國立交通大學

資訊工程學系

碩士論文

從大型三維影像中擷取適應式等值亮度表面

Adaptive Isosurface Extraction from Large Volume Data

研 究 生：倪安廷

指導教授：陳永昇 博士

中 華 民 國 九 十 四 年 八 月

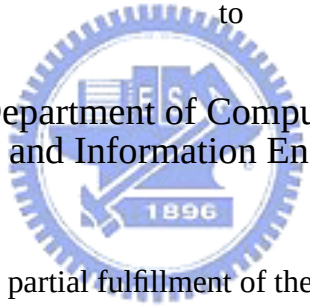
Adaptive Isosurface Extraction from Large Volume Data

A dissertation presented

by

An-Teen Ni

to



Department of Computer Science
and Information Engineering

in partial fulfillment of the requirements

for the degree of

Master of Science

in the subject of

Computer Science and Information Engineering

National Chiao Tung University

Hsinchu, Taiwan

2005

Adaptive Isosurface Extraction from Large Volume Data

by

An-Teen Ni

國立交通大學

論文口試委員會審定書

本校 資訊工程學系 碩士班 倪安廷 君

所提論文:

從大型三維影像中擷取適應式等值亮度表面

Adaptive Isosurface Extraction from Large Volume Data

合於碩士資格水準、業經本委員會評審認可。

口試委員：陳永昌 陳醫吉 陳永昇

指導教授：陳永昇

系主任：張明峰

中華民國九十四年六月二十九日

從大型三維影像中擷取適應式等值亮度表面

學生：倪安廷　　指導教授：陳永昇

國立交通大學資訊工程學系（研究所）碩士班

摘　　要

從磁振造影或斷層掃描之影像當中擷取的等值亮度表面，可以幫助我們了解潛藏在影像中實體的三維結構。利用走訪立方體法可以擷取單一解析度的三角形網格，但針對非常大型的三維影像可能會產生過多的三角形，造成儲存及顯像上的困難。針對這個問題，我們提出一個適應式等值亮度表面擷取法，可以在實體表面變化較劇烈的地方用較精細的三角形來建構，而表面較平坦之處則用較大的三角形來建構，如此可以兼顧三角形數量與表面模型準確性。首先，我們會在三維影像當中，以多維搜尋樹的方式快速找到有等值亮度表面通過的區域，然後利用一個快速演算法來分析其表面變化情形，並產生適應於表面的多重解析度三角形網格。根據我們的實驗結果，經由我們所提出的方法所產生的網格，其三角形個數最多只有走訪立方體法的四分之一，且並沒有損失過多的表面模型精準度。

Abstract

Isosurface extracted from volume data, such as CT, SPECT and MRI, can provide 3D structure information. The marching cubes algorithm has been extensively applied in isosurface extraction from volume data. This algorithm generates spatially regular triangle meshes and can be very storage-consuming and hard rendering when the volume data is huge. We present an adaptive isosurface extraction algorithm which can reduce the triangle number of isosurface by adapting to the variation of the isosurface. The adaptive isosurface contains larger but fewer triangles for low variation area and smaller but more triangles for high variation area. We use kd-tree method to rapidly locate the area containing the isosurface. Then a novel recursive algorithm is applied to analyze the volume data and to generate the triangles with sizes adaptive to the surface. According to our experiments, the triangle number obtained by using the proposed method can be at least four times fewer than that can be obtained by using the marching cubes method without losing significant surface accuracy.

誌　　謝

短短的碩士生生涯即將結束，在這段求學的過程中，首先我要感謝我的指導教授 陳永昇老師以及師母 陳麗芬老師帶我進入醫學影像處理的領域，不僅提供我良好的環境與實驗工具，在做研究的過程中也不斷的給我寶貴的意見，讓我的論文得以順利完成；除此之外，在老師的循循善誘之下，我不僅學習到許多專業的技能，在待人處世上也成熟不少，這些都對即將進入職場的我有不小的助益。其次要感謝我的口試委員 陳永昌教授，在口試過程中給我不少寶貴的意見，讓我的論文增色不少；再此一併致上最誠摯的謝意。

在研究的過程中，實驗室的學弟妹們以及一些我的好友也給了我不少的幫助，認真於專業領域研究的志瑜，愛開玩笑的怡岑、昭翰，喜歡運動的福慧、仲凱，以及活潑的新進學弟妹們，有你們在，讓我的研究生生活變得更充實。

最後感謝我的父母多年來的栽培與教誨，以及哥哥姊姊的疼愛與關心，沒有你們一路上的付出與幫忙，也不會有今天的我；還有在這段求學過程中與我互相扶持、互相砥礪的女友崴琳，妳是我不斷前進的動力。總之，再次感謝所有曾經幫助過我的親友、師長與同學們。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

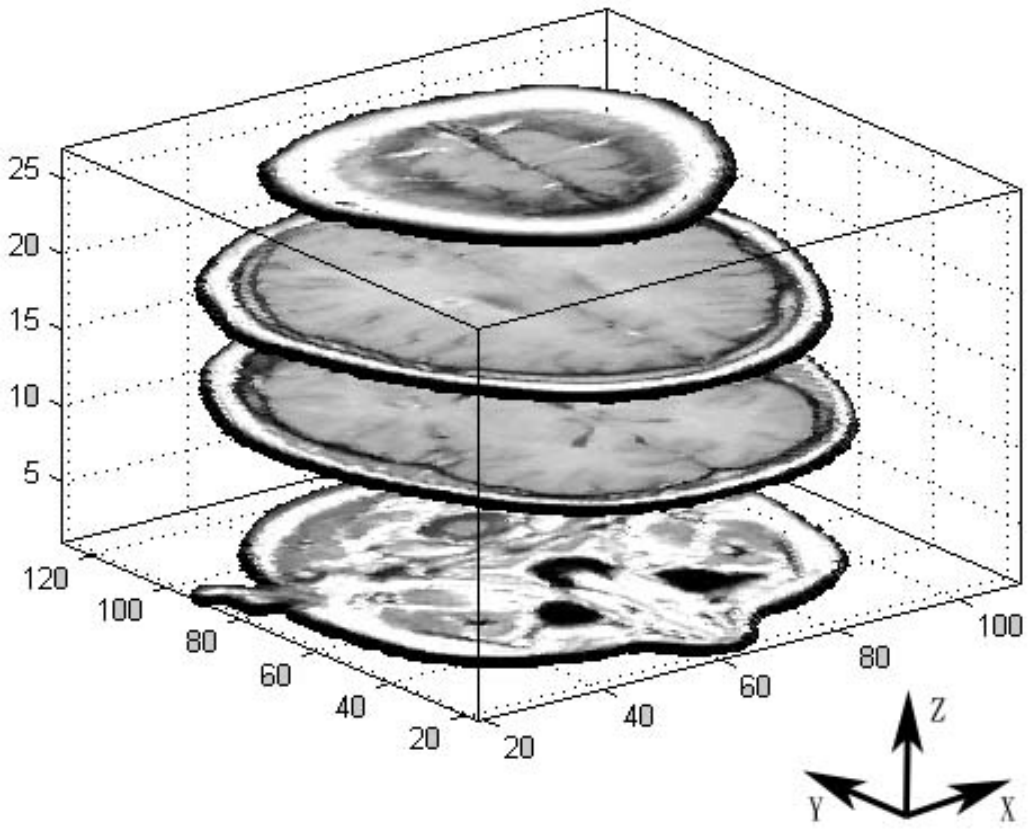


Figure 1.1: The image slices of brain MRI volume data.

1.1 Isosurface Extraction

MRI (Magnetic Resonance Imaging) has been extensively utilized for scanning human body. They can provide volume data composed of a series of 2D image slices as shown in Figure 1.1. In clinical diagnosis and medical research, it is often useful to reconstruct the 3D surface structure of targeted tissues from volume data because 3D surface provides more information for physicians to penetrate the complex structure of the tissue.

Under the assumption of homogeneity, the same tissue results in the same intensity value in MRI volume data. By analyzing the intensity distribution of tissues in the vol-

ume data, we can specify the intensity value of the tissue we want to reconstruct. This intensity value is referred to the *isovalue*. Isosurface represents the surface with the same intensity value that we want to extract in the volume data. We can obtain the boundary surface between two adjoining tissues by extracting the isosurface with the isovalue specified between the intensity values of these two tissues.

Marching cubes algorithm proposed by Lorensen and Cline [1] is considered as a standard approach to extract isosurface. It is a simple and practical algorithm which can extract a 3D triangle mesh to approximate the isosurface and can calculate the normal direction for each triangle. We can then render this 3D triangle mesh to visualize the surface structure of the tissue. Besides visualization, isosurface reconstruction is useful in functional brain mapping research. We can use the 3D triangle mesh of brain to calculate the BEM (Boundary Elementary Method, Figure 1.2) volume conductor model and use the model to calculate the brain activity [2].

1.2 Challenges of Isosurface Extraction

Due to the improvements of acquisition device, the volume data resolution is getting higher and higher rapidly. The triangle number of reconstructed 3D triangle mesh is also increasing when we use marching cubes to regularly extract the isosurface from a large volume data. It often exceed millions of triangles and will be very inefficient to extract, store, and visualize such a huge amount of triangles.

Many mesh simplification algorithms [3] [4] [5] have been proposed in the literature. Most of them take each triangle in the triangle mesh as a decimation target and decide to remove it or not under some error criteria. It is storage-consuming because a huge amount

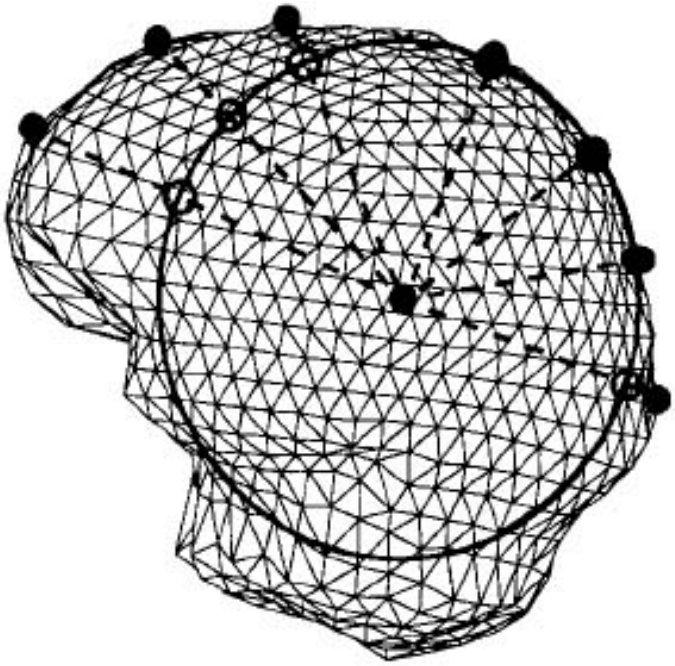


Figure 1.2: The reconstructed triangle mesh for building a BEM model of the brain [2].

of small triangles still have to be extracted first. It is also time-consuming to process such a huge amount of data by applying these algorithms. time-consuming by applying these algorithms to the isosurface with huge triangles reconstructed from large volume data.

The algorithm proposed by He et al. [6] successively applies low pass filter to downsample the volume data. The triangle number can be reduced when extracting isosurface from lower resolution volume data. However, some surface structure may be lost at the same time. Another similar algorithm proposed by Kraus and Ertl [7] downsample the volume data while retaining the critical points to preserve more surface structure. These algorithms reduce a larger amount of triangles and are also efficient. However, downsampling process may lose information that result in the possibility of missing important surface structure.

We propose in this thesis an adaptive isosurface extraction method that can reduce the

triangle number, preserve the topology, and speed up the surface extraction process, We reduce the triangle number of isosurface by adapting to the variation of the isosurface. The adaptive isosurface contains larger but fewer triangles for low variation area and smaller but more triangles for high variation area. We uses kd-tree method to rapidly locate the area containing the isosurface. Then a novel recursive algorithm is applied to analyze the volume data and to generate the triangles whose sizes are adaptive to the surface.

1.3 Thesis Organization

The dissertation is organized as follows: Chapter 2 presents the background and related works. Chapter 3 depicts an overview of our method. Chapter 4 introduces volume data analysis, including both kd-tree construction and k-d search, and the detail about simple cubic boxes construction. Chapter 5 introduces triangle mesh generation, including both isoline loop extraction and merging extraction, and describes the detail about vertex adjustment of final triangle mesh. Chapter 6 presents the experiment results and discussion. Chapter 7 draws the conclusions.


E S A
1896

Chapter 2

Background and Related Works

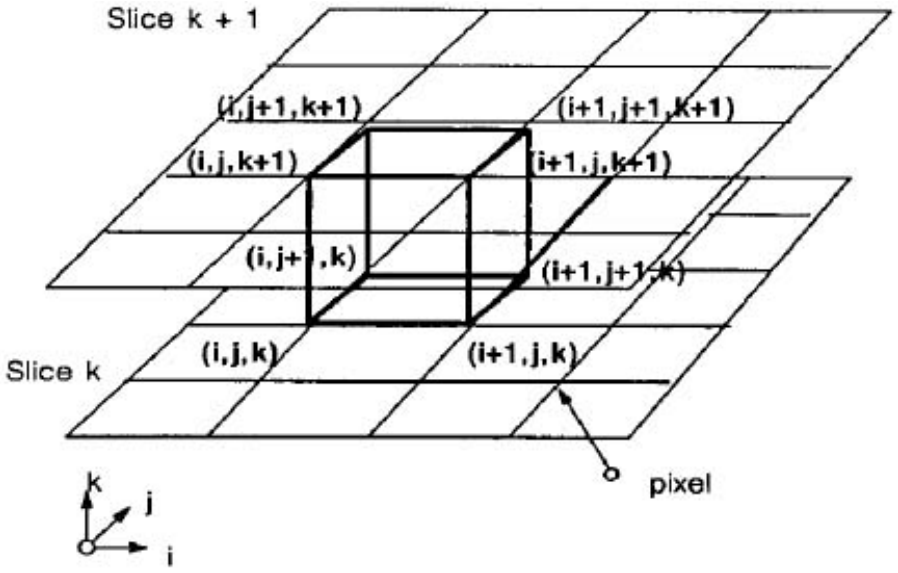


Figure 2.1: The cube can be defined as eight neighboring voxels between two consecutive 2D images [1].

2.1 Marching Cubes

Since we want to extract isosurface, we have to understand the most representative algorithm, marching cubes (MC). Lorensen and Cline proposed the first isosurface extraction algorithm marching cubes in 1986. By visiting the whole volume, it locates the cubes containing isosurface and then generates corresponding 3D triangle or triangles. A cube consists of eight neighboring voxels between two consecutive 2D slices (Figure 2.1) and there exits the isosurface if the intensity values of eight voxels are not all larger or all smaller than the isovalue. A voxel is usually called inside if its intensity value is larger than the isovalue and is called outside if its intensity value is smaller than the isovalue. According to the inside and outside conditions of the eight voxels, MC can determine how to extract triangle or triangles inside the cube. There are 256 situations to locate triangle or triangles in a cube. A look up table with 256 entries can be utilized to rapidly determine

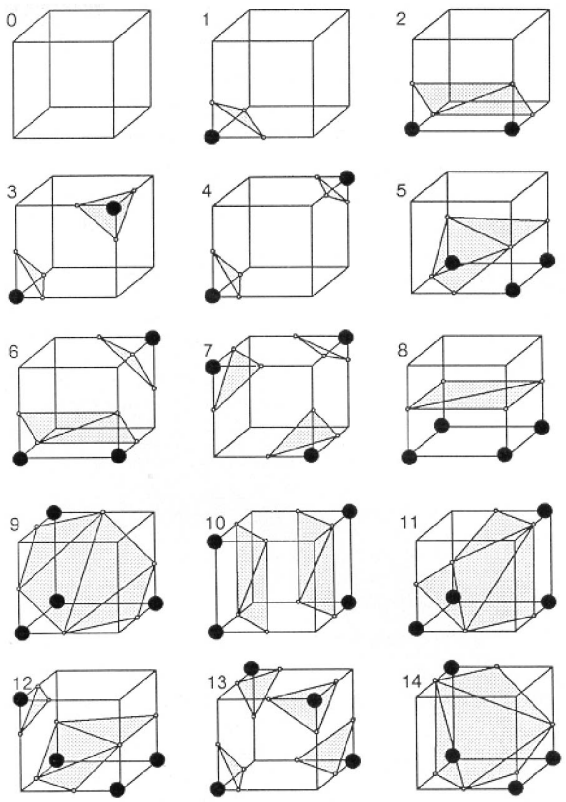


Figure 2.2: The reduced 15 cases of marching cubes algorithm when symmetry is considered [1]. Black voxel usually means inside the isosurface and white voxel means outside the isosurface.

the situation. Figure 2.2 shows the 15 situations when symmetry is considered. After looking up the table, the MC algorithm uses linear interpolation to obtain actual triangle vertex coordinate for each triangle. Before outputting the triangles, the MC algorithm calculates the surface normal ($G_x(i,j,k), G_y(i,j,k), G_z(i,j,k)$) for each triangle vertex (i,j,k) by using central differences along the three coordinate axes:

$$G_x(i,j,k) = \frac{D(i+1,j,k) - D(i-1,j,k)}{\Delta x},$$

$$G_y(i,j,k) = \frac{D(i,j+1,k) - D(i,j-1,k)}{\Delta y},$$

$$G_z(i,j,k) = \frac{D(i,j,k+1) - D(i,j,k-1)}{\Delta z},$$

where $D(i,j,k)$ is the intensity value of voxel (i,j,k).

2.2 Acceleration of Isosurface Extraction

Marching cube will visit all the cubes in the volume data and check whether the cube containing isosurface or not. Actually, it is time-consuming because only a few cubes contain isosurface and the effort of checking empty cubes are a waste of time. Thus, there are algorithms proposed in the literature that can improve the efficiency by skipping empty cubes, referred to a survey study performed by Philip M. Sutton et al. [11] in 2000, in which they compared the extraction time and memory overhead of different isosurface extraction method.

Wilhelm and Van Gelder proposed an octree method [8] in 1992. They used an octree to decompose the volume data into eight subvolume hierarchically until reaching eight unit cubes. The root node of the octree covers the region of whole volume and its eight child nodes cover the region of eight subvolumes. For each octree node, their method analyzes the minimum and maximum intensity values that this octree node covers. Then, depth-first-search strategy is used to traverse the octree to locate the leaf nodes whose isovalues are between the minimum and maximum values. Finally, corresponding triangles are generated for the located nodes.

T. Itoh and K. Koyyamada proposed Extrema Graph method [9] in 1994. It searches the whole volume from a seed cube containing isosurface and propagates to adjacent cubes containing isosurface by knowing the coherence of isosurface It keeps propagating until

all the cubes containing isosurface have been visited. The seed cube can be found by employing Extrema Graph [9].

Yaden Livnat et al. proposed kd-tree technique [10] in 1996. This technique is incorporated in our method and we will discuss the details of the kd-tree technique in Chapter 4.

2.3 Adaptive Isosurface Extraction

Besides accelerating extraction time, a large number of triangles generated by marching cubes hinder both the calculation and rendering. To solve this problem, many adaptive isosurface extraction algorithms have been proposed. We classify these algorithms as follows:

- Discretized-based method:

  C. Montani et al. proposed Discretized Marching Cubes (DMC) method [12] in 1994. This approach assumes discretization of the data set space and replaces cube edge interpolation with midpoint selection. Figure 2.3 illustrates all possible midpoints in this method. Under certain assumptions, the extracted isosurface consists of polygons with finite number of incidences (Figure 2.4). There are only 13 different kind of incidences and can be modeled as follows:

$$x = c, y = c, z = c,$$

$$x \pm y = c, x \pm z = c, y \pm z = c,$$

$$x \pm y \pm z = c.$$

  We can apply simple merging of the facets with the same incidence and then produce

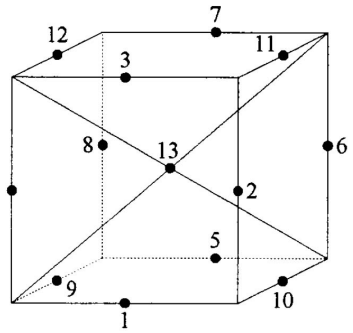


Figure 2.3: Midpoint selection. All possible 13 different kinds of midpoints in DMC algorithm [12].

larger and fewer coplanar facets. This technique will be adopted in our method and we will discuss the details in Chapter 5.

- Box-based method:

  Renbeb Shu et al. proposed the cubic box method [13] in 1995. This approach decomposes the volume data into initial blocks with equal size N, where N is the factor of two. Their method decomposes each initial block into cubic boxes with different resolutions according to the variation of isosurface. For each cubic box, their method uses marching cubes to extract corresponding triangles with different sizes. Unfortunately the crack problem will occur between the triangles in high- and low-resolution boxes. This problem can be solved by patching extra triangles.

  In 1998, Tim Poston et al. proposed Adaptive Skeleton Climbing (ASC) method [14]. This method also decomposes the volume data into initial blocks, but it analyzes each block as simple rectangle boxes by climbing from vertices (0-skeleton) to edges (1-skeleton), to faces (2-skeleton), and then to boxes (3-skeleton). The rectangle box is simple if there is at most one inside-outside exchange of voxels along each of the x,

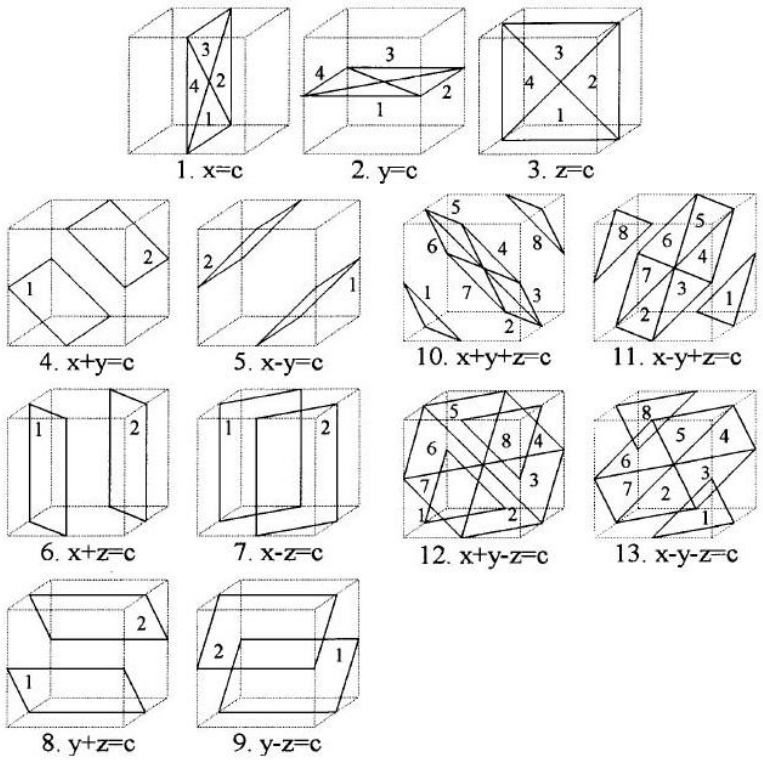


Figure 2.4: All possible 13 different kinds of incidences in DMC algorithm [12].

y and z direction. For each simple rectangle box, it will use isoline loop method to extract triangles without suffering the crack problem. This technique is also adopted in our method and we will discuss the details in Chapter 5.

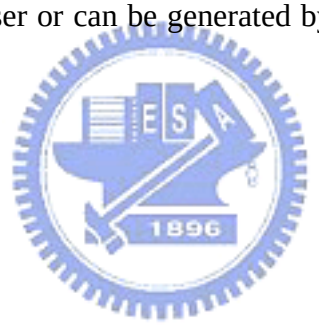
- Tetrahedral-based method:

  Yong Zhou et al. proposed Tetrahedral method [15] in 1997. This approach initially decomposes volume data into 12 initial tetrahedrons and then hierarchically subdivides each initial tetrahedral into tetrahedral pairs by using the so-called bisection process. Then, it uses bottom-up method to fuse tetrahedral pairs according the error criterion called EBM and extracts triangles for these multi-resolution tetrahedrons

by using the marching tetrahedral method [16]. Based on the property of tetrahedral, this method also has the advantage that it does not suffer from any crack problem. In 2000, T. Gerstner et al. proposed another fusion method [17] which can preserve and control the simplification of topology.

- Volume-warping-based method:

  In 2002, Laurent Balmelli et al. proposed volume warping method [18]. By using a relaxation algorithm [19] with a variation map, this approach warps the volume data to inflate the high variation area. Then, it extracts the inflated triangle mesh and then shrinks the triangle mesh back to the original scale. Here, the variation map can be manually specified by user or can be generated by using neighborhood-crossing method [18].



Chapter 3

Overview of Our Method

In this work, we improve the ASC and DMC methods and then incorporate both of them with the kd-tree method. The following flowchart illustrates an overview of our method.

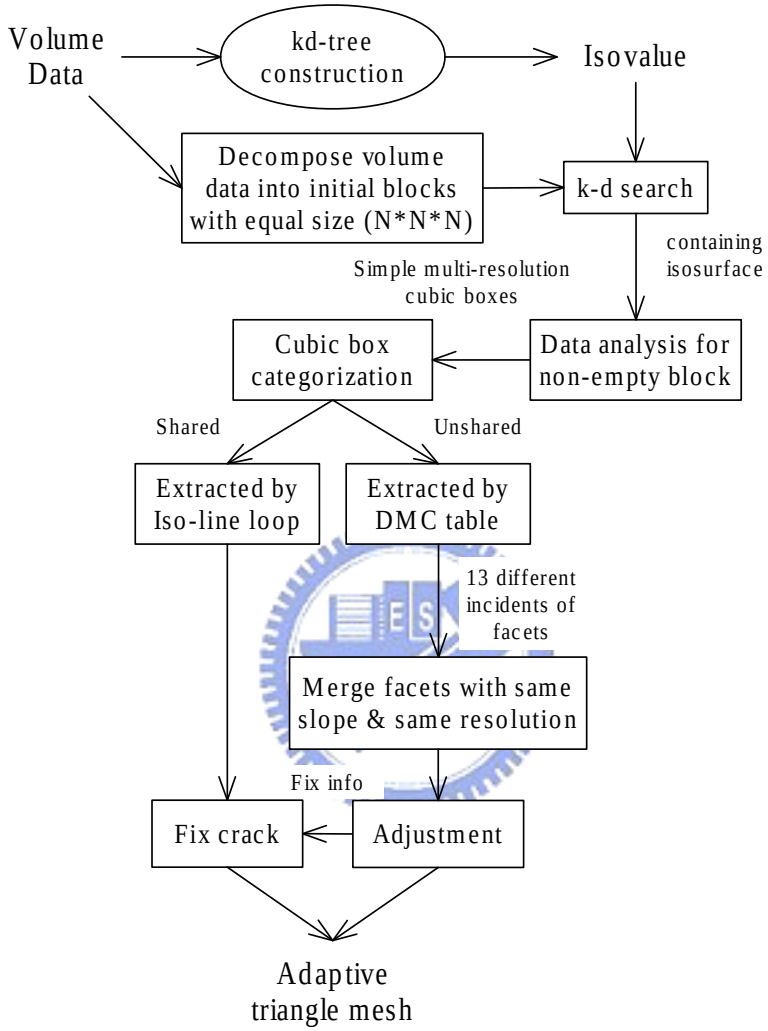


Figure 3.1: The flowchart of our method. The kd-tree construction, marked in the circle area is in the preprocessing stage.

We first take volume data as input and then decompose it into initial blocks with equal size $N \times N \times N$, where N is the factor of two. The total voxels containing in each initial block will be $(N+1) \times (N+1) \times (N+1)$. The choice of N will be discussed in Chapter 6.

The kd-tree will be constructed from the volume data in a preprocessing stage. We will use k-d tree and k-d search [10] to locate the blocks containing isosurface after user specifies an isovalue. After locating the blocks containing isosurface, we will analyze these blocks to construct simple cubic boxes. The cubic box is simple if there is at most one inside-outside exchange of voxels along each of the x, y and z direction. The size of cubic box must be the factor of 2 and the size of the largest cubic box will be $N \times N \times N$. Then, we will classify these simple cubic boxes as shared or unshared. A cubic box is called shared if there are higher resolution cubic boxes adjacent to it. Details about k-d tree, k-d search, volume data analysis and simple cubic box classification will be discussed in chapter 4.

There may exist crack problem around a shared box if we use marching cubes algorithm to extract triangles directly. Thus, we use isoline loop method with division information to extract triangles in these shared cubic box [14]. The other unshared cubic boxes can be extracted by marching cubes algorithm, considering only the intensity value of the voxels on the eight corners. This is because there is only one inside-outside exchange and we do not lose any structures in the simple cubic box when only the eight corner voxels are considered. However, we extract all the triangles for these multi-resolution unshared simple cubic boxes by looking up DMC table. Then we merge the triangles with the same slope and same resolution by using DMC method. In order to increase the accuracy of surface model, we adjust the vertices of facets extracted by DMC method to their actual intersection positions. Here, two kind of cracks may occur and are considered in our method. Details about isoline loop extraction, DMC extraction, adjustment, and crack fixing will be discussed in chapter 5.

Chapter 4

Volume Data Analysis

4.1 k-d Tree and k-d Search

As we mentioned previously, many methods have been proposed to improve the efficiency of isosurface extraction. In this work, we use k-d tree and k-d search to accelerate isosurface extraction. We describe the steps of k-d tree construction in our method as follows:

1. analyze the minimum and maximum intensity value of voxels in each initial block and store the min-max value and block address (x, y, z), which is the coordinate relative to the volume origin, in a k-d node.

2. sort these k-d nodes as a list with quick sort according to the minimum/maximum values (minimum or maximum value will be chosen alternatively).

3. pick the median one as root node and partition the list into left-sub list and right-sub list.

4. take the kd-nodes in left and right sub-list and go back to step 2 respectively .

In this way, we can get k-d tree recursively.

When user specifies an isovalue, we will locate the blocks containing isosurface by searching the k-d nodes whose isovalue is between the minimum and maximum value. Since k-d tree is a multi-dimensional search tree, it can apply multi-dimensional tree search techniques such as search-min-max and search-max-min methods presented in [10]. As Figure 4.1 illustrates, we only need to search the nodes at left-up corner when user specifies v as the isovalue such that the search process can be very efficient. When we find these nodes which contain a part of isosurface, we keep the block address for the analysis procedure.

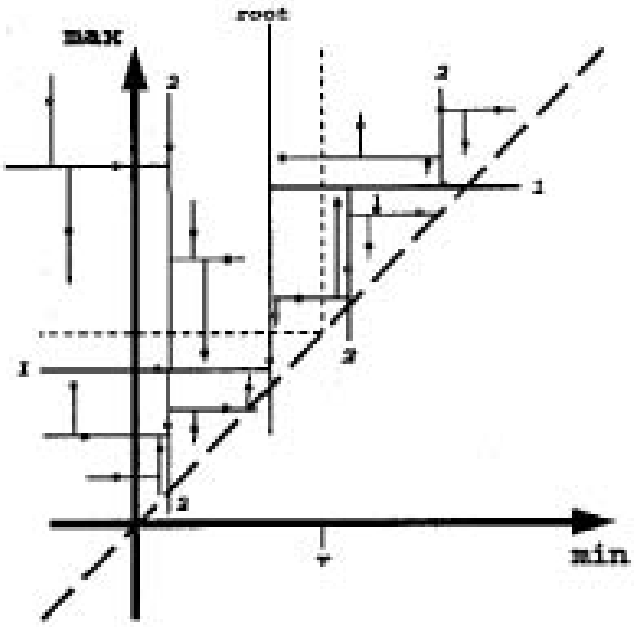


Figure 4.1: The k-d tree construction and the concept of k-d search. All the lines means a k-d node in k-d tree. The vertical line means the minimum value is used for sorting. The horizontal line means the maximum value is used for sorting. The value v means the specified isovalue [10].

4.2 Simple Cubic Boxes Construction

As we mentioned previously, we need to analyze each non-empty block to construct simple cubic boxes. First of all, we explain the lign and dike used in [14]. As shown in Figure 4.2 (a), a line with $2^n + 1$ number of voxels is called a lign, where $n = log_2^N$. As shown in Figure 4.2 (b), a line covers $a2^b$ to $(a + 1)2^b$ voxels is called a dike, where $0 \leq a < 2^{n-b}$ and $0 \leq b \leq \log_2^N$.

All dikes in a lign can be organized in an binary tree as shown in Figure 4.3 (a). The tree size will be 2^{N-1} in which we never use the first node. There are four kinds of variation situations among the voxels in each dike. We can use 2 bits to represent these

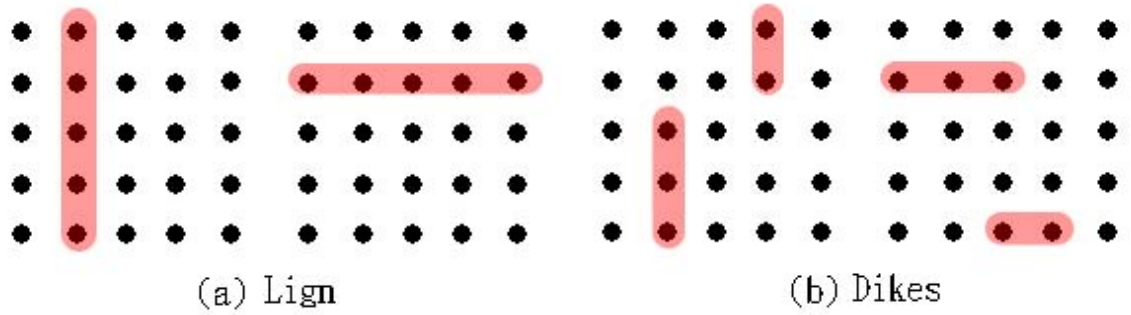


Figure 4.2: Lign and dike. The block size is 4 in this example and the maximum size of dike is equal to the block size.

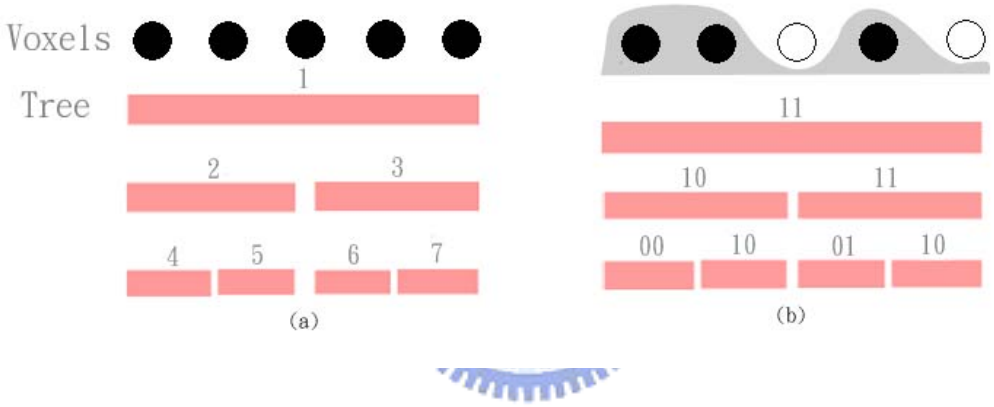


Figure 4.3: All the dikes in a lign can be organized in a binary tree and the variation information can be stored in the tree nodes.

four situations:

$$Ver[i] = \begin{cases} 00, & \text{No variation for dike } i. \\ 01, & \text{Variate from inside to outside for dike } i. \\ 10, & \text{Variate from outside to inside for dike } i. \\ 11, & \text{Variate more than once for dike } i. \end{cases}$$

Figure 4.3 (b) illustrates the value in the variation binary tree of a lign where isosurface pass through. We can construct the binary tree from bottom to top by OR operation as

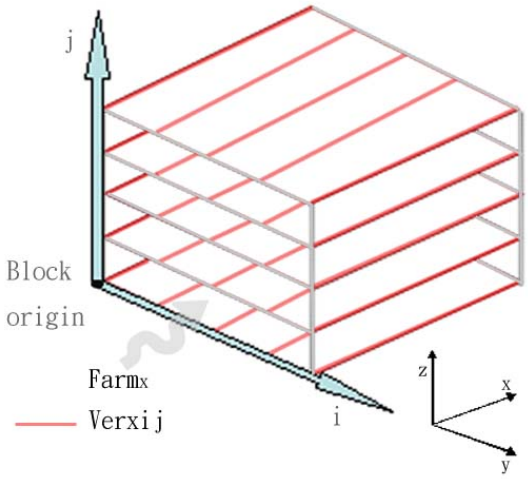


Figure 4.4: The example of all possible ligns in each farm along the x direction.

follows:

$Ver(i) = \{\ Ver(2i) \quad OR \quad Ver(2i),$ i is the index of current dike in the variation tree

A dike is simple if there is at most one inside-outside variation. We will construct the variation binary trees for all the ligns along each of the x, y and z direction. As shown in Figure 4.4, there are Verx$_{ij}$, Very$_{ij}$, and Verz$_{ij}$, where $i : 0 \sim N, j : 0 \sim N$, in all the slices of a block. These slices in a block are also called farms and there will be farm$_x$, farm$_y$ and farm$_z$ respectively.

Now, we can construct the simple cubic boxes for each initial block by a recursive algorithm. Figure 4.5 explains the main idea of this algorithm by a 2D example. In the 2D example, we will check relative dikes on relative ligns among each of x and y direction. If there is only one dike which is varied more than once, we will break this farm as four parts recursively. It is strategy forward to extend this algorithm to 3D case by considering

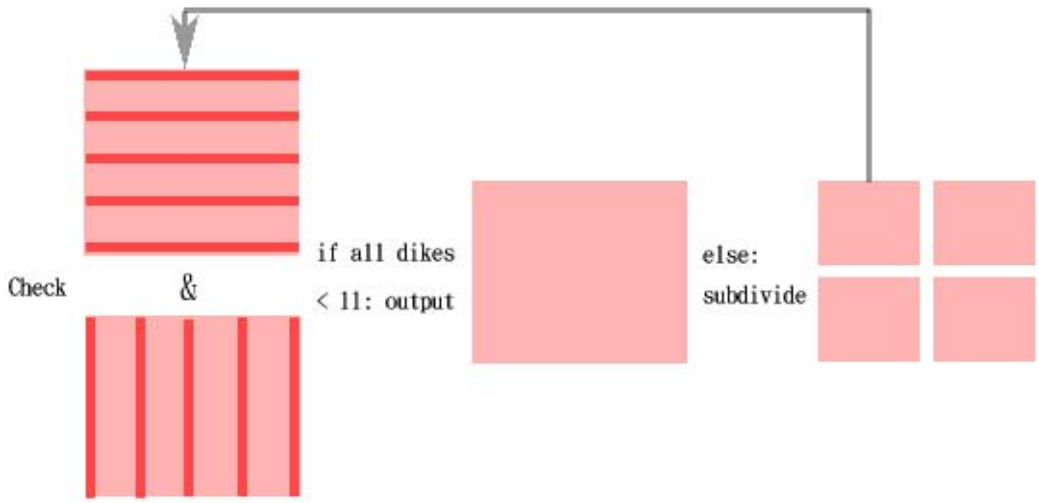


Figure 4.5: The 2D example of our fast recursive algorithm.

z direction.

4.3 Simple Cubic Boxes Classification

As shown in Figure 4.6, a simple cubic box may suffer crack problem if there are higher resolution cubic boxes adjacent to it. For each simple cubic box, we have to classify it as shared or unshared before generating triangles.

Before showing how to classify these simple cubic boxes, we first explain strip and plot used in [14]. As shown in Figure 4.7 (a), a strip is composed of two consecutive ligns in a farm. As shown in Figure 4.7 (b), a plot is composed of two consecutive dikes in a farm.

All plots in a strip can be organized in a binary tree. The tree is of size 2^{N-1} and its first node will never be used. Each plot in a farm may be divided by the simple cubic box

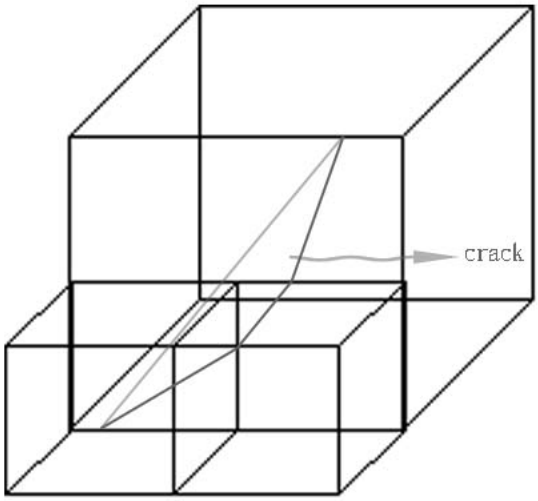


Figure 4.6: The crack problem occurs between low and high resolution simple cubic boxes. It occurs if we directly extract both of them by marching cube algorithm.

and we use a boolean variable to represent the division information:

$$Div[i] = \begin{cases} false, & \text{Does not divide by simple cubic boxes for plot } i. \\ true, & \text{Divided by simple cubic boxes for plot } i. \end{cases}$$

Figure 4.8 illustrates the value in the division binary tree when simple cubic boxes locate on it.

After traversing all the simple cubic boxes in each block, we can configure Divx$_{ij}$, Divy$_{ij}$, and Divz$_{ij}$, where $i : 0...N, j : 0...N$. Then, we need to use OR operation for the division information of strip pairs between neighboring blocks:

$$Div_c(i) = Div_c(i) \quad OR \quad Div_n(i),$$

where i is the index of the plot in the division tree,

c means current block, and n means an adjacent block of c.

This is because the adjacent simple cubic boxes may come from neighboring blocks.

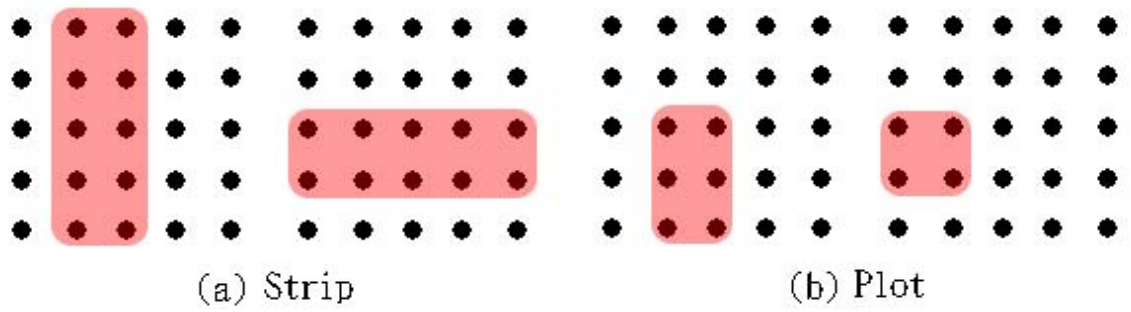


Figure 4.7: Strip and plot. The block size in this example is 4. The maximum size of a plot is equal to the block size.

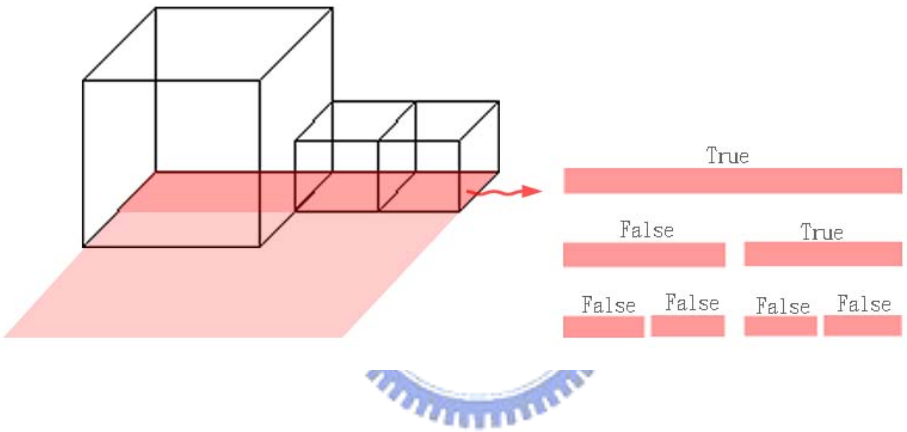


Figure 4.8: All the plots can be organized in a binary tree and the division information can be stored in the tree nodes.

For each box, we can check the division information for the six box faces to classify whether it is shared or unshared. If there is only one plot which is divided, it means the simple cubic box is shared.

Chapter 5

Triangle Mesh Generation

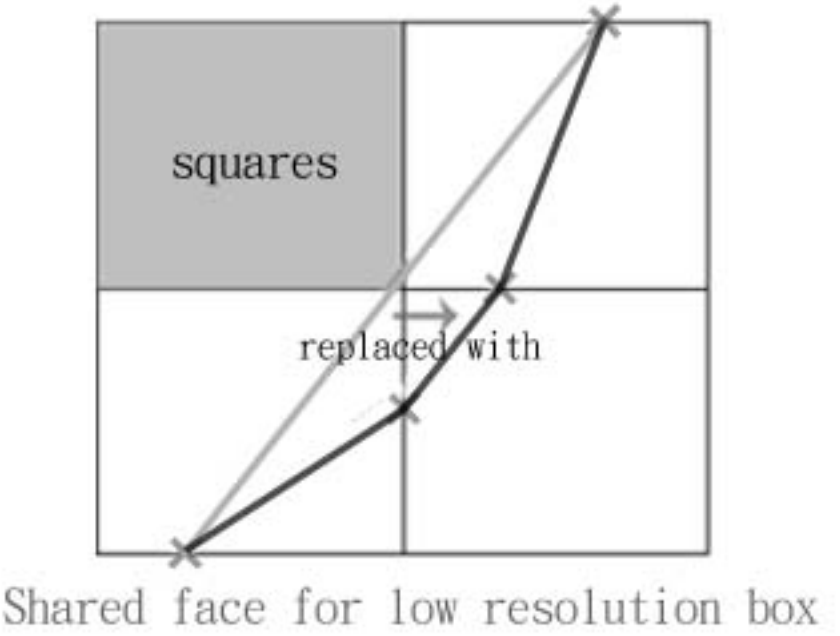


Figure 5.1: Crack on the shared face of low resolution cubic box. It can solve by dividing the low resolution face as squares and and extract corresponding isoline for each square.

5.1 Isoline Loop Extraction

As we mentioned before, we will use isoline loop method with division information to extract triangles for the shared simple cubic boxes. Before showing isoline loop extraction, we will explain the terminology isoline. Isoline, analogous to isosurface, is a line with the same intensity value. It shows on the face of cubic box when isosurface passes through. When isosurface passes through between different resolution simple cubic boxes, the isolines may mismatch each other and the crack problem occurs (Figure 4.6).

As shown in Figure 5.1, we can divide the low resolution face as squares and extract isoline for each square to avoid the crack. By checking the division information of the plots located on the low resolution face, we can divide the shared face with following algorithm:

```
   Pro_Squares(x, y)
{
    if(size of Square > 1){
       Check Div[x] of relative strips according y index.
       Check Div[y] of relative strips according x index.
       if(all are false)
          Output this Square
    else{
          Pro_Squares(2*x, 2*y);
          Pro_Squares(2*x+1, 2*y);
          Pro_Squares(2*x, 2*y+1);
          Pro_Squares(2*x+1, 2*y+1);
    }
    else Output this Square
}
```

For each shared simple cubic box, we will divide the six faces with the algorithm described above and extract the corresponding isolines. In [14], these isolines can be chained as loops and will take 3 consecutive vertices as a triangle each time but different vertex order will result in different surface geometry. Sometimes, the geometry may be incorrect and thus ASC algorithm will check the normal information of each triangle vertex to correct isosurface [14].

5.2 Merging Extraction

As we mentioned before, we extract and merge triangles by DSC algorithm for unshared simple cubic box. Under the assumption of midpoint selection, this algorithm generates a new loop-up table as Figure 5.2. For each unshared cubic box, the facets can be extracted by looking up this table. DMC algorithm will prepared 13 hash tables and then push these facets to corresponding hash table according to the incidence. After finishing facets extraction of the whole volume, this algorithm starts merging operation for the facets in each hash

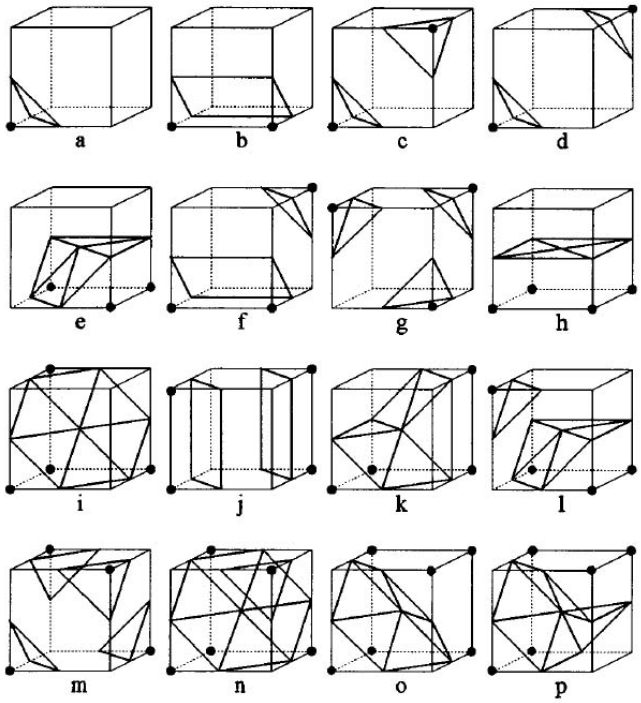


Figure 5.2: The discretized marching cubes look-up table [12].

table. For each facet in a hash table, this algorithm finds the neighboring facets by a hash function and uses Freeman's chain algorithm [20] to chain these facets as a DMC loop. For each DMC loop, this algorithm removes unnecessary vertices and then triangulates it. In our multi-resolution simple cubic boxes, we will utilize the DMC method for different resolution cubic boxes respectively.

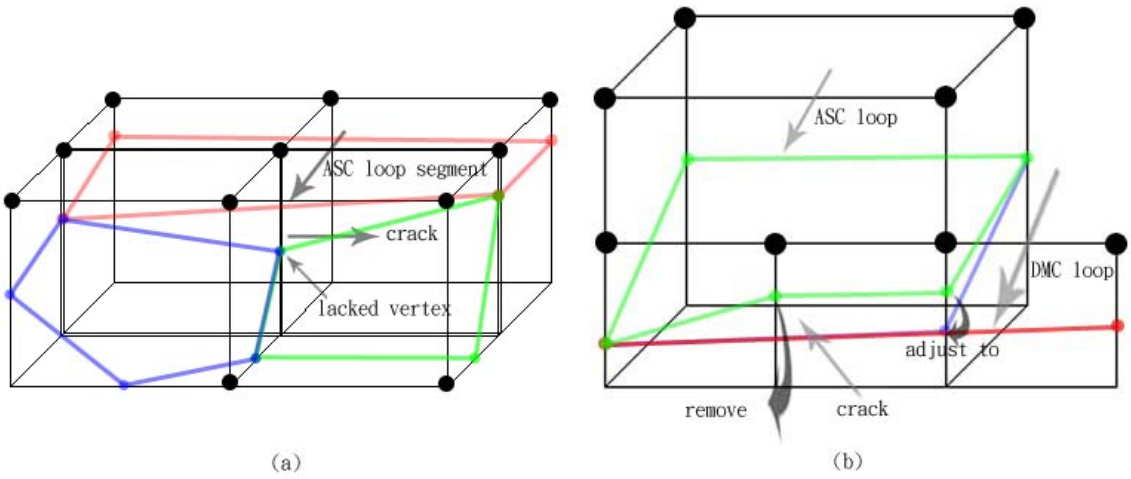


Figure 5.3: (a) The first kind of crack problem occurs between DMC loops. (b) The second kind of crack problem occurs between DMC and ASC loops.

5.3 Vertices Adjustment of Triangle Extracted from Merging Extraction

All possible vertices from DMC algorithm will locate on the mid-position of cube edge thus lower the accuracy. We will interpolate the actual position by adjusting the vertices of triangle extracted from DMC to increase accuracy. This adjustment will cause two kinds of crack problems that will be discussed in later section.

5.4 Crack Problem

The first kind of crack is due to the lack of some vertices on a segment of a DMC loop(see Figure 5.3 (a)) and we will add these vertices before we triangulate the DMC loop.

In order to find these vertices quickly, we can perform the following procedure:

1. Push all the vertices of DMC loops in a hash table by taking the coordinate of vertex as hash key in advance.

2. Get the coordinate of possible added vertex coordinate on each DMC loop segment.

3. Check whether the possible added vertex is in the hash table or not.

4. Add the vertex in current DMC loop if it exist.

As shown in Figure 5.3 (b), the second crack may occur when we blend ASC loop and DMC loop . This crack causes if isoline loop retain some vertices but DMC loop remove them (see Figure 5.3 (b)). For these isoline loop, we will adjust the position of a vertex if this vertex is on the boundary of the shared simple cubic box. If the vertex is not on the boundary of the shared simple cubic box, we will remove it.

5.5 Triangle Degeneration

The triangle in the shared or unshared simple cubic box will degenerate to line or point if the voxels in the box contain intensity values equal to the isovalue. It means that the isosurface passes through the edge or corner point of the simple cubic box. We will remove these degenerated triangles by checking vertex coordinates for each triangle.

Chapter 6

Experiment Results and Discussion

We will compare our method with ASC method, DMC method and MC method according to triangle number, accuracy corresponding with triangle mesh extracted from MC, and extraction time. According to ASC results [14], the triangle number is fewest in most cases when block size N is 4. Thus, we chose the block size N as 4 in the experiments when needed. The accuracy of the surface model is measured by the tool MESH [21]. MESH samples the two triangle mesh as two group of sample point and then calculates the average Hausdorff distance between them. The average error will be represented as the percentage of diagonal distance of volume data. We will evaluate these methods by using volume data sets, including engine, teapot, skull, and cortical surface. The hardware platform contained a AMD Athlon XP 2.01GHz CPU and 1G bytes memory. We also evaluate our method with different block size, (2,4, and 8), in our method.

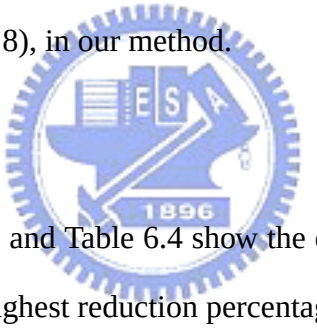


Table 6.1, Table 6.2, Table 6.3, and Table 6.4 show the comparison results. Compared with other methods, we have the highest reduction percentage of triangles for the four volume data sets. The triangle numbers are 4 to 15 times fewer than marching cubes method. The reduction percentage depends on the complexity and smoothness of the volume data and thus the engine data has the highest reduction percentage and the cortical surface data has the lowest reduction percentage. We has the best reduction percentage but we lose the accuracy of reduced triangle mesh. However, the increased amount of surface deviation is small. The efficiency of our method is at least 2 times faster than ASC method but at most 2 times slower than MC and DMC . Figure 6.1, Figure 6.2, Figure 6.3, and Figure 6.4 show the major difference between ASC and our method. As shown in the zoom area, our method merges triangles to generate fewer triangles than ASC does.

Table 6.5 shows the experiment result with different block size in our method. In most case, our method will reduce more triangles when we enlarge N. This is because we may generate larger and fewer triangles for lager block. When N is too small, our method is relatively slow because more efforts are required to analyze the volume data and to generate triangles. When N is too large, our method is also relatively because the kd-tree method is of no use. The efficiency of our method is best when N is 4 for these four data sets.

Table 6.1: Performance comparison for engine data.

Method	Triangle number	Mean error	Extraction time(secs)
MC	584950	0	2.297
DMC	132336	0.244	1.344
ASC	59835	0.117	8.641
Our	49693	0.193	2.922

Data size: $256 \times 256 \times 128$, Isovalue: 50

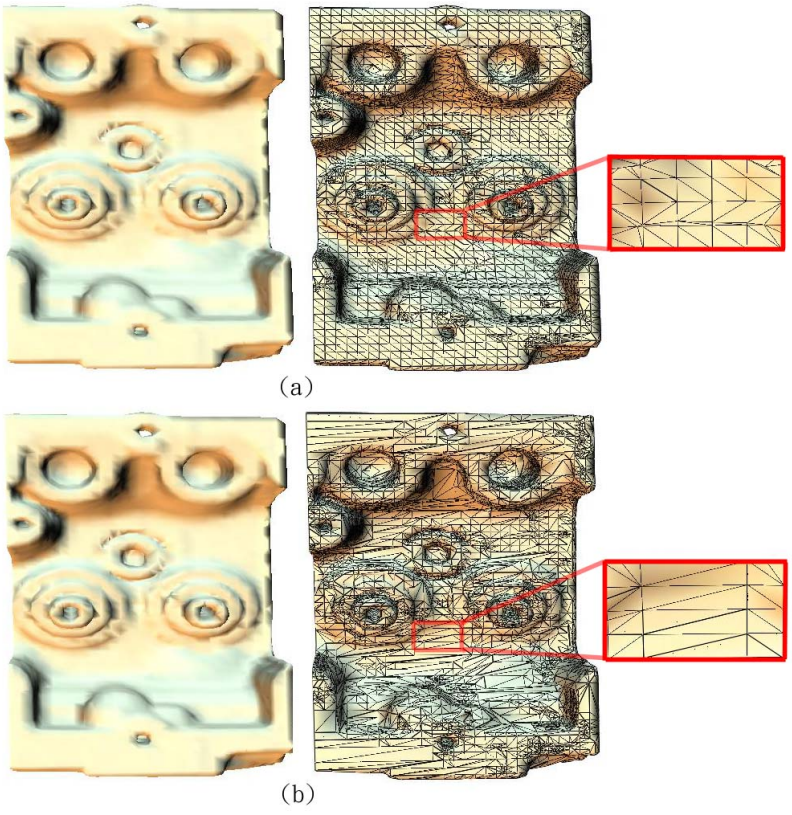


Figure 6.1: The triangle mesh of engine data generated by (a) ASC method and (b) Our method.

Table 6.2: Performance comparison for teapot data.

Method	Triangle number	Mean error	Extraction time(secs)
MC	713144	0	3.062
DMC	155630	0.242	1.797
ASC	66989	0.051	10.891
Our	43351	0.268	3.844

Data size: $256 \times 256 \times 178$, Isovalue: 50

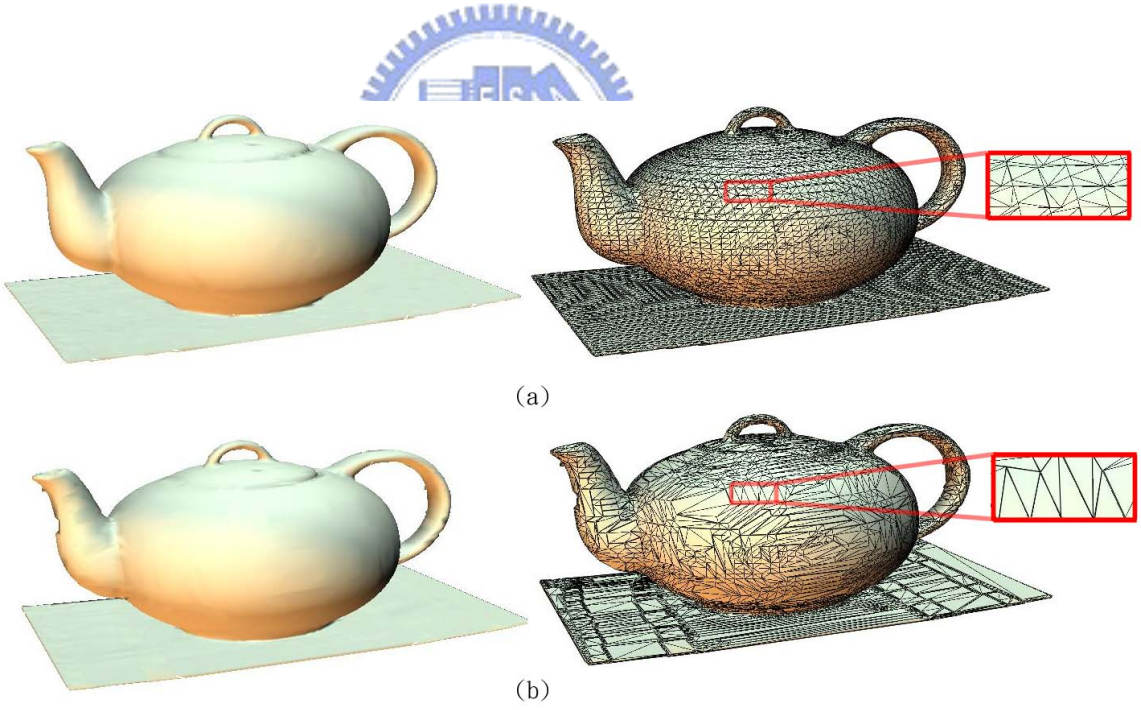


Figure 6.2: The triangle mesh of teapot data generated by (a) ASC method and (b) Our method.

Table 6.3: Performance comparison for skull data.

Method	Triangle number	Mean error	Extraction time(secs)
MC	1044777	0	4.485
DMC	405680	0.255	4.789
ASC	217805	0.114	17.75
Our	213639	0.127	8.266

Data size: $256 \times 256 \times 256$, Isovalue: 50

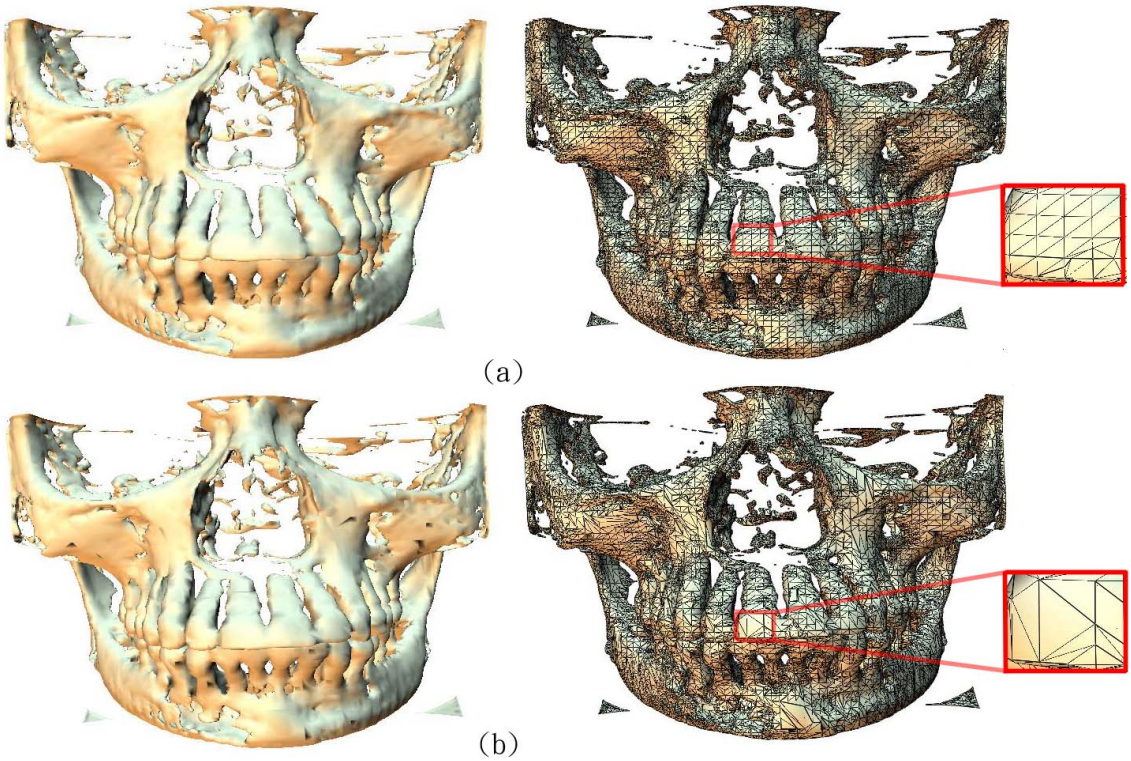


Figure 6.3: The triangle mesh of skull data generated by (a) ASC method and (b) Our method.

Table 6.4: Performance comparison for cortical surface data.

Method	Triangle number	Mean error	Extraction time(secs)
MC	976826	0	1.813
DMC	408470	0.213	2.485
ASC	265575	0.127	10.704
Our	256954	0.124	5.547

Data size: $157 \times 189 \times 156$, Isovalue: 100

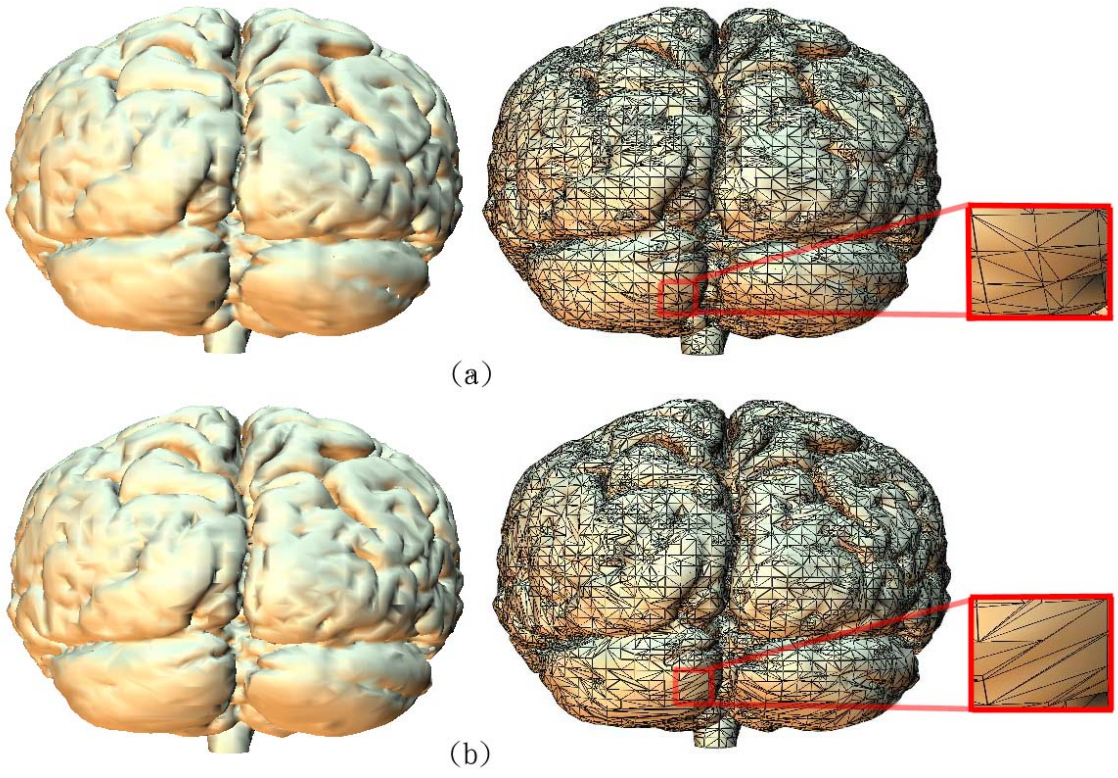


Figure 6.4: The triangle mesh of cortical surface data generated by (a) ASC method and (b) Our method.

Table 6.5: Performance of our method regarding to block size.

Data Set	Our method, N=2	Our method, N=4	Our method, N=8
engine	61100 Δ\ 5.328 sec.	49693 Δ\ 2.922 sec.	47094 Δ\ 5.344 sec.
teapot	61702 Δ\ 6.828 sec.	43351 Δ\ 3.844 sec.	46202 Δ\ 5.688 sec.
skull	228076 Δ\ 13.875 sec.	213639 Δ\ 8.266 sec.	212923 Δ\ 14.437 sec.
brain	274544 Δ\ 11.297 sec.	256954 Δ\ 5.547 sec.	256603 Δ\ 8.828 sec.

Δ: means triangle count

Chapter 7

Conclusions

We proposed in this thesis an adaptive isosurface extraction algorithm which can reduce the triangle number of isosurface by adapting to the variation of the isosurface. The adaptive isosurface contains larger but fewer triangles for low variation area and smaller but more triangles for high variation area. We use kd-tree method to rapidly locate the area containing the isosurface. Then a novel recursive algorithm is applied to analyze the volume data and to generate the triangles whose sizes are adaptive to the surface. According to our experiments, the triangle number obtained by using the proposed method can be at least four times fewer than that can obtained by using the marching cubes method without losing much surface accuracy.

Compared to the ASC method, our method has advantages and disadvantage listed below:

- Advantages:
  - The proposed method can reduce more triangles than ASC method in most cases. The reason is that we can merge the triangles to get larger one which stride through the initial block. The more smooth volume data will make our method to extract the isosurface with fewer triangles.
  - The extraction time of our method is at least double faster than the ASC method because we use the kd-tree to locate the non-empty blocks rapidly and our simple recursive algorithm is very efficient.
- Disadvantages:
  - On the average, the structure extracted by using our method has larger surface deviation than that extracted by using the ASC method.

Bibliography

[1] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of ACM SIGGRAPH Computer Graphics*, pages 163–169, 1987.

[2] John J. Ermer, John C. Mosher, Sylvain. Baillet, and Richard M. Leahy. Rapidly recomputable EEG forward models for realistic head shapes. *Physics in Medicine and Biology*, 46(4):1265–1271, 2001.

[3] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In *Proceedings of ACM SIGGRAPH Computer Graphics*, 26:163–169, 1992.

[4] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Proceedings of ACM SIGGRAPH Computer Graphics*, 27:19–26, 1993.

[5] Raj Shekhar, Elias Fayyad, Roni Yagel, and J. Fredrick Cornhill. Octree-based decimation of Marching Cubes surfaces. In *Proceedings of IEEE Visualization*, pages 335–342, 1996.

[6] Taosong He, Lichan Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel based object simplification. In *Proceedings of IEEE Visualization*, pages 296–303, 1995.

[7] M. Kraus and T. Ertl. Topology-guided downsampling. In *International Workshop on Volume Graphics*, 2001.

[8] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(2):201–227, 1992.

[9] Takayuki ITOH and Koji KOYAMADA. Surface: Isosurface generation by using extrema graphs. In *Proceedings of IEEE Visualization*, pages 77–83, 1994.

[10] Yarden Livnat, Han-Wei Shen, and Christopher R. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, 1996.

[11] Phil Sutton, Chuck Hansen, H.-W. Shen, and Dan Schikore. Surface: A case study of isosurface extraction algorithm performance. In *2nd Joint Eurographics-IEEE TCCG Symposium on Visualization*, 2000.

[12] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In *Proceedings of IEEE Visualization*, pages 281–287, 1994.

[13] Renben Shu, Chen Zhou, and Mohan S. Kankanhalli. Adaptive marching cubes. *The Visual Computer*, 11:202–217, 1995.

[14] Tim Poston, Tien-Tsin Wong, and Pheng-Ann Heng. Multiresolution isosurface extraction with adaptive skeleton climbing. In *EUROGRAPHICS*, 17(3):137–148, 1998.

[15] Yong Zhou, Baoquan Chen, and Arie Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In *Proceedings of IEEE Visualization*, pages 135–142, 1997.

[16] Yong Zhou, Baoquan Chen, and Arie Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In *Computers and Graphics*, 19(3):355–364, 1995.

[17] Thomas Gerstner and Renato Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proceedings of IEEE Visualization*, pages 259–266, 2000.

[18] Laurent Balmelli, Christopher J. Morris, Gabriel Taubin, and Fausto Bernardini. Isosurfaces: Volume warping for adaptive isosurface extraction. In *Proceedings of IEEE Visualization*, pages 467–474, 2002.

[19] Laurent Balmelli, Gabriel Taubin, and Fausto Bernardini. Space-optimized texture maps. In *EUROGRAPHICS*, 21(3):411V-420, 2002.

[20] Herbert Freeman. Computer Processing of Line-Drawing Images. *ACM Computing Surveys*, 6(1):57–97, 1974.

[21] Nicolas Aspert, Diego Santa-Cruz, and Touradj Ebrahimi. MESH: Measuring errors between surfaces using the hausdorff distance. In *IEEE International Conference on Multimedia and Expo*, 1:705V-708, 2002.