



**Jaypee University of Information Technology**

**Solan (H.P.)**

**LEARNING RESOURCE CENTER**

Acc. Num *SP03037* Call Num:

**General Guidelines:**

- ◆ Library books should be used with great care.
- ◆ Tearing, folding, cutting of library books or making any marks on them is not permitted and shall lead to disciplinary action.
- ◆ Any defect noticed at the time of borrowing books must be brought to the library staff immediately. Otherwise the borrower may be required to replace the book by a new copy.
- ◆ The loss of LRC book(s) must be immediately brought to the notice of the Librarian in writing.

**Learning Resource Centre-JUIT**



**SP03037**



# **“B-MIPT”: A TOOL FOR BIOMEDICAL IMAGE PROCESSING AND THEIR CLASSIFICATION USING NEAREST NEIGHBOUR & GENETIC ALGORITHMS**

By

**HARSH PAREEK - 031532**

**ABHILSHIT SONI - 031554**



**MAY-2007**

**Submitted in partial fulfillment of the Degree of Bachelor of  
Technology**

(Project Supervision)

Senior Lecturer

Department of Biotechnology & Bioinformatics

Jaypee University of Information Technology

Wahnaghat

**DEPARTMENT OF BIOTECHNOLOGY &  
BIOINFORMATICS  
JAYPEE UNIVERSITY OF INFORMATION  
TECHNOLOGY-WAKNAGHAT, SOLAN, H.P, INDIA**

## CERTIFICATE

**This is to certify that the work entitled, ““B-MIPT ” : A TOOL FOR BIOMEDICAL IMAGE PROCESSING AND THEIR CLASSIFICATION USING NEAREST NEIGHBOUR & GENETIC ALGORITHMS ” submitted by Harsh Pareek (031532) Abhilshit Soni (031554 ) in partial fulfillment for the award of degree of Bachelor of Technology in Bioinformatics of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.**



**Dr. Pradeep Kumar Naik**

**(Project Supervisor)**

**Senior Lecturer**

**Department of Biotechnology & Bioinformatics**

**Jaypee University of Information Technology**

**Waknaghat**

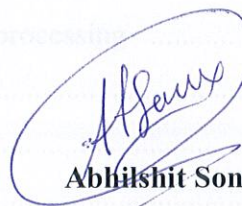


## ACKNOWLEDGMENT

I take this opportunity to extend my gratitude towards our project coordinator Dr Pradeep K Naik who has stood by us during the entire tenure of our project work. He has been like a father figure to us and we appreciate his efforts and his precious time invested in this work. This work has been one of the most important projects in our academic career by far and a lot of exertion has been put into this work. A lot many people have contributed significantly in this project and this space is too little to compose their names, but I would like to thank each one of them for their worthy contribution. We would also like to thank Prof. A. K. Jain, IOP , ( ICMR , Safdarjung campus) and Mr. Banjit Bastia for providing us their valuable support, resources and their valuable suggestions which facilitated us to complete our project.



Harsh Pareek



Abhilshit Soni



## CONTENTS

Chapter No.	Title	Page No.
I	ACKNOWLEDGEMENT .....	3
III	LIST OF FIGURES .....	6
IV	LIST OF TABLES .....	
V	LIST OF ABBREVIATIONS .....	7
VI	ABSTRACT .....	8
1	<b>Introduction</b> .....	
1.1	Computer vision .....	9
1.2	Image processing .....	10
1.3	Genetic Algorithms in image processing .....	11
1.3.1	Image restoration .....	13
1.3.2	Image restoration .....	13
1.3.3	Image enhancement .....	13
1.3.4	Applicability of Genetic Algorithms in preprocessing .....	14
1.3.5	Data reduction and feature extraction .....	15
1.3.6	Feature extraction applications .....	15
1.3.7	Image segmentation .....	15
1.3.8	Object recognition .....	16
1.3.9	What makes a GA special? .....	18
1.4	OBJECTIVE .....	21
2	<b>METHODOLOGY</b> .....	24
2.1	Implementation of Image Processing Techniques .....	25
2.2	Image Filters and Predefined Operations in Java 2D .....	26
2.3	GUI Components .....	33
2.3.1	Events handling Components in GUI .....	34
2.3.2	ColorModels and RGB Values for Image analysis .....	34

2.3.3	RGB ColorModel .....	35
2.4	Algorithms used for prediction and classification of images .....	35
2.4.1	Genetic Algorithm .....	36
2.4.2	Basic operators: mutation .....	45
2.4.3	The Basic Genetic Algorithm .....	45
2.4.4	NEAREST NEIGHBOR (NN) .....	48
2.4.5	Image Region Analysis (IRA) .....	48
3	Training, Testing and Validation of the Tools .....	50
3.1	Training Set Generation .....	53
3.2	Image Classification .....	53
3.3	Main Features .....	55
3.3.1	GUI Features .....	56
3.4	Inbuilt Libraries .....	58
3.4.1	JGAP (Java Genetic Algorithms Package) .....	59
3.4.2	3.4.2 NanoXML (the smallest XML parser) .....	59
3.5	Performance Measure .....	61
3.6	RESULTS AND DISCUSSION .....	61
3.7	CONCLUSION .....	65
	BIBLIOGRAPHY .....	66
	APPENDIX .....	67



## LIST OF FIGURES

Figure No.	Title	Page No.
1	Block Diagram of decision-theoretical pattern recognition system .....	9
1.1	Placental Image of CD31 Active Smoker .....	21
1.2	Placental Image of CD31 Passive Smoker. ....	21
1.3	Placental Image of Non-Smoker .....	22
1.4	Placental Image of a ET Active smoker .....	22
1.5	Placental Image of a ET Passive smoker .....	23
2	Schematic Representation of BufferedImage Model for Image Processing	25
2.1	RGB Color Space .....	34
2.2	Flow Chart to Depict the flow of control in GA .....	36
2.3	Binary crossover .....	43
2.4	Showing ET_active .xml contents .....	48
2.5	Snap shot showing Image Region Analysis .....	49
3	Overall flow control of B-MIPT .....	50
3.1	IRA Snapshot .....	52
3.2	Flow Diagram showing Test image classification .....	55
3.2	GUI Main Frame .....	56
3.3	Multiple Image Loading .....	56
3.4	Image Filters .....	57
3.5	Specify Your Own Training Set .....	57

## LIST OF TABLES

Table No.	Title	Page No.
3.1	Performance measure of the tool developed for prediction of rate of expression of proteins and their classification from the placental images. ....	60
3.2	RGB ranging among images of ET .....	60
3.3	RGB ranging among images of CD31 .....	60
3.4	Table 3.4 Result of classification of various test images .....	61

## LIST OF ABBREVIATIONS

GA	Genetic Algorithm
IRA	Image Region Analysis
NN	Nearest neighbour algorithm
API	Application Programming Interface
JGAP	Java GA package
ET	Endothelin
AWT	Abstract Window Toolkit
XML	Extensive Markup Language



## ABSTRACT

The rate of expression of proteins like endothelin and CD31 in the placental cell depends on the rate of inhalation of tobacco smoke. It has been seen that in case of the women who smokes tobacco very frequently (active smoker ) the rate of expression of these two proteins is more, whereas in the second category of women who does not smoke tobacco but they inhale the smoke coming from other sources (passive smoker), the rate of expression is low. The higher expression of these two proteins leads to the placental abnormalities subjected to birth failure. Hence pathologically the study of these two proteins and their expression is very important for proper medication. Traditionally the study can be done with the immunohistochemistry technique. The inference is generally derived based on the visualization of the image by an experienced pathologist. Since this is a manual method sometimes the diagnosis is erroneous and wrongly classifies the images. In this regard it is essential to make an automatic tool using reproducible algorithm for prediction, classification and proper diagnosis. Hence in this study, an attempt has been made to develop a tool using Image Processing and Genetic Algorithm to study the rate of expression of both the proteins in the placenta.

Using three derived features alone from the image we have been able to achieve 66 % correct detection of nucleus, cytoplasm and background of an image using first layer neural network (in the set of 88 images). The tool is also able to classify the images into cancerous/non-cancerous using three parameters (N, C and N/C) with an accuracy of 69% (in the set of 88 images). For the designed GA model the following performance measure (accuracy = 91 %,  $Q_{pred} = 91.0990$  %, sensitivity = 0.9106, specificity = 0.8216).

## CHAPTER 1

### Introduction

Image processing is a field distinct from the rest of computer graphics. It describes how digital images can be manipulated mathematically. Image processing is also called **image filtering**, a term that originated with the use of filters in photography. In computer terms it can be define as **“Image processing is any form of information processing for which the input is an image, such as photographs or frames of video; the output is not necessarily an image, but can be for instance a set of features of the image”**.

Computer imaging separated into two fields: Computer vision and Image processing.

#### 1.1 Computer vision

Computer vision is computer imaging where the application does not involve a human being in the visual loop. Image analysis involves the examination of the image data to facilitate solving a vision problem. The image analysis process involves two other topics: feature extraction is the process of acquiring higher-level image information, such as shape or color information, and pattern classification is the act of taking this higher-level information identifying objects within image.

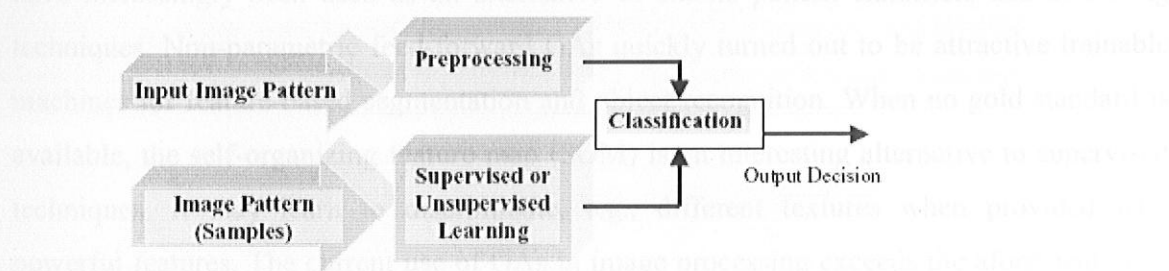


Fig. 1. Block diagram of decision-theoretical pattern recognition system.

The block diagram of decision-theoretical pattern recognition shown in Fig. 1 represented by three primary stages:



- Preprocessing
- Learning
- Classification

Preprocessing is used to remove noise and eliminate irrelevant, visually unnecessary information. Learning stage involves either supervised or unsupervised learning.

Classification stage classified the pattern, which is preprocessed in the first stage, by computing the output and considering the maximum output is the more similar image in database library to the unknown input pattern image.

## 1.2 Image processing

Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it. Techniques include convolution edge detection, mathematics, filters, trend removal, and image analysis. Traditional techniques from statistical pattern recognition like the Bayesian discriminant and the Parzen windows were popular until the beginning of the 1990s. Since then, genetic algorithm(GAs) have increasingly been used as an alternative to classic pattern classifiers and clustering techniques. Non-parametric feed-forward GAs quickly turned out to be attractive trainable machines for feature-based segmentation and object recognition. When no gold standard is available, the self-organizing feature map (SOM) is an interesting alternative to supervised techniques. It may learn to discriminate, e.g., different textures when provided with powerful features. The current use of GAs in image processing exceeds the aforementioned traditional applications. The role of feed-forward GAs and SOMs has been extended to encompass also low-level image processing tasks such as noise suppression and image enhancement. Hopfield GAs were introduced as a tool for finding satisfactory solutions to complex (NP-complete) optimization problems. This makes them an interesting alternative to traditional optimisation algorithms for image processing tasks that can be formulated as

optimisation problems. The different problems addressed in the field of digital image processing can be organized into what we have chosen to call the image processing chain. The various steps involved in the image processing chain are as follows:

1. **Preprocessing/filtering.** Operations that give as a result a modified image with the same dimensions as the original image (e.g., contrast enhancement and noise reduction).
2. **Data reduction/feature extraction.** Any operation that extracts significant components from an image (window) . The number of extracted features is generally smaller than the number of pixels in the input window.
3. **Segmentation.** Any operation that partitions an image into regions that are coherent with respect to some criterion. One example is the segregation of different textures.
4. **Object detection and recognition.** Determining the position and, possibly, also the orientation and scale of specific objects in an image, and classifying these objects.
5. **Image understanding.** Obtaining high level (semantic) knowledge of what an image shows.
6. **Optimization.** Minimization of a criterion function which may be used for, e.g., graph matching or object delineation. Optimization techniques are not seen as a separate step in the image processing chain but as a set of auxiliary techniques, which support the other steps. Besides the actual task performed by an algorithm, its processing capabilities are partly determined by the abstraction level of the input data. The following abstraction levels are included in image processing:

**A. Pixel level.** The intensities of individual pixels are provided as input to the algorithm.

**B. Local feature level.** A set of derived, pixel-based features constitutes the input.

**C. Structure (edge) level.** The relative location of one or more perceptual features (e.g., edges, corners, junctions, surfaces, etc.).

**D. Object level.** Properties of individual objects.



**E. Object set level.** The mutual order and relative location of detected objects.

**F. Scene characterization.** A complete description of the scene possibly including lighting conditions, context, etc.

### 1.3 Genetic Algorithms in image processing

In this section, we will review the application of Genetic Algorithms to perform various tasks as mentioned above in the image processing chain (3.1–3.6).

#### 1.3.1 Preprocessing:

The first step in the image processing chain consists of preprocessing. Loosely defined, by preprocessing we mean any operation of which the input consists of sensor data, and of which the output is a full image. Preprocessing operations generally fall into one of three categories: image reconstruction (to reconstruct an image from a number of sensor measurements), image restoration (to remove any aberrations introduced by the sensor, including noise) and image enhancement (accentuation of certain desired features, which may facilitate later processing steps such as segmentation or object recognition). Applications of GAs in these three preprocessing categories will be discussed separately below. The majority of the GAs was applied directly to pixel data (level A); only four networks were applied to more high-level data (features, level B).

##### 1.3.1.1 Image reconstruction

Image reconstruction problems often require quite complex computations and a unique approach is needed for each application.. An ADALINE network is trained to perform an electrical impedance tomography (EIT) reconstruction, i.e., a reconstruction of a 2D image based on 1D measurements on the circumference of the image. The Hopfield network contained “summation” layers to avoid having to interconnect all units. Regression feed-forward networks (that learn the mapping  $E(y|x)$ , with  $x$  the vector of input variables and  $y$  the desired output vector) to reconstruct images from electron holograms.

### 1.3.2 Image restoration

The majority of applications of GAs in preprocessing can be found in image restoration. In general, one wants to restore an image that is distorted by the (physical) measurement system. The system might introduce noise, motion blur, out-of-focus blur, distortion caused by low resolution, etc. Restoration can employ all information about the nature of the distortions introduced by the system, e.g., the point spread function. The restoration problem is ill-posed because conflicting criteria need to be fulfilled: resolution versus smoothness. In the most basic image restoration approach, noise is removed from an image by simple filtering. For noise suppression, the templates implement an averaging function; for edge detection, a Laplacian operator. The system operates locally, but multiple iterations allow it to distribute global information throughout the nodes. Although quite fast in application, a disadvantage is that the parameters in governing the network behaviour (the feedback and control templates) have to be set by hand.

Traditional methods for more complex restoration problems such as deblurring and diminishing out-of-focus defects, are maximum a posteriori estimation (MAP) and regularisation. Applying these techniques entails solving high-dimensional convex optimisation tasks. Often, mapping the problem turned out to be difficult, so in some cases the GA architecture had to be modified as well. Other types of networks have also been applied to image restoration developed a hybrid system consisting of order statistic filters for noise removal and for deblurring (by optimising a criterion function). The modulation transfer function had to be measured in advance.

### 1.3.3 Image enhancement

The goal of image enhancement is to amplify specific (perceptual) features. Among the applications where GAs have been developed for image enhancement, However, several enhancement approaches rely on a classifier, typically resulting in a binary output image. The most well-known enhancement problem is edge detection. A number of more complex, modular systems were also proposed. Formulating edge detection as an optimisation problem made it possible for enhancement of endocardiac borders. Some enhancement approaches utilize other types of GAs.

### ***1.3.4 Applicability of Genetic Algorithms in preprocessing***

There seem to be three types of problems in preprocessing (unrelated to the three possible operation types) to which GAs can be applied:

- optimization of an objective function defined by a traditional preprocessing problem;
- Approximation of a mathematical transformation used for image reconstruction,
- Mapping by an GA trained to perform a certain task, usually based directly on pixel data (neighbourhood input, pixel output).

To solve the first type of problems, traditional methods for optimization of some objective function may be replaced by a GAs. For the approximation task, regression (feed-forward) GAs could be applied. Although some applications such GAs were indeed successful, it would seem that these applications call for more traditional mathematical techniques, because a guaranteed (worst-case) performance is crucial in preprocessing. Secondly, when networks were adaptive, their architectures usually differed much from those of the standard GAs: prior knowledge about the problem was used to design the networks that were applied for image restoration or enhancement. The interest in non-adaptive GAs indicates that the fast, parallel operation and the ease with which GAs can be embedded in hardware may be important criteria when choosing for implementation of a specific preprocessing operation. However, the ability to learn from data is apparently of less importance in preprocessing. While it is relatively easy to construct a linear filter with a certain, desired behaviour, e.g., by specifying its frequency profile, it is much harder to obtain a large enough data set to learn the optimal function as a high-dimensional regression problem. This holds especially when the desired network behaviour is only critical for a small subset of all possible input patterns (e.g., in edge detection).

### ***1.3.5 Data reduction and feature extraction***

Two of the most important applications of data reduction are image compression and feature extraction. In general, an image compression algorithm, used for storing and transmitting images, contains two steps: encoding and decoding. For both these steps, GAs has been used. Feature extraction is used for subsequent segmentation



or object recognition. The kind of features one wants to extract often correspond to particular geometric or perceptual characteristics in an image (edges, corners and junctions), or application dependent ones, e.g., facial features.

### **1.3.6 Feature extraction applications**

Feature extraction can be seen as a special kind of data reduction of which the goal is to find a subset of informative variables based on image data. Since image data are by nature very high dimensional, feature extraction is often a necessary step for segmentation or object recognition to be successful. Besides lowering the computational cost, feature extraction is also a means for

### **1.3.7 Image segmentation**

Segmentation is the partitioning of an image into parts that are coherent according to some criterion. When considered as a classification task, the purpose of segmentation is to assign labels to individual pixels or voxels. Some neural-based approaches perform segmentation directly on the pixel data, obtained either from a convolution window (occasionally from more bands as present in, e.g., remote sensing and MR images), or the information is provided to a neural classifier in the form of local features.

#### **1.3.7.1. Image segmentation based on pixel data**

Many GA approaches have been presented that segment images directly from pixel or voxel data. A self-organizing architecture with fuzziness measures was used in Ref. [86]. Also, biologically inspired neural-network approaches have been proposed: the perception model, which is able to segment images from surfaces and their shading. Hierarchical segmentation approaches have been designed to combine GAs on different abstraction levels. GAs are dedicated to low level feature extraction/segmentation, and their results are combined at a higher abstraction level where another classifier performs the final image segmentation.

### 1.3.8 Object recognition

Object recognition consists of locating the positions and possibly orientations and scales of instances of objects in an image. The purpose may also be to assign a class label to a detected object. Our survey of the literature on object recognition using GAs indicates that in most applications, GAs have been trained to locate individual objects based directly on pixel data. Another less frequently used approach is to map the contents of a window onto a feature space that is provided as input to a neural classifier.

#### 1.3.8.1 Object recognition based on pixel data

Several novel network architectures have been developed specifically to cope with concomitant object variations in position, (in-plane or out-of-plane) rotation and scale (in one case, an approach has been developed that is invariant to changes in illumination). It is clear that a distinction needs to be made between invariant recognition in 2D (projection or perspective) images and in 3D volume images. An interesting approach that performs object recognition, which is invariant to 2D translations, in-plane rotation and scale, is the neurally inspired what-and-where filter. It combines a multiscale oriented filter bank (what) with an invariant matching module (where). Other approaches rely on learning the variations explicitly by training. Egmont-Petersen and Arts built a statistical intensity model of the object that should be detected. The convolution GA was trained using synthetic images of the (modeled) object with randomly chosen orientations. These GAs were trained partly with synthetic sub images of nodules. Others have developed approaches that are invariant to both translation and 2D rotation, or systems that through their architectures perform processing in a translation-invariant way. Clearly, when object recognition is performed by teaching a classifier to recognize the whole object from a spatial pattern of pixel intensities, the complexity of the classifier grows exponentially with the size of the object and with the number of dimensions (2D versus 3D). An interesting approach that circumvents this problem is iterative search through the image for the object centre. The output of the GA is the estimated displacement vector to the object centre. Depending on the contents of the scene, even context information may be required before the objects of interest can be recognised with confidence. The incorporation of context information may again lead to a large number of extra parameters and thereby a more complex classifier. A special type of GA that incorporates the scale information directly in a pyramidal form is the so-called

higher-order GA. This network builds up an internal scale-space-like representation by what is called coarse coding. However, higher-order GAs need to learn variations in scale explicitly too. They should be used with caution because the coarse coding scheme may lead to aliasing, as the high-resolution images are not blurred before computing the coarser image at the next level. Rare conditions such as object occlusion or the occurrence of multiple objects within the (sub) image that is processed by the classifier have hardly been considered explicitly.

#### ***1.3.8.2 Using pixels or features as input?***

Most GAs that have been trained to perform image segmentation or object recognition obtain as input either pixel/voxel data (input level A) or a vector consisting of local, derived features (input level B). For pixel- and voxel-based approaches, all information (within a window) is provided directly to the classifier. The perfect (minimal error-rate) classifier should, when based directly on pixel data, be able to produce the best result if the size of the window is comparable to that of the texture elements (texels) or the window encompasses the object and the (discriminative) surrounding background. When, on the other hand, the input to the classifier consists of a feature vector, the image content is always compressed. Two-dimensional image modalities such as radiography, 2D ultrasound and remote sensing often exhibit concomitant variations in rotation and scale. If such invariance are not built into a pixel-based GA, careful calibration (estimation of the physical size of a pixel) and subsequent rescaling of the image to a standard resolution are required steps to ensure a confident result. When only rotations occur, features obtained from a polar mapping of the window may ensure a good segmentation or detection result.

A major disadvantage of these approaches is that object variations in rotation and scale have to be learned explicitly by the classifier (translation can usually be coped with by convolution). This again calls for a very large, complete training set and a classifier that can generalise well. Model-based approaches have been presented that can generate such a complete training set. How to design robust pixel-based algorithms for segmentation and object recognition that can cope with the three basic transformations, is a challenging subject for future research. In situations where many concomitant degrees of freedom occur (2D or 3D rotation, scale, a One grey level transformations, changes in colour, etc.), an



Another advantage of feature-based approaches is that variations in rotation and scale may remain unnoticed by the user, who may then end up with a poor result. When there is no limited set of images on which an algorithm has to work (e.g., image database retrieval), the more flexible pixel-based methods can prove useful. The recommendation to prefer feature-based over pixel/voxel-based image processing (when significant variations in rotation and scale actually occur in the image material), puts emphasis on the art of developing and choosing features which, in concert, contain much discriminative power in relation to the particular image processing task. Prior knowledge regarding the image processing task (e.g., invariance) should guide the development and selection of discriminative features. Feature-based classifiers will, in general, be easier to train when the chosen features cope adequately with the degrees of freedom intrinsic to the image material at hand. The removal of superfluous features is often necessary to avoid the peaking phenomenon and guarantee a good generalizations ability of the classifier.

### 1.3.9 What makes a GA special?

What are the trademarks of a genetic algorithm (GA)? A GA is heuristic, which means it estimates a solution. You won't know if you get the exact solution, but that may be a minor concern. In fact, most real-life problems are like that: you estimate a solution rather than calculating it exactly.

For most problems you don't have any formula for solving the problem because it is too complex, or if you do, it just takes too long to calculate the solution exactly. An example could be space optimization - it is very difficult to find the best way to put objects of varying size into a room so they take as little space as possible. The most feasible approach then is to use a heuristic method.

Genetic algorithms are different from other heuristic methods in several ways. The most important difference is that a GA works on a *population* of possible solutions, while other heuristic methods use a single solution in their iterations. Another difference is that GAs are *probabilistic* (stochastic), not deterministic.

Each individual in the GA population represents a possible solution to the problem. The suggested solution is coded into the "genes" of the individual. One individual might have these genes: "1100101011", another has these: "0101110001" (just examples). The values (0 or 1) and their position in the "gene string" tells the genetic algorithm what solution the individual represents.

Here comes the clever part: you apply the rules of evolution to the individuals. You find the individuals you think are the best suggestions to your problem and then combine these individuals into new individuals. Using this method repeatedly, the population will hopefully evolve good solutions. Specifically, the elements of a GA are: *selection* (according to some measure of fitness), *cross-over* (a method of reproduction, "mating" the individuals into new individuals), and *mutation* (adding a bit of random noise to the offspring, changing their "genes"). As you can see, Darwin's principles have been a major inspiration to GAs.

### 1.3.10 Advantages and disadvantages of GAs

A GA has a number of advantages. It can quickly scan a vast solution set. Bad proposals do not effect the end solution negatively as they are simply discarded. The inductive nature of the GA means that it doesn't have to know any rules of the problem - it works by its own internal rules. This is very useful for complex or loosely defined problems.

GAs have drawbacks too, of course. While the great advantage of GAs is the fact that they find a solution through evolution, this is also the biggest disadvantage. Evolution is inductive; in nature life does not evolve towards a good solution - it evolves *away* from bad circumstances. This can cause a species to evolve into an evolutionary dead end. Likewise, GAs risk finding a suboptimal solution.

Consider this example: a GA must find the highest point in a landscape. The algorithm will favor solutions that lie on a peak (a hill or whatever). As the individuals gradually begin to propose solutions in the same vicinity (somewhere on the peak), they begin to look alike. In the end you may have individuals that are almost identical. The best of these suggest the top of the peak as the solution. But what if there is another, higher peak at the other end of our

map? It is too hard for the individuals to venture away from their current peak. The ones that do will be eliminated, because their solution is worse than the ones we have. An individual might get "lucky", but that would mean its "genes" are very different from the rest of the population, so this is unlikely. In other words, the algorithm produces a suboptimal solution - and we may not even know it.

#### 1.4 OBJECTIVE :

The rate of expression of proteins like endothelin and CD31 in the placental cell depends on the rate of inhalation of tobacco smoke. It has been seen that in case of the women who smokes tobacco very frequently (active smoker ) the rate of expression of these two proteins is more, whereas in the second category of women who does not smoke tobacco but they inhale the smoke coming from other sources (passive smoker), the rate of expression is low. The higher expression of these two proteins leads to the placental abnormalities subjected to birth failure. Hence pathologically the study of these two protein and their expression is very important for proper medication. Traditionally the study can be done with the immunohistochemistry technique. The inference is generally derived based on the visualization of the image by an experienced pathologist. Since this is a manual method sometimes the diagnosis is erroneous and wrongly classify the images. In this regard it is essential to make an automatic tool using reproducible algorithm for prediction , classification and proper diagnosis. Hence in this studies an attempt has been taken to develop a tool using Image Processing and Genetic Algorithm with following objectives.

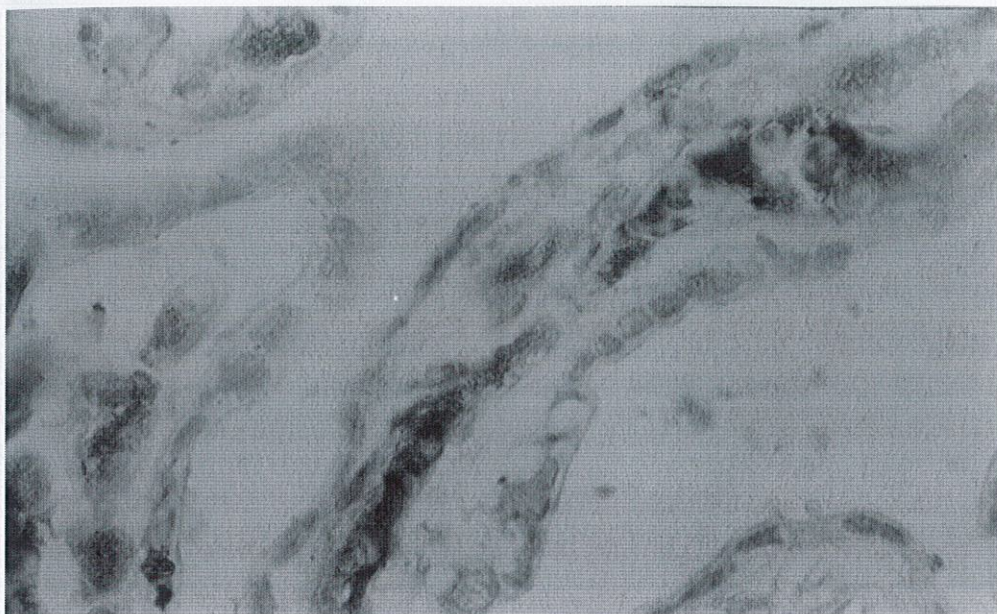
1. To develop an automatic tool for image processing and its validation with the case study.
2. To classify the images on the basis of RGB scan of the image obtained by the experimentation (immunohistochemistry) into the already specified classes for use in certain pathological procedures (active and passive smokers).
3. To predict the rate of expression of proteins like; Endothelin and CD31 in the placenta based on the difference in intensity of brown pixels from the image.

Fig 1.1 Placental Image of CD31 Positive Smoker.

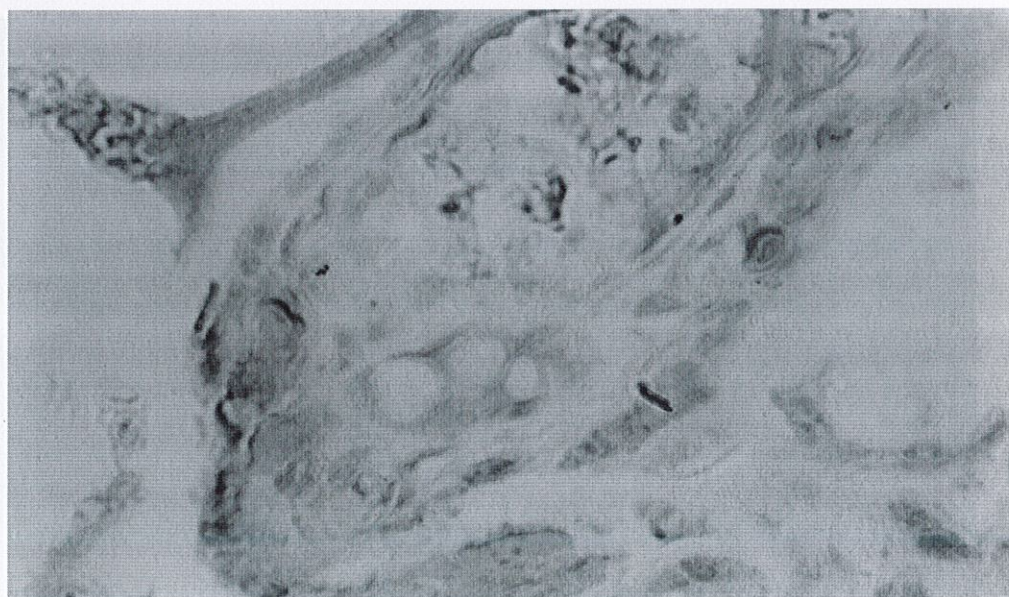




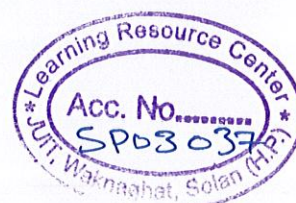
**Case Study:** In this study we have taken two sets of images of placenta from active and passive smoker and by using the tool developed here we want to predict the rate of expression of Endothelin and CD31 proteins.



**Fig 1.1** Placental Image of CD31 Active Smoker



**Fig 1.2** Placental Image of CD31 Passive Smoker.





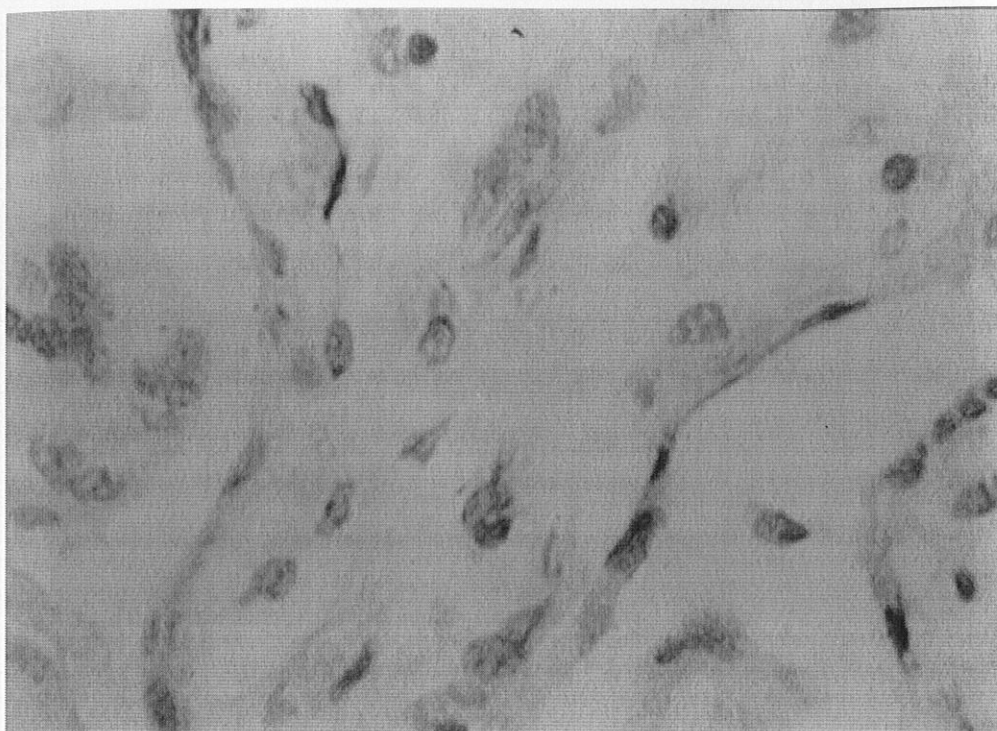
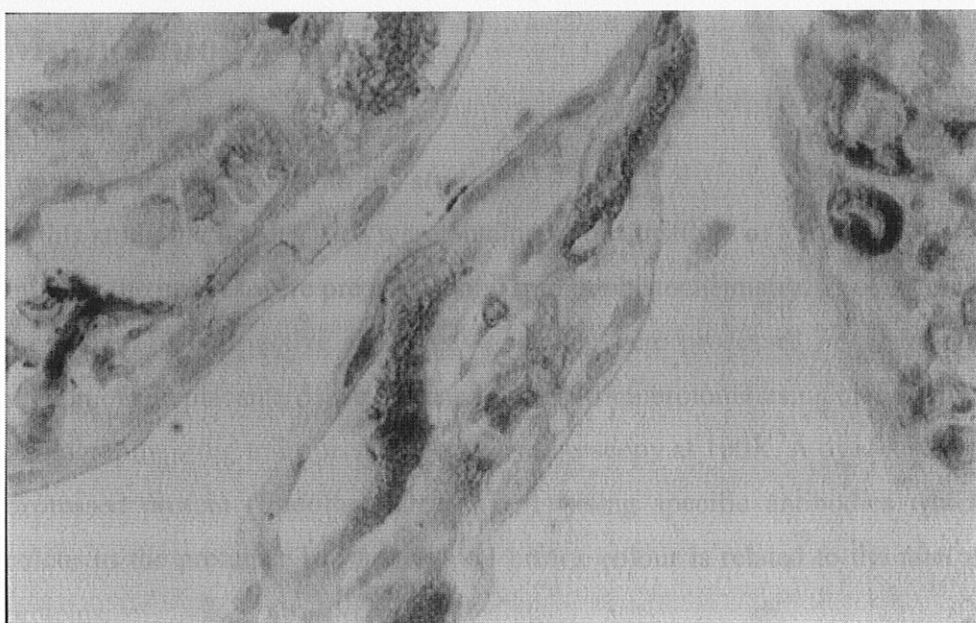


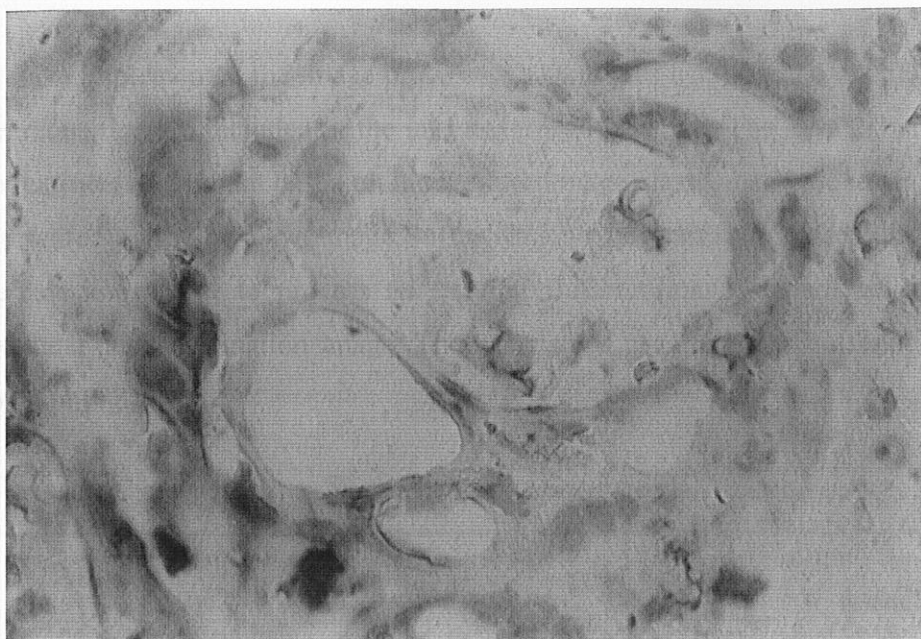
Fig 1.5 Placental Image of ET passive smoker

**Fig 1.3** Placental Image of a non- smoker

## CHAPTER 2



**Fig 1.4** Placental Image of a ET Active smoker



**Fig 1.5** Placental Image of ET passive smoker

## CHAPTER 2

### Methodology

#### Biomedical images used in the study

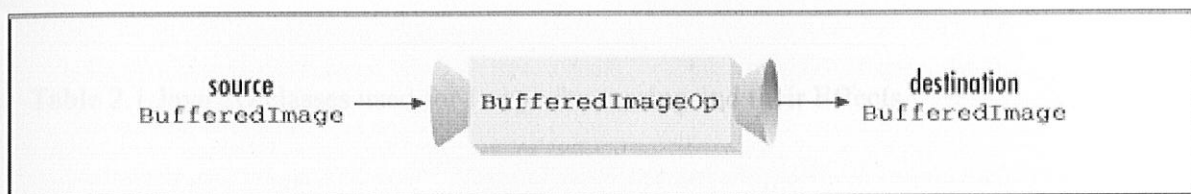
In this study the images used were obtained from Institute of Pathology, ICMR, New Delhi. Initially the images were prepared using immunohistochemistry. The placenta obtained from the patient of both active and passive smoker were processed in the laboratory. Ultrathin section of the placenta tissue obtained from ultramicrotome using carbon grid were used for photography using transmission electron microscopy at 100X. A dye was used to labeled the expressed protein (Endothelin and CD31) using specific antibodies which gives brown colour to the proteins. The intensity of brown colour is related to the rate of expression of proteins.

- Brightness, Darkening
- Color space conversion



## 2.1 Implementation of Image Processing Techniques

We have implemented the image processing techniques using the JAVA 2D API and image operations that are included in the java.awt.image package. The Java 2D API has a simple image processing model based on the BufferedImage class. Image processing operations are represented by the java.awt.image.BufferedImageOp interface. Classes that implement this interface know how to process an existing BufferedImage, called the source image, to produce a new destination image. The basic steps used here for buffering the images are shown in Figure 2.1.



**Figure 2.** Schematic Representation of BufferedImage Model for Image Processing

## 2.2 Image Filters and Predefined Operations in Java 2D

The 2D API comes complete with a handful of useful image operators. These are implementations of the BufferedImageOp interface. Table 2-1 summarizes the image operator classes that are defined in java.awt.image. The "In Place?" column indicates whether the operator is capable of in-place processing, where the source and destination image are the same.

Various Filters that can be created by using there classes are listed as under.

- Blur
- Sharpen
- Edge Detection
- Geometric transformation(zoom in-out)
- Brightening, Darkening
- Color space conversion

Class Name	Supporting Classes	Effect	In Place?
ConvolveOp	Kernel	Blurring, Sharpening, Edge Detection	No
AffineTransformOp	java.awt.geom.AffineTransform	geometric transformation	No
LookupOp	LookupTable, ByteLookupTable, ShortLookupTable	posterizing, inversion	Yes
RescaleOp		brightening, darkening	Yes
BandCombineOp	java.awt.color.ColorSpace	color space conversion	Yes

Table 2.1 Java 2D classes used for Image Processing and their Effects.

**1. The Blur Filter** - ConvolveOp is one of the most versatile image operators in the 2D API. It represents a spatial convolution, which is a fancy mathematical term that means that the color of each destination pixel is determined by combining the colors of the corresponding source pixel and its neighbors. A convolution operator can be used to blur or sharpen an image, among other things.

**Module:**

```
public BufferedImage processImage(BufferedImage image) {
    float[] blur Matrix = { 1.0f / 9.0f, 1.0f / 9.0f, 1.0f / 9.0f,
                           1.0f / 9.0f, 1.0f / 9.0f, 1.0f / 9.0f,
                           1.0f / 9.0f, 1.0f / 9.0f, 1.0f / 9.0f};
    BufferedImage blur Filter = new ConvolveOp (new Kernel(3, 3, blur Matrix),
        ConvolveOp.EDGE_NO_OP, null);
    return blurFilter.filter(image, null);
}
```

The convolution operator uses a kernel to specify how a source pixel and its neighbors are combined. A kernel is simply a matrix, where the center of the matrix represents the source pixel and the other elements correspond to the neighboring source pixels. The destination

color is calculated by multiplying each pixel color by its corresponding kernel coefficient and adding the results together.

2. **The Change Color Filter** – BandCombineOp is also one of image operators in the 2D API. It represents a spatial color matrix, which is applied to source and display destination image for a random color change event of a pixel

**Module:**

```
public BufferedImage processImage(BufferedImage image) {
    float[][] colorMatrix = { { 1f, 0f, 0f },
                              { 0.5f, 1.0f, 0.5f },
                              { 0.2f, 0.4f, 0.6f } };

    BandCombineOp changeColors = new BandCombineOp(colorMatrix, null);
    Raster sourceRaster = image.getRaster ();
    WritableRaster displayRaster = sourceRaster.createCompatibleWritableRaster ();
    changeColors.filter (sourceRaster, displayRaster);
    return new BufferedImage(image.getColorModel(), displayRaster, true, null);
}
```

The above module accepts a Buffered Image and returns a Buffered Image after a color change multiplied by color matrix.

3. **The Edge Detection Filter** - The goal of edge detection is to mark the points in a digital image at which the luminous intensity changes sharply. Sharp changes in image properties usually reflect important events and changes in properties of the world. These include

- Discontinuities in depth,
- Discontinuities in surface orientation,
- Changes in material properties and
- Variations in scene illumination.

Edge detection of an image reduces significantly the amount of data and filters out information that may be regarded as less relevant, preserving the important structural properties of an image. There are many methods for edge detection, but most of them can be grouped into two categories, search-based and zero-crossing based. The search-based



methods detect edges by looking for maxima and minima in the first derivative of the image, usually local directional maxima of the gradient magnitude. The zero-crossing based methods search for zero crossings in the second derivative of the image in order to find edges, usually the zero-crossings of the Laplacian or the zero-crossings of a non-linear differential expression.

Edges may be viewpoint dependent - these are edges that may change as the viewpoint changes, and typically reflect the geometry of the scene, objects occluding one another and so on, or may be viewpoint independent - these generally reflect properties of the viewed objects such as surface markings and surface shape. In two dimensions, and higher, the concept of perspective projection has to be considered.

A typical edge might be (for instance) the border between a block of red color and a block of yellow; in contrast a line can be a small number of pixels of a different color on an otherwise unchanging background. There will be one edge on each side of the line. Edges play quite an important role in many applications of image processing. During recent years, however, substantial (and successful) research has also been made on computer vision methods that do not explicitly rely on edge detection as a pre-processing step.

#### **Module:**

```
public BufferedImage processImage(BufferedImage bi)
{
    BufferedImage buff = new BufferedImage(bi.getWidth(), bi.getHeight(),
    bi.getType());

    Kernel kernel = new Kernel(3, 3, new float[] {
        -1f, -2f, -1f,
        0f, 0f, 0f,
        1f, 2f, 1f
    });

    ConvolveOp op = new ConvolveOp(kernel);
    op.filter(bi, buff);

    return buff;
}
```

The ConvolveOp operator does the same job here as in the case of blurring an image.

4. **The Invert Filter**- This Filter uses LookupOp operator class to invert image colors. Here either the values of red and blue are swapped byte by byte using the ByteLookupTable class or values of rgb for each pixel are subtracted from 255. Values of green may be left as it is or may be changed accordingly o

Module:

```
public BufferedImage processImage (BufferedImage image) {
    byte[] invertArray = new byte[256];
    for (int counter = 0; counter < 256; counter++)
        invertArray[counter] = (byte) (255 - counter);
    BufferedImageOp invertFilter = new LookupOp (new ByteLookupTable (0, invertArray),
    null);
    return invertFilter.filter (image, null);
}
```

The above module creates a byte array of length 256 that ranges from 0-255 which is the range of red , green and blue values of a pixel. It retrieves each pixel of the supplied BufferedImage and analyze its rgb values these rgb values are the subtracted from 255 to generate inverted buffered image using LookupOp operator and ByteLookupTable Class.

5. **The Sharpen Filter** - Sharpening is one of the most impressive transformations you can apply to an image since it seems to bring out image detail that was not there before. What it actually does, however, is to emphasize edges in the image and make them easier for the eye to pick out -- while the visual effect is to make the image seem sharper, no new details are actually created. Paradoxically, the first step in sharpening an image is to blur it slightly. Next, the original image and the blurred version are compared one pixel at a time. If a pixel is brighter than the blurred version it is lightened further; if a pixel is darker than the blurred version, it is darkened. The result is to increase the contrast between each pixel and its neighbors. The nature of the sharpening is influenced by the blurring radius used and the extent to which the differences between each pixel and its neighbor are exaggerated.

### Oversharpening

If you continue to sharpen an image, several things happen:

- Edges become unnaturally pronounced—dark objects are outlined with light Halos and light objects are outlined with dark halos.
- Normally invisible noise in the image is amplified and starts to show up as texture in areas that looked smooth in the original images. This can create an undesirable graininess in parts of photographs that should be smooth like clouds and skies.
- Extreme sharpening causes the image to break up as each individual pixel stands out more and more from its neighbors.

Thus as you progressively sharpen a slightly blurry image, it first starts to look better, but then as you begin to over-sharpen it, it starts to look worse again. Once you know what over-sharpening looks like, you will be able to spot it on the screen and back off a little

The Module given below accepts a buffered image and a sharpen matrix which is multiplied to rgb values of each pixel of image edges that are analyzed by ConvolveOp operator class.

#### Module:

```
public BufferedImage processImage(BufferedImage image) {

    float[] sharpenMatrix = { 0.0f, -1.0f, 0.0f,
                              -1.0f, 5.0f, -1.0f,
                              0.0f, -1.0f, 0.0f };

    BufferedImageOp sharpenFilter = new ConvolveOp(new Kernel(3, 3, sharpenMatrix),
        ConvolveOp.EDGE_NO_OP, null);

    return sharpenFilter.filter(image, null);

}
```

**6. Zoom-In Filter** – Following module does the job of zooming in the images. It multiplies the image size by 2 .

#### Module



```

public BufferedImage processImage(BufferedImage bi)
{
    int scale =2;
    int width = scale * bi.getWidth();
    int height = scale * bi.getHeight();
    BufferedImage biScale = new BufferedImage(width, height, bi.getType());
    for(int i=0; i<width; i++)
        for(int j=0; j<height; j++)
            biScale.setRGB(i, j, bi.getRGB(i/scale, j/scale));
    return biScale;
}

```

The scale can be changed accordingly to get more larger images.

**7. Zoom-Out Filter** – Following module does the job of zooming out the images. It divides the image size by 2 .

#### Module

```

public BufferedImage processImage(BufferedImage bi)
{
    int scale =2;
    int width = bi.getWidth()/scale ;
    int height = bi.getHeight()/scale;
    BufferedImage biScale = new BufferedImage(width, height, bi.getType());
    for(int i=0; i<width; i++)
        for(int j=0; j<height; j++)
            biScale.setRGB(i, j, bi.getRGB(i/scale, j/scale));
    return biScale;
}

```

## 2.3 GUI Components

As mentioned earlier the GUI was Implemented using Java Foundation Classes also known as JFC/Swing. The basic task of GUI was to load and display an Image. Handle various events from menu and buttons to provide user friendly interface to Image Filtering algorithms and to the Prediction of Active and Passive Smoker.

Following are the predefined classes defined in javax.swing package where javax stands for “Java Extension” that are used in making the GUI Components

1. JMenuBar : Creates a menu bar.
2. JMenu : Creates a pull-down menu
3. JMenuItem : Creates a menu item
4. JComponent : A subclass of component that can hold other components
5. JPanel : Provides general purpose containers for lightweight components
6. JScrollPane : A container that provides horizontal and / or vertical scroll bars for another component.
7. JLabel : Creates a label that displays a string or image or both
8. JTextField : Creates a single-line edit control
9. JButton : Creates a push button control
10. JCheckBox : Creates a check box control
11. JInternalFrame : Ceates an internal-window that has a title bar, resize corners and a menu bar and that can fit into JFrame
12. JFrame : Ceates a standard window that has a title bar, resize corners and a menu bar

### 2.3.1 Events handling Components in GUI

- Following are the predefined classes defined in java.awt.event package where awt stands for “Abstract Window Toolkit” that are used to tackle the events occur on the components

1. ActionListener : Defines one method, actionPerformed(), to receive action events.

2. ItemListner : Defines one method, `itemStateChanged()`, that is invoked when the state of an item changes.
3. KeyListner : Defines three methods, `keyPressed()`, `keyReleased()` and `keyTyped()`, are invoked when a key pressed, released and typed.
4. MouseListener : Defines five methods, `mouseClicked()`, `mouseEntered()`, `mouseExited()`, `mousePressed()`, `mouseReleased()`, to recognize when the mouse is clicked, enters a component, exits a component, is pressed, or is released.
5. WindowListener : Defines seven methods to recognize when a window is activated, close, deactivated, deiconified, iconified, opened, or quit.

### 2.3.2 ColorModels and RGB Values for Image analysis

A color model is a method to specify colors with respect to some reference framework. The color models can be grouped in additive color models and subtractive color models. In an **additive color model**, any combination of two or more colors results in a color with higher luminance than the original colors. An example of how an additive color model work is the cathodic ray tube found in TVs and computer monitors; a white point results from the combination of red, green and blue points. In a **subtractive color model**, any combination of two colors gives a color with lower luminance. An example of a subtractive color model working can be found in mixing watercolor; the resulting color becomes darker and darker, and ideally it will finally becomes black, although it will not due to the characteristics of the pigments. The most commonly used color models are

- RGB
- CYM
- HIS

We have initially applied all the above three color models about we found that the RGB model is most suitable for our case study.

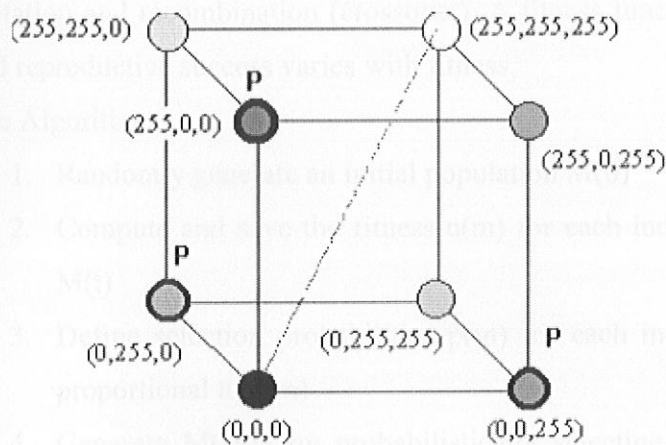
### 2.3.3 RGB (Red-Green-Blue) ColorModel



The RGB color model or RBG color standard (often spelled RBG in historical engineering literature) is an additive model in which red, green, and blue (often used in additive light models) are combined in various ways to reproduce other colors. The name of the model and the abbreviation 'RGB' come from the three primary colors, red, green, and blue and the technological development of cathode ray tubes which could display color instead of a monochrome phosphorescence (including grey scaling) such as black and white film and television imaging.

The term RGBA is also used, to mean Red, Green, Blue, Alpha. This is not a different color model, but a representation; the Alpha is used for transparency. These three colors should not be confused with the primary pigments of red, blue, and yellow, known in the art world as 'primary colors', as the latter combine based on

#### RGB Color Space:



**Fig 2.1** RGB Color Space. The colors with a **P** are the primary colors. The dashed line indicates where to find the grays, going from (0,0,0) to (255,255,255).

reflection and absorption of photons whereas RGB depends on emission of photons from a compound excited to a higher energy state by impact with an electron beam.

The RGB color model itself does not define what is meant by 'red', 'green' and 'blue' (spectroscopically), and so the results of mixing them are not specified as exact (but relative, and averaged by the human eye).

## 2.4 Algorithms used for prediction and classification of images

In the tool developed here we have used the following prediction algorithms to predict the rate of expression of both the proteins in terms of intensity and their classification into active or a passive smoker.

- Genetic Algorithms(GA)
- Nearest Neighbor(NN)
- Image Region Analysis(IRA)

### 2.4.1 Genetic Algorithm

#### 2.4.1.1 Brief Overview

GAs were introduced as a computational analogy of adaptive systems. They are modeled loosely on the principles of the evolution via natural selection, employing a population of individuals that undergo selection in the presence of variation-inducing operators such as mutation and recombination (crossover). A fitness function is used to evaluate individuals, and reproductive success varies with fitness.

The Algorithms

1. Randomly generate an initial population  $M(0)$
2. Compute and save the fitness  $u(m)$  for each individual  $m$  in the current population  $M(t)$
3. Define selection probabilities  $p(m)$  for each individual  $m$  in  $M(t)$  so that  $p(m)$  is proportional to  $u(m)$
4. Generate  $M(t+1)$  by probabilistically selecting individuals from  $M(t)$  to produce offspring via genetic operators
5. Repeat step 2 until satisfying solution is obtained.

The paradigm of GAs described above is usually the one applied to solving most of the problems presented to GAs. Though it might not find the best solution, more often than not, it would come up with a partially optimal solution.

Fig 2.3 Flow Chart to Depict the flow of control in GA

Genetic Algorithms (GAs) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem.

First pioneered by John Holland in the 60s, Genetic Algorithms has been widely studied, experimented and applied in many fields in engineering worlds. Not only does GAs provide alternative methods to solving problem, it consistently outperforms other traditional methods in most of the problems link

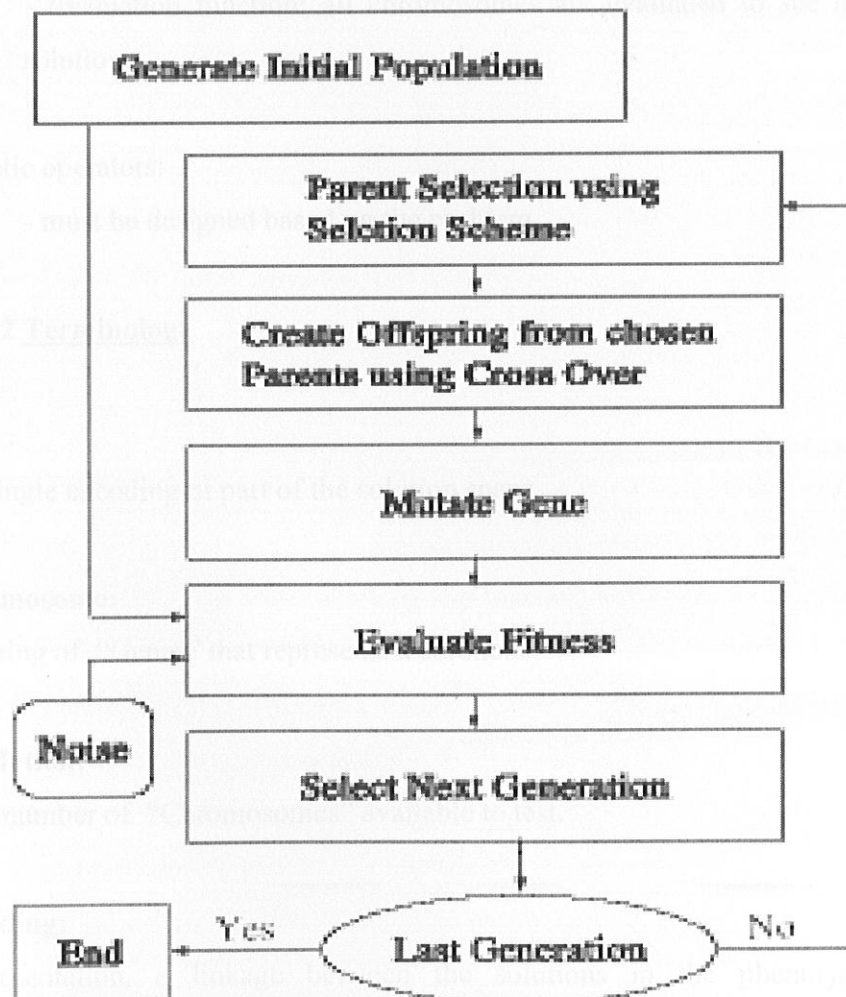


Fig 2.2 Flow Chart to Depict the flow of control in GA



Many of the real world problems involved finding optimal parameters, which might prove difficult for traditional methods but ideal for GAs. However, because of its outstanding performance in optimization, GAs has been wrongly regarded as a function optimizer.

Two basic processes from evolution:

- Inheritance
- Natural selection

What must be developed:

- Initialization of population: randomly to sample the search space uniformly without bias (robustness).
- Evaluation function: all chromosomes are evaluated to see how fit they are as solutions.

Genetic operators:

- must be designed based on the problem.

#### **2.4.1.2 Terminology**

##### **Gene :**

An single encoding of part of the solution space.

##### **Chromosome:**

A string of "Genes" that represents a solution.

##### **Population:**

The number of "Chromosomes" available to test.

##### **Encoding:**

Representation, a linkage between the solutions in the phenotype space and the chromosomes is the genotype space (binary or real).

**Selection:**

Darwinian natural evolution, fitter individuals have a greater chance to reproduce offspring  
**“survival of the fittest”**.

- Parent selection to create offspring.
- Generational selection from both parents and offspring.

**Crossover:**

Operator to be applied to two chromosomes to generate two offspring. Combines the schemata from two different solutions in various combinations (most important)

**Mutation:**

After creation of new individuals via crossover, mutation is applied usually with a low probability to introduce random changes into the population.

- replace gene values lost from the population or not initially present.
- evaluate more regions of the search space.
- avoid premature convergence.
- makes the entire search space reachable.

**Linkage:**

- Majority of DNA is non-coding.
- Modular genetic structure can develop due to the variability in non-coding segment length (or linkage).
- Certain blocks become more likely to stay together during recombination.
- Integral part of GA.
- In GA: probability that two genes will be separated after recombination.
- Adjacent genes have tighter linkage.
- Crossover is more likely between non-adjacent genes
- If linkages are allowed to adapt -> modularity.
- Evolution of modularity allows exploitation of good building blocks while exploring other viable solutions.

### 2.4.1.3 Basic operators: selection

#### 1. Fitness Proportionate Selection (FPS)

##### ➤ Scaling

#### 2. Rank-Based Selection

#### 3. Tournament Selection

#### 4. Deterministic Selection

### Fitness proportionate selection (FPS)

For chromosome  $k$  with fitness value  $F_k$  and selection probability  $p_k$ . It assigns selection probability to an individual based on its fitness (must scale).

Fitness scaling is needed:

- if range is too large ...
- if range is too small ...

### Scaling mechanisms:

#### (1) Linear:

- $a$  and  $b$  are normally selected such that an average chromosome receives one offspring copy on average, and the best receives the specified number of copies (usually two).
- Best chromosome gets a fixed number of expected offspring! prevents it from reproducing too many offspring.

#### (2) Sigma Truncation (Goldberg, 1989):

- $c$  is the sigma scaling factor,  $F$  is the average raw fitness value.
- Negative scaled fitness values are set to zero.
- Any individual worse than  $c$  standard deviations is not selected.
- Value of  $c$  in the literature is between 1 and 5.

#### (3) BoltzmgA Selection (Michalewicz, 1994):

- Nonlinear scaling method for proportionate selection.



- T is a user-defined control parameter.
- The selection pressure can be adjusted by assigning T high or low.

#### 2.4.1.4 Rank-Based Selection:

Selection probability based on rank instead of its raw fitness .

Two methods to map rank into reproductive fitness:

- (1) linear ranking
- (2) exponential ranking

(1) Linear ranking:

- q is the reproductive fitness for the best chromosome, and  $q_0$  is the reproductive fitness for the worst chromosome.
- setting  $q_0$  to zero! maximum selective pressure.

(2) Exponential ranking (Michalewicz, 1994):

- Larger value for q implies larger selective pressure
- Hancock proposed the following exponential ranking method where q is typically  $\sim 0.99$ . The best chromosome has a fitness of 1, and the last one receives q (pop size - 1).

#### 2.4.1.5 Tournament selection:

Two kinds: binary and standard

(1) Binary:

- Two individuals taken at random and the better is selected from the two.
- if done without replacement, the two individuals are set aside for the next selection operation.
- the original population size is restored after the new population is half filled.
- the best individual will be selected twice and the worst won't be selected.
- the number of copies selected of any individual can't be predicted (either zero, one, or two).

**(2) Standard Tournament Selection:**

- Random uniform sample of a certain size  $q > 1$  is taken from the population.
- select the best of these  $q$  individuals to survive the next generation.
- Very popular due to easy implementation, computational efficiency, and fine-tuning capability by increasing or decreasing  $q$ .

**2.4.1.6 Deterministic Selection:****(1) Generational replacement:**

- replaces the entire parent generation with their offspring .

**(2) Elitist selection:**

- Problem with FPS: no guarantee for best to be passed.
- Elitism: elitist selection guarantees asymptotic convergence.

**(3) The  $(\mu, \lambda)$  evolution strategy:**

- $\mu$  parents create  $\lambda > \mu$  offspring by means of recombination and mutation.
- The best offspring individuals are deterministically selected to replace the parents.
- This method allows for the best member of generation  $t+1$  to perform worse than the best individual at generation  $t$ .

**2.4.1.7 Basic operators: crossover**

Binary crossover

Real crossover:

**(1) Simple crossover****(2) Random crossover****(3) Arithmetic crossover****2.4.1.8 Binary crossover:**

This process follow the simple rule of genetic process called crossover, which is known to all as the two parental chromosomes (homologous) exchange fragments which led to the formation of new combination for the future offsprings and provide the individual identity

which is different from their parents, which may be even more successful than the parental combination.

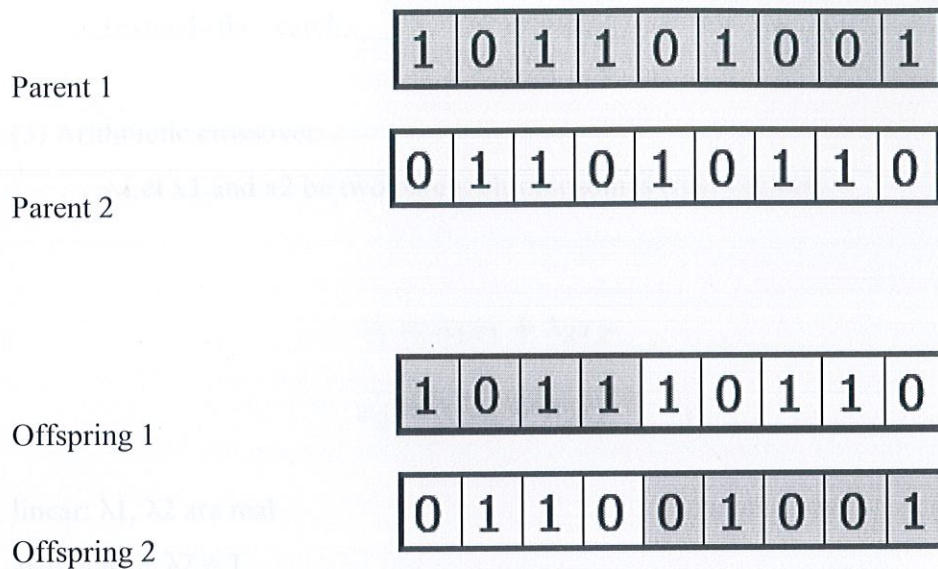


Fig 2.3 Binary crossover

### Real crossover

- Each chromosome is encoded as a vector of real numbers with the same length as the solution vector.

- Most belong to three categories:

- (1) Simple crossover
- (2) Random crossover
- (3) Arithmetic crossover

(1) Simple crossover:

- Analogous to binary (one or multi-point).
- Only difference real numbers are used instead of binary bits.



## (2) Random crossover:

- Flat: produces an offspring by uniformly picking a value for each gene from the range formed by the values of two corresponding parents' genes.
- Blend (BLX- $\alpha$ ): also picks values from a range but not formed by the parents genes (extends the search).

## (3) Arithmetic crossover:

- Let  $x_1$  and  $x_2$  be two parent chromosomes (real vectors):

$$x'_1 = \lambda_1 x_1 + \lambda_2 x_2$$

$$x'_2 = \lambda_1 x_2 + \lambda_2 x_1$$

linear:  $\lambda_1, \lambda_2$  are real

affine:  $\lambda_1 + \lambda_2 = 1$

convex:  $\lambda_1 + \lambda_2 = 1, \lambda_1 > 0, \lambda_2 > 0$

#### 2.4.2 Basic operators: mutation

- Crossover limitations.
- Replace a gene (real number) with a randomly selected real number within a specified range.
- Another kind is "boundary mutation" which replaces a gene with either the lower bound or the upper bound.
- Mutation is implemented as a background operator and is usually applied with low probability.
- If applied at high probability, the offspring will lose their resemblance to their parents and the ability of the algorithm to learn from the parents will be lost.

### 2.4.3 The Basic Genetic Algorithm

1. **[Start]** Generate random population of  $n$  chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness  $f(x)$  of each chromosome  $x$  in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete
  - a. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
  - b. **[Crossover]** With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
  - c. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
  - d. **[Accepting]** Place new offspring in the new population
4. **[Replace]** Use new generated population for a further run of the algorithm
5. **[Test]** If the end condition is satisfied, stop, and return the best solution in current population
6. **[Loop]** Go to step 2

### Recommended Parameters

#### Crossover rate:

Crossover rate should be high generally, about 80%-95%. (However some results show that for some problems crossover rate about 60% is the best.)

#### Mutation rate:

On the other side, mutation rate should be very low. Best rates seems to be about 0.5%-1%

#### Population size:

- Very big population size usually does not improve performance of GA (in the sense of speed of finding solution)
- Good population size is about 20-30, however sometimes sizes 50-100 are reported as the best.
- Some research also shows, that the best population size depends on the size of encoded string (chromosomes). It means that if you have chromosomes with 32 bits, the population should be higher than for chromosomes with 16 bits.

#### **Selection:**

Basic roulette wheel selection can be used, but sometimes rank selection can be better.

#### **Encoding:**

Encoding depends on the problem and also on the size of instance of the problem

#### **Crossover and mutation type:**

Operators depend on the chosen encoding and on the problem

### **2.4.4 NEAREST NEIGHBOR (NN)**

We are concerned with the following problem: we wish to label some Test image  $x$  with some class category.

- CD31 Active
- CD31 Passive
- ET Active
- ET Passive

We have complete statistical knowledge of the distribution of observation  $x$  and category . In this case, a standard Bayes analysis yields an optimal decision procedure.

Following Sequential Steps can result into classification of this image into an appropriate class.

#### **2.4.4.1 NN - ALGORITHM**



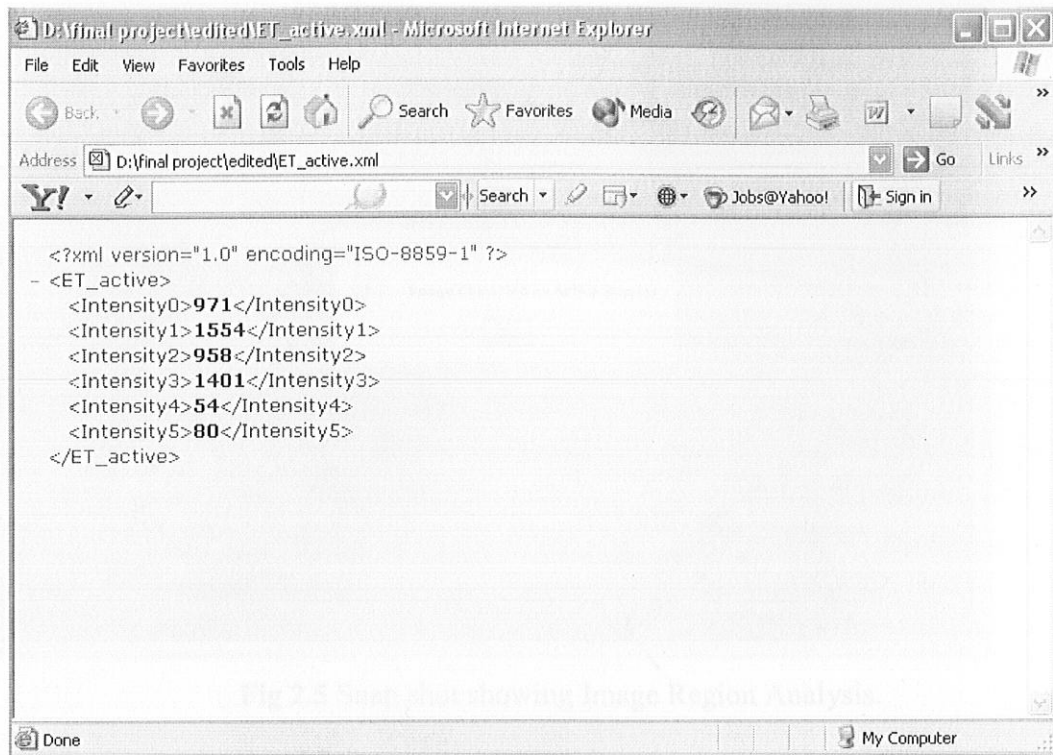
- Take all images of a particular class say CD31 Active analyze every pixel and derive the intensity of brown spot (which is directly proportional to no. of brown pixels in that image falling into certain range of RGB values) in each image.
- Write it down to an XML file tagged as CD31 Active say CD31\_Active.xml, this signifies that intensities registered in this file belongs to images falling under category CD31 Active.
- Repeat step 1 to 2 for other image classes like CD31 Active, CD31 Passive, ET Active and ET Passive.
- Now analyze the test image, find out the intensity of brown spot for this image.
- Find out the average of linear distance with all intensities registered under a particular class.
- Cross compare the average linear distance a class with all other classes.
- The class with minimum average linear distance is designated as the most nearest class and we can say that Test image may belong to that class.

Fig 2.4 Showing ET\_Active.xml contents

## 2 4.5 IMAGE REGION ANALYSIS (IRA)

Image Region analysis is a useful analogical technique to distinguish between active and passive smoker. It analyzes the clipped image region for the intensity of brown pixel. Fig 3.3 shows how an Image region can be clipped to find out intensity of brown pixels within a selected region.

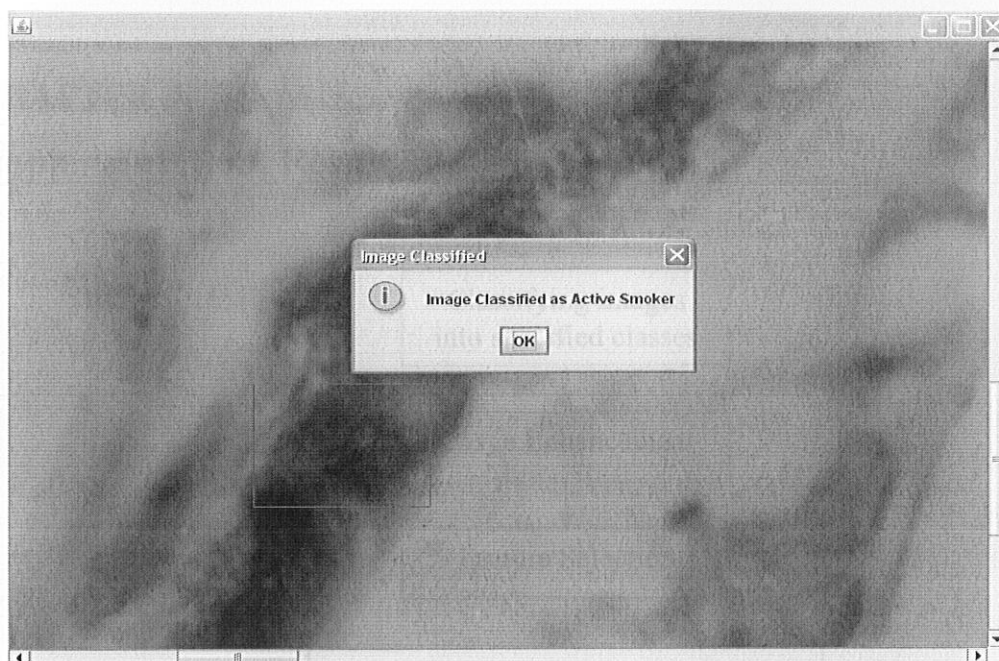
- Compare the intensity with already given thresholds
- if intensity > threshold then image belongs to a passive smoker
- if intensity < threshold then image belongs to active smoker



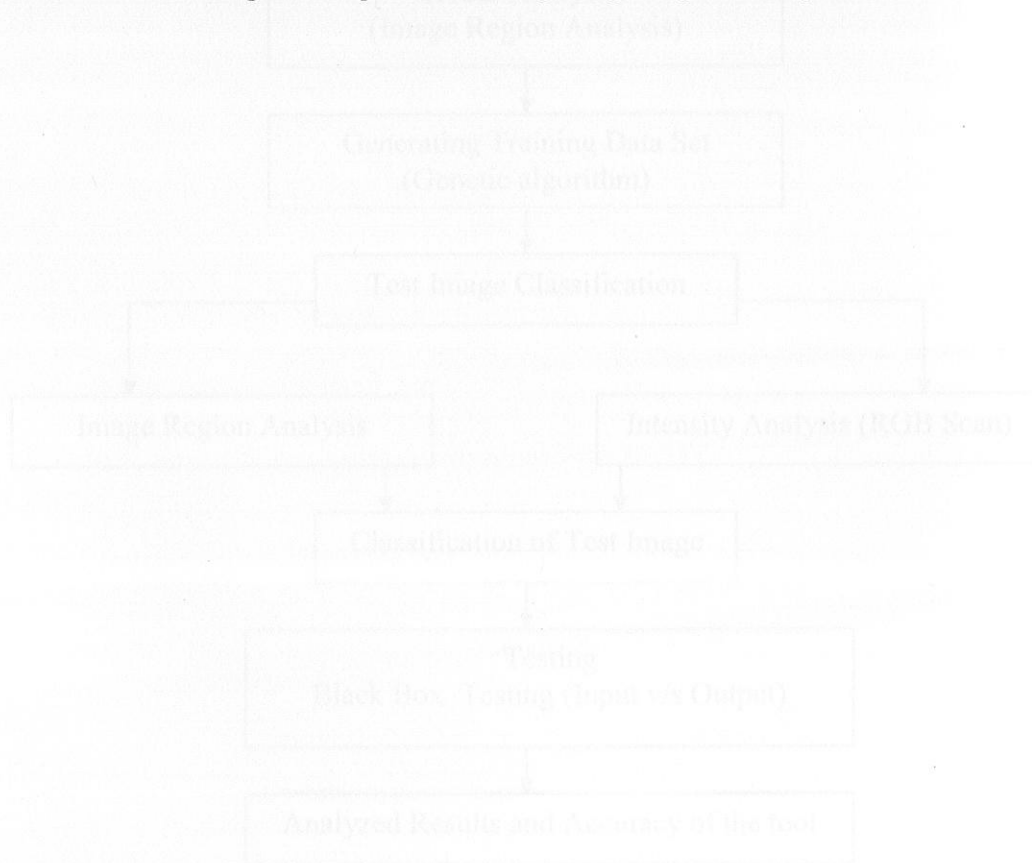
**Fig 2.4** Showing ET\_active .xml contents

### IRA Algorithm

- Clip an image
- Select the clipped image and convert it to buffered image
- Supply the buffered image for RGB analysis and analyze the intensity of brown pixels.
- Compare the intensity with already given thresholds
- If intensity < threshold then image belongs to a passive smoker
- If intensity > threshold then image belongs to active smoker



**Fig 2.5** Snap shot showing Image Region Analysis.



**Fig 3.** Overall flow control of B-MPT



## CHAPTER 3

### Implementation, Result and Discussion

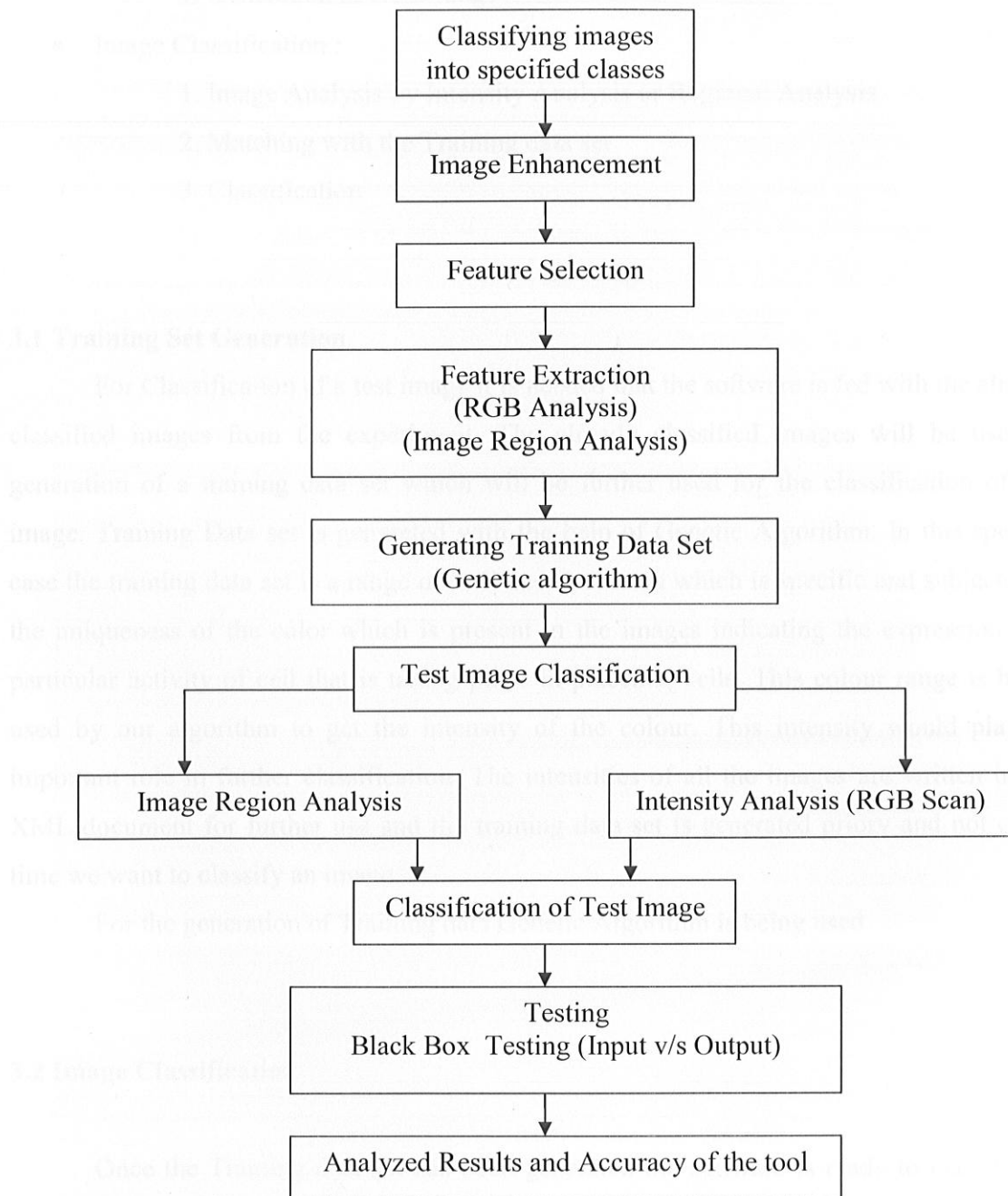


Fig 3. Overall flow control of B-MIPT

Now all the tools and techniques already described, we would now like to proceed with the actual classification process which involved all the above discussed applications. Whole of the classification process has been classified in which are as follows :

- Training Set Generation

1. Generation of RGB range

- Image Classification :

1. Image Analysis by Intensity Analysis or Regional Analysis.

2. Matching with the Training data set

3. Classification

### 3.1 Training Set Generation

For Classification of a test image it is needed that the software is fed with the already classified images from the experiment. The already classified images will be used in generation of a training data set which will be further used for the classification of test image. Training Data set is generated with the help of Genetic Algorithm. In this specific case the training data set is a range of R G and B colours which is specific and subjected to the uniqueness of the color which is present in the images indicating the expression of a particular activity of cell that is taking place in placental cells. This colour range is being used by our algorithm to get the intensity of the colour. This intensity would play an important role in further classification. The intensities of all the images are written into a XML document for further use and the training data set is generated priory and not every time we want to classify an image.

For the generation of Training data Genetic Algorithm is being used

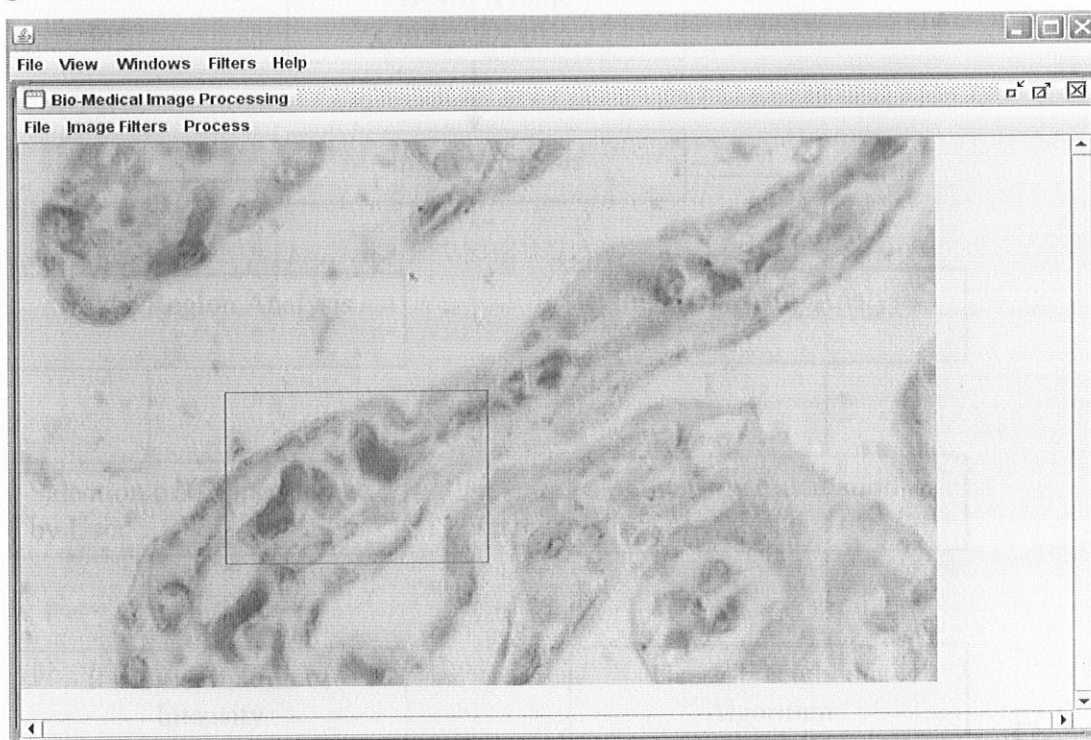
### 3.2 Image Classification

Once the Training data set has been generated the software is ready to classify the images into two different classes. The test image which is being loaded by the user into the tool into the main frame is being loaded with the help of Image. The tool provides a utility to classify the image by two different algorithms which will be specific the image size and

quality, still the use of algorithm is subjected to user's choice. The two algorithms are as follows

### 1. Image Region Analysis

This algorithm is recommended in the case of bad quality of image clarity and resolution, i.e. if we could see that the image is not so clear and only some particular regions of image are of good quality and can be used for classification. Under this algorithm the user is being given a choice of selection of a particular area which can be selected by the user which according to the user has some relevance to the classification. The region is being analyzed and the intensity is being calculated, by the intensity is being normalized according to the size of the area which is being selected by the user. This gives us the ability to classify the image according to the training data set which is being generated on images of a fixed resolution i.e. 600 X 400. The algorithm reflects good chances to classify the image.



**Fig 3.1 : IRA Snapshot**

### 2. Image Intensity Analysis

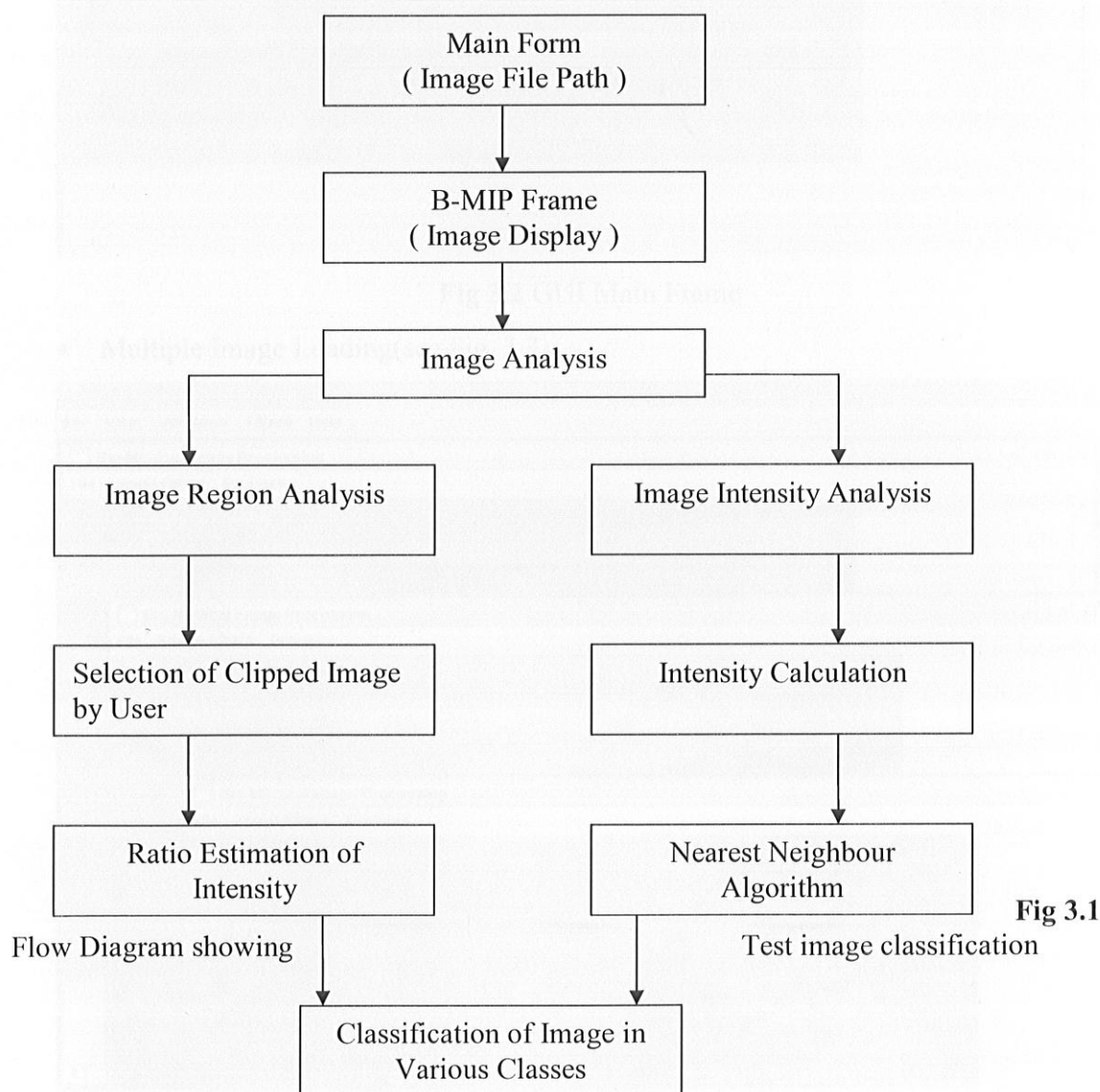
This algorithm is the default algorithm of the tool. In this algorithm the image loaded by the user is being analyzed and the intensity of the image is being



calculated with the help of training data set and RGB scan. The intensity is being calculated and written to an XML file. The intensity is then used by the nearest neighbour algorithm to classify the image into the given classes.

### 3.3 Main Features\_The main features of this software include

- Java's Platform IndependenceEnhanced User Friendly Graphical User Interface
- Genetic Algorithms
- Image region analysis
- Nearest neighbor method



### 3.3.1 GUI Features

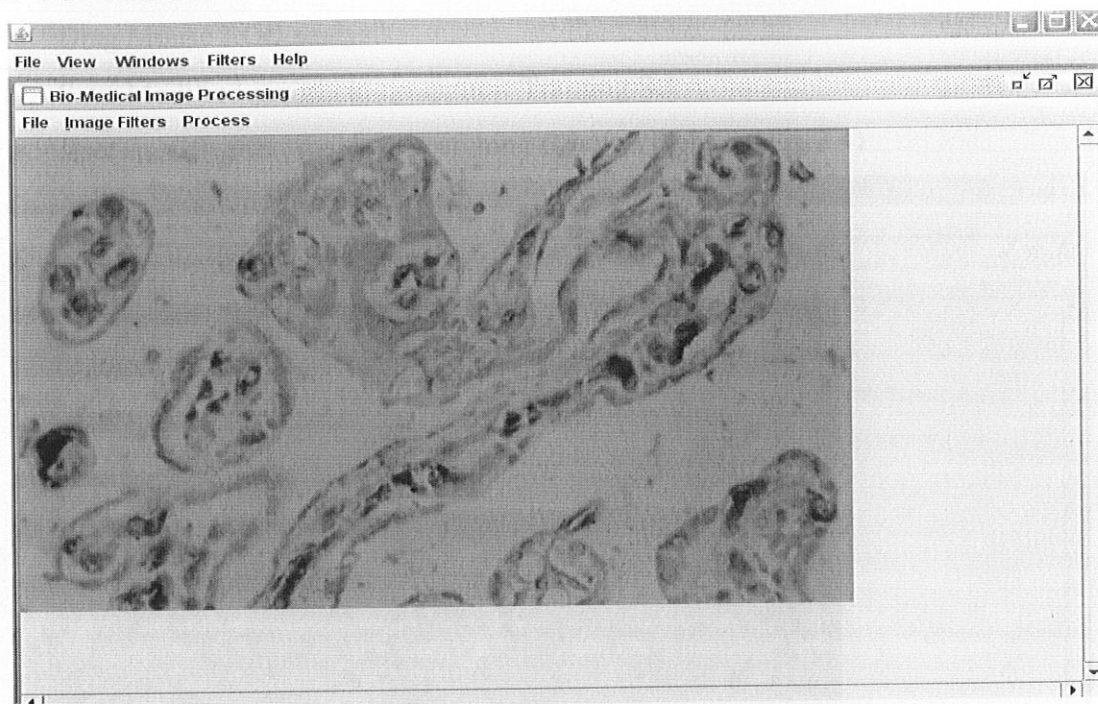


Fig 3.2 GUI Main Frame

- Multiple Image Loading(see Fig. 3.3)

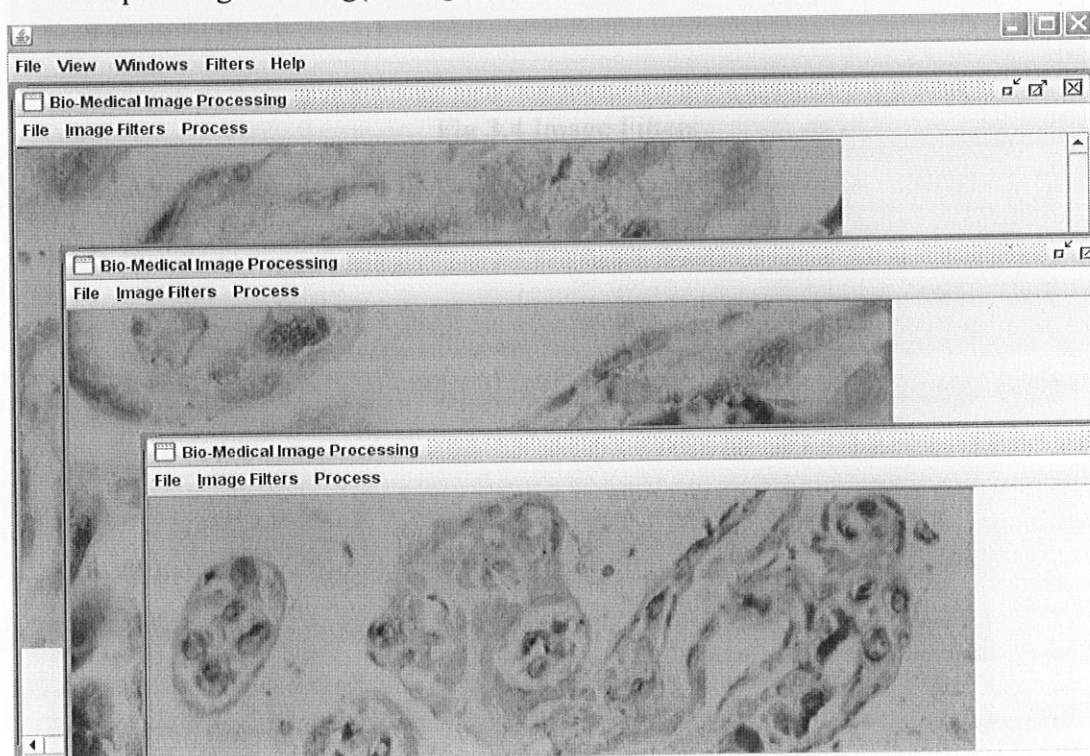
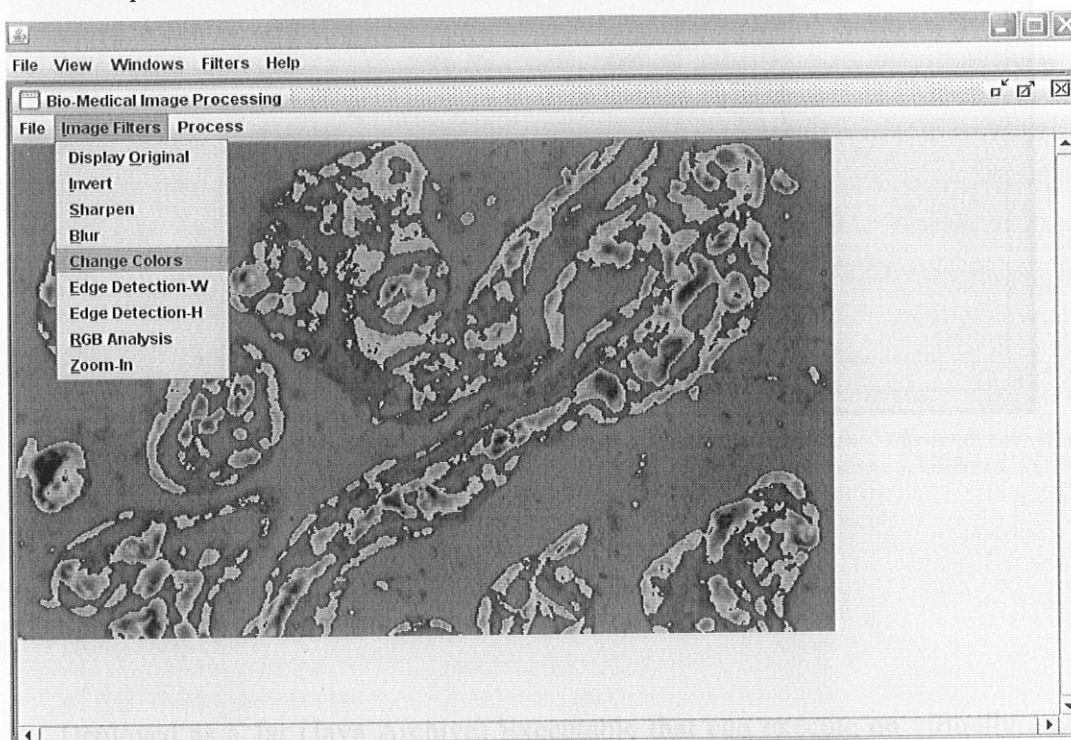


Fig 3.3 Multiple Image Loading

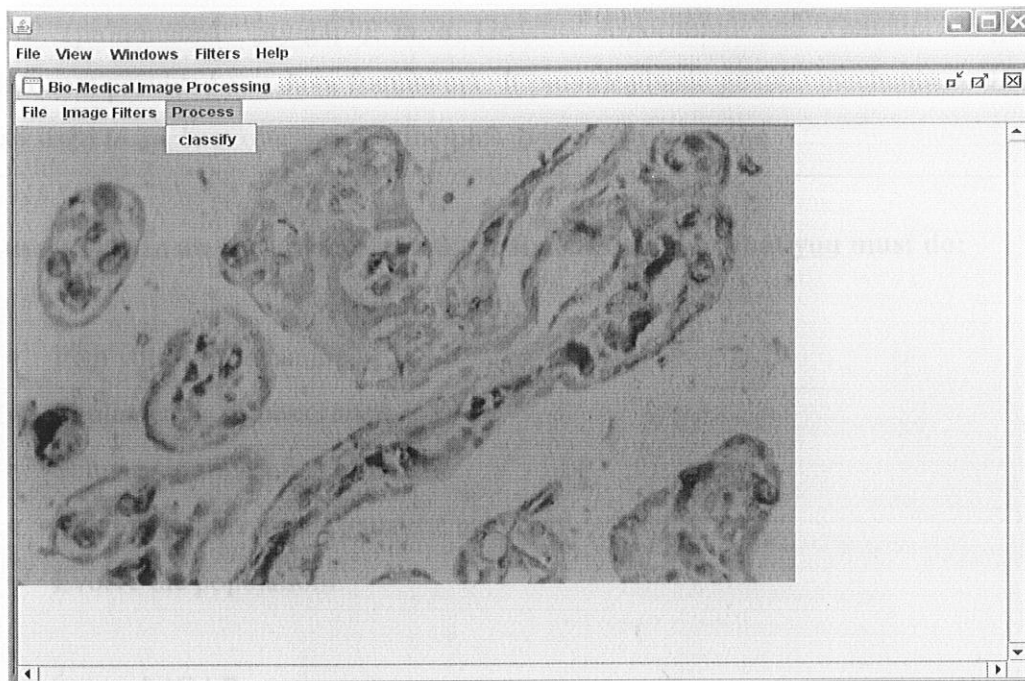
- Drag Mouse RGB analysis
- Save Modified Images to create new Training Set
- User specified threshold for searching Brown Pixels(see Fig 3.4)



**Fig 3.4 Image Filters**

- Drag Mouse to Implement IRA Algorithm





**Fig 3.5 Specify Your Own Training Set**

- Specify Your Own Training Set(see Fig3.5)
- Deployed as a Jar (Java Archive) Executable that can execute on virtually any OS like Linux, UNIX, Windows and Solaris.
- Printing Options to print a modified Image(see Fig 6.8)
- Image Filters like blur, sharpen edge detection, Invert , Change Colors(see fig 6.9)
- Image Rescaling like zooming in and zooming out and rotating images(see fig 6.10)

### 3.4 Inbuilt Libraries

- **JGAP (Java Genetic Algorithms Package)**
- **NanoXML (the smallest XML parser)**

#### 3.4.1 JGAP (Java Genetic Algorithms Package)

JGAP (pronounced "jay-gap") is a Genetic Algorithms and Genetic Programming component provided as a Java framework. It provides basic genetic mechanisms that can be easily used to apply evolutionary principles to problem solutions.

**To use JGAP in an application, there are five basic things that you must do:**

- Plan your Chromosome.
- Implement a "fitness function".
- Setup a Configuration object.
- Create a population of potential solutions.
- Evolve the population!

#### **Advantages of JGAP:**

- JGAP provides a Bulk fitness-function that can be used to create a fitness function that can evaluate all of the Chromosomes in a population at once.
- JGAP support real-valued fitness values or alleles

#### **Disadvantages of JGAP**

- JGAP does not currently offer any support for multi-threaded or distributed evaluation
- Natural selection in JGAP is statistical, so chromosomes that are more fit have a better statistical chance of being selected over chromosomes that are less fit, but it's not guaranteed. This is much like nature, where even the fittest of us can sometimes be unlucky!

### **3.4.2 NanoXML (the smallest XML parser)**

NanoXML (Lite) is a small XML parser for Java. This 6KB XML parser which is the successor of NanoXML 1. It only provides a limited functionality: no mixed content and the DTD is ignored.

Its core classes include

**XMLElement:** XMLElement is a representation of an XML object. The object is able to parse XML code.

**XMLParseException:** An XMLParseException is thrown when an error occurs while parsing an XML string. This class handles this exception.

The above classes can be used in following way to parse an XML file.

```
public static int[] parse(String file)
throws Exception {
    InputStream in = new FileInputStream(file);
    String xml = "";
    while (in.available () > 0) {
        byte [] b = new byte [in.available ()];
        in.read(b);
        xml = xml + new String(b); }
    in.close();
    XMLElement foo = new XMLElement();
    foo.parseString(xml, 0);
    Enumeration enum1 = foo.enumerateChildren();
    int i[] = new int[foo.countChildren()];
    int a = 0;
    while (enum1.hasMoreElements()) {
        XMLElement bar = (XMLElement)(enum1.nextElement());
        i[a]=Integer.parseInt(bar.getContents());
        a++;
        System.out.println(bar.getContents());
    }
    for(int k=0 ;k<foo.countChildren();k++)
    {
        System.out.println(i[k]);
    }
    return i; }
```



### 3.5 Performance Measure

The prediction results of our tool developed in the study were evaluated using the following statistical measures.

**1. Accuracy of the methods:** The accuracy of prediction for the tool were calculated as follows:

$$Q_{ACC} = \frac{P + N}{T}, \text{ where } T = (P + N + O + U)$$

Where  $P$  and  $N$  refer to correctly classified images in class I and II respectively, and  $O$  and  $U$  refer to over and under predictions, respectively.

**2. The Matthews correlation coefficient (MCC) is defined as:**

$$MCC = \frac{(P \times N) - (O \times U)}{\sqrt{(P + U) \times (P + O) \times (N + U) \times (N + O)}}$$

**3. Sensitivity ( $Q_{sens}$ ) and specificity ( $Q_{spec}$ ) of the prediction methods are defined as:**

$$Q_{sens} = \frac{P}{P + U}$$

$$Q_{spec} = \frac{N}{N + O}$$

**4.  $Q_{Pred}$  (Probability of correct prediction) and  $Q_{obs}$  (Percentage over coverage) are defined as:**

$$Q_{pred} = \frac{P}{P + O} \times 100$$

$$Q_{obs} = \frac{P}{P + U} \times 100$$

### 3.6 RESULTS AND DISCUSSION

#### 3.1. Predictability of images with image derived features

Both the algorithms such as GA and NN were trained with the image derived feature like RGB color intonations which are reflected in the images. Based on these three inputs the tool is able to generate range of RGB using GA which in turn used in the second layer for prediction of rate of expression of proteins with respect to the intensity of brown colour using Nearest Neighbour algorithm. All the 3 features calculated separately for both endotheln and CD 31 belonging to active and passive smokers were mentioned in table 2.1. As it is seen that all the three parameters are distinctly different for both active and passive smokers, these paramertes are sufficient to classify both types if images. The predictability and the classification of the images was found to be very accurate (91 %). The prediction results are presented in Table 2.3. The tool has achieved an MCC of **0.9106**. The other performance measures are: **Qpred = 91.0990 %**, **sensitivity = 0.9106** and **specificity = 0.8216**. The rate of expression of both the proteins for active and passive smokers is shown in table 2.4. The rate of expression was more for active smoker (048 and 1071) in comparison to the passive smoker (461 and 294). It is revealed a wide difference in the range of intensity between both the classes of images and hence it is possible to classify the user given image into its corresponding class.

Results are seen to be as follows :

Average Specificity	- 0.8216
Average MCC	- 0.9106
Average Sensitivity	- 0.9106
Average Accuracy	- 91 %
Average Q <sub>(predicted)</sub>	- 91.0990 %
Average Q <sub>(observed)</sub>	- 91.0662

Table 3.1. Performance measure of the tool developed for prediction of rate of expression of proteins and their classification from the placental images.

	Endothelin Active Smoker	Endothelin Passive Smoker	CD 31 Active Smoker	CD 31 Passive Smoker
Specificity	0.8170	0.8170	0.8262	0.8262
MCC	0.9122	0.9069	0.9433	0.88
Sensitivity	0.9069	0.9122	0.88	0.9433
$Q_{(total)}$ /Accuracy	0.91	0.91	0.9126	0.9126
$Q_{(predicted)}$	88.6363	92.8571	93.6170	89.2857
$Q_{(observerd)}$	90.6976	91.2280	88	94.3396

Table 3.2 RGB ranging among images of Endothelin protein

#### Endothelin

Colour Tone	Active			Passive		
	Max	Min	Avg	Max	Min	Avg
RED	134	24	79	153	47	100
GREEN	122	22	72	162	37	100
BLUE	70	36	53	135	36	85

Table 3.3 RGB ranging among images of CD31

#### CD 31

Colour Tone	Active			Passive		
	Max	Min	Avg	Max	Min	Avg
RED	138	17	77	144	38	91
GREEN	143	17	75	150	36	93
BLUE	121	19	70	124	50	87



Table 3.4 Result of classification of various test images

No of Images provided	No of Active images (classified)	No of Passive Images (classified)	Intensity of protein in Active Smoker			Intensity of protein in Passive Smoker		
	Right/Wrong	Right/Wrong	Max	Min	Avg	Max	Min	Avg
<b>Endothelin</b>								
<b>100</b>	<b>39 / 4</b>	<b>52 / 5</b>	<b>1554</b>	<b>543</b>	<b>1048</b>	<b>602</b>	<b>321</b>	<b>461</b>
<b>CD 31</b>								
<b>100</b>	<b>44 / 6</b>	<b>50 / 3</b>	<b>1524</b>	<b>618</b>	<b>1071</b>	<b>476</b>	<b>113</b>	<b>294</b>

The results demonstrate that the developed GA-NN based model for prediction of rate of expression of endothelin and CD31 protein as well as for binary classification of images of active and passive smoker is adequate and can be considered an effective tool for study of placenta abnormalities among tobacco smoking women. The results also demonstrate that the images derived parameters (RGB) can readily be accessible from the digital images only and produce a variety of useful information to be used in study of placenta abnormalities; clearly demonstrates an adequacy and good predictive power of the developed GA-NN model.

Presumably, accuracy of the approach operating by the image derived features can be improved even further by expanding the parameters or by applying more powerful classification techniques such as Support Vector Machines or Bayesian Neural Networks. Use of merely statistical techniques in conjunction with the image extracted parameters would also be beneficial, as they will allow interpreting individual parameter contributions into "placenta abnormalities image likeliness".

The results of the present work demonstrate that the image derived features with GA-NN model appear to be a very fast image identification mechanism providing good results, comparable to some of the current efforts in the literature.

### 3.7 CONCLUSION

From a practical point of view the most important aspect of a prediction method is its ability to make correct predictions. Till date there is no automated tool available for the study of placenta abnormalities from the immunohistochemistry derived images. This is a very tedious job and requires much costlier endeavors. The statistically derived features from image are important determinant for the detailed identification of placental abnormalities from the images. Therefore, a much accurate and reliable method is that which predicts the rate of expression of endothelin and CD31 proteins from the placental images and their categories into active and passive smoker images based on the above mentioned strategy.

Hong-Ching Lu, Shian-Teng Liang.

- Using "biological" genetic algorithms to solve the travelling salesman problem with applications in medical image processing; Pindhorst, G., Telford, H.
- A Local Registration Approach of Medical Images with Niche Genetic Algorithm; Wen-Peng; Rongfang Peng; Guoping Qian; Huiyang Dong.

#### Web Resources

- [www.cmu.edu](http://www.cmu.edu)
- Wikipedia
- IEEE
- Java tutorials
- Java streamIO

#### APPENDIX A : Source Code

##### TrainingSet.java

import java.awt.\*;

import java.awt.image.\*;

## BIBLIOGRAPHY

- **Strategies for optimizing image processing by genetic and evolutionary computation** Shimodaira, H.; Tools with Artificial Intelligence, 2000. ICTAI 2000. Proceedings. 12th IEEE International Conference **Using genetic algorithms in the design of morphological filters** Harvey, N.R.; Marshall, S.;
- **Genetic selection of features for clustering and classification** Smith, J.E.; Fogarty, T.C.; Johnson, I.R.;
- **Recent developments in evolutionary and genetic algorithms: theory and applications** Chaiyaratana, N.; Zalzala, A.M.S.;
- **Fuzzy filters design on image processing by genetic algorithm approach** Hung-Ching Lu; Shian-Tang Tzeng;
- **Using “biological” genetic algorithms to solve the travelling salesman problem with applications in medical image processing** Faulkner, G.; Talhami, H.;
- **A Local Registration Approach of Medical Images with Niche Genetic Algorithm** Wen Peng; Ruofeng Tong; Guiping Qian; Jinxiang Dong;

## Web Resources

- [www.Google.com](http://www.Google.com)
- Wikipedia
- IEEE
- Java tutorials
- Java almanac

## APPENDIX A : Source Code

### TrainingSet.java

```
import java.io.*;
import java.awt.image.*;
```



```

import java.awt.*;
import javax.swing.*;
import java.io.*;

class TrainingSet extends JFrame
{
    Container c = this.getContentPane();

    public static JTextArea ja = new JTextArea();

    // ja.setForeground(new Color(Color.GREEN));
    JScrollPane jp = new JScrollPane(ja);

    int intensity=0;
    FileOutputStream fout;
    BufferedOutputStream bout;
    OutputStreamWriter out;
    public static int TEST_INTENSITY=0;
    public TrainingSet(String directory)
    {
        c.add(jp);
        this.setSize(400,400);
        this.setVisible(true);
    }

    try
    {
        fout= new FileOutputStream(directory+".xml");

        bout= new BufferedOutputStream(fout);

        out = new OutputStreamWriter(bout, "8859_1");

        out.write("<?xml version='1.0' ");

        out.write("encoding='ISO-8859-1'>>\r\n");
        out.write("<"+directory+"> ");

    }
    catch(Exception e){
        System.out.println("Hello6");
        e.printStackTrace();
    }
    String children[] = getPFiles(directory);
    ImageFrame2 if2 = new ImageFrame2();
    for(int i=0;i<children.length;i++)

```

```

        {
            System.out.println("Image Files: "+ children[i]);

            intensity = if2.getIntensity(directory+"/"+children[i]);
            try{

                out.write("<Intensity"+i+">"+intensity+"</Intensity"+i+">\r\n");
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }

        }
        try
        {
            out.write("</"+directory+"> ");
            out.flush(); // Don't forget to flush!
            out.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
        this.setVisible(false);
    }

    public TrainingSet(String directory,int ix)
    {

        c.add(jp);
        this.setSize(400,400);
        this.setVisible(true);
        ImageFrame2 if2 = new ImageFrame2();
        intensity = if2.getIntensity(directory);
        TrainingSet.TEST_INTENSITY = intensity;

        JOptionPane.showMessageDialog(null,"Intensity of Your Image is
        "+intensity,"Intensity",JOptionPane.INFORMATION_MESSAGE) ;
        try
        {

            Classifier classify = new Classifier(intensity);
        }
        catch(Exception e)

        {

            e.printStackTrace();
        }
    }

```

```
this.setVisible(false);
}
```

```
public static String[] getPFiles(String directory)
{
```

```
    File dir = new File(directory);
```

```
    String[] children = dir.list();
```

```
    if (children == null) {
```

```
        // Either dir does not exist or is not a directory
```

```
    } else {
```

```
        for (int i=0; i<children.length; i++) {
```

```
            // Get filename of file or directory
```

```
            String filename = children[i];
```

```
            // System.out.println(filename);
```

```
        }
```

```
    }
```

```
    FilenameFilter filter = new FilenameFilter() {
```

```
        public boolean accept(File dir, String name) {
```

```
            return name.endsWith(".jpg");
```

```
        }
```

```
    };
```

```
    children = dir.list(filter);
```

```
    return children;
```

```
}
```

```
public static String[] getAFiles(String directory)
{
```

```
    File dir = new File(directory);
```

```
    String[] children = dir.list();
```

```
    if (children == null) {
```

```
        // Either dir does not exist or is not a directory
```

```
    } else {
```

```
        for (int i=0; i<children.length; i++) {
```

```
            // Get filename of file or directory
```

```
            String filename = children[i];
```

```
            // System.out.println(filename);
```

```
        }
```

```
    }
```

```
    FilenameFilter filter = new FilenameFilter() {
```



```

        public boolean accept(File dir, String name) {
            return name.endsWith(".jpg");
        }
    };
    children = dir.list(filter);

    return children;
}

public static void main(String args[])
{
    TrainingSet ts = new TrainingSet("C:\\Documents and
    Settings\\Administrator\\Desktop\\abhilshit\\CancerPred\\CancerPred\\classes\\cd31_active")
    ;
    TrainingSet ts1 = new TrainingSet("C:\\Documents and
    Settings\\Administrator\\Desktop\\abhilshit\\CancerPred\\CancerPred\\classes\\ET_active");
    TrainingSet ts2 = new TrainingSet("C:\\Documents and
    Settings\\Administrator\\Desktop\\abhilshit\\CancerPred\\CancerPred\\classes\\ET_passive")
    ;
    TrainingSet ts3 = new TrainingSet("C:\\Documents and
    Settings\\Administrator\\Desktop\\abhilshit\\CancerPred\\CancerPred\\classes\\cd31_passive
    ");
}
}

```

CancerPred.java

```

import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.net.URL;
import javax.swing.*;
import java.io.File;
import javax.swing.JScrollBar;
import javax.swing.border.EtchedBorder;

public class CancerPred extends JFrame {
    private JMenu filterMenu = new JMenu("Image Filters");
    JMenu process = new JMenu("Process");
    JMenu file = new JMenu("File");
    private ImagePanel imagePanel;
    JScrollBar js = new JScrollBar();
    private MyFilter invertFilter = new InvertFilter();
}

```

```

private MyFilter sharpenFilter = new SharpenFilter();

private MyFilter blurFilter = new BlurFilter();

private MyFilter colorFilter = new ColorFilter();
private MyFilter edgeFilter = new EdgeFilter();
private MyFilter edgeFilter2 = new EdgeFilter2();
private MyFilter zoomin= new ZoomIn();
URL uri;
public CancerPred() {
    super("Bio-Medical Image Processing",true,true,true,true);

    //getContentPane().add(js, null);
    try
    {
        File f = new File(MainForm.filepath);
        System.out.println("File "+f.toString());
        uri = f.toURL();
        System.out.println("URL "+uri.toString());

        imagePanel = new ImagePanel(uri);
    }
    catch(Exception e)
    {
        System.out.println("Malformed URL Exception "+e);
    }

    JMenuBar menuBar = new JMenuBar();
    setJMenuBar(menuBar);
    filterMenu.setMnemonic('T');

    JMenuItem originalMenuItem = new JMenuItem("Display Original");
    originalMenuItem.setMnemonic('O');

    originalMenuItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent action) {
            imagePanel.displayOriginalImage();
        }
    });

    JMenuItem classify = new JMenuItem("classify");
    JMenuItem save = new JMenuItem("Save");
    JMenuItem invertMenuItem = createMenuItem("Invert", 'I', invertFilter);
    JMenuItem sharpenMenuItem = createMenuItem("Sharpen", 'S', sharpenFilter);
    JMenuItem blurMenuItem = createMenuItem("Blur", 'B', blurFilter);
    JMenuItem changeColorsMenuItem = createMenuItem("Change Colors", 'C', colorFilter);
    JMenuItem edgeMenuItem = createMenuItem("Edge Detection-W", 'E', edgeFilter);
    JMenuItem edgeMenuItem2 = createMenuItem("Edge Detection-H", 'F', edgeFilter2);
    JMenuItem zoominItem = createMenuItem("Zoom-In", 'Z', zoomin);

```

```

JMenuItem RGB = createMenuItem2("RGB Analysis", 'R');
filterMenu.add(originalMenuItem);
filterMenu.add(invertMenuItem);
filterMenu.add(sharpenMenuItem);
filterMenu.add(blurMenuItem);
filterMenu.add(changeColorsMenuItem);
filterMenu.add(edgeMenuItem);
filterMenu.add(edgeMenuItem2);
filterMenu.add(RGB);
filterMenu.add(zoominItem);
process.add(classify);
file.add(save);
save.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent action) {
        try
        {
            imagePanel.saveImage();
        }
        catch(Exception e){
        }
    }
});

classify.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent action) {
        try
        {
            TrainingFrame tf = new TrainingFrame();
        }
        catch(Exception e){
        }
    }
});
menuBar.add(file);
menuBar.add(filterMenu);
menuBar.add(process);

JScrollPane jsp = new
JScrollPane(imagePanel,ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,Scrol
lPaneConstants.HORIZONTAL_SCROLLBAR_ALWAYS);
getContentPane().add(jsp);
// ;
// getContentPane().add(js);

```



```

        //getContentPane().add(imagePanel,"Center");
    }

    public JMenuItem createMenuItem(String menuItemName, char mnemonic, final MyFilter
filter) {
        JMenuItem menuItem = new JMenuItem(menuItemName);
        menuItem.setMnemonic(mnemonic);
        menuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent action) {
                imagePanel.applyFilter(filter);
            }
        });
        return menuItem;
    }

    public JMenuItem createMenuItem2(String menuItemName, char mnemonic) {
        JMenuItem menuItem = new JMenuItem(menuItemName);
        menuItem.setMnemonic(mnemonic);
        menuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent action) {
                ImageFrame f = new ImageFrame();
            }
        });
        return menuItem;
    }

    static void renderSplashFrame(Graphics2D g, int frame) {
        final String[] comps = {"CancerPredor", "Image Utilities & Toolbars", "File Menus"};
        g.setComposite(AlphaComposite.Clear);
        g.fillRect(130,250,280,40);
        g.setPaintMode();
        g.setColor(Color.BLACK);
        g.drawString("Loading "+comps[(frame/5)%3]+"...", 130, 260);
        g.fillRect(130,270,(frame*10)%280,20);
    }

    /* public static void main(String args[]) {
        /* final SplashScreen splash = SplashScreen.getSplashScreen();
        if (splash == null) {
            System.out.println("SplashScreen.getSplashScreen() returned null");
            return;
        }
        Graphics2D g = (Graphics2D)splash.createGraphics();

```

```
        if (g == null) {
            System.out.println("g is null");
            return;
        }
        for(int i=0; i<100; i++) {
            renderSplashFrame(g, i);
            splash.update();
            try {
                Thread.sleep(60);
            }
            catch(InterruptedException e) {
            }
        }
        splash.close();

    }*/
}
```