PROCESSING OF THREE-DIMENSIONAL SCANNED IMAGES OF HUMAN BRAIN

Hariharan.D[1], Jagadeeswaran.S[2], Gowsik.B[3], Thiagarajan Kittappa[4], Dr.K.Regin Bose[5]
[1,2,3]Department of Artificial Intelligence and Data Science, Rajalakshmi Institute of Technology, Chennai, India; [4]Department of Mathematics, Rajalakshmi Institute of Technology, Poonamallee, Chennai, India ;
[5]Professor, Artificial Intelligence and Data Science Department, Rajalakshmi Institute of Technology, Poonamallee, Chennai 600124, India
[1]hariharan.d.2122@gmail.com; [2]sjagadees59@gmail.com; [3]gowsikb178@gmail.com;
[4]vidhyamannan@yahoo.com; [5]reginbosek@gmail.com

Abstract

This paper presents a novel method of processing three-dimensional (3-D) digital images of human brain, which are acquired by scanners like Computed Tomography (CT) system, Magnetic Resonance Imaging (MRI) system, Positron Emission Tomography (PET) system. The main objective is to find feasible solutions to some of the problems for detecting superficial and volumetric features of three-dimensional digital images. 'Superficial features' means surfaces, points and lines. 'Volumetric features' means contours, edges and skeletal forms. In order to carry out the operations of extracting these features from three dimensional images, it is proposed here to make use of either mathematical morphological algorithms or cellular logic array processing (CLAP) algorithms. With the help of the extracted features, one will be able to decide the status of any tumour present in the brain.

Keywords: MRI Scan, CT Scan, PET System, Features of three-dimensional images

1. Introduction

Some of the other superficial and volumetric features of a 3-D image are 'points', 'lines', 'curves', 'surfaces', 'textures', 'skeletal forms' and 'wireframe models'. Those detected on plain surfaces are called as superficial features and all others as volumetric features. First, a three-dimensional digital image is viewed here as a subset of Z×Z×Z where Z is the ring of integers and its processing as a state-of-the-art technology consisting of two components, one visualization of 3-D digital images on a two-dimensional digital monitor and the other processing the image by convolving it with a three dimensional mask. The visualization is called '***volume rendering***' and the convolution '***volume processing'***. Brief explanation of these two components is given below.

Volume rendering

This is a technique for visualizing 3-D sampled data on to two-dimensional digital monitor by computing its 2D projections. There are two methods of rendering a volume. The first method is called '**slice based rendering**' and the second '**volume based rendering'**. Many different methods of volume-based rendering are available in standard literature[111]. They are (i) ray casting, (ii) splatting, (iii) shear warping, (iv) texture mapping and (v) maximum intensity projection (MIP). This paper advocates Kruger and Westermann's ray casting technique for 3-D image visualization. Basically, this technique implements texture-based rendering of volume on a graphics processing unit (GPU) by integrating acceleration techniques like early ray termination and empty space skipping. Fig.1 shows a sample image rendered on the 2-D monitor using Kruger and Westermann's ray casting technique.

(a) (b)

Fig.1: (a) Sagittal view of the 3D image
(b) Frontal view of the 3-D image

A 3-D image as seen on a monitor is a bunch of light intensities and one will not be able to visualize the hidden parts in order to make meaningful interpretations. In this context, processing the 3-D image data is nothing but extraction of physical, morphological and structural properties of the hidden parts of the image. For example, the MRI image data of say a human heart is processed to extract hidden details, mostly visual in nature, about various parts of the heart like pulmonary artery, aorta, thoracic chamber, tricuspid and mitral valves, auricles and ventricles. Precise details about these parts could be obtained only when the processing techniques are reliable and robust.

A 3-D digital image data consists of voxels, each voxel exhibiting a specific property like light intensity, pressure, and temperature. The voxels are arranged in the form of a 3-D number array. So, 3-D image processing ultimately turns out to be updating of the numbers in the 3-D array using a formula or algorithm developed in a formal mathematical or logical framework. Many researchers have been trying to develop methodologies and algorithms to process 3-D digital images. From most of whatever has been reported in the literature one would be able to infer that no algorithm or methodology could be accepted as a universal tool to process all kinds of 3-D image data. In fact, one would feel the necessity of having different algorithms for carrying out a particular operation on different data which are obtained using a sensor or different sensors. Three-dimensional digital image processing is an important area of research that has numerous applications in the fields of medical imaging. Depending on the requirement for solving a specific problem, one has to identify various features of three-dimensional images while applying suitable algorithms. For example, the quantitative analysis of anatomical structures in MRI/CT images requires extraction of volumes and shapes of certain parts of human body. In order to do this, there is a need for detecting features such as points, lines, contours, surfaces, wireframes, textures and skeletal forms from three dimensional images. Detecting such superficial and volumetric features of three-dimensional images has always been a difficult task and there is little progress made and found in literature. This problem is the actual motivating factor behind the work carried out and reported in this paper. 'Superficial features' means surfaces, points and lines. 'Volumetric features' means contours, edges and skeletal forms. In order to carry out the operations of extracting these features from three dimensional images, it is proposed here to make use of either mathematical morphological algorithms or cellular logic array processing (CLAP) algorithms. Further details about these two paradigms are given below.

Paradigms adopted in the research work

Processing of 3-D images using morphological methods

The basic morphological operations involved in extracting superficial and volumetric features are 3-D erosion and 3-D dilation. It is to be noted here that the shape and structure of a processed 3-D image depends exclusively on the shape and structure of the structuring element used while carrying out these operations. There is no formal method to generate three-dimensional structuring elements of different shapes and structures.

Processing of 3-D images using cellular logic methods

There is another novel paradigm called 'Cellular Logic Array Processing (CLAP)' introduced by E. G. Rajan in 1989, in which one can develop fast algorithms for processing digital images. Cellular Logic Array Processing is a logico-mathematical framework developed using the fundamental notions of Markov's 'Normal Algorithms' and von Neumann's 'Cellular Automata'. Normal algorithm is an idealized serial processor like a Turing machine and a cellular automaton is an idealized parallel processor. CLAP is a logical technique of realizing cellular automata using normal algorithms as shown in the Fig. 2.

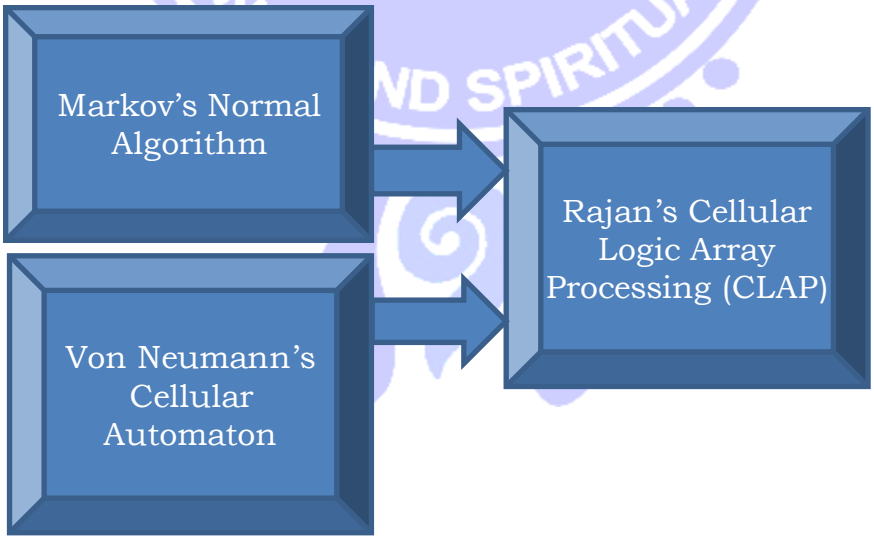


Fig. 2: Cellular Logic Array Processing (CLAP)

Scope of research carried out

The area covered while carrying out research in 'Detecting the Superficial and Volumetric features in 3-D Digital Images' is brain tumour detection in an MRI scanned image. Fig. 3 shows the MRI scanned image of a human brain and brain tumour detected in the image,

Fig. 3: MRI scanned image of a human brain and the tumour detected

2. Concept of Cellular Logic Array Processing

Filter is an ordered dichotomy <accepted, rejected>. That is the function of a filter is to separate a given set of items from a domain into disjoint subsets by virtue of a stipulated set property. In the formulation and design of various techniques for filtering of signals, one important principle called Rajan Sinha Principle (RSP) could be seen to play a significant role. RSP states "System models of signal processing operations and those of sources from which signals originate should belong to the same class". As per this principle, signals could be thought of as outputs of sources, and their description is not actually complete unless one specifies source models. To go by the general pattern of relationships that this principle suggests, it is only appropriate that for signals that evolve with time and whose sources can be modelled as time-invariant systems governed by differential or difference equations, e.g., audio signals, we should look for processors that can likewise be modelled as time-invariant systems. For pictures and images on the other hand, there is good reason to believe that the traditional 2-D extensions of time-invariant systems governed by differential equations and partial differential equations will not serve as canonical models. It is not surprising therefore that the traditional filter theory does not measure up to the task in this case and one has to look for alternative formulations. It is significant to note in this connection that for a large class of pictures and images of high complexity, an effective modelling tool for sources generating them has been found to be that of Cellular Automata. Invoking the principle just set forth, it is then reasonable to surmise that filtering operations on pictures and images, such as those of thinning, edge detection, segmentation, erosion and dilation may be canonically modelled and realized using cellular automata. There is yet another point that the principle prompts us to take into account. In the traditional theory of filtering, we are able to numericalize our problems by viewing signals as members of a vector space over which it is possible for us to characterize their properties or attributes of interest in terms of the notion of spectrum. But in the case of pictures and images, some of the pertinent attributes are such that a straightforward numericalization of this kind is not possible, and the notion of spectrum in its present form is not very useful. Interesting and promising new possibilities, however, emerge if instead of working with numerical representations of signals and systems, we study filtering operations as formal symbol manipulating algorithms. Markov's Normal Algorithms seem to provide a rich repertoire of tools for this purpose. In general, given a string of symbols one can transform it into another desired string by systematically rearranging or rewriting the symbols contained in the given string according to a finite set of rules. A prescription, which gives a set of such rewriting rules along with the details of how they should be applied, is known as a string manipulating algorithm. The concept of a normal algorithm refers to one such string manipulating algorithm. Essentially, a normal algorithm is an ordered list of a finite number of what are called substitution formulas. A substitution formula is analogous to a semi-Thue production of Chomsky type-0 grammar. Cellular automata are idealized parallel processing systems, whereas normal algorithms are idealized sequential (single processor) systems. Cellular Logic Array Processing is a state-of-the-art of realizing cellular automata using normal algorithms. A one-dimensional N-ary valued cellular automation C consists of a 1-D array of cells. Each cell takes on any of N given values, and all the values are simultaneously updated at a particular stage of evolution according to a rule involving the values contained in certain neighboring cells. The evolutions of a cellular automaton are assumed to take place in successive stages. The values of a cellular automaton are assumed to take place in successive stages. The values of a cell i, at a

particular stage of evolution t, is an element from a set of number X = {0, 1, … , N-1 }, and it is denoted by $\xi^{(t)}_i$. A numerical sequence corresponding to the values contained in the linear array of cells at a stage t, is called a configuration of C. From a given configuration, a new configuration is evolved by updating the given cell values in accordance with a rule that decides the new value of each cell based on its old values and those of some of its neighbors. By the r-neighborhood of a cell, we mean the cell itself together with (r-1/2) adjacent cells to its immediate let (r-1/2) adjacent cells to its immediate right; is always assumed to be odd. For example, a 3-neighborhood of a cell i of a cellular automaton C is a structure formed by the cell i and its immediate left and right neighbours as shown below:

i-1	i	i+1

At a state t, let $\xi^{(t)}_{i-1}$, $\xi^{(t)}_i$ and $\xi^{(t)}_{i+1}$ be the values held by the 3-neighborhood of i. Now, let us consider an updating rule ϕ that applies to all 3-neighborhoods and let us apply it to the configuration at stage t. Then, the value of i^{th} cell $\xi^{(t)}_i$ is updated as $\xi^{(t+1)}_i$ where, $\xi^{(t+1)}_i = \phi\ (\xi^{(t)}_{i-1}\ ,\ \xi^{(t)}_i\ ,\ \xi^{(t)}_{i+1})$. All the cell values in the configuration are updated simultaneously in this manner. For the sake of illustration, let us take the case of a one-dimensional three-neighborhood cellular automaton whose cell variables take on values from the set X = {0, 1}. An updating rule ϕ , in this case, would be a Boolean function of the sites within the neighborhood. As an example, let us consider a Boolean function ϕ: $\xi^{(t+1)}_i = \xi^{(t)}_{i-1} \oplus \xi^{(t)}_{i+1}$ which is a modulo two rule. According to this rule, the value of a particular cell is simply the sum modulo two of the values of its two neighboring cells on the previous time step. Thus, one can generate a total of 256 rules (Boolean Functions) for a binary valued, three-neighborhood one-dimensional cellular automaton. In general, one can generate a total of N^{Nr} rules for an N-ary values, r-neighborhood one-dimensional cellular automaton. A two dimensional N-ary valued cellular automaton C consists of 2-D array of cells. Each cell takes on any of the N given values, and all the values are simultaneously updated at a particular stage of evolution according to a rule involving the values contained in certain surrounding cells. Though one can think of several possible 2-D arrays and neighborhood structures, we are concerned here only with square arrays with 5 and 9-neighborhood structures as shown in Fig. 4.

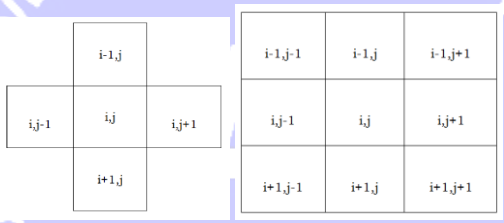


Fig. 4: Five and nine neighborhood structures

Let us consider a square array and let (i,j) be one of its cells. Then the value, $\xi^{(t)}_{i,j}$, held by the cell (i,j) at a stage t is from a set of numbers X={0, 1, … , N-1}. Now, the value held by the cell (i,j) could be updated in accordance with a 5-neighborhood or a 9-neighborhood rule. A 5-neighborhood cellular automaton evolves according to a rule ϕ :

$$\xi^{(t+1)}_{i,j} = \phi\ (\xi^{(t)}_{i,j}\ ,\ \xi^{(t)}_{i,j+1}\ ,\ \xi^{(t)}_{i+1,j}\ ,\ \xi^{(t)}_{i,j-1}\ ,\ \xi^{(t)}_{i-1,j}\ )$$

Similarly, a 9-neighborhood cellular automaton evolves according to a rule

$$\xi^{(t+1)}_{i,j} = \phi\ (\xi^{(t)}_{i,j}\ ,\ \xi^{(t)}_{i,j+1}\ ,\ \xi^{(t)}_{i+1,j+1}\ ,\ \xi^{(t)}_{i+1,j}\ ,\ \xi^{(t)}_{i-1,j-1}, \xi^{(t)}_{i,j-1}\ ,\ \xi^{(t)}_{i-1,j-1}\ ,\ \xi^{(t)}_{i-1,j}\ ,\ \xi^{(t)}_{i,j+1}\ )$$

N^{N^5} local rules for a five-neighborhood N-ary valued two-dimensional cellular automaton could be constructed and similarly N^{N^9} rules for a nine-neighborhood N-ary valued two-dimensional cellular automaton could be constructed.

For example, consider a binary valued five-neighborhood two-dimensional cellular automaton. Local rules in this case are 32-bit words. we have a total of 2^{32} such local rules. Similarly, an updating rule of a nine-neighborhood automaton is a 512-bit word, and we have a total of 2^{512} such local rules. Analogous to a two-dimensional cellular automaton, a three-dimensional N-ary valued cellular automaton consists of a 3-D array of cells. Each cell takes on any of the N given values, and all the values are simultaneously updated at a particular stage of evolution according to a rule involving the values contained in certain surrounding cells. Though it is possible to have several 3-D neighborhood structures, we are concerned here only with two types, seven and nine neighborhood structures as shown in Fig.5(a) and Fig. 5(b).

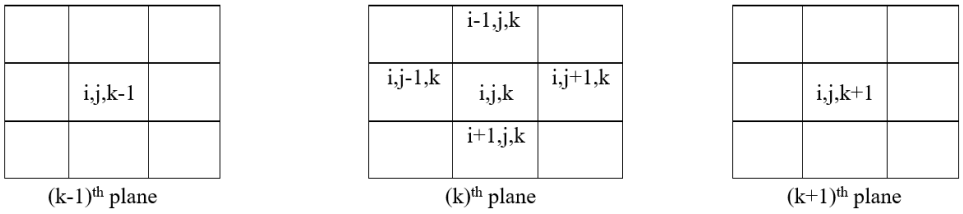


Fig. 5 (a): A 3-D 7-neighborhood structure

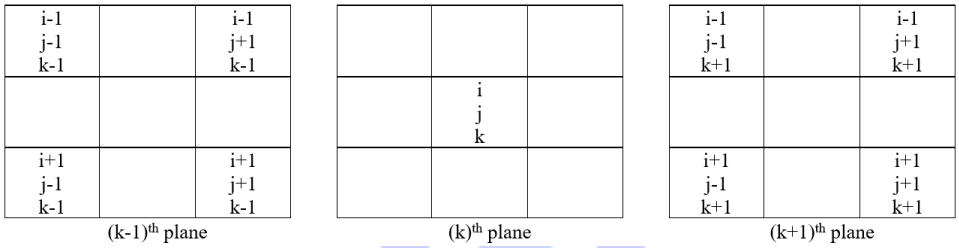


Fig. 5(b): A 3-D 9-neighborhood structure

Let us consider a 3-D array and let (i,j,k) be one of its cells. Then the value, $\xi^{(t)}_{i,j,k}$, held by the cell (i,j,k) at a stage t is an element from a set of numbers X = { 0 , 1, … , N-1 } . Now, the value held by the cell (i,j,k) could be updated in accordance with a 7-neighborhood or a 9-neighborhood rule. A 3-D, t-neighborhood cellular automaton evolves according to a rule ϕ:

$$\xi^{(t+1)}_{i,j,k} = \phi (\xi^{(t)}_{i,j,k} , \xi^{(t)}_{i,j,k+1} , \xi^{(t)}_{i,j+1,k} , \xi^{(t)}_{i+1,j,k} , \xi^{(t)}_{i,j,k-1} , \xi^{(t)}_{i,j-1,k} , \xi^{(t)}_{i-1,j,k})$$

Similarly, a 3-D 9-neighborhood cellular automaton evolves according to a rule

$$\xi^{(t+1)}_{i,j,k} = \phi (\xi^{(t)}_{i,j,k} , \xi^{(t)}_{i+1,j+1,k-1} , \xi^{(t)}_{i+1,j-1,k-1} , \xi^{(t)}_{i+1,j-1,k+1} , \xi^{(t)}_{i+1,j+1,k+1} ,$$
$$\xi^{(t)}_{i-1,j+1,k+1} , \xi^{(t)}_{i-1,j+1,k-1} , \xi^{(t)}_{i-1,j-1,k-1} , \xi^{(t)}_{i-1,j-1,k+1})$$

N^{N^7} local rules for a seven-neighborhood N-ary valued three-dimensional cellular automaton could be constructed and N^{N^9} rules for a nine-neighborhood N-ary valued three-dimensional cellular automaton could be constructed.

Generalized Cellular Logic Scheme

The notion of cellular logic array processing refers to the implementation of a cellular automaton rule using normal algorithmic substitution formulas. With an idea of reducing space occupied by various cellular automata rules, and to provide a generalized mechanism of rule-writing, the notion of generalized cellular logic scheme was introduced. An r-neighborhood cellular automaton rule is a look-up table consisting of two columns one having a list of N^r patterns and the other N^r values. As N becomes large, it turns out to be a tedious and error-prone job to make entries in the corresponding look-up table. In addition, the look-up table would occupy large amount of storage space. In order to overcome this difficulty, either we adhere to a structured approach to rule-writing, or use a language for expressing a rule and a mechanism for interpreting it so as to translate our needs into a look-up table. In fact, one can use a programming language for interpreting our needs, but it is difficult to get them translated into a look-up table without using some specialized primitives of the language for realizing them in a machine.

On the other hand, the notion of a generalized cellular logic scheme does not require specialized primitives of a high-level language. It actually deals with the relational aspects of cell values in a particular neighborhood and makes use of pattern directed symbol-rewriting rules. The term *pattern* refers to a group of cell values in a particular neighborhood arranged in a manner described by a relational expression or by a logical formula. Any programming language could be used for constructing logical formulas corresponding to various updating rules. It is to be noted that a logical formula will not generate a complete look-up table for a rule. Alternatively, it will generate a minimal set of pattern-directed rules required for the processing of a configuration. In fact, we treat an ordered list of such a minimal set of pattern-directed rules as the scheme of a Generalised Markov Algorithm which is a dimensional extension of string processing normal algorithm. A simple example would clarify this notion.

Consider a two-dimensional binary-valued cellular automaton and a five-neighborhood rule described by the logical expression: if $(\xi^{(t)}_{i,j}, \xi^{(t)}_{i,j+1}, \xi^{(t)}_{i+1,j}, \xi^{(t)}_{i,j-1}, \xi^{(t)}_{i-1,j})_1$ is odd, then $\xi^{(t)}_{i,j}$ is 1; otherwise it is 0. The term $(\xi^{(t)}_{i,j}, \xi^{(t)}_{i,j+1}, \xi^{(t)}_{i+1,j}, \xi^{(t)}_{i,j-1}, \xi^{(t)}_{i-1,j})_1$ refers to the number of 1s in a sub array under the five neighborhood structure.

This expression refers to a Generalized Markov Algorithm consisting of the following **sixteen** pattern-directed rules.

N^{PAR} :	Substitution formulas	Formula number
	00001→1	00
	00010→1	01
	00100→1	02
	00111→1	03
	01000→1	04
	01011→1	05
	01101→1	06
	01110→1	07
	10000→1	08
	10011→1	09
	10101→1	10
	10110→1	11
	11001→1	12
	11010→1	13
	11100→1	14
	11111→1	15

But the expression $\xi^{(t)}_{i,j} = \xi^{(t)}_{i,j} \oplus \xi^{(t)}_{i,j+1} \oplus \xi^{(t)}_{i+1,j} \oplus \xi^{(t)}_{i,j-1} \oplus \xi^{(t)}_{i-1,j}$ which is used to perform the same operation given above would generate **thirty-two** rules listed below.

00000→0	01000→1	10000→1	11000→0
00001→1	01001→0	10001→0	11001→1
00010→1	01010→0	10010→0	11010→1
00011→0	01011→1	10011→1	11011→0
00100→1	01100→0	10100→0	11100→1
00101→0	01101→1	10101→1	11101→0
00110→0	01110→1	10110→1	11110→0
00111→1	01111→0	10111→0	11111→1

So, it is observed from the above that relational pattern-directed approach to cellular automata realizations should be a better and more efficient means of doing array processing. We call this as a '**Generalized Cellular Logic Scheme**' and also as '**Cellular Logic Array Processing**'.

3. Processing of MRI Scanned Image of a Human Brain

The given 3-D digital image is segmented by a threshold-based quantization scheme and at that moment boundaries of the quantized regions are extracted. The given 3-D digital image is plane-wise raster-scanned by the seven-neighborhood window which is shown below.

On each move, the 3X3X3 sub image covered by this window is examined to see whether the gray-distance, say D, which is the difference between the maximum and the minimum gray value corresponding to that sub image is less than or equal to the user specified threshold value, say T. If D is less than or equal to T, then the gray-value 0 is assigned to all the seven cells in the given image. For D greater than T, the original values contained in these cells are left undisturbed. This procedure is repeated until the entire image is scanned. The final result is that the edges of various 3-D solid regions in the given image that appear to be uniform are retained and their remaining interior parts are erased thus giving us the edge of the original image.

On each move, the 3X3X3 sub image covered by this window is examined to see whether the gray-distance, say D, which is the difference between the maximum and the minimum gray value corresponding to that sub image is less than or equal to the user specified threshold value, say T. If D is less than or equal to T, then the gray-value 0 is assigned to all the seven cells in the given image. For D greater than T, the original values contained in these cells are left undisturbed. This procedure is repeated until the entire image is scanned. The final result is that the edges of various 3-D solid regions in the given image that appear to be uniform are retained and their remaining interior parts are erased thus giving us the edge of the original image.

The above algorithm has been used on the MRI scanned brain image called 'Cerebrix' and its detected version are shown below in Fig. 6.

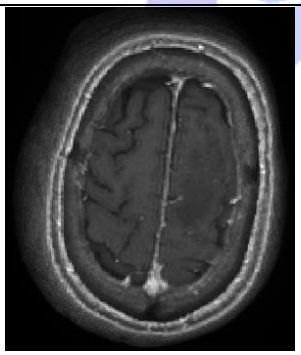
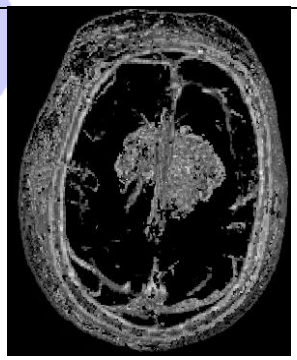
Algorithm	Input Image	Processed Image
Cellular Logic Array Processing Based Edge Detection CLAP-BED		

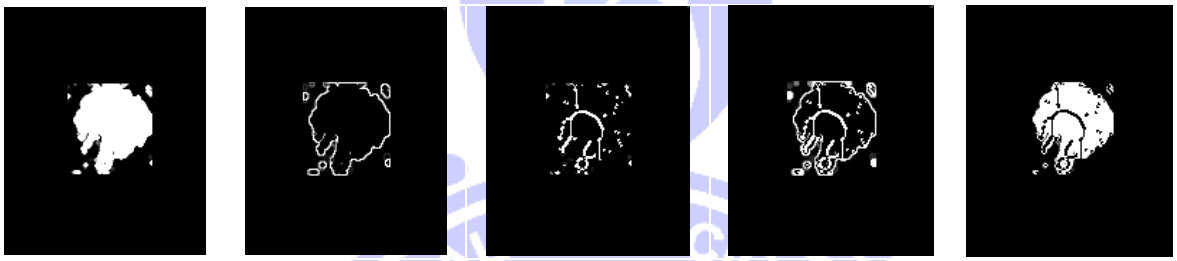
Fig. 6: MRI scanned image of human brain and tumour isolated in it

The time taken to process the 'Cerebrix' image was evaluated and result shown below.

Algorithm	Number of Voxels Before Processing	Number of Voxels After Processing	Processing Time (In Seconds)
CLAP-BED	1205998	158059	1.27

In the time complexity, n is the size of 3D image and r is the size of the structuring element. For example, if the size of the 3D image is 256x256x256 then n is 256. The structuring element size can vary as 3x3x3, 5x5x5, 7x7x7 and so on so that r turns out to be 3, 5, 7 accordingly. Now, the time complexity of the CLAP-BED algorithm for detecting edges in a 3D image is evaluated as $O(n^3r^3)+O(n^3)$.

The tumour portion alone is isolated, its contour, skeleton forms obtained. Fig. 7 shows image of tumour isolated alone and its processed versions.



Isolated brain tumour | Contoured version | Skeletonized version | Skeleton superimposed over contour | Skeleton superimposed over tumour image

Fig. 7: Tumour alone isolated, its skeleton forms obtained and superimposed on contour and the original tumour image

With reference to Fig. 7, one can explore the possibilities of evaluating the volume of the tumour and the direction of proliferation of the brain tumour.

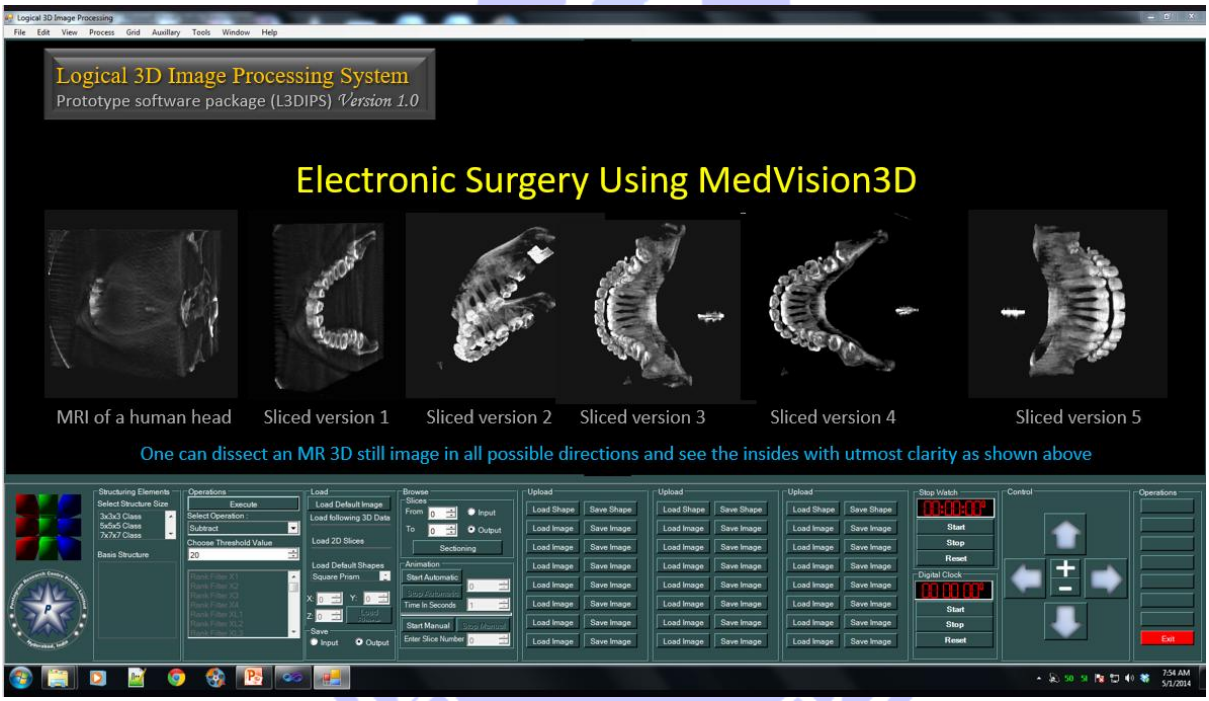
4. Conclusions and Future Perspectives

To summarize, there is a need for developing reliable techniques to extract hidden features in a three-dimensional image. A novel paradigm called 'Cellular Logic Array Processing' is introduced in this paper, in which one can develop fast and reliable algorithms to process MRI scanned images. One such algorithm is '3D Edge Detection' and another algorithm '3D Skeletonization' which is a complement of 3D contouring have been discussed with a practical application on an MRI scanned image called "Cerebrix'. As future perspective, one can explore the possibilities of creating an Artificial Intelligence based system to predict the proliferation directions of brain tumour.

References

[1] H.K. Liu; "Two and Three dimensional boundary detection"; Computer Graphics Image Proc.; Vol. 6; pp.123-134; 1977.
[2] S.W. Zucker, R.A. Hummel; "A Three Dimensional Edge Operator"; IEEE Trans. On PAMI; Vol. 3; Issue 3; May 1981.
[3] M.F. Heuckel, "An operator which locates edges in digitized pictures", j. Ass. Comput. Mach., Vol. 18, pp. 113-125, 1971.
[4] M. Morgenthaler, A. Rosenfeld; "Multidimensional edge detection by hypersurface fitting"; IEEE Trans. PAMI; Vol.3; Issue 4; July; 1981.
[5] O. Monga and R. Deriche; "A new three dimensional boundary detection"; In Proc. Of International Conference on Pattern Recognition; Paris; 1986.
[6] R. Deriche; "Using canny's criteria to derive a recursively implemented optimal edge detector"; International Journal of Computer Vision; 1(2); May 1987.
[7] O. Monga and R. Deriche, and J-M. Rocchisani "3D Edge detection using recursive filtering : Application to scanner images"; CVGIP: Image Understanding; vol.53; no.1; pp.78-87 (1991)
[8] Y. J. Zhang; "Quantitative study of 3D gradient operators"; Image and Vision Computing; vol. 11; pp. 611-622; 1993.
[9] P. Bhattacharya and D. Wild; "A new edge detector for gray volumetric data" Comput. Biol. Med.; vol. 26; pp. 315-328; 1996.

[10] R. M. Haralick and O. A. Zuniga; "Integrated directional derivative gradient operator"; IEEE Trans. Systems; Man and Cybernetics; vol. 17; pp. 508-517; 1987.
[11] Marek Brejl and Milan Sonka; "Directional 3D Edge Detection in Anisotropic Data: Detector Design and Performance Assessment"; Computer Vision and Image Understanding; Vol. 77; Issue9; Feb;2000.
[12] Luo LM, Hamitouche C, Dillenseger JL, Coatrieux JL; "A Moment based three dimensional edge operator"; IEEE Trans. Biomed. Eng.; Vol. 40; Issue 7; pp. 693-703; 1993.
[13] Christian Bähnisch, Peer Stelldinger, and UllrichKöthe; "Fast and Accurate 3D Edge Detection for Surface Reconstruction"; LNCS 5748; pp. 111–120; Springer; 2009.
[14] Zhao Yu-qian, Gui Wei-bua, Chen Zhen-cheng, Tang Jing-tian, and Li Ling-yun; "Medical images edge detection based on mathematical morphology"; Proc. 27th Annual Conf. EMBC; Shanghai; China; 2005.
[15] J. Amanatides and A. Woo; "A Fast Voxel Traversal Algorithm for Ray Tracing"; Proceedings of the 8th Euro graphics Conference; Amsterdam; NL; August 24-28; 1987; 3-10.



Courtesy: Pentagram Technologies, Hyderabad, India