

```
Tc = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3];

k=2; % k: number of regions
g=2; % g: number of GMM components

beta=1; % beta: unitary vs. pairwise
EM_iter=10; % max num of iterations
MAP_iter=10; % max num of iterations



 file=handles.InputImage;

%    [filename, pathname] = uigetfile('*.jpg', 'Pick a MATLAB code file');
%diff=[];
%     file=imread(filename);
    file=rgb2gray(file);
    file=adaptivemedian(file);
    [Xk,GMMk,ShapeTexture]=image_kmeans(file,k,g);
    PreProcessedImage(:,:,1)=file;
    PreProcessedImage(:,:,2)=file;
    PreProcessedImage(:,:,3)=file;


    stats=
gmmsegmentation(Xk,PreProcessedImage,GMMk,k,g,beta,EM_iter,MAP_iter,ShapeTe
xture);

    ShapeTexture=stats.ShapeTexture;

    for i=1:60

         statsa=ExtractedFeature{i};
         ShapeTexturea=statsa.ShapeTexture;


         diff1(i)=corr2(stats.autoc,statsa.autoc);
         diff2(i)=corr2(stats.contr,statsa.contr);
         diff3(i)=corr2(stats.corrm,statsa.corrm);
         diff4(i)=corr2(stats.cprom,statsa.cprom);
         diff5(i)=corr2(stats.cshad,statsa.cshad);
         diff6(i)=corr2(stats.dissi,statsa.dissi);
         diff7(i)=corr2(stats.energ,statsa.energ);
         diff8(i)=corr2(stats.entro,statsa.entro);
         diff9(i)=corr2(stats.homom,statsa.homom);
         diff10(i)=corr2(stats.homop,statsa.homop);
         diff11(i)=corr2(stats.maxpr,statsa.maxpr);
         diff12(i)=corr2(stats.sosvh,statsa.sosvh);
         diff13(i)=corr2(stats.savgh,statsa.savgh);
         diff14(i)=corr2(stats.svarh,statsa.svarh);
         diff15(i)=corr2(stats.senth,statsa.senth);
         diff16(i)=corr2(stats.dvarh,statsa.dvarh);
         diff17(i)=corr2(stats.denth,statsa.denth);
         diff18(i)=corr2(stats.inf1h,statsa.inf1h);
         diff19(i)=corr2(stats.inf2h,statsa.inf2h);
         diff19(i)=corr2(stats.indnc,statsa.indnc);
         diff19(i)=corr2(stats.idmnc,statsa.idmnc);
         diff20(i)=corr2(ShapeTexture,ShapeTexturea);
    end

    [val1 index1]=max(diff1);
    [val2 index2]=max(diff2);
    [val3 index3]=max(diff3);
    [val4 index4]=max(diff4);
    [val5 index5]=max(diff5);
    [val6 index6]=max(diff6);
    [val7 index7]=max(diff7);
    [val8 index8]=max(diff8);
    [val9 index9]=max(diff9);
    [val10 index10]=max(diff10);
    [val11 index11]=max(diff11);
    [val12 index12]=max(diff12);
    [val13 index13]=max(diff13);
    [val14 index14]=max(diff14);
    [val15 index15]=max(diff15);
    [val16 index16]=max(diff16);
    [val17 index17]=max(diff17);
    [val18 index18]=max(diff18);
    [val19 index19]=max(diff19);
    [val20 index20]=max(diff20);


T = ind2vec(Tc);

spread = 1;

net = newpnn(P,T,spread);

A = sim(net,P);
Ac = vec2ind(A);
p1(1) = index20;
p1(2) = index1;
p1(3) = index2;
p1(4) = index3;
p1(5) = index4;
p1(6) = index5;
p1(7) = index6;
p1(8) = index7;
p1(9) = index8;
p1(10) = index9;
p1(11) = index10;
p1(12) = index11;
p1(13) = index12;
p1(14) = index13;
p1(15) = index14;
p1(16) = index15;
p1(17)= index16;
p1(18) = index17;
p1(19) = index18;
p1(20) = index19;
```

```
% pl = index20;
a = sim(net,pl);
ac = vec2ind(a);
disp(ac);
ac=num2str(ac)

set(handles.edit1,'String',ac);

warndlg('Process completed');

% --- Executes on button press in loaddatabase.
function loaddatabase_Callback(hObject, eventdata, handles)
    clc;
    ;
%     [filename, pathname] = uigetfile('*.jpg', 'Pick a MATLAB code file');
% load diff;
    k=2; % k: number of regions
    g=2; % g: number of GMM components

    beta=1; % beta: unitary vs. pairwise
    EM_iter=10; % max num of iterations
    MAP_iter=10; % max num of iterations


    helpdlg('In case of error please rerun the same program on system with
8gb ram to avoid empty clusters');

    len=1;
    len1=21;
    len2=41;
             h = waitbar(0,'Please wait...');

for num=1:20

waitbar(num/20,h)

    filename1=strcat('Beningn',num2str(num),'.jpg');
    filename2=strcat('Malign',num2str(num),'.jpg');
    filename3=strcat('Malign',num2str(num),'.jpg');
    a=imread(filename1);
    b=imread(filename2);
    c=imread(filename3);

    a=rgb2gray(a);
    b=rgb2gray(b);
    c=rgb2gray(c);
    a=adaptivemedian(a);

    b=adaptivemedian(b);
    c=adaptivemedian(c);



   [Xka,GMMka,ShapeTexturea]=image_kmeans(a,k,g);
   [Xkb,GMMkb,ShapeTextureb]=image_kmeans(b,k,g);
   [Xkc,GMMkc,ShapeTexturec]=image_kmeans(c,k,g);

   PreProcessedImagea(:,:,1)=a;
```

```
    PreProcessedImagea(:,:,2)=a;
    PreProcessedImagea(:,:,3)=a;

    PreProcessedImageb(:,:,1)=b;
    PreProcessedImageb(:,:,2)=b;
    PreProcessedImageb(:,:,3)=b;

    PreProcessedImagec(:,:,1)=c;
    PreProcessedImagec(:,:,2)=c;
    PreProcessedImagec(:,:,3)=c;


    statsa=
gmmsegmentation(Xka,PreProcessedImagea,GMMka,k,g,beta,EM_iter,MAP_iter,Shap
eTexturea);
    statsb=
gmmsegmentation(Xkb,PreProcessedImageb,GMMkb,k,g,beta,EM_iter,MAP_iter,Shap
eTextureb);
    statsc=
gmmsegmentation(Xkc,PreProcessedImagec,GMMkc,k,g,beta,EM_iter,MAP_iter,Shap
eTexturec);

     diff{len}=statsa;
     diff{len1}=statsb;
     diff{len2}=statsc;

    len=len+1;
    len1=len1+1;
    len2=len2+1;

end
save extractedfeatures diff
close(h);

[val index]=max(diff);

disp('exit');

warndlg('Process completed');

% --- Executes on button press in TrainPNN.
function TrainPNN_Callback(hObject, eventdata, handles)
A=1:20;
B=21:40;
C=41:60;

P = [A B C];
Tc = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3];

T = ind2vec(Tc);
spread = 1;
net = newpnn(P,T,spread);

warndlg('Training Completed Sucessfully');


function edit1_Callback(hObject, eventdata, handles)
```

```
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=ones(256,256);
axes(handles.axes1);
imshow(a);
axes(handles.axes2);
imshow(a);
a='0'
clear;
set(handles.edit1,'String',a);

function [X GMM ShapeTexture]=image_kmeans(Y,k,g)
[m n temp]=size(Y);

if temp==3
   b=rgb2gray(Y);
   ShapeTexture=wlt4(b);

elseif temp==1

      ShapeTexture=wlt4(Y);
        Y1(:,:,1)=Y;
        Y1(:,:,2)=Y;
        Y1(:,:,3)=Y;
Y=Y1;
end

y=reshape(Y,[m*n 3]);
x=kmeans(y,k);
X=reshape(x,[m n]);

GMM=get_GMM(X,Y,g);
function [out] = GLCM_Features1(glcmin,pairs)

if ((nargin > 2) || (nargin == 0))
   error('Too many or too few input arguments. Enter GLCM and pairs.');
elseif ( (nargin == 2) )
    if ((size(glcmin,1) <= 1) || (size(glcmin,2) <= 1))
       error('The GLCM should be a 2-D or 3-D matrix.');
    elseif ( size(glcmin,1) ~= size(glcmin,2) )
        error('Each GLCM should be square with NumLevels rows and NumLevels
cols');
    end
elseif (nargin == 1) % only GLCM is entered
    pairs = 0; % default is numbers and input 1 for percentage
    if ((size(glcmin,1) <= 1) || (size(glcmin,2) <= 1))
       error('The GLCM should be a 2-D or 3-D matrix.');
    elseif ( size(glcmin,1) ~= size(glcmin,2) )
```

```
        error('Each GLCM should be square with NumLevels rows and NumLevels
cols');
    end
end


format long e
if (pairs == 1)
    newn = 1;
    for nglcm = 1:2:size(glcmin,3)
        glcm(:,:,newn)  = glcmin(:,:,nglcm) + glcmin(:,:,nglcm+1);
        newn = newn + 1;
    end
elseif (pairs == 0)
    glcm = glcmin;
end

size_glcm_1 = size(glcm,1);
size_glcm_2 = size(glcm,2);
size_glcm_3 = size(glcm,3);

% checked
out.autoc = zeros(1,size_glcm_3); % Autocorrelation: [2]
out.contr = zeros(1,size_glcm_3); % Contrast: matlab/[1,2]
out.corrm = zeros(1,size_glcm_3); % Correlation: matlab
out.corrp = zeros(1,size_glcm_3); % Correlation: [1,2]
out.cprom = zeros(1,size_glcm_3); % Cluster Prominence: [2]
out.cshad = zeros(1,size_glcm_3); % Cluster Shade: [2]
out.dissi = zeros(1,size_glcm_3); % Dissimilarity: [2]
out.energ = zeros(1,size_glcm_3); % Energy: matlab / [1,2]
out.entro = zeros(1,size_glcm_3); % Entropy: [2]
out.homom = zeros(1,size_glcm_3); % Homogeneity: matlab
out.homop = zeros(1,size_glcm_3); % Homogeneity: [2]
out.maxpr = zeros(1,size_glcm_3); % Maximum probability: [2]

out.sosvh = zeros(1,size_glcm_3); % Sum of sqaures: Variance [1]
out.savgh = zeros(1,size_glcm_3); % Sum average [1]
out.svarh = zeros(1,size_glcm_3); % Sum variance [1]
out.senth = zeros(1,size_glcm_3); % Sum entropy [1]
out.dvarh = zeros(1,size_glcm_3); % Difference variance [4]
%out.dvarh2 = zeros(1,size_glcm_3); % Difference variance [1]
out.denth = zeros(1,size_glcm_3); % Difference entropy [1]
out.inf1h = zeros(1,size_glcm_3); % Information measure of correlation1 [1]
out.inf2h = zeros(1,size_glcm_3); % Informaiton measure of correlation2 [1]
%out.mxcch = zeros(1,size_glcm_3);% maximal correlation coefficient [1]
%out.invdc = zeros(1,size_glcm_3);% Inverse difference (INV) is homom [3]
out.indnc = zeros(1,size_glcm_3); % Inverse difference normalized (INN) [3]
out.idmnc = zeros(1,size_glcm_3); % Inverse difference moment normalized
[3]

% correlation with alternate definition of u and s
%out.corrm2 = zeros(1,size_glcm_3); % Correlation: matlab
%out.corrp2 = zeros(1,size_glcm_3); % Correlation: [1,2]

glcm_sum  = zeros(size_glcm_3,1);
glcm_mean = zeros(size_glcm_3,1);
glcm_var  = zeros(size_glcm_3,1);

u_x = zeros(size_glcm_3,1);
u_y = zeros(size_glcm_3,1);
```

```
s_x = zeros(size_glcm_3,1);
s_y = zeros(size_glcm_3,1);

% checked p_x p_y p_xplusy p_xminusy
p_x = zeros(size_glcm_1,size_glcm_3); % Ng x #glcms[1]
p_y = zeros(size_glcm_2,size_glcm_3); % Ng x #glcms[1]
p_xplusy = zeros((size_glcm_1*2 - 1),size_glcm_3); %[1]
p_xminusy = zeros((size_glcm_1),size_glcm_3); %[1]
% checked hxy hxy1 hxy2 hx hy
hxy  = zeros(size_glcm_3,1);
hxy1 = zeros(size_glcm_3,1);
hx   = zeros(size_glcm_3,1);
hy   = zeros(size_glcm_3,1);
hxy2 = zeros(size_glcm_3,1);

%Q    = zeros(size(glcm));

for k = 1:size_glcm_3 % number glcms

    glcm_sum(k) = sum(sum(glcm(:,:,k)));
    glcm(:,:,k) = glcm(:,:,k)./glcm_sum(k); % Normalize each glcm
    glcm_mean(k) = mean2(glcm(:,:,k)); % compute mean after norm
    glcm_var(k)  = (std2(glcm(:,:,k)))^2;

    for i = 1:size_glcm_1

        for j = 1:size_glcm_2

            out.contr(k) = out.contr(k) + (abs(i - j))^2.*glcm(i,j,k);
            out.dissi(k) = out.dissi(k) + (abs(i - j)*glcm(i,j,k));
            out.energ(k) = out.energ(k) + (glcm(i,j,k).^2);
            out.entro(k) = out.entro(k) - (glcm(i,j,k)*log(glcm(i,j,k) +
eps));
            out.homom(k) = out.homom(k) + (glcm(i,j,k)/( 1 + abs(i-j) ));
            out.homop(k) = out.homop(k) + (glcm(i,j,k)/( 1 + (i - j)^2));
            % [1] explains sum of squares variance with a mean value;
            % the exact definition for mean has not been provided in
            % the reference: I use the mean of the entire normalized glcm
            out.sosvh(k) = out.sosvh(k) + glcm(i,j,k)*((i -
glcm_mean(k))^2);

            %out.invdc(k) = out.homom(k);
            out.indnc(k) = out.indnc(k) + (glcm(i,j,k)/( 1 + (abs(i-
j)/size_glcm_1) ));
            out.idmnc(k) = out.idmnc(k) + (glcm(i,j,k)/( 1 + ((i -
j)/size_glcm_1)^2));
            u_x(k)          = u_x(k) + (i)*glcm(i,j,k); % changed 10/26/08
            u_y(k)          = u_y(k) + (j)*glcm(i,j,k); % changed 10/26/08
            % code requires that Nx = Ny
            % the values of the grey levels range from 1 to (Ng)
        end

    end
    out.maxpr(k) = max(max(glcm(:,:,k)));
end
% glcms have been normalized:
% The contrast has been computed for each glcm in the 3D matrix
% (tested) gives similar results to the matlab function
```

```
for k = 1:size_glcm_3

    for i = 1:size_glcm_1

        for j = 1:size_glcm_2
            p_x(i,k) = p_x(i,k) + glcm(i,j,k);
            p_y(i,k) = p_y(i,k) + glcm(j,i,k); % taking i for j and j for i
            if (ismember((i + j),[2:2*size_glcm_1]))
                p_xplusy((i+j)-1,k) = p_xplusy((i+j)-1,k) + glcm(i,j,k);
            end
            if (ismember(abs(i-j),[0:(size_glcm_1-1)]))
                p_xminusy((abs(i-j))+1,k) = p_xminusy((abs(i-j))+1,k) +...
                    glcm(i,j,k);
            end
        end
    end

end

% marginal probabilities are now available [1]
% p_xminusy has +1 in index for matlab (no 0 index)
% computing sum average, sum variance and sum entropy:
for k = 1:(size_glcm_3)

    for i = 1:(2*(size_glcm_1)-1)
        out.savgh(k) = out.savgh(k) + (i+1)*p_xplusy(i,k);
        % the summation for savgh is for i from 2 to 2*Ng hence (i+1)
        out.senth(k) = out.senth(k) - (p_xplusy(i,k)*log(p_xplusy(i,k) +
eps));
    end

end
% compute sum variance with the help of sum entropy
for k = 1:(size_glcm_3)

    for i = 1:(2*(size_glcm_1)-1)
        out.svarh(k) = out.svarh(k) + (((i+1) -
out.senth(k))^2)*p_xplusy(i,k);
        % the summation for savgh is for i from 2 to 2*Ng hence (i+1)
    end

end
% compute difference variance, difference entropy,
for k = 1:size_glcm_3
    for i = 0:(size_glcm_1-1)
        out.denth(k) = out.denth(k) -
(p_xminusy(i+1,k)*log(p_xminusy(i+1,k) + eps));
        out.dvarh(k) = out.dvarh(k) + (i^2)*p_xminusy(i+1,k);
    end
end

% compute information measure of correlation(1,2) [1]
for k = 1:size_glcm_3
    hxy(k) = out.entro(k);
    for i = 1:size_glcm_1

        for j = 1:size_glcm_2
            hxy1(k) = hxy1(k) - (glcm(i,j,k)*log(p_x(i,k)*p_y(j,k) + eps));
            hxy2(k) = hxy2(k) - (p_x(i,k)*p_y(j,k)*log(p_x(i,k)*p_y(j,k) +
eps));
        end
        hx(k) = hx(k) - (p_x(i,k)*log(p_x(i,k) + eps));
        hy(k) = hy(k) - (p_y(i,k)*log(p_y(i,k) + eps));
    end
    out.inf1h(k) = ( hxy(k) - hxy1(k) ) / ( max([hx(k),hy(k)]) );
    out.inf2h(k) = ( 1 - exp( -2*( hxy2(k) - hxy(k) ) ) )^0.5;
end

corm = zeros(size_glcm_3,1);
corp = zeros(size_glcm_3,1);
for k = 1:size_glcm_3
    for i = 1:size_glcm_1
        for j = 1:size_glcm_2
            s_x(k)  = s_x(k)  + (((i) - u_x(k))^2)*glcm(i,j,k);
            s_y(k)  = s_y(k)  + (((j) - u_y(k))^2)*glcm(i,j,k);
            corp(k) = corp(k) + ((i)*(j)*glcm(i,j,k));
            corm(k) = corm(k) + (((i) - u_x(k))*((j) -
u_y(k))*glcm(i,j,k));
            out.cprom(k) = out.cprom(k) + (((i + j - u_x(k) -
u_y(k))^4)*...
                glcm(i,j,k));
            out.cshad(k) = out.cshad(k) + (((i + j - u_x(k) -
u_y(k))^3)*...
                glcm(i,j,k));
        end
    end

 s_x(k) = s_x(k) ^ 0.5;
    s_y(k) = s_y(k) ^ 0.5;
    out.autoc(k) = corp(k);
    out.corrp(k) = (corp(k) - u_x(k)*u_y(k))/(s_x(k)*s_y(k));
    out.corrm(k) = corm(k) / (s_x(k)*s_y(k));
end

function  b=getbinary(a,X);
Centeroid=180;
a=uint8(a);
ag=rgb2gray(a);
[r c p]=size(ag);
for i=1:r
    for j=1:c
        data=ag(i,j);
        if data >Centeroid;
           X(i,j,1)=255;
           X(i,j,2)=255;
           X(i,j,3)=255;
        end

    end
end
b=X;
function GMM=get_GMM(X,Y,g)

k=max(X(:));
GMM=cell(k,1);

for i=1:k
    index=(X==i);
    Y1=Y(:,:,1);
    Y2=Y(:,:,2);
    Y3=Y(:,:,3);
    XX=[Y1(index) Y2(index) Y3(index)];
    GMM{i} = gmdistribution.fit(XX,g,'Regularize',1);
end
function statsa=
gmmsegmentation(X,PreProcessedImage,GMM,k,g,beta,EM_iter,MAP_iter,ShapeText
ure);

        Y=double(PreProcessedImage);
        [X,Y,GMM]=HMRF_EM(X,Y,GMM,k,g,EM_iter,MAP_iter,beta);
        Y=Y*80;
        Y=uint8(Y);
      %OutImage=Y;

       Y=rgb2gray(Y);
       Y=double(Y);

       statsa = glcm(Y,0,ShapeTexture);


function input=wlt4(a);
[LL,LH,HL,HH]=dwt2(a,'haar');

[LL1,LH1,HL1,HH1]=dwt2(LL,'haar');
[LL2,LH2,HL2,HH2]=dwt2(LL1,'haar');

[LL3,LH3,HL3,HH3]=dwt2(LL2,'haar');
input=LL3;
```