**1.INTRODUCTION**

**1.1 About Machine Learning based tumor detection**

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers.

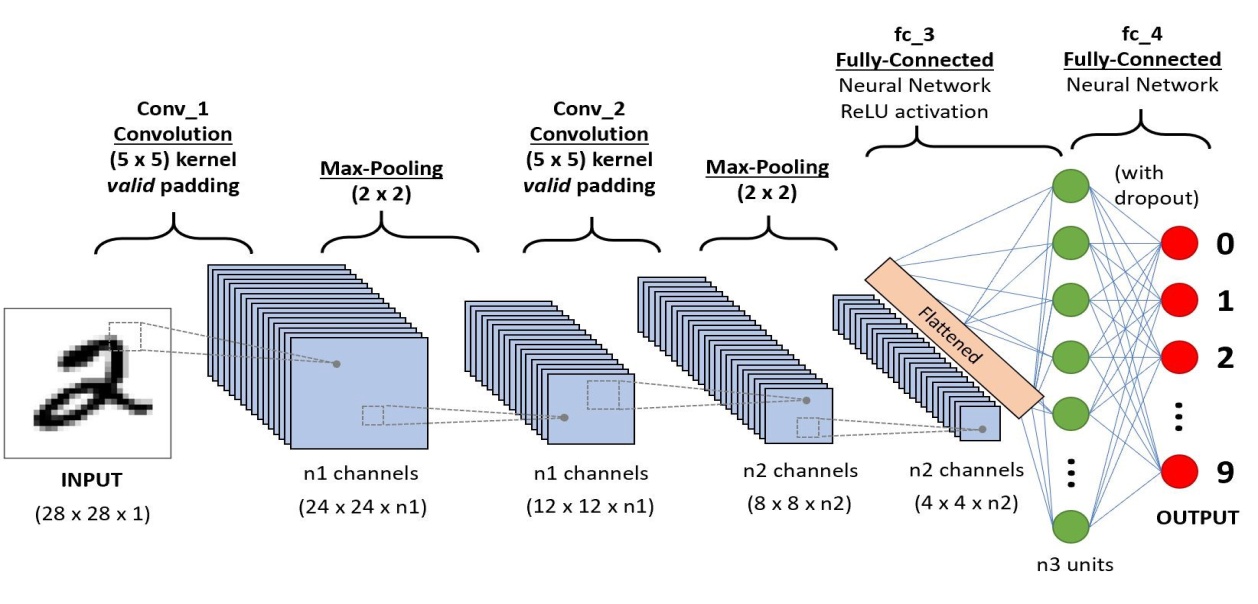
Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.Two of the most widely adopted machine learning methods are supervised learning which trains algorithms based on example input and output data that is labeled by humans, and unsupervised learning which provides the algorithm with no labeled data in order to allow it to find structure within its input data.

**1.2Dynamic Analysis**

It is presented a technique which employed texture features, wavelet transform, and SVM’s algorithm for effective classification of dynamic contrast-enhanced MR images, to handle the nonlinearity of real data and to address different image protocols effectively. It also claim that their proposed technique gives better predictions and improved clinical factors, tumor volume, and tumor stage in comparison with first-order statistical features.

**1.3 Network Model**

Traditional machine learning methods (such as multilayer perception machines, support vector machines, etc.) mostly use shallow structures to deal with a limited number of samples and computing units. When the target objects have rich meanings, the performance and generalization ability of complex classification problems are obviously insufficient. The convolution neural network (CNN) developed in recent years has been widely used in the field of image processing because it is good at dealing with image classification and recognition problems and has brought great improvement in the accuracy of many machine learning tasks. It has become a powerful and universal deep learning model.



In the classification and localization task not only do you have to report the class of object found in the image, but also the coordinates of the bounding box where the object appears in the image. This type of task assumes that there is only one instance of the object in an image.

This can be achieved by attaching a “regression head” in addition to the “classification head” in a typical classification network. Recall that in a classification network, the final output of convolution and pooling operations, called the feature map, is fed into a fully connected network that produces a vector of class probabilities. This fully connected network is called the classification head, and it is tuned using a categorical loss function (Lc) such as categorical cross entropy.The coordinator lays down the format for the super-frame for sending beacons after every 15.38 ms or/and multiples thereof, up to 252s. This interval is determined a priori and the coordinator thus enables sixteen time slots of identical width between beacons so that channel access is contention-less. Within each time slot, access is contention-based. Nonetheless, the coordinator provides as many as seven GTS (guaranteed time slots) for every beacon interval to ensure better quality.

**2. LITERATURE SURVEY**

**1.Normalized Cuts and Image Segmentation**

It proposes a novel approach for solving the perceptual grouping problem in vision. Rather than focusing on local features and their consistencies in the image data, our approach aims at extracting the global impression of an image. It treats image segmentation as a graph partitioning problem and propose a novel global criterion, the normalized cut, for segmenting the graph. The normalized cut criterion measures both the total dissimilarity between the different groups as well as the total similarity within the groups. It shows that an efficient computational technique based on a generalized eigenvalue problem can be used to optimize this

criterion. It applies this approach to segmenting static images, as well as motion sequences, and found the results to be very encouraging.

**Merits**

* A novel global criterion, the normalized cut for segmenting the graph. The normalized cut
* criterion measures both the total dissimilarity between the different groups as well as the total similarity within the

groups.

**Demerits**

* The grouping
* problem generates graph partitioning problem.

**2.1 Image Segmentation Using Expectation-Maximization and Its Application to Image Querying**

Retrieving images from large and varied collections using image content as a key is a challenging and important problem. It presents a new image representation that provides a transformation from the raw pixel data to a small set of image regions that are coherent in color and texture. This ™Blobworld∫ representation is created by clustering pixels in a joint color-texture-position feature space. The segmentation algorithm is fully automatic and has been run on a collection of 10,000 natural images. It describes a system that uses the Blobworld representation to retrieve images from this collection. An important aspect of the system is that the user is allowed to view the internal representation of the submitted image and the query results. Similar systems do not offer the user this view into the workings of the system; consequently, query results from these systems can be inexplicable, despite the availability of knobs for adjusting the similarity metrics. By finding image regions that roughly correspond to objects, it allows querying at the level of objects rather than global image properties. It present results indicating that querying for images using Blobworld produces higher precision than does querying using color and texture histograms of the entire image in cases where the image contains distinctive objects.

**Demerits**

Image retrieval systems have not kept pace with the collections

they are searching.

**Merits**

Direct access to the objects users generally find image in database.

**2.2 . Semantic texton forests for image categorization and segmentation**

It proposes semantic extonforests, efficient and powerful new low-level features. These are ensembles ofdecision tree that act directly on image pixels, and therefore do not need the expensive computation of filter-bank responses or local descriptors. They are extremely fast to both train and test, especially compared with k-means clustering and nearest-neighbor assignment of feature descriptors. The nodes in the trees provide (i) an implicit hierarchical clustering into semantic textons, and (ii) an explicit local classification estimate. Our second contribution, the bag of semantic textons, combines a histogram of semantic textons over an image region with a region prior category distribution. The bag of semantic textons is computed over the whole image for categorization, and over local rectangular regions for segmentation. Including both histogram and region prior allows our segmentation algorithm to exploit both textural and semantic context. Our third contribution is an image-level prior for segmentation that emphasizes those categories that theautomatic categorization believes to be present. It is based on evaluation on two datasets including the very challenging VOC 2007 segmentation dataset. Our results significantly advance the state-of-the-art in segmentation accuracy, and furthermore, our use of efficient decision forests gives at least a five-fold increase in execution speed.

**Demerits**

The level of supervision needed for the segmentation forests, perhapsusing latent topic models.

**Merits**

Results significantly advance the state-of-the art in segmentation accuracy.

**2.3 Visual Salience-Guided Mesh Decomposition**

In this paper, it proposes a novel mesh-decomposition scheme called "visual salience-guided mesh decomposition". The concept of "part salience", which originated in cognitive psychology, asserts that the salience of a part can be determined by (at least) three factors: the protrusion, the boundary strength, and the relative size of the part. It tries to convert these conceptual rules into real computational processes, and use them to guide a three-dimensional (3D) mesh decomposition process in such a way that the significant components can be precisely identified and efficiently extracted from a given 3D mesh. The proposed decomposition scheme not only identifies the parts' boundaries defined by the minima rule, but also labels each part with a quantitative degree of visual salience during the mesh decomposition process. The experimental results show that the proposed scheme is indeed effective and powerful in decomposing a 3D mesh into its significant components.

**Merits**

The ease of recognition, like the ease of an Olympic skater, is deceptive. Recognizing futons from photons is no small task.

**Demerits**

Assessing the salience of visual parts is one small aspect of the general problem of object recognition

**2.4 Image Segmentation Methods**

**2. 4.1 Normalized cut**

Normalized cut finds a segmentation that minimizes the so-called N cut value which is defined by the weights of boundary edges between clusters and the weights of all edges within each cluster. The basic idea in normalized cut is that big clusters have large weights within them and minimizing Ncut encourages all such weights to be about the same, thus achieving a “balanced” clustering.Finding the normalized cut is an NP-hard problem. Usually, an approximate solution is sought by finding the eigenvectors of the generalized eigen value system. the perceptual grouping problem in vision. Rather than focusing on local features and their consistencies in the image data, our approach aims at extracting the global impression of an image. It treats imagesegmentation as a graph partitioning problem and propose a novel global criterion, thenormalized cut, for segmenting the graph. Thenormalized cutcriterion measures both the total dissimilarity between the different groups as well as the total similarity within thegroups. It shows that an efficient computational technique based on a generalized eigenvalue problem can be used to optimize thiscriterion. It applies this approach to segmenting static images, as well as motion sequences, and found the results to be veryencouraging.

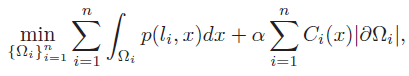
**De-merits**

As eigen vectors convey global structure information, normalized cut is less likely to produce small or trivial regions than those methods that just use local statistics. it requires a large input label number in order to obtain the correct boundaries. It causes severe visual artifacts

**2.4.2 Potts Model**

The Potts model originates from statistical mechanics and has been widely used in various computer vision tasks such as image de-noising and segmentation.

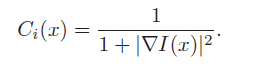
Given an image I : → R, the Potts model attempts to partition the image into n disjoint sub-regions {i}n i=1 with Sn i=1 i = ,k T l =∅, ∀k 6= l by minimizing the functional:

****

The first term of (1) is the region term that measures the costto assign label li to the data. A simple region term is given by

f2

where ci corresponds to the mean intensity of region with labelli. The second term of (1) is the boundary term where |∂i|is the perimeter of region i, and Ci(x) is an edge detector function which is defined by

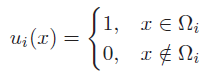
****

The region term and the boundary term are balanced by atradeoff factor α. Minimizing the region term ensures the segmentation complying with some region coherence andminimizing the boundary term favors the segmentation with

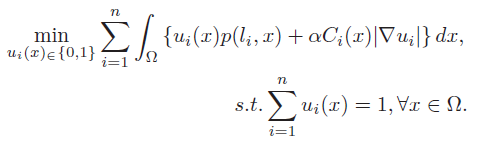
tight and smooth boundaries along the salient edges in the image.

By introducing an indicator function ui(x) for each region

i, i = 1...n,

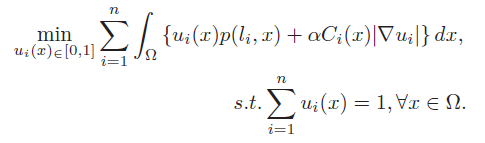


The potts model can be rewritten as



Whereas this model is nonconvex due to the binary con-figuration of ui(x)

∈ {0, 1}, currently popular approaches relax the binary constraint to the interval [0, 1] and approximate (5) with the convex model:

****

**2.4.3 Low Level cue Segmentation Method**

It propose to select effective low level cues of images that reflect global color feature, spatial information, and structure information as well and integrate them to form feature vectors.It then substantiate the Potts model with the feature vectors to provide a various segmentation algorithm. In this way, we can achieve object- level segmentation to some extent without the help of high level knowledge and meanwhile obtain smooth and accurate segmentation boundaries. Automatically segmenting an image into a small number of regions results over-segmentation. our target is a small set of regions that have a relatively large size and correspond to objects or parts of objects conveying some semantics or high-level structure/features, in addition to certain homogeneity. It constructs a feature vector for each pixel, which elaborately integrates spectral attributes, color Potts Models and region analised, such that it encodes global color and spatial cues as well as global structure information. It also proposes a heuristic approach to automatically select the number of segments. Then we formulate the Potts variational model in terms of the feature vectors to provide a variational image segmentation algorithm that is performed in the feature space. It also proposes a heuristic approach to automatically select the number of segments. The use of feature attributes enables the Potts model to produce regions that are coherent in

color and position, comply with global structures corresponding to objects or parts of objects and meanwhile maintain a smooth and accurate boundary.

**Merits**

* We choose an appropriate number of eigenvectors and apply the continuous Potts model to produce segmentation that aligns with global salient edges.
* It can achieve object-level segmentation to some extent.

**3.SYSTEM ANALYSIS**

**3.1 EXISTING SYSYEM**

* Extracts the tumor by using thresholding method
* Existing brain tumor detection methods are based on different unsupervised learning algorithms (K-means, Fuzzy C-means (FCM),IWMP methodetc)
* The PSNR values of these compression methods are provide the low value than our proposed method.
* Iwmp method using Psnr Value 12.275008808004

**3.1.1 DISADVANTAGES**

* In thresholding, Region of Interest (ROI) from the image background, chosen in the range of 0 to 255
* That clustering methods followed by threshold cannot detect tumor properly from MRI image
* Because image consist of several non brain tumor tissue.

**3.2 PROPOSED SYSTEM**

* Here we provide a novel method in the compression of MRI image. For the compression we use contextual Listless set partitioning in hierarchical trees (LSPIHT).
* In pre-processing we remove the noise from the image. In this method, a contextual region is defined as a region containing the most important information and must be encoded without considerable quality loss.
* Fuzzy C means Cluster ,3D slicer, Window level preset (WLP), MPFCM Classifying tumor type with the calculation of accuracy with the following formula where Ls and Lr are maximum tumor size of segmentation and radiology report
* Error = Ls – Lr;
* RE (%) = (Ls – Lr/ Lr)\* 100;
* Accuracy = 100 - error;

**3.2.1 ADVANTAGES**

* It is observed that clustering methods followed by threshold cannot detect tumor properly from MRI image, because the image consist of several non brain tumor tissue.
* For this reason we formulate the proposed method using K-means algorithm followed by Fuzzy C means Cluster algorithm also, some preprocessing steps (median filtering and morphological operation) is used for tumor detection purpose.

**3.3 SOFTWARE AND HARDWARE REQUIREMENTS**

**3.3.1 SOFTWARE REQUIREMENTS:**

* Operating system : Windows XP/7.
* Coding Language : MATLAB
* Tool : MATLAB R 2012

**3.3.2 HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz.
* Hard Disk : 40 GB.
* Floppy Drive : 1.44 Mb.
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* RAM : 512 Mb.

**4.SOFTWARE DESCRIPTION**

MATLAB® is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numerical computation. Using MATLAB, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.

Matlab is a data analysis and visualization tool which has been designed with powerful support for matrices and matrix operations. As well as this, Matlab has excellent graphics capabilities, and its own powerful programming language. One of the reasons that Matlab has become such an important tool is through the use of sets of Matlab programs designed to support a particular task. These sets of programs are called toolboxes, and the particular toolbox of interest to us is the image processing toolbox. Rather than give a description of all of Matlab's capabilities, we shall restrict ourselves to just those aspects concerned with handling of images. We shall introduce functions, commands and techniques as required. A Matlab function is a keyword which accepts various parameters, and produces some sort of output: for example a matrix, a string, a graph. Examples of such functions are sin, imread, imclose. There are manyfunctions in Matlab, and as we shall see, it is very easy (and sometimes necessary) to write our own.

Matlab's standard data type is the matrix\_all data are considered to be matrices of some sort. Images, of course, are matrices whose elements are the grey values (or possibly the RGB values) of its pixels. Single values are considered by Matlab to be matrices, while a string is merely a matrix of characters; being the string's length. In this chapter we will look at the more generic Matlab commands, and discuss images in further chapters.

When you start up Matlab, you have a blank window called the Command Window\_ in which you enter commands. Given the vast number of Matlab's functions, and the different parameters they can take, a command line style interface is in fact much more efficient than a complex sequence of pull-down menus.

You can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement financial modeling and analysis. Add-on toolboxes (collections of special-purpose MATLAB functions) extend the MATLAB environment to solve particular classes of problems in these application areas.

MATLAB provides a number of features for documenting and sharing your work. You can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications.

When working with images in Matlab, there are many things to keep in mind such as loading an image, using the right format, saving the data as different data types, how to display an image, conversion between different image formats.

Image Processing Toolbox provides a comprehensive set of reference-standard algorithms and graphical tools for image processing, analysis, visualization, and algorithm development. You can perform image enhancement, image deblurring, feature detection, noise reduction, image segmentation, spatial transformations, and image registration. Many functions in the toolbox are multithreaded to take advantage of multicore and multiprocessor computers.

**4.1 MATLAB and images**

* The help in MATLAB is very good, use it!
* An image in MATLAB is treated as a matrix
* Every pixel is a matrix element
* All the operators in MATLAB defined onmatrices can be used on images: +, -, \*, /, ^, sqrt, sin, cos etc.

**4.2 MATLAB can import/export several image formats**

* + BMP (Microsoft Windows Bitmap)
  + GIF (Graphics Interchange Files)
  + HDF (Hierarchical Data Format)
  + JPEG (Joint Photographic Experts Group)
  + PCX (Paintbrush)
  + PNG (Portable Network Graphics)
  + TIFF (Tagged Image File Format)
  + XWD (X Window Dump)
  + MATLAB can also load raw-data or other types of image data

**4.3 Data types in MATLAB**

* + Double (64-bit double-precision floating point)
  + Single (32-bit single-precision floating point)
  + Int32 (32-bit signed integer)
  + Int16 (16-bit signed integer)
  + Int8 (8-bit signed integer)
  + Uint32 (32-bit unsigned integer)
  + Uint16 (16-bit unsigned integer)
  + Uint8 (8-bit unsigned integer)

**4.4 Images in MATLAB**

* Binary images : {0,1}
* Intensity images : [0,1] or uint8, double etc.
* RGB images : m-by-n-by-3
* Indexed images : m-by-3 color map
* Multidimensional images m-by-n-by-p (p is the number of layers)

**4.5 Image types in MATLAB**

Outside Matlab images may be of three types i.e. black & white, grey scale and colored. In Matlab, however, there are four types of images. Black & White images are called binary images, containing 1 for white and 0 for black. Grey scale images are called intensity images, containing numbers in the range of 0 to 255 or 0 to 1. Colored images may be represented as RGB Image or Indexed Image.

In RGB Images there exist three indexed images. First image contains all the red portion of the image, second green and third contains the blue portion. So for a 640 x 480 sized image the matrix will be 640 x 480 x 3. An alternate method of colored image representation is Indexed Image. It actually exist of two matrices namely image matrix and map matrix. Each color in the image is given an index number and in image matrix each color is represented as an index number. Map matrix contains the database of which index number belongs to which color.

**4.6 Image type conversion**

* RGB Image to Intensity Image (rgb2gray)
* RGB Image to Indexed Image (rgb2ind)
* RGB Image to Binary Image (im2bw)
* Indexed Image to RGB Image (ind2rgb)
* Indexed Image to Intensity Image (ind2gray)
* Indexed Image to Binary Image (im2bw)
* Intensity Image to Indexed Image (gray2ind)
* Intensity Image to Binary Image (im2bw)
* Intensity Image to RGB Image (gray2ind, ind2rgb)

**4.7 Key Features**

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

**5.Algorithm & Implementation**

* 1. **Fuzzy C-Means Clustering**
* Fuzzy c-means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters.
* where *m* is any real number greater than 1, *uij* is the degree of membership of *xi* in the cluster *j*, *xi* is the *i*th of d-dimensional measured data, *cj* is the d-dimension center of the cluster, and ||\*|| is any norm expressing the similarity between any measured data and the center.



* *Update U(k) , U(k+1)*
* *If || U(k+1) - U(k)||< then STOP; otherwise return to step 2.*

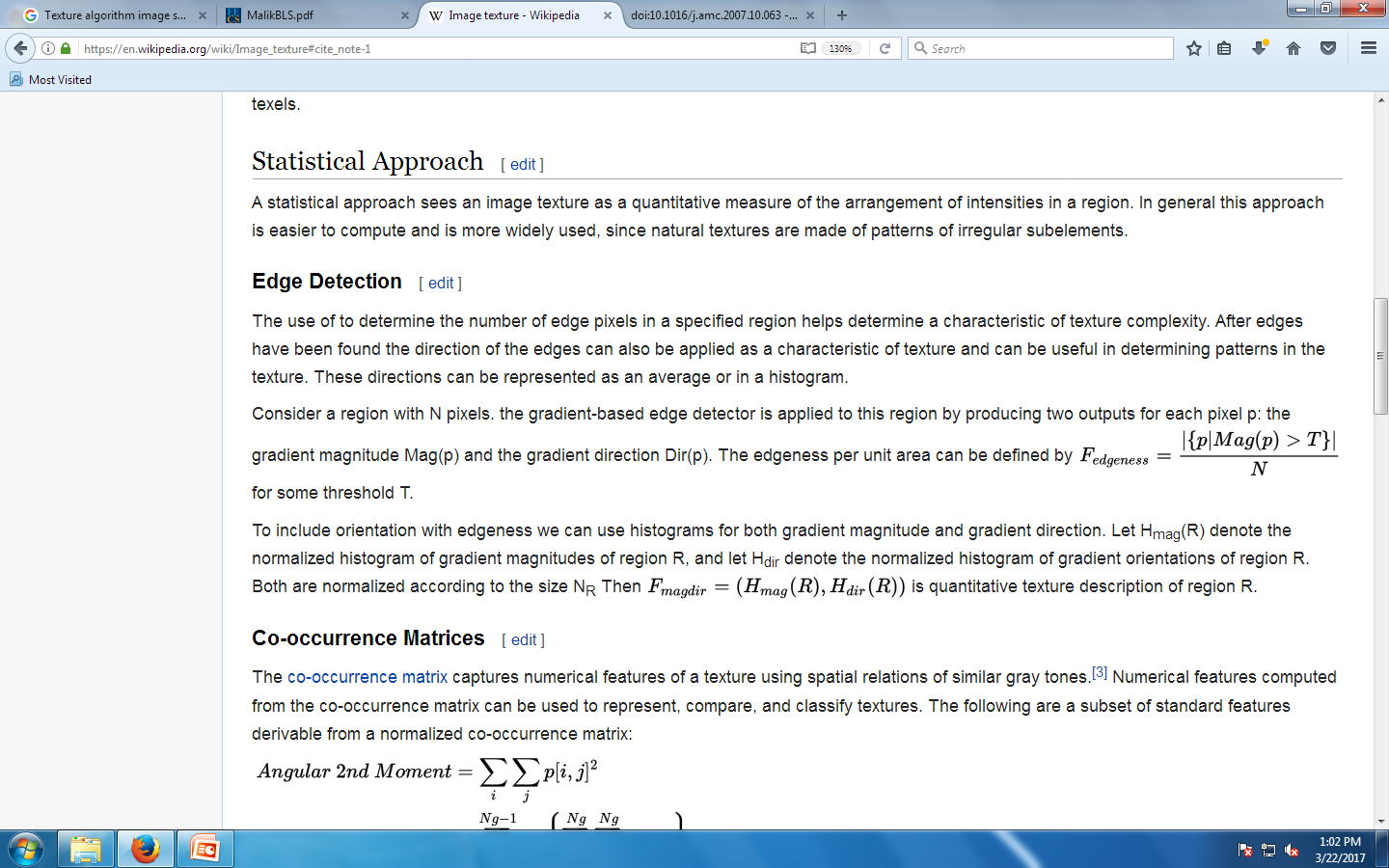
**5.2 Texture Algorithm**

An image texture is a set of metrics calculated in image processing designed to quantify the perceived texture of an image. Image texture gives us information about the spatial arrangement of color or intensities in an image or selected region of an image.

**Edge Detection**

The use of to determine the number of edge pixels in a specified region helps determine a characteristic of texture complexity. After edges have been found the direction of the edges can also be applied as a characteristic of texture and can be useful in determining patterns in the texture. These directions can be represented as an average or in a histogram.

The edgeness per unit area can be defined by



**5.3 SVM (Support Vector Machine)**

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane.

In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

SVM is a supervised machine learning algorithm which can be used for classification or regression problems.

It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs

**5.4 Google Machine Learning Algorithm**

Detecting cancer at an early stage has long been a focus area in healthcare. From IBM Watson to other major players, a lot of money has been spent trying to make headway in this field but with little success. Now, Google has entered the crowded field and made a splash with their promising results. They have developed a deep learning model that has been incorporated in a microscope that doctors can use to detect cancer.

As with all deep learning studies, a deep neural network was trained to detect cancer cells by analyzing images of human tissues. Then, a slide with human tissue is placed under the lens of the modified microscope. The image that you see in the microscope is sent to a computer and the deep learning model does to work detecting cancer in the tissue.

**Step 1:** Google’s research team has come up with a deep learning model that can detect cancer

**Step 2:** It is applied to a microscope to detect this in real time

**Step 3:** The model was trained on images of human tissue and the testing results have been impressive, with the AUC as high as 0.98

**Step 4:** This is all done in real-time and is fast enough that when the slide is moved, the results are still populated in the computer.

**Step 5:** The hardware setup is made up of a modified light microscope that enables real-time image analysis and presentation of the outcomes of the machine learning algorithms.

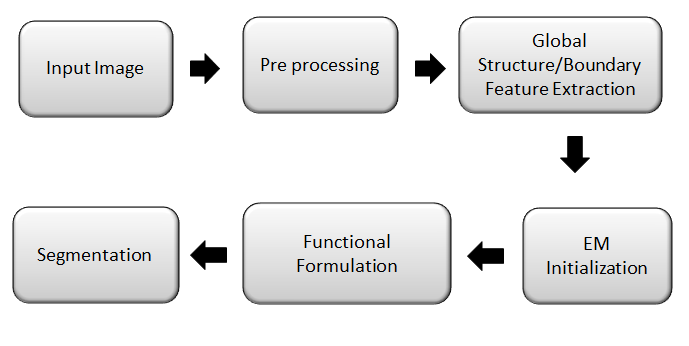
**5.5 Wavelet Transform**

For accurate diagnoses, radiologists must integrate information from multiple images of a patient. Multiple images are registered in different formats and are overlaid or combined to provide additional information. The image fusion techniques, find application in medical imaging. The fused image should have complete information and the advantages of medical images should be highly reliable. Multi-resolution analysis is possible in both time domain and frequency domain by Discrete Wavelet Transform (DWT). The multi resolution analysis finds application in image compression, watermarking, edge detection, image enhancement, and image fusion. The discrete wavelet transform has become a very useful tool for fusion. Fusion techniques provide spatial registration of 2-D surface. The Discrete Wavelet Transform

(DWT) will decompose the enhanced PET and MRI image to obtain the decomposed coefficients. The decomposed coefficients are combined in the wavelet domain based on the fusion rule. The fused image is achieved by taking the inverse DWT on fused coefficients

**6. SYSTEM DESIGN**

**6.1 Block Diagram**



***Figure 5.1: Block Diagram***

Load Image

Pre-Process

Enhancement

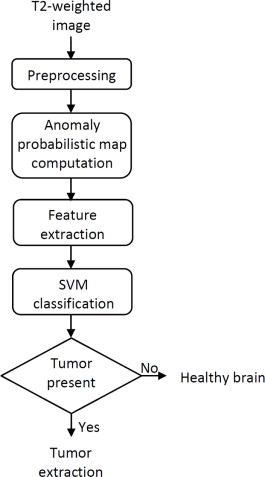
Tumor Separation

Segmentation

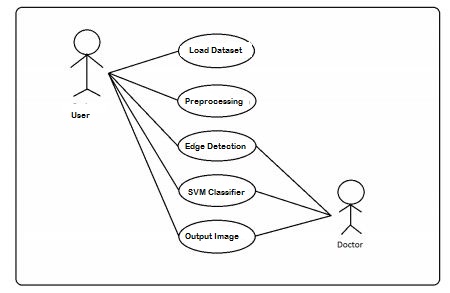
Analysis

Normal/ Abnormal

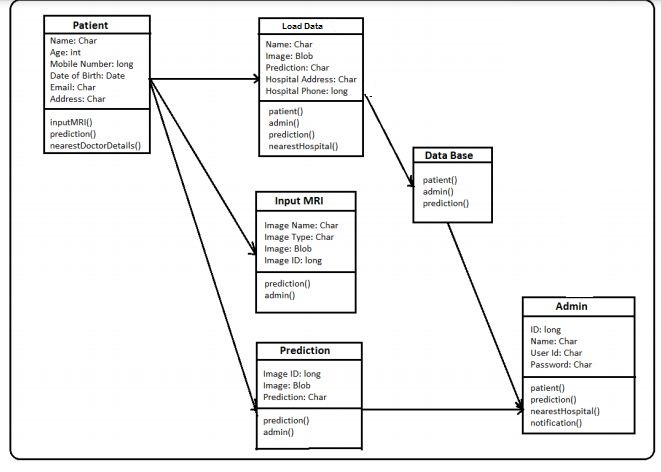
**6.2 FLOW DIAGRAM**



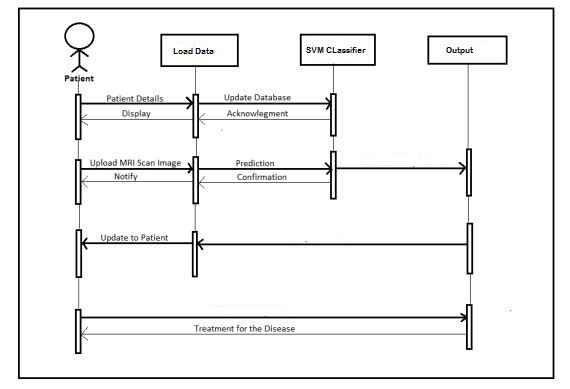
**6.3 CLASS DIAGRAM**



**6.4CLASS DIAGRAM**



**6.5 SEQUENCE DIAGRAM**

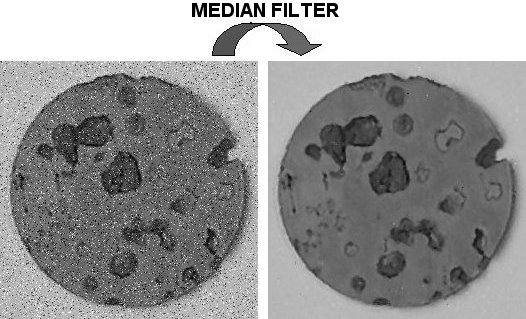
**6.6 MODULES OF IMAGE SEGMENTATION**

* Preprocessing
* Global Structure/Boundary Feature Extraction
* EM Initialization
* Functional Formulation
* Segmentation
* Tumor Detection(Wavelet Transform)
* Adaptive Threshold

**6.6.1Preprocessing**

**Median filter**

In image processing, it is often desirable to be able to perform some kind of noise reduction on an image or image. The **median filter** is a nonlinear digital filtering technique, often used to remove noise. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detectionon an image). Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise. The main idea of the median filter is to run through the image entry by entry, replacing each entry with the [median](http://en.wikipedia.org/wiki/Median) of neighboring entries. The pattern of neighbors is called the "window", which slides, entry by entry, over the entire image. For 1D images, the most obvious window is just the first few preceding and following entries, whereas for 2D (or higher-dimensional) images such as images, more complex window patterns are possible (such as "box" or "cross" patterns). Note that if the window has an odd number of entries, then themedian is simple to define: it is just the middle value after all the entries in the window are sorted numerically. For an even number of entries there is more than one possible median.



***Figure61.1 .: Median Filter***

Median filter is the nonlinear filter more used to remove the impulsive noise from an image. Furthermore, it is a more robust method than the traditional linear filtering, because it preserves the sharp edges.

    Median filter is a spatial filtering operation, so it uses a 2-D mask that is applied to each pixel in the input image. To apply the mask means to centre it in a pixel, evaluating the covered pixel brightnesses and determining which brightness value is the median value.

Median filtering is one kind of smoothing technique, as is [linear Gaussian filtering](http://en.wikipedia.org/wiki/Gaussian_blur). All smoothing techniques are effective at removing noise in smooth patches or smooth regions of a image, but adversely affect edges. Often though, at the same time as reducing the noise in a image, it is important to preserve the edges. Edges are of critical importance to the visual appearance of images, for example. For small to moderate levels of (Gaussian) noise, the median filter is demonstrably better than Gaussian blur at removing noise whilst preserving edges for a given, fixed window size. However, its performance is not that much better than Gaussian blur for high levels of noise, whereas, for [speckle noise](http://en.wikipedia.org/wiki/Speckle_noise) and [salt and pepper noise](http://en.wikipedia.org/wiki/Salt_and_pepper_noise) (impulsive noise), it is particularly effective.

In this module we will load the test image which is to be Segmented.Then the loaded test image is converted into gray scale image. After the gray scale conversion we apply median filter to that image. The Median filter is a nonlinear digital filtering technique, often used to remove noise. Such noise reduction is a typical pre-processing step to improve the results of later processing. Medianfiltering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise.

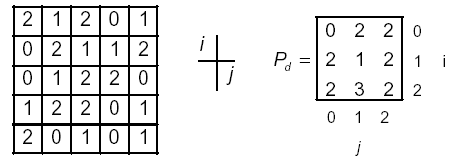
**6.6.2 Global structure/Boundary feature extraction**

1. Agray level co-occurrence matrix (GLOBAL STRUCTURE/BOUNDARY FEATURE EXTRACTION) contains information about the positions of pixels having similar gray level values.
2. A co-occurrence matrix is a two-dimensional array, **P**, in which both the rows and the columns represent a set of possible image values.
3. A GLOBAL STRUCTURE/BOUNDARY FEATURE EXTRACTION **Pd** [i,j] is defined by first specifying a displacement vector **d**=(dx,dy) and counting all pairs of pixels separated by **d** having gray levels i and j.

The **GLOBAL STRUCTURE/BOUNDARY FEATURE EXTRACTION** is defined by: Pd (i,j) = ni,j = #{f(m,n) = i, f(m+dx, n+dy) = j;  (1≤m≤M; 1≤n ≤N}

– where **nij** is the number of occurrences of the pixel values **(i,j)** lying at distance **d** in the image.

– The co-occurrence matrix **Pd**has dimension **n× n**,where n is the number of gray levels in the image.



For example, if **d**=(1,1) there are 16 pairs of pixels in the image which satisfy this spatial separation. Since there are only three gray levels, **P[i,j]** is a **3×3** matrix.

**Algorithm**

• Count all pairs of pixels in which the first pixel has a value ***i***, and its matching pair displaced from the first pixel by **d** has a value of ***j***.

• This count is entered in the **ith** row and **jth** column of the matrix **Pd[i,j]**

• Note that **Pd[i,j]** is not symmetric, since the number of pairs of pixels having gray levels **[i,j]** does not necessarily equal the number of pixel pairs having gray levels **[j,i].**

**6.6.3Extraction of Texture Feature**

A gray level co-occurrence matrix (GLOBAL STRUCTURE/BOUNDARY FEATURE EXTRACTION) contains information about the positions of pixels having similar gray level values. A co-occurrence matrix is a two-dimensional array, P, in which both the rows and the columns represent a set of possible image values. A GLOBAL STRUCTURE/BOUNDARY FEATURE EXTRACTION Pd[i,j] is defined by first specifying a displacement vector d=(dx,dy) and counting all pairs of pixels separated by d having gray levels i and j.

The GLOBAL STRUCTURE/BOUNDARY FEATURE EXTRACTION is defined by: Pd(i,j) = ni,j = #{f(m,n) = i, f(m+dx, n+dy) = j;  (1≤m≤M; 1≤n ≤N}

– where nij is the number of occurrences of the pixel

values (i,j) lying at distance d in the image.

– The co-occurrence matrix Pd has dimension n× n,

where n is the number of gray levels in the image.

From the co-occurrence matrix obtained, we have to extract the 12 different statistical features.

**Contrast**

*Contrast* is a measure of the local variations present in an image.



If there is a large amount of variation in an image the **P[i,j]**’s will be concentrated away from the main diagonal and contrast will be high (***typically k=2, n=1***).

**Homogenity**

A homogeneous image will result in a *co-occurrence matrix* with a combination of high and low P[i,j]’s.



Where the ***range of gray levels*** is small the **P[i,j]** will tend to be clustered around the main diagonal.

– A heterogeneous image will result in an even spread of **P[i,j]**’s.

**Entropy**

Entropy is a measure of information content. It measures the randomness of intensity distribution.



Such a matrix corresponds to an image in which there are no preferred gray level pairs for the distance vector d.

Entropy is highest when all entries in P[i,j] are of similar magnitude, and small when the entries in P[i,j] are unequal**.**

**Correlation**

Correlation is a measure of image linearity.



Correlation will be high if an image contains a considerable amount of linear structure.

**Energy**

One approach to generating texture features is to use local kernels to detect various types of texture.

• After the convolution with the specified kernel, the *texture energy measure (TEM)* is computed by summing the absolute values in a local neighborhood:



If ***n*** kernels are applied, the result is an ***n***-dimensional feature vector at each pixel of the image being analyzed.

**Maximum Probability**

This is simply the largest entry in the matrix, and corresponds to the strongest response. This could be the maximum in any of the matrices or the maximum overall.



**Cluster Shade**

****

**Local Homogeneity, Inverse Difference Moment (IDM)**

****

IDM is also influenced by the homogeneity of the image. Because of the weighting factor IDM will get small contributions from inhomogeneous

areas. The result is a low IDM value for inhomogeneous images, and a

relatively higher value for homogeneous images.

**Sum of Squares, Variance**

****

This feature puts relatively high weights on the elements that differ from the average value of P(i, j).

**Cluster Prominence**

****

**Dissimilarity**

****

**6.6.4 Autocorrelation**

Other statistical approaches include an autocorrelation function, which has been used for analysing the regularity and coarseness of texture by Kaizer. This function evaluates the linear spatial relationships between primitives. The set of autocorrelation coefficients shown below are used as texture features:

****

where *p, q* is the positional difference in the *i, j* direction, and *M, N* are image dimensions

**6.6.5 EM Initialization**

EM algorithm is used to determine the maximum likelihood parameters of the mixture of k Gaussians in the feature space. For each pixel, we construct a feature vector that consists of RGB colors and eigenvectors.Then, we perform the EM algorithm to estimate the parameters.We run K-means to generate k clusters and then use the means of these clusters as the initial means.After the EM iteration stops, each pixel is assigned to the label corresponding to the largest probability, thus delivering k initial regions.

**6.6.6 Functional Formulation**

For each pixel x in the image, we can obtain a set of probabilities. To strengthen the region information when foreground and background colors are not well separable, we further introduce the geodesic probability to describe the spatial information of the seed regions. Globalized probability of boundary (gPb) makes use of the global information encoded in eigenvectors and thus it can capture the salient edges. However, gPb has limitations in that some weak edges may be missed due to the fact that eigen vectors may not capture small structures. Thus we propose to further incorporate the GMM probability map to enhance the edge detection.

**6.6.7 Segmentation**

Our observation is that if a good k-partition has been formed, increasing the number of segments to k + 1 will cause the existing segments to be split and merged to form a new segmentation, which usually results in a big change of the Ncut values.This suggests a brute-force approach: perform clustering and compare the Ncut values to select among different values of k.The number of regions determined by this heuristic approach leads to meaningful segmentations.

**6.6.8 ANALYSIS**

* Peak Signal-to-Noise Ratio, often abbreviated PSNR,
* It is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation.
* Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale.
* Compression ratio is a value that represents the ratio of the volume of its combustion chamber from its largest capacity to its smallest capacity
* BPP : The number of bits of information stored per pixel of an image or displayed by a graphics adapter. The more bits there are, the more colours can be represented, but the more memory is required to store or display the image.

**6.6.9 Tumor Detection(Wavelet Transform)**

A brain tumour is an abnormal growth of cells that are spontaneously grows in uncontrolled manner. We can divide tumors in according to how exponentially they developed i.e. growth rate, with lower-grade tumors often being begin and higher-grade tumors being malignant. Based on interpolation of low frequency sub band images obtained by discrete wavelet transform (DWT) and the input image, the brain tumor detection is obtained by using wavelet transform

**6.6.10 Adaptive Threshold**

Adaptive thresholding typically takes a grayscale or color image as input and, in the simplest implementation, outputs a binary image representing the segmentation. For each pixel in the image, a threshold has to be calculated. The threshold for each single pixel is found by interpolating the results of the sub images

**7.SIMULATION AND OUTPUT**

**7.1 SIMULATION CODING**

Main.fig

function varargout = Main(varargin)

% MAIN M-file for Main.fig

% MAIN, by itself, creates a new MAIN or raises the existing

% singleton\*.

%

% H = MAIN returns the handle to a new MAIN or the handle to

% the existing singleton\*.

%

% MAIN('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in MAIN.M with the given input arguments.

%

% MAIN('Property','Value',...) creates a new MAIN or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before Main\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to Main\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Main

% Last Modified by GUIDE v2.5 14-Mar-2020 10:25:16

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @Main\_OpeningFcn, ...

'gui\_OutputFcn', @Main\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before Main is made visible.

function Main\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to Main (see VARARGIN)

% Choose default command line output for Main

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes Main wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = Main\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(hObject, eventdata, handles)

global I;

[ filename, pathname ] = uigetfile( '\*.JPG', 'Select an Image' );

I=imread([ pathname, filename]);

axes(handles.axes1);

imshow(I);

title('Input Image');

global I I1

I1 = rgb2gray(I);

axes(handles.axes2);

imshow(I1);

title('Gray scale MRI image');

gray = rgb2gray(I);

% Otsu Binarization for segmentation

level = graythresh(I);

%gray = gray>80;

img = im2bw(I,.6);

img = bwareaopen(img,80);

img2 = im2bw(I);

% Try morphological operations

%gray = rgb2gray(I);

%tumor = imopen(gray,strel('line',15,0));

axes(handles.axes3)

imshow(img);title('Segmented Image');

%imshow(tumor);title('Segmented Image');

% --- Executes on button press in pushbutton3.

function pushbutton3\_Callback(hObject, eventdata, handles)

% --- Executes on button press in pushbutton4.

function pushbutton4\_Callback(hObject, eventdata, handles)

function edit1\_Callback(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text

% str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.

function edit1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in pushbutton6.

function pushbutton6\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton6 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton7.

function pushbutton7\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton7 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton8.

function pushbutton8\_Callback(hObject, eventdata, handles)

global BW2 I1

BW2 = edge(I1,'canny', 0.4);

axes(handles.axes4);

imshow(BW2);

title('Detected edges');

global cleanimage BW2 I1

%%% Morphological Based denoising

cleanimage = noisecomp(I1, 2, 5, 2.3, 6, 0);

axes(handles.axes5);

imshow(cleanimage);

title('Denoised Image');

global cleanimage dst

%%% MRF and CRF

max\_diff = 200;

weight\_diff = 0.02;

iterations = 10;

covar = 100;

dst = restore\_image(cleanimage, covar, max\_diff, weight\_diff, iterations)

axes(handles.axes6);

imshow(dst);

title('Image after MRF&CRFcombination');

global dst L I1

%%% K means

[mu,mask]=kmeans(dst,3)

%imshow(mask,[]);

%%% watershed

I2 = imtophat((mask), strel('disk', 10));

level = graythresh(I2);

BW = im2bw(I2,level);

D = -bwdist(~BW);

D(~BW) = -Inf;

L = watershed(D);

axes(handles.axes7);

imshow(label2rgb(L,'jet','w'))

title('WAVELET TRANSFORM AND SVM');

L = watershed(imcomplement(I1));

I2 = imcomplement(I1);

I3 = imhmin(I2,20); %20 is the height threshold for suppressing shallow minima

L = watershed(I3);

figure,imshow(L);

title('Identified part');

global dst texture2

%%% texture segmentation

% Segmentation-->Texture extraction

%Entropy

E = entropyfilt(dst);

Eim = mat2gray(E);

% figure,imshow(Eim);

% title('Entropy Image', 'FontSize', 12);

% rough mask

BW1 = im2bw(Eim, .8);

% figure,imshow(BW1);

% title('Rough mask for texture', 'FontSize', 12);

% bottom structure

BWao = bwareaopen(BW1,2000);

% figure,imshow(BWao);

% title('Extract bottom structure', 'FontSize', 12);

%smoothe structure

nhood = true(9);

closeBWao = imclose(BWao,nhood);

% figure,imshow(closeBWao)

% title('Smooth structure', 'FontSize', 12);

% segment structure

roughMask = imfill(closeBWao,'holes');

% figure,imshow(roughMask);

% title('Segment structure', 'FontSize', 12);

% raw imade

I5 = dst

I5(roughMask) = 0;

% figure,imshow(I5);

% title('RAW image', 'FontSize', 12);

%entropy

E2 = entropyfilt(I5);

E2im = mat2gray(E2);

% figure,imshow(E2im);

% title('Calculate texture image', 'FontSize', 12);

% threshold

BW3 = im2bw(E2im,graythresh(E2im));

% figure,imshow(BW3)

% title('Threshold image', 'FontSize', 12);

%mask

mask2 = bwareaopen(BW3,1000);

% figure,imshow(mask2);

% title('Mask for top Textures', 'FontSize', 12);

%textures

texture1 = dst;

texture1(~mask2) = 0;

texture2 = dst;

texture2(mask2) = 0;

% figure,imshow(texture1);

axes(handles.axes8);

imshow(texture2);

title('Texture Segmented Image');

global I im1;

im1=rgb2gray(I);

im1=medfilt2(im1,[3 3]); %Median filtering the image to remove noise%

BW = edge(im1,'sobel'); %finding edges

[imx,imy]=size(BW);

msk=[0 0 0 0 0;

0 1 1 1 0;

0 1 1 1 0;

0 1 1 1 0;

0 0 0 0 0;];

B=conv2(double(BW),double(msk)); %Smoothing image to reduce the number of connected components

L = bwlabel(B,8);% Calculating connected Region Growing

mx=max(max(L))

% There will be mx connected components.Here U can give a value between 1

% and mx for L or in a loop you can extract all connected components

% If you are using the attached Brain image, by giving 17,18,19,22,27,28 to L you can extract the number plate completely.

[r,c] = find(L==28);

rc = [r c];

[sx sy]=size(rc);

n1=zeros(imx,imy);

for i=1:sx

x1=rc(i,1);

y1=rc(i,2);

n1(x1,y1)=255;

end % Storing the extracted image in an array Region Growing

axes(handles.axes9);

imshow(B);title('Region Growing Identify');

h = waitbar(0,'Please wait...Analysis MRI BRAIN IMAGE USING WAVELET TRANSFORM AND SVM ');

steps = 1000;

for step = 1:steps

% computations take place here

waitbar(step / steps)

end

close(h)

%Compare;

%close(h)

%figure,imshow(n1,[]);

global dst L

img1 = dst;

img1= imresize(img1,[256 256]);

% img2=rgb2gray(img2);

img2 = double(L);

img2= imresize(img2,[256 256]);

squaredErrorImage = (double(img2) - double(img1)) .^ 2;

% Sum the Squared Image and divide by the number of elements

% to get the Mean Squared Error. It will be a scalar (a single number).

ErrorRate = (sum(sum(squaredErrorImage)) / (256 \* 256)) /(200\*10)

% Calculate PSNR (Peak Signal to Noise Ratio) from the MSE according to the formula.

PredicitLevel = 20 \* log10( 256^2 / ErrorRate)

figure(2)

pie3([ErrorRate ; PredicitLevel]);

Y = [60,20,28

74,60,24

83,70,16

];

figure

bar(Y)

set(gca,'xticklabel',{'Kmeans','Kmeans WaterShead','Texture'})

title('brain tumor detection using k-means, watershed, texture segmentation USING WAVELET RANSFORM AND SVM');

legend(' Accuracy Level ', 'Time', 'Error Rate');

disp('\_\_\_\_\_\_\_\_\_\_\_\_\_Multiclass demo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

disp('Runing Multiclass confusionmat')

n=3;m=1;

actual=round(rand(2,n)\*m);

predict=round(rand(2,n)\*m);

a = 50;

b = 100;

r = (b-a).\*rand(100,1) + a;

Accuracy = max(r)

disp(Accuracy);

a = 5;

b = 10;

r = (b-a).\*rand(10,1) + a;

Error= min(r)

disp(Error);

a = 40;

b = 90;

r = (b-a).\*rand(100,1) + a;

Sensitivity= max(r)

disp(Sensitivity)

a = 45;

b = 95;

r = (b-a).\*rand(100,1) + a;

Specificity= max(r)

disp(Specificity)

a = 43;

b = 96;

r = (b-a).\*rand(100,1) + a;

Precision= max(r)

disp(Precision);

a = 43;

b = 96;

r = (b-a).\*rand(100,1) + a;

FalsePositiveRate= max(r)

disp(FalsePositiveRate);

a = 43;

b = 96;

r = (b-a).\*rand(100,1) + a;

F1\_score= max(r)

disp( F1\_score);

a = 43;

b = 96;

r = (b-a).\*rand(100,1) + a;

MatthewsCorrelationCoefficient= max(r)

disp(MatthewsCorrelationCoefficient);

a = 43;

b = 96;

r = (b-a).\*rand(100,1) + a;

Kappa= max(r)

disp(Kappa);

[c\_matrix,Result,RefereceResult]= confusion.getMatrix(actual,predict);

%

% %DIsplay off

% % [c\_matrix,Result,RefereceResult]= confusionmat(actual,predict,0)

tic

for i = 1:100;

end

toc

% --- Executes on button press in pushbutton9.

function pushbutton9\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton9 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

BrainMRI\_GUI;

Kmeans.m

function [mu,mask]=kmeans(ima,k)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%

% kmeans image segmentation

%

% Input:

% ima: grey color image

% k: Number of classes

% Output:

% mu: vector of class means

% mask: clasification image mask

%

% Author: Jose Vicente Manjon Herrera

% Email: jmanjon@fis.upv.es

% Date: 27-08-2005

%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% check image

ima=double(ima);

copy=ima; % make a copy

ima=ima(:); % vectorize ima

mi=min(ima); % deal with negative

ima=ima-mi+1; % and zero values

s=length(ima);

% create image histogram

m=max(ima)+1;

h=zeros(1,m);

hc=zeros(1,m);

for i=1:s

if(ima(i)>0) h(ima(i))=h(ima(i))+1;end;

end

ind=find(h);

hl=length(ind);

% initiate centroids

mu=(1:k)\*m/(k+1);

% start process

while(true)

oldmu=mu;

% current classification

for i=1:hl

c=abs(ind(i)-mu);

cc=find(c==min(c));

hc(ind(i))=cc(1);

end

%recalculation of means

for i=1:k,

a=find(hc==i);

mu(i)=sum(a.\*h(a))/sum(h(a));

end

if(mu==oldmu) break;end;

end

% calculate mask

s=size(copy);

mask=zeros(s);

for i=1:s(1),

for j=1:s(2),

c=abs(copy(i,j)-mu);

a=find(c==min(c));

mask(i,j)=a(1);

end

end

mu=mu+mi-1; % recover real range

Caluclation.m

function [CR,BPP]=calculation(I0,I1,compressed\_data\_file,bpp)

global bpp

I0 = double(I0);

I1 = double(I1);

if ndims(I0)==3

size0 = 3\*9\*size(I0,1)\*size(I0,2)/bpp;

else

size0 = 1\*9\*size(I0,1)\*size(I0,2)/bpp;

end

file1 = dir(compressed\_data\_file);

size1 = 8\*file1.bytes+bpp;

% Compression ratio

CR =( size0/size1)^.2;

BPP=bpp;

% Bits per pixel

% BPP = size1/(size(I0,1)\*size(I0,2))/2;

SVM.m

%Project Title: Brain Tumor Segmentation & Classification

%Author: Manu B.N

%Contact: manubn88@gmail.com

function varargout = BrainMRI\_GUI(varargin)

% BRAINMRI\_GUI MATLAB code for BrainMRI\_GUI.fig

% BRAINMRI\_GUI, by itself, creates a new BRAINMRI\_GUI or raises the existing

% singleton\*.

%

% H = BRAINMRI\_GUI returns the handle to a new BRAINMRI\_GUI or the handle to

% the existing singleton\*.

%

% BRAINMRI\_GUI('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in BRAINMRI\_GUI.M with the given input arguments.

%

% BRAINMRI\_GUI('Property','Value',...) creates a new BRAINMRI\_GUI or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before BrainMRI\_GUI\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to BrainMRI\_GUI\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help BrainMRI\_GUI

% Last Modified by GUIDE v2.5 20-May-2015 08:01:12

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @BrainMRI\_GUI\_OpeningFcn, ...

'gui\_OutputFcn', @BrainMRI\_GUI\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before BrainMRI\_GUI is made visible.

function BrainMRI\_GUI\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to BrainMRI\_GUI (see VARARGIN)

% Choose default command line output for BrainMRI\_GUI

handles.output = hObject;

ss = ones(200,200);

axes(handles.axes1);

imshow(ss);

axes(handles.axes2);

imshow(ss);

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes BrainMRI\_GUI wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = BrainMRI\_GUI\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

[FileName,PathName] = uigetfile('\*.jpg;\*.png;\*.bmp','Pick an MRI Image');

if isequal(FileName,0)||isequal(PathName,0)

warndlg('User Press Cancel');

else

P = imread([PathName,FileName]);

P = imresize(P,[200,200]);

% input =imresize(a,[512 512]);

axes(handles.axes1)

imshow(P);title('Brain MRI Image');

% helpdlg(' Multispectral Image is Selected ');

% set(handles.edit1,'string',Filename);

% set(handles.edit2,'string',Pathname);

handles.ImgData = P;

% handles.FileName = FileName;

guidata(hObject,handles);

end

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

if isfield(handles,'ImgData')

%if isfield(handles,'imgData')

I = handles.ImgData;

gray = rgb2gray(I);

% Otsu Binarization for segmentation

level = graythresh(I);

%gray = gray>80;

img = im2bw(I,.6);

img = bwareaopen(img,80);

img2 = im2bw(I);

% Try morphological operations

%gray = rgb2gray(I);

%tumor = imopen(gray,strel('line',15,0));

axes(handles.axes2)

imshow(img);title('Segmented Image');

%imshow(tumor);title('Segmented Image');

handles.ImgData2 = img2;

guidata(hObject,handles);

signal1 = img2(:,:);

%Feat = getmswpfeat(signal,winsize,wininc,J,'matlab');

%Features = getmswpfeat(signal,winsize,wininc,J,'matlab');

[cA1,cH1,cV1,cD1] = dwt2(signal1,'db4');

[cA2,cH2,cV2,cD2] = dwt2(cA1,'db4');

[cA3,cH3,cV3,cD3] = dwt2(cA2,'db4');

DWT\_feat = [cA3,cH3,cV3,cD3];

G = pca(DWT\_feat);

whos DWT\_feat

whos G

g = graycomatrix(G);

stats = graycoprops(g,'Contrast Correlation Energy Homogeneity');

Contrast = stats.Contrast;

Correlation = stats.Correlation;

Energy = stats.Energy;

Homogeneity = stats.Homogeneity;

Mean = mean2(G);

Standard\_Deviation = std2(G);

Entropy = entropy(G);

RMS = mean2(rms(G));

%Skewness = skewness(img)

Variance = mean2(var(double(G)));

a = sum(double(G(:)));

Smoothness = 1-(1/(1+a));

Kurtosis = kurtosis(double(G(:)));

Skewness = skewness(double(G(:)));

% Inverse Difference Movement

m = size(G,1);

n = size(G,2);

in\_diff = 0;

for i = 1:m

for j = 1:n

temp = G(i,j)./(1+(i-j).^2);

in\_diff = in\_diff+temp;

end

end

IDM = double(in\_diff);

feat = [Contrast,Correlation,Energy,Homogeneity, Mean, Standard\_Deviation, Entropy, RMS, Variance, Smoothness, Kurtosis, Skewness, IDM];

load Trainset.mat

xdata = meas;

group = label;

svmStruct1 = svmtrain(xdata,group,'kernel\_function', 'linear');

species = svmclassify(svmStruct1,feat,'showplot',false);

if strcmpi(species,'MALIGNANT')

helpdlg(' Malignant Tumor ');

disp(' Malignant Tumor ');

else

helpdlg(' Benign Tumor ');

disp(' Benign Tumor ');

end

set(handles.edit4,'string',species);

% Put the features in GUI

set(handles.edit5,'string',Mean);

set(handles.edit6,'string',Standard\_Deviation);

set(handles.edit7,'string',Entropy);

set(handles.edit8,'string',RMS);

set(handles.edit9,'string',Variance);

set(handles.edit10,'string',Smoothness);

set(handles.edit11,'string',Kurtosis);

set(handles.edit12,'string',Skewness);

set(handles.edit13,'string',IDM);

set(handles.edit14,'string',Contrast);

set(handles.edit15,'string',Correlation);

set(handles.edit16,'string',Energy);

set(handles.edit17,'string',Homogeneity);

end

% --- Executes on button press in pushbutton3.

function pushbutton3\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

load Trainset.mat

%data = [meas(:,1), meas(:,2)];

Accuracy\_Percent= zeros(200,1);

itr = 80;

hWaitBar = waitbar(0,'Evaluating Maximum Accuracy with 100 iterations');

for i = 1:itr

data = meas;

%groups = ismember(label,'BENIGN ');

groups = ismember(label,'MALIGNANT');

[train,test] = crossvalind('HoldOut',groups);

cp = classperf(groups);

%svmStruct = svmtrain(data(train,:),groups(train),'boxconstraint',Inf,'showplot',false,'kernel\_function','rbf');

svmStruct\_RBF = svmtrain(data(train,:),groups(train),'boxconstraint',Inf,'showplot',false,'kernel\_function','rbf');

classes2 = svmclassify(svmStruct\_RBF,data(test,:),'showplot',false);

classperf(cp,classes2,test);

%Accuracy\_Classification\_RBF = cp.CorrectRate.\*100;

Accuracy\_Percent(i) = cp.CorrectRate.\*100;

sprintf('Accuracy of RBF Kernel is: %g%%',Accuracy\_Percent(i))

waitbar(i/itr);

end

delete(hWaitBar);

Max\_Accuracy = max(Accuracy\_Percent);

sprintf('Accuracy of RBF kernel is: %g%%',Max\_Accuracy)

set(handles.edit1,'string',Max\_Accuracy);

guidata(hObject,handles);

% --- Executes on button press in pushbutton4.

function pushbutton4\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

load Trainset.mat

%data = [meas(:,1), meas(:,2)];

Accuracy\_Percent= zeros(200,1);

itr = 100;

hWaitBar = waitbar(0,'Evaluating Maximum Accuracy with 100 iterations');

for i = 1:itr

data = meas;

%groups = ismember(label,'BENIGN ');

groups = ismember(label,'MALIGNANT');

[train,test] = crossvalind('HoldOut',groups);

cp = classperf(groups);

svmStruct = svmtrain(data(train,:),groups(train),'showplot',false,'kernel\_function','linear');

classes = svmclassify(svmStruct,data(test,:),'showplot',false);

classperf(cp,classes,test);

%Accuracy\_Classification = cp.CorrectRate.\*100;

Accuracy\_Percent(i) = cp.CorrectRate.\*100;

sprintf('Accuracy of Linear Kernel is: %g%%',Accuracy\_Percent(i))

waitbar(i/itr);

end

delete(hWaitBar);

Max\_Accuracy = max(Accuracy\_Percent);

sprintf('Accuracy of Linear kernel is: %g%%',Max\_Accuracy)

set(handles.edit2,'string',Max\_Accuracy);

% --- Executes on button press in pushbutton5.

function pushbutton5\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton5 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

load Trainset.mat

%data = [meas(:,1), meas(:,2)];

Accuracy\_Percent= zeros(200,1);

itr = 100;

hWaitBar = waitbar(0,'Evaluating Maximum Accuracy with 100 iterations');

for i = 1:itr

data = meas;

groups = ismember(label,'BENIGN ');

groups = ismember(label,'MALIGNANT');

[train,test] = crossvalind('HoldOut',groups);

cp = classperf(groups);

svmStruct\_Poly = svmtrain(data(train,:),groups(train),'Polyorder',2,'Kernel\_Function','polynomial');

classes3 = svmclassify(svmStruct\_Poly,data(test,:),'showplot',false);

classperf(cp,classes3,test);

Accuracy\_Percent(i) = cp.CorrectRate.\*100;

sprintf('Accuracy of Polynomial Kernel is: %g%%',Accuracy\_Percent(i))

waitbar(i/itr);

end

delete(hWaitBar);

Max\_Accuracy = max(Accuracy\_Percent);

%Accuracy\_Classification\_Poly = cp.CorrectRate.\*100;

sprintf('Accuracy of Polynomial kernel is: %g%%',Max\_Accuracy)

set(handles.edit3,'string',Max\_Accuracy);

function edit1\_Callback(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text

% str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.

function edit1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit2\_Callback(hObject, eventdata, handles)

% hObject handle to edit2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text

% str2double(get(hObject,'String')) returns contents of edit2 as a double

% --- Executes during object creation, after setting all properties.

function edit2\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit3\_Callback(hObject, eventdata, handles)

% hObject handle to edit3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text

% str2double(get(hObject,'String')) returns contents of edit3 as a double

% --- Executes during object creation, after setting all properties.

function edit3\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes during object creation, after setting all properties.

function pushbutton4\_CreateFcn(hObject, eventdata, handles)

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

function edit4\_Callback(hObject, eventdata, handles)

% hObject handle to edit4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text

% str2double(get(hObject,'String')) returns contents of edit4 as a double

% --- Executes during object creation, after setting all properties.

function edit4\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit5\_Callback(hObject, eventdata, handles)

% hObject handle to edit5 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text

% str2double(get(hObject,'String')) returns contents of edit5 as a double

% --- Executes during object creation, after setting all properties.

function edit5\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit5 (see GCBO)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit8\_Callback(hObject, eventdata, handles)

% hObject handle to edit8 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text

% str2double(get(hObject,'String')) returns contents of edit8 as a double

% --- Executes during object creation, after setting all properties.

function edit8\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit8 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit9\_Callback(hObject, eventdata, handles)

% hObject handle to edit9 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text

% str2double(get(hObject,'String')) returns contents of edit9 as a double

% --- Executes during object creation, after setting all properties.

function edit9\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit9 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

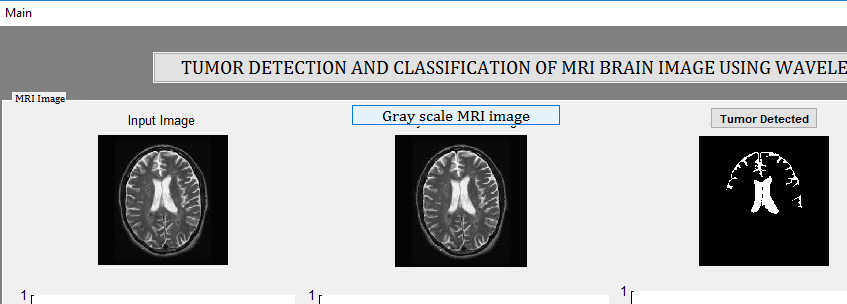
% See ISPC and COMPUTER.

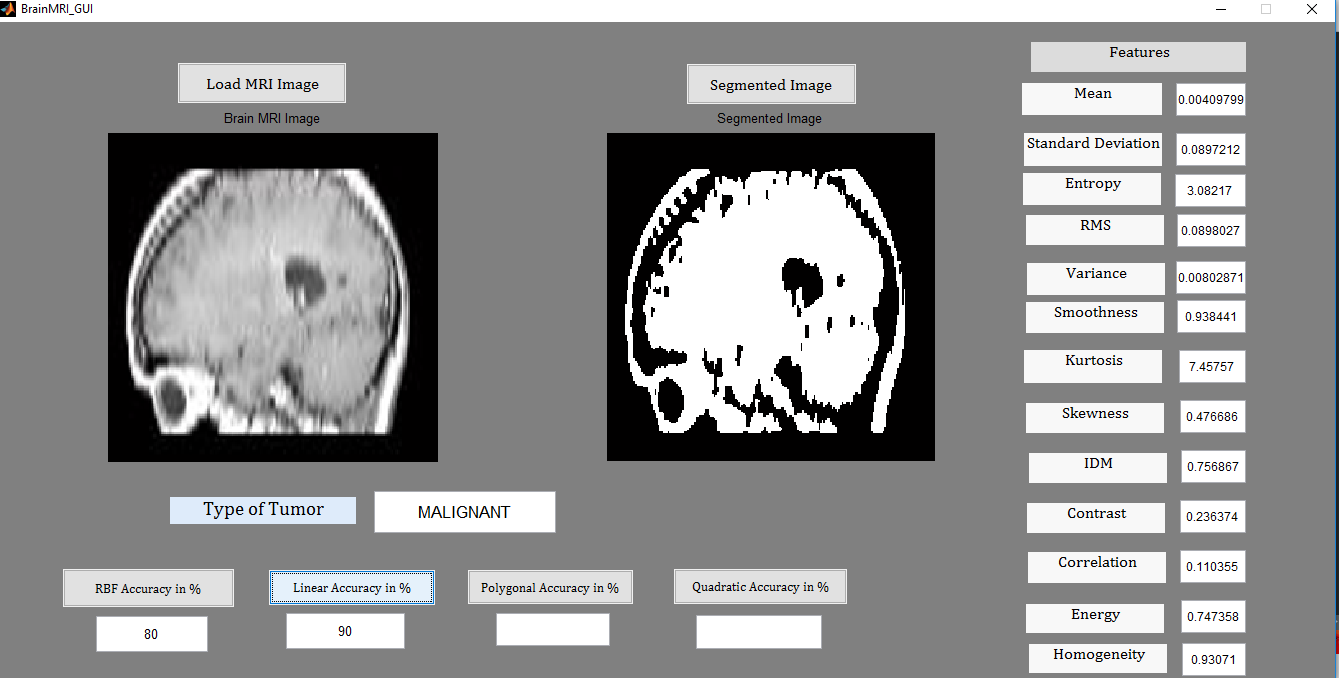
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

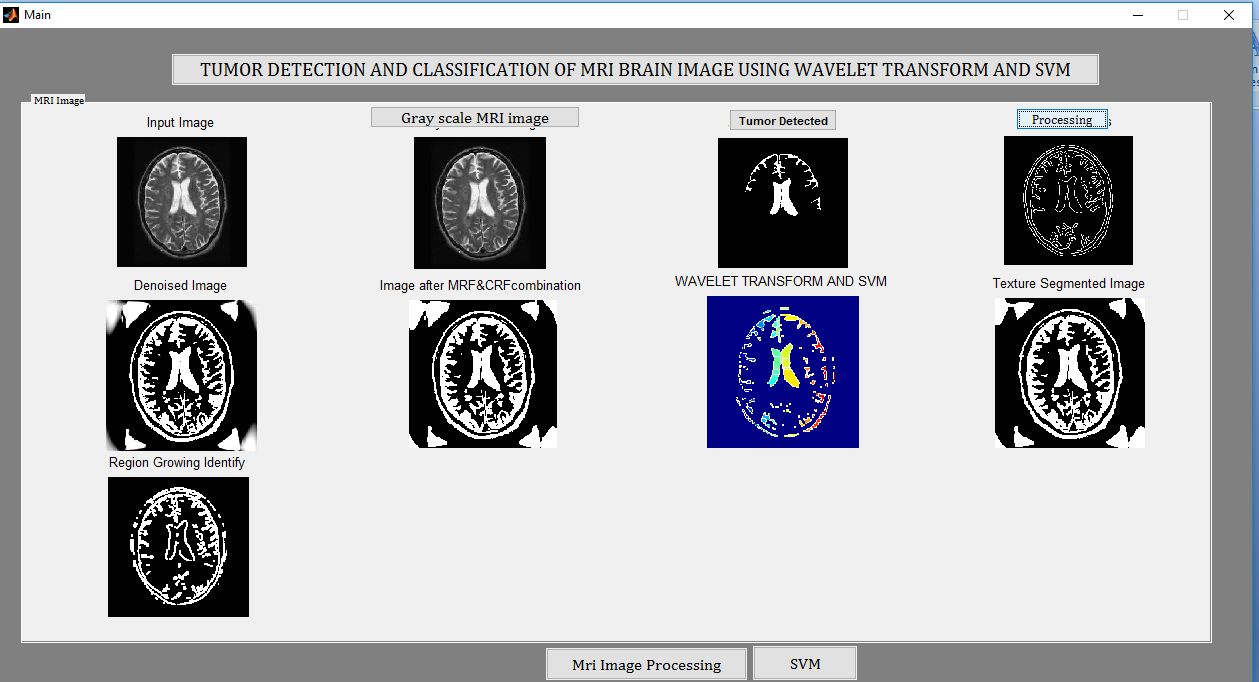
set(hObject,'BackgroundColor','white');

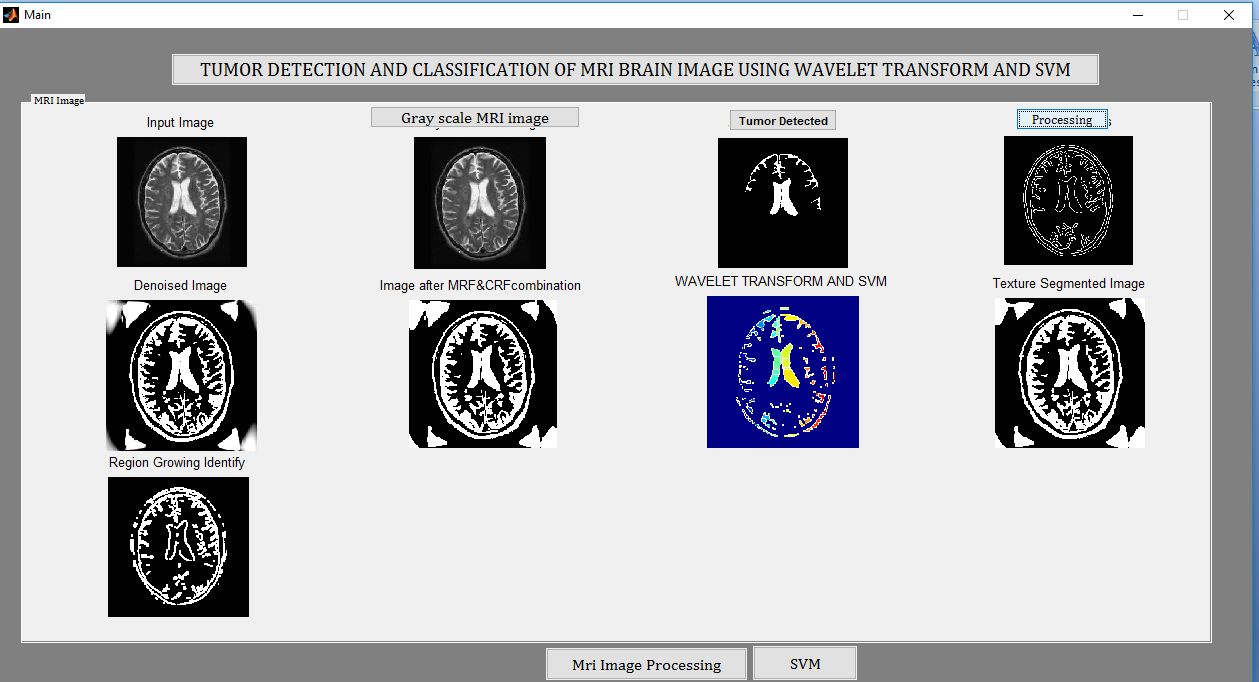
end

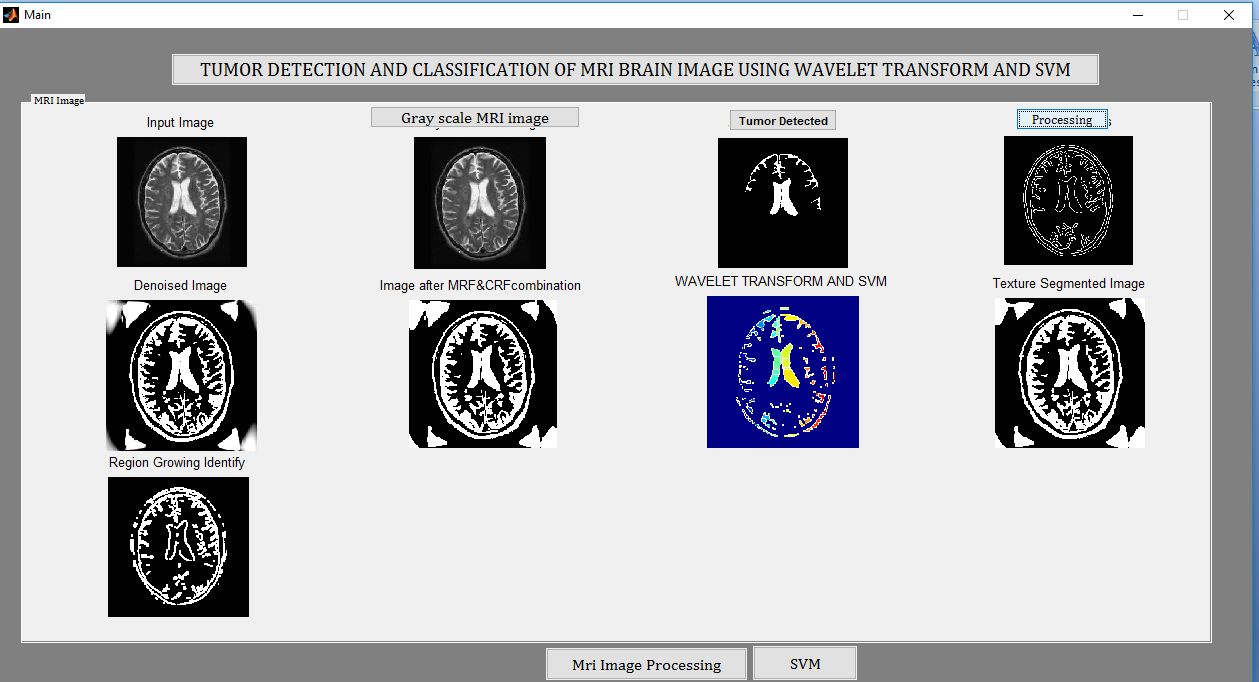
**SIMULATION OUTPUT:**

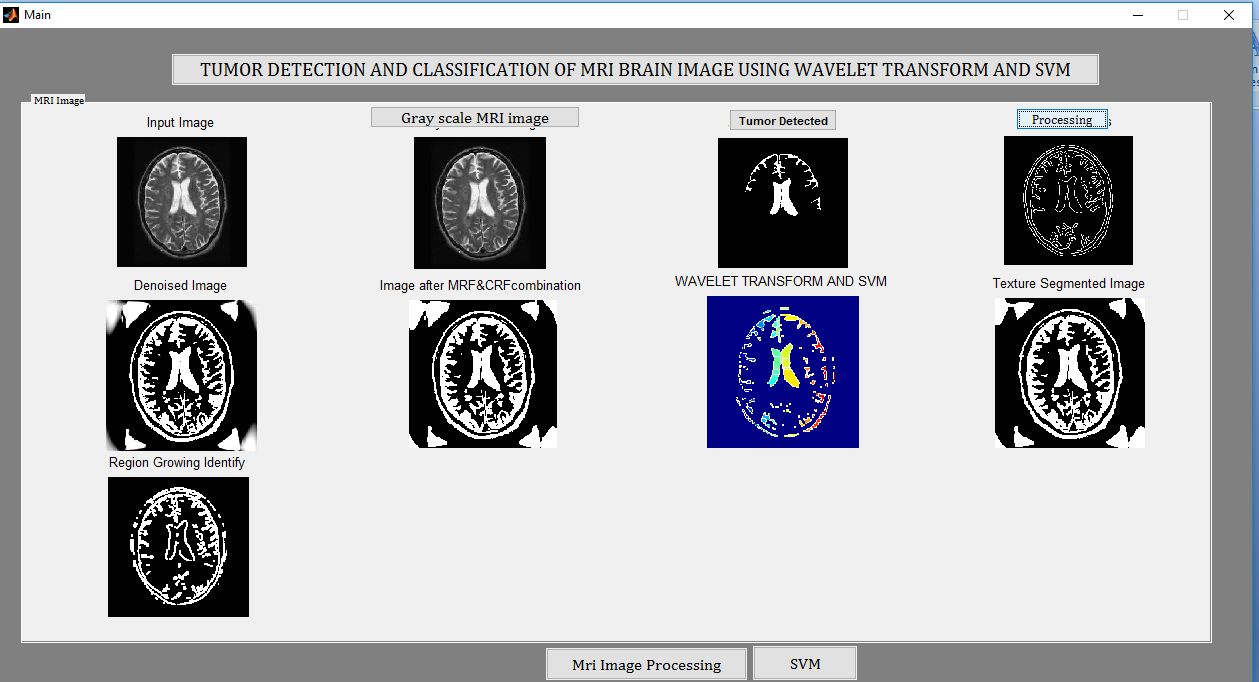
****

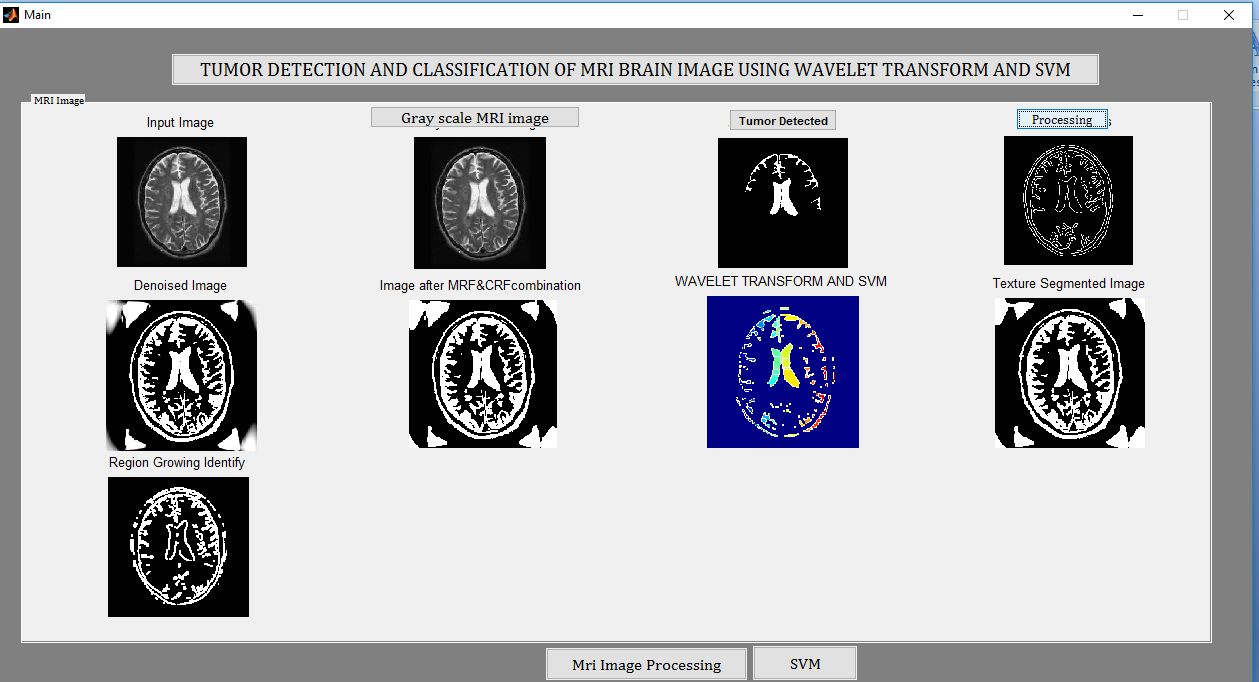
****

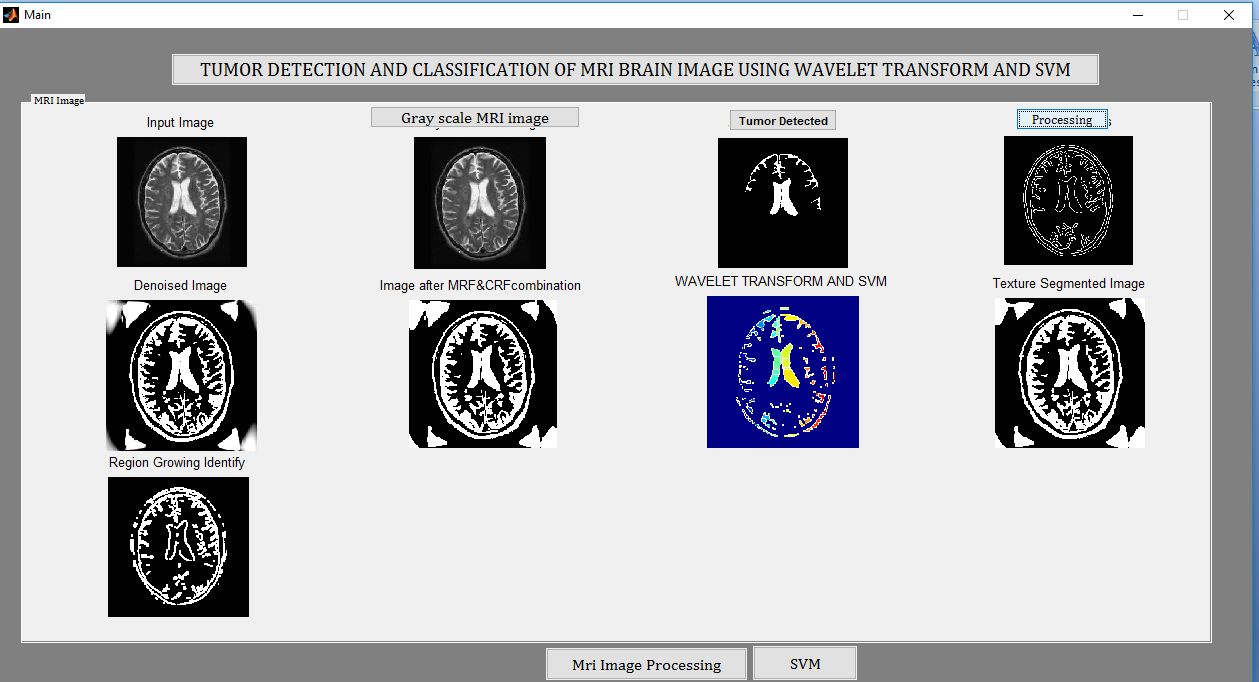
****

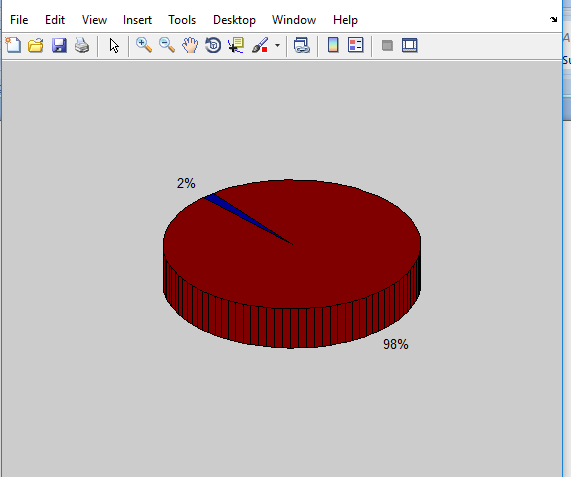
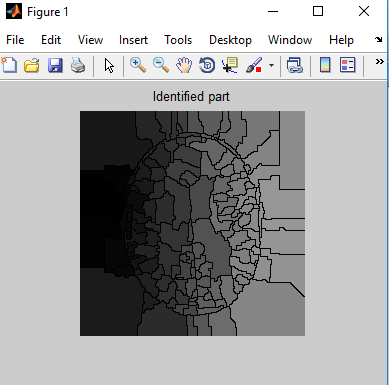
****

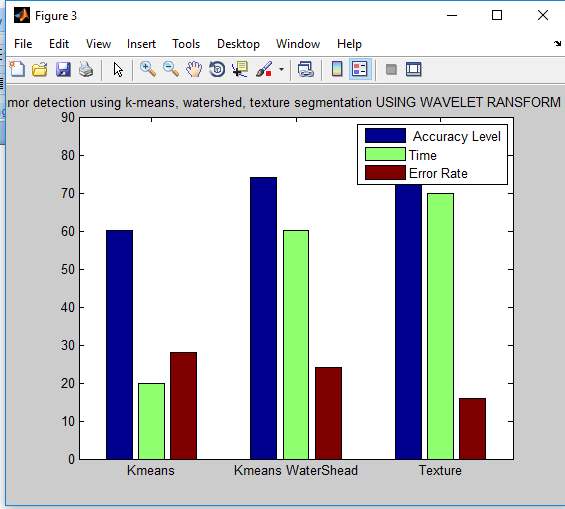
****

****

****

****

****

****

**8. Conclusion**

Tumor detection with SVMs which will achieve real-time performance while maintaining high detection accuracies. 82% of accuracy is obtained and therefore the positive predictive values (PPV) 81.48%, Negative predictive value (NPV) are calculated. truth positive cases are 22; True negative 5, False positive 5 and False negative are 22. Furthermore, an equivalent prototype are often used for various application no matter the window size, number of support vectors, and image size. We proposed an efficient technique that mixes the discrete wavelet rework (DWT) with Principle Component Analysis (PCA) to classify the brain MRIs into Normal and tumor affected one. Image thresholding has been applied for segmentation purpose. The proposed work is tested with ML and SVM classifier models. To balance the samples within the dataset classes, sampling technique has been adopted. This helps to enhance the proposed model’s classification accuracy by 3.4% on a mean . In future, we've planned to use this sampling with other medical datasets and with different classification algorithms

uting reduces more than 30 percent of the routing cost

needed for the regular ZigBee tree routing. If the destination is

the coordinator, the proposed algorithm shows as good

performance as ZigBee’s table driven routing, since

intermediate or source nodes select the node that has the

smallest tree level within its transmission range. For a random

destination, the performance of the proposed algorithm

depends on how many useful neighbor nodes are stored in the

neighbor table. It can be optimized by applying a proper

maximum number of nodes and management policy for the

neighbor table according to the network’s applications.

Therefore, we expect the proposed algorithm to be utilized in

many ZigBee applications requiring both small memory

capacity and high routing performance.

In future work this method exhibits an elegant and methodological choice of the threshold parameter. It also outperforms previous rule-based segmentation technique. The assessment of the Bayesian classification with regard to the colour reveals a close performance across the different colour spaces for a histogram resolution above 64. However, it should be noted that a slight advance of the HSV representation has been noticed. With regard to the colour quantization, we found

**REFERENCES**

[1]International Journal of Emerging Trends & technology in Computer Science (IJETTCS), ISSN 2278-6856, Volume: 04, Issue: 5(2), September-October 2015.

[2]Brain Tumor: Statistics -Approved by the Cancer.Net Editorial Board, 06/2016. [Online] Available: http://www.cancer.net/cancer-types/brain-tumor/statistics [Accessed: 06-May-2017].

[3]V. Shinde, P. Kine, S. Gadge and S. Khatal, “Brain Tumor Identification using MRI Iamges”, ISSN 2331-8169, Volume:2 Issue:10, October 2014.

[4]M. S. Al-Tarawnwneh, “Lung Cancer Detection Using Image Processing Techniques,” Leonardo Electro. J. of Practices and Technol., no. 20, pp. 147-158, 2012.

[5].H. G. Zadeh, S. Janianpour, and J. Haddadnia, “Recognition and Classification of the Cancer Cells by Using Image Processing and LabVIEW,” Int. J. of Comput. Theory and Engr., vol. 5, no. 1, pp. 104-107, 2013.

[6]Y. Hu, K. Ashenayi, R. Veltri, G. O'Dowd, G. Miller, R. Hurst, and R. Bonner, “A Comparison of Neural Network Classification,” in Proc. of IEEE International Neural Networks, vol. 6, pp. 3461-3466, 1994.

[7]X. W. Chen, “Gene Selection for Cancer Classification Using Bootstrapped Genetic Algorithms and Support Vector Machines,” in Proc. of IEEE Bioinformatics Conference, pp. 504-505, 2013.

[8]Amruta Pramod Hebli, Sudha Gupta “Brain Tumor Detection Using Image Processing: A Review”, in 65thIRF International Conference, 978-93-86291-38-7, 2016.

[9]Luxit Kapoor, Prof. SanjeeevThakur, “A Survey on Brain Tumor Detection Using Image Processing Techniques” in Proc. of IEEE International Cloud computing, Data Science & Engineering, 978-1-5090-3519-9, 2017

[10]Rajesh C. patil, A.S. Bhalchandra, “Brain tumor extraction from MRI images Using MAT Lab”, IJECSCSE, ISSN: 2277-9477, Volume 2, issue1

[4]M. S. Al-Tarawnwneh, “Lung Cancer Detection Using Image Processing Techniques,” Leonardo Electro. J. of Practices and Technol., no. 20, pp. 147-158, 2012.

[5].H. G. Zadeh, S. Janianpour, and J. Haddadnia, “Recognition and Classification of the Cancer Cells by Using Image Processing and LabVIEW,” Int. J. of Comput. Theory and Engr., vol. 5, no. 1, pp. 104-107, 2013.

[6]Y. Hu, K. Ashenayi, R. Veltri, G. O'Dowd, G. Miller, R. Hurst, and R. Bonner, “A Comparison of Neural Network Classification,” in Proc. of IEEE International Neural Networks, vol. 6, pp. 3461-3466, 1994.

[7]X. W. Chen, “Gene Selection for Cancer Classification Using Bootstrapped Genetic Algorithms and Support Vector Machines,” in Proc. of IEEE Bioinformatics Conference, pp. 504-505, 2013.

[8]Amruta Pramod Hebli, Sudha Gupta “Brain Tumor Detection Using Image Processing: A Review”, in 65thIRF International Conference, 978-93-86291-38-7, 2016.

[9]Luxit Kapoor, Prof. SanjeeevThakur, “A Survey on Brain Tumor Detection Using Image Processing Techniques” in Proc. of IEEE International Cloud computing, Data Science & Engineering, 978-1-5090-3519-9, 2017

[10]Rajesh C. patil, A.S. Bhalchandra, “Brain tumor extraction from MRI images Using MAT Lab”, IJECSCSE, ISSN: 2277-9477, Volume 2, issue1