Summer Internship 2016

Mentored by – Dr. Amit Chattopadhyay

Marching Cube and Marching Tetrahedron Algorithm

Presented by :-
Samarth Mishra

23 May 2016

Abstract

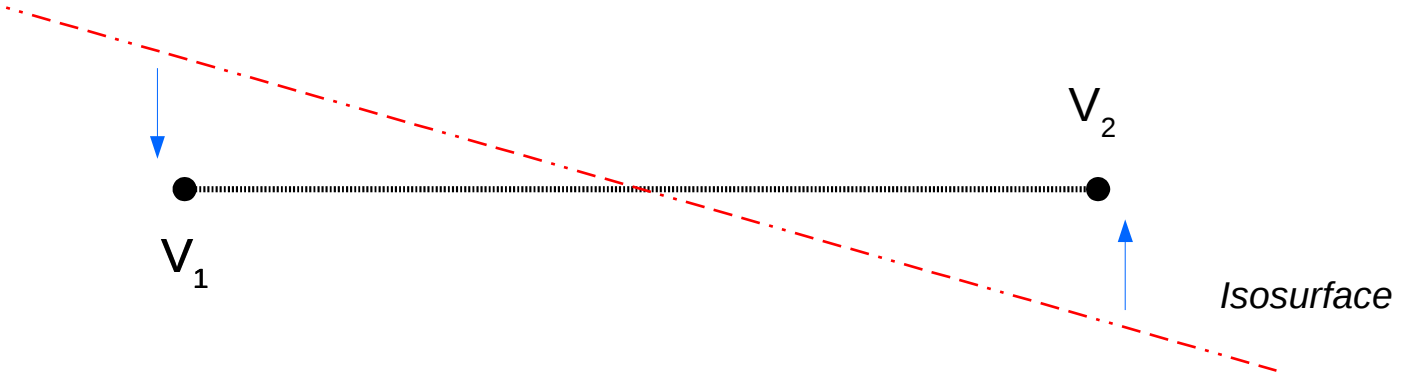
- This algorithm was developed by William E. Lorensen and Harvey E. Cline.
- Marching cube is a computer graphics algorithm that was cited in 1987 in a conference called *SIGGRAPH (Special Interest Group on GRAPHics and Interactive Techniques).*
- **Objective:** extracting polygonal mesh of an isosurface from a three dimensional discrete scalar field (called *Voxels*)
- **Applications:** Medical Visualization like CT and MRI scan data images, single-photon emission computed tomography (SPECT)

Marching Cube Algorithm

Algorithm for creating a polygonal surface representation of an isosurface of a 3D scalar field.

Basis: form facets(place faces) approximation to an isosurface through a scalar field sampled on a 3D rectangular grid.

Each possibility is characterised by the number of vertices that have values above or below the isosurface.

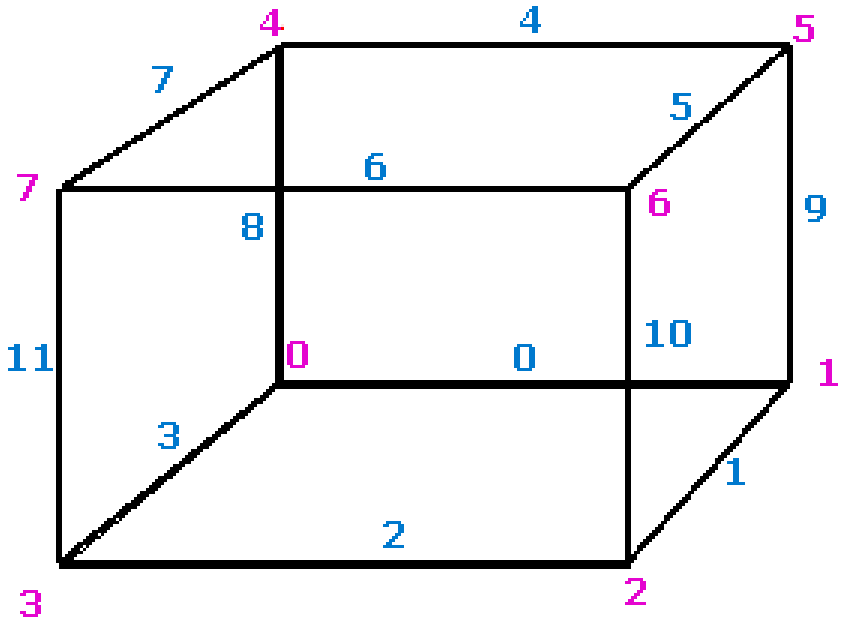


- If one vertex is above and another is below then we know isosurface cuts the edge between these two vertices.

  Where exactly will the surface touch/cut the edge?
  -> It depends on the relationship between the values at V_1 and V_2 and isolevel value. That will give the ratio of the length.

Index Convention

--- Edge Index
--- Vertex Index



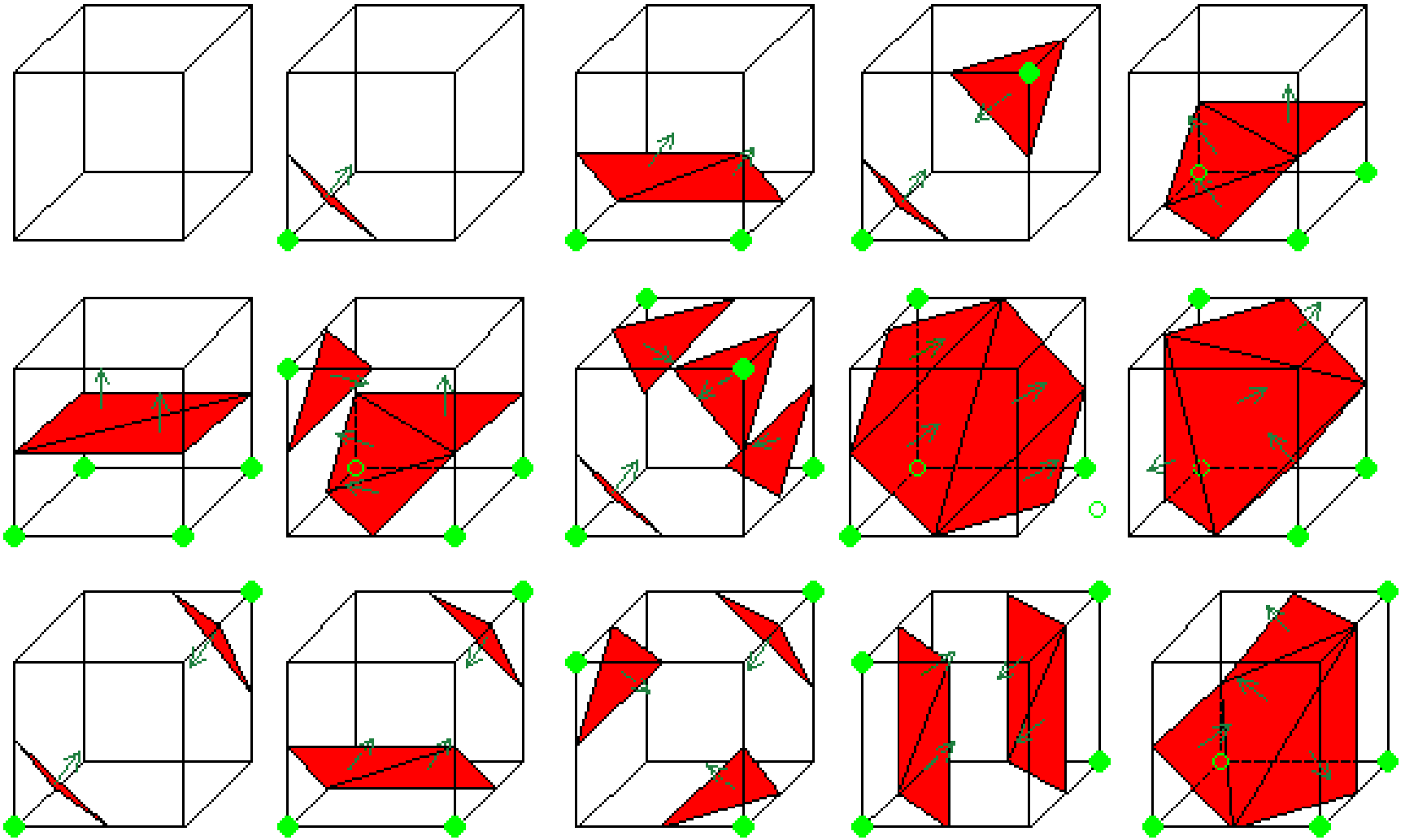
Total Number of Conditions

We have 8 vertices. Any number of them can lie above and thus the rest will lie below the isosurface.

If we see the total number of combinations we get

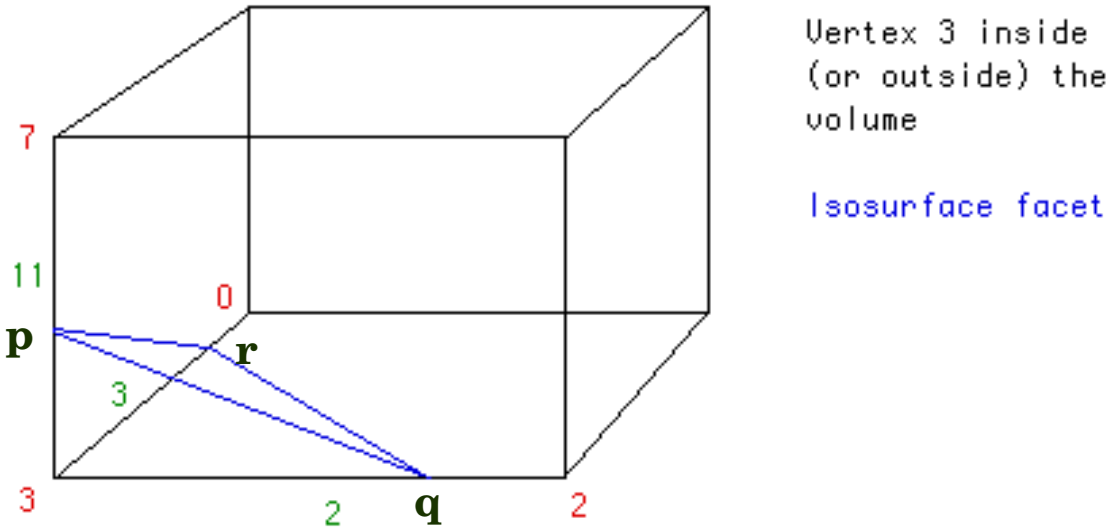
$$^8C_0 + ^8C_1 + ^8C_2 + ^8C_3 + ^8C_4 + ^8C_5 + ^8C_6 + ^8C_7 + ^8C_8 = 2^8 = 256$$

In a 3D space we enumerate 256 different situations for the marching cubes representation. All these cases can be generalized in 15 families by rotations and symetries :



Example

Vertex 3 is below isosurface and others are above.



Now to find points p,q and r. We need to know the relationship between V_3-V_7,V_3-V_2,V_3-V_0

Edge Table

The first part of the algorithm uses a table (edgeTable) which maps the vertices under the isosurface to the intersecting edges. An 8 bit index is formed where each bit corresponds to a vertex.
cubeindex is an 8 bit binary number formed after the following operation. Each bit corresponding to a vertex.

V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0

```
int cubeindex = 0;
   if (grid.val[0] < isolevel)
        cubeindex |= 1;
   if (grid.val[1] < isolevel)
        cubeindex |= 2;
   if (grid.val[2] < isolevel)
        cubeindex |= 4;
   if (grid.val[3] < isolevel)
        cubeindex |= 8;
   if (grid.val[4] < isolevel)
        cubeindex |= 16;
   if (grid.val[5] < isolevel)
        cubeindex |= 32;
   if (grid.val[6] < isolevel)
        cubeindex |= 64;
   if (grid.val[7] < isolevel)
        cubeindex |= 128;
```

Generalized code

```
for ( int i=0 ; i< 8; i++)
{
if (grid.val[i]<isolevel) //condition check
cubeindex = cubeindex | pow(2,i); //OR operation
}
```

Edge Table

Now checking in the edgeTable[ ], it returns a 12 bit number telling which edges are intersected.

E_{11}	E_{10}	E_9	E_8	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0

Eg:-
cubeindex = 8;
EdgeTable[8] = 1000 0000 1100;
Here it shows E_2,E_3,E_{11} are intersected by the isosurface.

Edge Table

Now checking in the edgeTable[ ], it returns a 12 bit number telling which edges are intersected.

E_{11}	E_{10}	E_9	E_8	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0

Eg:-
cubeindex = 8;
EdgeTable[8] = 1000 0000 1100;
Here it shows E_2,E_3,E_{11} are intersected by the isosurface.

triTable

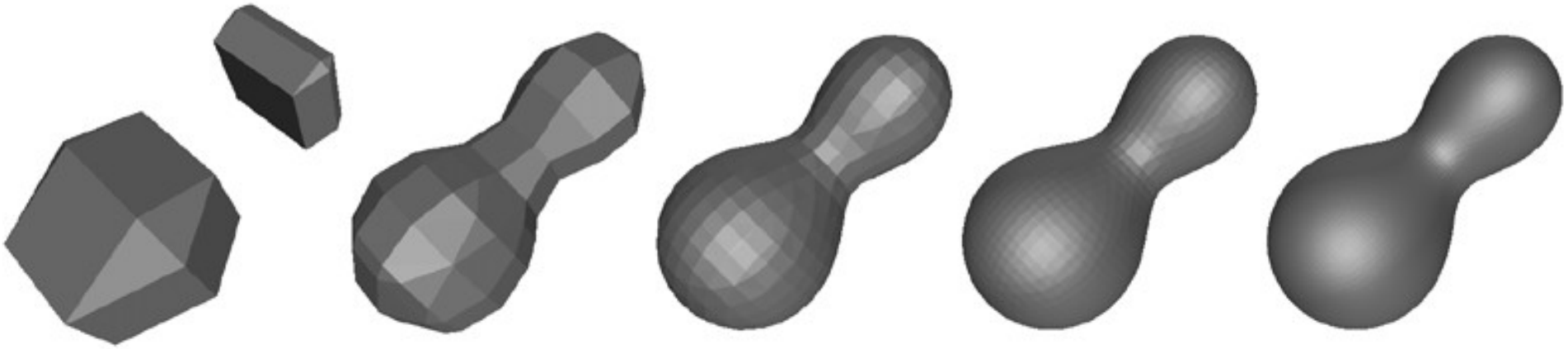
- uses the same **cubeindex** but allows the vertex sequence to be looked up for as many triangular facets are necessary to represent the isosurface within the grid cell. There at most 5 triangular facets necessary.
  TriTable[256][16] -> 256 entries and each entry has 16 slots.

- in the previous step we calculate the intersecting points along edge 2,3, and 11. The 8th element in triTable is :
  8^{th} element :
  {3, 11, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1}

- 9^{th} element :
  {0, 11, 2, 8, 11, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
  This corresponds to 2 triangular facets, one between the intersection of edge 0 11 and 2. The other between the intersections along edges 8 11 and 0.

Grid Resolution

- Fine approximation to the isosurface to be generated depending on the smoothness required and/or the processing power available to display the surface.
- Smaller the size of the grid, the surface obtained is smoother.



Grid size=10
70 Facets

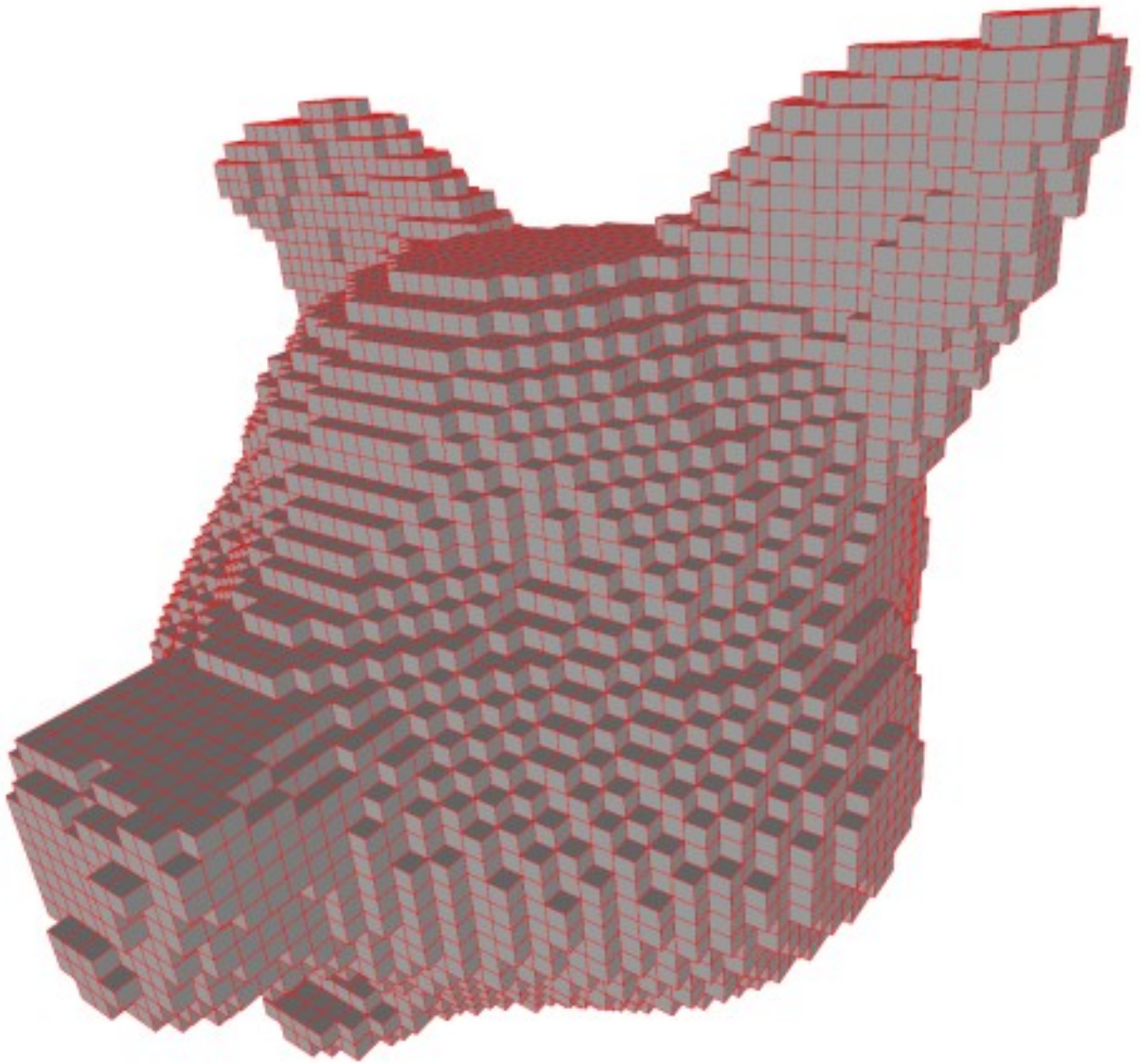
Grid size=5
220 Facets

Grid size=2
1700 Facets

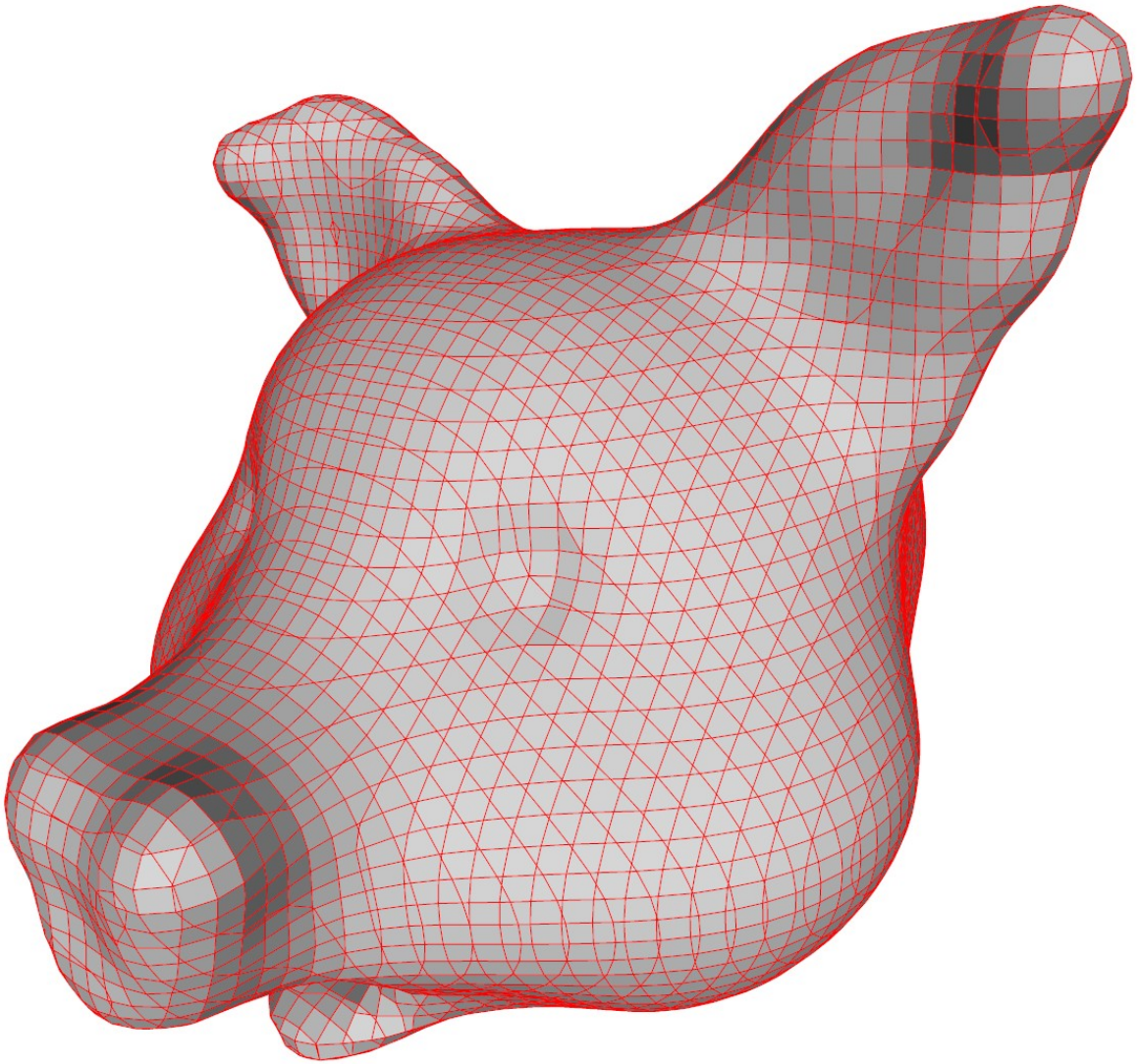
Grid size=1
6800 Facets

Grid size=0.5
27000 Facets

One of the first ideas for visualizing volumes was to draw each voxel as a small cube. If we do that we get something like the result shown below.



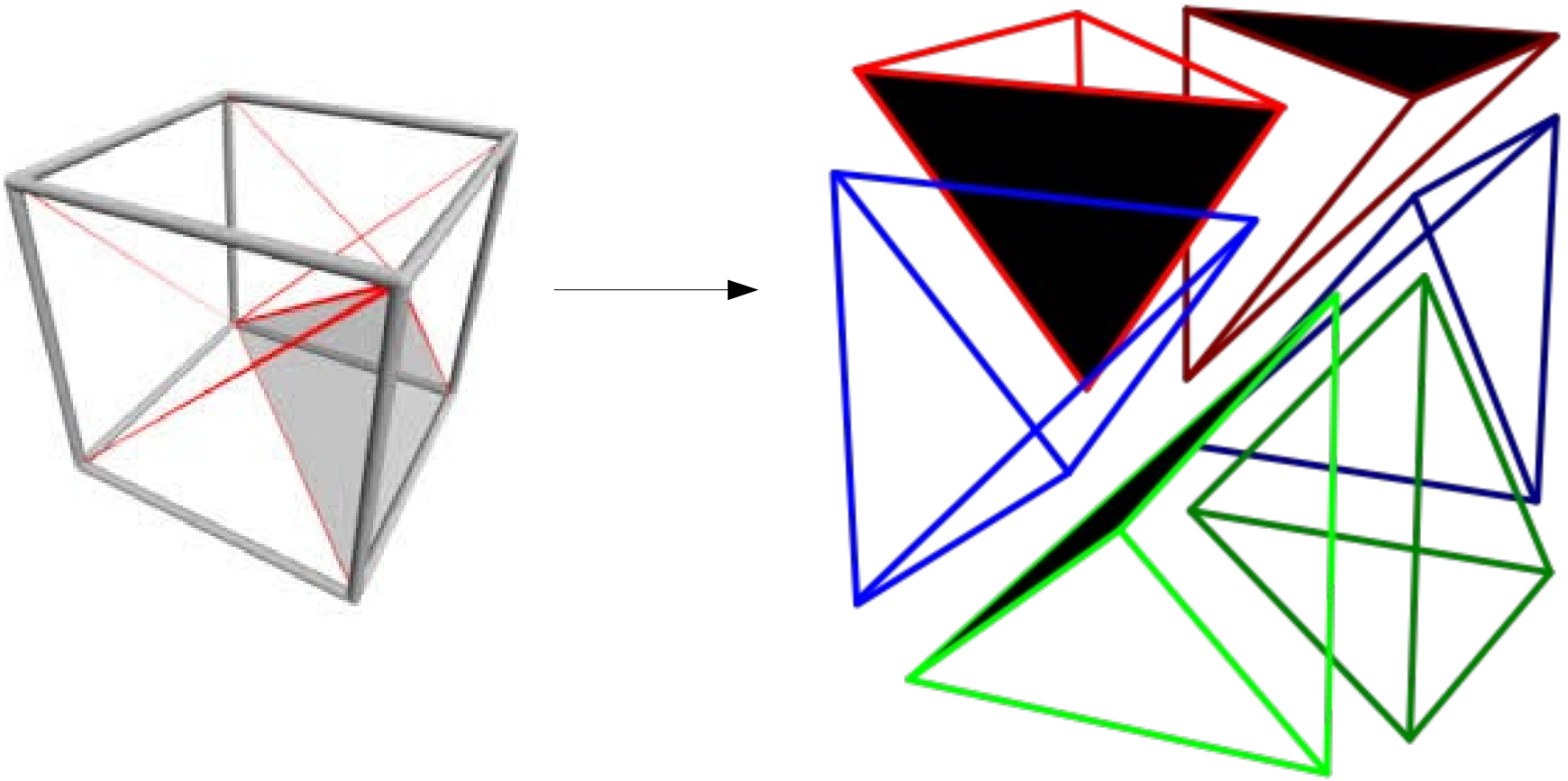
The result which is shown below is, in fact, the precise same result as would have been produced by marching cubes except that the faces have not been polygonized.



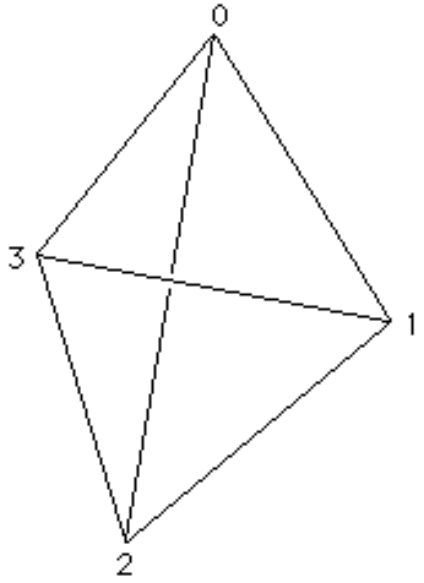
Marching Tetrahedron Algorithm

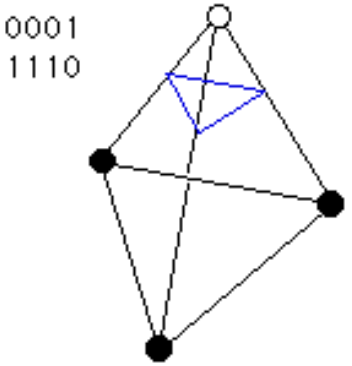
- It clarifies a minor ambiguity problem of the marching cubes algorithm with some cube configurations.
- The space is sampled at the vertices of a rectangular 3D mesh. Each mesh cell is split into 6 tetrahedrons and passed to the tetrahedron isosurface function.
- one could equally sample onto the tetrahedral mesh directly.
- The planar facet approximation to the isosurface is calculated for each tetrahedron independently.
- The facet vertices are determined by linearly interpolating where isosurface cuts the edges of the tetrahedron.

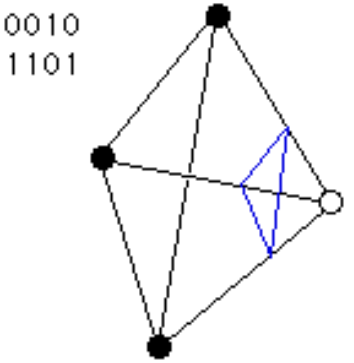
Cube divided in to Tetrahedrons

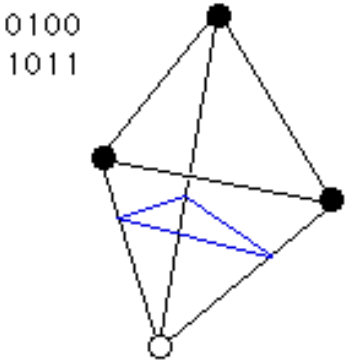


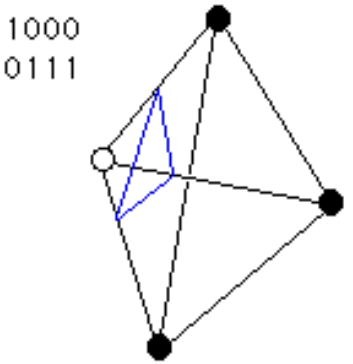
Cases

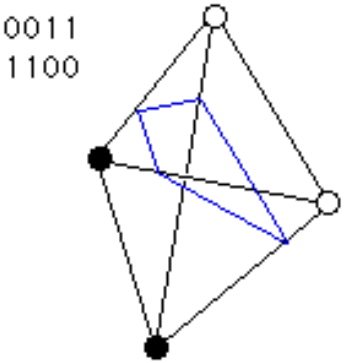

0
1
2
3

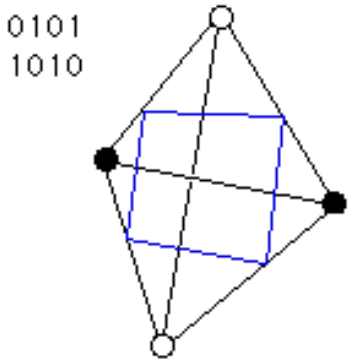

0001
1110

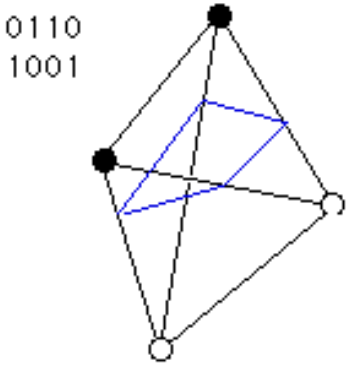

0010
1101


0100
1011


1000
0111


0011
1100


0101
1010


0110
1001

Advantages of M.T. over M.C.

- M.T. does not suffer from the ambiguities in the traditional marching cubes algorithms.
- Lesser number of cases in M.T.

M.T. images