UNIVERSITÉ CATHOLIQUE DE LOUVAIN

ÉCOLE POLYTECHNIQUE DE LOUVAIN

Segmentation of the breast region with pectoral muscle suppression and automatic breast density classification.

By Jaume Sastre Tomàs

A thesis submitted for the degree of Master Ingénieur Civil Électricien

Promoters:
Prof. Benoit Macq
Vincent Nicolas

Academic year 2010-2011

Acknowledgements

I would like to thank specially to my promoters, Prof. Benoit Macq (UCL) and Prof. Ferran Marquès (UPC), to give me the opportunity to realize my master thesis in the Université Catholique de Louvain.

I also want to thank my assistant professor Vincent Nicolas for all the assessment that he has given to me along this project. Without his unconditional support and guidance during these months this thesis would not have been possible.

I would like to thank also to Christophe de Vleeschouwer and Antonin Descampe for their support, ideas and contributions during my thesis. Important mention to all people in TELE department to make me feel comfortable during this period.

Finally, I dedicate this thesis to my girlfriend Neus and to my parents Jaime and Rosa, for their unconditional support and encouragement during all these months.

Abstract

Breast cancer is one of the major causes of death among women. Nowadays screening mammography is the most adopted technique to perform an early breast cancer detection ahead other procedures like screen film mammography (SFM) or ultrasound scan. Computer assisted diagnosis (CAD) of mammograms attempts to help radiologists providing an automatic procedure to detect possible cancers in mammograms.

Suspicious breast cancers appear as white spots in mammograms, indicating small clusters of micro-calcifications. Mammogram sensitivity decreases due some factors like density of the breast, presence of labels, artifacts or even pectoral muscle. The pre-processing of mammogram images is a very important step in the breast cancer analysis and detection because it might reduce the number of false positives.

In this thesis we propose a method to segment and classify automatically mammograms according to their density. We perform several procedures including pre-processing (enhancement of the image, noise reduction, orientation finding or borders removal) and segmentation (separate the breast from the background, labels and pectoral muscle present in the mammograms) in order to increase the sensitivity of our CAD system.

The final goal is the classification for diagnosis, in other words, finding the density class for an incoming mammography in order tot determine if more tests are needed to find possible cancers in the image.

This functionality will be included in a new clinical imaging annotation system for computer aided breast cancer screening developed by the Communications and Remote Sensing Department at the Université Catholique de Louvain[11].

The source code for the pre-processing and segmentation step has been programmed in C++ using the library of image processing ITK and CMake compiler. The performed method has been applied to medio-lateral oblique-view (MLO) mammograms as well as on caniocauldal mammograms (CC) belonging to different databases. The classification step has been implemented in Matlab.

We have tested our pre-processing method obtaining a rate of 100% success removing labels and artifacts from mammograms of mini-MIAS database. The pectoral removal rate has been evaluated subjectively obtaining a rate of good removal of 57.76%. Finally, for the classification step, the best recognition rate that we have obtained was 76.25% using only pixel values, and 77.5% adding texture features, classifying images belonging to mini-MIAS database into 3 different density types. These results can be compared with the actual state of the art in segmentation and classification of biomedical images.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Importance and motivation

Breast cancer is considered one of the most important health problems in western countries and indeed it is the most common cancer among women. Approximately, only the 25% of all suspicious lesions to have cancer were confirmed as malignant and the other 75% were confirmed as benignant [30]. The high rate of false-positives is due to the difficulty to obtain accurate diagnosis from mammograms. That is where Computer-Aided Diagnosis (CAD) appears necessary.

Mammography is the most used screening tool for abnormality detection, because it allows an easy way to identify the cancer. However, it is widely believed that not all cancers can be detected using this technique. The detected cancers after a negative mammography are called interval cancers, and is one of the goals of the CAD systems to keep low the rate of these cancers.

CAD systems are a set of tools that help radiologists to detect and diagnose possible cancers. However, recent studies have shown that the sensitivity of these systems decreases as long as the density of the breast increases [46]. The role of these CAD programs is to provide a preliminary diagnosis, with abnormal cases referred for further assessment, for example by a biopsy or an ultrasound.

It has been studied that the relation between breast density and breast cancer risk may be the result of genetic and/or environmental factors that determine breast density. Breast density should be associated with family history of breast cancer. Breast density is widely classified using the BIRADS criteria: 1 = almost entirely fatty, 2 = scattered fibroglandular tissue, 3 = heterogeneously dense and 4 = extremely dense.

Breast density was associated with age, menopause status, body mass index (BMI), and current hormone replacement therapy (HRT). It can also been related to family history because women with higher breast density were more likely to have a first-degree relative who had breast cancer than women with low breast density [46].

1.2 Requirements

The main goal of this thesis is obtain an automatic, fast and reliable procedure to determine the density of a mammogram without the user interaction. It is more difficult to the radiologists finding possible cancers in dense mammograms, so our objective is create a procedure in order to facilitate this task determining if the image is dense enough or not, to ask for a more reliable test like an ultrasound, but also more expensive than a normal mammography.

In order to achieve this objective, according to all the documentation that describes the state of the art, segmentation of the mammogram image into its various regions is an important first step for further processing. This segmentation step also requires pre-processing procedures that facilitates this separation of the breast from other objects in the mammography.

The classification step includes the sampling of the images, model learning and the posterior testing and the presentation of the results. This method also has to involve the minimum interaction with the user and is done with a reasonably computational time.

Summarizing, the main requirements to achieve with our segmentation and classification algorithms in this thesis are: automatic procedure, fast, reliable and minimum interaction with the user.

1.3 Image Datasets

We have in our possession 3 different databases to test our procedures. First of all, we have a set of 41 images manually classified by our own radiologist. Then, we have access to the two main well known databases in the world of mammogram images, mini-MIAS and DDSM.

1.3.1 Own images

We deal with 41 mammogram images manually annotated by a radiologist. There are MLO and CC mammograms and their density is annotated according to the four classes defined in the BIRADS standard. These mammograms don't have any labels or artifacts present in the image.

These images are in DICOM format, but due to some technical problems we have only had access to the jpeg version of them, loosing a huge range of intensity values. Furthermore, the CC images have 3328x4084 pixels and the MLO images have 4915x5355 pixels, so we have sub-sampled the images by a factor of 4 and 5 respectively to reduce the computational time.

1.3.2 mini-MIAS database

We also test our segmentation and classification method with the mini-MIAS database of mammograms. This database is free for scientific research and consists of 161 pairs of medio-lateral oblique (MLO) view mammograms. The base of this database is the original MIAS Database but digitized at 50 micron pixel edge and reduced to 200 micron pixel edge and clipped so that each image is 1024×1024 pixels.

The images of this database have their origin in the United Kingdom National Breast Screening Program. The images are annotated according to their breast density by expert radiologists, using three distinct classes: Fatty (F) (106 images), Fatty-glandular (G) (104 images) and Dense-glandular (D) (112 images). The abnormalities are also described with their kind, location and even their coordinates in the image. The types of possible lesions are: calcifications, well-defined/circumscribed masses, ill-defined masses, architectural distortions or asymmetries. The severity of each abnormality is indicated as benign or malignant.

1.3.3 Digital Database of Screening Mammograms (DDSM)

The Digital Database for Screening Mammography (DDSM) is a resource used for the mammogram image analysis research community. The main purpose of the database is to facilitate the research in the development of computer aided algorithms. The database contains approximately 2.500 studies. Each study includes two images of each breast, along with some associated patient information and image information.

1.4 DICOM Images

Some of our images are in Digital Imaging and Communications in Medicine (DICOM) format, a standard developed by a union of the American College of Radiology (ACR) and National Electrical Manufacturers Association (NEMA) with the collaboration of other standardization organizations. DICOM was developed to enable integration of scanners, servers, workstations and network hardware from multiple vendors into an image archiving and communication system.

DICOM files are formed by a header and a body of image data. The set of standardized fields is called the public DICOM dictionary. A single DICOM file can contain multiples frames, allowing storage of volumes or animations. Image data can be compressed using a large variety of standards, including JPEG (both lossy and lossless), LZW (Lempel Ziv Welch), and RLE (Run-length encoding). For a more detailed description see DICOM Standard.

We worked with DICOM functionalities in ITK, provided by the GDCM library. This open source library was developed by the CREATIS Team at INSA-Lyon.

1.5 Thesis Outline

This thesis is organized in seven chapters, being the first this introduction speaking about the importance and requirements of the thesis, describing the three main used datasets and the format of some images that we have dealt with.

Chapter 2 consists of an overview of the state of the art about mammogram segmentation and breast density classification. There are explained the different used methods through the time and a bit of story and definition of the main used concepts. There is also an explanation of the selected methods and the results of the related work.

The software toolkit used in this thesis is fully explained in chapter 3. It has been differentiated the one used for the segmentation from the other one for the classification step. In concrete, it is explained the Eclipse IDE with its plugin for C++ development(CDT) and the compiler CMake for the segmentation part, the Matlab software for the classification task and the bash scripting for automatizing the process of launching the binary files.

Chapters 4 and 5 conform the main body of the work done in this thesis. It explains carefully the proposed methods for segmentation and classification procedures. Chapter 4 talks about the whole segmentation procedure and its steps are explained in the corresponding subsections. They involve the lecture of the images, the reduction of its computational time if it is necessary, the pre-processing steps like image enhance or noise reduction, find the orientation or the procedure to set the correct region of interest (ROI). Then, it is fully explained the method of segmentation consisting in region growing for the labels and artifacts removal and the Hough transform for the pectoral muscle removal.

In chapter 5, is explained the breast density classification method consisting in the extraction of random subwindows and building an ensemble of trees to train the model. Then, for the testing step, the extracted subwindows from test images are propagated through the ensemble of trees and the final decision is chosen averaging the result of each tree. Also is explained the included cross validation method and the way to present the results in order to visualize the performance of our algorithm.

Chapter 6 shows the obtained results using both methods and with the different used databases. Segmentation results are presented in two groups, the first one indicating the removing labels and artifacts procedure whereas the second showing the pectoral muscle removing results. Then, the classification results for the used method are shown distinguishing between some tested parameters and different types of images. Then, with the best achieved result, there are presented several tests in order to construct charts showing the performance of these parameters related to the recognition rate reached.

The final chapter 7 shows the conclusions and the future work, in other words, the possible lines of investigation related to the thesis and some ideas to improve the results of the presented algorithms in the future.

Also included at the end of the thesis but not considered as a chapters, there are the used references and the annexes.

Chapter 2

State of the art

2.1 Segmentation

Segmentation refers to the process of partitioning a digital image into multiple segments formed by the union of connected pixels. The goal of segmentation is to simplify the representation of an image into something that is more meaningful and easier to analyze, regions. Image segmentation is typically used to locate objects and boundaries. More precisely, image segmentation is the process of assigning a label to every pixel with the same visual characteristics such as color, intensity, or texture. The result of image segmentation is a set of regions that collectively cover the entire image, or a set of contours extracted from the image.

2.1.1 Related Work

There are a lot of procedures used for breast segmentation. Some of them deals the problem of segmenting the whole breast from labels, artifacts and background and then face the problem of removing the pectoral muscle from the breast. In the table 2.1 we can see some of the used methods through the last twenty years[34].

Methods	1980's	1990's	2000's
Histogram	Hoyer79[15]	Lau91[18] Yin91[44] Bick95[2] Byng98[4] Hein98[14]	Masek00[23]
Gradient	Semmlow80[37]	Méndez96[25] Abdel-Mottaleb96[1] Morton96[27] Karssemeijer97[16]	Zhou01[45]
Polynomial Modeling		Stomatakis94[39] Chandrasekhar96[6] Goodsitt98[12]	
Active Contours		Ojala99[29]	Ferrari00[10] McLoughlin00[24] Wirth04[43]
Classifiers		McLou91[24]	Saha01[36] Rickard03[35] Wirth04[43] Tromans04[40]

Table 2.1: Segmentation methods through the recent years.

Some of the techniques that appear in the previous table are explained below.

2.1.1.1 Histogram based techniques

Simple histogram thresholding was one of the first methods to attempt the separation of the breast region. Hoyer et al.[15], Lau et al.[18], Yin et al.[44] and Byng et al.[4] used a simple threshold to segment the breast from the background. Other works of Bick et al.[2] use a combination of local thresholding, region growing and morphological filtering. Hein et al.[14] proposed their own global histogram thresholding, while Masek et al.[23] proposed a local thresholding method.

2.1.1.2 Gradient based techniques

Breast region extraction techniques based on gradient have their origin in the work of Semmlow et al.[37]. He used spatial filters and a Sobel edge detector to obtain the breast boundary. Méndez et al.[25] use a combination of two-level histogram threshold to obtain the breast region and track the boundary using the gradient. Karssemeijer et al.[16] used a multi-resolution scheme to process in low-resolution and extrapolating the result. They processed a preliminary region obtained with global thresholding, with a 3x3 Sobel operator and the pectoral muscle position is estimated via Hough transform. The work of Abdel-Mottaleb et al.[1] was based on different thresholds to find the breast edge. Using the gradient of two images and its union they obtain a possible breast contour. Morton et al.[27] found the edge by a line-by-line gradient analysis. Zhou et al.[45] presented an improvement of this last approach.

2.1.1.3 Polynomial Modeling based techniques

The work of Stomatakis et al.[39] proposes an image pre-processing technique to enhance the response of non-dark pixels, noise reduction and a histogram threshold to obtain the breast region. The boundary is smoothed using Cubic B-splines and samples at fixed pixel intervals are extracted. Then a smooth curve is generated through cubic polynomial calculations. The work of Chandrasekhar et al. [6] was the first attempt to use polynomial modeling to breast segmentation. Finally, a quadratic/cubic polynomial fitting method was proposed by Goodsitt et al.[12].

2.1.1.4 Active Contours based techniques.

Ojala et al.[29] developed one of the first methods that includes active contours to attempt the problem of breast segmentation. Firstly, they apply a global threshold and model the breast with the pixels inside the region. Then, a snake algorithm is applied to obtain the final boundary. Ferrari et al.[10] propose a method that firstly enhances the image with a logarithmic transformation, then it uses an iterative method to find the optimal threshold. Wirth et al.[43] proposed an active contour to segment the breast based on obtaining two preliminary regions using a convolution matrix to enhance the edges and a dual threshold obtained by different techniques.

2.1.1.5 Classifiers based techniques.

Clustering algorithms was also used for breast segmentation. Lou et al.[24], used it to obtain the initial region and then estimates the real boundary extrapolating the detected points. Fuzzy connectedness algorithm was used by Saha et al.[36]. Rickard et al.[35] used an unsupervised artificial neural network model. Wirth et al. [35] also used fuzzy segmentation and evaluated the results in terms of completeness and correctness. Tromans[40], used Expectation Maximization algorithm (EM) combining Fourier approach and the mathematical representation of the image background.

2.1.2 Results of the state of the art segmentation methods

In the table 2.2 we can observe an example of the obtained results in the different pectoral muscle segmentation methods in this particular field.

The results are shown in terms of radiologist assessment (Kwok et al.[17] and Raba et al.[34]), false positives and negatives (Ferrari et al.[10]), overlap ratio (RS) between the original image that contains the muscle and another one that contains the radiologist defined-edge segmentation, percentage of corrected segmented region (Chaabani et al.), Tannimoto Coefficient (TC) and dice similarity coefficient(DSC) (Tzikopoulos et al.[41]).

Author	Method	Database	Results
Kwok et al.[17]	Straight line estimation, validation and iterative cliff detection	mini-MIAS 322 img.	243 (75.5%) line segment. 280 (87.0%) curve segment.
Zhili Chen et al.[7]	Hist. thresholding, edge detection, active contour and polynomial fitting	EPIC 240 img.	93.5% acceptable 87.9% nearly accurate
Ferrari et al.[10]	Hough transform or Gabor wavelets	mini-MIAS 84 img.	FP: 1.98% (1) and 0.58% (2) FN: 25.19% (1) and 5.77% (2)
Carvalho et al.[8]	Mathematical morphology	mini-MIAS 100 img.	79% optimal 18% adequate 3%sub-optimal 0%inadequate
Chaabani et al.[5]	Hough transform and optimization by active contour	DDSM 80 img.	92.5% acceptable 7.5% unacceptable
Mario Mustra et al.[28]	Wavelet decomposition	Patients 40 img.	47.5% good, 15% acceptable 15% unacceptable
S. Tzikopoulos et. al[41]	Straight line estimation and validation. Iterative cliff detection.	mini-MIAS 322 img.	TC: mean=0.9 std=0.079 DSC: mean=0.945 std=0.055
D. Raba et al.[34]	Region growing intensity threshold estimation and boundary refinement	mini-MIAS 320 img.	86% good muscle extractions

Table 2.2: Results for the different segmentation methods commented in the state of the art regarding to pectoral muscle, labels and artifacts removal.

2.1.3 Selected segmentation method

About all the possibilities, we decided to use region growing methods to separate the breast from labels and artifacts present in the images(case mini-MIAS database) and the Hough transform to remove the pectoral muscle from the breast.

Our decision were based on the possibility to implement this methods with ITK, the complexity of the solution and the obtained results. We implemented them using basic filters of ITK that allow to create a pipeline to process sequentially the input image.

2.1.3.1 Region Growing

Region growing algorithms are an easy and effective solution for image segmentation. We have used region growing algorithms to separate the whole breast from the labels, background and tape artifacts present in the image.

Region growing algorithms have proven to be an effective approach for image segmentation. The basic approach of a region growing algorithm is to start from a seed region that are considered to be inside the object to be segmented. The pixels neighboring this region are evaluated to determine if they should also be considered part of the object. If the pixels are considered inside the object, they are added to the region until pixels without the object are found. Region growing algorithms vary depending on the criteria used to decide whether a pixel should be included in the region or not, the type connectivity used to determine neighbors, and the strategy used to visit neighboring pixels.

Of the several implementations of the region growing algorithms present in ITK, after testing almost all of them and we decided to use connected threshold algorithm, fully described in 4.2.1.3 for its simplicity and its better results compared with the other ones.

2.1.3.2 Edge detection with Hough transform

The other segmentation problem that we have faced up is removing the pectoral muscle from the breast. The main problems that we have encountered removing the pectoral muscle were the difference in the breast dimension and intensity level in the mammogram images that difficult the task to find a general procedure that worked with all images.

The Hough transform is a widely used technique for detection geometrical features in images. It is based on mapping the image into a parametric space in which it may be easier to identify if particular geometrical features are present in the image. The transformation is specific for each desired geometrical shape. With the Hough transform we wanted to detect the straight-line that represents the boundary of pectoral muscle in mammogram images. The implementation of this method characterize a line following the next equation:

$$\rho = (x - x_o) \cdot cos\theta + (y - y_o) \cdot sin\theta$$

where $(x - x_o)$ is the origin of the coordinate system, (ρ, θ) represent, respectively, the distance and the angle between $(x - x_o)$ and the coordinates of the pixel being analyzed, as is shown in the figure 2.1.

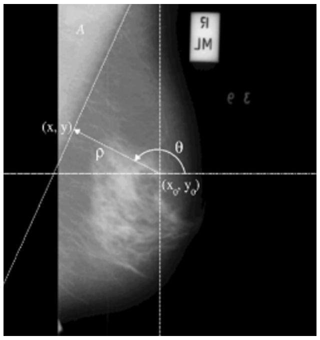


Figure 2.1: Hough transform: Detection for the pectoral muscle

The possible (ρ, θ) values defined by each (x_o, y_o) which are co-linear points in Cartesian image space, map to curves (sinusoids) which intersect in a common point in the polar Hough parameter space. This is the Hough transformation for the straight lines.

The transform is implemented by quantizing the Hough parameter space into finite intervals or accumulator cells. As the algorithm runs, each (x_o, y_o) point is transformed into a discretized curve and the accumulator cells which lie along this curve are incremented. Resulting peaks in the accumulator array represent strong evidence that a corresponding straight line exists in the image. We can see some examples in the figure 2.2:

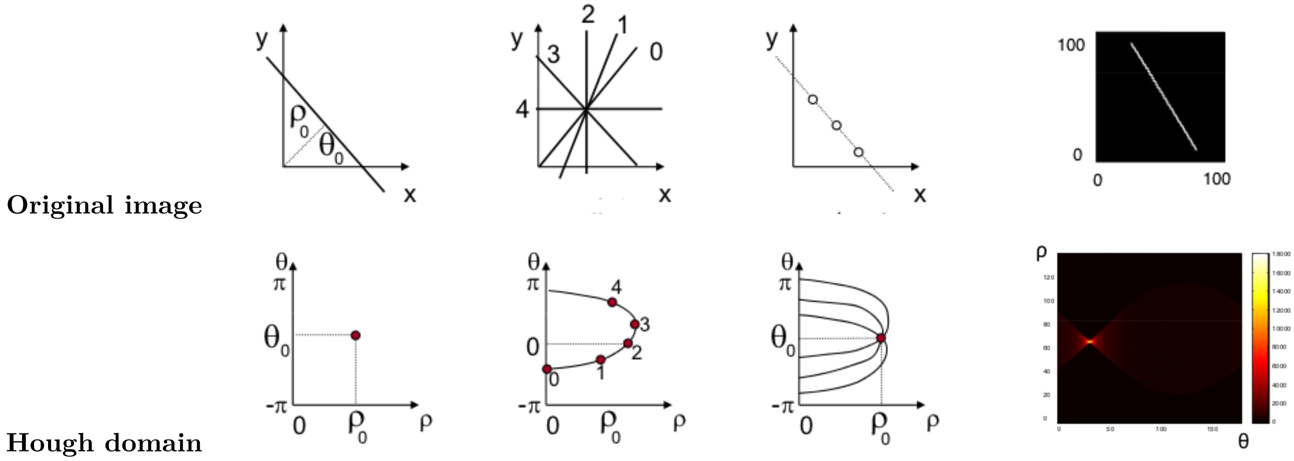


Figure 2.2: Examples of different geometrical shapes in space and Hough domain.

2.2 Classification

Image or pattern classification goal is building models in order to predict the class of new images given a training set where each image is labeled with one single class (supervised learning). The objective is perform an algorithm capable of classifying automatically a new vector using the built model. The usual procedure is [9]:

1. Divide the database in train and test.
2. Select the features to use.
3. Select the classification algorithm.
4. Training the algorithm.
5. Testing the algorithm.

When the database is labeled, it is said as a supervised learning and all of these steps can be summarized in the following figure 2.3:

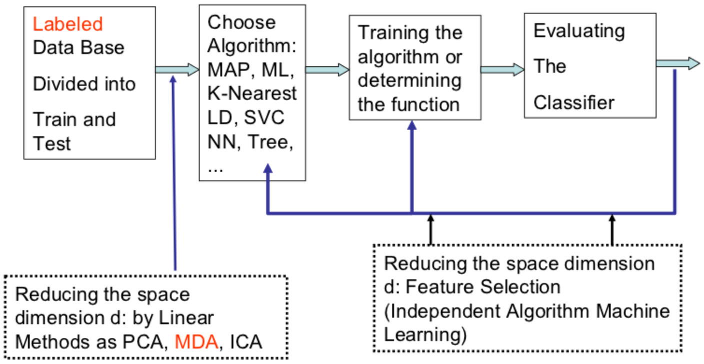

Labeled Data Base
Divided into
Train and Test
Choose Algorithm: MAP, ML, K-Nearest LD, SVC NN, Tree, ...
Training the algorithm or determining the function
Evaluating The Classifier
Reducing the space dimension d: by Linear Methods as PCA, MDA, ICA
Reducing the space dimension d: Feature Selection (Independent Algorithm Machine Learning)

Figure 2.3: Procedure for supervised learning

The classification of breast density is motivated by its use as prior knowledge in breast cancer detection. Many studies have pointed out the importance of breast density as an important factor in the development and risk of breast cancer[46]. Mammogram images with high measures of breast density, require the carefully examination of the radiologist. Thus, quantitative measures of breast density are crucial tools for assessing the association between the risk of breast cancer and mammogram density. Moreover, denser tissue can decrease mammogram sensitivity of the cancer detection CAD systems and cover possible existing lesions.

However, segmentation of the breast in a simple fatty and non-fatty set of regions is much more difficult than it appears due to the large differences in appearances and the variability of the image acquisition and the acquisition parameters.

In this master thesis we propose a new approach to classify mammogram images according to their breast density. As we are going to see in the next chapter, our classification will use information directly extracted from the value of the image pixels. Classification performance was based on the training of an ensemble of trees and its evaluation involves the testing of a extracted subwindows from set of images and their propagation through these trees.

According to the American College of Radiology (ACR) and the Breast Imaging Reporting and Data System (BIRADS), there are four major groups for classifying breast density, as we can observe in the present images of the following figure 2.4:

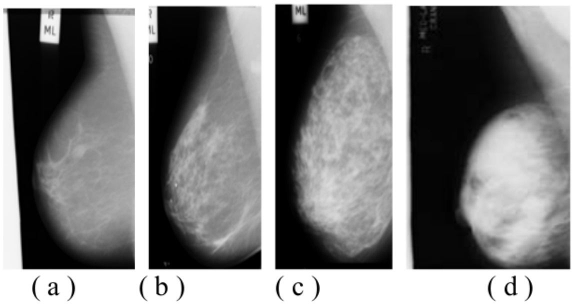


Figure 2.4: Types of breast density according to BIRADS. (a) Predominantly fat; (b) Fat with some fibroglandular tissue; (c) Heterogeneously dense; (d) Extremely dense

2.2.1 Related work

There are lots of classifiers and algorithms that can be used to classify a set of images between a number of classes. Our study case is only one of the possibles fields where these methods can be applied.

For example, Miller and Astley[26] have used the underlying texture to discriminate between breast density types with granulometric techniques and Laws texture masks. Taylor et. al. [31] that have investigated the classification of fatty and dense breast types using an automated method of extracting the region of interest (ROI) based on texture.

Keir Bovis and Sameer Singh[3] extracts 316 feature vector using four approaches to determine texture and constructing Spatial Grey Level Dependency (SGLD) matrices, using the application of the Fourier transform, convolving each mammogram image with each combination of Laws' texture masks and following application of the Discrete Wavelet Transform (DWT) and also extract a series of statistical features. After the feature extraction they reduce the dimensionality of the dataset using Principal Component Analysis (PCA) selecting the first 30 eigenvalues. Once the features have been extracted, they implemented a combination of n Artificial Neural Networks (ANN's) classifiers with a 10-fold cross-validation method using different combination rules.

Stylianos Tzikopoulos et al.[42], extracts first order statistics and fractal features of the mammogram image and performs the classification task using Classification and Regression Trees (CARTs).

Arnau Oliver et al.[30] assumes that each group of mammograms with the same ratio of density have different texture values and they group the pixels with similar tissue as a segmentation strategy. They extract and compare texture features from each cluster and train two different classifiers, k-NN and ID3 decision trees.

Styliani Petroudi and Michael Brady[32] uses breast parenchymal segmentation with information extracted from texture and incorporating multi-vector Markov Random Fields in the classification step.

Styliani Petroudi, Timor Kadir and Michael Brady[33] based their approach in general texture classification, defining a texture classes as statistical distributions (histograms) over "texton" dictionaries. Classification was done it comparing histograms using an appropriate distance measure.

Some of the most important classifiers used in these algorithms are explained below:

2.2.1.1 K-Nearest neighbor

The k-Nearest Neighbor (kNN) consists in classifying test samples based on using the most k similar training samples. Is a type of instance-based learning where the function is approximated locally and the computation is done it until the classification. An object is classified by the majority vote of its neighbors, being assigned to the most common class among its k nearest neighbors.

The training samples are composed as vectors in a multidimensional feature space, each one with a class label. In the training step, the vectors and labels are stored. In the classification phase, k is a user-defined constant, and an unlabeled test vector is classified by assigning the label which is most frequent among the k training samples nearest to that query point. A drawback to the basic "majority voting" classification is that the classes with the more frequent examples tend to dominate the prediction of the new vector. One way to overcome this problem is to weight the classification taking into account the distance from the test point to each of its k nearest neighbors.

2.2.1.2 Classification and regression trees (CART)

Decision tree learning is a model that has the objective to predict the value of a target variable based on several input variables. A training labeled dataset is used to create a classification tree. All the elements start from a root node and successive branches linked to other nodes are created. The decision tree progressively splits the set of training samples into smaller subsets using a sequence of questions in each node to classify a pattern. When all the samples in a subset have the same category the branch of the tree is terminated. A branch can be alternatively terminated with a mixture subset and declared leaf. A feature can be tested once or more with different thresholds. The splitting stops when a node contains only patterns of the same class or when splitting no longer adds useful information to the predictions.

One of the main advantages to select CART's as the classification method is its simplicity and the assumption of underlying distributions of the values of the predicted variables. It is used only simple thresholds and it is not needed any feature reduction. However, the difficulty in using these type of algorithms relies in choosing the correct features, the tests and the thresholds to use in the nodes of the tree.

The property to be tested at each node has to make the immediate descendant node as pure as possible. The main tests used are:

- Entropy impurity:
$$i_E(N) = -\sum_j P(w_j) \cdot log_2(w_j) \qquad P(w_j) = \frac{n_j}{n_{TOTAL}}$$
- Gini impurity:
$$i_G(N) = \sum_i P(w_i) \cdot (1 - P(w_i)) = 1 - \sum_j P^2(w_j)$$
- Misclassification impurity:
$$i_M(N) = 1 - max_j(P(w_j))$$

Entropy impurity measure is the most acceptable in most cases (our approach).

If the training set is very big, the obtained tree can be overfitted. In extreme splitting, each leaf corresponds to a single training vector. There are different stopping criteria to stop splitting a CART:

1. The impurity lose must be higher than a specific threshold.
2. Minimum description length (MDL): a criterion function is minimized→$\alpha \cdot size + \sum_{N\,leaf} i(N)$
3. Minimizing the error on the validation data.
4. When a node presents less than a determined number of vectors (our approach).

A label is assigned of the category that has more vectors represented.

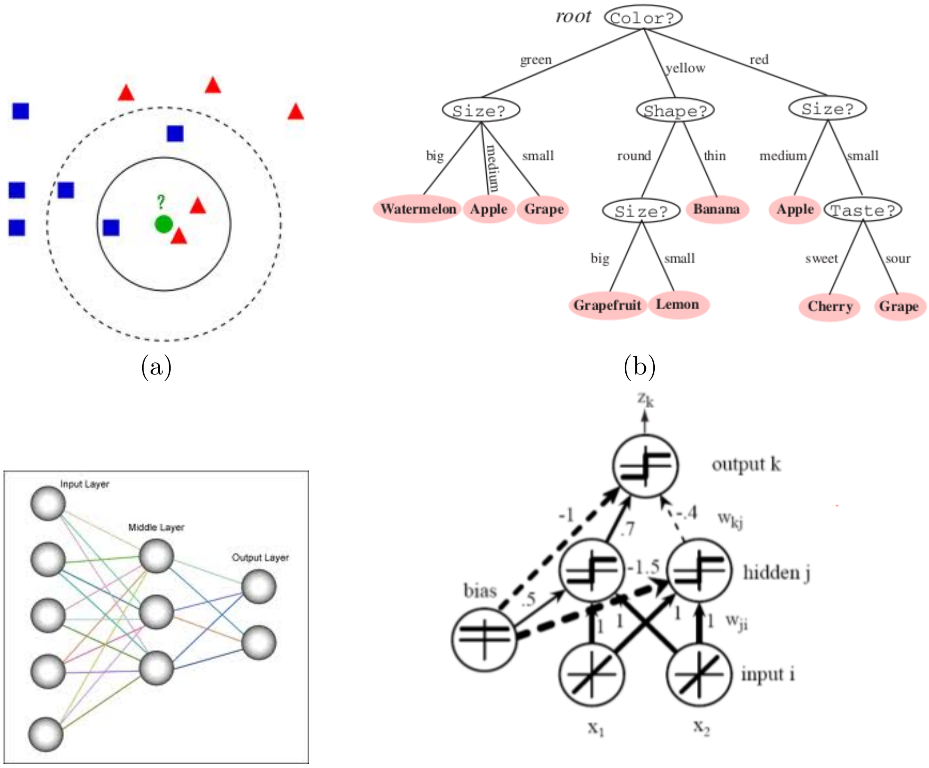
If none splitting criteria has applied it exists a way to cut the tree if it is too big: pruning. Where some subtrees are substituted for leafs following different pruning criteria. Pruning is preferred over stopped splitting but it is computationally worse.

2.2.1.3 Neural networks

Artificial Neural Networks (ANN) appears from the idea to model mathematically the human intellectual abilities. They are formed by some layers (input, hidden and output) and a classifier is used to obtain arbitrary decision regions. A NN of two layers is called perceptron and is the simplest artificial network that uses a linear function. A multilayer neural network or a multilayer perceptron is a non linear function where the parameters are learned from the training database.

Optimal network topology is very dependent of each problem and a complexity adjustment is necessary to decide how many free parameters we need.

In the following figure 2.5, we can observe an example of each one of the commented methods:



(a) (b)

(c) (d)

Figure 2.5: Different classification methods: (a) k-NN, (b) CART, (c) and (d) Neural networks

2.2.2 Results of the state of the art classification methods

In the table 2.3 it is shown the recognition results about the main state of the art classification methods dealing with the problem of breast density classification with different databases. The results are expressed either in recognition rate or in confusion matrices.

<table>
<tr><th>Author</th><th>Method</th><th>Results</th><th>Database</th></tr>
<tr><td>K. Bovis et al.[3]</td><td>SGLD matrices + PCA
Artificial NN</td><td>71.4% 4 class problem
96.6% 2 class problem</td><td>DDSM
377 img</td></tr>
<tr><td>K. Bovis et al.[3]</td><td>ANN with back prop.
kNN
ANN: perceptron
ANN with gradient desc.</td><td>70.4%
71%
68%
69%</td><td>mini-MIAS
322 img</td></tr>
<tr><td>S. Tzikopoulos et al.[42]</td><td>Class. & Regres.
Trees (CARTs)</td><td>
<table>
<tr><td colspan="2">Breast density</td><td colspan="3">True class</td></tr>
<tr><td></td><td></td><td>F</td><td>G</td><td>D</td></tr>
<tr><td>Predicted class</td><td>F</td><td>82</td><td>21</td><td>7</td></tr>
<tr><td></td><td>G</td><td>20</td><td>52</td><td>28</td></tr>
<tr><td></td><td>D</td><td>8</td><td>32</td><td>77</td></tr>
</table>
</td><td>mini-MIAS
322 img</td></tr>
<tr><td>T. MacGahan et al.[19]</td><td>Support Vector Machine and pseudo-random splits of the dataset</td><td>Mean(splits) 63.6% 83.0 % 80.3 %
(1) Images 1x1, (2) Images divided 4 equal size 2x2, (3) 9 images 3x3</td><td>mini-MIAS
322 img</td></tr>
<tr><td rowspan="3">A. Oliver et al.[30]</td><td>k-NN classifier 40.3%</td><td>
<table>
<tr><td colspan="2">Breast dens.</td><td colspan="4">True class</td></tr>
<tr><td></td><td></td><td>I</td><td>II</td><td>III</td><td>IV</td></tr>
<tr><td>Predic-ted class</td><td>I</td><td>17</td><td>24</td><td>5</td><td>4</td></tr>
<tr><td></td><td>II</td><td>27</td><td>35</td><td>30</td><td>8</td></tr>
<tr><td></td><td>III</td><td>3</td><td>25</td><td>54</td><td>18</td></tr>
<tr><td></td><td>IV</td><td>3</td><td>9</td><td>23</td><td>15</td></tr>
</table>
</td><td rowspan="3">DDSM</td></tr>
<tr><td>ID3 decision 43.3%</td><td>
<table>
<tr><td colspan="2">Breast dens.</td><td colspan="4">True class</td></tr>
<tr><td></td><td></td><td>I</td><td>II</td><td>III</td><td>IV</td></tr>
<tr><td>Predic-ted class</td><td>I</td><td>24</td><td>20</td><td>3</td><td>3</td></tr>
<tr><td></td><td>II</td><td>34</td><td>49</td><td>10</td><td>7</td></tr>
<tr><td></td><td>III</td><td>18</td><td>17</td><td>40</td><td>25</td></tr>
<tr><td></td><td>IV</td><td>9</td><td>5</td><td>19</td><td>17</td></tr>
</table>
</td></tr>
<tr><td>tree combination 47%</td><td>
<table>
<tr><td colspan="2">Breast dens.</td><td colspan="4">True class</td></tr>
<tr><td></td><td></td><td>I</td><td>II</td><td>III</td><td>IV</td></tr>
<tr><td>Predic-ted class</td><td>I</td><td>23</td><td>21</td><td>3</td><td>3</td></tr>
<tr><td></td><td>II</td><td>32</td><td>48</td><td>12</td><td>8</td></tr>
<tr><td></td><td>III</td><td>3</td><td>22</td><td>52</td><td>23</td></tr>
<tr><td></td><td>IV</td><td>5</td><td>7</td><td>20</td><td>18</td></tr>
</table>
</td></tr>
<tr><td>S. Petroudi et al. [32]</td><td>Markov Random Fields</td><td>Dense: 96.7% Fatty: 93.8%
Breast edge: 92.8%</td><td>“Screen”
32 img</td></tr>
<tr><td>S. Petroudi et al.[33]</td><td>Texton dictionaries</td><td>
<table>
<tr><td>Breast dens.</td><td>(%)</td><td colspan="4">True class</td></tr>
<tr><td></td><td></td><td>I</td><td>II</td><td>III</td><td>IV</td></tr>
<tr><td>Predic-ted class</td><td>4 class</td><td>91</td><td>64</td><td>70</td><td>78</td></tr>
<tr><td></td><td>2 class</td><td colspan="2">91</td><td colspan="2">94</td></tr>
</table>
</td><td>DDSM</td></tr>
</table>

Table 2.3: Results for the different breast density classification methods commented in the state of the art.

2.2.3 Selected method: Random sub-windows with decision trees

The method is based on random sub-window extraction and building an ensemble of extremely randomized trees. The algorithm follows several steps: feature extraction consisting of random sub-window extraction from images of the dataset, construction of a model by machine learning based on a set of trees using the extracted sub-windows and the classification of a new image, extracting its sub-windows and the application of the built model to them. These extracted subwindows are propagated through the ensemble of trees and the final class is obtained averaging the outcomes of all trees.

The basics of the algorithm used for the classification method has been developed by Antonin Descampe based on the work of Raphael Marée et. al [21, 20] in order to try to solve the problem of biomedical image classification in a set of predefined classes.

Our approach uses the base of his algorithm to classify mammograms according to their density. We have added some blocks to implement cross validation, results presentation and some scripts to test the experiments using different parameters values in order to find the best possible result. As well as, we have been capable of understanding the followed method in the algorithm and providing some new possibilities to test in each node.

The performance of the algorithm will be fully described in the section 5.1.

Chapter 3

Development environment

3.1 ITK Insight Toolkit

One of the first requirements of the project was performing the algorithms in C++ language and using an image processing library called ITK. We have been developed our C++ applications using existing ITK classes and we have became familiar with its palette of objects and the ways of combining them.

The Insight Toolkit (ITK) is an open-source, which developers from around the world can use, debug, maintain and extend the software. The toolkit is used for performing registration (aligning and finding correspondence between data) and segmentation (the process of identifying and classifying data extracted from digitally sampled representation acquired from medical images as CT or MRI scanners) among other applications.

ITK is implemented in C++ and due to the wrapping process, is able to generate interfaces between C++ and interpreted programming languages such as Tcl, Java, and Python. ITK has been developed and tested across different combinations of operating systems including MS-Windows, Linux, Solaris, IRIX, Mac OSX, and Cygwin.

ITK's implementation is done following a generic way of programming, which uses templates, to apply the same code generically to any class or type, and making the code highly efficient. Many software problems are discovered at compile-time, rather than at run-time during the program execution.

The challenge of supporting ITK across platforms has been solved through the use of CMake, a cross-platform, open-source building system. CMake generates native Makefiles under UNIX and Cygwin systems and Visual Studio workspaces under Windows. The information used by CMake is provided by CMakeLists.txt.

He have used synaptic to install ITK in our computer through the libraries libInsight and libITK in their latest versions. Then, install CMake also in its latest version. It is also possible, instead of installing it, download the source code from the ITK website and configure it using CMake. This downloaded Insight source directory contain applications, CMake files, code, documentation, testing and validation files, examples, utilities and wrapping files.

It is also possible to include new classes that hadn't been tested or in review. The source file .txx has to be included in the folder /usr/include/InsightToolkit and after a re-compilation process with CMake is already possible to use them. We had to do that procedure in some cases when we wanted to try some filters that hadn't been added to ITK yet but their code is already available.

3.2 CMake

Compiling our C++ applications using ITK was other of the problems that we have encountered. CMake fixed the problem because is an extensible, open-source system that manages the build process in many operating systems independently of the compiler. The configuration files placed in each source directory (CMakeLists.txt files) are used to generate standard build files. CMake is designed to support complex directory hierarchies and applications dependent on several libraries and projects consisting of multiple toolkits with several directories.

CMake can generate a native build environment that will compile source code, create libraries, generate wrappers and build executables in arbitrary combinations. When CMake runs, it locates include files, libraries, and executables, and may encounter optional build directives and gathers them into the cache, which may be changed by the user prior to the generation of the native build files. The build process is controlled by creating one or more CMakeLists.txt files in the working directory(or even subdirectories) that makes the project, consisting of one or more commands. Each command has the form COMMAND(name of the command) + args(white-space separated list of arguments). CMake provides many predefined commands, but is possible to add new ones.

In the following code we can visualize an example of the CMakelists.txt file used to compile our C++ applications. This folder can be much more extensive but this is the minimum necessary to compile correctly our programs.

```
# This is the root ITK CMakeLists file.
# Set the version and the policy
CMAKE_MINIMUM_REQUIRED(VERSION 2.4)
IF(COMMAND CMAKE_POLICY)
  CMAKE_POLICY(SET CMP0003 NEW)
ENDIF(COMMAND CMAKE_POLICY)

# This project is designed to be built outside the Insight source tree.
PROJECT(BreastDensityAutomaticClassification)

# Find ITK.
FIND_PACKAGE(ITK REQUIRED)
INCLUDE(${ITK_USE_FILE})

# Add the main executable and the programs that belongs to it
ADD_EXECUTABLE(MainProcess Main.cpp  BreastMLOSegmentation.cpp BreastDICOMSegmentation.cpp
     BreastLabelsAndArtifactsRemoval.cpp ConvertRaw2Dicom.cpp)
TARGET_LINK_LIBRARIES(MainProcess ITKCommon ITKIO ITKBasicFilters ITKStatistics)
```

In our implementation, we have decided to use CMake with Eclipse, one of the most powerful integrated development environments to facilitate the task of developing source code.

3.3 Eclipse, CDT Plug-in and C++ language

Eclipse is the chosen software to implement our code in C++. It is an integrated development environment (IDE) and an extensible plug-in system that allows to develop applications in other languages than Java, for example C, C++, COBOL, Perl, PHP or Python. The plugin used for developing code in C++ is called Eclipse CDT.

Eclipse began as an IBM Canada project and it was developed by Object Technology International (OTI). Eclipse is under the Eclipse Public License as a free and open source software. Eclipse provides the Eclipse Rich Client Platform (RCP) for developing general purpose applications. Some of the components are:

- A core (micro-kernel) manager.
- A standard bundling framework.
- A portable widget toolkit.
- File buffers, text handling, text editors.
- A workbench (views, editors, perspectives, wizards).
- Data binding.
- Update manager.

The CDT Project provides a fully functional C and C++ Integrated Development Environment based on the Eclipse platform. Includes support for project creation, build and compile the source files, source navigation, various source knowledge tools, such as hierarchy, call graph, browser, code editor with syntax highlighting, folding and hyper-link navigation, source code refactoring and code generation, visual debugging tools, including memory, registers and disassembly viewers.

The C and C++ languages are among the most popular and widely used programming languages in the world. C++ is a statically typed, free-form, multi-paradigm, compiled, general-purpose programming language and it comprises a combination of both high-level and low-level language features. It was developed as an enhancement to the C language. First adding classes, then virtual functions, operator overloading, multiple inheritance, templates and exception handling among other features. C++ is one of the most popular programming languages and its application domains include systems software, application software, device drivers, embedded software, high-performance server, client applications and entertainment software. C++ has greatly influenced by many other popular programming languages, like C# and Java. After years of development, the C++ programming language was considered an standard.

3.4 Eclipse and CMake

CMake includes a generator for Eclipse CDT 4.0 or newer. This generator creates a set of .project/.cproject files that can be imported in Eclipse as an "Existing Eclipse project".

In our case, the source tree of our project is /home/name/workspace/project_name. We had a proper CMakeLists.txt file in the src directory, showed in the previous section. We had to create a build directory, go there and run CMake, using the following commands:

- mkdir /home/name/workspace/Build
- cd /home/name/workspace/Build
- cmake -G"Eclipse CDT4 - Unix Makefiles" -D CMAKE_BUILD_TYPE=Debug ../Build

It's important that the project's name should be different from the executable name and from the build folder name. Then it's time to import the created project file into using Menu File→Import Select General→Existing projects into workspace, browsing and selecting the build tree. Then it's possible to develop and compile the C++ project using CMake and Eclipse.

3.5 MATLAB

MATLAB is developed by MathWorks and it is a programming language for numerical data that allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces and interfacing with programs written in other languages, including C, C++, Java, and Fortran. Firstly MATLAB was designed for numerical computing and their different toolboxes allow access to computing different capabilities. Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems. MATLAB is widely used in academic and research institutions as well as industrial enterprises.

Matlab is used in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis and computational biology. Many toolboxes or collections of special-purpose functions extend the Matlab environment to solve particular classes of problems in these application areas.

Some of the key features that make Matlab one of the most important and powerful languages for computational analysis and other tasks are the following ones:

- High-level language for technical computing.
- Development environment for managing code, files, and data.
- Interactive tools for iterative exploration, design, and problem solving.
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration.
- 2-D and 3-D graphics functions for visualizing data.
- Tools for building custom graphical user interfaces.
- Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel.

Matlab supports the entire data analysis process, from acquiring data from external devices and databases, through preprocessing, visualization, and numerical analysis. It also provides interactive tools and command-line functions for data analysis operations.

For performing the classification task, we have based our work in a previous algorithm already implemented in Matlab, so we have followed the implementation in this high-level language and interactive environment.

3.6 Scripts bash

Bash is a Unix shell distributed widely as the shell for the GNU operating system as the default shell on Linux, Mac OS X and Darwin. Bash is a command processor that is typically run in a text window, allowing the user to type commands which cause actions. Bash can also read commands from a file, called script. These scripts are a tool for building applications using the entire repertoire of system calls, tools, utilities, and compiled binaries available for invocation by the shell. It is possible also to perform testing, conditionals and loop constructs, making more powerful and flexible these scripts.

Due to the problem of launching experiments on the segmentation and classification algorithms using all the images in a single directory, we have learned how to use bash scripts.

In the classification part, we didn't have this problem because with Matlab allows an easy way to do it, but the problem appeared with the segmentation procedure implemented in C++. We have adopted the decision to use bash scripts to iterate through a directory and launch the binary executable compiled in C++ for each one of the found images. For example, pre-processing functions like noise removal or flipping images, labels and artifacts removal, pectoral muscle removal or to convert images into specific formats (pgm, png, etc).

In the following code, we can observe an example of launching our executable for the segmentation step from a bash script:

```
#!/bin/sh
#SBATCH --time=1

IMGS_DIR=/home/jaume/Escritorio/PFC/mias_database/converted
RESULTS_DIR=/home/jaume/Escritorio/PFC/mias_database/converted/flipped

# For each image in this directory
for IMG in $IMGS_DIR/*.png; do
 echo 'Processing image: '$IMG
 IMG_OUT=`basename $IMG`
 # Launching the C++ executable
 /home/jaume/workspace/Build/MainProcess $IMG $RESULTS_DIR/$IMG_OUT
done
echo 'done'
```

Chapter 4

Methodology for Segmentation

In this chapter we are going to describe the whole procedure that we have done to achieve the complete segmentation of the breast from background, labels, artifacts and pectoral muscle in mammogram images. To accomplish our objective, we have followed the procedure presented in the following diagram:

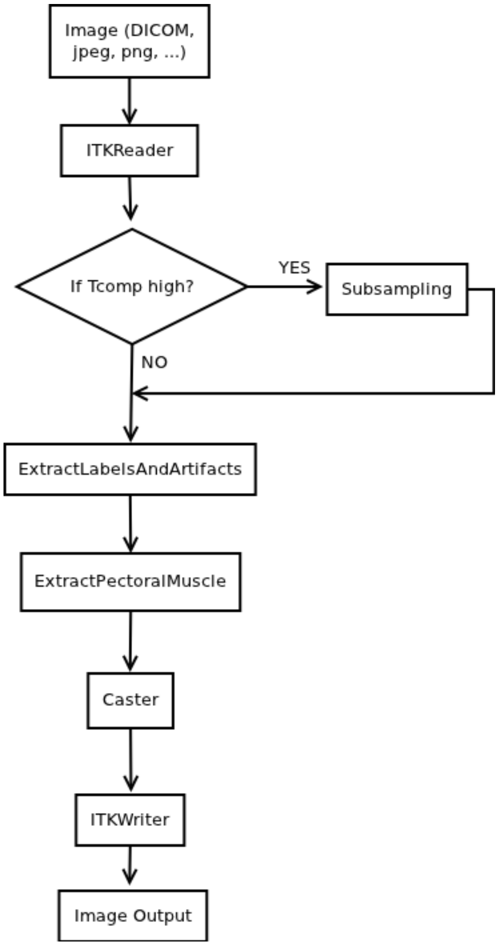


Figure 4.1: Flow diagram of the whole procedure for the segmentation step.

The previous diagram covers the whole range of possibilities that we have in the image's treatment. In each of the following sections we will show all the procedure involved in each step.

4.1 Initial procedure

4.1.1 Lecture of the image

For reading any kind of image is required to include the header file of the itk::ImageFileReader class. Then, the image type should be defined by specifying the type used to represent pixels and the dimension of the image. Finally, it's time to instantiate the image reader class.

The reader type can now be used to create one reader object, an itk::SmartPointer, used to receive the reference to the newly created reader using the New() method. The minimum information required by the reader is the filename of the image to be loaded in memory, provided through the SetFileName() method. The file format here is inferred from the filename extension.

The reader objects take part into a pipeline and this mechanism ensures that the reader only executes when a data request is made, the update invocation on the filter, triggers an update of the reader. To access to the newly read image is done it by calling the GetOutput() method on the reader. This method has to be called before the update request is sent to the reader, otherwise you will obtain an ITK error.

We can observe in the following code, an example of reading an image in ITK:

```
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"

...

// Define the input image
typedef   float          InternalPixelType;
typedef itk::Image< InternalPixelType , 2 >  InternalImageType;

// Define the output image
typedef unsigned char OutputPixelType;
typedef itk::Image< OutputPixelType , 2 > OutputImageType;

// We define the reader and writer
typedef itk::ImageFileReader< InternalImageType > ReaderType;
typedef itk::ImageFileWriter< OutputImageType > WriterType;

// We instantiate reader and writer types
ReaderType::Pointer  reader = ReaderType::New();
WriterType::Pointer  writer = WriterType::New();

reader->SetFileName( filenameInput );
reader->Update();

...
filter->SetInput( reader->GetOutput );
```

It is important to mention that DICOM images has an special treatment. ITK provides classes for reading and writing DICOM files, these functionalities are provided by the GDCM library (itkGDCMImageIO.h).

Following the same procedure showed previously, we have to declare include files, image type, its dimension and pixel type, declare and instantiate the reader. If we want to save the image in other format file different than DICOM, we can use a normal writer as we used in the last example. However, if some modifications are done in the image, we have to specify the DICOM format to the writer and tell the writer not to use the MetaDictionary from the input because the image has suffered modifications. In the following code we can observe an example to read and write a DICOM image in ITK:

```
// Includes
#include "itkMetaDataDictionary.h"
#include "itkMetaDataObject.h"
#include "itkGDCMImageIO.h"
...

// Declarations of the ImageType and the reader and writer
...

// The GDCMImageIO object is created in order to provide the services for
// reading and writing DICOM files. The newly created image IO class is
// connected to the reader.

typedef itk::GDCMImageIO            ImageIOType;
ImageIOType::Pointer gdcmImageIO = ImageIOType::New();
reader->SetImageIO( gdcmImageIO );

//writer->UseInputMetaDataDictionaryOff(); // To save to DICOM format an image with
     modifications
writer->SetImageIO( gdcmImageIO );
```

4.1.2 Reducing computational time: Sub-sampling

We have worked with the 41 images of our radiologist, in DICOM format, that have large dimension and require high computational time. So, we have decided to sub-sample them to reduce the computational complexity and the time required to process them.

Before the sub-sampling, the image must be preprocessed to avoid aliasing so it has been used the itk::Recursive-GaussianImageFilter in both dimensions, along the x and y axis. And also specifying the correct sigma and direction.

In ITK, the filter used to perform re sampling is the itk::ResampleImageFilter through the use of itk::Transforms. The expected inputs by this filter are an image, a transform and an interpolator. In our case, we have used the itk::IdentityTransform and the itk::NearestNeighborInterpolateImageFunction following one of the examples of ITK toolkit (Examples/Filtering/ResampleImageFilter.cxx).

The space coordinates of the image are mapped through the transform in order to generate a new image. The extent and spacing of the resulting image are selected by the user. Re-sampling is performed in space coordinates(millimeters), not pixel/grid coordinates.

The sub-sampling filter will need a transform in order to map point coordinates. In this particular case we use the itk::IdentityTransform because the image is going to be sub-sampled by preserving the physical extent of the sampled region. It is used by methods that require a transform but being certainly that the transform will have no effect in the outcome of the process. The affinity transform is defined using the image dimension and the type used for representing space coordinates. An instance of the transform object is instantiated and passed to the sub-sampling filter.

The filter will also need an interpolator in order to compute intensity values for the new sub-sampled image. The interpolator is required since the mapping from one space to the other will often require evaluation of the intensity of the image at non-grid positions. It generally assumes that intensity varies linearly between grid positions. It is defined using the full image type and the type used for representing space coordinates and also an instance of the interpolator object is instantiated and passed to the sub-sampling filter.

This interpolator computes the intensity value of non-existing pixels by averaging the two nearest neighbors and produces a natural smoothing of the image. In our case, he didn't have suffered this effect because our sub-sampling procedure only involves taking one of four pixels in the x axis and one of five in the y axis, obtaining better results in terms of computational time.

4.2 Breast Segmentation

While we were performing this step, we realized that we had to follow some hypotheses regarding to the appearance of the breast and the localization of the pectoral muscle for automatize the process:

1. Labels and artifacts only appear in images from mini-MIAS database. So the procedure to remove them it has only to be applied to this database.
2. Pectoral muscle only appears in medio-lateral oblique mammograms (MLO) as a roughly triangular region taking up a corner of the mammogram. Therefore, it's not necessary to perform the pectoral muscle removal for CC images.
3. To perform the segmentation part, it was necessary to found the orientation of the image and flip it to follow the same direction for consistency.
4. The pectoral muscle is usually defined as a region of higher intensity than the surrounding tissue. It doesn't happen in all the images, and it is one of the main problems that we have encountered while we were doing the experiments.

4.2.1 Labels and artifacts removal

In this section we are going to present the procedure to remove background noise, labels and artifacts present in the mammogram images. It involves a region growing algorithm and requires a pre-processing step. The following figure 4.2 shows the whole procedure.

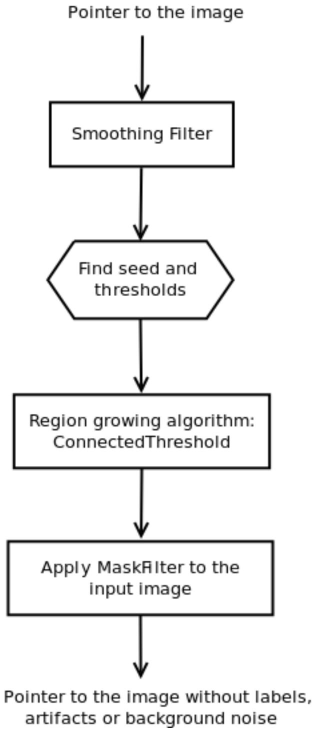

Pointer to the image
Smoothing Filter
Find seed and thresholds
Region growing algorithm: ConnectedThreshold
Apply MaskFilter to the input image
Pointer to the image without labels, artifacts or background noise

Figure 4.2: Flow diagram of the labels and artifacts step.

4.2.1.1 Image enhancement and noise problems

In order to improve the results of our method, we have implemented a pre-processing procedure to enhance the image and remove the background noise of the image.

Different types of noise appear in mammography images can be observed in figure 4.3. High intensity noise is characterized as noise that has high values of intensities, such as labels or scanning artifacts. The algorithm used to find these regions and extract them from the original image is presented in detail in 4.2.1.3 and uses a region growing algorithm. Tape artifacts are any markings left by tapes, or other shadows.

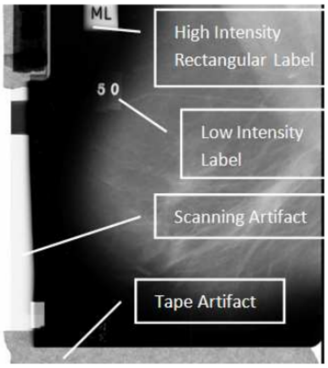


Figure 4.3: Different type of noise in mammograms

Noise present in the image can reduce the capacity of this filter to grow large regions. We have pre-processed the image using an edge preserving smoothing filter, itk::CurvatureFlowImageFilter, presented in the following section in order to avoid as much as possible the noise presence.

4.2.1.2 Remove speckle noise: Median filter vs CurvatureFlowImageFilter

The images of the database contain also noise from the digitization process, such as speckle noise. This type of noise should be removed, in order to enhance the quality of the image and make the segmentation task easier. We tested some techniques to to remove noise and preserve the edges of the image and avoid effects from micro-texture that could appear in some regions.

Firstly, we tried with the median filter with radius 5x5 described in ITK, itk::MedianImageFilter, commonly used as a robust approach for noise reduction. This filter is particularly efficient against salt-and-pepper noise. In other words, it is robust to the presence of gray-level outliers. MedianImageFilter computes the value of each output pixel as the statistical median of the neighborhood of values around the corresponding input pixel. The statistical median of the neighborhood on the left is passed as the output value associated with the pixel at the center of the neighborhood. The drawback of this filter is that increases the computational time of the algorithm so we looked for another one faster than this one.

We also tested the itk::CurvatureFlowImageFilter. This filter performs edge-preserving smoothing similar to classical anisotropic diffusion. The filter uses a level set formation where the iso-intensity contours in an image are viewed as level sets, and pixels of a particular intensity form one level set. The level set function is then evolved under the control of a diffusion equation and with speed proportional to the curvature. The results with this filter were faster than with median filter so we selected it for our experiments.

4.2.1.3 Segmentation procedure: Region growing using Connected threshold algorithm

Several implementations of region growing are available in ITK and after testing most of them we have chosen the connected threshold algorithm(itk::ConnectedThresholdImageFilter). This algorithm use the simple criterion for including pixels in a growing region to evaluate intensity value inside a specific interval. Thus the algorithm establishes a criterion to decide whether a particular pixel should be included in the current region or not. The criterion used by the ConnectedThresholdImageFilter is based on an interval of intensity values provided by the user. Values of lower, upper threshold and initial seed should be provided. The region growing algorithm includes those pixels whose intensities are inside the interval:

$$I(X)\epsilon[lower, upper]$$

The main difficulty in this procedure is finding the correct thresholds and the suitable seed to begin the algorithm. We wanted to keep the large element (breast) from all the background, labels and artifacts present in the image, so we put the seed inside this element. To calculate this initial seed, after the examination of all the images in the database, we have decided to use the center of the image as initial point. The reason was because in all images, the center point become part of the breast. Furthermore, we had to find an automatic and general procedure that it worked with all the images in the database.

We also set manually two pixel values for the lower and higher thresholds to run the algorithm. The values that we have chosen were 50 and 250 because we wanted to avoid the labels, with high values close to 255, and the background close to 0.

We tested the procedure with the 322 images of the mini-MIAS database, because is the only database affected by labels and artifacts in their images, and we managed to remove all the labels and background noise of the all images except in one of them.

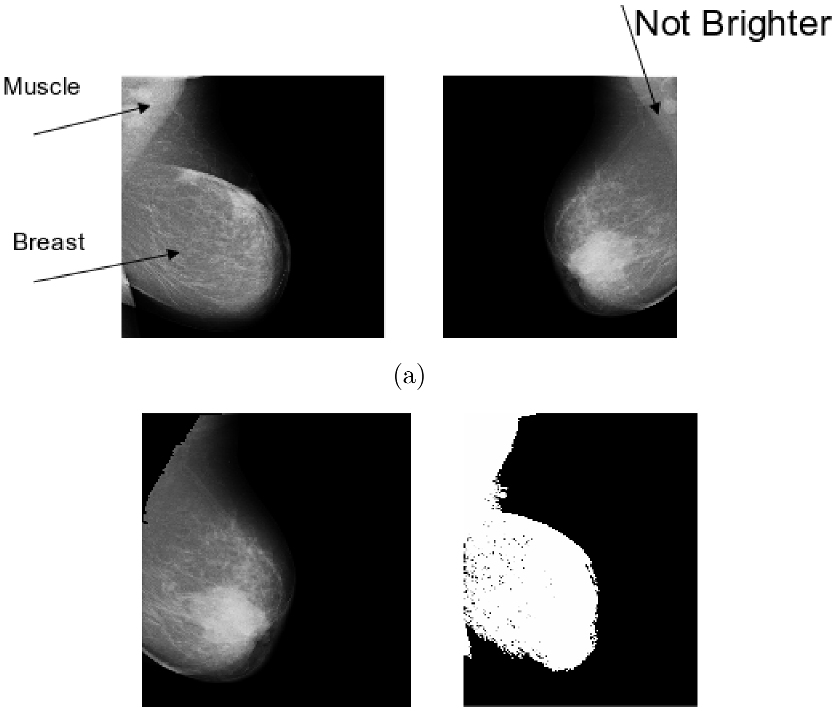
After finding the largest region and segmenting it from the mammogram, we have applied the itk::MaskFilter to keep only the segmented part from the original image and then save it in a new image.

Is remarkable that with this procedure we don't need to perform any other procedure to our images. Furthermore, it is a fast and simple procedure.

4.2.2 Pectoral muscle removal

4.2.2.1 Problems with region growing

We have made a lot of tests before deciding for the actual procedure and we encountered many problems. The mainly one is due to we have images where the muscle is brighter than the breast, where the region growing algorithm works good, but in others the muscle isn't and the region growing doesn't work properly. The main problem using this method is that you have to establish an upper and lower threshold to separate both objects (breast and muscle) and if the intensity of the muscle pixels are different in some images this threshold won't be established without user's interaction, which was one of the initial requirements. In the figure 4.4 we can observe the mentioned problem.

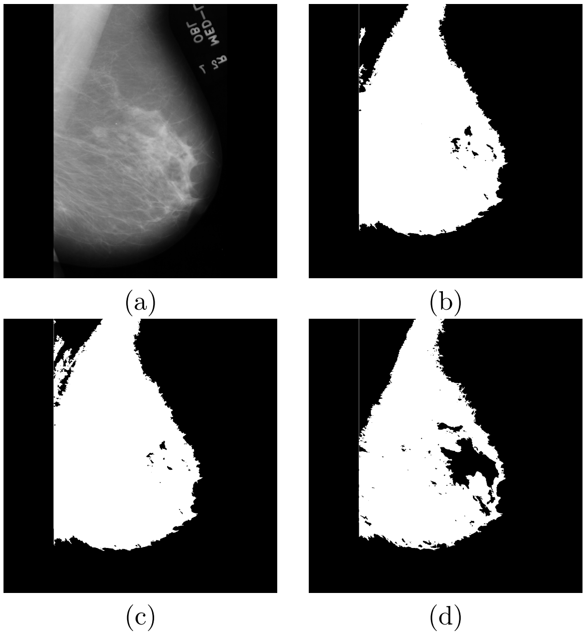


(a)

(b)

Figure 4.4: Problem of the region growing algorithm due to the different breast intensity. (a) Different intensity breast in mammogram images. (b) Problems of the region growing algorithm used.

In the figure 4.5 we can observe some examples when we were trying to use different region growing algorithms to remove the pectoral muscle:



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

Figure 4.5: Other methods implemented for extracting pectoral muscle: (a) Original Mammography, (b) Connected threshold, (c) Isolated Connected, (d) Confidence connected.

4.2.2.2 Adopted procedure: Hough Transform

After all of these tests we have chosen a solution that works in most of the cases based on the edge detection filter and the Hough transform. Also the computational time was lower than using the region growing algorithms.

As in the previous parts, in the following figure 4.6 we can observe the flow diagram that shows the whole procedure implemented to achieve the final goal:

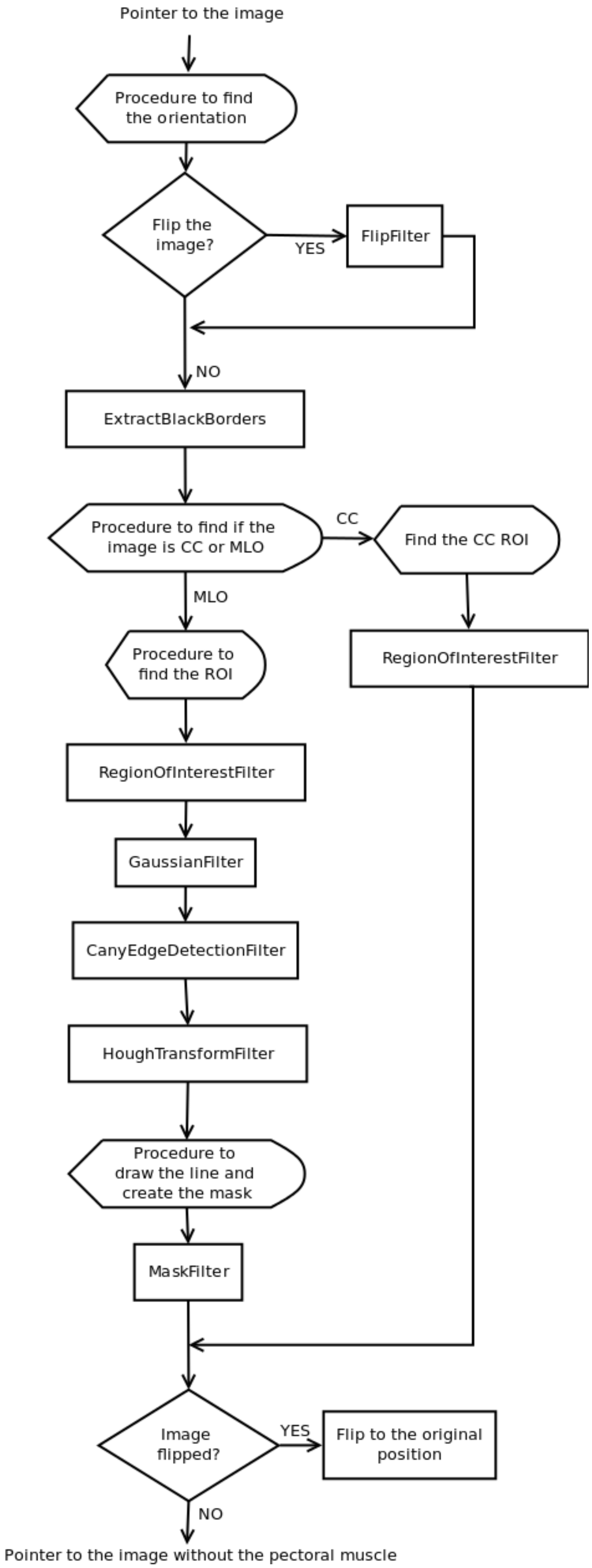

Pointer to the image
Procedure to find the orientation
Flip the image?
YES
FlipFilter
NO
ExtractBlackBorders
Procedure to find if the image is CC or MLO
CC
Find the CC ROI
MLO
Procedure to find the ROI
RegionOfInterestFilter
RegionOfInterestFilter
GaussianFilter
CanyEdgeDetectionFilter
HoughTransformFilter
Procedure to draw the line and create the mask
MaskFilter
Image flipped?
YES
Flip to the original position
NO
Pointer to the image without the pectoral muscle

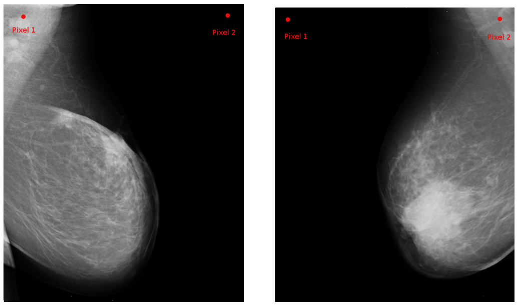
Figure 4.6: Flow diagram of the whole procedure for the pectoral muscle removal.

The used procedure involves several pre-processing procedures that will be explained below.

4.2.2.3 Finding breast orientation

First of all, we had to find the breast orientation in order to simplify further segmentation steps, such as location of the pectoral muscle and the region of interest (ROI). Orientation will also determine if the image is MLO or CC, even though if it's CC it isn't necessary to flip the image because it is not necessary to remove the pectoral muscle because it is not present in the image. Our images are always MLO or CC and we are assuming that the image is not upside down.

We use a simple orientation algorithm that was utilized by Chandrasekhar[6]. We extract a pixel near the corners and the pixel with brightest value will determine the orientation of the breast. We work with right orientation, so we've extracted two values of pixels from the images, as it is seen in the figure 4.7, and use the procedure described next:



(a) (b)

Figure 4.7: Breast MLO mammograms: (a) Right orientation (b) Left orientation

If $Pixel_1 > Pixel_2$ then orientation=right → No changes are needed.

else $Pixel_1 < Pixel_2$ orientation = left → Change orientation is needed.

else $Pixel_1 = Pixel_2$ image is CC → No changes are needed(figure 4.8).

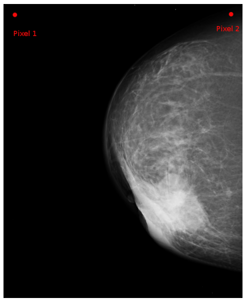


Figure 4.8: CC mammogram image

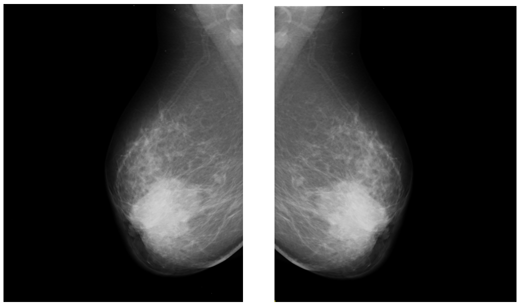
4.2.2.4 Flip the image

We performed that step using the filter present in ITK itk::FlipImageFilter that flips an image across under specified axes. The flip axes are set via the method SetFlipAxes(array). The image is flipped across axes for which array[i] is true. In this case we flipped across the X axis so the array is set to:

flipArray[0] = 1; // X axis

flipArray[1] = 0; // Y axis

In the figure 4.9 we can see the results of the flip process.



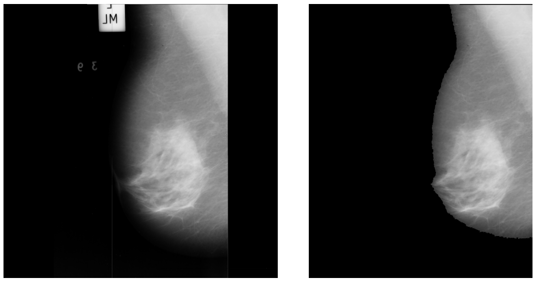
(a) (b)

Figure 4.9: Breast MLO mammograms: (a) Left orientation ready to flip (b) Right orientation flipped

4.2.2.5 Black border extraction

Once the image has been flipped or not, we realized that in mini-MIAS database, the images appear with a black border that difficult the segmentation procedure. So, we performed a function in charge of removing this black border. The main reason was due to we need to place our region of interest in the left up corner appearing the pectoral muscle as a triangular region.

Using image iterators, the algorithm iterates over the image and skips all the black pixels until find a non zero pixel. Then, create a region of interest starting from the found point and with the image dimensions. Finally, applying a RegionOfInterestFilter and obtaining a pointer to the new image without borders. A resultant image after the use of this function can be seen in the following figure 4.10:



(a) (b)

Figure 4.10: Black border extraction: (a) Original image (b) Image output without borders

4.2.2.6 Extract the Region of interest (ROI): RegionOfInterestFilter

Once black borders are extracted, it's time to find the perfect ROI to create the input for the muscle segmentation algorithm. We have tested the full image as a ROI, but in this case, the Hough transform was unable to find the correct line that forms the edge of the breast. So, thus we have placed the pectoral zone as a triangular region in the left up corner and after removing the black border we are able to find the ROI that contains all the pectoral muscle.

The used procedure creates a rectangular ROI. In the x axis, we have set a dimension until the algorithm finds the first pixel with value equal to zero. In the other hand, to find the y axis was a bit more complicated because not in all the images the pectoral muscle have the same dimension and we had to establish a general procedure. Finally, we have set the y dimension of the ROI equal to the half of the image's dimension. An example of the selected ROI is seen in the figure 4.11(a).

4.2.2.7 Enhance the image and noise removal: DiscreteGaussianImageFilter

After finding the ROI, there is necessary to enhance the quality of the image, smoothing the contrast and remove possible noise in the image in order to make easier the task of the edge detection filter. He have chosen the ITK:DiscreteGaussianImageFilter to remove the noise from images. It is implemented as a convolution with a Gaussian kernel taking advantage of its separability.

4.2.2.8 Edge detection: CannyEdgeDetectionImageFilter

The input of the Hough transform is the output of an edge detection filter. For perform this function, we have chosen the CannyEdgeDetectionImageFilter. This filter is widely used for edge detection since it is the optimal solution satisfying the constraints of good sensitivity, localization and noise robustness. As an input, this filter has the output of the Gaussian filter. The objective of this filter is to get an output image with all the possible edges in the image. An example of the output's filter can be seen in the figure 4.11(b):

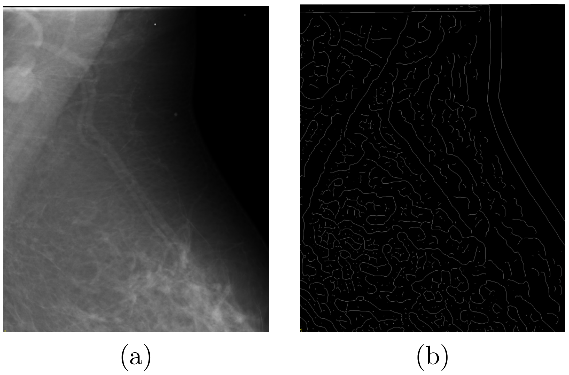


Figure 4.11: (a) Selected ROI for the edge detection input. (b) Edge detection output.

4.2.2.9 Hough transform

We used the Hough transform because it is a widely used technique for detection of geometrical features in images. We want to identify the line that separates the muscle from the breast in mammograms. Our segmentation algorithm generates a straight line approximating the pectoral edge.

To increase the speed of the algorithm and to find the correct line, we apply the procedure to remove the pectoral muscle to a reduced part of the image, the region of interest (ROI), and then expand the result to the whole image. The width and height of the whole image are denoted by (n_x, n_y) respectively. R is the initial region of interest. The straight line AB is an approximation to the pectoral edge. The end-points of the breast border are C and D. In the following figure 4.12 we can visualize an example of the Hough implementation and the commented parameters:

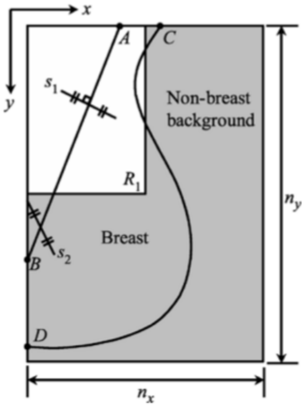


Figure 4.12: Hough example

Other techniques as Know[17], uses this straight line as the input to iterative cliff detection and the estimation cycle repeated. The second part of their algorithm is iterative cliff detection in which the straight line is refined to a curve that more accurately depicts the pectoral margin. In our approach, we kept with the initial straight line obtained with the Hough transform.

Once the straight line is found, we have implemented a procedure to create a black image with the line drawn in white. Furthermore, the found line in the ROI is extrapolated to the image global dimensions. After drawing the line, we created a mask to apply to the original image with an image with the muscle region in black pixels, because is the region that we want to remove, and the rest, the part that we want to keep, in white pixels. Finally, we applied the MaskImageFilter and we finally obtained an image without the pectoral muscle. Then, it is possible to save the output image using the ITK writer.

In the figure 4.13 we can observe an example the whole Hough procedure with each one of the commented outputs:

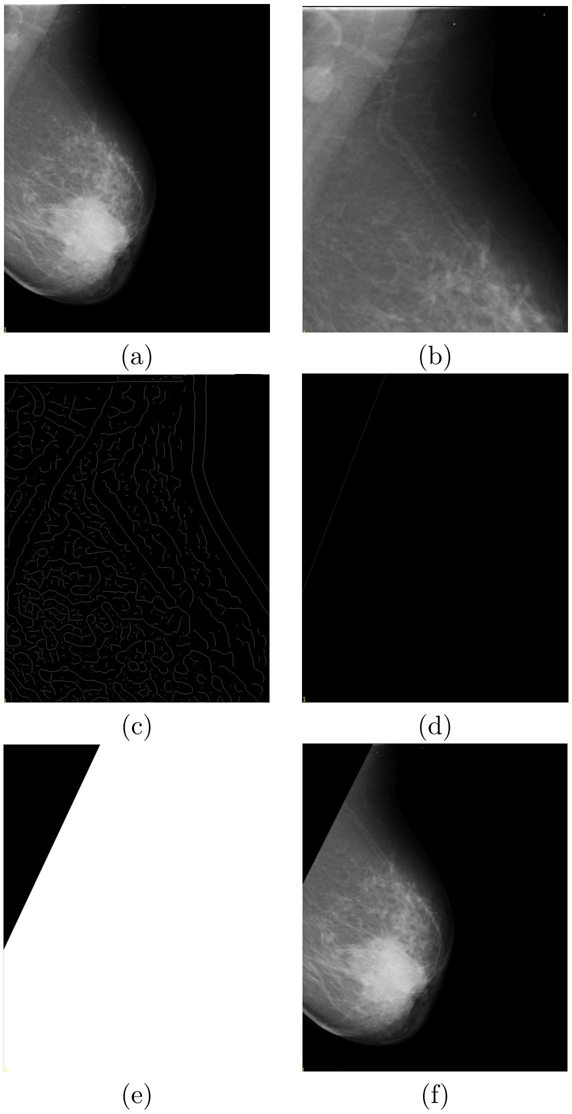


Figure 4.13: Hough transform procedure: (a) Original image, (B) ROI extracted, (c) Canny output, (d) Line found Hough, (e) Mask applied, (f) Muscle segmented

Chapter 5

Methodology for Density Classification

5.1 Random Sub-windows and Decision Trees

The method is based on random sub-window extraction of each image, build a model based on an ensemble of extremely randomized trees with the previously extracted subwindows from the training images and testing this model using the extracted samples from the tests images and propagating them through the ensemble of trees. The final class will be obtained averaging the outcomes of each tree.

We have decided for this algorithm because it doesn't need any other information or variables that the simple value of the pixes. The extracted sub-windows form a 256 feature vector that contains only the value of these pixels inside the window. Thanks to this generality, he have adapted the algorithm to our specific database and we have added some functionalities in order to create a cross validation method or to visualize the results. We have applied to our database and to mini-MIAS database, obtaining promising results comparable to the actual state of the art.

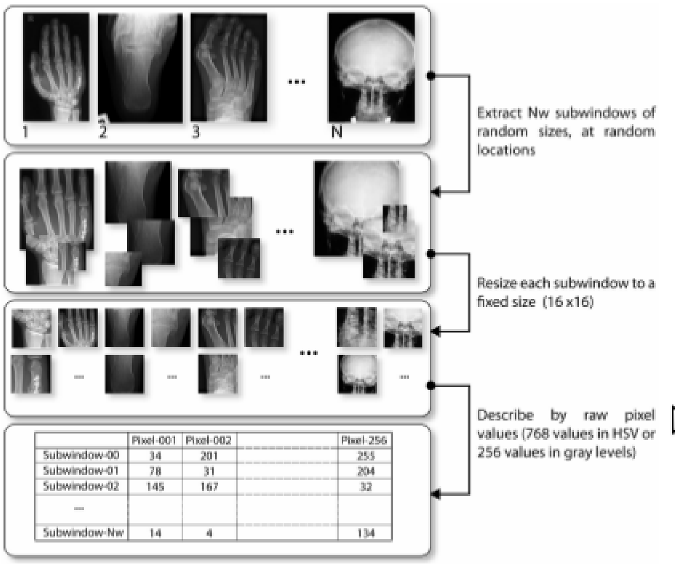
The involved steps in this procedure are described below.

5.1.1 Database Sampling and Feature Extraction of the Random Sub-windows

Firstly, the algorithm extracts a large number (N_w) of square sub-windows, possibly overlapping, of random sizes and at random positions from training images. Each sub-window size is randomly chosen between 1×1 pixels and the minimum horizontal or vertical size of the current training image. The position is then randomly chosen so that each sub-window is fully contained in the image. Sub-windows are resized to a fixed scale (16×16 pixels) and encoded by their gray-value pixels creating a feature vector. This process is generic and can be applied to any kind of images.

Each sub-window is described by a feature vector of 256 integer values corresponding with the value of each pixel conforming the square sub-window 16x16 in case of gray level images. For color images, the feature vector contains HSV (hue-saturation-value) values resulting into 768 positions. The number of extracted sub-windows in the training and testing step can be parametrized and add a variability in the results. Some results as recognition rate, computational time or number of nodes in a tree are very dependent on this parameters.

The procedure is shown in figure 5.1.


1
2
3
...
N
Extract Nw subwindows of random sizes, at random locations
...
Resize each subwindow to a fixed size (16 x16)
...
Describe by raw pixel values (768 values in HSV or 256 values in gray levels)

	Pixel-001	Pixel-002		Pixel-256
Subwindow-00	34	201		255
Subwindow-01	78	31		204
Subwindow-02	145	167		32
...				
Subwindow-Nw	14	4		134

Figure 5.1: Extraction of random sub-windows described by their pixel values.

The input of the algorithm is a text file that contains a row for each of the images in the database and its real class. The image's format used by the algorithm is .pgm. It is also possible to specify the wanted number of samples per image.

We have tested the algorithm using different number of samples used to train or to test the model. In the table 6.7 we will compare the obtained results with different number of samples.

5.1.2 Dividing sets and Cross validation

We added a procedure to implement a cross-validation method in our algorithm. It is a technique for assessing how the results of a statistical analysis will generalize to an independent data set and estimate how accurately a predictive model will perform in practice.

The goal of cross-validation involves partitioning a dataset into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions and the validation results are averaged over the rounds.

In K-fold cross-validation, the original sample is randomly partitioned into K subsamples. Of the K subsamples, a single subsample is retained as the validation data for testing the model, and the remaining K-1 sub-sets are used as training data. The cross-validation process is then repeated K times, with each of the K subsamples used exactly once as the validation data. The K results from the folds then can be averaged to produce a single estimation.

Before creating the partitions, the data is randomized in order to introduce different combinations of samples into training/validation thereby providing the required perturbation of the input space. We have used a simple randomization process of the original data set together with a 10-fold cross-validation method. The 10-fold cross-validation method trains an individual classifier for each fold using 90% of data for training and 10% for testing. Each partition used for training and testing in each iteration is always different from the others.

5.1.3 Tree building

At the learning step, a model is automatically built by machine learning using the extracted sub-windows from training images. The input of the algorithm are the N_w training sub-windows described by 256 integer pixel values and its discrete output class. Then, an ensemble of decision trees is built using the information of the extracted subwindows from the training images. The learning procedure can be seen in the following figure 5.2.

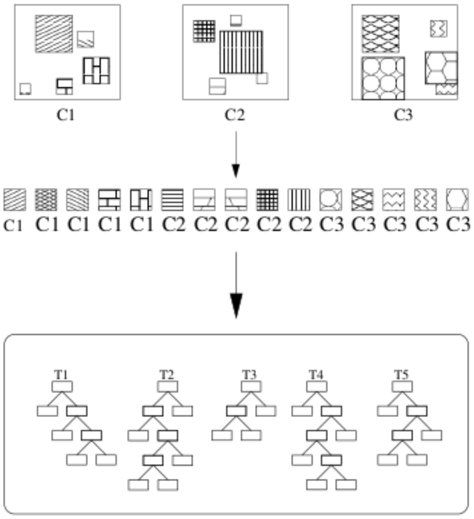


Figure 5.2: Learning model and building the ensemble of extra-trees.

The method builds many extremely randomized trees and its split thresholds at internal nodes are selected randomly, not based in any measure, distance or score. Each decision tree is grown until it perfectly classifies the training sample. This method is faster than other ensemble method due to of its randomization.

The algorithm begins selecting the samples stored for the learning step, calculating the entropy of the parent node, choosing a random feature (random permutation between all the possible features) to be tested, a random threshold and calculates the score of the node.

To split the elements in each node uses a vector, called rind, that indicates the indices of the samples to be considered in the learning set during the building process. Each time a node is split, this vector is updated with the indices of the elements remaining in the current node.

To compute the threshold, the maximum and minimum values of the learning set column corresponding to the chosen feature are computed. Then, the threshold is calculated through the equations described below to fit it in the range $[f_{max}, f_{min}]$:

$$th = rand(1,1) * (f_{max} - f_{min}) + f_{min}$$

where f_{max} and f_{min} are the maximum and minimum values of the subset.

$$f_{max} = max(lset(rind, f))$$

$$f_{min} = min(lset(rind, f))$$

To calculate the score, it follows the next diagram that shows the figure 5.3:

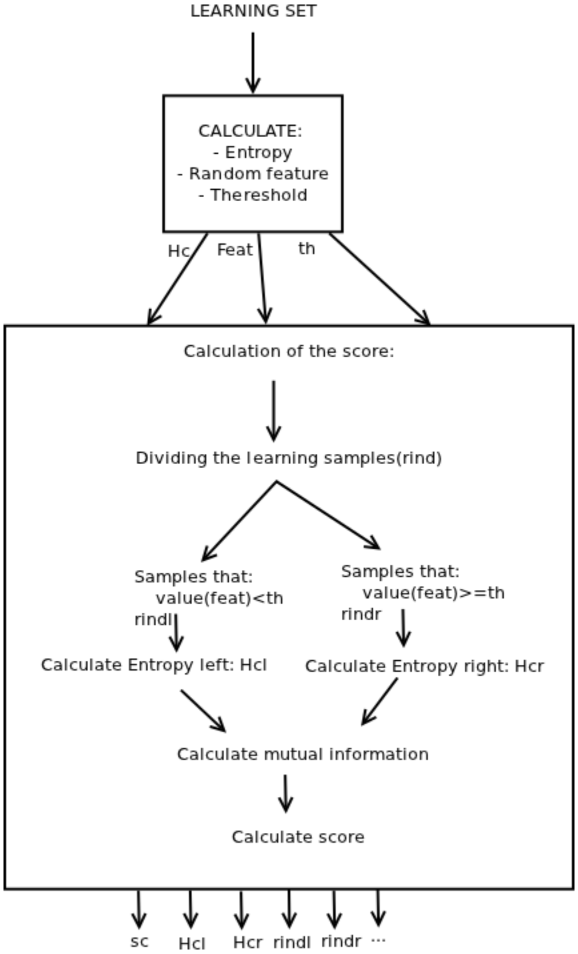


Figure 5.3: Flow diagram to understand the calculation of the score in a node

Firstly the ensemble of samples to be considered is split according if the value of the learning set corresponding to the random feature is greater or minor than the specified threshold. Here is when the vector rind is split in rindr(5.2) and rindl(5.1) indicating the indices of the remaining positions in the learning set. The realized checking is:

$$rindl = rind(lset(rind, f) < th); \tag{5.1}$$

$$rindr = rind(lset(rind, f) >= th); \tag{5.2}$$

Then, is computed the entropy of the sub-sets following the formula 5.3, the mutual information using the equation 5.4, where the probability of belonging to each side is easily calculated as the number of samples in the actual side by the total number of samples, and finally, the score is computed with the equation 5.5.

$$H = \sum_{i}^{n_{samples}} -pr_i \cdot log_2 i \tag{5.3}$$

$$I = H_c - pr_l \cdot H_{cl} - pr_r \cdot H_{cr} \tag{5.4}$$

$$s = \frac{2 \cdot I}{(H_c + H_s)} \tag{5.5}$$

where Hc is the classification entropy of the learning set before splitting and Hs is the split entropy.

The algorithm ends when all the samples in a node belong to the same class or it has been specified a minimum number of elements in a node to initiate a split different than two. There is done for each of the 10 built trees.

5.1.4 Testing trees

In this step, the learned model is used to classify the extracted sub-windows of a test image. The extracted sub-windows from the training images are no longer used after training the model, and can be discarded.

Each sub-window extracted from a test image is simply propagated into each ensemble of trees. The output of each tree is a conditional class probability for each sub-window. All the predictions are averaged and the final class is the largest aggregated probability and is assigned to the image. The mentioned procedure is illustrated in the figure 5.4.

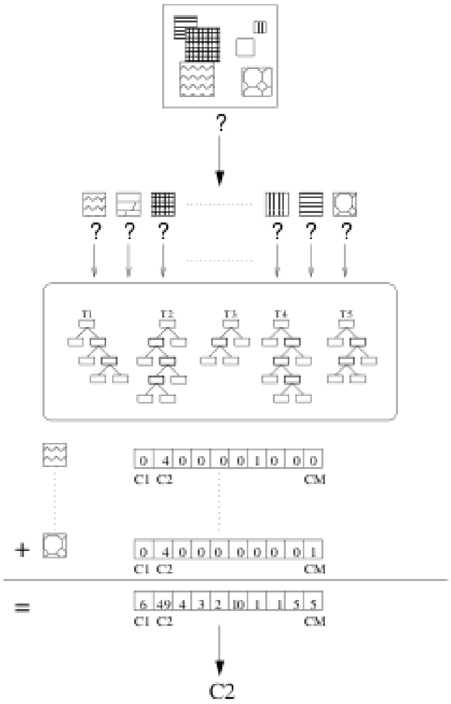


Figure 5.4: Recognition step where the random sub-windows are propagated through the ensemble of trees. The majority class is then assigned to the image.

The classification result is stored in a variable that gives the classification of each image. The classification results are always given in the same way as it can be observed in the table 5.1:

1 : NC	NC+1 : NC+NC	2NC+1
Score of each class	Decreasing sorted list of best classes candidates	True class

Table 5.1: Results of the classification step

5.1.5 Presentation of the results

In order to visualize better the results of the procedure we have implemented a procedure that shows the probabilities of recognition, its variance and the confusion matrices with the percentage of recognition for each class.

Confusion matrices should be read as follows: columns indicate the real class of the object to be recognized and rows indicate the class that it has been found. This matrix is useful for observing which classes have miss-identified items as other classes. So, the good results are placed in the diagonal, where it matches the real and the found class, and in the other positions it is shown the relation between the wrong classification of an image belonging to an specific class and recognized as another one.

5.1.6 Improvements in the algorithm

In order to improve the performance of the algorithm in our study case of density mammogram classification, we have added some new features as the possibility to test new parameters in each node in the moment of the tree building or adding texture features to the feature vector.

5.1.6.1 Different operation modes

First of all we thought that might be useful to add the functionality to choose between three possibilities to test in each node in the learning and testing step. In each node, a simple value is compared to a random threshold and the simple answer if the value is greater or minor that the threshold can propagate the sample until reaching the final or leaf node. The implemented and measured tests are the following ones:

1. Absolute value of the pixel corresponding to the feature used in this node (already implemented).
2. Pixel difference value between the value corresponding of the feature used and other feature chosen randomly.
3. Possibility to compute a test according to the maximum between two pixels.

The operation mode it will be controlled by a input variable and the test done in each node will be done according to the selected mode. In the testing, it will also be done the same test in each node that in the training to be coherent with the algorithm.

5.1.6.2 Adding new features to the feature vector

Feature scale:

We increased the dimension of the feature vector in one position for adding a feature related to the scale of the extracted sub-window. Before being resized to 16x16, the size of these sub-windows is chosen randomly. So we have decided to compute the scale between the size of this sub-window with the size of the image, computed as:

$$feature_{scale} = \frac{subwindow_{dim}}{image_{dim}}$$

This feature gives us a rate between the extracted random sub-window and the dimension of the image, giving us a measure to compute if the sub-window are comparable to the image dimension or if it is so small to give us some information about the possible classification.

Texture features:

Furthermore, we have thought that adding texture features in our algorithm might help in distinguishing between the more similar classes. Texture classification is responsible for segmenting a textured image into several regions with the same texture features. Texture features can be applied in three main different ways: statistical, structural and spectral. In statistical approaches, texture statistics such as the moments of the gray-level histogram, or statistics based on gray-level co-occurrence matrix are computed to discriminate different textures. For structural approaches, “texture primitive”, is used to form more complex texture pattern by grammar rules which specify the generation of texture pattern. And in spectral approaches, the textured image is transformed into frequency domain[38].

For biomedical images not all the methods are available and due to the characteristics of the pixels in the texture pattern are not similar everywhere, we thought that the statistic approach might help in the classification step. In our algorithm, there are some texture features that they are still underlying in the implementation, like the mean(average intensity). So, we have thought about which features, that they are not already implemented, and could help in distinguishing between density classes.

Our extracted features, as in the work of Seenapa H. et al.[38], are based on the statistical features and use the mathematical model for the moments to compute them. However the used expressions are different and we are not going to use the information of intensity histogram.

These texture features are added in the feature vector giving them a certain weight related to the absolute value of the pixels. We have added some positions of each extracted feature to give them more importance or less in moment of the classification. He have tested the results using 1 and 10 positions of the vector. He have tested with texture features of the random extracted sub-windows before resizing them and of the whole image. The results are presented in the table 6.12.

He have calculated these features in the moment of the random subwindow extraction and it was necessary to compute its histogram, mean and variance to compute them. The used expressions to compute the mathematical moments are:

- Moment n:

$$\mu_n = \frac{1}{N} \cdot \sum_{i=0}^{N-1} (z_i - m)^n$$

where z_i is the value intensity of the pixel, m is the mean and N are the size of the subwindow ($rows \cdot cols$).

- Mean

$$m = \frac{1}{N} \cdot \sum_{i=0}^{L-1} z_i$$

- Variance:

$$\sigma^2 = \frac{1}{N} \cdot \sum_{i=0}^{L-1} (z_i - m)^2$$

- Standard deviation:

$$\sigma = \sqrt{\sigma^2}$$

- Smoothness: measure of the relative smoothness of the intensity region.

$$R = 1 - \frac{1}{(1 + \sigma^2)}$$

- Skewness: measure of pdf’s symmetry.

$$\frac{\mu_3}{\sigma^3} = \frac{1}{N} \cdot \frac{1}{\sigma^3} \cdot \sum_{i=0}^{L-1} (z_i - m)^3$$

- Kurtosis: measure of the "peakedness" of the pdf

$$\frac{\mu_4}{\sigma^4} = \frac{1}{N} \cdot \frac{1}{\sigma^4} \cdot \sum_{i=0}^{L-1} (z_i - m)^4$$

- Uniformity: measure of the uniformity of the measures based on the Coefficient of Uniformity (CU).

$$U = 1 - \frac{\sigma}{m}$$

In the chapter results we are going to see the performance of all this tests.

Chapter 6

Results

6.1 Tests on mini-MIAS Database

We have tested our segmentation and classification algorithm in a public well known and very referenced database, called mini-MIAS, which was explained previously, to compare our results with the state of the art.

The results of the segmentation part will be presented in two different sections, one dedicated to the results of the pre-processing step of removing labels, artifacts and background from the original image, and the other related to the ability to remove the pectoral muscle from the breast. The results of the first part will be done by a simple probability, as the number of images with the correct label removal divided by the total amount of images, and the second one will be presented subjectively observing if the result image is correct according to the amount of pectoral muscle showed in the original one.

In the other hand, the classification results are calculated, taking into account a lot of parameters and the results are presented in terms of probability, variance, computational time, number of nodes in the trees or confusion matrices.

6.1.1 Pre-processing, labels and artifacts removal

This section will show the results of segmenting the image from background, labels, scanning artifacts or tapes.

We also have implemented other pre-processing steps as noise removal with Gaussian filters, determine the breast orientation, finding the initial seed and the correct thresholds, still underlying in the performed tests. Also we implemented a procedure to extract the black borders that have mainly all images, however, when we applied this kind of images to the classification step, we realized that the results were worse than we expected because the removed part were different in each image and the resulting dimensions were also different, getting worse the classification results.

In the following image, we can observe three of the main types of noise that are willing to remove from our images: high intensity noise regions including the bright rectangular labels, low intensity labels and high intensity scanning artifacts. This image corresponds to the mdb149 of mini-MIAS database. The labels in the MIAS database images are at several different angles and in some cases only part of the label is visible on the image.

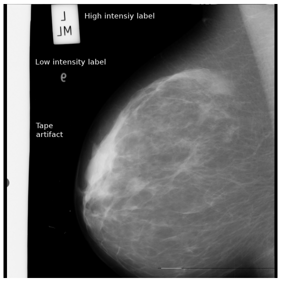


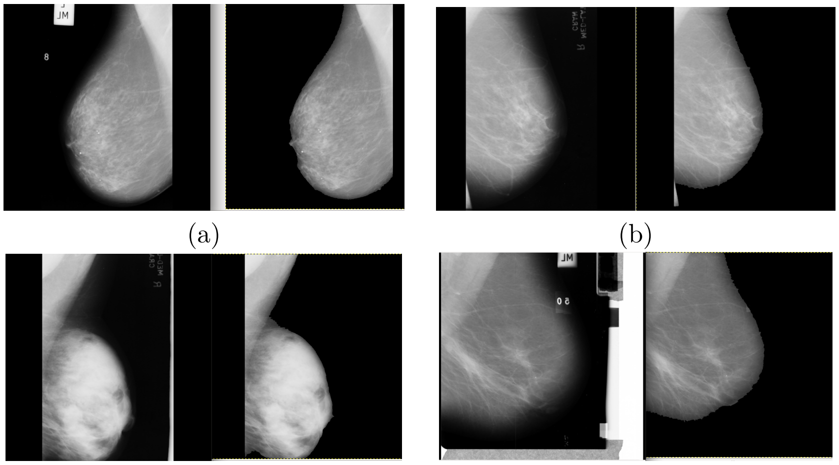
Figure 6.1: Image mdb149 that shows the possible types of noise in the database.

In the following table we will be able to observe the results of the different pre-processing implemented methods over the 322 images of the mini-MIAS database. This database contains 162 images with high intensity rectangular labels and 30 images with bright scanning artifacts present.

Type of noise present on the image	Rate of removal	Example
High intensity rectangular label	100%	Figure 6.2(a)
Low intensity label	100%	Figure 6.2(b)
Scanning artifact	100%	Figure 6.2(c)
Tape artifact	99,68%	Figure 6.3

Table 6.1: Results of the segmentation pre-processing step

The examples of the commented results can be shown next:


(a)

(b)

(c)

(d)

Figure 6.2: Results in the labels processing: (a) High intensity label removed mdb057 (b) Low intensity label removed mdb061 (c) Scanning artifact removal mdb068 (d) Removed all labels problems mdb274.

We can observe that the three first figures show each one of the commented labels problems, which we could remove perfectly. In addition, one of the images of the database, mdb283, contains all the possible labels that we wanted to remove and our algorithm has managed to do it.

In the following figure 6.3 we can observe the only error that we have obtained with our removing labels algorithm. The label was not removed due to a tape artifact has a part placed into the breast and a part into the image background, preventing it from being separated.

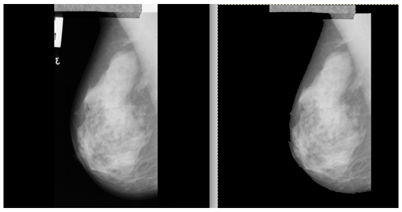


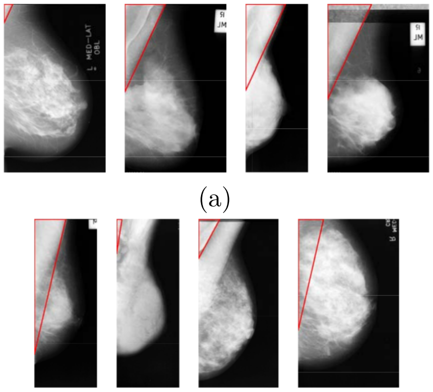
Figure 6.3: Image mdb283 that shows the unique problem encountered in the removing labels procedure.

The obtained results can be compared with the work of Martin Masek in its Doctoral Thesis[22], that obtained a removal rate of 99.38% (all unless mdb006) for square high intensity labels and a removal rate of 96.67% for the bright scanning artifacts (image mdb149). For the 162 labels present in the mini-MIAS database, he had obtained a 81.48% (30 img. not removed) of removal rate.

In conclusion, our algorithm gets better results that the one commented in the work of Martin Masek obtaining a higher rate of removal labels. In terms of computational time, it took 28 minutes to compute all the 322 images, so 5.217 sec/img.

6.1.2 Muscle removal

In this particular section we are going to show the obtained results of the breast muscle removal segmentation. We will see that the results are no as good as in the previous step because the muscle has some peculiarity features that increases the difficulty to obtain good results. This features, previously commented and observed in the figure 6.4, are the different size and different intensity pixel values in the images present in the database.



(a)

(b)

Figure 6.4: (a), (b): Different sizes and intensities of the pectoral muscle in breast mammograms.

The pectoral muscle appears in the mediolateral oblique view as a bright triangular region in one corner of the image, so our procedure has to be capable of:

- Find if the image is a MLO or CC.
- Flip the image just in case the orientation is different to the required.
- Find the line that separates the pectoral muscle from the breast tissue
- Apply a mask to separate the muscle from the breast.

The pectoral edge represents a line between the pectoral region and the fatty breast tissue. Our algorithm, following the flow diagram showed in the figure 4.1, takes a region of interest (ROI) before applying the edge detection filter and the Hough transform algorithm, in order to have a lower region to find the separation line and obtain better results. However, the encountered problems in this step are mainly due to the selection of the ROI's size. The found line depends on the size of this region and it was impossible to find a ROI that worked perfectly for all the images.

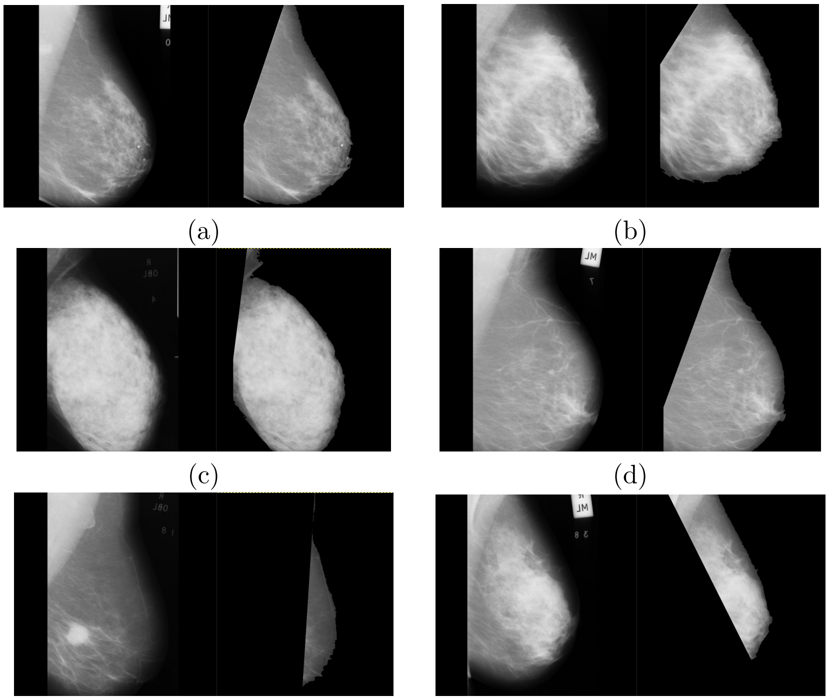
Results are presented from testing on the entire database of 322 images. The tested algorithms provides a degree of success with the straight-line that fits, producing visually acceptable results with the pectoral muscle edge. As we have explained, our results are based on subjective analysis of the resultant image at first glance, compared with the original one. The automatic procedure has lower rate of success than the removing labels algorithm. However, it is possible to improve the results of the wrong images manually, only changing the value of the selected ROI.

The table 6.2 shows the experimental results after applying our method to the mini-MIAS database. Also we can visualize some corrected problems after the manual manipulation of the ROI in the figure 6.6.

Subjective evaluation	**Result**	**Example**
Good	186 (57.76%)	Figure 6.5 (a),(b)
Acceptable	93 (28.88%)	Figure 6.5 (c),(d)
Unacceptable	43 (13.35%)	Figure 6.5 (e),(f)

Table 6.2: Results of the segmentation pre-processing step

In the following figure 6.5 we can observe one example of each case showed in the previous table.



(a) (b) (c) (d) (e) (f)

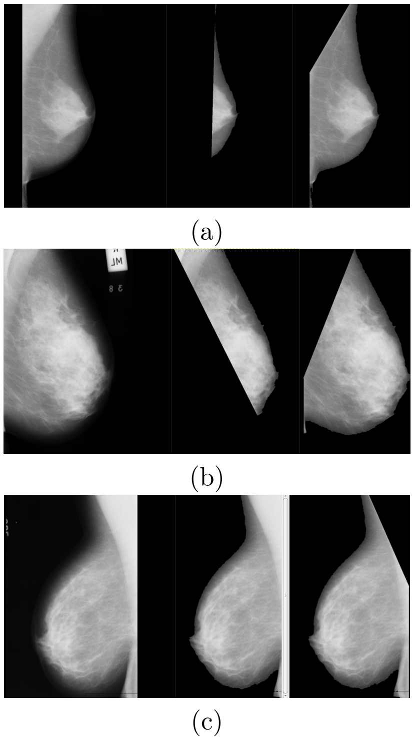
Figure 6.5: Results of the muscle removal algorithm: (a) (b) Good pectoral removal, (c) (d) Acceptable removal, (e) (f) Unacceptable pectoral removal.

First of all, we can observe that using this algorithm we encountered three main problems that caused the failing of the procedure in some images. These three main problems will be explained next.

1. The removed portion of muscle is too big according to the original image. In this case, the Hough transform fail in its objective to find the line that separates the pectoral muscle and the breast and finds another line that removes too much portion of the breast. It happens in the images: mdb6, 28, 40, 41, 43, 61, 72, 73, 74, 117, 165, 166, 177, 233, 287, 288, 302.

2. The found line of the Hough transform is wrong according to the situation of the pectoral muscle in MLO mammograms. The finding of this line causes the removal of a wrong portion of breast that don't correspond to the pectoral muscle. It happens in images: mdb98, 134, 152, 154, 162, 184, 236, 280, 304, 305, 318, 313.

3. The Hough transform don't find the line that separates the pectoral muscle from the breast or the found line isn't sufficient to remove all the pectoral muscle portion. That happens in images: 15, 49, 121, 206, 217, 218, 226, 235, 263, 264, 283, 284, 295, 315.



(a)

(b)

(c)

Figure 6.6: Wrong and corrected results in the muscle removal algorithm: (a) Too much portion of pectoral muscle removed (b) Incorrect line found (c) Line not found.

To correct the wrong results, it has manually changed the size of the selected ROI in order to facilitate the task to the Hough transform. Referring to the computational time, it is a fast procedure that lasted 22 minutes to process all the 322 images of the mini-MIAS database, approximately 4 sec/img.

6.1.3 Classification step

Many tests have been done in this step in order to find the parameters to achieve the best possible results. In this section we are going to explain them and show the obtained outcomes.

We have tested the parameters that could improve the result of the algorithm's performance and increase the recognition rate. Among them are the type of used image, the number extracted subwindows, the number of samples used in the learning or in the testing step and the minimum number of elements to initiate an split.

With all these tests, we have obtained different values to compare the performance of each experiment. The results are obtained referring to probability of recognition (shown in confusion matrices), variance in the result, number of nodes in each tree and computational time required to finish the experiment.

To conclude, we have also done an experiment to assess the possible variability of a class decision. We wanted to ensure the validation of the method for its final goal: classification for diagnosis. Due to randomization of the classification algorithm, the trees computed during a learning phase will be different to another learning phase, even if the learning set is the same, so, he have tested the possibly if an unknown image would be classified differently if it was tested against these two different trees.

6.1.3.1 Initial results

Referring to the type of the image, we have tested different input images in order to check with which one we could obtain better results. We are going to present them sorted in chronological order.

In the first testings with mini-MIAS database, we have tried with the original images, images without the labels, artifacts and black borders and images without the pectoral muscle. We have tested with 100.000 and 10.000 samples in the learning step obtaining the following results:

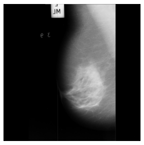
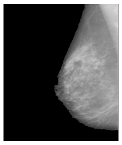
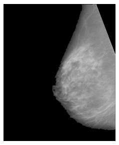
Type of experiment	Image Example	Samples	Recognition	Confusion matrix				
Original	 (a) Original image mdb007	100K	63.44%	Breast density		True class		
						F	G	D
				Predicted class	F	87	22	4
					G	10	25	16
					D	7	56	91
		10K	61.87%	Breast density		True class		
						F	G	D
				Predicted class	F	82	28	7
					G	9	19	8
					D	14	56	97
Processed labels and	 (b) Without labels and border	100K	55.94%	Breast density		True class		
						F	G	D
				Predicted class	F	85	20	5
					G	7	7	14
					D	14	76	87
black border extracted		10K	54.34%	Breast density		True class		
						F	G	D
				Predicted class	F	83	21	4
					G	10	7	15
					D	14	78	85
Processed labels, border	 (c) Without pectoral muscle	100K	64.0625%	Breast density		True class		
						F	G	D
				Predicted class	F	82	24	7
					G	16	44	25
					D	7	36	79
and muscle removal		10K	60.32%	Breast density		True class		
						F	G	D
				Predicted class	F	85	19	3
					G	3	2	3
					D	18	82	106

Table 6.3: First experiments with different types of images.

After analyzing the obtained results, we can see than the recognition probability decreased if less samples are used in the training step, as we expected. But, it also decreased using the images that had been preprocessed in order to remove labels, artifacts, black border or even the pectoral muscle, elements that in the beginning we thought that could disturb the final classification decision. Our first thought was that the images, after extracting the black border, didn't have the same dimension or even the same orientation. Then, we implemented a number of improvements in order to increase the results:

- Not remove the black borders from the images.
- Flip all images to have the same orientation.
- Test with original images, images without labels and artifacts and images with a region of interest (ROI).
- Test other type of experiment in each node in the moment of the building trees.
- Build the trees using soft voting instead of hard voting.

After performing these changes, he have found a big improvement using the images without extracting the black border, in other words, with the same dimension, finding the orientation of the images and flipping them if it is the case and using soft voting. In the table 6.4, showed below, it can be seen such a difference.

Type of image	**Different orientation**					**Same orientation**				
Original image	61.87%					70.31%				
	Breast density		True class			Breast density		True class		
			F	G	D			F	G	D
	Predicted class	F	82	28	7	Predicted class	F	88	19	5
		G	9	19	8		G	14	51	20
		D	14	56	97		D	4	33	86

Table 6.4: Results using the same image but with or without the same orientation

In the other hand, some of the new implemented tests didn't worked properly and didn't gave us better results that the previous ones. For instance the combination of pixel difference test and feature scale did not show any improvement. Even using the pixel difference test got worse our results.

To assess all the possibilities in order to find the one who gives better results we have implemented lots of tests changing the following parameters:

- Types of images: original, processed labels and artifacts and region of interest (ROI).
- Minimum number of elements in a node to initiate the split: 4, 8 and 50.
- Type of test done in each pixel: absolute value, difference between pixels or maximum between pixels.
- Add texture features to the possible tests with a specific weight in relation to the value of the pixels in the feature vector.

After all the performed tests we were capable of determine which configuration gave us better results.

6.1.3.2 Types of used image

We have tested some different kind of images to end up concluding that the best results are obtained with the entire original image that includes all labels, artifacts and background.

We have tested 4 types of images:

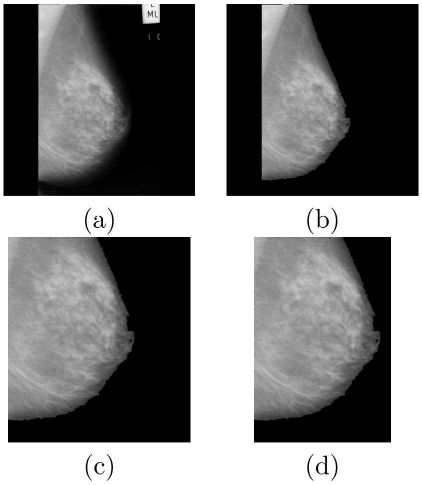
- Original images from mini-MIAS database, with all the labels, artifacts or background present.
- Images with segmentation process of removing labels and artifacts.
- Extraction of a region of interest (ROI) of a determined size:

  $$(Rows, Cols) = [\frac{Image_{size}}{3}, 2 \cdot \frac{Image_{size}}{3}]$$

  that covers all the breast dimension including also a bit of background.
- Extraction of a second ROI1 that includes less portion of background. Its dimension is:

  $$(Rows, Cols) = [\frac{Image_{size}}{3}, \frac{Image_{size}}{2}]$$

We can see an example of the four different tested images in the following figure 6.7:



(a) (b)

(c) (d)

Figure 6.7: Different type of used images: (a) Original Image (b) Pre-processed image (c) ROI extracted (d) ROI1 extracted

As we have said, the results using the original image without any kind of pre-processing are better than the others. In the following table 6.5 we can see an example of this conclusion using nsplit 4, 150 sub-windows per image, 100.000 samples for the learning step and 150 for the testing.

Type of image	Recognition rate
Original image	70.062%
Processed labels image	69.0625%
ROI	66.56%
ROI1	65.31%

Table 6.5: Results using different type of images.

We understand the decreasing recognition rate according to the decreasing dimension of the image as the loose of surface to extract randomly new sub-windows and not to have enough points to compare and decide how to classify the mammogram images. Furthermore in the ROI images is made clearer the difference between breast dimensions, affecting in the classification step.

6.1.3.3 Number of elements in a node to initiate an split

Another improvement that we have noticed in the recognition rate was the use of soft-voting instead of hard-voting. Using hard voting means that the build trees, split their nodes when the number of elements in a node were at least 2, in other words, the final leaf corresponds to one element labeled with the corresponding class. Testing this trees, the decision is taken by propagating each subwindow until it reaches the final leaf and the class corresponding to that leaf is assigned to this subwindow.

Soft-voting relies in building the trees but leaving more elements in the final leaf, setting the minimum number to initiate an split higher than 2 elements. The final decision on a specific class, after propagating the sub-window, relies in assigning to the class with more members in the leaf.

We have done many experiments changing this minimum number of elements in order to find the one that gave us better results. In the following table 6.6 we can see an example of the obtained results with different values of nsplit using images without labels, sampled with 150 samples per image, 10.000 samples in the learning, 150 samples in the testing and with absolute value test:

Nsplit	**Rate**	**Number of nodes (average)**
2	67.5%	13.500
4	70.93%	9.000
8	70.82%	5.300
10	70.62%	4.400
15	68.12%	3.300
20	68.43%	2.500
30	68.12%	1.800
40	68.12%	1.400
50	66.56%	1.200
100	63.75%	670

Table 6.6: Results using different nsplit

Another results related to this parameter are the number of nodes that form the trees and the computational time used to perform the experiment.

According to the previous table it can be observed that soft-voting results with nsplit from 4 to 40 are better than hard-voting ones. When the number of elements no initiate a split becomes greater the results got worse. It is also seen that, as we expected, the more elements in a node necessary to initiate a split, the less number of nodes the tree has.

6.1.3.4 Number of extracted sub-windows

We have done many tests without changing the number of extracted subwindows of each image in the sampling step, set by default to 150. Also the default values for the learning and testing set were 100.000 and 150 respectively.

Sampling the images with 150 subwindows (16x16) per image means creating a dataset of:

$$[\#total_{samples}, \#total_{features}] = [48300, 258]$$

where:

$$\#total_{samples} = \#samples_{img} \cdot \#img = 150 \cdot 322 = 48.300$$

However, if a number of samples is chosen for the learning and testing step, the algorithm, using the function GEURTS_select_samples, only selects the assigned number of samples from either the learning or the testing set. This functions is created to select only a smaller set of samples and decrease the computational time.

We thought that a good way to increment the recognition rate was increasing either the number of subwindows used to sampling the images and using all the samples in the learn and test sets to perform the experiments, although increasing the computational time and the number of nodes.

In the following table 6.7 we can observe the difference between the number of samples used in each experiment and how the recognition rate increases as the samples do. The type of tested image were the original, we have used the absolute value test and a nsplit equal to 4.

#Sub wind	**Data set**	**#Samp learn**	**Learn set**	**#Samp test**	**Test set**	**Rec. rate**	**Conf. matrix**				
150	48.300	10.000	10.080	150	4800	63.75%	Breast density		True class		
									F	G	D
							Predicted class	F	81	22	11
								G	19	42	20
								D	5	39	81
150	48.300	100.000	43.200	150	4800	64.68%	Breast density		True class		
									F	G	D
							Predicted class	F	83	24	7
								G	16	47	28
								D	6	32	77
1000	322.000	10.000	10.080	150	4800	68,43%	Breast density		True class		
									F	G	D
							Predicted class	F	92	17	6
								G	7	55	18
								D	7	30	88
1000	322.000	All	289.800	All	32.200	**76.25%**	Breast density		True class		
									F	G	D
							Predicted class	F	92	15	3
								G	8	60	17
								D	5	28	92

Table 6.7: Results using different subwindows and number of samples.

Due to the cross validation method and the fact that mini-MIAS database has 322 images we have divided the sets in order to have the same number of images in each one. So, we have included 32 images in each one. It means that in each experiment there are 2 images that don't take part in it. That images are chosen randomly.

6.1.3.5 Measures to assess the reliability of the classifier with the best obtained result

Precision and recall measures

Precision and recall are two widely used metrics for evaluating the correctness of a pattern recognition algorithm. Recall is then computed as the fraction of correct instances among all instances that actually belong to the relevant subset while precision is the fraction of correct instances among those that the algorithm believes to belong to the relevant subset. Precision can be seen as a measure of exactness or fidelity, whereas recall is a measure of completeness.

In a classification task, the **precision** for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). It follows the next equation:

$$Precision_{class_i} = \frac{\#correct\,detections_{class_i}}{\#total\,detections_{class_i}}$$

Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been). Recall or sensitivity is a measure of the ability of a system to present all relevant instances, so it is used for evaluating the completeness of results.

$$Recall_{class_i} = \frac{\#correct\,detections_{class_i}}{\#total\,elements_{class_i}}$$

In information retrieval, a perfect precision score of 1.0 means that every recognized result was relevant (but says nothing about whether all relevant documents were recognized) whereas a perfect recall score of 1.0 means that all relevant documents were recognized by the search (but says nothing about how many irrelevant documents were also recognized).

F-measure

F-measure considers precision and recall as a measure of a test's accuracy used in the field of information retrieval for measuring search, document classification, and query classification performance. The F-measure score can be interpreted as a weighted average of the precision and recall, where an F-measure score reaches its best value at 1 and worst score at 0. It follows the next equations:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

To give different weight to precision and recall, the F-measure was derived in the F_β measure that computes the effectiveness of the recognition with respect to a user who attaches β times as much importance to recall as precision and which follows the next equation:

$$F_\beta = \left(1 + \beta^2\right) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

Two other commonly used F measures are the F_2 measure, which weights recall higher than precision, and the $F_{0.5}$ measure, which puts more emphasis on precision than recall.

We are going to compute all this measures for the best result that we have obtained for each one of the different density classes in the mammogram images. The best results are remained below with the computation of the commented measures shown in the table 6.8:

<table>
<tr><th>Best results:</th><th>Classes</th><th>Precision</th><th>Recall</th><th>F</th><th>F_2</th><th>$F_{0.5}$</th></tr>
<tr><td rowspan="2">Recognition rate: <b>76.25%</b></td><td>Fatty</td><td>0.8363</td><td>0.8761</td><td>0.8557</td><td>0.8678</td><td>0,8439</td></tr>
<tr><td>Glandular</td><td>0.7059</td><td>0.5825</td><td>0.6382</td><td>0,49975</td><td>0,677</td></tr>
<tr><td>Confusion matrix:
<table>
<tr><td rowspan="2" colspan="2">Breast density</td><td colspan="3">True class</td></tr>
<tr><td>F</td><td>G</td><td>D</td></tr>
<tr><td rowspan="3">Predicted class</td><td>F</td><td>92</td><td>15</td><td>3</td></tr>
<tr><td>G</td><td>8</td><td>60</td><td>17</td></tr>
<tr><td>D</td><td>5</td><td>28</td><td>92</td></tr>
</table>
</td><td>Dense</td><td>0.736</td><td>0.8214</td><td>0.776</td><td>0,8027</td><td>0,7516</td></tr>
</table>

Table 6.8: Measures of precision, recall and F-measure for the best obtained results.

After analyzing the obtained results in the table, we can see that the worse results are got when the algorithm tries to classify the class 2 (glandular) either in precision and recall parameters. However, regarding to precision, we can see than the proportion between found elements and correct elements of class 2 are quite acceptable (70.59%) but the problem is that many times class 2 isn't correctly detected in relation to the total of elements belonging to this class (recall) with a rate of 58.25%. It is seen that the parameter F is nearest in cases of classes 1 or 3 than in class 2.

As we expected, F_2 and $F_{0.5}$ measures show that the result is better when gives more importance to the best parameter, precision or recall respectively.

6.1.3.6 Graphical overview of the obtained results

Regarding all the possible parameters that we have changed in order to improve the algorithm performance and the recognition results, we have been able to plot some charts that show the evolution of the results according to some parameters. We have done a total of 72 experiments varying the commented set of parameters to find the combination that gave us better results. The obtained charts are presented in the annex 8.2.

We are going to present 8 charts that describe the behavior of the four types of used images (original, without labels, ROI, ROI1) according to a set of tested parameters: nsplit (4,8,50), test performed (absolute value, abs.+pixel diff.+feat_sc, abs.+max.) and number of samples used for the learning and testing step (all, 10000 and 150). The obtained results are shown in terms of probability of recognition, variance of the result, number of nodes of the trees and computational time used to perform the experiments.

In the first chart (figure 8.1) corresponding to the original images and taking all the samples, we can observe that the best recognition rate achieved was 73.43% obtained with nsplit=4 and absolute value. This rate is decreasing as long as the nsplit is increasing or other test is used. In the other hand, is shown in the second chart that using only a set of the possible samples (figure 8.2), the test that gave us better results was the combination between absolute and maximum value of pixels with a rate of 72.18%. Referring to the variance in the results, is clearly observed that using the test 2 (abs.+pix.diff.+feat_sc), is obtained the highest value while with the other tests is similar. According to the number of nodes, as expected, the more nsplit the less number of nodes.

In the second pair of charts, corresponding to the images without labels and artifacts, taking all the samples, the best recognition rate is obtained also with the absolute value and nsplit equal to four (67.81%). However, taking only 10000 samples for learning and 150 for testing, we have obtained the best rate using absolute and maximum test (67.5%). Regarding to the other parameters conclusions, they are the same than in the previous chart.

In the performed experiments with a region of interest (figure 8.5 and 8.6), we can see that either with all samples or only with 10000 for learning and 150 for testing, experiment with maximum values is better than absolute pixel. However, these results are quite worse than using the original image.

Results using the smaller ROI were quite surprising because they have reached the best recognition rates that we had obtained with the whole original image. As in the first experiment, nsplit=4 and using all the samples gave us the best result (reaching the 73,4375% of success) with absolute pixel, nevertheless with nsplit 8 and 50 and using only a sub-set of samples, maximum between pixels experiment was the best (72,5% and 70,625% respectively with all samples and 72,1875% with a sub-set of samples and nsplit=4). The conclusions about the other parameters are the same that in the previous charts corresponding to the first extracted ROI.

6.1.3.7 Improvement 1: Type of test used in the pixel

We have implemented different possibilities to test when the trees are built and test. We have thought that this experiments could improve the recognition rate because might show a greater difference between a non-dense mammogram and a dense one. However, we were wrong, due to the randomness of the extracting subwindow, some of these tests didn't provide any improvement.

We have tested many combinations of the possible tests in each node, even giving some weights to force that on feature will be chosen more than the others, with two different kind of images and the obtained results are summarized in the following table 6.9:

Type of image/Type of test	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**(8)**
Processed labels image	69.68%	55.93%	32.81%	49.68%	60.93%	36.81%	34.37%	40.31%
ROI1	65.31%	31%	33.75%	34.37%	34.37%	36.81%	34%	33%

Table 6.9: Results using different experiments

And the legend of the performed tests in each node are:

1. Absolute pixel value.
2. Difference between two pixel values.
3. Feature scale.
4. Random combination between (1), (2) and (3).
5. Random combination between (1) and (2).

6. Random combination between (1) and (3).
7. Random combination between (2) and (3).
8. Random combination between (1), (2) and (3) but giving more weight to some feature.

After analyzing the results, the test that gave us better results was the absolute value of the pixels and it is easy to see that the results are always better using the whole image than a simple region of interest. Furthermore, the use of difference pixel alone or in some combination only deteriorates the results.

After realizing that the difference between pixel values didn't provide any improvement, we have decided to implement another test, the maximum value between two random pixels. In this case, the combination absolute pixel and maximum between two pixel values gave much better results than pixel difference and even in most of cases than the absolute pixel value. We can see in this results in the table 6.10:

Type of test	**Rate**		**Type of test**	**Rate**	
Absolute pixel	All samples	10.000 learn 150 test	Abs. + maximum	All samples	10.000 learn 150 test
nsplit4	73.43%	68,43%	nsplit4	72.5%	72.18%
nsplit8	71.56%	68.75%	nsplit8	72.18%	70%
nsplit50	66.56%	63.75%	nsplit50	68.12%	66.56%

Table 6.10: Results using different nsplit taking all the samples or 10.000 and 150 in learning and testing sets.

With this results we can affirm that the inclusion of the maximum pixel test even improve the recognition rate in most of the cases. In the situation that not all the pixels are selected for performing the learning and testing steps, improves all the results comparing to the absolute pixel ones.

6.1.3.8 Histogram information

In the following table 6.11 we can observe the differences between the different kind of images used in the classification step and their histograms. It is seen that after the process of labels, artifacts and background removal, the range of values 0-50, corresponding to the background, is removed and mapped to the 0 pixel value. Images of class 1, with the gray values more distributed show a peak around the gray mean value. Images of class 2 have more variability in their histograms because it's a mid-classification between class 1(fatty) and class 3(dense). Images belonging to class 3 show the most part of their pixels in the right side, indicating their high intensity.

Analyzing the histograms in the processed images (labels, muscle or ROI), we can see that the differences between classes 2 and 3 are less discriminative. Thus the results of classification are worse than using the whole images. The differences between images from class 1 and class 3 are more discriminative, so this is the reason why differentiating between dense and no-dense class gave us better results.

We can see in the histograms of the processed muscle images, than the peak near the high intensity values, correspondent to the pectoral muscle, has been removed from the histogram. In the image of class 3, the number of pixels near the high intensity values are decreased due to the pectoral muscle removal.

Thus the commented problems, we needed to add some kind of feature, test or procedure, that may help the discrimination between the more similar classes. Our first though was inserting texture features related to the histogram in the classification algorithm. In the following sub-section we are going to present the obtained results.

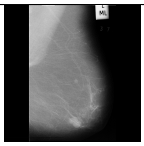
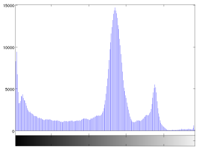
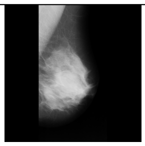
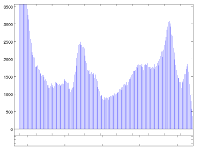
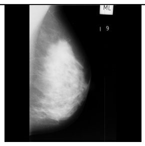
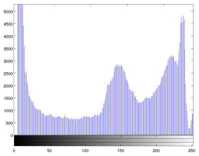
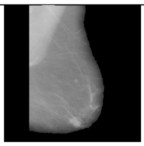
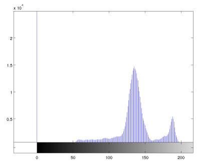
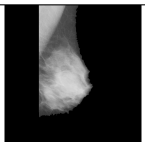
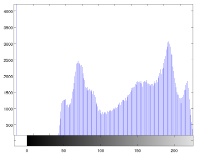
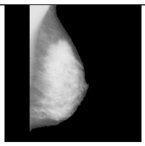
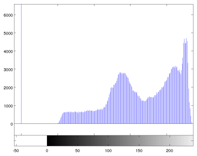
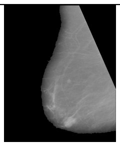
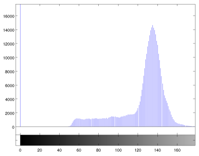
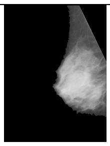
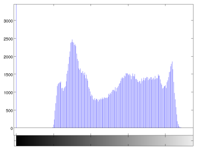
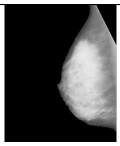
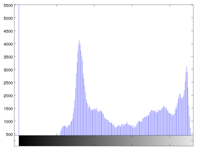
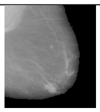
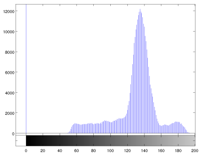
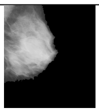
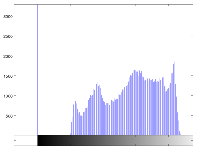
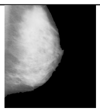
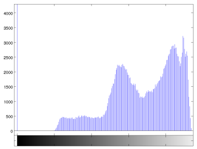
	Class1 mdb005	**Class2** mdb001	**Class3** mdb003
Original image	 	 	 
Processed labels	 	 	 
Processed muscle	 	 	 
ROI	 	 	 

Table 6.11: Histogram information about the different type of used images

6.1.3.9 Improvement 2: Adding more features to the algorithm: feature scale and texture features

We have obtained the following results adding texture features and different combination of parameters (#subwindows, number of positions for each feature), with the original images, nsplit 4 and the absolute value as the used test. We have also tested extracting the texture features either from the whole image or each one of the subwindows. They can be seen in the table 6.12:

Subwindows	Features	Rate	Weight	Confusion matrix				
				Breast density		True class		
						F	G	D
150	Whole image	70.31%	1 pos/feat	Predicted class	F	90	21	6
					G	9	52	23
					D	6	30	83
				Breast density		True class		
						F	G	D
150	Sub-window	72.5%	1 pos/feat	Predicted class	F	89	17	3
					G	11	56	22
					D	5	30	87
				Breast density		True class		
						F	G	D
150	Whole image	70.0%	10 pos/feat	Predicted class	F	89	18	6
					G	11	53	24
					D	6	31	82
				Breast density		True class		
						F	G	D
150	Sub-window	72.81%	10 pos/feat	Predicted class	F	92	20	6
					G	11	58	22
					D	3	25	83
				Breast density		True class		
						F	G	D
1000	Whole image	73.43.%	1 pos/feat	Predicted class	F	90	18	4
					G	10	59	22
					D	5	26	86
				Breast density		True class		
						F	G	D
1000	Sub-window	75.93%	1 pos/feat	Predicted class	F	91	12	4
					G	10	67	21
					D	5	25	85

Table 6.12: Results of using texture features extracted from the whole image or the sub-windows.

After analyzing the previous table, we can see that weighting the feature don't give us better results. Furthermore, it is shown that there are little difference between using the features extracted from each subwindow or from the whole image. Extract the features from each subwindow gives better results than to the whole image because it's more accurate and the features are computed locally instead of globally. These features are computed for each subwindow and a value is inserted to the feature vector for each line correspondent to the learning sample. In the other hand, when the feature is extracted from the whole image, the same value is added for all the extracted subwindows in the specific image.

As we expected, the more subwindows have been extracted the better results we have obtained. There also more difference between using the whole image or the subwindows because we have more features values that determine the final class of an image. The problem of using low number of subwindows and the features extracted from them is that there are a lot of low values corresponding to subwindows where borders or low portion of breast is extracted, whereas the high values, the ones that achieve a good classification difference, correspond to subwindows that surround great portion of breast. In the case that we have a lot of subwindows, the proportion of low and high values is less discriminative than with lower number of subwindows.

After observing the previous results and analyzing the values of the texture features for the extracted subwindows, we realized that in some cases, when the extracted subwindow corresponds to a square with all its pixels equal to 0, some texture features result like the skewness, were NaN due to the division by the variance. So, we thought that the classification algorithm doesn't know how to deal with this number. Finally, we have implemented a little checkout that outputs a 0 value instead of a NaN in these cases and the results of using texture features, as we have expected at the beginning, were better than using simple pixel values. They are summarized in the following table:

Subwindows	**Features**	**Rate**	**Weight**	**Confusion matrix**				
				Breast density		True class		
						F	G	D
150	Sub-window	**74.68%**	1 pos/feat	Predicted class	F	95	19	5
					G	8	60	21
					D	3	25	84
				Breast density		True class		
						F	G	D
150	Whole image	72.81%	1 pos/feat	Predicted class	F	92	20	6
					G	11	58	22
					D	3	25	83
				Breast density		True class		
						F	G	D
150	Sub-window	70.31%	10 pos/feat	Predicted class	F	90	21	6
					G	9	52	23
					D	6	30	83
				Breast density		True class		
						F	G	D
150	Whole image	70.0%	10 pos/feat	Predicted class	F	89	18	6
					G	11	53	24
					D	6	31	82
				Breast density		True class		
						F	G	D
1000	Sub-window	**77.5%**	1 pos/feat	Predicted class	F	95	14	3
					G	7	66	21
					D	4	23	87

Table 6.13: Results of using texture features extracted from the whole image or the sub-windows with no NaN.

So, we can observe that all the obtained results using texture features, correcting the problem of the NaN values are better than in the previous table and even better than using only pixel values.

6.1.3.10 Unbalanced learning sets

After analyzing the results in each one of the 10 iterations that integrate the cross-validation process we realized that in some of them the results were significantly worse than in the others. An example can be seen in the following table 6.14:

Iteration/	**1**	**2**	**3**	**4**	**5**
Recognition rate	53.125%	75.00%	75.00%	78.125%	65.625%
	6	**7**	**8**	**9**	**10**
Total: 71'56%	71.8750%	62.5%	84.375%	78.125%	71.875%

Table 6.14: Results in different iterations unbalanced sub-sets

It is easily seen that in the first, fifth and sixth iteration the results are significantly worse than in the other iterations and make it worse the final cross validation's result.

Our first thought was that in that cases, the learning sets were unbalanced, in other words they had a different number of images belonging to each class. And it was true, we checked the images in these sets and they mostly had more images of class 2 than the others and that caused the lost of recognition rate.

So, we did a test where we used 300 images of the mini-MIAS database, 100 of each density class. We also implemented a procedure to create a random vector with 10 images of each class in each one of the 10 sub-sets for the training and testing steps. Finally, we built a model and tested the trees using these subsets with equal number of images belonging of each class.

The obtained results were even worse than with the unbalanced sub-sets, they can be seen in table.

Iteration/	**1**	**2**	**3**	**4**	**5**
Recognition rate	76.66%	66.66%	56.66%	66.66%	73.33%
	6	**7**	**8**	**9**	**10**
Total: 69%	63.33%	63.33%	76.66%	70.00%	76.66%

Table 6.15: Results in different iterations balanced sub-sets

The conclusion is that the problem isn't due to the unbalanced number of images belonging of different classes, however we have detected that the problem, as we have commented, occurs when class 2 is trying to be classified. Some images belonging to this class in some cases are very similar to images belonging to class 1 and in other cases so similar to class 3. That is the reason why it costs so much to classify this kind of mammograms. The problem in the iterations where the recognition rate is significantly worse are due to there are some images of class 2 that are really difficult to classify correctly because they are very similar to images belonging to other classes.

6.1.3.11 Assess the degree of certitude for the decision tool

The final goal for the used method is classification for diagnosis, so at the end of the thesis we asked the possibility to assess the degree of certitude for the decision tool.

Due to the randomness of the algorithm, although the learn and test sets are the same in two consecutive experiments, the build trees may be different. So our question was if the final decision for an image would be different in consecutive experiments.

And the answer was yes. We have implemented an experiment to perform 20 consecutive times the same experiment with the same learning and testing sets and we obtained a variability of some images in the final decision. The experiment was done using images with the pre-processing step done, nsplit 4, abs. value and taking only 10.000 and 150 samples for learning and testing.

The results after the 20 iterations, can be shown in the table 8.1. The conclusion is that 235 images, out of 322, don't change the final result of the classification. That images are the most discriminative ones, which the final recognition rate has no doubt about the final class. There are 85 that change the classification result in some iterations, for instance, there are 17 images that change only in 1 iteration, it means that they are quite discriminative but in one iteration out of 20 (95%), due to the randomness of the algorithm, the final recognized class are different than in the other 19 iterations. Finally there are 4 images that change in 10 of the 20 iterations, meaning that there are the less discriminant images and the ones that are more similar to images belonging to a different class, and the algorithm misclassified them in many iterations.

The presence of the few images badly recognized in consecutive experiments means that some feature or test can be enhanced, added or weighted in order to make the learning phase stronger and ensures a better uniform validation. Normally the confusion is between images of class 2 and 3 that look very similar and the classifier makes a mistake in the decision. There can be also a problem of image classification by the radiologist because is a manual subjective procedure and in some cases depend on the visual perception of the human being.

6.1.3.12 Results with 2 classes

The last performed test with mini-MIAS database was re-assigning the image's classes using 2 (Dense, no-Dense) classes instead of 3, as we have seen in the work of Keir Bovis and Sameer Singh[3]. The results are computed using the original image, nsplit 4, absolute pixel and using 150 or 1000 samples.

We have tested two ways to reduce from three classes to two. The first one only reclassify all the images belonging to class 2 into images of class 1. The other way was subjectively classifying the class 2 images into class 1 or 3 depending on their opacity and their visible density. The obtained results are presented next:

<table>
<tr><th colspan="4">Class 1(Dense) = 210
Class2(No dense) = 112</th><th colspan="4">Class 1= 163
Class2 = 159</th></tr>
<tr><td colspan="4">Recognition
rate: 77.8125%</td><td colspan="4">Recognition
rate: **85%**</td></tr>
<tr><td colspan="4">Confusion matrix:</td><td colspan="4">Confusion matrix:</td></tr>
<tr><td rowspan="2" colspan="2">Breast density</td><td colspan="2">True class</td><td rowspan="2" colspan="2">Breast density</td><td colspan="2">True class</td></tr>
<tr><td>D</td><td>ND</td><td>D</td><td>ND</td></tr>
<tr><td rowspan="2">Predicted class</td><td>D</td><td>205</td><td>66</td><td rowspan="2">Predicted class</td><td>D</td><td>TP:132</td><td>FP:22</td></tr>
<tr><td>ND</td><td>5</td><td>44</td><td>ND</td><td>FN:26</td><td>TN:140</td></tr>
</table>

Table 6.16: Results using classes dense and no-dense

With the previous results is easily seen that the more balanced the sets are the better recognition rate we obtain. In the first table the proportion of images belonging to each class was very unbalanced so, after the process of reassigning classes to have a more balanced sets, we have obtained a 85% of good recognition when we deal with 2 classes dense or not dense.

We have also implemented a Matlab program that plots the Receiver operation characteristic (ROC) of the classification results for each output class. It is a graphical plot of the true positive rate or sensitivity ($TPR = \frac{TP}{TP+FN}$) vs. false positive rate ($FPR = \frac{FP}{FP+TN}$) or 1-specifity for a binary classifier varying values of the threshold t. The 45° diagonal line connecting (0,0) to (1,1) is the ROC curve corresponding to random chance. The ideal ROC curves the line connecting (0,0) to (0,1) and (0,1) to (1,1). Generally, ROC curves lie between these 2 extremes, where our curves for dense and no-dense classes are. We can see in our obtained chart that the recognition results for class dense are better than for no-dense class so the curve of the ROC is more approximated to the left up corner and further from the diagonal.

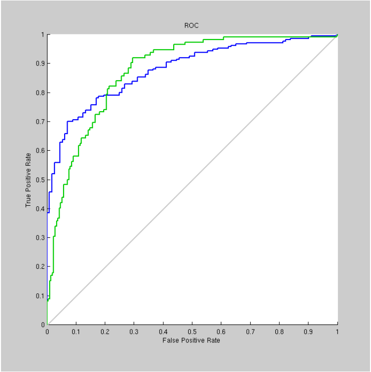


Figure 6.8: Receiver operation characteristic (ROC) for two class problem:

6.1.3.13 Results comparison with the state of the art

To conclude this chapter, we are going to compare our best obtained results with the actual state of the art procedures that deal with the problem of density classification using mini-MIAS database in the table 6.17.

<table>
<tr>
<td rowspan="2"><b>Our results</b></td>
<td><b>3 classes - 1000 subwindows</b>
Recognition rate: <b>76.25%</b>
Confusion matrix:
<table>
<tr><td colspan="2" rowspan="2">Breast density</td><td colspan="3">True class</td></tr>
<tr><td>F</td><td>G</td><td>D</td></tr>
<tr><td rowspan="3">Predicted class</td><td>F</td><td>92</td><td>15</td><td>3</td></tr>
<tr><td>G</td><td>8</td><td>60</td><td>17</td></tr>
<tr><td>D</td><td>5</td><td>28</td><td>92</td></tr>
</table>
</td>
<td><b>3 classes - 1000 subwindows - Add. feat</b>
Recognition rate: <b>77.5%</b>
Confusion matrix:
<table>
<tr><td colspan="2" rowspan="2">Breast density</td><td colspan="3">True class</td></tr>
<tr><td>F</td><td>G</td><td>D</td></tr>
<tr><td rowspan="3">Predicted class</td><td>F</td><td>95</td><td>14</td><td>3</td></tr>
<tr><td>G</td><td>7</td><td>66</td><td>21</td></tr>
<tr><td>D</td><td>4</td><td>23</td><td>87</td></tr>
</table>
</td>
</tr>
<tr>
<td><b>3 classes - 150 subwindows</b>
Recognition rate: <b>73.43%</b>
Confusion matrix:
<table>
<tr><td colspan="2" rowspan="2">Breast density</td><td colspan="3">True class</td></tr>
<tr><td>F</td><td>G</td><td>D</td></tr>
<tr><td rowspan="3">Predicted class</td><td>F</td><td>92</td><td>17</td><td>6</td></tr>
<tr><td>G</td><td>7</td><td>55</td><td>18</td></tr>
<tr><td>D</td><td>7</td><td>30</td><td>88</td></tr>
</table>
</td>
<td><b>3 classes - 150 subwindows - Add. feat</b>
Recognition rate: <b>74.68%</b>
Confusion matrix:
<table>
<tr><td colspan="2" rowspan="2">Breast density</td><td colspan="3">True class</td></tr>
<tr><td>F</td><td>G</td><td>D</td></tr>
<tr><td rowspan="3">Predicted class</td><td>F</td><td>95</td><td>19</td><td>5</td></tr>
<tr><td>G</td><td>8</td><td>60</td><td>21</td></tr>
<tr><td>D</td><td>3</td><td>25</td><td>84</td></tr>
</table>
</td>
</tr>
<tr>
<td><b>State of the art</b></td>
<td>K. Bovis et al.[3]
kNN. 71%
ANN with back prop.: 70.4%
ANN perceptron: 68%
ANN with gradient desc.: 69%
<hr>T. MacGahan et al.[19]
Image subdivision with SVM
(1x1) 63.9% (2x2) 83% (3x3) 80.3%</td>
<td>S. Tzikopoulos et al.[42]
Recognition rate: 65.52%
Confusion matrix:
<table>
<tr><td colspan="2" rowspan="2">Breast density</td><td colspan="3">True class</td></tr>
<tr><td>F</td><td>G</td><td>D</td></tr>
<tr><td rowspan="3">Predicted class</td><td>F</td><td>82</td><td>21</td><td>7</td></tr>
<tr><td>G</td><td>20</td><td>52</td><td>28</td></tr>
<tr><td>D</td><td>8</td><td>32</td><td>77</td></tr>
</table>
</td>
</tr>
</table>

Table 6.17: Results comparison classification methods using mini-MIAS database.

From the previous table, we can observe that our results are almost better that all the collected results from the state of the art using this database. The only algorithm that got better results were the work of T. MacGahan et al.[19] using sub-divisions of the original image. Our best achieved recognition rate is 73.43% using 150 subwindows and 76.25% using 1000 subwindows.

We have seen that our results are very dependent of a lot of parameters: number of splits, type of image, adding more features, type of test, etc. We have done a lot of experiments trying to find the best ones to achieve the best recognition rate. We have seen that doing some improvements in the images, same orientation or use soft voting, the recognition rate increased significantly.

6.2 Test in our database

Our database is composed by 41 images that our radiologist has given to us. There are either MLO or CC images but only the 17 MLO's have pectoral muscle to remove. Only one of them has an artifact to remove, however due to it is placed inside the breast, it was impossible to remove it with our algorithm.

We are going to present the obtained results in terms of pectoral muscle removal in the segmentation step and in terms of recognition rate and confusion matrices in the classification step.

Mention that these images would be in DICOM format, but due to some problems with the software, we are only capable of dealing with the jpeg version of them, loosing the whole range of intensity values that

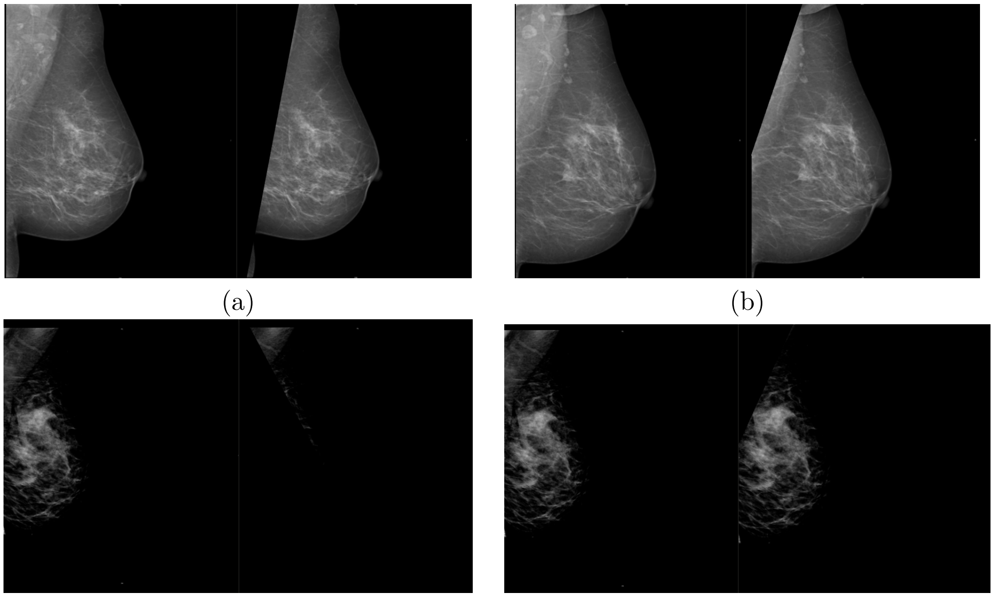
DICOM provides. This problem causes the fault of discrimination between some classes, e.g. almost all images are classified as class 4. That is the mainly reason of the bad results obtained in the classification step with these images.

6.2.1 Segmentation

With our automatic procedure of pectoral muscle removal in the mammogram images, we have obtained higher results than with the mini-MIAS database, although we also have less number of images. We have tested the results also subjectively. The examples of the commented results can be shown in the table 6.18 and in the figure 6.9:

Subjective evaluation	Result	Example
Good	64.70 (%)	Figure 6.9(a)
Acceptable	29.41 (%)	Figure 6.9(b)
Unacceptable	5.82 (%)	Figure 6.18(c)

Table 6.18: Results of the segmentation pre-processing step



(a) (b)

(c) (d)

Figure 6.9: Results in the muscle removal: (a) Good results (b) Acceptable results (c) Unacceptable results (d) Unacceptable corrected results

The only one unacceptable result obtained was corrected changing the size of the region of interest, as we did with mini-MIAS database.

6.2.2 Classification

In the case of our 41 images, they are manually classified by our radiologist into 4 different density classes according to BIRADS criterion:

- I: Entirely fatty → 8 images.
- II: Fibro-glandular tissue → 8 images.
- III: Heterogeneously-dense → 10 images.
- IV: Extremely-dense → 15 images.

The used parameters are the same that with mini-MIAS database: nsplit, #subwindows, type of image and type of performed test.

We are going to present the results using two types of images (original and without pectoral muscle), nsplit 4, absolute test and 150 subwindows in order to find the best possible results.

<table>
<tr><th>Type of image</th><th colspan="6">Original image</th><th colspan="6">Pectoral muscle removed</th></tr>
<tr><td rowspan="7">Original image</td><td colspan="6">Recognition rate: 42.5%
Confusion matrix:</td><td colspan="6">Recognition rate: 55%
Confusion matrix:</td></tr>
<tr><td colspan="2" rowspan="2">Breast dens.</td><td colspan="4">True class</td><td colspan="2" rowspan="2">Breast dens.</td><td colspan="4">True class</td></tr>
<tr><td>I</td><td>II</td><td>III</td><td>IV</td><td>I</td><td>II</td><td>III</td><td>IV</td></tr>
<tr><td rowspan="4">Predicted class</td><td>I</td><td>0</td><td>0</td><td>0</td><td>0</td><td rowspan="4">Predicted class</td><td>I</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>II</td><td>1</td><td>3</td><td>3</td><td>1</td><td>II</td><td>0</td><td>5</td><td>2</td><td>1</td></tr>
<tr><td>III</td><td>0</td><td>1</td><td>0</td><td>0</td><td>III</td><td>0</td><td>1</td><td>2</td><td>0</td></tr>
<tr><td>IV</td><td>7</td><td>4</td><td>6</td><td>14</td><td>IV</td><td>7</td><td>2</td><td>5</td><td>14</td></tr>
</table>

Table 6.19: Results our images without adding new features

We also tested this database using the additional features that we had implemented, feature scale and texture features, no finding any improvement as we can see in the following table 6.20:

<table>
<tr><th>Type of image</th><th colspan="6">1 position each feature</th><th colspan="6">10 positions each feature</th></tr>
<tr><td rowspan="7">Original image</td><td colspan="6">Recognition rate: 30%
Confusion matrix:</td><td colspan="6">Recognition rate: 35%
Confusion matrix:</td></tr>
<tr><td colspan="2" rowspan="2">Breast dens.</td><td colspan="4">True class</td><td colspan="2" rowspan="2">Breast dens.</td><td colspan="4">True class</td></tr>
<tr><td>I</td><td>II</td><td>III</td><td>IV</td><td>I</td><td>II</td><td>III</td><td>IV</td></tr>
<tr><td rowspan="4">Predicted class</td><td>I</td><td>0</td><td>0</td><td>1</td><td>0</td><td rowspan="4">Predicted class</td><td>I</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>II</td><td>1</td><td>2</td><td>2</td><td>1</td><td>II</td><td>0</td><td>2</td><td>1</td><td>1</td></tr>
<tr><td>III</td><td>1</td><td>3</td><td>2</td><td>5</td><td>III</td><td>4</td><td>3</td><td>2</td><td>4</td></tr>
<tr><td>IV</td><td>6</td><td>3</td><td>5</td><td>8</td><td>IV</td><td>3</td><td>3</td><td>6</td><td>9</td></tr>
</table>

Table 6.20: Results our images adding new features extracting 150 subwindows.

The bad obtained results are due to tree main factors:

1. We have only 41 images manually annotated by our radiologist and they aren't enough to achieve a good classification rates. We would need a richer database with more images of each class.

2. We have a dataset with more images belonging to the class 4 than the others. That factor produces an unbalanced learning set that triggers the classification of this class in too occasions.

3. The main reason is that our images are in jpeg format, instead of the original DICOM, so we lost a lot of variability between the range of intensities that this format provides. This problem may affect in the discrimination of classes and disturb greatly the classification process.

Chapter 7

Conclusions and future work

7.1 Conclusions

Computer-aided design systems as breast density classification or breast pectoral segmentation are very useful step that help radiologists in his first assessment to the mammogram images to detect and diagnose possible cancers. High dense mammograms or the presence of pectoral muscle may interfere with the correct operation of this CAD systems. The final goal of this thesis is the density classification of mammograms for final diagnosis. In this study case many implementations have been proposed through the recent years and we have adopted a procedure to obtain the best possible results following a set of requirements.

In this master thesis, we have proposed an automatic method to segment the breast from the pectoral muscle and background on the MLO views of mammograms before the classification. This method is based on the Hough transform that overcomes the initial requirement of finding the straight-line that separates the pectoral muscle from the breast using a fast procedure and without user interaction. This segmentation process also includes a pre-processing of the image to find the pectoral muscle, enhance the image or remove the labels and artifacts presents in the mammogram. This label extraction is done automatically with a region growing algorithm and using the ITK library.

The method was tested on the 322 digitized mammograms of the mini-MIAS database showing a general good behavior. The pre-processing techniques have shown almost perfect results. It achieved to remove all the labels present in the mammogram images (100%). Referring to the artifacts, we also could remove all of them except one label that has one portion placed inside the breast and the other inside the background, reaching a 99.68% removal rate. With the pectoral suppression method we have obtained less results due to the variability in intensity levels, shape and dimensions of the breast. Nevertheless these inconvenient, he have reached a 57.76% of good results and 28.88% of acceptable results. The strength of this method is his low complexity and short computational time (less than 30 minutes for the whole database).

We have also proposed an automatic classification procedure to identify the density in mammogram images. The method is based on random extraction of subwindows from the test images, learning of a model based on the ensemble of trees and testing the model with the propagation of the correspondent extracted subwindows through the built trees and averaging all the obtained probabilities of belonging to a specific class.

The method was evaluated over the mini-MIAS database and it showed promising results achieving a best recognition rate of 76.25% only using the pixel values and a 77.5% using texture features in the three class problem and a 85% in the two class problem. These results are better than the obtained rates in the literature from the actual state of the art in this particular field.

We have done some modifications to the algorithm in order to obtain the best possible results as well as adding some new blocks to implement other functionalities like cross validations or presentation of the results. The implemented improvements includes: possibility of different tests in each node of the three(absolute, difference or maximum value) or inclusion of new features in the vector(feature scale and texture features). We have done a lot of experiments with a wide range of parameters (type of used image, number of splits or type of test) to determine with which the best results are obtained.

Our method for segmenting and classifying mammogram images has also been applied to a set of 41 images previously annotated by our radiologist according to the BIRADS criterion. These images were in DICOM format but due to some problems we only had access to the jpeg version of them. The segmentation results were better than with the mini-MIAS because the images had less variability, achieving a 64.70% of good and 29.41% of acceptable results. However, thus this low variability and the low range of intensity values that provides jpeg compared with DICOM, the segmentation results were significantly worse than with the other images. The obtained rate only was 42.5% for the original images and a 55% with the images with the pectoral muscle removed.

The strength and weakness of the classification method is its randomness. It is a strength because the classification only relies in the intensity value of the pixels and don't take any other consideration of the images. It's a simple procedure to extract the features for its classification and the computational time is also not excessive. It's a weakness because, as we have demonstrated, two consecutive experiments, even with the same learning and testing set, may construct different trees and their final recognized class may be different. It happens in a few group of images and not in all the iterations. The conclusion extracted from these final results is that there are a group of images whose classification is very clear because they show a very good discrimination between other images belonging to a different class. Nevertheless, there are a few group of images, whose make the result of classification worse, that are very similar to images belonging to a different class and the algorithm misclassified them in some iterations.

7.2 Future work

In future work about the segmentation step, it would be interesting to use the found straight line that determines the pectoral edge as a initial input for an iterative procedure that refines this straight line approximation into a curve that describes more accurately the pectoral muscle as the work of Know et al.[17] or Ferrari et al.[10]. It also might help the deeply study of other techniques as the Gabor wavelets with whose good results were obtained in the work of Ferrari et al.[10].

For further analysis and studies in the classification step, it would be interesting as well the deeply study of new texture features that may help to distinguish between the more similar mammograms. It is also necessary to find a way to decrease the randomness in the final results for their final goal, classification for diagnosis. Finally, further efforts has to be put in order to increase the results with our database, dealing with DICOM images and a greater range of intensity values, rely the experiments with a more number of mammograms and try not to have an unbalanced set of mammograms.

Another future work may be the study of the extraction texture features from the co-occurrence matrices (SGLD) [13] as in the work of Mustra et al.[28], Bovis and Singh [3] and Oliver et. al. [30].

For a future research and after adapting the algorithm to deal with DICOM images, the classification algorithm can be applied also to another well known database of mammograms, DDSM, which because the lack of time, it has not been possible.

Bibliography

[1] M. Abdel-Mottaleb and C. Carman. Locating the boundary between the breast skin edge and the background in digitized mammograms. In *Digital Mammography*, pages 467–470, 1996.

[2] U. Bick and M Giger. Automated segmentation of digitized mammograms. In *Academic Radiology*, volume 2, pages 1–9, 1995.

[3] K. Bovis and S. Singh. Classification of mammographic breast density using a combined classifier paradigm.

[4] J. Byng and N. Boyd. Automated analysis of mammogram densities. In *Medical Physics*, volume 41, pages 909–923, 1998.

[5] A. C. Chaabani, A. Boulelben, A. Mahfoudhi, and M. Abid. An automatic-pre-processing method for mammographic images. In *International Journal of Digital Content Technology and its Applications*, volume 4, 2010.

[6] R. Chandrasekhar and Y. Attikiouzel. Gross segmentation of mammograms using a polynomial model. In *International Conference of the IEEE in Medicine and Biology Society*, volume 3, pages 1056–1058, 1996.

[7] Z Chen and R. Zwiggelaar. Segmentation of the breast region with pectoral muscle removal in mammograms. 2010.

[8] I.M. De Carvalho, L.M.S. Luz, A.V. Alvarenga, A.F.C. Infantosi, and C.M. Pereira, W.C.A.and Azevedo. An automatic method for delineating the pectoral muscle in mammograms. In *CLAIB 2007, IFMBE Proceedings*, pages 271–275, 2007.

[9] R. O. Duda and P. E. Hart. *Pattem Classification and Scene Analysis.* New York Wiley, 1973.

[10] R. Ferrari and R. Rangayyan. Segmentation of mammograms: Identification of the skin boundary and the pectoral muscle. In *IWDM*, volume 23, 2000.

[11] M. Gemo, A. Gouze, B. Debande, A. Grivegnee, G. Mazyb, and B. Macq. A versatile knowledge-based clinical imaging annotation system for breast cancer screening. volume 6514, 2007.

[12] M. Goodsitt and H. Chan. Classification of compressed breast shapes for the design of equalisation filters in x-ray mammography. In *Medical Physics*, volume 25, pages 937–947, 1998.

[13] R.M. Haralick, K.S. Shanmugan, and I. Dunstein. Textural features for image classification. In *IEEE Trans. Syst., Man, Cybern*, volume 3, pages 610–621, 1973.

[14] J. Hein and M. Kallargi. Multiresolution wavelet approach for separating the breast region from the background in high resolution digital mammography. In *Digital Mammography, Nijmegen, Kluwer Academic Publishers*, pages 295–298, 1998.

[15] A. Hoyer and W Spiesberg. Computerized mammogram processing. In *Phillips Technical Review*, volume 38, pages 347–355, 1979.

[16] N. Karssemeijer and G. te Brake. Combining single view features and asymmetry for detection of mass lesions. In *IWDM*, pages 95–102, 1998.

[17] S. Kwok, R. Chandrasekhar, and Y. Attikiouzel. Automatic pectoral muscle segmentation on mammograms by straight line estimation and cliff detection. In *IIS Conference*, pages 67–72, 2001.

[18] T. Lau and W Bischoff. Automated detection of breast tumors using the asymmetry approach. In *Computers and Biomedical Research*, volume 24, pages 273–295, 1991.

[19] T. MacGahan, M. R. Pacheco, and A. Wong. A hybrid approach to automatic mammography classification. 2007.

[20] R. Maree, J. Geurts, and L. Wehenkel. Content-based image retrieval by indexing random subwindows with randomized trees. volume 1, pages 46–57, 2009.

[21] R. Maree, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR 2005*, volume 1, pages 34–40, 2005.

[22] M. Masek. *Hierarchical segmentation of mammograms based on pixel intensity.* PhD thesis, University of Western Australia, 2004.

[23] M. Masek and Y. Attikiouzel. Skin-air interface extraction from mammograms using an automatic local thresholding algorithm. In *ICB, Brno, CR*, pages 204–206, 2000.

[24] K. McLoughlin and P. Bones. Locating the breast-air boundary for a digital mammogram image. In *Image and Vision Computing*, 2000.

[25] A. Mendez and P. Tahoces. Automatic detection of breast border and nipple in digital e mammograms. In *Computer Methods and Programs in Biomedicine*, volume 49, pages 253–262, 1996.

[26] P. Miller and S. Astley. Classification of breast tissue by texture analysis. pages 277–282, 1992.

[27] A. Morton, H. Chan, and M. Goodsitt. Automated model-guided breast segmentation algorithm. In *Med Phys*, pages 1107–1108, 1996.

[28] M. Mustra, M. Grgic, and K. Delac. Feature selection for automatic breast density classification. In *Proc. PROCEEDINGS ELMAR*, pages 9–16, 2010.

[29] T. Ojala and J. Liang. Interactive segmentation of the breast region from digitized mammograms with united snakes. In *Technical Report 315, Turku Centre for Computer Science*, 1999.

[30] A. Oliver, J. Freixenet, and R. Zwiggelaar. Automatic classification of breast density. In *Proc. IEEE Int. Conf. Image Processing ICIP 2005*, volume 2, 2005.

[31] M-H Dilhuydy P. Taylor, S. Hajnal and B. Barreau. Measuring image texture to separate difficult from easy mammograms. pages 456–463, 1994.

[32] S. Petroudi and M. Brady. Segmentation using texture. In *Digital Mammography / IWDM'06*, pages 609–615, 2006.

[33] S. Petroudi, T. Kadir, and M. Brady. Automatic classification of mammographic parenchymal patterns: a statistical approach. In *Proc. 25th Annual Int Engineering in Medicine and Biology Society Conf. of the IEEE*, volume 1, pages 798–801, 2003.

[34] D. Raba, A. Oliver, J. Marti, M Peracaula, and J. Espunya. Breast segmentation with pectoral muscle suppression on digital mammograms. In *IbPRIA 2005, LNCS 3523*, pages 471–478, 2005.

[35] H. Rickard, G. Tourassi, and A. Elmaghraby. Self-organizing maps for masking mammography images. In *IEEE EMBS*, pages 302–305, 2003.

[36] P. Saha and J. Udupa. Breast tissue density quantification via digitized mammograms. In *IEEE Transactions on Medical Imaging*, volume 20, pages 792–803, 2001.

[37] J. Semmlow and A. Shadagopappan. A fully automated system for screening xero-mammograms. In *Computers and Biomedical Reseach*, volume 13, pages 350–362, 1980.

[38] H. S. Sheshadri and Kandaswamy A. Breast tissue classification using statistical feature extraction of mammograms. 2006.

[39] E. Stomatakis and A. Cairns. A novel approach to aligning mammograms. in: Digital mammography. In *Digital Mammography*, pages 255–364, 1994.

[40] C. Tromans, J. Brady, and R. Warren. A high accuracy technique for breast air boundary segmentation and the resulting improvement from its use in breast density estimation. In *IWDM*, pages 17–18, 2004.

[41] S. Tzikopoulos, H. Georgiou, M. Mavroforakis, N. Dimitropoulos, and S. Theodoridis. A fully automated complete segmentation scheme for mammograms. In *Proc. 16th Int Digital Signal Processing Conf*, pages 1–6, 2009.

[42] S. Tzikopoulos, H. Georgiou, M. Mavroforakis, N., and S. Theodoridis. A fully automated scheme for breast density estimation and asymmetry detection of mammograms. In *17th European Signal Processing Conference (EUSIPCO 2009)*, 2009.

[43] M. Wirth and A. Stapinski. Segmentation of the breast region in mammograms using snakes. In *Canadian Conference on Computer and Robot Vision*, pages 385–392, 2004.

[44] F. Yin and M Giger. Computerized detection of masses in digital mammogram: analysis of bilateral subtraction images. In *Medical Physics*, volume 28, pages 955–963, 1991.

[45] C. Zhou and H. Chan. Computerized image analysis: Estimation of breast density on mammograms. In *Med. Phys.*, volume 28, pages 1056–1069, 2001.

[46] E. Ziv, J. Shepherd, R. Smith-Bindman, and K. Kerlikowske. Mammographic breast density and family history of breast cancer. *Journal of the National Cancer Institute*, Vol. 95:7, 2003.

Chapter 8

Annexes

8.1 Description files

8.1.1 Segmentation files

```
/*
 * SegmentationFilesDescription.cpp
 * This class includes all the segmentation files and its description
 *  Created on: 07/06/2011
 *      Author: Jaume Sastre
 */

/*
 * Main.cpp
 *  Main program used to launch other binary files.
 *   Instanciate the others C++ object in charge of doing segmentation procedures.
 *   As an input has:
 *    inputFileName: name of the file where the image is loaded
 *    outputFileName: name of the file where the image will be saved
 *
 * Main.h
 *  Header file of Main.cpp
 */

/*
 * BreastLabelsAndArtifactsRemoval.cpp
 *  Principal segmentation function that has four main methods:
 *   ExtractLabelsAndArtifacts: method that removes all labels and artifacts using the
     region growing algorithm.
 *   ExtractMuscleHough: method that extracts the pectoral muscle using the implementation
     of the Hough transform.
 *   ExtractBlackBorder: method that extracts the black borders present in mini-MIAS images.
 *   SubsamplingImage: method that performs subsampling in our images using a linear
     interpolator and a identity    transform.
 *
 *   Input:
 *    - inputFileName
 *    - outputFileName
 *   Output:
 *    - save the output image in the correspondent outputFileName
 *
 * BreastLabelsAndArtifactsRemoval.h
 *   Header file.
 */

/*
 *   BreastDICOMSegmentation.cpp:
 *    Program that loads a DICOM image and tries to remove labels and muscle either using
      region growing of Hough    transform.
 *    Includes other methods to extract the histogram, extract borders or compute pixel
      values.
 *
 *   BreastDICOMSegmentation.h: header file
 *
 */

/*
 * ConvertRawToDicom.cpp
 *  Program that iterates over the images in a DDSM case and converts the images from raw to
       DICOM format
 *   This program also takes the width and high of the images from a file inside the folders
 *
 *ConvertRawToDicom.h
 * Header file.
 */

/*
 * CMakeLists.txt
 * CMake file that includes all the necessary commands to compile the files using CMake.
 */
```

8.1.2 Classification files

```
%------------------------------
%MATLAB DESCRIPTION FILES
%------------------------------
%   ORIGINAL FILES:
%
%       - GEURTS_scrip.m: script that reproduces all the classfication steps
%       in order to reproduce the whole prodecure.
%       - GEURTS_proces_db.m: processes images in a given directory, and
%       stores extracted samples in a given file.
%       - GEURTS_generate_smp.m: generates the specified number of samples
%       from input image
%       - GEURTS_divide_set.m: divide the set into learning and testing set
%       - GEURTS_select_samples.m: selects a specified number of samples from
%       the sets.
%       - GEURTS_build_trees.m: build an ensemble of trees
%       - GEURTS_use_trees.m: classify the images in the test set
%       - use_tree.m: classify a sample using the tree T
%       - treecst.m: data struct with all the fields in the tree
%
%   MODIFIED OR NEW FILES:
%
%       - SCRIP_mias_cross_validation.m: main script that performs the
%       whole classification procedure to the images in a database, where
%       its location is indicated by the variable path. Their main steps
%       are:
%           + Sampling the images and creating the dataset: extract the
%           subwindows of all the images and extract the features vectors
%           with the pixel values and additional features (texture, scale)
%           + Implementation of the cross validation method and saving the
%           partitions to the disk.
%           + In each iteration(10-fold), select the learning and testing
%           set, build the ensemble of trees with the learning set and
%           test them using the test set.
%           + Compute the obtained global results in all the iterations and
%           present the results in form of probabilities and confusion
%           matrices.
%
%       - TestExperiments.m: program that performs 96 experiments changing
%       the input parameters and launching the previous program: path, nsplit,
%       operation mode and used number of samples in the learning and
%       testing sets.
%
%       - SCRIPT_mias_database.m: program that tests the whole procedure to
%       the images in mini-MIAS database without using cross validation.
%       - SCRIPT_mias_database_cross_validation20Exp.m: program that
%       performs 20 consecutive experiments to determine the degree of
%       assessment of out classifier.
%
%       - GEURTS_process_db_mod1.m: process the images in the given
%       directory including the additional features.
%       - GEURTS_generate_smp_mod1.m: computes the samples for a given
%       image including the feature_scale and texture features and includes
%       them in the feature vector.
%       - GEURTS_build_trees_mod.m: build the threes using the different
%       implemented modes: absolute pixel, pixel difference or maximum
%       pixel according to the variable diff. Save the new fields in the
%       treecst.m file.
%
%       - GEURTS_use_trees_validation1.m: classify the images in the test
%       set in each of the cross validation iterations.
%
%       - use_tree_mod.m: test each test sample using the built trees and
%       with the correspondent test used in the building in each node.
%
%       - CalculateFeatures.m: program that compute the texture features
%       either from the subwindows or the whole image. It computes: mean,
%       variance, standard deviation, smoothness, skewness, kurtosis and
%       uniformity.
%
%       - Chartx.m: program that plots the charts for for different
%       experiments. As an input needs the vector of: probabilities,
%       variances, number of nodes and computational time.
%
```

8.1.3 Scripts

```
# SCRIPT DESCRIPTION FILES #
#
# - converter2pgm_script.sh: script that converts all the images in a directory
# to the pgm format for the classification step.
#
# - converter2png_script.sh: script that converts all the images in a directory
# to the png format for the segmentation step.
#
# - processer_muscleRemoval_script.sh: script that launch the Main.cpp binary
# process for all the segmentation steps.
#
# - scrip_converter1.sh: script that processes all the images in a DDSM case.
# The performed steps are:
#  + Uncompress the LJPEG file using the jpeg program. The output is a raw image.
#  + Launch the C++ binary file that converts raw to DICOM.
#  + Converts from DICOM to pgm to use the classification method.
#  + Erase all the temporal created files.
#
```

8.2 Charts of the classification algorithm performance

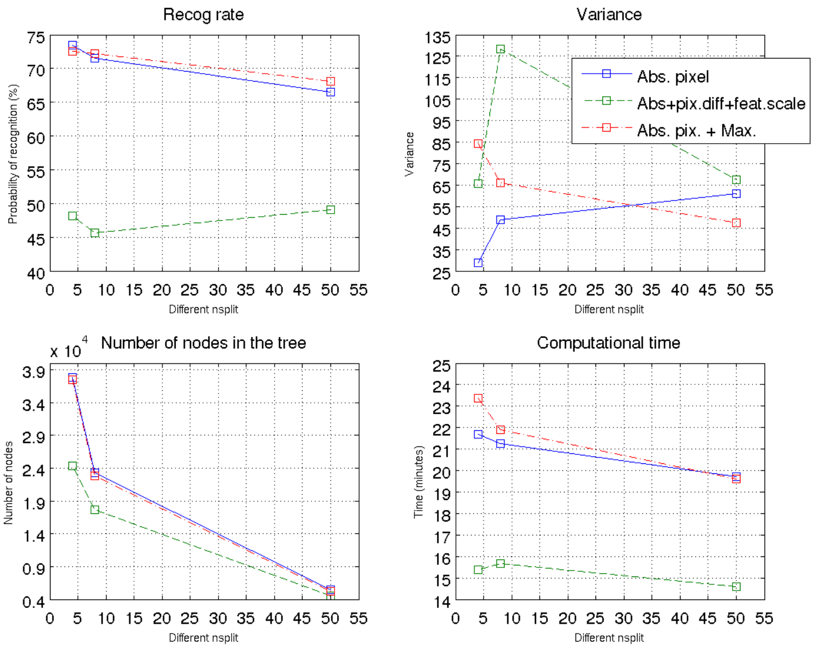


Figure 8.1: Experiments with original images and taking all the samples for learning and testing.

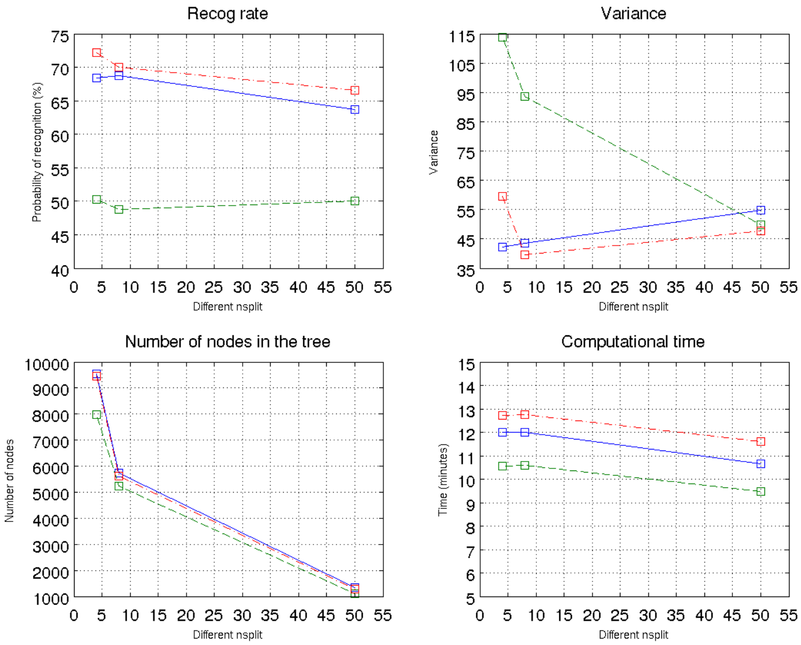


Figure 8.2: Experiments with original images and taking 10.000 samples for learning and 150 for testing.

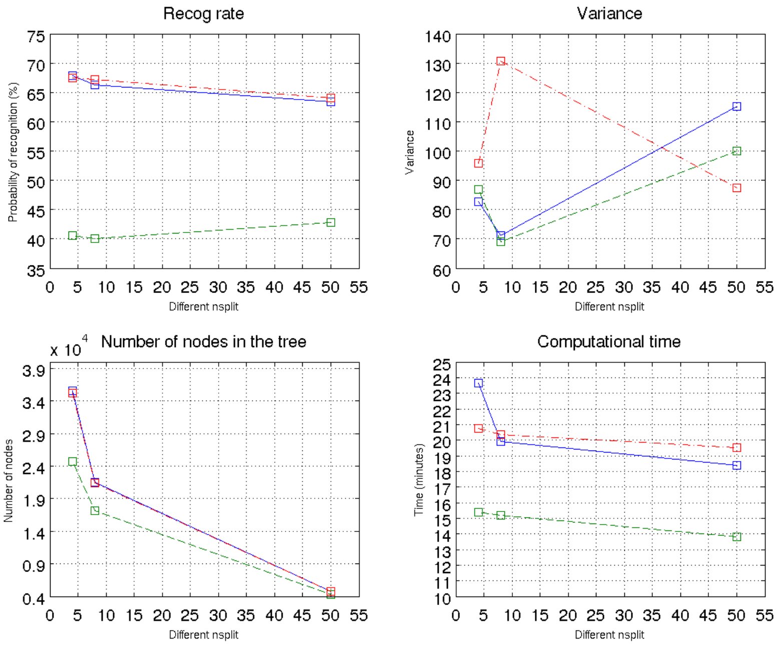


Figure 8.3: Experiments using images without labels or artifacts and taking all the samples for learning and testing.

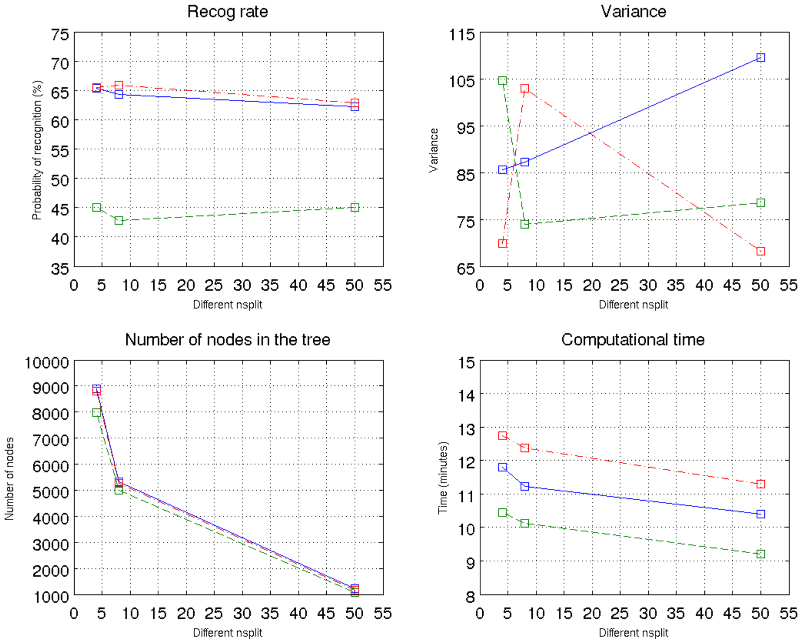


Figure 8.4: Experiments using images without labels or artifacts and taking 10.000 samples for learning and 150 for testing.

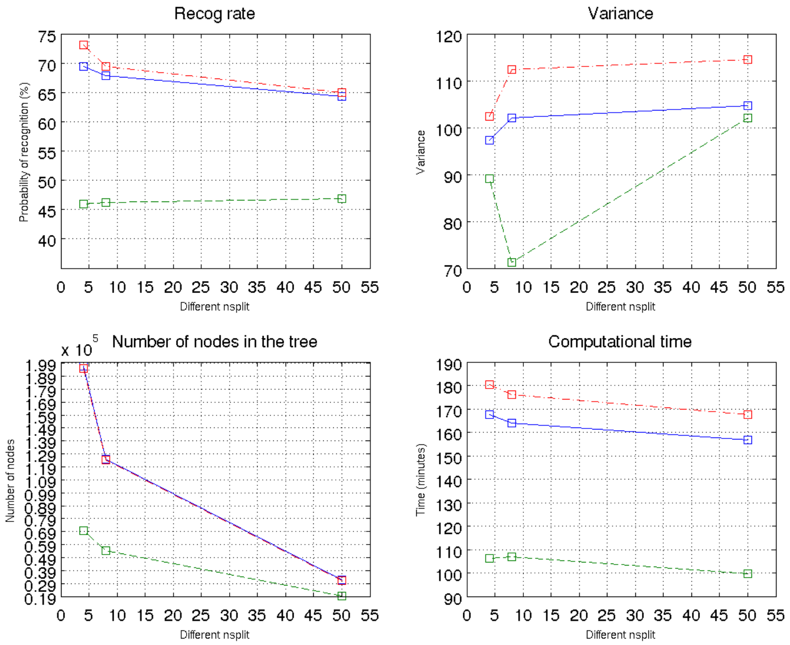


Figure 8.5: Experiments with ROI and taking all the samples for learning and testing.

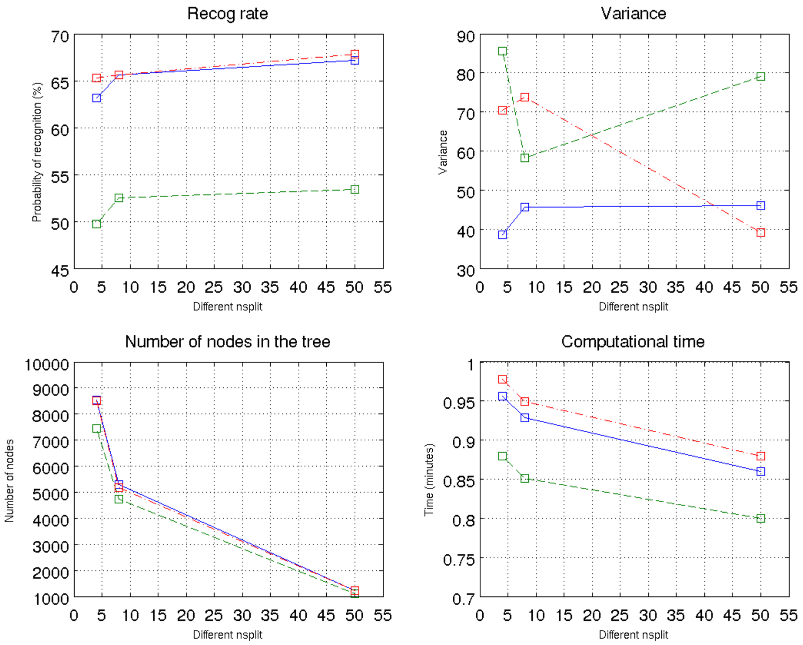


Figure 8.6: Experiments with ROI images and taking 10.000 samples for learning and 150 for testing.

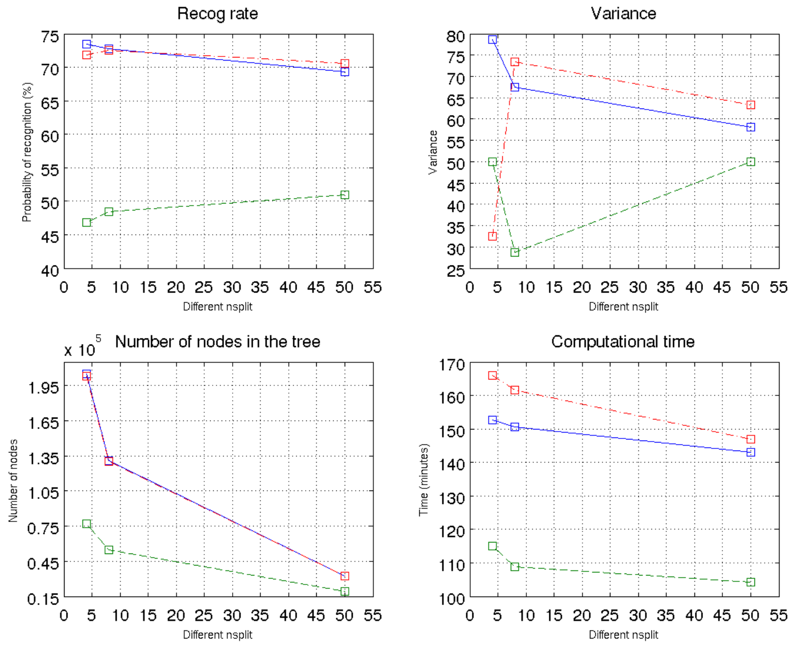


Figure 8.7: Experiments using ROI1 and taking all the samples for learning and testing.

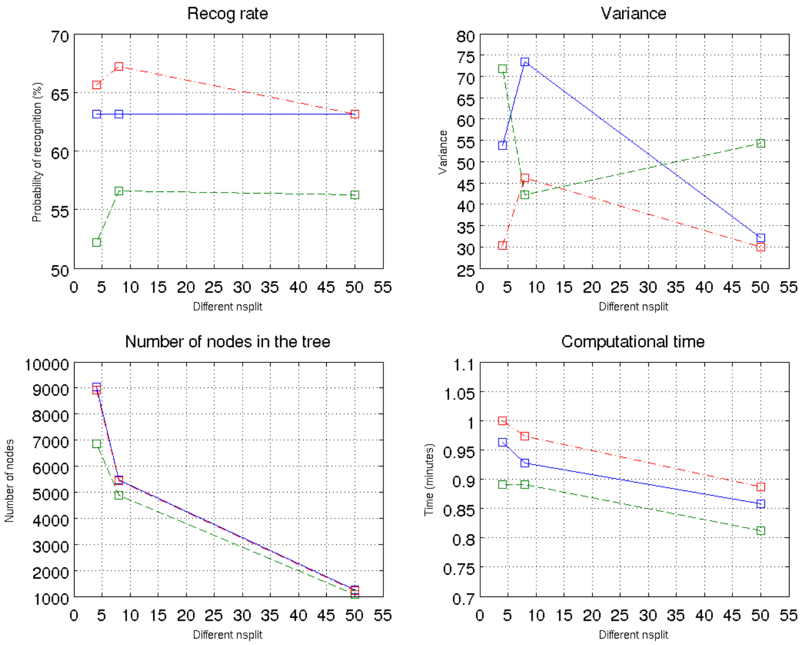


Figure 8.8: Experiments using ROI1 and taking 10.000 samples for learning and 150 for testing.

8.3 Degree of assessment with 20 experiments

Number of images same classification after 20 exp	**Probability same class**
235	100%
17	95%
17	90%
13	85%
10	80%
3	75%
6	70%
3	65%
6	60%
6	55%
4	50%

Table 8.1: Results in different iterations balanced sub-sets