

LIVER TUMOR CLASSIFICATION AND ANALYSIS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology degree in
Biomedical Engineering

By

PERIYANAYAGAM ROZARIO.B(39240028)

DILIP.E.K(39240015)

TAMILMARAN.K(39240039)



**DEPARTMENT OF BIOMEDICAL ENGINEERING
SCHOOL OF BIO AND CHEMICAL ENGINEERING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MARCH 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

www.sathyabama.ac.in



DEPARTMENT OF BIOMEDICAL ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Interdisciplinary Project is the bonafide work of **PERIYANAYAGAM ROZARIO.B (Reg. No.39240028), DILIP.E.K (Reg. No. 39240015) TAMILMARAN.K (Reg. No.39240039)** who carried out the Interdisciplinary project entitled "**LIVER TUMOR CLASSIFICATION ANALYSIS**" under our supervision from **December 2021 to April 2022**.

**Dr.T.SUDHAKAR, M.Sc., Ph.D.,
Guide and Supervisor**

**Dr. T.SUDHAKAR, M.Sc., Ph.D.,
Head of the Department**

Submitted for Viva voce Examination held on 30th april 2022.

Internal Examiner

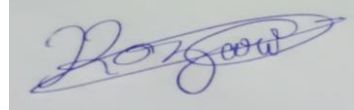
External Examiner

DECLARATION

We, **PERIYANAYAGAM ROZARIO.B**(Reg. No. 39240028), **DILIP.E.K** (Reg. No. 39240015), **TAMILMARAN.K** (Reg. No.39240039) hereby declare that the Interdisciplinary Project entitled “**LIVER TUMOR CLASSIFICATION AND ANALYSIS**” done by us under the guidance of **Dr.T.SUDHAKAR, Department of Biomedical Engineering** is submitted in partial fulfillment of the requirements for the award of **Bachelor of Technology** degree in **Biomedical Engineering**.

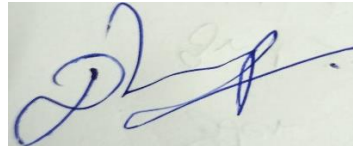
DATE: 28/4/2022

1.

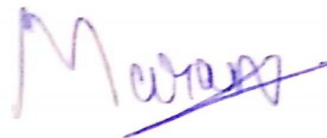


PLACE: Chennai

2.



3.



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T. Sudhakar, M.Sc., Ph.D.**, Head of the Department, **Department of Biomedical Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr.T.SUDHAKAR, M.Sc., Ph.D., Department of Biomedical Engineering** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all **Teaching and Non-teaching** staff members of the **Department of Biomedical Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT:

The liver cancer rate is increasing year by year. Generally, liver cancer's death rate is very high because the disease causes no symptoms, so it's often not caught until it's in final stages. We propose methods for liver cancer detection which is based on concepts of neural network. Wavelet and glcm features extracted from input image. This glcm features classify to dataset and then get result is normal or abnormal stages. Many techniques have been developed for the detection of liver tumour utilizing the aberrant lesion size and shape. The novel methodology for the detection and diagnosis of liver tumour is additionally proposed in this paper for the detection and diagnosis of liver tumour.

TABLE OF CONTENTS

CHAPT ER NO	TITLE	PAGE NO
	ABSTRACT	I
	LIST OF ABBREVIATION	iv
	LIST OF FIGURES	V
1	INTRODUCTION	1
	1.1 Overview	1
	1.2 Scope of the Project	1
	1.3 Domain Overview	2
2	LITERATURE SURVEY	3
3	AIM AND SCOPE	9
	3.1 AIM	9
	3.2 SCOPE	9
	• System Analysis	9
	• Principal Component Analysis	9
	• DCT & shape features	9
	• KNN Classifier	9
	• Drawbacks of Exisitng System	9
	• Proposed Method	9
	• Advantages	9

4	MATERIALS AND METHODS	10
	• HARDWARE AND SOFTWARE USED	10
	• MATLAB SYSTEM	10
	• MANUPULATING MATRIXES	11
	• ALGORITHM(NEURAL NETWORKING)	11
	• STEPS OF ALGORITHM	11
5	RESULTS AND DISCUSSION	22
	5.1 RESULTS	22
	5.2 DISCUSSION	25
6	SUMMARY AND CONCLUSION	26
	6.1 SUMMARY	26
	6.2 CONCLUSION	26
	REFERENCES	27

CHAPTER-1

INTRODUCTION

Early diagnosis is vital in treating a disease. Diagnosis relies on the skills, experience, and knowledge of the practicing physicians, although human errors may sometimes occur. Recently, various artificial intelligence-based methods are being increasingly used for liver disorder diagnosis to assist doctors in the diagnosis of patients. Extended cirrhosis and statuses liver disease may cause the appearance of malignant or benign tumors in the liver. Any abnormality in the liver adversely affects the rest of the body, as well as the general health of the patient Fatty liver disease (FLD) occurs when the human body produces a considerable amount of fat or does not efficiently metabolize fat. This excess fat is stored in the liver cells, where it accumulates triglycerides in the blood, causing FLD It is believed that FLD involves the pathogenesis of various common disorders, such as Type II diabetes and cardiovascular diseases. The liver cancer rate is increasing year by year. Generally, liver cancer's death rate is very high because the disease causes no symptoms, so it's often not caught until it's in final stages. We propose methods for liver cancer detection which is based on concepts of neural network. Wavelet and glcm features extracted from input image. This glcm features classify to dataset and then get result is normal or abnormal stages

1.1 Overview

Aim of the project is to create an artificial intelligent model which classifies liver cancer is affected or not using CT/MRI images with deep learning technique.

1.2 Scope of the Project

The main contributions of this project therefore are

- Data Analysis
- Dataset Preprocessing
- Training the Model
- Testing of Dataset

1.3 Domain Overview

Deep neural networks are now the state-of-the-art machine learning models across

a variety of areas, from image analysis to natural language processing, and widely deployed in academia and industry.

These developments have a huge potential for medical imaging technology, medical data analysis, medical diagnostics and healthcare in general, slowly being realized. We provide a short overview of recent advances and some associated challenges in machine learning applied to medical image processing and image analysis.

Long before deep learning was used, traditional machine learning methods were mainly used. Such as Decision Trees, SVM, Naïve Bayes Classifier and Logistic Regression.

These algorithms are also called flat algorithms. Flat here means that these algorithms can not normally be applied directly to the raw data (such as .csv, images, text, etc.). We need a preprocessing step called Feature Extraction.

The result of Feature Extraction is a representation of the given raw data that can now be used by these classic machine learning algorithms to perform a task. For example, the classification of the data into several categories or classes.

Feature Extraction is usually quite complex and requires detailed knowledge of the problem domain. This preprocessing layer must be adapted, tested and refined over several iterations for optimal results.

On the other side are the artificial neural networks of Deep Learning. These do not need the Feature Extraction step.

The layers are able to learn an implicit representation of the raw data directly and on their own. Here, a more and more abstract and compressed representation of the raw data is produced over several layers of an artificial neural-nets. This compressed representation of the input data is then used to produce the result. The result can be, for example, the classification of the input data into different classes.

CHAPTER-2

LITERATURE SURVEY

s.no	Author name	Title	Journal name	Year
1	Clinton j.wang,Charlie a.hamm,lynn j.savic	Deep learning for liver tumor diagnosis part2:convoluti onal neural network interpretation using radiologic imaging features	European journal of radiology	2019
2	Author:Huseyin kutlu and Engin avci	Classification and mutation prediction based on histopatholog y H&E image in liver cancer using deep learning	Precision oncology	2020
3	Huseyin kutlu and engin avci	A novel method for classifying liver and brain tumor using convolution neural network,discre te wavelet transform and long short- term memory network	Sensor	2019
4	Pierre-maxence vaysse,heike i. grabsch, mari F.C.M. van den hout	Real-time lipid pattern to classify viable and necrotic liver tumors	Usa cap	2020
5	A.keerthana,G.sri vidhya,R.J.hemal atha,jaya rubi	Classification of liver tumor on ct image using machine learning	Blue eyes intelligence engineering &sciences publication	2019

CHAPTER-3

AIM AND SCOPE

3.1 AIM:

- **The liver cancer rate is increasing year by year. Generally, liver cancer's death rate is very high because the disease causes no symptoms, so it's often not caught until it's in final stages.**
- **We propose methods for liver cancer detection which is based on concepts of neural network. Wavelet and glcm features extracted from input image.**
- **This glcm features classify to dataset and then get result is normal or abnormal stages. Many techniques have been developed for the detection of liver tumor utilizing the aberrant lesion size and shape.**
- **The novel methodology for the detection and diagnosis of liver tumor is additionally proposed in this paper for the detection and diagnosis of liver tumor.**

3.2 SCOPE:

System Analysis

Existing Systems

- **Principal Component Analysis**
- **DCT and shape features**
- **KNN and SVM classifier**

Principal Component Analysis

The principal component analysis is based on the fact that neighboring bands of hyperspectral images are highly correlated and often convey almost the same information about the object. The analysis is used to transform the original data so to remove the correlation among the bands.

Abstract Principal component analysis (PCA) is one of the statistical techniques frequently used in signal processing to the data dimension reduction or to the data decorrelation.

DCT & shape features

The DCT is the most widely used transformation technique in signal processing, and by far the most widely used linear transform in data compression. DCT data compression has been fundamental to the Digital Revolution.

Uncompressed digital media as well as lossless compression had impractically high memory and bandwidth requirements, which was significantly reduced by the highly efficient DCT lossy compression technique, capable of achieving data compression ratios from 8:1 to 14:1 for near-studio-quality, up to 100:1 for acceptable-quality content. The wide adoption of DCT compression standards led to the emergence and proliferation of digital media technologies, such as digital images, digital photos, digital video, streaming media, digital television, streaming television, video-on-demand (VOD), digital cinema, high-definition video (HD video), and high-definition television

KNN Classifier

In pattern recognition, the k-nearest neighbour algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space. k-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The k-

nearest neighbour algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbours, with the object being assigned to the class most common amongst its k nearest neighbours (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of its nearest neighbour.

The same method can be used for regression, by simply assigning the property value for the object to be the average of the values of its nearest neighbours. It can be useful to weight the contributions of the neighbours, so that the nearer neighbours contribute more to the average than the more distant ones. (A common weighting scheme is to give each neighbour a weight of $1/d$, where d is the distance to the neighbour. This scheme is a generalization of linear interpolation.)

KNN Classifier

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.

Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback.

This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approaches.

Drawbacks of Existing System

- High Computational load and poor discriminatory power.
- SVM is slow training for large feature set.
- Less accuracy in classification

Proposed Method

- DTCWT and GLCM Features
- NN Classifier
- K-means Clustering

Advantages

- The segmentation algorithm Proves to be simple and effective
- Grey scale Co-occurrence matrix performed well in NN
- Better texture and edge representation
- Segmentation provides better clustering efficiency

CHAPTER-4

MATERIALS AND METHODS

4.1 BLOCK DIAGRAM OF LIVER TUMOR

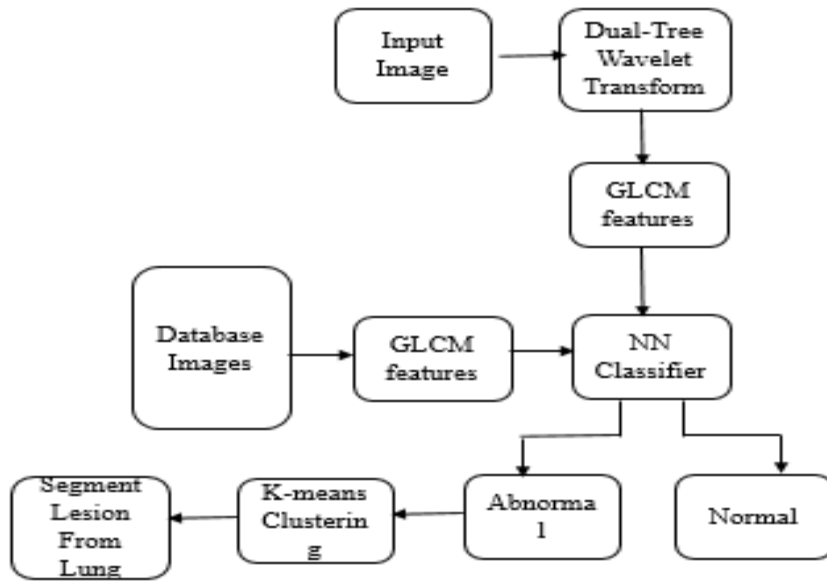


Fig: 4.1: BLOCK DIAGRAM WITH EXPLANATION

Image restoration is the operation of taking a corrupted/noisy image and estimating the clean original image. Corruption may come in many forms such as motion blur, noise, and camera misfocus. Image restoration is different from image enhancement in that the latter is designed to emphasize features of the image that make the image more pleasing to the observer, but not necessarily to produce realistic data from a scientific point of view. Image enhancement techniques (like contrast stretching or de-blurring by a nearest neighbour procedure) provided by "Imaging packages" use no a priori model of the process that created the image. With image enhancement noise can be effectively be removed by sacrificing some resolution, but this is not acceptable in many applications. In a Fluorescence Microscope resolution in the z-direction is bad as it is. More advanced image processing techniques must be applied to recover the object. De-Convolution is an example of image restoration method. It is capable of: Increasing resolution, especially in the axial direction removing noise increasing contrast.

4.2 HARDWARE AND SOFTWARE USED

Hardware Requirements

- system
- 4 GB of RAM
- 500 GB of Hard disk

Software Requirement

- MATLAB 2014a

4.3 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or FORTRAN.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB uses software developed by the LAPACK and ARPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In

industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

The MATLAB System

The MATLAB system consists of five main parts:

Development Environment. This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

The MATLAB Mathematical Function Library. This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

The MATLAB Language. This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

Handle Graphics®. This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

The MATLAB Application Program Interface (API). This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

DEVELOPMENT ENVIRONMENT

Introduction

This chapter provides a brief introduction to starting and quitting MATLAB, and the tools and functions that help you to work with MATLAB variables and files. For more information about the topics covered here, see the corresponding topics under Development Environment in the MATLAB documentation, which is available online as well as in print.

Starting and Quitting MATLAB

Starting MATLAB

On a Microsoft Windows platform, to start MATLAB, double-click the MATLAB shortcut icon on your Windows desktop.

On a UNIX platform, to start MATLAB, type matlab at the operating system prompt.

After starting MATLAB, the MATLAB desktop opens - see MATLAB Desktop.

You can change the directory in which MATLAB starts, define startup options including running a script upon startup, and reduce startup time in some situations.

Quitting MATLAB

To end your MATLAB session, select Exit MATLAB from the File menu in the desktop, or type quit in the Command Window. To execute specified functions each time MATLAB quits, such as saving the workspace, you can create and run a finish.m script.

MATLAB Desktop

When you start MATLAB, the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB.

The first time MATLAB starts, the desktop appears as shown in the following illustration, although your Launch Pad may contain different entries.

You can change the way your desktop looks by opening, closing, moving, and resizing the tools in it. You can also move tools outside of the desktop or return them back inside the desktop (docking). All the desktop tools provide common features such as context menus and keyboard shortcuts.

You can specify certain characteristics for the desktop tools by selecting Preferences from the File menu. For example, you can specify the font characteristics for Command Window text. For more information, click the Help button in the Preferences dialog box.

Desktop Tools

This section provides an introduction to MATLAB's desktop tools. You can also use MATLAB functions to perform most of the features found in the desktop tools. The tools are:

Command Window

Use the Command Window to enter variables and run functions and M-files.

Command History

Lines you enter in the Command Window are logged in the Command History window. In the Command History, you can view previously used functions, and copy and execute selected lines. To save the input and output from a MATLAB session to a file, use the diary function.

Running External Programs

You can run external programs from the MATLAB Command Window. The exclamation point character! is a shell escape and indicates that the rest of the input line is a command to the operating system. This is useful for invoking utilities or running other programs without quitting MATLAB. On Linux, for example,!emacs magik.m invokes an editor called emacs for a file named magik.m. When you quit the external program, the operating system returns control to MATLAB.

Launch Pad

MATLAB's Launch Pad provides easy access to tools, demos, and documentation.

Help Browser

Use the Help browser to search and view documentation for all your Math Works products. The Help browser is a Web browser integrated into the MATLAB desktop that displays HTML documents.

To open the Help browser, click the help button in the toolbar, or type help browser in the Command Window. The Help browser consists of two panes, the Help Navigator, which you use to find information, and the display pane, where you view the information.

Help Navigator

Use the Help Navigator to find information. It includes:

Product filter - Set the filter to show documentation only for the products you specify.

Contents tab - View the titles and tables of contents of documentation for your

products.

Index tab - Find specific index entries (selected keywords) in the MathWorks documentation for your products.

Search tab - Look for a specific phrase in the documentation. To get help for a specific function, set the Search type to Function Name.

Favorites tab - View a list of documents you previously designated as favorites.

Display Pane

After finding documentation using the Help Navigator, view it in the display pane. While viewing the documentation, you can:

Browse to other pages - Use the arrows at the tops and bottoms of the pages, or use the back and forward buttons in the toolbar.

Bookmark pages - Click the Add to Favorites button in the toolbar.

Print pages - Click the print button in the toolbar.

Find a term in the page - Type a term in the Find in page field in the toolbar and click Go.

Other features available in the display pane are: copying information, evaluating a selection, and viewing Web pages.

Current Directory Browser

MATLAB file operations use the current directory and the search path as reference points. Any file you want to run must either be in the current directory or on the search path.

Search Path

To determine how to execute functions you call, MATLAB uses a search path to find M-files and other MATLAB-related files, which are organized in directories on your file system. Any file you want to run in MATLAB must reside in the current directory or in a directory that is on the search path. By default, the files supplied with MATLAB and MathWorks toolboxes are included in the search path.

Workspace Browser

The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory. You add variables to the workspace by using functions, running M-files, and loading saved workspaces.

To view the workspace and information about each variable, use the Workspace browser, or use the functions `who` and `whos`.

To delete variables from the workspace, select the variable and select Delete from the Edit menu. Alternatively, use the `clear` function.

Array Editor

Double-click on a variable in the Workspace browser to see it in the

Array Editor. Use the Array Editor to view and edit a visual representation of one- or two-dimensional numeric arrays, strings, and cell arrays of strings that are in the workspace.

Editor/Debugger

Use the Editor/Debugger to create and debug M-files, which are programs you write to run MATLAB functions. The Editor/Debugger provides a graphical user interface for basic text editing, as well as for M-file debugging.

You can use any text editor to create M-files, such as Emacs, and can use preferences (accessible from the desktop File menu) to specify that editor as the default. If you use another editor, you can still use the MATLAB Editor/Debugger for debugging, or you can use debugging functions, such as `dbstop`, which sets a breakpoint.

MANIPULATING MATRICES

Entering Matrices

The best way for you to get started with MATLAB is to learn how to handle matrices. Start MATLAB and follow along with each example.

You can enter matrices into MATLAB in several different ways:

- Enter an explicit list of elements.
- Load matrices from external data files.
- Generate matrices using built-in functions.
- Create matrices with your own functions in M-files.

Start by entering Dürer's matrix as a list of its elements. You have only to follow a few basic conventions:

- Separate the elements of a row with blanks or commas.
- Use a semicolon, `;`, to indicate the end of each row.
- Surround the entire list of elements with square brackets, `[ ]`.

To enter Dürer's matrix, simply type in the Command Window

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

MATLAB displays the matrix you just entered.

A =

```
16   3   2  13
  5  10  11   8
  9   6   7  12
  4  15  14   1
```

This exactly matches the numbers in the engraving. Once you have entered the matrix, it is automatically remembered in the MATLAB workspace. You can refer

to it simply as A.

Expressions

Like most other programming languages, MATLAB provides mathematical expressions, but unlike most programming languages, these expressions involve entire matrices. The building blocks of expressions are:

Variables

MATLAB does not require any type declarations or dimension statements. When MATLAB encounters a new variable name, it automatically creates the variable and allocates the appropriate amount of storage. If the variable already exists, MATLAB changes its contents and, if necessary, allocates new storage. For example,

```
num_students = 25
```

Creates a 1-by-1 matrix named num_students and stores the value 25 in its single element.

Variable names consist of a letter, followed by any number of letters, digits, or underscores. MATLAB uses only the first 31 characters of a variable name. MATLAB is case sensitive; it distinguishes between uppercase and lowercase letters. A and a are not the same variable. To view the matrix assigned to any variable, simply enter the variable name.

Functions

MATLAB provides a large number of standard elementary mathematical functions, including abs, sqrt, exp, and sin. Taking the square root or logarithm of a negative number is not an error; the appropriate complex result is produced automatically. MATLAB also provides many more advanced mathematical functions, including Bessel and gamma functions. Most of these functions accept complex arguments. For a list of the elementary mathematical functions, type

```
helpfun
```

For a list of more advanced mathematical and matrix functions,

Some of the functions, like sqrt and sin, are built-in. They are part of the MATLAB core so they are very efficient, but the computational details are not readily accessible. Other functions, like gamma and sinh, are implemented in M-files. You can see the code and even modify it if you want. Several special functions provide values of useful constants.

GUI

A graphical user interface (GUI) is a user interface built with graphical objects, such as buttons, text fields, sliders, and menus. In general, these objects already have meanings to most computer users. For example, when you

move a slider, value changes; when you press an OK button, your settings are applied and the dialog box is dismissed. Of course, to leverage this built-in familiarity, you must be consistent in how you use the various GUI-building components.

Applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work. The action that results from a particular user action can be made clear by the design of the interface.

The sections that follow describe how to create GUIs with MATLAB. This includes laying out the components, programming them to do specific things in response to user actions, and saving and launching the GUI; in other words, the mechanics of creating GUIs. This documentation does not attempt to cover the "art" of good user interface design, which is an entire field unto itself. Topics covered in this section include:

Creating GUIs with GUIDE

MATLAB implements GUIs as figure windows containing various styles of GUI control objects. You must program each object to perform the intended action when activated by the user of the GUI. In addition, you must be able to save and launch your GUI. All of these tasks are simplified by GUIDE, MATLAB's graphical user interface development environment.

GUI Development Environment

The process of implementing a GUI involves two basic tasks:

- Laying out the GUI components
- Programming the GUI components

GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the call backs - the functions that execute when users activate components in the GUI.

Features of the GUIDE-Generated Application M-File

GUIDE simplifies the creation of GUI applications by automatically generating an M-file framework directly from your layout. You can then use this framework to code your application M-file. This approach provides a number of advantages:

The M-file contains code to implement a number of useful features (see *Configuring Application Options* for information on these features). The M-file adopts an effective approach to managing object handles and executing call back routines (see *Creating and Storing the Object Handle Structure* for more information). The M-files provides a way to manage global data (see *Managing GUI Data* for more information).

Beginning the Implementation Process

To begin implementing your GUI, proceed to the following sections:

Getting Started with GUIDE - the basics of using GUIDE

Selecting GUIDE Application Options - set both FIG-file and M-file options.

Using the Layout Editor - begin laying out the GUI.

Understanding the Application M-File - discussion of programming techniques used in the application M-file.

Application Examples - a collection of examples that illustrate techniques Which are useful for implementing GUIs.

Command-Line Accessibility

When MATLAB creates a graph, the figure and axes are included in the list of children of their respective parents and their handles are available through commands such as find object, set, and get. If you issue another plotting command, the output is directed to the current figure and axes.

GUIs are also created in figure windows. Generally, you do not want GUI figures to be available as targets for graphics output, since issuing a plotting command could direct the output to the GUI figure, resulting in the graph appearing in the middle of the GUI.

In contrast, if you create a GUI that contains axes and you want commands entered in the command window to display in this axes, you should enable command-line access.

User Interface Controls

The Layout Editor Component palette contains the user interface controls that you can use in your GUI. These components are MATLAB uicontrol objects and are programmable via their Call back properties. This section provides information on these components.

Push Buttons

Push buttons generate an action when pressed (e.g., an OK button may close a dialog box and apply settings). When you click down on a push button, it appears depressed; when you release the mouse, the button's appearance returns to its non depressed state; and its call back executes on the button up event.

Properties to Set

String - set this property to the character string you want displayed on the push button.

Tag - GUIDE uses the Tag property to name the call-back sub function in the application M-file. Set Tag to a descriptive name (e.g., close button) before activating the GUI.

Programming the Call-back

When the user clicks on the push button, its call back executes. Push buttons do not return a value or maintain a state

Toggle Buttons

Toggle buttons generate an action and indicate a binary state (e.g., on or off). When you click on a toggle button, it appears depressed and remains depressed when you release the mouse button, at which point the call-back executes. A subsequent mouse click returns the toggle button to the non depressed state and again executes its call back.

Programming the Call back

The call back routine needs to query the toggle button to determine what state it is in. MATLAB sets the Value property equal to the Max property when the toggle button is depressed (Max is 1 by default) and equal to the Min property when the toggle button is not depressed (Min is 0 by default).

Radio Buttons

Radio buttons are similar to checkboxes, but are intended to be mutually exclusive within a group of related radio buttons (i.e., only one button is in a selected state at any given time). To activate a radio button, click the mouse button on the object. The display indicates the state of the button.

Checkboxes

Check boxes generate an action when clicked and indicate their state as checked or not checked. Check boxes are useful when providing the user with a number of independent choices that set a mode (e.g., display a toolbar or generate call back function prototypes).

Edit Text

Edit text controls are fields that enable users to enter or modify text strings. Use edit text when you want text as input. The String property contains the text entered by the user.

Axes

Axes enable your GUI to display graphics (e.g., graphs and images). Like all graphics objects, axes have properties that you can set to control many aspects of its behavior and appearance. See Axes Properties for general information on axes objects.

Figure

Figures are the windows that contain the GUI you design with the Layout Editor. See the description of figure properties for information on what figure characteristics you can control.

Algorithm

Neural networks (NN)

Neural Network segmentation relies on processing small areas of an image using an artificial neural network or a set of neural networks. After such processing the decision-making mechanism marks the areas of an image accordingly to the category recognized by the neural network. A type of network designed especially for this is the Kohonen map.

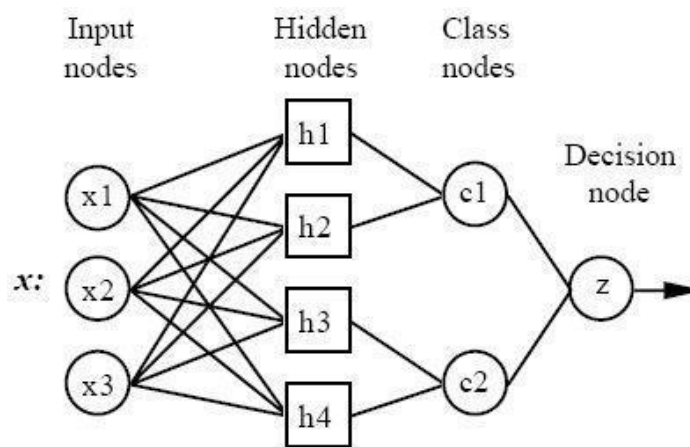
Pulse-Coupled Neural Networks (PCNNs) are neural models proposed by modeling a cat's visual cortex and developed for high-performance biomimetic image processing. In 1989, Eckhorn introduced a neural model to emulate the mechanism of cat's visual cortex. The Eckhorn model provided a simple and effective tool for studying small mammal's visual cortex, and was soon recognized as having significant application potential in image processing. In 1994, the Eckhorn model was adapted to be an image processing algorithm by Johnson, who termed this algorithm Pulse-Coupled Neural Network. Over the past decade, PCNNs have been utilized for a variety of image processing applications, including: image segmentation, feature generation, face extraction, motion detection, region growing, noise reduction, and so on. A PCNN is a two-dimensional neural network. Each neuron in the network corresponds to one pixel in an input image, receiving its corresponding pixel's color information (e.g. intensity) as an external stimulus. Each neuron also connects with its neighboring neurons, receiving local stimuli from them. The external and local stimuli are combined in an internal activation system, which accumulates the stimuli until it exceeds a dynamic threshold, resulting in a pulse output. Through iterative computation, PCNN neurons produce temporal series of pulse outputs. The temporal series of pulse outputs contain information of

input images and can be utilized for various image processing applications, such as image segmentation and feature generation. Compared with conventional image processing means, PCNNs have several significant merits, including robustness against noise, independence of geometric variations in input patterns, capability of bridging minor intensity variations in input patterns, etc.

Convolutional neural networks (CNN):

Convolutional neural network (CNN) and General Regression Neural Networks (GRNN) have similar architectures, but there is a fundamental difference: Probabilistic networks perform classification where the target variable is categorical, whereas general regression neural networks perform regression where the target variable is continuous. If you select a CNN/GRNN network, DTREG will automatically select the correct type of network based on the type of target variable.

Architecture of a CNN:



All CNN networks have four layers:

1. **Input layer** — There is one neuron in the input layer for each predictor variable. In the case of categorical variables, $N-1$ neurons are used where N is

the number of categories. The input neurons (or processing before the input layer) standardizes the range of the values by subtracting the median and dividing by the interquartile range. The input neurons then feed the values to each of the neurons in the hidden layer.

2. **Hidden layer** — This layer has one neuron for each case in the training data set. The neuron stores the values of the predictor variables for the case along with the target value. When presented with the x vector of input values from the input layer, a hidden neuron computes the Euclidean distance of the test case from the neuron's center point and then applies the RBF kernel function using the sigma value(s). The resulting value is passed to the neurons in the pattern layer.
3. **Pattern layer / Summation layer** — The next layer in the network is different for CNN networks and for GRNN networks. For CNN networks there is one pattern neuron for each category of the target variable. The actual target category of each training case is stored with each hidden neuron; the weighted value coming out of a hidden neuron is fed only to the pattern neuron that corresponds to the hidden neuron's category. The pattern neurons add the values for the class they represent (hence, it is a weighted vote for that category).
4. **Decision layer** — The decision layer is different for CNN and GRNN networks. For CNN networks, the decision layer compares the weighted votes for each target category accumulated in the pattern layer and uses the largest vote to predict the target category.

CONVOLUTIONAL NEURAL NETWORK ALGORITHM

Consider a network with a single real input x and network function F . The derivative $F'(x)$ is computed in two phases:

Feed-forward: the input x is fed into the network. The primitive functions at the nodes and their derivatives are evaluated at each node. The derivatives are stored.

Back propagation: The constant 1 is fed into the output unit and the network is run backwards. Incoming information to a node is added and the result is multiplied by the value stored in the left part of the unit. The result is transmitted to the left of the unit. The result collected at the input unit is the derivative of the network function with respect to x .

STEPS OF THE ALGORITHM

The Convolutional neural algorithm is used to compute the necessary corrections, after choosing the weights of the network randomly. The algorithm can be decomposed in the following four steps:

- i) Feed-forward computation
- ii) Convolutional neural to the output layer
- iii) Convolutional neural to the hidden layer
- iv) Weight updates

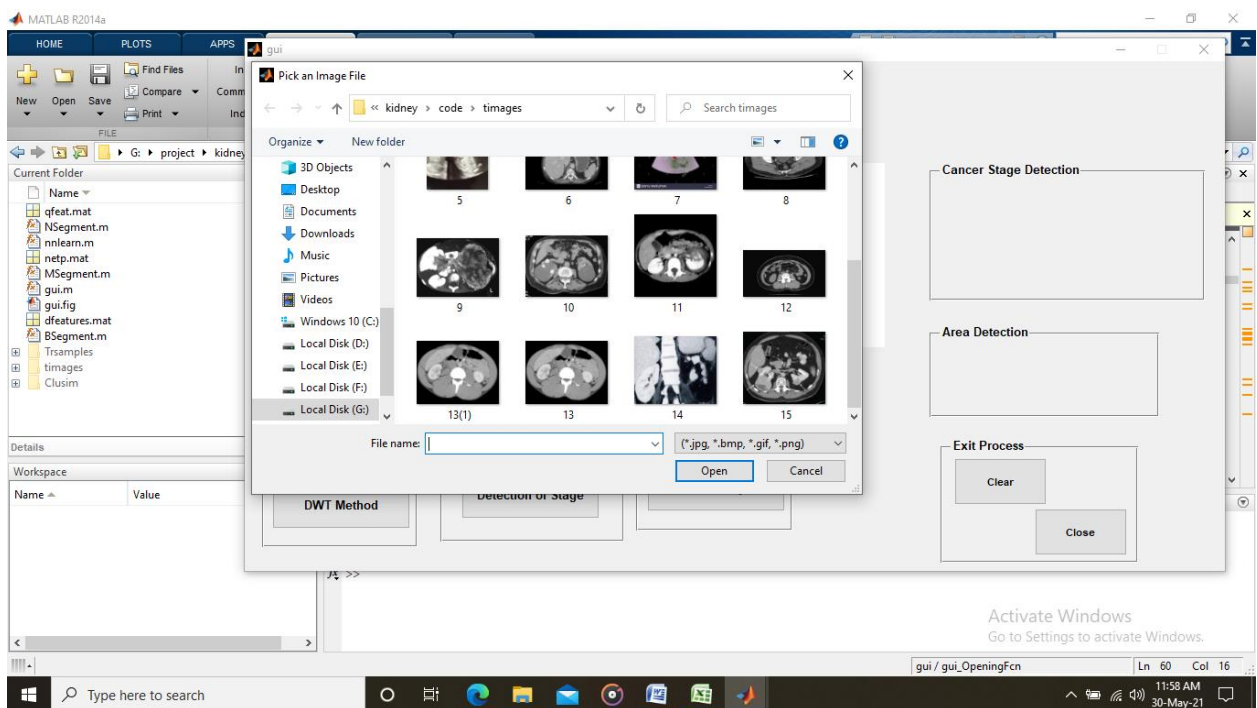
CHAPTER – 5

RESULT AND DISCUSSION

5.1 RESULT

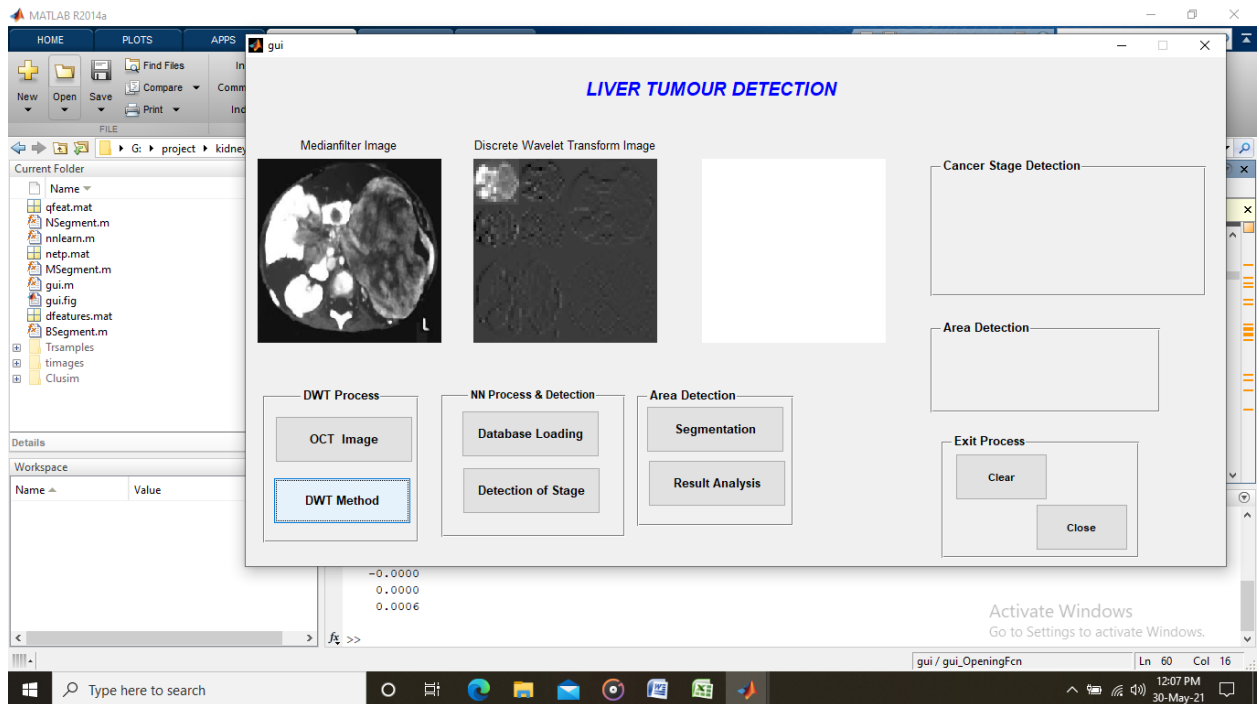
The input is given after execution and the liver tumor is detected..

Fig 1:



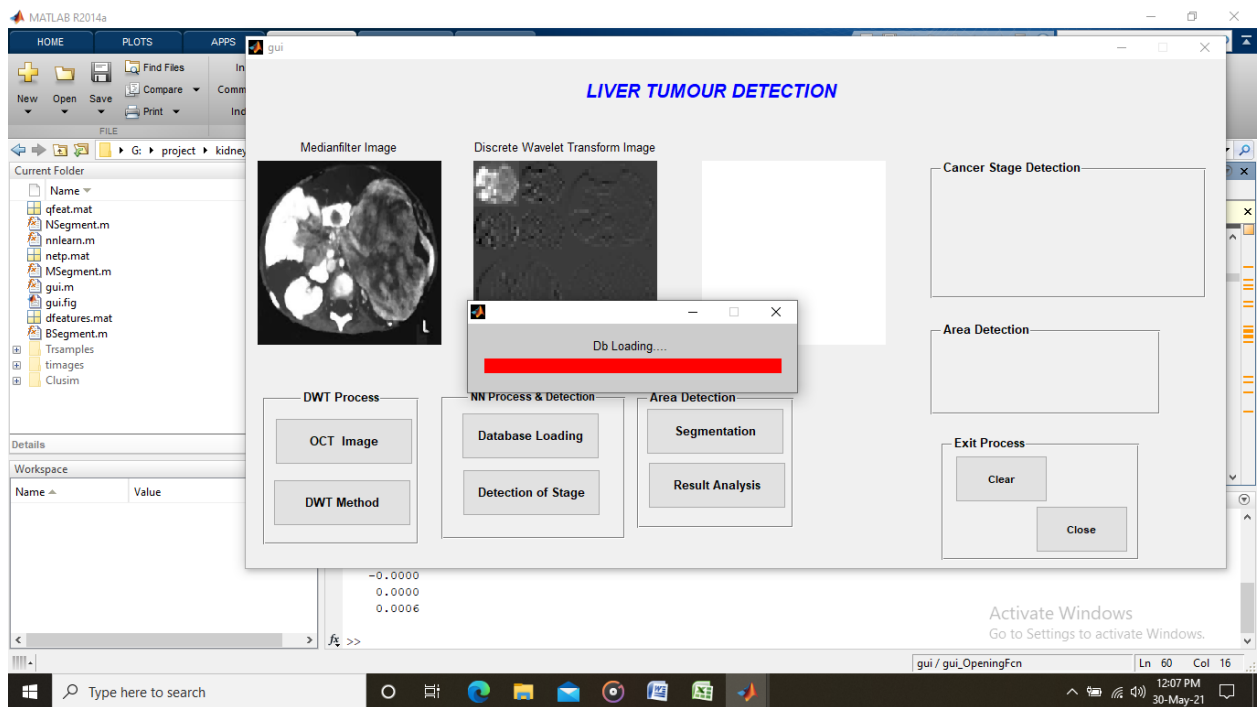
Uploading the input of the image

Fig 2:



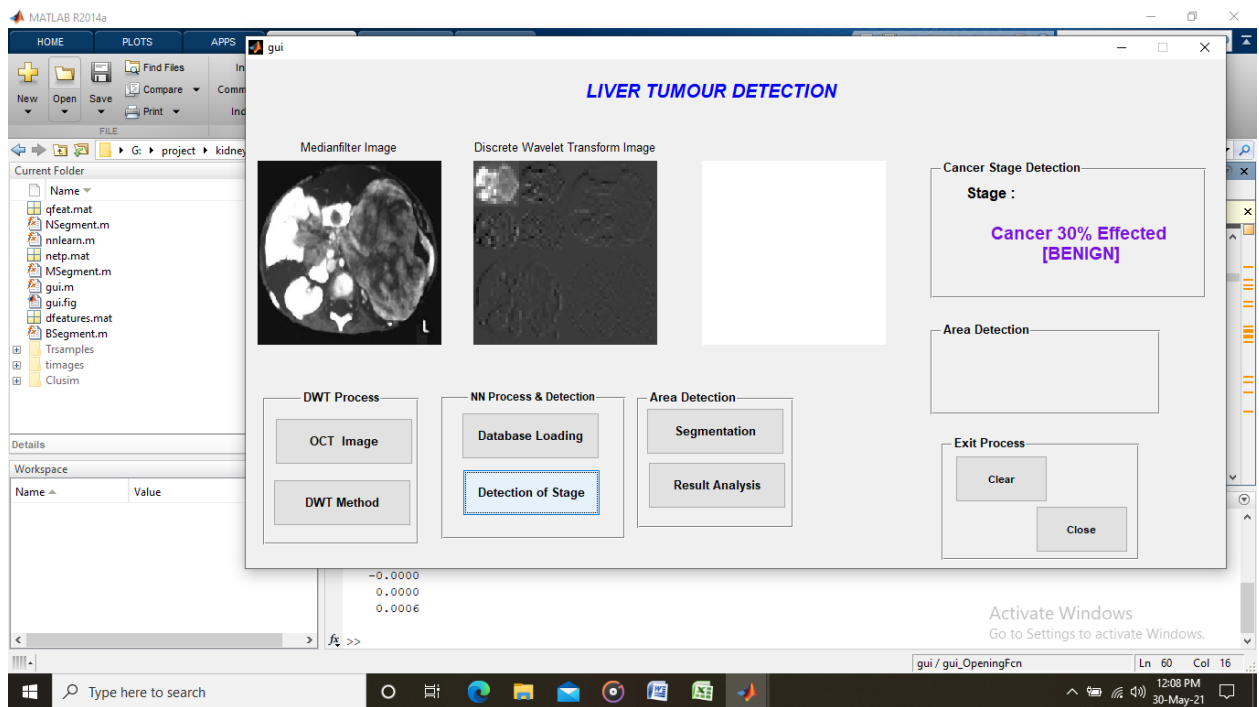
Finding discrete wavelet transform

Fig 3:



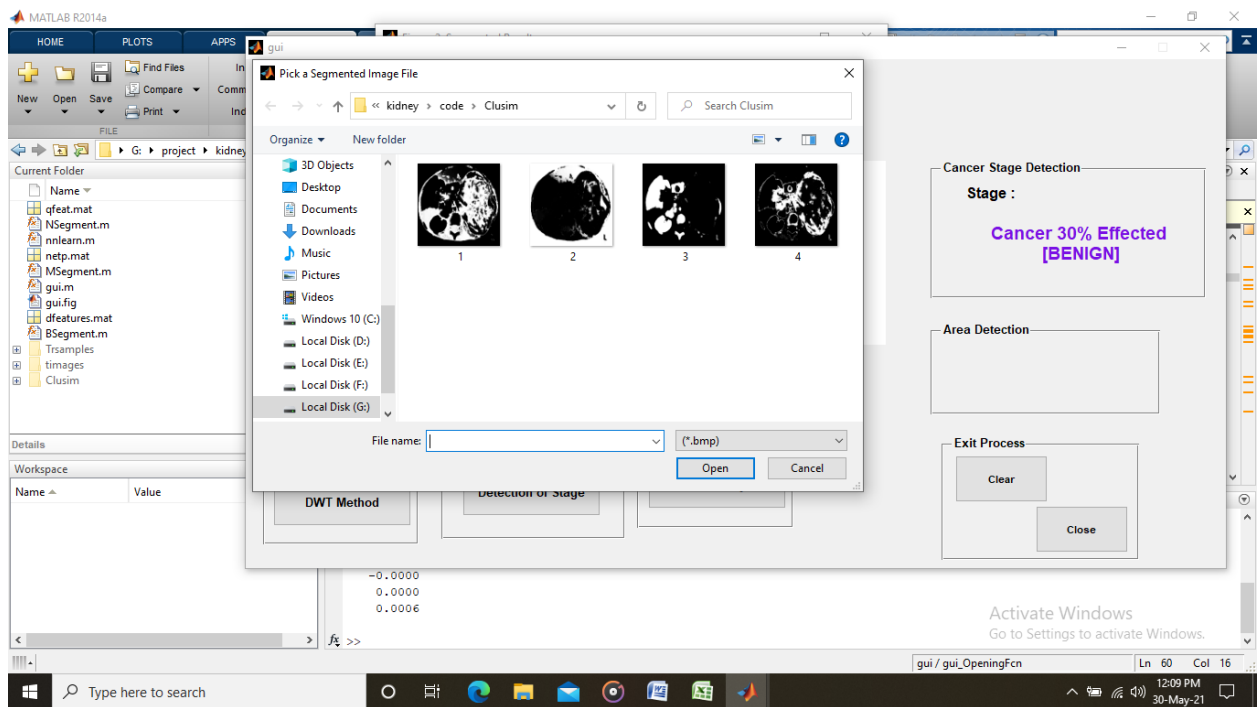
Loading of data base

Fig 4:



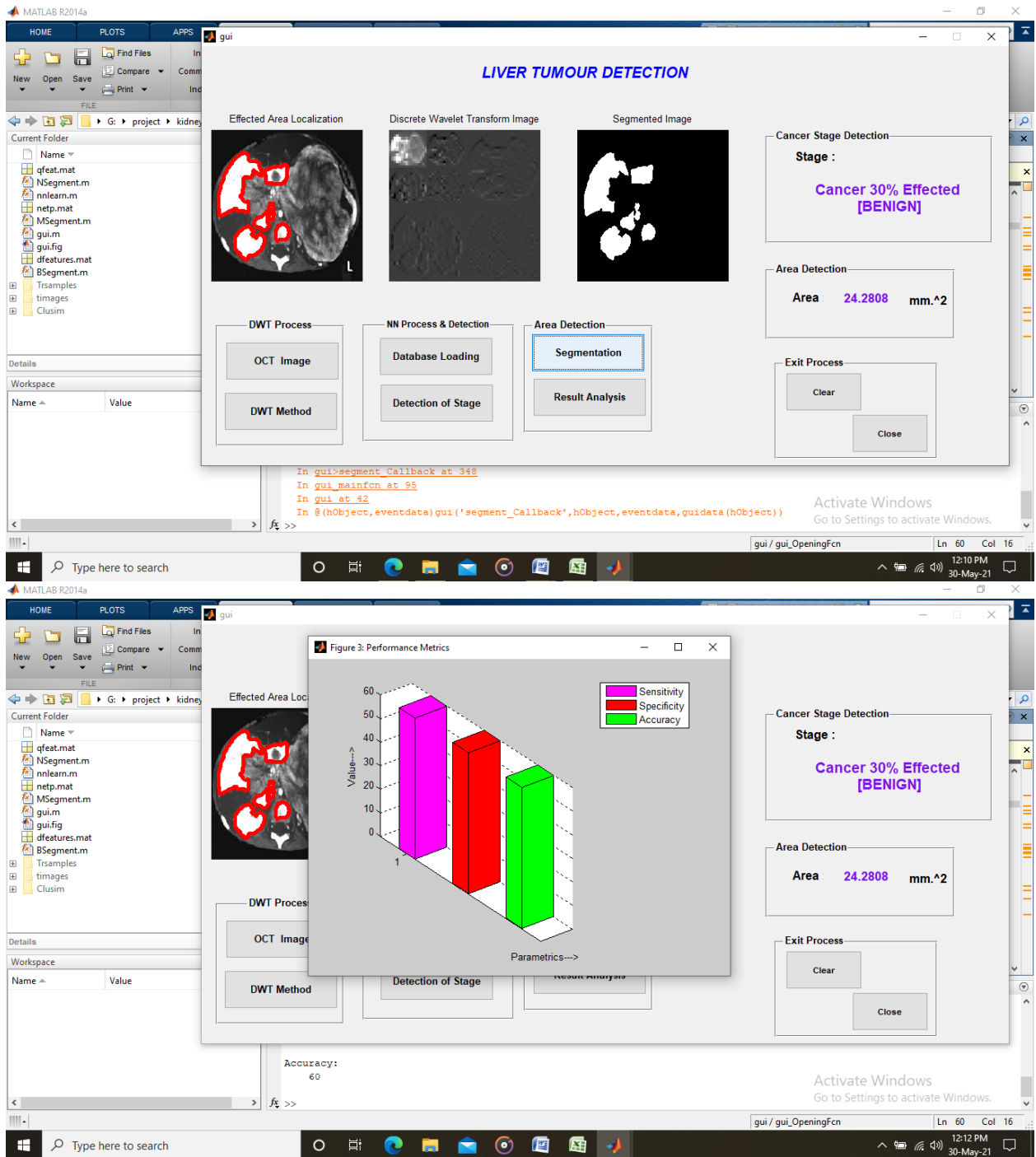
Finding the detection stage of tumor

Fig 5:



Showing cancer 30% effected

Fig 6:



Finally finding the output

DISCUSSION

Generally, cell proliferation and growth was done in both LT and LR, but the former are malignant, and the latter are controlled stringently.

According to our data, proliferation-promoting genes *pcna*, *ccne1*, *cdk4*, *ahr*, *wee1*, *ccna2*, *pin1*, *nek6* and *smo* were up-regulated in both LT and LR, and proliferation-inhibiting genes *creb3l3*, *pten*, *kit*, *gjb1*, *tff1* and *csda*, were down-regulated in both, indicating that these genes promote cell proliferation in the two events. Notably, the abundance of *CCNA2* mRNA in LT was approximately five-fold higher than that in normal liver, and it reached its peak with 45 folds of control at 66 h after PH, which might be associated with an increased proportion of regenerated hepatocytes. Growth-promoting genes *hspb1*, *grn*, *tgfb1* and *serpine1*, whose expression levels were elevated in LT, were up-regulated at metaphase of LR. Among these four genes, *serpine1* having the highest expression (16.7 folds higher than control) at 6 h following PH might explain why it played an important role in growth of the regenerated hepatocytes. Dysregulated expression of anti-apoptosis gene *bcl-2* was present in LT as well as at metaphase and anaphase of LR, and another anti-apoptotic gene *mcl1*, whose change trend toward up-regulation in LT was identical to that in LR; and down-regulation of pro-apoptosis genes *casp9* and *dlc1* occurred in LT and the metaphase of LR, which supported the idea that *mcl1*, *casp9* and *dlc1* might be related with cell survival in the two events.

CHAPTER – 6

SUMMARY AND CONCLUSION

6.1 SUMMARY:

- These results are offered to the community as an additional reference point, enabling a wide range of cancer biologists, clinical investigators, and genome and computational scientists to easily incorporate TCGA into the backdrop of ongoing research.
- While every effort is made to ensure that Firehose input data and algorithms are of the highest possible quality, these analyses have not been reviewed by domain experts..

6.2 CONCLUSION

To this aim detect the liver cancer by using deep learning concept here we create a graphical user interface for getting input MRI images, analysing and classifying the output. we comprehensively illustrate the wavelet transform and GLCM feature extraction technique, neural network, clustering and segmentation process to classify the output. By using these methods we got a more accuracy, specificity, and the performance also be increased .

REFERENCES

- . [1] Upadhyay, Y. and Wasson, V. 2014. "Analysis of Liver MR Images for Cancer Detection using Genetic Algorithm". International Journal of Engineering Research and General Science. Vol.2, No.4, PP: 730-737.
- [2] Kumar, P. Bhalerao, S. 2014. "Detection of Tumor in Liver Using Image Segmentation and Registration Technique". IOSR Journal of Electronics and Communication Engineering (IOSR-JECE). Vo.9, No.2, PP: 110-115.
- [3] Selle, D.; Spindler, W.; Preim, B. and Peitgen, H. O. 2000. "Mathematical Methods in Medical Imaging: Analysis of Vascular Structures for Liver Surgery Planning". PP: 1-21.
- [4] Zimmer, C. and Olivo-Marin, J. C. 2005. "Coupled Parametric Active Contours". Transactions on pattern Analysis and Machine Intelligence. Vol.27, No.11, PP: 1838-1841.
- [5] Chitra, S. and Balakrishnan, G. 2012. "Comparative Study for Two Color Spaces HSCbCr and YCbCr in Skin Color Detection". Applied Mathematical Sciences. Vol.6, No.85, PP: 4229 – 4238.
- [6] M. Lalonde, M. Beaulieu, and L. Gagnon, "Fast and robust optic disc detection using pyramidal decomposition and Hausdorff-based template matching," *IEEE Trans. Med. Imag.*, vol. 20, no. 11, pp. 1193–1200, Nov. 2001
- [7] T.Kauppi and H.Kälviäinen, "Simple and robust optic disc localization using colour decorrelated templates," in *Proc. 10th Int. Conf. Advanced Concepts for Intell. Vision Syst.*, Berlin, Germany: Springer-Verlag, 2008
- [8] Leida Li, Shushang Li, Hancheng Zhu, "An efficient scheme for detecting Copy-Move forged images by local binary patterns", Journal of

Information Hiding and Multimedia Signal Processing, Vol. 4, No. 1, pp. 46-56, January 2013.

- [9] Hailing Huang, Weiqiang Guo, Yu Zhang, “Detection of Copy-Move Forgery in Digital Images Using SIFT Algorithm”, IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, Wuhan, China, 2008.
- [10] Y.Sutcu, B.Coskun, H.T.Sencar, N.Memmon, “Tamper detection based on regularity of wavelet transform coefficients”, IEEE International Conference on Image Processing, 2007