ALGORITHMS FOR GENERATING CONVEX POLYHEDRONS OVER THREE DIMENSIONAL RECTANGULAR GRID

G. Ramesh Chandra[1] and E.G. Rajan[2]

[1]Associate Professor, Department of Computer Engineering,
V.N.R. Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India.
rameshchandra_g@vnrvjiet.in
[2]Founder President, Pentagram Research Centre, Hyderabad, India
rajaneg@yahoo.co.in

ABSTRACT

This paper discusses various algorithms related to generation of 3-D convex polyhedrons. These convex polyhedrons can be used as structuring elements in the mathematical morphological processing of 3-D digital images. The first algorithm proposes a new way of automatic construction of 256 convex polyhedrons in a 3-D rectangular grid by removing the duplicate subsets. The second algorithm proposes a way for hierarchy construction in visualizing the relationships between sets and their corresponding subsets of all 256 convex polyhedrons. The third algorithm proposes a way of visualizing all 256 convex polyhedrons depending on user selection.

KEYWORDS

Geometric Filters, Subset construction, 3-D structuring elements, 3D rectangular grid, Hierarchy construction, 3-D image processing.

1. INTRODUCTION

In very broad set-theoretic terms, a system is a filter which has a separate, to partition a given set of entities in to disjoint subsets, based on whether they have or do not hat a stipulated property. One does, in principle, conceive of a procedure for such a partitioning, one that is a function of time and space, and which depends for its realization on physical device, be it a sieve, or on electronic circuit, or a computer.

In the study and design of such filtering procedures for signals, one crucial underlying principle seems to be the following:
"System models of signal processing operations and those of sources from which signals originate should belong to the same class"

This principle was introduced by Rajan and Sinha , and is referred here as "R-S Principle(RSP)". It is implied here that signals are to be thought of as outputs of sources, and that their description is not really complete unless we specify source models.

The existing time invariant systems which can serve as filters, are governed by differential equations do not serve as canonical models. These systems of filter theory cannot be able to serve the purpose of efficient image processing operations. In general, complex systems that are

DOI : 10.5121/sipij.2012.3214

traditionally modelled in terms of differential equations and partial differential equations could be ideally modelled and studied in the framework of cellular automata.

Alternatively, one can create generalized spatial domain techniques for processing images and pictures, which make use of shape based filters. Morphological image processing is one such technique, which deals with processing of form and structure of digital images. Yet this technique does not measure up to fulfil the requirements of shape based filtering similar to traditional frequency based filtering. The question that arises now is whether it is possible to develop a technique or at least set a trend which leads to the formulation of a theory for shape based filtering. The theoretical foundations for such shape based filters are studied in the framework of Geometric filters also called G-Filters. The G-Filters are practically implemented in the processing of 2-D images to effectively and efficiently identifying the shape features such as corners, curves, lines, dots and skeleton. The Logical Image processing system (LIPS) version 3.0 implemented these G-Filters for efficient shape feature detection for 2-D images.

There are no such techniques formulated till now for the 3-D images under the rectangular grid. The algebraic intricacies and algorithms development to simulate these theoretical foundations of 3D G-filters are separated in to two papers.

The paper [11] discusses the theoretical foundations of 2-D and 3-D G-Filters over the rectangular grid. The same paper initially discusses the 2-D Geometrical filters, and then proposes the 3-D Geometrical filters (3-D G-Filters) and their algebra.

In this paper we deal with the implementation and algorithms related to 3-D G-Filters. The algorithm in the section-2 discusses automatic generation of convex polyhedron sets and the corresponding subsets. In Section-3, we discuss the algorithm for subset construction under a specific set and their relationships formed in hierarchical fashion. Section-4 discusses the algorithm for generating and visualizing the 3-D convex polyhedron called 3-D structuring element.

2. ALGORITHM FOR CONSTRUCTING COMPLETE DISTRIBUTIVE LATTICE SETS AND SUBSETS OF POLYHEDRON FOR RECTANGULAR GRID

The following formatting rules must be followed strictly. This (.doc) document may be used as a template for papers prepared using Microsoft Word. Papers not conforming to these requirements may not be published in the conference proceedings.

Consider a 3x3x3 rectangular grid, in which there are 27 neighbourhood elements including central pixel. The central idea is to construct 256 unique convex polyhedrons in a 3-D rectangular grid(3x3x3 array of cells), by dropping one corner, two corners, three corners, four corners, fiver corners, six corners, seven corners and eight corners at a time with different probable combinations. The initial set A contains all the eight different corner voxels of the 3x3x3 voxel grid. (for more details on numbering the corner voxels refer the paper [11]) i.e.

A= {1, 3, 7, 9, 19, 21, 25, 27}

By dropping one voxel at a time we can construct 8 different subsets as follows:
B1= {3, 7, 9, 19, 21, 25, 27}
B3= {1, 7, 9, 19, 21, 25, 27}
B7= {1, 3, 9, 19, 21, 25, 27}
B9= {1, 3, 7, 19, 21, 25, 27}
B19= {1, 3, 7, 9, 21, 25, 27}
B21= {1, 3, 7, 9, 19, 25, 27}

B25= {1, 3, 7, 9, 19, 21, 27}
B27= {1, 3, 7, 9, 19, 21, 25}

The dropped voxel can be easily identified by looking at the subset index. For example, the voxel 1 is dropped from the set of corners i.e. A and formed a subset B1, here one can observe that the dropped voxel is considered as a index of the new subset B. For complete list of the 256 convex polyhedrons refer to Paper [11].

For reducing complexity in finding subsets from the set, only one element is removed from the set at each level. For example by removing on element from the set A (level 1) we found subsets B1, B3, B7, B9, B19, B21, B25, B27 (level 2). In the similar way by removing one element from B1 we found C1,3, C1,7, C1,9, C1,19, C1,21, C1,25, C1,27 (level 3). In the level 3 we found 28 such unique C subsets. In the similar way we have 56 D subsets in level 4 by discarding one element from C sets, 70 E subsets in level 5 by discarding one element from D sets, 56 F subsets in level 6 by discarding one element from E sets, 28 G subsets in level 7 by discarding one element from F sets, 8 H subsets in level 8 by discarding one element from G sets, and 1 E subset in level 9 by discarding one element from H sets. When all the subsets are properly arranged we get a complete distributive lattice structure. The total number of convex polyhedrons is calculated by adding all the subsets i.e. 256. The list of all the subsets and their corresponding elements are discussed in paper [11].

So this section discusses the algorithm for generation of subsets from the set A.

Algorithm: Subset Construction

```
Input: Set A. i.e. A= {1, 3, 7, 9, 19, 21, 25, 27}
Output: 255 subsets
Steps:
Step1: Declare a array a[ ] and initialize it with
      {1, 3, 7, 9, 19, 21, 25, 27}
Step 2:
For Subset Group B:
for ( int i = 0; i <8; i++)
  {
      for (int j = 0; j <8; j++)
      {  Print B+ a[i]+"{="
          if ((j == i) )
          {
             //Don't do anything
          }
         else
         {
           Print a[j] + ","
         }
      } print "}\n"
  }
Group C:
for(int i=0;i<=7;i++)
{
  for (int j = i + 1; j <= 7; j++)
  {
    Print C+a[i]+a[j]+"{="
    for (int d = 0; d <= 7; d++)
    {
      if (d == i || d == j)
      { //Don't do anything
      }
```

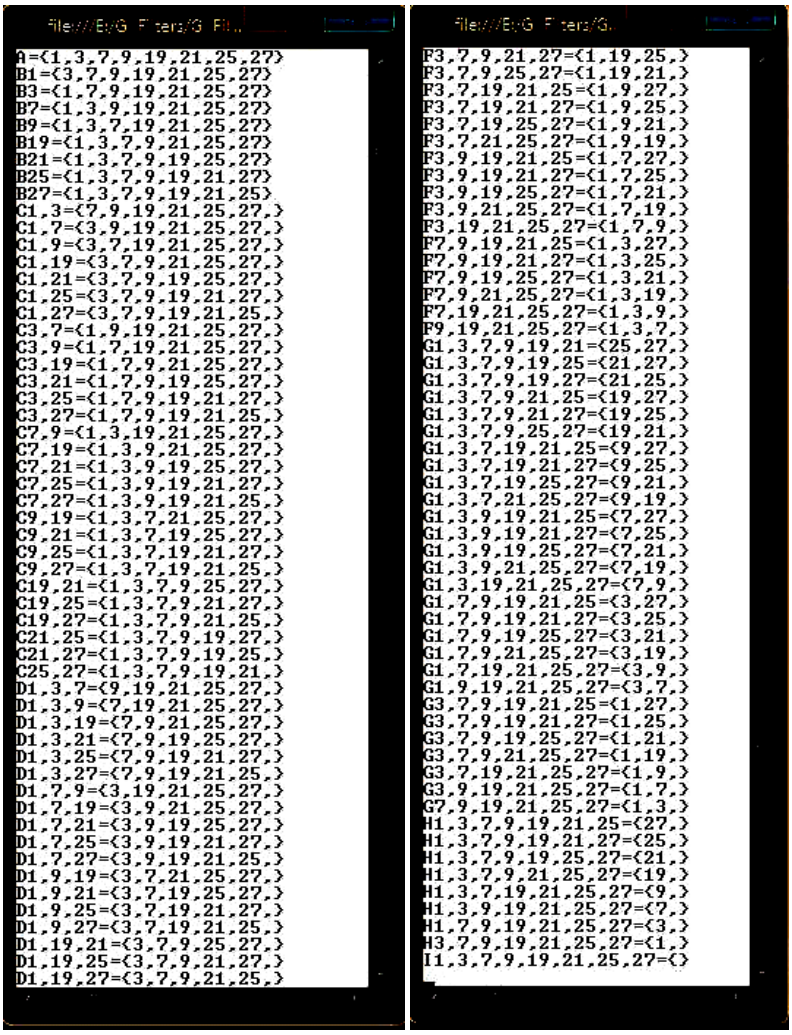
```
            else
            {
              Print a[d]+","
            }
      }
    } print "}\n"
}
```

Similarly the group D subset can be generated with four for loops. Expect out for loop and inner most for loop every for loop is intialized to its previous for loop variable+1.

For Group E→ 5 Nested for loops
Group F→ 6 Nested for loops
Group G→ 7 Nested for loops
Group H→ 8 Nested for loops
Group I→ 9 Nested for loops

****

The output of the above said algorithm is discussed in the results section i.e. Section 5.



(a) (b)

Figure 1: (a) Results window showing subsets from A to D1, 19, 27
(b) Results of subsets showing F, G, H and I

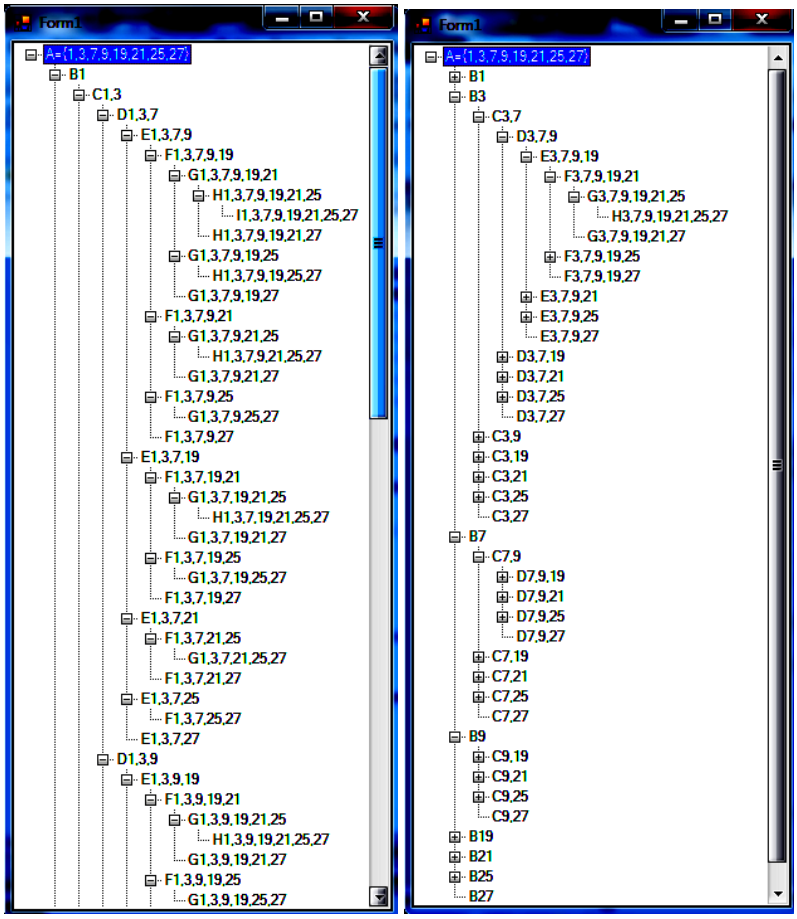
3. ALGORITHM FOR CONSTRUCTING HIERARCHICAL RELATIONSHIPS AMONG SETS AND THEIR CORRESPONDING SUBSETS

The purpose of this algorithm is to make relationships among sets and their subsets. As discussed in the previous section the sets C1,3, C1,7, C1,9, C1,19, C1,21, C1,25, C1,27 are subsets of B1. This algorithm will construct a tree view of the relationships mention above such sets. The algorithm present in this section will form a hierarchical relationship among sets and subsets. The hierarchical relationships are shown in the Figure 2(a) & 2(b).

Algorithm: Hierarchy construction

```
Input: Set A. i.e. A= {1, 3, 7, 9, 19, 21, 25, 27}
Output: Tree view of sets and their subsets
Steps:
Step1: Declare a array a[ ] and initialize it with
      {1, 3, 7, 9, 19, 21, 25, 27}
Step 2:
For Subset Group B:
Create Empty tree
Add set A as root node to the tree
for ( int i = 0; i <8; i++)
  {
Create Empty BNode
Name node as  "B" + a[i];
Add the node generated to the
         Root node i.e. A.
       for (int j = 0; j <8; j++)
       {
         // Logic for Subset Creation
       }
}
Group C:
for(int i=0;i<=7;i++)
{
  for (int j = i + 1; j <= 7; j++)
  {
Create Empty CNode
Name node as  C+a[i]+",”+a[j]
Add the node generated to the
Parent node as follows:
   mainNode.Nodes[i].Nodes.Add(CNode);
    for (int d = 0; d <= 7; d++)
    {
      // Logic for Subset Creation
    }
  }
}
Group D:
for (int i = 0; i <= 7; i++)
{
   for (int k = i + 1; k <= 7; k++)
   {
     for (int j = k + 1; j <= 7; j++)
     {
Create Empty Node
Name node as D+a[i]+a[k]+a[j]
Add the node generated to the
```

```
Parent node as follows:
            m = ((k - 1) - i);
mainNode.Nodes[i].Nodes[m].
Nodes.Add(DNode);
    for (int d = 0; d <= 7; d++)
    {
      // Logic for Subset Creation
    }
  }
}
Similarly the group E subset can be generated with five for loops.
For Group F→ 6 Nested for loops
    Group G→ 7 Nested for loops
    Group H→ 8 Nested for loops
  Group I→ 9 Nested for loops
                ************************************
```



(a) (b)

Figure 2:

(a) The hierarchical relationships showing upper part of the results

(b) The hierarchical relationships showing lower part of the results.

4. ALGORITHM FOR CONSTRUCTING 3-D IMAGES OF ALL THE 256 CONVEX POLYHEDRONS

The notion of structuring elements plays a vital role in processing the 2D/3D images using mathematical morphological operations like erosion, dilation, opening, and closing. While most of the textbooks [5-8] and papers [1-4] discuss 2-D structuring elements and their extended notion of 3-D structuring elements, this paper has introduced a novel way of systematically constructing 3x3x3 structuring elements. Note that the structuring elements are constructed by dropping chosen cells of the eight corner cells only because the resulting structuring elements would form geometric convex shapes. Alternatively, if one drops cells other than the eight corner cells, the resulting structuring element would have a geometric concave shape.

Generally, concave shapes are not used to perform morphological operations. The algorithm given below could be used for constructing 3x3x3 structuring element images.

Algorithm: Constructing Structuring Element Images

Input: List of all 256 convex polygons in XML file with the following Details:

- Group name such as A, B, C, D, E, F, G, H and I.
- Name of convex polyhedron.
- Elements of the convex polyhedron.

Output: 3D images

Steps:

Step 1: Declare a vertex structure which contains information such as position, color and normal of the vertices.

Step 2: Setup camera details such as field of view, position of the camera, viewing angle etc. Also enable the Z-buffer.

Step 3: Initialize the graphics device, vertex buffer, index buffer, light direction and effect file.

Step 4: Create the vertex information and index information for all the 27 voxels in the structuring element.

Step 5:

```
// Load all the voxels in the cube
for(float z = 1; z > -2; z--)
{
   for(float y = 1; y > -2; y--)
   {
     for(float x = -1; x < 2; x++)
     {
//Give Red color to front plane, Green colour to center plane and blue color to back plane.
// List the voxel numbers which you would like to remove from the structuring element.
     }
   }
```

Step 6: Place vertex information and index information on the device object created earlier.

Few examples of convex polyhedrons constructed in a 3X3X3 grid of pixels are shown in Figure 3. Red represents front plane (k-1), Green represents central plane (k), Blue represents rear plane (k+1).

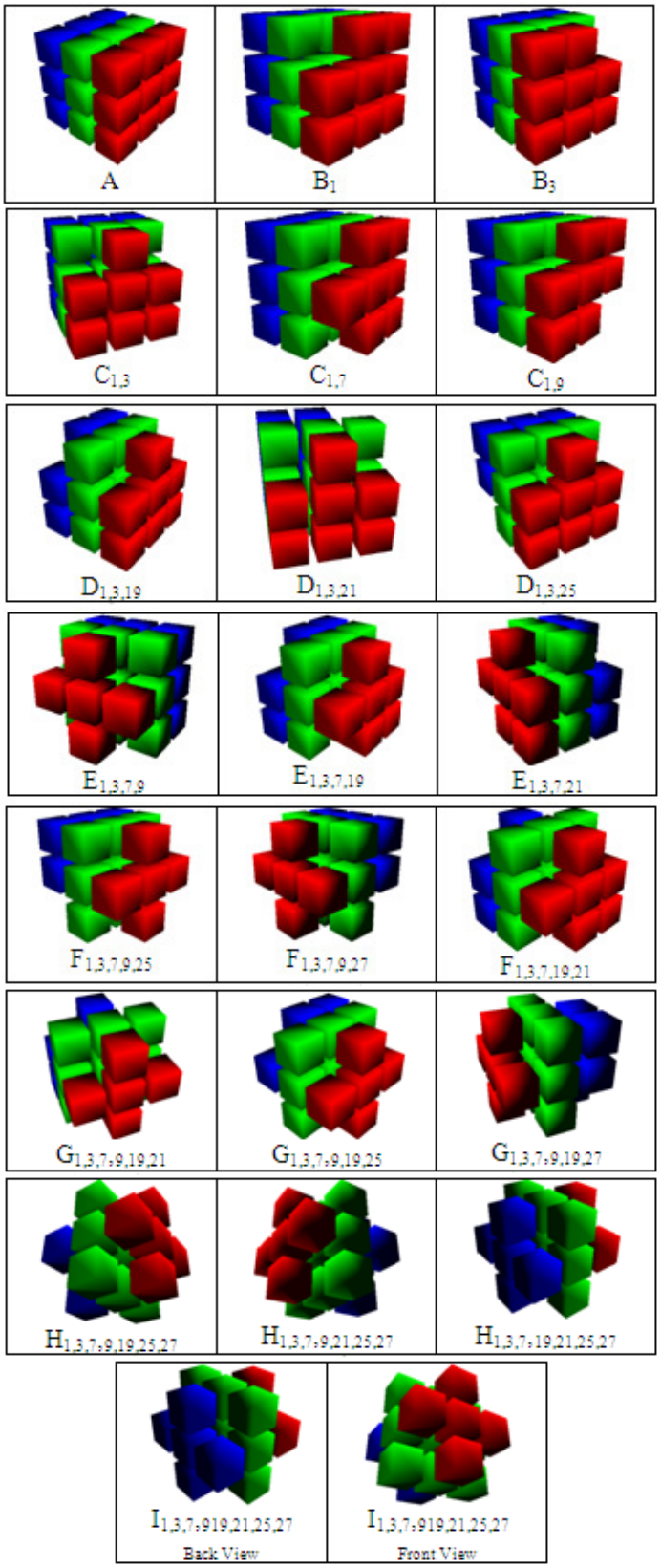


Fig 3: Convex Polyhedrons

5. RESULTS

The algorithms discussed in the section II and III are implemented in C#.NET language. The results of the sub set construction are shown in the Fig 1(a). The Fig 1(a) shows results of some of the subsets. The other subsets ending with F, G, H and I are shown in the Fig 1(b). The algorithm discussed in the section II is able to generate all 255 convex polyhedrons. The complete list of convex polyhedrons cannot be listed here as output. So the partial list is shown in the Fig 1(a) & 1(b). The results of algorithm discussed in the section III is shown in Fig 2(a). This figure shows the relationships among sets and their corresponding subsets. The Fig 2(b) shows the lower part of the some more results of the algorithm for hierarchical subset construction. By looking at the output one can clearly understand that H is subset of G, G is subset of F, F is subset of E, E is subset of D, D is subset of C, C is subset of B and finally B is subset of A. Algorithm in the section IV discusses the generation and visualization of convex polyhedrons. Few of the 3-D convex polyhedrons are shown in the figure 3.

6. CONCLUSIONS

Manual construction of 256 convex polyhedrons in a 3X3X3 rectangular grid is cumbersome and highly error prone. One may end up with constructing subsets repeatedly and above all it has been empirically verified that constructing all 256 convex polyhedrons does consume 48 man hours. To overcome this problem, we proposed a new algorithm for subset construction. This algorithm is discussed in the section II, which generates 255 polyhedron subsets in a fraction of a second.

Secondly, manual construction of paths in the lattice from supremum to infimum is almost impossible task [11] [12]. Each path defines a GE filter. Each GE filter is of some use in processing 3-D images. So, one comes across the usual problem of enumerating potentially very large GE filters defined over a finite set. To overcome this problem, we proposed in this paper a novel algorithm for Hierarchical Path Enumeration.

Finally, we discussed about the algorithm for generating convex polyhedrons and visualizing it as a 3x3x3 cube. These convex polyhedrons can be used as 3-D structuring elements, whose application is to process the 3-D images. These structuring elements can be visualised based on the section by the user, for example if we select B1 then corresponding structuring element is generated as shown in the figure1.

ACKNOWLEDGEMENTS

The constant support and encouragement extended by Mrs. G. Chitra, Managing Director of Pentagram Research this original treatise on Geometric Filters would not have taken this shape. The authors are very much thankful to her and to the entire organization for the support.

REFERENCES

[1] Rafael C. Gonzalez & Richard E.Woods, Digital Image processing, Second Edition, PHI.

[2] Rajan E.G., Symbolic Computing-Signal&Image Processing, BS Publications, 2003, India

[3] Choquet G., Topology. New York: Academic, 1966.

[4] Rajan E. G. The notion of geometric filters and their use in computer vision, 1995 IEEE International Conference on Systems, Man and Cybernetics, Vancouver, B.C., Canada, 1995, pp 4250-4255.

[5] Rajan E. G., On the notion of a geometric filter and its relevance in the neighbourhood processing of digital imagesin hexagonal grids, 4th International Conference on Control, Automation, Robotics and Vision, Westim Stamford, Singapore 1996.

[6] Vasantha N., Rajan E. G., On the notion of geometric filters, National Conference organized by the the Institution of Engineers, India and Annamalai University, SURGE'94, 1994.

[7] G.Matheron,Random sets and Integral Geometry .New York:Wiley,1975

[8] J.Serra,Image Analysis and Mathematical Morphology , New York:Acadamic 1982

[9] P.Maragos and R.W.Scharer, "Applications of morphological filtering to image processings and analysis, " in Proc. 1986 IEEE Int. Conf. Acoust., Speech, Signal Processing, Tokyo, Japan, Apr. 1986, pp. 2067-2070.

[10] Image Processing, Analysis,and Machine Vision By: Milan Sonka, Vaclav Hlavac, and Roger Boyele. Second edition

[11] Towheed Sultana, Rajan E G., Algebra of Geometric Filters Defined over Three Dimensional Rectangular Lattice- Part I, International conference on computer science and information technology(CCSIT-2012), Bangalore, India, pp.101-111.

[12] G. Ramesh Chandra, Rajan E G., Algorithm for Constructing Complete Distributive Lattice of Polyhedrons Defined over Three Dimensional Rectangular Grid- Part II, International conference on computer science and information technology(CCSIT-2012), Bangalore, India, pp.202-210.

AUTHORS PROFILE

Mr. G. Ramesh Chandra did his M.Tech in Computer Science and Engineering from JNTU, Hyderabad. Presently he is working as associate professor in the department of computer science, V.N.R. Vignana Jyothi Institute of Engineering and technology, Hyderabad. He is currently pursuing his PhD in Jawaharlal Nehru Technological University, Hyderabad. He awarded as "Distinguished Official" for successful conducting of International Conference ICSCI-2011. He developed a compiler for "Xervo Script Engine" for Cyber Motion Technologies, Hyderabad. He presently involved in developing Logical Image Processing System 4.0 and Logical Pattern Generation System 3.0. He published 5 papers in international journals and 7 papers in international conferences. He has 11 years of teaching experience.

Prof. Dr. E. G. Rajan is an Electronics Engineer and a Professor of Signal Processing having about 30 years of experience in teaching, research and administration. He is an editor of the journal of AMSE, Barcelona, Spain. He received his Ph.D degree in Electrical Engineering, from Indian Institute of Technology (IIT), Kanpur, U.P., M.E. degree in Applied Electronics, from Madras University, the then famous DMIT in Electronics Engineering from the Madras Institute of Technology, Chromepet, Madras and B.Sc degree in Physics from Madras University. His contribution to the state-of-the-art of Electronic Warfare has been well recognized in the Government and industrial sectors. He was a noted teacher in the department of Electrical Engineering of the Indian Institute of Technology, Kanpur. He received Distinguished Scientist and Man of the Millennium Award from Who is Who Bibliographical Records, Cambridge, 2000. He authored many books like, Symbolic Computing – Signal and Image Processing, Electronic Order of Battle Records of Military Radars, Computers and Information Technology and Foundations of Information Technology. Two of his fifteen Ph.D scholars are involved in the design of digital circuits using organic molecules and some of them are working in the novel area of Genomic Signal Processing. He has brought out more than 25 original concepts. One concept is a Mathematical Transform, which goes by his name as Rajan Transform. This transform is ideally used for Pattern Recognition and Analysis purposes. He leads the company in his own professorial style and the company is technically strong and steady. Prof. Rajan is now working on multiple valued logic and (2n+1)-ary discrete algebras supported by VLSI implementation of multiple valued logic gates and switching circuits.