



P-1648



IMAGE SEGMENTATION FOR MRI BRAIN TUMOR

A PROJECT REPORT

Submitted by

V.DIVYA	71202205012
SOUMYA VENUGOPALA MENON	71202205049
J.VITHYA	71202205057

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025


APRIL 2006

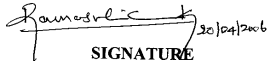
BONAFIDE

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "IMAGE SEGMENTATION FOR MRI BRAIN TUMOR" is the bonafide work of "Ms. V.DIVYA, Ms. SOUMYA VENUGOPALA MENON and Ms. J.VITHYA" who carried out the project work under my supervision.


SIGNATURE
Dr. G. Gopalsamy
HEAD OF THE DEPARTMENT



SIGNATURE
Mr. K. Ramasubramaniam
SUPERVISOR

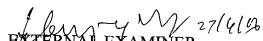
Dept of Information Technology,
Kumaraguru College of Technology,
Coimbatore - 641006.

Dept of Information Technology,
Kumaraguru College of Technology,
Coimbatore - 641006.

These candidates with University Register Nos. 71202205012, 71202205049, 71202205057 were examined in the project viva-voce examination held on 27.04.2006.....

DECLARATION


INTERNAL EXAMINER


EXTERNAL EXAMINER

DECLARATION

We,

V.DIVYA	71202205012
SOUMYA VENUGOPALA MENON	71202205049
J.VITHYA	71202205057

Declare that the project entitled "IMAGE SEGMENTATION FOR MRI BRAIN TUMOR", submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and the guidance of **Mr.K.Ramasubramaniam, M.E.**, Lecturer, Department of Information Technology, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore

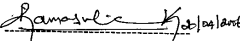
Date: 27.04.2006


[V.DIVYA]


[SOUMYA VENUGOPALA MENON]


[J.VITHYA]

Project Guided by


[Mr.K.Ramasubramaniam, M.E.,]

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

With all humility and sincerity, we express our gratitude to all those who have helped us in the completion of this project.

We use this opportunity to thank **Dr.K.K.Padmanabhan B.Sc. (Engg.), M.Tech., Ph.D.**, Principal, Kumaraguru College of Technology, Coimbatore, for his kind patronage.

We express our sincere gratitude to **Dr.G.Gopalsamy, Ph.D.**, Professor and Head of the Department of Information Technology, for the immense support he has provided us throughout our project.

We extend our sincere thanks to our Project Coordinator **Prof.K.R.Baskaran, B.E., M.S.**, Assistant Professor, Department of Information Technology, for his constant support and encouragement.

We are grateful and immensely indebted to our project guide **Mr.K.Ramasubramaniam, M.E.**, a benevolent and technically proficient personality, for his able guidance and motivation, in situations, when our project was in rough waters.

We are proud indeed to thank and be grateful to our Class Advisor, **Ms.N.Suganthi, M.E.**, an agile and dynamic person for her concern and motivation through the years.

We are very much grateful to all staff members of Information Technology Department for their kind support during the course of our project.

Words can but in a meager way, our thanks to all lab assistants, who have put in a lot of efforts in order to give us the right kind of tools and other important facilities. We also thank our parents for the backing we got from them during the course of completion of the project. Last but not the least we wish to express our humble thanks to our fellow classmates who made our lives on campus a cherishable experience which would be indelible in our minds for years from now.

ABSTRACT

ABSTRACT

This project describes a framework for automatic brain tumor segmentation from Magnetic Resonance (MR) images. The task of detecting the position of a tumor is the starting point for a medical treatment, the conformal radiotherapy, in which the tumor cells are irradiated and killed with a very high precision, avoiding, in the meanwhile, damage to the neighboring healthy tissues. Whereas many other tumor segmentation methods especially the manual brain tumor segmentation, the current gold standard, rely on the intensity enhancement produced by the gadolinium contrast agent in any one of the sequence image, the method proposed here does not require contrast enhanced image channels. The only required input for the segmentation procedure is the MR image channel, but it can make use of any additional non-enhanced image channels for improved tissue segmentation.

The segmentation framework is composed of three stages. First, we acquire and process the pixel data of the input image. We then make use of the basic arithmetic formulae to derive the results required for the following stages. In the second stage, we cluster the image based on the pixel data. We adopt KMeans clustering algorithm to perform this clustering operation. Finally, we apply geometric and spatial constraints to reconstruct the tumor and display segment tumor cluster with accurate boundary.

TABLE OF CONTENTS

ii

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	ii
	LIST OF FIGURES	v
1.	INTRODUCTION	1
	1.1. THE EXISTING SYSTEM	2
	1.2. THE PROPOSED SYSTEM	3
2.	SYSTEM REQUIREMENT ANALYSIS	5
	2.1. PROJECT DEFINITION	5
	2.2. PROJECT PLAN	5
	2.3. SOFTWARE REQUIREMENT SPECIFICATION	8
	2.3.1. PURPOSE	8
	2.3.2. SCOPE	8
	2.4. GENERAL DESCRIPTION	8
	2.4.1. PRODUCT OVERVIEW	8
	2.4.2. GENERAL ASSUMPTIONS	8
	2.5. DEVELOPMENT AND OPERATING SYSTEM	9
	2.5.1. SOFTWARE CONSTRAINTS	9
	2.5.2. HARDWARE CONSTRAINTS	9
	2.6. PERFORMANCE CONSTRAINTS	9

3.	SYSTEM STUDY	10
	3.1. IMAGE SEGMENTATION	10
	3.1.1. IMAGE SEGMENTATION MEHODS	11
	3.2. KMEANS CLUSTERING	12
	3.3. MAGNETIC RESONANCE IMAGING	13
	3.3.1. THE MAGNETS	14
	3.3.2. UNDERSTANDING THE TECHNOLOGY:ATOMS	15
	3.3.3. UNDERSTANDING THE TECHNOLOGY:RF	17
	3.3.4. ADVANTAGES	18
	3.4. BRAIN TUMORS	19
	3.5. JAVA	21
	3.5.1. JAVA FEATURES	21
	3.5.2. JAVA ENVIRONMENT	24
	3.5.3. JAVA PACKAGES	26
4.	DESIGN DOCUMENTS	28
	4.1. INPUT DESIGN	28
	4.2. PROCESS DESIGN	28
	4.3. OUTPUT DESIGN	31
5.	PRODUCT TESTING	32
6.	FUTURE ENHANCEMENTS	35
7.	CONCLUSION	36
8.	APPENDIX	37
	8.1. SAMPLE CODE	37
	8.2. SAMPLE OUTPUT	51
9.	REFERENCES	55

iii

iv

LIST OF FIGURES

S.NO.	FIG.NO.	NAME OF THE FIGURE	PAGE.NO.
1	3.3.1	A Sample MR Image of Brain with Tumor	14
2	3.3.2.1	A Hydrogen atom spins about a Magnetic field	16
3	3.3.2.2	Flow of atoms	17
4	3.3.4.1	Axial, Coronal and Sagittal slices	19
5	4.2.1	Flowchart representing KMeans clustering algorithm	31
6	8.2.1	Input image loaded	51
7	8.2.2	Cluster image 1	51
8	8.2.3	Cluster image 2	51
9	8.2.4	Cluster image 3	52
10	8.2.5	Cluster image 4	52
11	8.2.6	Cluster image 5	52
12	8.2.7	Cluster image 6	52
13	8.2.8	Cluster image 7	53
14	8.2.9	Cluster image 8	53
15	8.2.10	Cluster image 9	53
16	8.2.11	Cluster image 10	53
17	8.2.12	Reconstructed tumor image	54

INTRODUCTION

v

1. INTRODUCTION

Segmentation of medical imagery is a challenging task due to the complexity of the images, as well as to the absence of models of the anatomy that fully capture the possible deformations in each structure. Brain tissues are particularly complex structures, and its segmentation is an important step for derivation of computerized anatomical atlases, as well as pre- and intra-operative guidance for therapeutic intervention.

Brain tumors are difficult to segment because they have a wide range of appearance and effect on surrounding structures. Following are some of the general characteristics of brain tumors:

- vary greatly in size and position,
- vary greatly in the way they show up in MRI,
- may have overlapping intensities with normal tissue,
- may be space occupying (new tissue that moves normal structure) or infiltrating (changing properties of existing tissue),
- may enhance fully, partially, or not at all, with contrast agent, and
- may be accompanied by surrounding edema (swelling).

Image segmentation is one of the primary steps in image analysis for object identification. Segmentation partitions an image into distinct regions that are meant to correlate strongly with objects or features of interest in the image.

MRI segmentation has been proposed for a number of clinical investigations of varying complexity.

1.1. THE EXISTING SYSTEM

Presently manual segmentation is the common method followed for the process of segmenting brain tumors. Manual segmentation of brain tumors from magnetic resonance images is a challenging and time-consuming task.

The task of manually segmenting brain tumors from magnetic resonance imaging (MRI) is generally time-consuming and difficult. In most settings, the task is performed by marking the tumor regions slice-by-slice, which limits the human rater's view and generates jaggy images.

Manual segmentation is also typically performed largely based on a single image with intensity enhancement provided by an injected contrast agent. As a result, the segmented images are less than optimal.

Manual brain tumor segmentation, the current gold standard, is a time-consuming and tedious process that involves identifying image regions in 3-D volumes that deviate from the unexpected intensities.

DISADVANTAGES

- Greatly time-consuming and difficult
- Limits the human rater's view
- Generates jaggy images
- Involves more of user guidance
- The segmented images are less than optimal

1.2. THE PROPOSED SYSTEM

Automatic brain tumor segmentation from Magnetic Resonance (MR) images is a challenging task that offers exposure to various disciplines covering pathology, MRI physics, radiologist's perception, and image analysis based on intensity and shape.

Previous work (Manual Segmentation) on brain tumor segmentation typically uses the enhancement provided by the gadolinium contrast agent in the T1 channel (one of the sequence in MR Images) or blobby shaped tumors with uniform intensity. Even though the intensity enhancement can aid the segmentation process, we show that it is not always necessary to obtain good results. In fact, the use of a contrast agent can be problematic. Typically, tumors are only partially enhanced and some early developing tumors are not enhanced at all. Blood vessels also generally appear enhanced by the contrast agent. These inconsistencies create an ambiguity in the image interpretation, which makes the T1-enhanced image channel a less than ideal feature for tumor segmentation.

Our method combines the model of the normal tissues and the geometric and spatial model of tumor. It relies only on the information provided in any of the sequence image channels. Tumor is treated as intensity abnormalities or outliers. After identifying the abnormalities, a clustering technique is applied to the intensity features before utilizing geometric and spatial constraints.

3

The method proposed here for the automatic segmentation of brain tumor processes the input image in three stages as follows:

1. *Basic Image Processing*
 - Acquiring and processing pixel data
2. *Cluster Separation*
 - Based on a Pixel-based classification method, KMeans clustering
3. *Reconstruction of Tumor*
 - Retaining only the segmented tumor cluster and eliminating other clusters

This approach with the inclusion of the above three steps is found to be efficient than the previous approaches.

ADVANTAGES

The advantages of our approach are as follows:

- Reduces the load on the human raters
- Generates segmentations that take the information within the entire 3D multiparameter images
- Automatic and user guidance not required
- Structural and intensity characteristics are well-known up to a natural biological variability
- Optimal and improved efficiency

4

2. SYSTEM REQUIREMENT ANALYSIS

System study is an activity that encompasses most of the tasks that we have collectively called computer system engineering. System study is conducted with the following:

- Identify the needs
- Evaluate the system concept for feasibility
- Perform economic and technical analysis
- Allocate functions to hardware, software ,people and other system elements
- Create a system definition that forms a foundation for all subsequent engineering works

2.1. PROJECT DEFINITION

Our project proposes a method that automatically segments brain tumor, has a significant advantage that structural and intensity characteristics are well known up to a natural biological variability or the presence of pathology. The segmentation process is performed based on the clustering algorithm.

2.2. PROJECT PLAN

In the analysis phase, we learned about the various technologies and identified the one most suitable for the project implementation. From this study, the most suitable language selected is Java. Differentiation of modules, user interface design and other crucial aspects were identified and designed as necessary. Other ambiguities which may exist are identified and taken care of.

5

SYSTEM REQUIREMENTS ANALYSIS

After the analysis phase, the design phase commences in which the various modules and functionalities are identified. The complete system flow of control and data are also identified.

Next the implementation phase is taken care of in which the design is translated into code. Each module is coded separately and finally integrated to form the entire system. Care is taken to make the code easily understandable by future users.

In the testing phase each module is tested thoroughly and finally the integrated modules are tested together to ensure the correct working of the entire system. Testing is also done to ensure that the product satisfies the specified requirements and set criteria.

WORK	DURATION
<ul style="list-style-type: none"> • Feasibility Analysis • Abstract Preparation • Requirements Gathering • Selecting sample website 	One week
<ul style="list-style-type: none"> • Study and analysis of the concepts of Image Processing, MRI and Brain tumor 	Three weeks
<ul style="list-style-type: none"> • Study about Java components 	Two weeks
<ul style="list-style-type: none"> • Creating a module for loading of the image • Acquiring and processing pixel data 	One week
<ul style="list-style-type: none"> • Study about KMeans Clustering algorithm 	Two weeks
<ul style="list-style-type: none"> • Implementing the clustering algorithm on the image for segmentation 	Two weeks
<ul style="list-style-type: none"> • Reconstructing the tumor from the segmented image 	Three weeks
<ul style="list-style-type: none"> • Integrating the modules and finally testing 	One week

2.3. SOFTWARE REQUIREMENTS SPECIFICATION

2.3.1. PURPOSE

The purpose of this document is to specify the requirements of our project "Image Segmentation for MRI Brain Tumor". It describes the interfaces for the system. The document also provides a detailed explanation for image processing.

2.3.2. SCOPE

SRS forms the basis of agreement between the client and the supplier and what the software product will do. It also provides a reference for the validation of the final project.

Any changes made to the SRS in the future will have to go through formal change approval process.

2.4. GENERAL DESCRIPTION

2.4.1. PRODUCT OVERVIEW

This project aims to provide high-quality segmentation of brain tumors from MR images. It makes use of the KMeans clustering algorithm to partition the image into clusters with varying values. Based on the results obtained from clustering the tumor is segmented with accurate boundary.

2.4.2. GENERAL ASSUMPTIONS

- The number of clusters to initialize the KMeans clustering algorithm is taken as ten.
- The initial set of centroids formed based on the range calculated from the image.
- The optimum threshold limit set for the tumor intensity.

2.5. DEVELOPMENT AND OPERATING ENVIRONMENT

2.5.1. SOFTWARE CONSTRAINTS

Operating System : Windows XP
 Language : Java
 IDE : NetBeans IDE 3.6

2.5.2. HARDWARE CONSTRAINTS

Processor : Intel Pentium IV
 RAM : 256 MB
 Hard disk : 10 GB

2.6. PERFORMANCE CONSTRAINTS

The system should run smoothly as long as a valid input image is given. The system should consider and process each pixel to provide maximum accuracy.

SYSTEM STUDY

3. SYSTEM STUDY

3.1. IMAGE SEGMENTATION

Segmentation is generally the first stage in any attempt to analyze or interpret an image automatically. Segmentation partitions an image into distinct regions that are meant to correlate strongly with objects or features of interest in the image. Segmentation can also be regarded as a process of grouping together pixels that have similar attributes. For Segmentation to be useful, the regions or groups of pixels that are generated should be meaningful.

Segmentation bridges the gap between low-level image processing, which concerns itself with the manipulation of pixel grey level or color to correct defects or enhance certain characteristics of the image, and high-level processing, which involves the manipulation and analysis of groups of pixels that represent particular features of interest. Some kind of segmentation technique will be found in any application involving the detection, recognition and measurement of objects in images. Examples of such applications include:

- Industrial inspection
- Optical Character Recognition (OCR)
- Tracking of objects in a sequence of images
- Classification of terrains visible in satellite images
- Detection and measurement of bone, tissue, etc., in medical images

Segmentation techniques can be classified as either *contextual* or *non-contextual*. Non-contextual techniques ignore the relationships that exist between features in an image; pixels are simply grouped together on the basis of some *global* attribute, such as grey level. Contextual techniques, on the other hand, additionally exploit the relationships between image features. Thus, a contextual technique might group together pixels that have similar grey levels *and* are close to one another.

3.1.1. IMAGE SEGMENTATION METHODS

- **Edge-based methods**, in which the edge information is used to determine boundaries of objects. The boundaries are then analyzed and modified, if needed, to form closed regions belonging to the objects in the image.
 - Edge Detection
 - Boundary Tracking
 - Hough Transform
- **Pixel-based direct classification methods**, in which heuristics or estimation methods derived from the histogram statistics of the image are used to form closed regions belonging to the objects in the image.
 - Optimal Global Thresholding
 - Pixel Classification Through Clustering
 - K Means Clustering
 - FUZZY C Mean Clustering
 - Adaptive FCM Algorithm
- **Region-based methods**, in which pixels are analyzed directly for a region growing process based on a pre-defined similarity criterion to form closed regions belonging to the objects in the image.
 - Region Growing
 - Region Splitting

3.2. K MEANS CLUSTERING

K-means (MacQuenn, 1976) is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more. Finally, this algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function

$$J = \sum_{i=1}^k \sum_{j=1}^n \|x_i^{(j)} - c_j\|^2$$

where $\|x_i^{(j)} - c_j\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j , is an indicator of the distance of the n data points from their respective cluster centers.

The algorithm is composed of the following steps:

- Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
- Assign each object to the group that has the closest centroid.
- When all objects have been assigned, recalculate the positions of the K centroids.
- Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

The main advantages of this algorithm are its simplicity and speed which allows it to run on large datasets. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments. It maximizes inter-cluster (or minimizes intra-cluster) variance, but does not ensure that the result has a global minimum of variance.

3.3. MAGNETIC RESONANCE IMAGING (MRI)

Magnetic Resonance Imaging is based on the measurement of magnetic RadioFrequency(RF) signals from hydrogen molecules.

- The contrast differences in MR signals are based primarily on the amount of hydrogen present in different tissues.

In an MRI machine, the basic design used in most is a giant **cube**. The cube in a typical system might be 7 feet tall by 7 feet wide by 10 feet long (2 m by 2 m by 3 m), although new models are rapidly shrinking. There is a **horizontal tube** running through the **magnet** from front to back. This tube is known as the **bore** of the magnet.

13

magnet above about the 0.3-tesla level would be prohibitively expensive.

- A **permanent magnet** is just that -- permanent. Its magnetic field is always there and always on full strength, so it costs nothing to maintain the field. The major drawback is that these magnets are extremely heavy: They weigh many, many tons at the 0.4-tesla level. A stronger field would require a magnet so heavy it would be difficult to construct. Permanent magnets are getting smaller, but are still limited to low field strengths.
- **Superconducting magnets** are by far the most commonly used. A Superconducting Magnet is somewhat similar to a resistive magnet -- coils or windings of wire through which a current of electricity is passed create the magnetic field. The important difference is that the wire is continually bathed in liquid helium at 452.4 degrees below zero. Superconductive systems are still very expensive, but they can easily generate 0.5-tesla to 2.0-tesla fields, allowing for much higher-quality imaging.

3.3.2. UNDERSTANDING THE TECHNOLOGY: ATOMS

The human body is made up of untold billions of atoms, the fundamental building blocks of all matter. The nucleus of an atom spins, or **precesses**, on an axis. There are many different types of atoms in the body, but for the purposes of MRI, we are only concerned with the **hydrogen atom**. It is an ideal atom for MRI because its nucleus has a **single proton** and a large **magnetic moment**. The large magnetic moment means that, when placed in a magnetic field, the hydrogen atom has a strong tendency to line up with the direction of the magnetic field.

15

In conjunction with radio wave pulses of energy, the MRI scanner can pick out a very small **point** inside the patient's body and ask it, essentially, "What type of tissue are you?" The point might be a cube that is half a millimeter on each side. The MRI system goes through the patient's body point by point, building up a 2-D or 3-D **map** of tissue types. It then integrates all of this information together to create **2-D images** or **3-D models**.

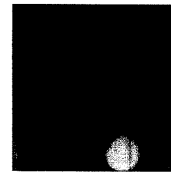


Fig.3.3.1. A sample MR Image of Brain with Tumor

By changing exam parameters, the MRI system can cause tissues in the body to take on different appearances. This is very helpful to the radiologist (who reads the MRI) in determining if something seen is normal or not.

3.3.1. THE MAGNETS

The biggest and most important component in an MRI system is the **magnet**. The magnet in an MRI system is rated using a unit of measure known as a **tesla**. **Metal objects** can become dangerous projectiles if they are taken into the scan room. There are three basic types of magnets used in MRI systems:

- **Resistive magnets** consist of many windings or coils of wire wrapped around a cylinder or bore through which an electric current is passed. This causes a magnetic field to be generated. To operate this type of

14

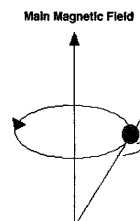
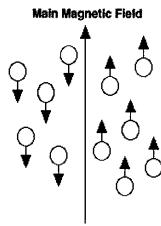


Fig 3.3.2.1. A hydrogen atom spins about a magnetic field.

Inside the **bore** of the scanner, the magnetic field runs straight down the center of the tube in which we place the patient. This means that if a patient is lying on his or her back in the scanner, the hydrogen protons in his or her body will line up in the direction of either the feet or the head. The vast majority of these protons will **cancel each other out** -- that is, for each one lined up toward the feet, one toward the head will cancel it out. Only a couple of protons out of every million are not canceled out. This doesn't sound like much, but the sheer number of hydrogen atoms in the body gives us what we need to create wonderful images.

16



All of the hydrogen protons will align with the magnetic field in one direction or the other. The vast majority cancels each other out, but, as shown here, in any sample there is one or two "extra" protons.

Fig 3.3.2.2. Flow of Atoms

3.3.3. UNDERSTANDING THE TECHNOLOGY: RF

The MRI machine applies an RF (radio frequency) pulse that is specific only to hydrogen. The system directs the pulse toward the area of the body we want to examine. The pulse causes the protons in that area to **absorb the energy** required to make them spin, or **precess**, in a different direction. This is the "**resonance**" part of MRI. The RF pulse forces them (only the one or two extra unmatched protons per million) to spin at a particular frequency, in a particular direction. The specific frequency of resonance is called the **Larmour frequency** and is calculated based on the particular tissue being imaged and the strength of the main magnetic field.

These RF pulses are usually applied through a **coil**. MRI machines come with many different coils designed for different parts of the body: knees, shoulders, wrists, heads, necks and so on. These coils usually conform to the

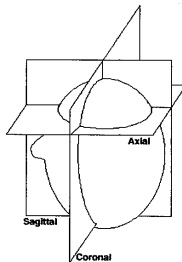


Fig 3.3.4.1. Axial, coronal and sagittal slices

3.4. BRAIN TUMORS

Brain tumors are masses created by the growth of abnormal cells or uncontrolled proliferation of cells in the brain. Any abnormal growth within the skull creates a special problem because it is in a confined space and will press on normal brain tissue and will therefore interfere with the functions of the body controlled by the affected parts.

The causes of brain tumor are not known. About 5% of primary brain tumors are associated with hereditary disorders, including Li-Fraumeni cancer family syndrome, p53 defects, tuberous sclerosis, von Recklinghausen's disease (neurofibromatosis), von Hippel Landau disease, familial polyposis (Turcot's syndrome), and Osler-Weber-Rendu syndrome.

contour of the body part being imaged, or at least reside very close to it during the exam. At approximately the same time, the three **gradient magnets** jump into the act.

When the RF pulse is turned off, the hydrogen protons begin to slowly (relatively speaking) return to their **natural alignment** within the magnetic field and **release** their excess stored energy. When they do this, they give off a signal that the coil now picks up and sends to the computer system. What the system receives is mathematical data that is converted, through the use of a **Fourier transform**, into a picture that we can put on film. That is the "imaging" part of MRI.

3.3.4. ADVANTAGES

MRI is ideal for:

- Diagnosing **multiple sclerosis** (MS)
- Diagnosing **tumors** of the pituitary gland and brain
- Diagnosing **infections** in the brain, spine or joints
- Visualizing **tear ligaments** in the wrist, knee and ankle
- Evaluating **bone tumors, cysts and bulging or herniated discs** in the spine

These are but a few of the many of reasons to perform an MRI scan.

Another major advantage of MRI is its ability to image in **any plane**. CT is limited to **one plane**, the axial plane. An MRI system can create **axial images** as well as images in the **sagittal plane** (slicing the bread side-to-side lengthwise) and **coronally** (think of the layers of a layer cake) or any degree in between, without the patient ever moving.

Brain tumors produce a variety of symptoms ranging from headache to stroke. They are **great mimics** of other neurologic disorders. Symptoms occur if the tumor directly damages the nerves in the brain or central nervous system or if its growth imposes pressure on the brain. Some gliomas develop gradually and symptoms may be subtle for a long time, making an early diagnosis difficult.

Brain tumor is treated surgically. As a result of recent progress in the methods of brain surgery, many cases of brain tumor can now be operated on successfully. Some may be completely excised (removed). Tumors that are deep, or that infiltrate brain tissue, may be debulked (removal of much of the mass of the tumor to reduce its size). Surgery may reduce intracranial pressure and relieve symptoms in cases when the tumor cannot be removed. Radiation therapy may be advised for tumors that are sensitive to this treatment.

3.5. JAVA

Java is a general-purpose, object-oriented programming language developed by Sun Microsystems of USA in 1991. Java was designed for the development of software for consumer electronic devices like TVs, VCRs, toasters and such other electronic machines. This goal had a strong impact on the development team to make the language simple, portable and highly reliable. The Java team which included Patrick Naughton discovered that the existing languages like C and C++ had limitations in terms of both reliability and portability. However, they modelled their new language Java on C and C++ but removed a number of features of C and C++ that were considered as sources of problems and thus made Java a really simple, reliable, portable, and powerful language.

3.5.1. JAVA FEATURES

Although the fundamental forces that necessitated the invention of Java are portability and security, other factors also played an important role in molding the final form of the language. The key considerations are:

- Simple
- Secure
- Portable
- Object-oriented
- Robust
- Multithreaded
- Architecture-neutral
- Interpreted
- High Performance
- Distributed
- Dynamic

21

ROBUST

The multiplatformed environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. Thus, the ability to create robust programs was given a high priority in the design of Java. To better understand how Java is robust, consider two of the main reasons for program failure: memory management mistakes and mishandled exceptional conditions.

MULTITHREADED

Java was designed to meet the real-world requirement of creating interactive, networked programs. To accomplish this, Java supports multithreaded programming, which allows us to write programs that do many things simultaneously. The Java run-time system comes with an elegant yet sophisticated solution for multiprocess synchronization that enables us to construct smoothly running interactive systems.

ARCHITECTURE-NEUTRAL

A central issue for the Java designers was that of code longevity and portability. Operating system upgrades, processor upgrades, and changes in core system resources can all combine to make a program malfunction. The Java designers made several hard decisions in the Java language and the Java Virtual Machine in an attempt to alter this situation. Their goal was "write once; run anywhere, any time, forever." To a great extent, this goal was accomplished.

INTERPRETED AND HIGH PERFORMANCE

Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java bytecode. This code can be interpreted on any system that provides a Java Virtual Machine.

23

SIMPLE

Java was designed to be easy for the professional programmer to learn and use effectively. Java inherits the C/C++ syntax and many of the object-oriented features of C++. Also, some of the more confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. Beyond its similarities with C/C++, Java has another attribute that makes it easy to learn: it makes an effort not to have surprising features. In Java, there are a small number of clearly defined ways to accomplish a given task.

SECURITY

Java provides a "firewall" between a networked application and our computer. When we use a Java-compatible Web browser, we can safely download Java applets without fear of viral infection or malicious intent. Java achieves this protection by confining a Java program to the Java execution environment and not allowing it access to other parts of the computer.

PORTABILITY

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed. This mechanism helps ensure security and also helps create portability.

OBJECT-ORIENTED

Although influenced by its predecessors, Java was not designed to be source-code compatible with any other language. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance nonobjects.

22

DISTRIBUTED

Java is designed for the distributed environment of the Internet, because it handles TCP/IP protocols. In fact, accessing a resource using a URL is not much different from accessing a file. The original version of Java (Oak) included features for intra-address-space messaging. This allowed objects on two different computers to execute procedures remotely. Java revived these interfaces in a package called Remote Method Invocation (RMI). This feature brings an unparalleled level of abstraction to client/server programming.

DYNAMIC

Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time. This makes it possible to dynamically link code in a safe and expedient manner. This is crucial to the robustness of the applet environment, in which small fragments of bytecode may be dynamically updated on a running system.

3.5.2. JAVA ENVIRONMENT

Java environment includes a large number of development tools and hundreds of classes and methods. The development tools are part of the system known as Java Development Kit (JDK) and the classes and methods are part of the Java Standard Library (JSL), also known as the Application Programming Interface (API).

JAVA DEVELOPMENT KIT

The Java Development Kit comes with a collection of tools that are used for developing and running Java programs. They include:

- appletviewer (for viewing Java applets)
- javac (Java compiler)

24

- java (Java interpreter)
- javap (Java disassembler)
- javah (for C header files)
- javadoc (for creating HTML documents)
- jdb (Java debugger)

APPLICATION PROGRAMMING INTERFACE

The Java Standard Library (or API) includes hundreds of classes and methods grouped into several functional packages. Most commonly used packages are:

- **Language Support Package:** A collection of classes and methods required for implementing basic features of Java.
- **Utilities Package:** A collection of classes to provide utility functions such as date and time functions.
- **Input/Output Package:** A collection of classes required for input/output manipulation.
- **Networking Package:** A collection of classes for communicating with other computers via Internet.
- **AWT Package:** The Abstract Window Tool Kit package contains classes that implements platform-independent graphical user interface.
- **Applet Package:** This includes a set of classes that allows us to create Java applets.

25

java.lang

Provides classes that are fundamental to the design of the Java programming language.

java.math

Provides classes for performing arbitrary-precision integer arithmetic (**BigInteger**) and arbitrary-precision decimal arithmetic (**BigDecimal**).

3.5.3. JAVA PACKAGES

javax.swing

Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

java.io

Provides for system input and output through data streams, serialization and the file system.

java.awt.Mediatracker

Provides for the programmers to load multiple images synchronously by checking the status of an arbitrary number of images in parallel.

java.awt.image

Provides classes for creating and modifying images.

java.awt

Contains all of the classes for creating user interfaces and for painting graphics and images.

java.util

Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).

26

DESIGN DOCUMENTS

4. DESIGN DOCUMENTS

4.1. INPUT DESIGN

The input image is a grey-scale Magnetic Resonance Image (MRI) of a brain. The image can be of any sequence and channel. This segmentation method proposed here accepts both enhanced and non-enhanced image channel of the MR image of a brain.

4.2. PROCESS DESIGN

Automatic brain tumor segmentation from MR images is a difficult task that involves various disciplines covering pathology, MRI physics, radiologist's perception, and image analysis based on intensity and shape. There are many issues and challenges associated with brain tumor segmentation. Brain tumors may be of any size, may have a variety of shapes, may appear at any location, and may appear in different image intensities.

As mentioned earlier, the segmentation framework is composed of three stages:

- Basic image processing
- Cluster separation
- Reconstruction of tumor

First, we acquire and process the pixel data from the input image to calculate the data required for the following processes. These are calculated using the basic arithmetic formulae.

The second stage is cluster separation. Clustering is a data mining technique that separates your data into groups whose members belong together. Each object is assigned to the group it is most similar to. Clustering does not require a prior knowledge of the groups that are formed and the members who must belong to it. We have adopted KMeans clustering algorithm to perform the clustering.

28

Stable clusters are formed when new iterations or repetitions of the K-Means clustering algorithm does not create new clusters as the cluster center or Arithmetic Mean of each cluster formed is the same as the old cluster center. There are different techniques for determining when a stable cluster is formed or when the k-means clustering algorithm procedure is completed.

There are many methods for finding a satisfactory set of centroids given a set of data. The simplest is to pick an initial set of centroid randomly (assuming we know how many clusters we want) and to assign each point to its closest centroid. After each assignment, we need to update the assigned centroid by adding in the coordinates of the new point (a simple calculation). Assigning all points to a set of successively updated centroids constitutes one iteration of the k-means algorithm.

Each new iteration consists of a re-assignment of all points, until no point can be moved to a centroid closer than the one for the cluster it is already a member of. Every time a point is re-assigned, its old centroid must be down dated and its new centroid must be updated.

30

KMEANS CLUSTERING

The K-Means algorithm partitions a dataset into clusters as follows:

- It accepts the **number of clusters** to group data into, and the **dataset** to cluster as input values.
- It then creates the first **K initial clusters** (K= number of clusters needed) from the dataset by choosing K rows of data randomly from the dataset.
- The K-Means algorithm calculates the **Arithmetic Mean** of each cluster formed in the dataset. The Arithmetic Mean of a cluster is the mean of all the individual records in the cluster.
- Next, K-Means assigns each record in the dataset to **only one** of the initial clusters. Each record is assigned to the **nearest cluster** (the cluster which it is most similar to) using a measure of distance or similarity like the **Euclidean Distance Measure** or **Manhattan/City-Block Distance Measure**.
- K-Means re-assigns each record in the dataset to the most similar cluster and re-calculates the arithmetic mean of all the clusters in the dataset. The arithmetic mean of a cluster is the arithmetic mean of all the records in that cluster. This new arithmetic mean becomes the **center of this new cluster** (centroid). Following the same procedure, new **cluster centers** are formed for all the existing clusters.
- K-Means re-assigns each record in the dataset to **only one** of the new clusters formed. A record or data point is assigned to the **nearest cluster** (the cluster which it is most similar to) using a measure of distance or similarity like the **Euclidean Distance Measure** or **Manhattan/City-Block Distance Measure**
- The preceding steps are repeated until **stable clusters** are formed and the K-Means clustering procedure is completed.

29

This process can be briefly explained with the help of the following flow diagram:

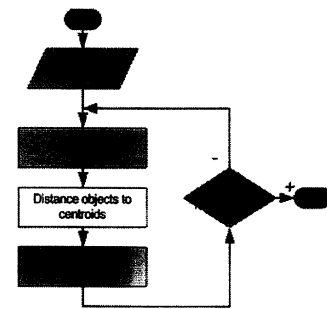


Fig.4.2.1. Flowchart representing KMeans clustering algorithm

Finally, we apply geometric and spatial constraints to reconstruct the tumor and display segment tumor cluster with accurate boundary.

4.3. OUTPUT DESIGN

The output is a high-quality segmented image of the tumor which is present in the input MR image with a precise delineation of tumor boundaries.

31

PRODUCT TESTING

EVENT TESTING

Each item present in the menu will be tested to check whether the appropriate actions are triggered. This in a way helps in checking the functionality of menu items.

INTEGRATION TESTING

This is done to check whether the modules present in the developing system are interconnected as per the requirements. This sort of checking helps in analyzing the behavior of the program when subjected to perform certain functions.

DISPLAY TESTING

In fact this test is the effective test to be carried out. The display procedures are tested since the data displayed is of much importance. The consistency of the display and the attractiveness of the display were also tested.

A testing is an examination with the intent of finding the errors. Concentration was more on errors than resting on the glory of apparently perfect outputs. In the case of Image Segmentation, two types of test were mainly conducted namely

- Unit Testing
- Integrity testing

UNIT TESTING

In unit testing each individual program is tested and would be checked whether it satisfies with the required output. As an illustration in the project:

- Verification for loading of grayscale image
- Checking for pixel processing
- Testing for updated centroid values

5. PRODUCT TESTING

Testing is a vital process to the success of any system. The system as a whole will be tested for the following

- Consistency with the application
- Displaying the resultant image
- Integrity of the modules
- Referential integrity test

System testing makes a logical assumption that if all the parts of the system are correct, the system will be successfully achieved. The objective of testing is to discover errors. To fulfill these objectives, a series of test is planned to execute. Software testing can be looked upon as one among many processes. This is the last opportunity to correct any possible flaws in the developed system. Software testing includes selection test data that have more probability of finding errors.

Systems are the stage of implementation that is aimed at ensuring that the system works accurately and efficiently before live operation commences. In principle, system proving is an on-going activity throughout the project.

The first step in system testing is to develop a plan that tests all the aspects of the system. Completeness, correctness, reliability and maintainability of the software are to be tested for the best quality assurance that the system meets the specification and the requirements for its intended users and performance. System testing is most useful practical process of executing a program with explicit intention of finding errors that make the program fail. The following phases are developed for the purpose of testing the system.

MODULE TESTING

Each individual program module will test for any possible errors. They will be tested for specification i.e., to see whether they work as per what the program should do and how it should perform under various conditions.

- Display of the reconstructed image

The individual programs are then combined to form modules.

INTEGRITY TESTING

It is the testing performed to detect errors in the interconnections between the modules. Integrity Testing will be performed on each of the modules and again the validity will be checked. After all the modules are brought under single module, integrity testing is performed and the result should yield success.

SYSTEM TESTING

The system is tested against the system requirements to see if all the requirements are met and to ensure that the system performs as per the specified requirements. The system is tested as a whole to check for its functionality.

VALIDATION TESTING

This test is done to check for the validity of the entered input. The input provided by the user is the grayscale image file. All other images are ignored and further processing is not done.

6. FUTURE ENHANCEMENTS

The segmentation method presented in detects abnormal regions in the brain based on the image intensities. Other properties can also be used for this process. This can include geometric properties such as curvature or brain asymmetry. Although the contrast enhanced image channel leads to ambiguous information, there are cases where it leads to more straightforward identification of brain tumors, assuming that enhanced blood vessels and noise can be properly identified. Robust estimation schemes other than the MCD may be necessary for these extensions.

An issue that goes together with the issue of knowing the deformation induced by tumor is the problem of determining the possible shapes of brain tumors. The shape model for tumor enforced using region competition snake constrains the segmented tumor to have rather smooth shapes. The notion of spatial coherence for brain tumors need to be properly enforced in order to segment wider varieties of brain tumors. This is a difficult issue because tumors can appear in many different sizes and shapes.

Our method can integrate the detection of edema in addition to tumor as a combined approach, although knowledge of the extent of edema is critical for planning and treatment.

FUTURE ENHANCEMENTS

35

7. CONCLUSION

Our method presents a new approach for automatic segmentation of tumors from non-enhancing multichannel MRI. Most methods so far have been applicable only to enhancing, homogeneous tumors. Furthermore, they require user-guidance in training a supervised classifier or to obtain a rough outline of the region of interest. Here, we show that segmentation and outlier detection can be a promising new concept for detecting abnormalities in the brain.

The comparison between the new method and manual segmentations obtained from medical experts demonstrate comparable quality and significant reduction of operator time. Our algorithm may fail in cases where the intensity value distribution of the tumor is highly inhomogeneous and shows large spectral overlap with spatially close brain tissue. In such a case the initial tumor estimate may impose an incorrect spatial constraint on the classification process that may not be resolved in subsequent iterations. However, this can easily be corrected by extending the manual interaction.

In overall, the result is an efficient, automatic segmentation method that defines tumor and the algorithm produces results of comparable accuracy to those of the manual segmentation in shorter time.

CONCLUSION

36

8. APPENDIX

8.1. SAMPLE CODE

```
//Image Segmentation for MRI Brain Tumor

import javax.swing.*;
import java.io.*;
import java.awt.MediaTracker.*;
import java.awt.image.*;
import java.awt.*;
import javax.imageio.*;
import java.util.*;
import java.lang.Math.*;

public class brain extends javax.swing.JFrame
{
    Image bi;
    int NoOfCluster=10;
    double e=0.1;
    static ImageIcon im2;

    public brain()
    {
        initComponents();
    }

    private void initComponents()
    {
        java.awt.GridBagConstraints gridBagConstraints;
        JPanel1 = new javax.swing.JPanel();
        JButton1 = new javax.swing.JButton();
        JLabel2 = new javax.swing.JLabel();
        JPanel2 = new javax.swing.JPanel();
        JLabel1 = new javax.swing.JLabel();
        JPanel3 = new javax.swing.JPanel();
        JButton2 = new javax.swing.JButton();
        JButton4 = new javax.swing.JButton();
        JButton3 = new javax.swing.JButton();

        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        JPanel3.setBackground(new java.awt.Color(153, 255, 255));
        JButton2.setBackground(new java.awt.Color(51, 204, 255));
        JButton2.setFont(new java.awt.Font("MS Sans Serif", 1, 11));
        JButton2.setText("Segmentation");
        JButton2.setBorder(new
        javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        JButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                JButton2ActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.insets = new java.awt.Insets(9, 2, 8, 2);
        JPanel3.add(JButton2, gridBagConstraints);
        JButton4.setBackground(new java.awt.Color(51, 204, 255));
        JButton4.setFont(new java.awt.Font("MS Sans Serif", 1, 11));
        JButton4.setText("Reconstruction");
        JButton4.setBorder(new
        javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        JButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                JButton4ActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.insets = new java.awt.Insets(7, 0, 7, 0);
        JPanel3.add(JButton4, gridBagConstraints);
        JButton3.setText("Exit");
        JButton3.setBorder(new
        javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        JButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                JButton3ActionPerformed(evt);
            }
        });

        JPanel3.add(JButton3, new java.awt.GridBagConstraints());
        getContentPane().add(JPanel3, java.awt.BorderLayout.CENTER);
        java.awt.Dimension screenSize =
        java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-500)/2, (screenSize.height-400)/2, 500, 400);
    }
}

}
});

JPanel1.setLayout(new java.awt.GridBagLayout());
JPanel1.setBackground(new java.awt.Color(153, 255, 255));
JButton1.setBackground(new java.awt.Color(102, 204, 255));
JButton1.setFont(new java.awt.Font("MS Sans Serif", 1, 11));
JButton1.setForeground(new java.awt.Color(0, 51, 51));
JButton1.setText("Browse");
JButton1.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
JButton1.setMargin(new java.awt.Insets(12, 24, 12, 24));
JButton1.setPreferredSize(new java.awt.Dimension(75, 17));
JButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        JButton1ActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.gridheight = 3;
gridBagConstraints.insets = new java.awt.Insets(11, 13, 12, 11);
JPanel1.add(JButton1, gridBagConstraints);
JLabel2.setBorder(new javax.swing.border.EtchedBorder(java.awt.Color.lightGray,
java.awt.Color.gray));
JLabel2.setPreferredSize(new java.awt.Dimension(255, 255));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridheight = 6;
gridBagConstraints.ipadx = 32;
gridBagConstraints.ipady = 49;
gridBagConstraints.anchor = java.awt.GridBagConstraints.SOUTH;
gridBagConstraints.insets = new java.awt.Insets(0, 1, 0, 1);
JPanel1.add(JLabel2, gridBagConstraints);
getContentPane().add(JPanel1, java.awt.BorderLayout.WEST);
JPanel2.setLayout(new java.awt.BorderLayout());
JPanel2.setBackground(new java.awt.Color(153, 255, 255));
JLabel1.setBackground(new java.awt.Color(0, 153, 255));
JLabel1.setFont(new java.awt.Font("Times New Roman", 1, 14));
JLabel1.setForeground(new java.awt.Color(51, 51, 0));
JLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
JLabel1.setText("Brain Tumour Segmentation");
JLabel1.setBorder(new javax.swing.border.MatteBorder(null));
JPanel2.add(JLabel1, java.awt.BorderLayout.CENTER);
getContentPane().add(JPanel2, java.awt.BorderLayout.NORTH);
JPanel3.setLayout(new java.awt.GridBagLayout());
```

APPENDIX

```

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{
    System.exit(0);
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{
    jLabel2.setIcon(im2);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
    int height=bi.getHeight(null);
    int width=bi.getWidth(null);
    System.out.println("Image: width="+width+" height="+height);
    int rgb[]=new int[width*height];
    int index=0;
    int pixel[][]=new int[height][width];
    double inputpix[][]=new double[height][width];
    double resultpix[][]=new double[height][width];
    int outpix[][]=new int[height][width];
    double outpix1[][]=new double[height][width];
    int pix[][]=new int[height][width][4];
    BufferedImage b1=new
BufferedImage(width,height,BufferedImage.TYPE_BYTE_GRAY);
    Graphics2D gt=b1.createGraphics();
    gt.drawImage(bi,null,null);
    b1.getRGB(0,0,width,height,rgb,0, width);
    for(int i=0;i<height;i++)
    {
        for(int j=0;j<width;j++)
        {
            pixel[i][j]=rgb[index];
            index++;
        }
    }
    pix=convertToThreeDim(rgb,width,height);
    System.out.println("Image intensity values :");
    for(int i=0;i<height;i++)
    {
        for(int j=0;j<width;j++)
        {
            inputpix[i][j]=((double)pix[i][j][1])/255;
            System.out.println(inputpix[i][j]+"t"+i+"t"+j);
            try

```

40

```

double[] dist=new double[10];
double[] temp=new double[10];
for(int i=0;i<NoOfCluster;i++)
{
    dist[i]=centroid[i][0];
}
while(max1(dist)>threshold)
{
    for(int i=0;i<NoOfCluster;i++)
    {
        temp[i]=centroid[i][0];
    }
    double diff[]=new double[10];
    double tempr=0.0,tempr1=0.0,tempr2=0.0;
    //initial clustering
    for(int i=0;i<height;i++)
    {
        for(int j=0;j<width;j++)
        {
            for(int k=0;k<NoOfCluster;k++)
            {
                diff[k]=Math.abs(inputpix[i][j]-centroid[k][0]);
                if(inputpix[i][j]!=0.0)
                {
                    tempr=inputpix[i][j];
                    tempr2=centroid[k][0];
                    tempr=diff[k];
                }
                int minval=minval(diff);
                outpix[i][j]=minval;
            }
        }
    }
    //new centroid
    double[] sum=new double[NoOfCluster];
    int[] count=new int[NoOfCluster];
    for(int i=0;i<height;i++)
    {
        for(int j=0;j<width;j++)
        {
            for(int k=0;k<NoOfCluster;k++)
            {
                if(outpix[i][j]==k)
                {
                    sum[k]=sum[k]+outpix1[i][j];
                    count[k]=count[k]+1;
                }
            }
        }
    }

```

42

```

{
    Thread.sleep(1);
}
catch(Exception e)
{}
}

double mean=mean(inputpix);
System.out.println("mean : "+mean);
double threshold=mean*e;
System.out.println("threshold : "+threshold);
double minpix=min(inputpix);
System.out.println("minpix : "+minpix);
double maxpix=max(inputpix);
System.out.println("maxpix : "+maxpix);
double range=maxpix-minpix;
System.out.println("range : "+range);
double step=range/NoOfCluster;
System.out.println("step : "+step);
double interval=step/10;
System.out.println("interval : "+interval);
double[][] centroid=new double[NoOfCluster][1];
System.out.println("Initial centroids : ");

for(int i=0;i<NoOfCluster;i++)
{
    centroid[i][0]=interval;
    interval=interval+0.11;
    System.out.println(centroid[i][0]);
    try
    {
        Thread.sleep(500);
    }
    catch(Exception e)
    {}
}
for(int i=0;i<height;i++)
{
    for(int j=0;j<width;j++)
    {
        outpix1[i][j]=inputpix[i][j];
    }
}
int loop=1;

```

41

```

}
}
// update centroid
System.out.println("Updated centroids : ");
for(int k=0;k<NoOfCluster;k++)
{
    centroid[k][0]=sum[k]/count[k];
    try
    {
        Thread.sleep(200);
    }
    catch(Exception e)
    {}
    System.out.println(" "+centroid[k][0]);
}
for(int i=0;i<NoOfCluster;i++)
{
    dist[i]=0;
}
//new distance
System.out.println("New distances : ");
for(int ii=0;ii<NoOfCluster;ii++)
{
    dist[ii]=temp[ii]-centroid[ii][0];
    try
    {
        Thread.sleep(200);
    }
    catch(Exception e)
    {}
    System.out.println(dist[ii]);
}
for(int i=0;i<height;i++)
{
    for(int j=0;j<width;j++)
    {
        resultpix[i][j]=(double)centroid[(outpix[i][j])[0]];
    }
}
double[][] result=new double[10][height][width];
for(int k=0;k<10;k++)
{
    for(int i=0;i<height;i++)
    {

```

43


```

        for(int j=0;j<width;j++)
        {
            result[k][i][j]=0;
        }
    }
}

for(int i=0;i<10;i++)
{
    for(int i1=0;i1<height;i1++)
    {
        for(int j1=0;j1<width;j1++)
        {
            if(outpix[i1][j1]==i)
            {
                result[i][i1][j1]=centroid[i][0];
                System.out.println(" "+ result[i][i1][j1]);
            }
        }
    }
}

double[] centval=new double[10];
centval[0]=0.0;
centval[1]=0.1;
centval[2]=0.2;
centval[3]=0.3;
centval[4]=0.4;
centval[5]=0.5;
centval[6]=0.6;
centval[7]=0.7;
centval[8]=0.8;
centval[9]=0.9;
int indx=0;
int[][] resultmat=new int[10][height*width];
int[] resultmat1=new int[height*width];

for(int i=0;i<10;i++)
{
    for(int i1=0;i1<height;i1++)
    {
        for(int j1=0;j1<width;j1++)
        {
            if(result[i][i1][j1]>centval[i])
            {
                resultmat[i][indx]=1*255;
                indx++;
            }
        }
    }
}

```

44

```

        {
            int element = row * imgCols + col;
            aRow[col] = oneDPix[element];
        }
    }
    for(int col = 0; col < imgCols; col++)
    {
        //Alpha data
        data[row][col][0] = (aRow[col] >> 24)
            & 0xFF;
        //Red data
        data[row][col][1] = (aRow[col] >> 16)
            & 0xFF;
        //Green data
        data[row][col][2] = (aRow[col] >> 8)
            & 0xFF;
        //Blue data
        data[row][col][3] = (aRow[col])
            & 0xFF;
    }
}
return data;
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    jf=new JFileChooser();
    File fl=null;
    int s= jf.showOpenDialog(this);
    if(s==jf.APPROVE_OPTION)
    {
        fl=jf.getSelectedFile();
    }
    try
    {
        File file=new File(fl.getAbsolutePath());
        bi=ImageIO.read(file);
        ImageIcon im=new ImageIcon(bi);
        jLabel2.setIcon(im);
    }
    catch(Exception e){}
}

private void exitForm(java.awt.event.WindowEvent evt)
{
    System.exit(0);
}

```

46

```

        }
        else
        {
            resultmat[i][indx]=(int)result[i][i1][j1]*255;
            indx++;
        }
    }
}
indx=0;
}

for(int i=0;i<10;i++)
{
    for(int j=0;j<resultmat[i].length;j++)
    {
        resultmat[i][j]=((255<<24) & 0xFF000000)/((resultmat[i][j]<<16) &
0x00FF0000)/((resultmat[i][j]<<8))/(resultmat[i][j] & 0x000000FF);
    }
    BufferedImage b2=new
BufferedImage(width,height,BufferedImage.TYPE_BYTE_GRAY);
    b2.setRGB(0,0,width,height, resultmat[i], 0, width);
    ImageIcon im=new ImageIcon(b2);
    imload=new ImageLoader();
    imload.load(im);
    imload.show();
}

for(int j=0;j<resultmat[9].length;j++)
{
    resultmat[9][j]=((255<<24) & 0xFF000000)/((resultmat[9][j]<<16) &
0x00FF0000)/((resultmat[9][j]<<8))/(resultmat[9][j] & 0x000000FF);
}
BufferedImage b3=new
BufferedImage(width,height,BufferedImage.TYPE_BYTE_GRAY);
    b3.setRGB(0,0,width,height,resultmat[9], 0, width);
    im2=new ImageIcon(b3);
}

int[][] convertToThreeDim(
int[] oneDPix,int imgCols,int imgRows)
{
    int[][] data=new int[imgRows][imgCols][4];
    for(int row = 0; row < imgRows; row++)
    {
        int[] aRow = new int[imgCols];
        for(int col = 0; col < imgCols; col++)

```

45

```

        }
    }
}

public static void main(String args[])
{
    new brain().show();
}

public double mean(double[][] image)
{
    int height=image.length;
    int width=image[1].length;
    double sum=0,count=0;

    for(int i=0;i<height;i++)
    {
        for(int j=0;j<width;j++)
        {
            sum=sum+image[i][j];
            count++;
        }
    }
    return (sum/count);
}

public double min(double[][] image)
{
    double temp=0.0;
    for(int i=0;i<image.length;i++)
    {
        for(int j=0;j<image[i].length;j++)
        {
            if(temp==0)
            {
                temp=image[i][j];
            }
            else if(temp>image[i][j])
            {
                temp=image[i][j];
            }
        }
    }
    return temp;
}

public double max(double[][] image)
{

```

47

```

double temp=0;
for(int i=0;i<image.length;i++)
{
    for(int j=0;j<image[i].length;j++)
    {
        if(temp==0)
        {
            temp=image[i][j];
        }
        else if(temp<image[i][j])
        {
            temp=image[i][j];
        }
    }
}
return temp;
}

private double max1(double[] dist)
{
    double temp=0;
    for(int i=0;i<dist.length;i++)
    {
        if(i==0)
        {
            temp=dist[i];
        }
        else if(temp<dist[i])
        {
            temp=dist[i];
        }
    }
    return temp;
}

private double min1(double[] dist)
{
    double temp=0;
    for(int i=0;i<dist.length;i++)
    {
        if(i==0)
        {
            temp=dist[i];
        }
        else if(temp>dist[i])
        {
            temp=dist[i];
        }
    }
}

```

48

```

        temp=dist[i];
    }
}
return temp;
}

public int minval(double[] value)
{
    double temp=0.0;
    int index=0;
    for(int i=0;i<value.length;i++)
    {
        if(i==0)
        {
            temp=value[i];
            index=i;
        }
        else if(temp>value[i])
        {
            temp=value[i];
            index=i;
        }
    }
    return index;
}

private javax.swing.JFileChooser jf;
private imloader imload;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
}

```

IMAGE LOADER

```

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

public class imloader extends javax.swing.JFrame

```

49

```

{
    public imloader()
    {
        initComponents();
    }
    private void initComponents()
    {
        jLabel1 = new javax.swing.JLabel();
        getContentPane().setLayout(null);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });
        getContentPane().add(jLabel1);
        jLabel1.setBounds(40, 30, 250, 270);
        java.awt.Dimension screenSize =
        java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-353)/2, (screenSize.height-360)/2, 353, 360);
    }

    private void exitForm(java.awt.event.WindowEvent evt)
    {
        this.dispose();
    }
    public static void main(String args[])
    {
        new imloader().show();
    }

    public void load(ImageIcon im)
    {
        jLabel1.setIcon(im);
    }

    private javax.swing.JLabel jLabel1;
}

```

50

8.2. SAMPLE OUTPUT

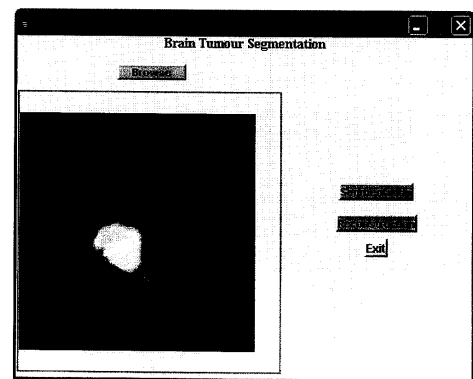


Fig.8.2.1. Input image loaded

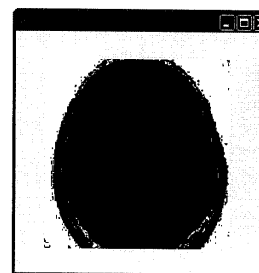


Fig.8.2.2. Cluster image 1

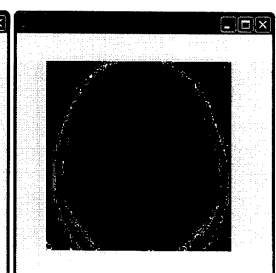
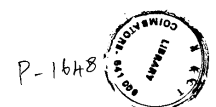


Fig.8.2.3. Cluster image 2

51



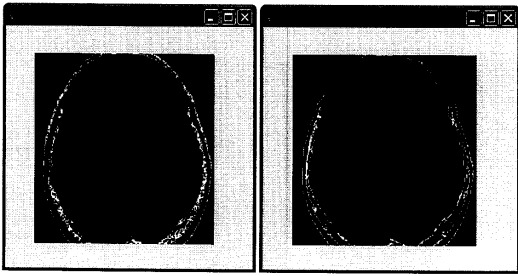


Fig.8.2.4. Cluster image 3

Fig.8.2.5. Cluster image 4

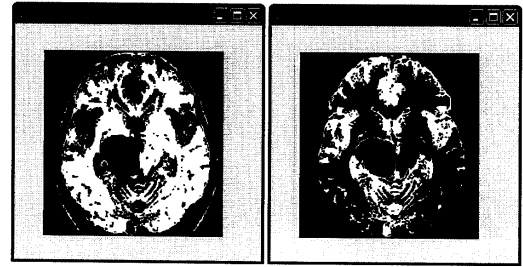


Fig.8.2.8. Cluster image 7

Fig.8.2.9. Cluster image 8

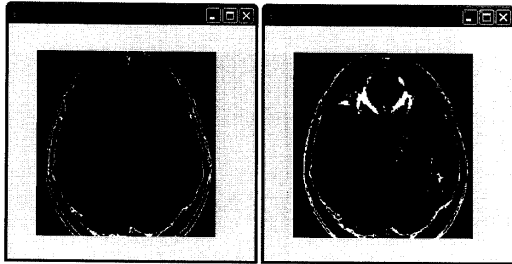


Fig.8.2.6. Cluster image 5

Fig.8.2.7. Cluster image 6

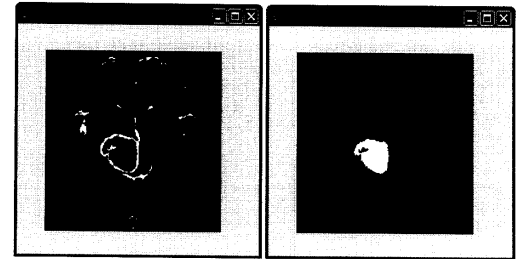


Fig.8.2.10. Cluster image 9

Fig.8.2.11. Cluster image 10

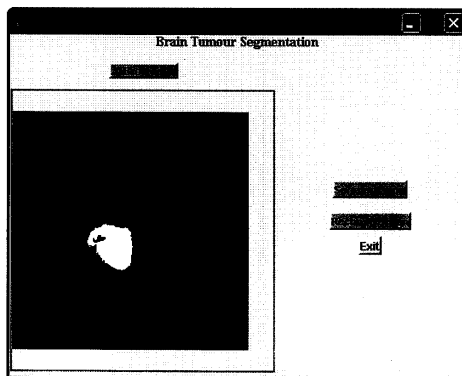


Fig.8.2.12. Reconstructed tumor image

REFERENCES

9. REFERENCES

1. Nick Efford, "Digital Image Processing – A Practical Introduction Using Java", Pearson Education Limited, 2000.
2. Brummer, M.E., Eisner, R.L., "Automatic Detection of Brain Contours in MRI Data Sets," IEEE Transactions on Medical Imaging, 1993.
3. Clark, M.C., Hall, L.O., Goldof, D.B., Clarke, L.P., Velthuizen, R.P., Silbiger, M.S., "MRI segmentation using fuzzy clustering techniques", IEEE Engineering in Medicine and Biology, November 1994.
4. Moon, N., Bullitt, E., Van Leemput, K., Gerig, G.: Automatic brain and tumor segmentation. In Dohi, T., Kikinis, R., eds.: Medical Image Computing and Computer-Assisted Intervention MICCAI 2002.
5. Clarke LP, Velthuizen RP, Camacho J, et al., "MRI segmentation: methods and applications", Magnetic Resonance Imaging 1995
6. Helmuth Späth, "Cluster Analysis Algorithms for Data Reduction and Classification of Object," Halsted Press, 1980.
7. Anil K. Jain and Richard C. Dubes. "Algorithms for Clustering Data" Prentice Hall, 1988.
8. Dan Pelleg and Andrew Moore, "Accelerating exact k-means algorithms with geometric reasoning", Carnegie Mellon University, Pittsburgh, PA.
9. James Theiler and Galen Gisler, "A contiguity-enhanced k-means clustering algorithm for unsupervised multispectral image segmentation", Proc SPIE 3159, 1997.