International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume – 03, Issue – 03, March 2018, PP – 24-31

Surface Reconstruction & Shape Representation

Rajiv Kumar

Punjab Institute of Management & Technology (Near GPS, Mandi Gobindgarh), Alour, Khanna, Punjab, India

Abstract: Surface reconstruction and shape representation of objects from the various location points, in 2D and 3D, is an important topic of research. The work is motivated by number of applications in many fields like computer vision, graphics, surface design and 3D reconstruction. The objects are represented as three coordinates of a set of points obtained mainly from scanning the objects with range scan, ultrasound scan etc. The present work focuses on the use of triangular patches to represent surface and shape of the object.

Key words: Marching Cube, Surface reconstruction, Triangulation

I. Introduction

The area of contouring, surface reconstruction and shape representation has received significant attention in computational geometry and graphics.

Contouring

Graphical representation of results is essential to understand numerical model behavior. The simple visual inspection, as facilitated by contour plotting, substitutes the large data analysis process. The term contour is defined mathematically as f(x,y)=c. This is the plot/curve of f(x,y) function at some constant value c. Contour distinguish different regions by showing their boundaries. Contours are used to represent the mechanical properties, material properties at some specific value. The contours can be generated with the help of contouring algorithm. The contouring algorithm treats the input matrix Z=f(x,y) as a regularly spaced grid, with each element connected to its nearest neighbors. The algorithm scans this matrix comparing the values of each block of four neighboring elements (i.e., a cell) in the matrix to the contour level value i.e. c. If a contour level falls within a cell, the algorithm performs a linear interpolation to locate the point at which the contour crosses the edges of the cell. The algorithm connects these points to produce a segment of a contour line. Present work deals with the implementation of marching square as contour plotting algorithm.

Marching Square Algorithm

Marching squares is a computer graphics algorithm that generates contours for a two-dimensional scalar field (rectangular array of individual numerical values). The algorithm generates lines and vertices as output. The lines are then used to represent the contour desired for the scalar data. The algorithm is implemented as detailed below:

1. Select a cell

a. Calculate inside/outside state for each vertex (x_i,y_j) as contour can pass through a cell in a finite number of ways:

inside contour if scalar value (f_{ij}) > contour (isoval)

outside contour if scalar value (f_{ij})< contour (isoval)

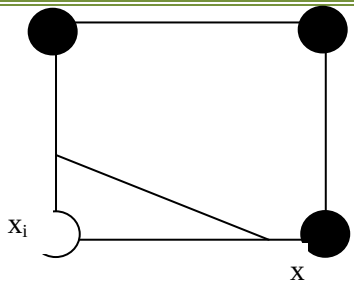
b. Create an index by storing binary state of each vertex (in or out) in a separate bit

c. Use index to lookup topological state of cell in a case table

d. Calculate contour location (geometry) for each edge via linear interpolation as per equation 1

As there is need to calculate the mid-point between x_i, x_{i+1} and y_j, y_{j+1}.

$F_{i,j}$ = a is closer to isoval than b = $f_{i+1,j}$ then intersection is closer to (x_i, y_j):



$f_{i,j}$= a < c c < b= $f_{i+1,j}$

$$(x-x_i)/(x_{i+1}-x) = (isoval-a)/(b-isoval) \quad \ldots (1)$$

e. Connect with straight line

2. March to next cell (order/direction non-important)
3. Need to merge co-located vertices into single polyline

Surface Reconstruction

The surface reconstruction from a set of points is most favorable research problem in the field of computer graphics. The points representing the surface of an object are obtained from scanning the objects or from using implicit functions for the object. These points represent the <x,y> in case of two dimensional or <x,y,z> in case of three dimensional object. The goal is to represent the surface of object by using simple geometric elements like lines, triangles or tetrahedral iteratively. The volume visualization has been applied in many problem domains and has become an important tool for exploring data and discovering knowledge. Many methods exist for this; here the focus is on Marching Cubes algorithm for surface reconstruction of the object from its volumetric dataset.

Marching Cube Algorithm

Marching Cubes [6] uses a divide-and-conquer approach to locate the surface in a logical cube created from eight pixels; four each from two adjacent slices. The algorithm determines how the surface intersects this cube, then moves to the next cube. This approach is also known as *iso-surface extraction*. [7] An iso surface of interest has a property S_F ={(x,y,z) : T(x,y,z)= 0}. To find the surface desired in a cube, we assign a one to the cube's vertex if the data value at that exceeds or equals the iso-value of the surface we are constructing. These vertices are inside or on the surface desired. Cube vertices with values below the surface receive a zero and are outside the surface. The surface intersects those cube edges where one vertex is outside and other is inside the surface. This determines the topology of the surface within a cube. The intersection points are approximated using the linear interpolation method: If a unit-length edge E has end points Vs and Ve whose scalar values are Ls and Le, respectively, then given an isovalue iso, the location of intersection I =(I_x, I_y, I_z) has components of the form as shown in equation 2:

$$I_{(x,y,z)} = Vs_{(x,y,z)} + rho(Ve_{(x,y,z)} - Vs_{(x,y,z)}) \quad \ldots (2)$$

where rho=(iso- Ls)/(Le- Ls) and iso is the iso-value (user input)

The intersections can follow any of the 256 cases from an edge table depending on the 8-bit index for each cube/grid cell. This provides the edges that may intersect the iso-surface. The duplicate and ambiguous case were ignored and not considered. By determining which edges of the cube are intersected by the isosurface, we can create triangular patches which divide the cube between regions within the isosurface and regions outside. By connecting the patches from all cubes on the isosurface boundary, we get a faceted surface representation. This approach is also known as voxel visualization. The final step in marching cubes is to calculate a unit normal for each triangle vertex. It is used to create shaded images.

The marching cubes algorithm is implemented in MATLAB script. The program accepts the scalar volume data, iso value (surface value required to reconstruct the surface) as input, then it computes faces and vertices. This geometric information is displayed using the patched trainagulation. The resulted surface depicts the actual object and was reconstructed with speed. The proposed algorithm is tested on various datasets and results are discussed in section III and IV.

II. Related Work

In this section, we focus on mainly used methods for surface reconstruction and shape representation. The focus is on Marching Cube and triangulation method for surface reconstruction along with topology preservation. The main reconstruction algorithms are based on spatial subdivision, distance functions, surface warping and incremental surface growing [1]. The spatial subdivision is based on the fact that the bounding volume around the input data set is divided into disjoint cells. The cells related to the shape of the surface/ object are found. Hoppe et. al.[2] use signed distance function from any point to find the cell on surface. This approach involves Marching Cube algorithm. Marching cube technique includes subdivision of space into small size cubes finding the cube traversed by the surface say f(x,y,z) at a constant scalar value and joining the point of intersections through polyhedral as geometric primitive to generate the iso-surface.

Distance function of a surface gives the shortest distance from any point to the surface. The piecewise linear surface with triangular faces passes through the zero set of the distance function. This approach leads to approximation of surface. Hoppe et. al. [2] estimate a tangent plane, after computing consistent normal using Riemannian graph, throughout the surface using k nearest neighbors and uses the distance to the plane as the signed distance. The zero set of this function is then sampled at grid points and polygonalized using the marching cube algorithm – a spatial subdivision algorithm. The method of Curless[3] is similar but suited for large sized laser range data.

Incremental surface growing mainly focuses on an initial triangle (seed) for surface and continuously attaching more triangles to reconstruct the desire surface. Yu et.al.[4] used similar triangulation method for surface reconstruction. They built the original triangle (initial) near the centre of all the points and increase triangles according to slick rule to reconstruct surface. There is also a combined approach that constructs a geometric data structure such as Delaunay triangles and extracts the facets that approximates the surface [5].

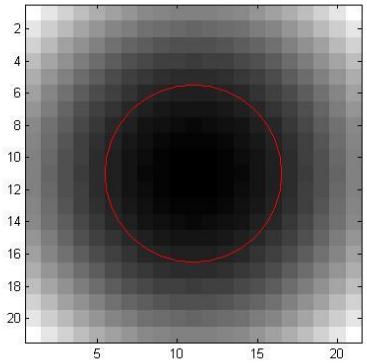
III. Algorithm Implementation & Results

MATLAB implementation of marching squares and marching cubes is used and applied on set of organized uniformly distributed scalar field data. The case of randomly selected points and a scalar value is also tested to predict the desired surface using marching cubes. The MATLAB program is developed for contouring using Marching Square method. It accepts the two-dimensional scalar field and iso-value. The iso-value is used to generate a binary image[8] containing 1 where the data value is above iso-value and 0 where the data value is below the iso-value. This gives us a contouring grid taking 2X2 blocks of pixels. Each cell in the contouring grid is processed to calculate the 4-bit index to pre-built lookup table containing the cases of edges' intersection in the cell. Then linear interpolation is applied to get the exact position of the contour line in the each cell.

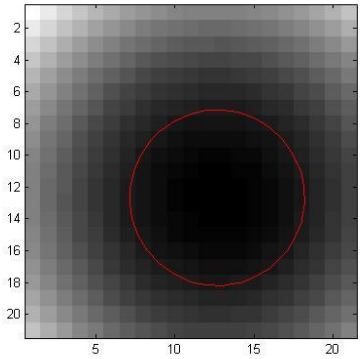
In our work, we used various implicit equations to generate input data to classic Marching Square code written in Matlab and draw their respective contours at some constant value represented as f(x,y)=isoval. The contour is drawn using the plot MATLAB function on coordinates on the vertices where the iso-value or contour value intersects the edge of the square. The contour is drawn for many data samples as detailed in the table 1 below and results are shown in the figure A (FigA1-FigA5). The marching cubes algorithm is implemented using MATLAB7.0 on Dell Core 2 Duo computer on Windows XP. The code takes the scalar volume data, iso value (surface value that is needed to reconstruct the surface) as input, it returns faces and vertices.

Table 1. Contouring Details 2D contours

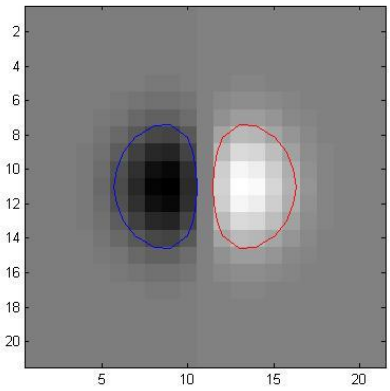
Sr. No.	Data Size	Iso Value	No. of Lines	Shape
1	21X21	0.49412	14	Circle using implicit equation $(x-a)^2+(y-b)^2-r^2$ with a=b=0
2	21X21	0.51373	14	Circle using implicit equation $(x-a)^2+(y-b)^2-r^2$ with a=b=0.5
3	21X21	0.1333	24	Ovals of Cassini using implicit equation: $x.*\exp(-x^2-y^2)$
4	256X256	0.51373	7465	Image as input rice.png
5	17X10	70	24	The iso thermal data of some Iceland



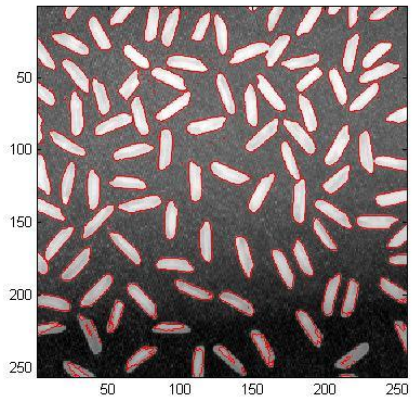
FigA1



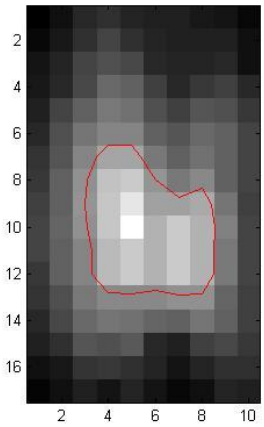
FigA2



FigA3



FigA4



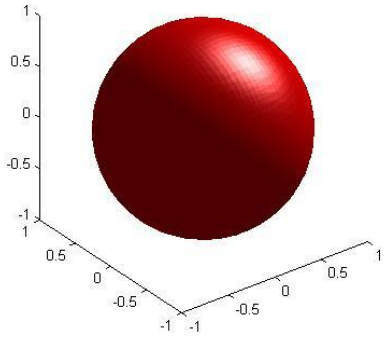
FigA5

Figures A. (FigA1 to FigA5) Reconstructed Surfaces

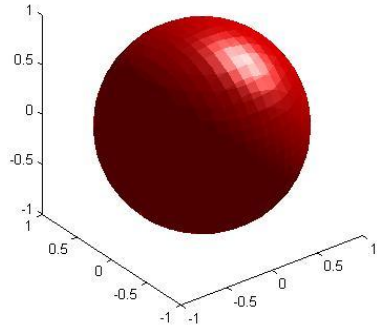
This geometric information is displayed using the patch function of MATLAB. The surface thus generated depicts the actual object and speed of reconstruction is good. Various inputs are tested to reconstruct the surface, see table 2 below and results are shown in figure B (FigB1-FigB4).

Table 2. Contouring details 3D contours

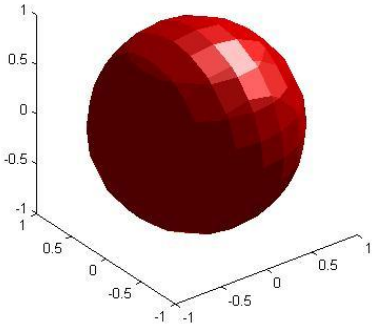
Sr. No.	Data Size	Grid step size	No. of Facets	No. of Vertices	Iso value	Speed of execution	Result
1	81 X 81 X81	0.05	13400	26808	0.65	Moderate	High Resolution (see FigB1)
2	41 X 41 X 41	0.1	3512	7032	0.65	Fast	Good Resolution (see FigB2)
3	17 X 17 X 17	0.25	536	1080	0.68	Fast	Less Resolution (see FigB3)
4	11 X 11 X 11	0.4	248	504	0.63	Faster	Low Resolution (see FigB4)



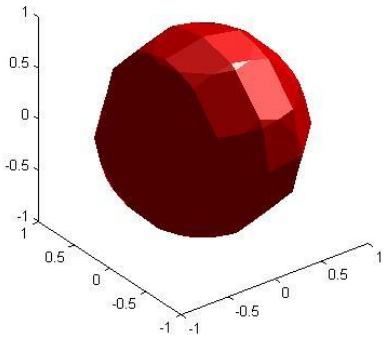
FigB1



FigB2



FigB3



FigB4

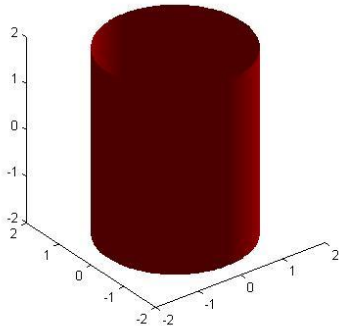
Figure B (FigB1-FigB4). Reconstructed surfaces

It is observed from Table 2 and figures (B1-B4) that to get a fine surface reconstruction the grid cell increment size should be small, producing large number of elements to patch the surface. The surface is illuminated and shaded using the phong shading method. The above used inputs were already on regularized grid of dimensions -2 to 2 for all the three axes and radius of sphere was 0.5. The volume data was generated using implicit function for sphere $f(x,y,z)=x^2+y^2+z^2-rad^2$. The same algorithm was tested on another scalar volume data on regularized grid. The output is shown as the Table 3.

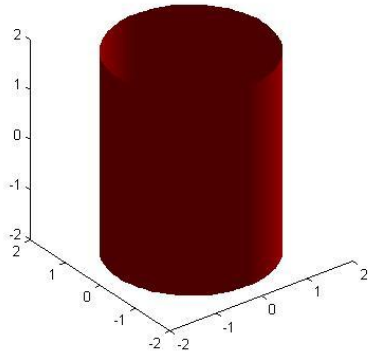
Table 3. Test Results geometric information obtained

Sr. No.	Data Size	Grid Step Size	No. of Facets	No. of Vertices	Iso value	Speed of execution	Result
1	81 X 81 X81	0.05	40320	80640	0.51	Moderate	FigC1
2	41 X 41 X 41	0.1	9920	19840	0.53	Fast	FigC2

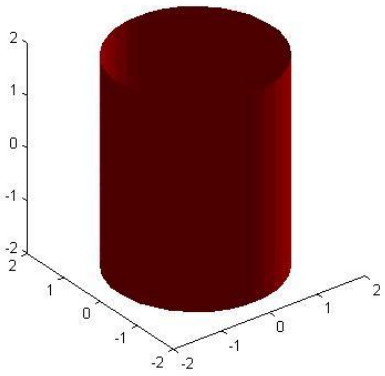
3	17 X 17 X 17	0.25	1664	3328	0.53	Fast	FigC3
4	11 X 11 X 11	0.4	560	1120	0.32	Faster	FigC4



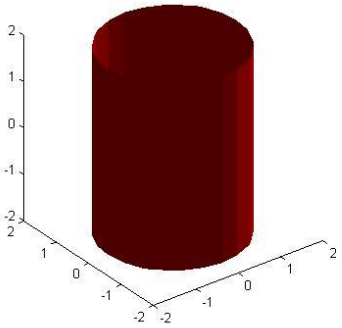
FigC1



FigC2



FigC3



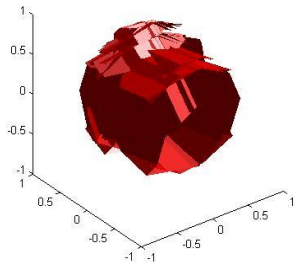
FigC4

Figure C. (FigC1 to FigC4) Reconstructed surfaces test dataset

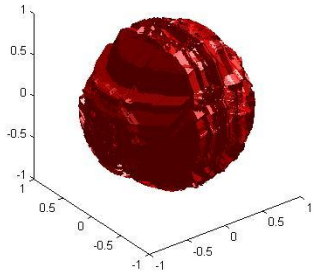
The same algorithm is tested on randomly generated grid and the results are shown below in table 4. The corresponding resulted surfaces are presented in figure D (FigD1 to FigD4). It has been observed that with increase in number of vertices the resolution of the reconstructed surface improves and a smooth surface is resulted.

Table 4. Geometric information obtained for randomly selected input data

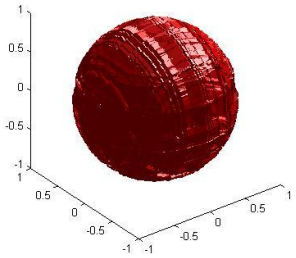
Sr. No.	Data Size	No. of Facets	No. of Vertices	Iso value	Speed of execution	Result
1	10 X 10 X 10	548	1184	0.63	Fast	Poor Resolution (FigD1)
2	30 X 30 X 30	21725	46141	0.63	Fast	Low Resolution (FigD2)
3	50 X 50 X 50	118034	247802	0.63	Moderate	Good Resolution (FigD3)
4	70 X 70 X 70	314847	668319	0.63	Slow	Better Resolution (FigD4)



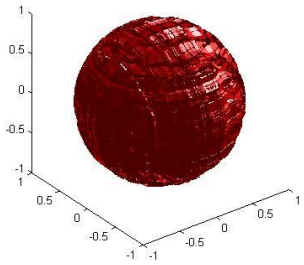
FigD1



FigD2



FigD3



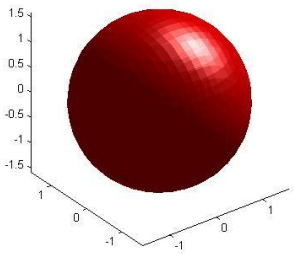
FigD4

Figure D. (FigD1 to FigD4) Resulted surfaces

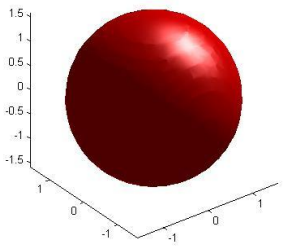
It is observed that the algorithm works for even randomly distributed points and have scalar volume data following the sphere function of three variables. The surface is understood and clearly follows the desired shape as the points are increased. Thus, the point density should be high in case of applying the marching cubes algorithm on the data otherwise the reconstructed surface will not be smooth and clear.

IV. Post Processing on Vertex Set & Result

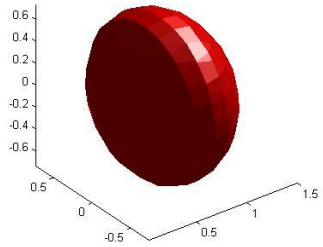
This basic marching cube generates many duplicate vertices also. These duplicate vertices should be removed to save memory and should not degrade the surface. Duplicate vertices are removed by sorting the vertices list and followed by finding the difference between the adjacent vertices in the vertex set. Then, the non-zero elements are indexed and only the vertices related to these indexes are kept in the vertices matrix. The cumulative sum of the differences is performed to predict the faces if there is any change. This generated the lass number of vertices in the vertex matrix (nearly 70% vertices are reduced in number). The facets remain same in number. The resulted surface reconstructed is smoother as compared to produced using the classical marching cubes method. Figure E (FigE1 to FigE4) depicts the reconstructed surfaces after post processing of vertex set as per the proposed approach. It is observed that the resulted surfaces are more smooth and observe the shape of original object. FigE1 is the surface generated before applying the proposed vertex post processing and FigE2 depicts the surface after the post processing operation is performed on the resulted vertex set.

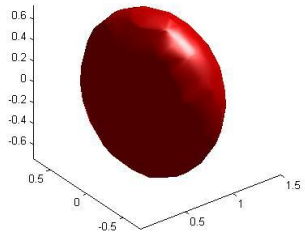


FigE1



FigE2





FigE3 FigE4

Figure E. (FigE1 to FigE4) The left side images shows surface Before post-processing and the right side images show the surface after post-processing of vertex set.

V. Conclusions

The classic marching square and marching cube algorithms can be used as the basis for generating the contour and reconstructing the iso-surfaces from implicit functions representing objects as a scalar field value. This algorithm can be used to process any scalar 2D or 3D volume data and represent the contour/surface respectively. The reconstructed surface is smooth and good quality. The vertices are reduced by 70% (approx.) as observed after performing various tests on the data sets which is a significant improvement that helps to save the memory for storage of the vertices.

VI. Future Work & Scope

This work can be enhanced for any unorganized or complex geometries where the volume data is very difficult to obtain. The marching squares can be enhanced to use for drawing contours over a triangulated surface representing the physical or mechanical properties. Delaunay triangulations can be refined for generating smoother triangulated surface both for 2D and 3D data from their point coordinate information only.

References

[1]. R. Mencl and H. Muller (1998), "Interpolation and approximation of surfaces from three-dimensional scattered data points", *State of the Art Reports, Eurographics '98*, pp 51-67.

[2]. H. Hoppe, T. Derose, T. Duchamp, J. McDonald and W. Stuetzle (1992), "Surface reconstruction from unorganized point clouds", *In Proceedings of ACM Siggraph*, pp 71-78.

[3]. B. Curless and M. Levoy (1996), "A volumetric method for building complex models from range images", *In Proceeding SIGGRAPH 96*, pp 303-312.

[4]. S. Yu, Y. Wei, K. He and X. Yu (2010), "A New triangulation method for surface unorganized points", *Information Technology Journal*, 9(7), pp 1421-1425.

[5]. J.D. Boissonnat and F. Cazals (2002), "Smooth surface reconstruction via natural neighbour interpolation of distance functions", *Comput. Geom.* 22, pp 185-203.

[6]. W.E. Lorensen and H.E. Cline (1987), "Marching Cubes: A high resolution 3D surface construction algorithm", *Computer Graphics*, 21(4), pp 163-169.

[7]. G.M. Nielson (2003), "On Marching Cubes", *IEEE Transactions on Visualization and Computer Graphics*, 9(3), pp 283-297.

[8]. http://en.wikipedia.org/wiki/Marching_squares <Last visited 1 Jan 2018>

