Geometric Desing and Space Planning Using the Marching Squares and Marching Cube Algorithms

Carsten Maple
Departement of Computing and Information Systems
University of Luton, U.K.
carsten.maple@luton.ac.uk

Abstract

In the paper we present a method for area and volume approximation using modifications to the Marching Cubes algorithm of Lorensen and Cline [8]. Approximations to two and three dimensional objects using marching squares and marching cubes have been covered extensively. Given an approximation to an object, an algorithm is presented which allows a simple method that can approximate the area or volume of the object. More interestingly the method can be used to estimate the area encapsulated between two points on the surface and a line or the volume encapsulated between three points on the surface and a plane. This is of use in room and space planning operations or the design of rooms and manufactured products.

1. Introduction

There has been a great deal of work lately involving room and space planning regarding geometric constraints. Various key application areas exist including protein modelling and binding, planning and designing floorspace and robot motion planning.

There are various geometric properties of proteins that are important, see [14] for example, and inter-related. For an excellent review of the area the reader is directed to [3]. The general method for calculating the volume of a protein or parts of a protein is based upon van der Waals radii of atoms. This is the radius of influence of an atom an varies for each type of atom. There are a number of methods available for examining the structure and volume of a protein. Three of the major bases for such methods are Voronoi polyhedra, Delaunay triangulation and marching cubes. Voronoi diagrams can be traced back to 1850 and the work of G.L. Dirichlet. Dirichlet's work was largely theoretical use in two and three dimensions. In 1908 Voronoi explored the geometric construction of Dirichlet's work and generalised it extending the work beyond the third dimension, see [15]. Delaunay tringulation is the dual of Voronoi polyhedra and has also been applied to protein modelling for example [13]. The use of Voronoi polyhedra for protein modelling is extensive, examples include [4], [11].

Methods based upon the marching cubes algorithm include the POCKET software tool and the application of boundary cubes for determining candidate binding sites [5], [9]. Such methods overcome the degenerate cases that can occur in the two previous methods.

A major area of interest is the examination of geometric properties of cavities and the packing efficiency (also known as the packing density or packing coefficient) of proteins [2], [6]. The packing efficiency of a given atom is defined to be the ratio of the space it could minimally occupy to the space that it actually does occupy. This ratio can be expressed as thevan der Waals volume of an atom divided by its Voronoi volume

There is a great deal of work in robot motion planning and the reconstruction of three dimensional surfaces and objects [1], [7], [10]. These reconstructions have a number of uses including space and room planning and optimal configurations.

In this paper we present an algorithm for approximating the area and volume of two and three dimensional objects, subsections thereof and cavities. We further present a definition of efficiency of space utilisation based upon this method.

The organisation of the paper is such that in section two we describe the marching squares and marching cubes algorithm, and explain how these can be used for

Proceedings of the 2003 International Conference on Geometric Modeling and Graphics (GMAG'03)
0-7695-1985-7/03 $17.00 © 2003 **IEEE**

area and volume approximation in section three. Section four applies the methods to space planning problems before conclusions and further work are presented in section five.

2. Marching Squares and Marching Cubes

In 1987 Lorensen and Cline, [8], presented their seminal work detailing the Marching Cubes algorithm. The algorithm is a surface rendering technique used to produce 3D images given point data from 3-space. The data is given on orthogonal grids over parallel planes. The distance between the planes is the same as the grid mesh size in each plane. Data is given at each of the vertices of every square in every slice. This pixel data is transformed into values of either 0 or 1; a 0 corresponding to a vertex lying outside the surface, a 1 corresponding to all other vertices. Although there exists a number of methods for the 3D visualisation of medical data 3D volume slicing is still the most commonly used [1].

The traditional data structure for this type of information is a standard three-dimensional array, where the value of the pixel, 0 or 1, at position (i, j, k) is stored in position [i, j, k] of the array. There is a simple mapping from a pixel-wise representation to a marching cube representation. Two consecutive slices, k and k-1, say, are used to define a layer of marching cubes. Four neighbouring pixels that form a square from each of the two slices comprise a cube.

The marching cubes algorithm forms a piecewise-planar approximation to an object. The boundary of the object is assumed to pass exactly half way between a pixel that is exterior to the boundary of an object and a neighbouring pixel that is not. This determines the way in which the surface intersects a cube, if indeed it does. There are 8 vertices to each cube, each of which can be outside the surface of the object or not. There are, therefore, 256 different ways in which a cube can intersect the surface. In order to uniquely identify each of the 256 ways we assign a weighting value to each vertex of a cube as shown in Fig. 1.

2.1 Marching Squares

This method can be used to give a piecewise-linear approximation to a two-dimensional object. An image is, usually, electronically scanned and the bitmap pixel values then used to give function values over a grid. A shape may also be defined by a series of equations over intervals; these equations can then be used to determine the two-dimensional object they enclosed, which in turn

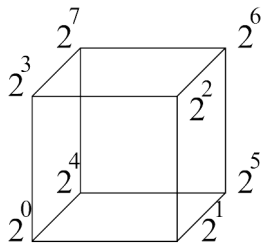

Fig. 1. *The weightings given to the vertices of each cube.*

can be used to define the pixel values over a uniform grid.

In a manner analogous to that of marching cubes, we assign each vertex of each square a value of either 0 or 1; one corresponding to a vertex on the boundary or in the interior of the shape with which we are concerned, and zero corresponding to all other vertices. These values can then be used to assign a value to each square that denotes the manner in which the shape boundary passes through the square. If we allow the lower left vertex of a square to contribute 0 or 1 to the square's value, the lower right to contribute 0 or 2, and so on round the four vertices, then the binary numbers 0000 to 1111 can be used to uniquely identify this intersection. We store the decimal equivalent of these binary numbers. The 16 possible ways in which the boundary and square inter-sect are given in Figure 2, along with the relevant index.

2.2 Marching Cubes

When considering a marching cubes implementation, data is given over a series of parallel planes as explained earlier. The standard data storage mechanism for a marchning cubes representation is a three-dimensional array. For a fixed $k = K$ the elements `marching_cubes (i, j, K)` for varying i and j, represent a *slice* of cubes.

Though there are 256 possible values that a marching cube may assume, depending upon which or its vertices are inside the three-dimensional object, the number of unique representations can be reduceed to 14. We define an equivalence relation $\equiv_3$ such that `marching_cube_1` $\equiv_3$ `marching_cube_2` $\Leftrightarrow$ the representation `marching_cube_1` can be transformed into `marching_cube_2` by elementary operations involving rotations, reflections in a plane or inversion. We define the inverse of a marching cubes representation `marching_cube_a` to be that obtained when all vertices in `marching_cube_a` with a value of 0 are

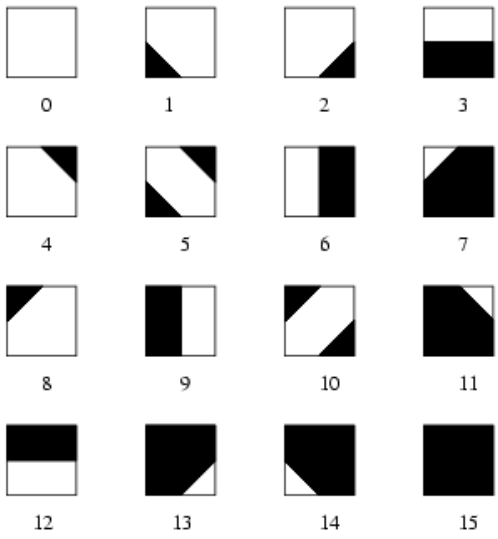

Fig. 2. *The approximations to the way in which the boundary intersects a grid square.*

assigned a value of 1, and all vertices a value of 1 are assigned a value of 0. The 14 equivalence classes are represented in Figure 3.

3 Using marching squares and marching cubes to form area and volume approximations

We begin by calculating the area of the marching square that is assumed to be part of the object. This will be used to calculate an approximation to the area of an object or the area encapsulated between two points on the surface and a line.

For the 16 possible marching square configurations illustrated in Figure 1, we can define an equivalence relation $\equiv_2$ such that `marching_square_1` $\equiv_2$ `marching_square_2` $\Leftrightarrow$ the representation `marching_square_1` can be transformed into `marching_square_2` by elementary operations involving rotations, reflections in a line or inversion. We assume that the squares are of size 2 units by 2 units without loss of generality. The areas for squares in equivalence class [3] (that is the equivalence class containing the marching square labelled 3 in Figure 1), the area is 4 square units. The areas for squares in equivalence class [1], the area is 1 square unit (or 7 for the inverse).

The volumes for the marching cubes are given in Table 1. Again we assume that each side (in this case of the cube) is of length 2.

Equivalence Class	Volume
1	0.1666
2	0.6666
3	1
4	5
5	0.3333
6	3
7	3.8333
8	0.3333
9	3
10	1.1666
11	4
12	2
13	0.5
14	4

Table 1. *The volume associated with each equivalence class of marching cubes*

4 Using marching squares and marching cubes for room and space planning

In this section we present an algorithm for using piecewise linear and piecewise planar approximations for space planning in two and three dimensions respectively.

4.1 Algorithm for approximating the area between an object and a straight line

We consider finding the area between an object and a straight line such that the straight line joins two points on the boundary of the object. An example is shown in Figure 4. We must place a stipulation on the two points on the boundary : the straight line joining them must not cross the boundary of the object. The reason for this is simple. One might reasonably define such an area to be infinite. If this is the case then there is no need for computation. If one believes the area to be finite, then the area would simply be a sum of smaller areas between points on the boundary which also lie on the line. In this case the current algorithm is simply applied a number of times. We define A:= (x_A, y_A) and B:= (x_B, y_B) then the straight line between A and B can be given by $l_{AB} := A + \lambda (B - A)$, $0 \leq \lambda \leq 1$. We can now present the algorithm for approximating the area between an object and a straight line, based upon marching squares.

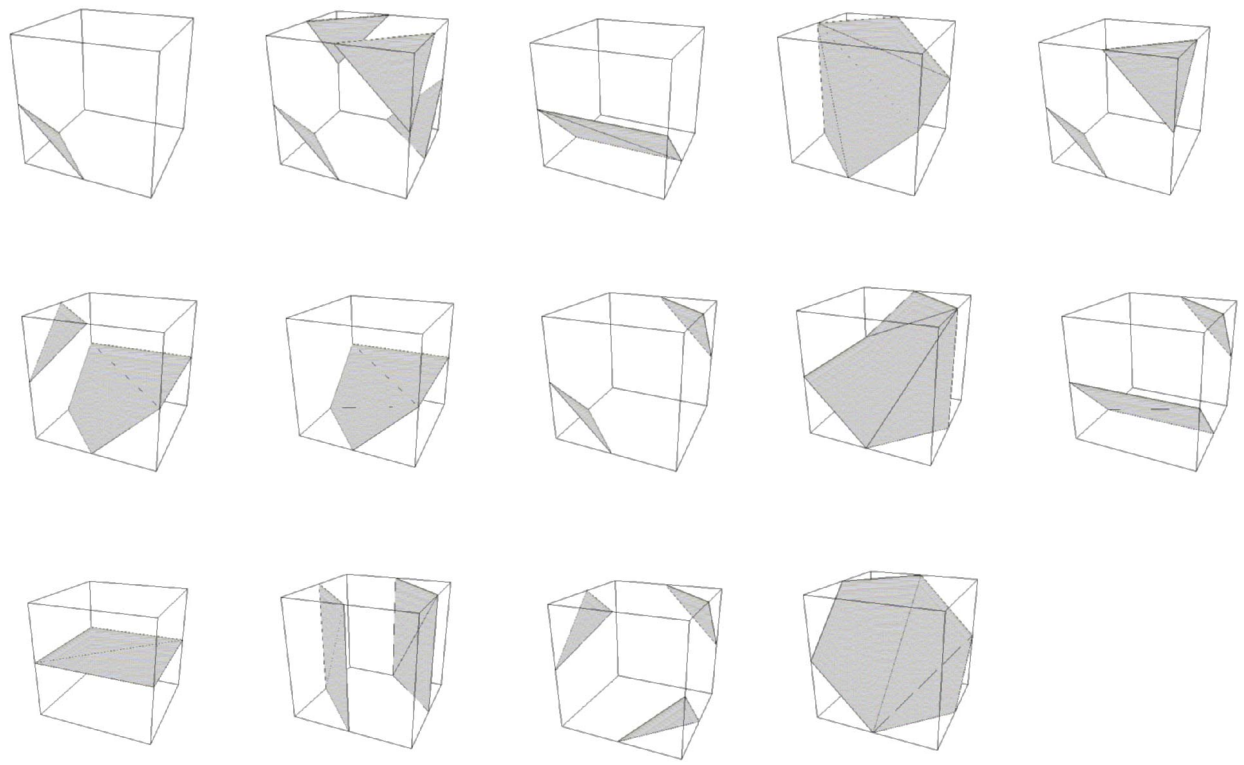

Figure 3: The Marching Cube representations can be reduced to 14 equivalence classes. Each class is represented in the figure.

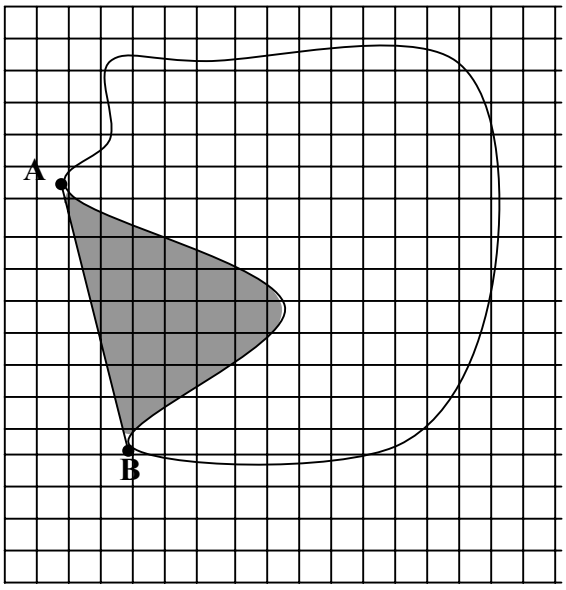

Figure 4. An example of an orthogonal grid placed over an arbitrary object. The area of the shaded region is sought

Step 1 : Obtain a marching squares representation of the object
Step 2 : Isolate the squares between the straight line l_{AB} and the boundary of the object.
Step 3 : Consider the marching squares through which the line l_{AB} passes. Calculate a coarse approximation to the area between the line l_{AB} and the boundary of the surface that is contributed by these squares. The contribution by each square is approximated as max (0, 2 – marching square area approximation value). This will minimise the maximum possible error.
Step 4 : Contribution from the remaining squares is calculated by summing the individual marching square area approximation values.

Consider the example shown in Figure 4. A marching cubes representation can be found for this arbitrary object. The shaded area is an example of an area that may require determining. The algorithm described above can be used to find this area. Using the grid shown in the figure, the shaded area can be approximated as 109.75

square units. Using a finer grid can give a more accurate approximation to the area.

A similar method can be employed in the three dimensional case using marching cubes. The volume between a plane and the surface of a three dimensional object is found. The volume is defined to be between three points on the surface, A, B, and C, say, and a plane. Again there must be the condition that the plane does not cut the plane at any points in the interior of the triangle ABC. The equation of the plane is given by $l_{ABC} := A + \lambda (B - A) + \mu (C - A)$.Cubes through which the plane passes are considered to contribute max (0, 4 – marching cube volume approximation value). Again this will minimise the maximum possible error.

We now have a simple method to calculate approximations to the the area encapsulated between two points on the surface and a line or the volume encapsulated between three points on the surface and a plane. We note that this is a piecewise linear or piecewise planar approximation respectively. It is possible that finding higher-order approximations may be useful, however there is no justification for this. Given that we are approximating in this linear manner, the accuracy of the method is as accurate as the grain of the grid.

The area and volume methods can now be applied to space and room planning. We can represent a general room or docking object (depending on the situation) using a marching squares or marching cubes based method, depending upon the number of dimensions we are working in.

When considering space or room planning there is a need for a measure of the efficiency of a configuration. We now present a new measure of efficiency based upon the marching squares method. The efficiency is defined for an object being placed in the proximity of the boundary of a housing object.

$$\text{Efficiency} = \frac{\text{area of object being placed}}{\text{area of cavity}} \min\left(\frac{\text{length of boundary of object being placed}}{\text{length of boundary of area for housing}}, \frac{\text{length of boundary of object being placed}}{\text{length of boundary of area for housing}}\right)$$

We can now give an algorithm for space planning using the marching squares algorithm.

Step 1 : Obtain marching squares representation for the housing object.
Step 2 : Calculate the (approximate) area between the boundary of the object and l_{AB}.
Step 3 : Calculate the area of the object to be placed.
Step 4 : If the area of the object to be placed is greater than the (approximate) area between the boundary of the object and l_{AB} then stop; object cannot be placed.
Step 5 : If the object to be placed cannot be positioned without the boundary passing through l_{AB} then stop; object cannot be placed.
Step 6 : Object can be placed, calculate efficiency.

Figure 5 shows a housing object and an object for placement. This method can be extended to the three dimensional case using marching cubes. The efficiency can be defined by the equation

$$\text{Efficiency} = \frac{\text{volume of object being placed}}{\text{volumeof cavity}} \min\left(\frac{\text{area of boundary of object being placed}}{\text{area of boundary of area for housing}}, \frac{\text{area of boundary of object being placed}}{\text{area of boundary of volume for housing}}\right)$$

A similar algorithm to the preceding method can be constructed for the three-dimensional case.

4.2 Multilevel modelling

The method we have presented has used a grid that is of the size of the pixels. However, consideration should be given to multilevel modelling. The size of the grid for constructing a marching cubes or marching squares representation should be chosen appropriately, with consideration given to the size of the object to be placed.

Figure 5 shows the same object as in Figure 4, but with a coarser grid. Using the algorithm on this grid gives an approximation of 27.75 square units. Converting this to the same units of area that are used in Figure 4 an approximation of 111 is obtained. This is very close to the 109.75 found using the fine grid. There is no guarantee that the two areas will consistently be so close – a 1% difference. However, it is clear that the two approximations will be of the same magnitude, providing that the area to be found is sufficiently large relative to the grid size.

Using a coarser grid has the advantage it is much less computationally intensive both in approximation construction and area or volume approximation.

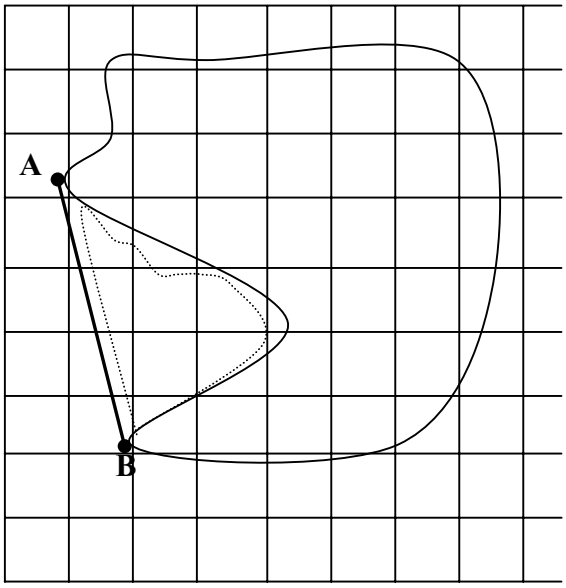

Figure 5. The length of the boundary of the area for housing. is defined to be the combined lengths of the boundary between A and B and the length of the straight line l_{AB}

5 Conclusions and further work

In this paper we have presented algorithms for approximating the areas of two dimensional objects and subareas thereof. The area between two points on the boundary and a line can also be determined. Similar methods have been presented for the three dimensional case, where the volume between three points on a boundary.

Algorithms have also been presented for the placement of objects on or in proximity to the boundary of housing objects. The algorithm not only determines whether the placement is possible, but also returns an efficiency value fo9r the fitting that is equal to 1 for a perfect fit.

Improvements to the work include the expansion to cases in which we are concerned not with a straight line or plane but an arbitrary curve or manifold. Any such consideration should be straightforward.

References

[1] Figueiredo, O., Hersch, R. D, (2003) Parallel unfolding and visualization of curved surfaces extracted from large three dimensional volumes, accepted for publication

[2] Gerstein, M. Chothia, C (1996). Packing at the Protein-Water Interface. Proc. Natl. Acad. Sci. USA 93, 10167-10172.

[3] Gerstein, M, Richards, F.M. (1999). Protein Geometry : Volumes, Areas, and Distances, The International Tables for Crystallography, Volume F: Macromolecular Crystallography, 22: Molecular Geometry and Features.

[4] Gerstein, M, Tsai, J & Levitt, M (1995). The volume of atoms on the protein surface: Calculated from simulation, using Voronoi polyhedra. J. Mol. Biol. 249: 955-966.

[5] Hannaford, G., Maple, C., Mullins, J., (2001) Finding and Characterising Candidate Binding Sites, Proceedings of the 5th IEEE International Conference on Information Visualisation, InfVis2001, 157-161.

[6] Hubbard, S J & Argos, P (1994). Cavities and packing at protein interfaces. Protein Science 3, 2194-2206.

[7] Laugier, C., et al, (1994) Solving complex motion planning problems by combining geometric and physical models, Workshop on Algorithmic Foundations of Robotics.

[8] Lorensen, W.E., Cline, H.E., (1987) Marching Cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics, 21, 4, 163-169.

[9] Maple, C., (2001) A Boundary Representation Technique for Three-Dimensional Objects, Proceedings of the 5th IEEE International Conference on Information Visualisation, InfVis2001, 397-403.

[10] Precup, D., (2000) Temporal Abstraction in Reinforcement Learning. PhD thesis, University of Massachusetts, Amherst, Department of Computer Science.

[11] Procacci, P & Scateni, R (1992). A General Algorithm for Computing Voronoi Volumes: Application to the Hydrated Crystal of Myoglobin. Int. J. Quant. Chem. 42

[12] Richards, F M & Lim, W A (1994). An analysis of packing in the protein folding problem. Quart. Rev. Biophys. 26, 423-498.

[13] Singh, R K, Tropsha, A & Vaisman, I I (1996). Delaunay Tessellation of Proteins: Four Body Nearest-Neighbor Propensities of Amino Acid Residues. J. Comp. Biol. 3, 213-222.

[14] Tsai, J & Gerstein, M (1999). Volume Calculations of Protein Atomic Groups: Factors Affecting the Calculation and Derivation of a Minimal yet Optimal Set of Volumes. Proteins.

[15] Voronoi, G F (1908). Nouvelles applications des paramétres continus à la théorie des formes quadratiques. J. Reine Angew. Math. 134, 198-287.