

MRI and CT Image Based on 3D Reconstruction and Medical Rapid Prototyping (MRP)

By
Jiman Han

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
In
MECHANICAL ENGINEERING
UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
2005

Approved by:

[Redacted Signature]

Miguel Vélez-Reyes, Ph.D.
Member, Graduate Committee

Nov. 7, 2005

Date

[Redacted Signature]

Frederick A. Just-Agosto, Ph.D.
Member, Graduate Committee

Nov 7, 2005

Date

[Redacted Signature]

Donald C. Dunbar, Ph.D.
Member, Graduate Committee

7 November 2005

Date

[Redacted Signature]

Yi Jia, Ph.D.
President, Graduate Committee

Nov. 7, 2005

Date

[Redacted Signature]

Fernando Gilbes, Ph.D.
Representative of Graduate Office

Nov. 21, 2005

Date

[Redacted Signature]

Paul A. Sundaram Ph.D.
Chairperson of the Department

11/22/05

Date

Abstract

In this research, an interactive software tool has been developed for creating 3D models of anatomical organs and other body structures from 2D medical imaging data. 3D models are generated by using the Marching Cubes algorithm and Planar Contour method by SolidWorks developed in Visual Basic Language. The research includes transferring CT and MRI images into digital binary matrixes, entering digital binary matrixes into SolidWorks environment, building feature library for 3D reconstruction, creating medical rapid prototyping models, and performing biomedical rapid design and manufacturing. CT and MRI images processing is obtained by capturing the patient scan data, converting the image format, extracting the gray scale of bone image, and transferring CT and MRI image into digital binary matrixes. 3D reconstruction is created by edge configuration generation and triangulated cube configuration generation in Marching Cubes algorithm and by capturing section contour points from medical image per slice, creating B-spline curve with the control points in each layer, producing solid model construction in Planar Contours method. Medical rapid prototyping models are performed in SolidWorks, including three views or any combination of views, for biomedical rapid designing and manufacturing according to the biomedical needs. Layered manufacturing techniques are used for producing parts of arbitrary complexity.

The results of this research are the first to develop image processing 3D visualization in SolidWorks Application Programming Interface (API) using Visual Basic Language. The system performance is tested using truth CT and MRI data, and 3D physical models teeth and knee joint for MRP are created directly from SolidWorks. The results reveal that the accuracy of 3D reconstruction is acceptable.

Keywords: 3D Reconstruction, Image Processing, Computer Aided Surgery, Medical Rapid Prototyping, SolidWorks

Sumario

En esta investigación, la herramienta de un software interactivo ha sido desarrollado para crear modelos tridimensionales de órganos anatómicos y otras estructuras del cuerpo humano partiendo de datos obtenidos en imágenes medicas de dos dimensiones. Los modelos tridimensionales son generados por medio de la utilización del algoritmo “Marching Cubes” o cubos marchantes y el método de perfiles de contornos planos del programa SolidWorks desarrollado en lenguaje Visual Basic. La investigación incluye la transferencia de imágenes CT y MRI en matrices digitales binarias ingresándolas al ambiente SolidWorks, y creando una librería de representaciones claves para la reconstrucción tridimensional; de esta forma se generan modelos de RP para aplicaciones medicas y se desarrollan diseños biomédicos rápidos para su fabricación.

El procesamiento de las imágenes de CT o MRI son obtenidas capturando los datos del escaner de un paciente, y convirtiendo el formato de la imagen desde la extracción de datos de la escala de tonalidades grises de las imágenes de huesos y transfiriendo dichas imágenes CT y MRI en matrices digitales binarias. La reconstrucción 3D por medio de la generación de configuración de bordes y la generación de configuración de cubos triangulados en algoritmos de cubos marchantes; de esta manera se capturan los puntos de contorno de cada sección de la imagen medica por capas. De esta forma se van creando curvas tipo B-s por tiras con los puntos de control de cada capa, produciendo así un modelo sólido de construcción por el método ya mencionado de perfiles de contorno planos. Los modelos médicos de estereolitografía o RP son previamente desarrollados en SolidWorks, con las cuales se pueden ver los modelos desde sus tres diferentes vistas o la combinación de las mismas, siendo de gran ventaja para el diseño y fabricación rápido de modelos anatómicos de acuerdo con necesidades biomédicas.

Las técnicas de manufactura por capas o laminados son usadas para producir partes de complejidad arbitraria. El principal objetivo de este proyecto es desarrollar el procesamiento de visualización de imágenes 3D en SolidWorks mediante Interfase de

Aplicación Programada (API) usando lenguaje Visual Basic. Los datos para generar los modelos en 3D para la estereolitografía o prototipado rápido (RP) son creados simultáneamente. El desarrollo del sistema es probado usando datos reales de CT y MRI y un ejemplo de modelo de RP de dientes o articulaciones de rodilla fueron manufacturados. Los resultados revelan que la exactitud de la reconstrucción 3D es bastante aceptable.

Palabras Claves: Reconstrucción Tridimensional, Procesamiento de imágenes, Cirugía asistida por computador, Estereolitografía Medica, SolidWorks

Acknowledgments

First of all, I would like to express my gratitude towards my advisor, Dr. Yi, Jia, who was an excellent mentor and who supported and guided me over the years. He provided an ideal research environment for exploring new ideas and every discussion with him was a learning experience.

I would also like to thank Dr. Miguel Velez-Reyes, Dr. Frederick A. Just-Agosto, and Dr. Donald C. Dunbar for serving on my advisory committee and giving me invaluable comments and suggestions about my research. I deeply thank Alexander Pulliza for his Rapid Prototyping manufacturing in 3D physical models, Luvina Reyes for her translation abstract from English to Spanish, and Ke Sun for his help in my presentation. I appreciate their time and insights.

Most of all, I thank my husband Tian Yu and my daughter Jenny Han Yu. The tremendous efforts involved in conducting my research thesis would not have been possible without their love, support and endless patience.

This research has been supported in part by the NIH-MBRS SCORE Program, University of Puerto Rico, Mayaguez campus.

To my husband Tian Yu and my daughter Han Yu (Jenny)

Table of Contents

Abstract.....	II
Sumario.....	III
Acknowledgments.....	V
Table of Contents.....	VI
List of Figures.....	VIII
List of Tables.....	XI
Chapter 1 Introduction	
1.1 Introduction.....	1
1.2 Backgrounds and Related Work.....	3
1.2.1 Iso-surfacing Reconstruction.....	3
1.2.2 Direct Volume Rendering.....	7
1.3 SolidWorks.....	12
1.4 Motivation.....	13
1.5 Objective.....	14
1.6 Frame of Thesis.....	15
Chapter 2 CT and MRI Digital Image Processing	
2.1 CT and MRI Image.....	19
2.2 Image Processing.....	21
Chapter 3 Marching Cubes Algorithm	
3.1 Marching Cubes Algorithm.....	27
3.2 Edge Configuration Generation.....	29
3.3 Triangulated Cube Configuration Generation.....	32
Chapter 4 Feature Library and Database Library	
4.1 Feature Library.....	38
4.1.1 Building Feature Library	38
4.1.2 Type of Feature Library.....	40

4.2 Database.....	42
Chapter 5 3D Reconstruction by Marching Cubes	
5.1 Programming Development.....	44
5.1.1 Flow Chart.....	44
5.1.2 Creation of Macro File.....	44
5.1.3 Image Transformation.....	46
5.1.4 Reading Database.....	47
5.1.5 Determining Cube Index.....	49
5.1.6 Display of 3D Reconstruction Model.....	49
5.2 Results and Discussions.....	50
Chapter 6 3D Reconstruction by Planar Contours	
6.1 Section Contour Points	58
6.2 B-spline Curve Creation.....	60
6.3 Solid Model Construction.....	61
6.4 Results of Knee Joint Solid Models.....	63
Chapter 7 Biomedical Rapid Prototyping	
7.1 Rapid Prototyping Technologies.....	67
7.2 RP Application in Biomedical Field.....	76
7.2.1 Surgical Planning	76
7.2.2 Anatomical Implants.....	77
7.3 Biomedical Rapid Design and Manufacturing.....	78
Chapter 8 Conclusions and Future Works	
8.1 Conclusions	87
8.2 Future Works.....	88
References	89
Appendix A: Feature Library	95
Appendix B: Programming Codes	103

List of Figures

Figure 1.1 Method of Planar Contours.....	4
Figure 1.2 Marching Cube.....	5
Figure 1.3 March Cube with Triangle Isovalue Surface.....	6
Figure 1.4 Representative Case for Triangular in Double-Time Cubes.....	7
Figure 1.5 Transformations from Object Space to Sheared Object Space.....	8
Figure 1.6 Transformations from Object Space to Sheared Object Space.....	8
Figure 1.7 the Architecture for 3D Texture Mapping based Volume Rendering.....	10
Figure 1.8 the Principles of Ray Casting.....	10
Figure 1.9 Examples of SolidWorks 3D Models.....	12
Figure 1.10 the Produce and Main Milestones of the Thesis	17
Figure 2.1 Medical Image of Skull.....	20
Figure 2.2 Gray-level Thresholding Functions.....	22
Figure 2.3 Arrangement of Pixels.....	23
Figure 2.4 CT Scan Image of a Head.....	24
Figure 2.5 Inverted Cleaned Image.....	25
Figure 2.6 Digital Binary Matrixes File	25
Figure 3.1 Definitions in Marching Cubes Algorithm	28
Figure 3.2 Vertex Classifications.....	28
Figure 3.3 Vertex Index and Edge Index.....	28
Figure 3.4 Cube Index	29
Figure 3.5 Edge Configuration Generation.....	30
Figure 3.6 Interpolation in 1D Case between i^{th} and $(i+1)^{th}$ Point	31
Figure 3.7 2D and 3D Ambiguity.....	37
Figure 4.1 Establishment of Base Feature and Feature Library.....	39
Figure 5.1 Flow Chart of 3D Reconstruction.....	45
Figure 5.2 Result from CT Scan to 3D model for Teeth.....	53

Figure 5.3 Result from CT Scan to 3D model for Knee.....	55
Figure 6.1 Section Contour Points Capture Flow Chart.....	59
Figure 6.2 Section Contour Points Capture in SolidWorks.....	60
Figure 6.3 B-spline Curve Creation.....	61
Figure 6.4 18 Layers Closed Curves Selected for a Loft Feature	62
Figure 6.5 Solid Model by Planar Contours Method.....	62
Figure 6.6 Solid Model rendered in SolidWorks.....	63
Figure 6.7 (a) Capturing Contour Points.....	64
Figure 6.7 (b) Creating B-spline Curves.....	64
Figure 6.7 (c) Lofting Features.....	65
Figure 6.7 (d) Building Solid Models.....	65
Figure 6.7 (e) Rendering Solid Models.....	66
Figure 7.1 RP Model	69
Figure 7.2 Process of RP for SLA Technique.....	70
Figure 7.3 the Process of Selective Laser Sintering.....	72
Figure 7.4 the Process Fused Deposition Modeling.....	74
Figure 7.5 Z Corp. 3D Printer.....	75
Figure 7.6 Z Corp. 3D Printer in Making 3D Solid Model Processes.....	75
Figure 7.7 RP model for Surgery Planning.....	77
Figure 7.8 RP model for Implant.....	78
Figure 7.9 Medical Design of 3D Reconstruction Model.....	79
Figure 7.10 2D View of 3D Model.....	79
Figure 7.11 (a) 3D Model in Layer by Layer.....	81
Figure 7.11 (b) Biomedical Manufacturing with Bolt Joint.....	82
Figure 7.11 (c) Biomedical Manufacturing with Joint Hole.....	82
Figure 7.12 (a) 3D Physical Model of Knee Joint	83
Figure 7.12 (b) D Physical Model of Knee Joint Layer by Layer.....	84

Figure 7.12 (c) D Physical Model of Teeth.....84

List of Tables

Table 3.1 Triangulated Cube Configuration.....	33
Table 4.1 Feature Library.....	40
Table 5.1 File Names and Functions.....	50

Chapter 1 Introduction

1.1 Introduction

The three-dimensional (3D) reconstruction of human anatomical organs and structures from a series of cross section image has been an intriguing problem in recent decades. New challenges have been created in the field of image analysis and pattern recognition by the introduction of modern image data collection techniques such as Computed Tomography (CT) and Magnetic Resonance Imaging (MRI). With the development of advanced bio-medical techniques, 3D geometric representations of human anatomical organs rather than the two-dimensional (2D) photographic images using CT or MRI are frequently required. These 3D geometric models, either simulation generated by computer or 3D Rapid Prototyping (RP), can be used for diagnosis of physical disorders, visualization of anatomical organs for surgical planning, and the implantation of human organs and other structures. RP is the process of converting a 3D Computer Aided Design (CAD) file into a 3D physical model “rapidly”. Medical Rapid Prototyping (MRP) is the production of the medical models using rapid prototyping methods [1-3]. The application of RP techniques is an invaluable contribution of engineering technology to the field of medicine [4]. For example, if a patient requires multi surgeries to repair severely fractured skull, a 3D physical model of the patient’s skull could be constructed prior to surgery using RP technology in order to visualize the trauma and make informal decisions on how to proceed into repair.

The MRP process contains three stages. First, a medical image dataset is obtained using a CT or MRI on other medical scanner. Second, this dataset is processed to obtain a polygonal surface model. Third, the polygonal surface model is imported into a rapid prototyping machine using the STL format. Producing 3D model consists of two steps: segmentation and extraction. A medical image always contains multiple tissues and bones.

Segmentation is the name given to the algorithms used to isolate the desired bones from the tissues in the image. After the bones are isolated, the extraction technique constructs 3D representations of the bones. Most extractions are based on the concept of the isosurface [5-8]. An isosurface is a surface passing through a medical image volume that corresponds to a given integer value. The location of this surface can be computed in many ways, but the simplest and most common method is called the Marching Cubes algorithm.

Applications do currently exist for 3D reconstruction of organs and other structures from CT images. Fabien Vivodtzev, et al [9] provided an approach to describe the characteristics of a surface, which is to segment it into regions of uniform curvature behavior and to construct an abstract representation given by a topology graph for human brain. Armin Kanitsar et al [10] presented the visualization of tubular structures, such as blood vessels, by using CPR (Curved Planar Reformation). Yongjie Zhang et al [11] gave an algorithm to extract adaptive and quality 3D meshes directly from volumetric CT and MRI imaging data. Kunio Nakamura [12] uses Marching Cubes surface construction algorithm to render the volumetric image to lead to faster image registration refinement for the human brain. In 1989, the National Library of Medicine (NLM) began an ambitious project to create a digital atlas of human anatomy using Marching Cubes algorithm. The extracted tetrahedral and hexahedral meshes are extensively used in finite element simulations. Stefan Futterling et al [13] generate a 3D finite element model of an individual patient's mandible inserted with dental implants. These geometric models are converted to finite element models using adaptive tetrahedral meshing [14]. Several research institutions and commercial organizations integrate CAD and RP systems with medical imaging systems to fabricate medical devices or generate 3D hard copies of these objects for uses in surgical rehearsal, custom implant design, and casting [15, 16].

Very few medical researchers, however, are interested in Medical Rapid Prototyping (MRP) because this technology is useful only in product design and manufacturing, not

as a surgical tool. Moreover, correctly converting and visualizing the 3D geometry of anatomical organs on structures from 2D medical images for MRP manufacturing is difficult.

1.2 Backgrounds and Related Work

Over the last twenty years, the techniques of CT and MRI have developed rapidly. Researchers have reported many different methods to visualize the 3D geometry of such structures. 3D reconstruction includes two main approaches: *Iso-surfacing Reconstruction* and *Direct Volume Rendering* [17-19]. Iso-surfacing Reconstruction consists of two major classes: (1) Planar Contour and (2) Marching Cubes and Double-Time Cubes. Direct Volume Rendering includes Shear-Warp, 3D Texture Mapping, and Ray Casting.

1.2.1 Iso-surfacing Reconstruction

The majority of the reconstruction techniques produce planar approximations of the data set. There are primarily two classes of surface reconstruction techniques.

(1) Planar Contours

The first class of surface reconstruction methods initially constructs planar contours in each CT/MRI data slice and then connects these contours by a triangulation in three-dimensional space. The triangulation process is complicated by the occurrence of multiple contours on a data slice.

Figure 1.1 shows the approximation method. The method consists of joining points of neighboring contour lines to triangles in such a manner that one obtains triangular planar elements, which delimit a polyhedron approximating the surface of interest.

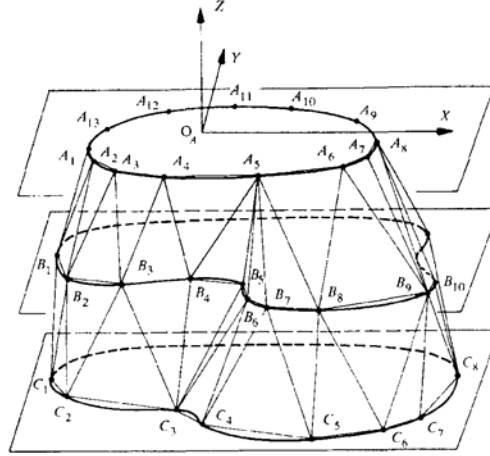


Fig. 1.1 Method of Planar Contours

Early contributions to this method were made by E. Keppel [20] and Fuchs et al [21], Christiansen and Sederberg [22] and F. Preparata [23]. The optimal algorithms due to the work by Keppel [20] and Fuchs et al [21] computes a graph in which each node represents a spanning arc and each edge in the graph represents a triangle defined uniquely by two spanning arcs that share a point. A shortest path algorithm is used to find the path that corresponds to the triangulation of minimum weight. Another algorithm of a heuristic nature, due to Christiansen and Sederberg [22], performs a seeking walk around each adjacent pair of contours selecting segments. A segment on one contour and a point on the other define the triangle. The choice of the new segment is determined by the triangle with the shorter edge length.

The algorithm of Buissonnat computes a 2D Delaunay triangulation [24] of the planar contours and then a 3D Delaunay triangulation of the entire collection of triangles lying in parallel planes. However, a major complaint about Delaunay-based methods is that they are slow and cannot handle amounts of data. Tamal K. Dey, Joachim Giesen, and James Hudson [25] adjusted this criticism by extending the algorithm to handle supersize data. This modification is the first Delaunay-based surface reconstruction algorithm that can handle data containing more than a million sample points on a model machine.

The limitation of Planar Contours is that the connected contour algorithms throw away the inter-slice connectivity that exists in the original data. The triangulation problem resides in the fact that contour lines do not contain sufficient information regarding the system of gradients associated with the surface they describe. Moreover, the combinatorial aspect of the problem becomes apparent when one considers how many different triangular arrangements can be constructed for a fixed number of contour points.

(2) The second class of surface reconstruct methods mostly includes the follows algorithms:

Marching Cubes algorithm --- William E. Lorensen, Harvey E. Cline, 1987 [26]

The algorithm, which is one of the most famous volume visualization techniques, creates a representation that consists of triangles of an isovalue surface. The triangles are then rendered to produce an image. Marching Cubes consists of eight sample points, known as voxels (volume elements). An octuple of neighboring voxels represents a cube, which from a sub-cube are used to create triangles at one time. Depending on whether the voxels are within or outside the object, a surface of up to four triangles is placed inside the cube. Then the algorithm “marches” on to the next sub cube in scan-line order (See Figure 1.2)

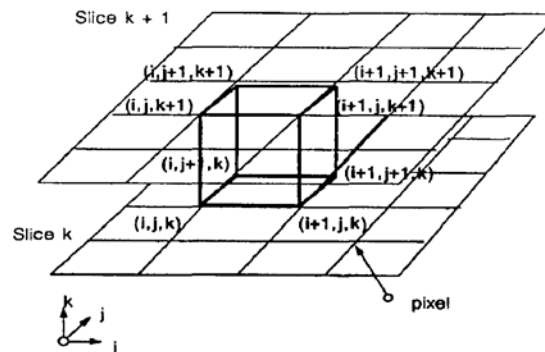


Fig. 1.2 Marching Cube

The original Marching Cubes algorithm identified 256 configurations for the cube, depending on whether each of the eight vertices is inside or outside the object. Figure 1.3,

for example, shows a triangle for a sub cube. The large point in the figure is included in the isovalue surface forming the triangle surface because it is inside the object. The voxels at other corners do not participate in the isovalue surface. Marching Cubes uses linear interpolation between voxel values to compute the location of the triangles' vertices. The result of considering all sub-cube in this way is a collection of triangles, which approximate the location of an isovalue surface.

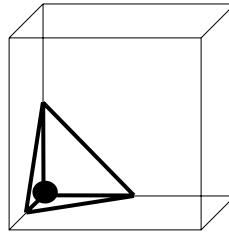


Fig. 1.3 March Cube with Triangle Isovvalue Surface

Since its original conception, the Marching Cubes algorithm has been the subject of much further research to improve the quality of its surface representation and performance on large data sets [27-30]. Adriano Lopes and Ken Brodlie were concerned with the quality of representation of the trilinear interpolating surface [31].

The advantage of the Marching Cubes algorithm is that the resulting triangle model can be displayed on conventional graphics systems using standard rendering algorithms because it uses information from the original 3D data to derive inter-slice connectivity, surface location, and surface gradient. In addition, because the algorithm uses a case table of the edge intersections to describe how a surface cuts through each cube in a 3D data set, programming is convenient and performance is fast.

Double-Time Cubes --- Dennis J. Bouvier, 1992 [32]

The Double-Time Cubes algorithm creates a representation of an iso-value surface from volume data. As with Marching Cubes the algorithm considers sub-cube of eight voxels at a time, in scan-line order. The Double-Time Cube algorithm differs from Marching Cubes, however, in that it places the triangle's vertices at voxel locations. Thus, interpolation is not necessary. Figure 1.4 shows the representative case for triangulation

in Double-Time Cubes

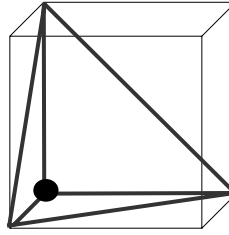


Fig. 1.4 Representative Case for Triangular in Double-Time Cubes

Although the Double-Time Cubes algorithm is a faster alternative to Marching Cubes for generating an estimated isovalue surface from volumetric data, it is limited in that it computes a less detailed representation compared to Marching Cubes, resulting representations that are rougher approximations composed of fewer triangles.

1.2.2 Direct Volume Rendering

Three main direct volume rendering algorithms have emerged in the last decade and are widely used: the Shear-Warp method, the 3D Texture Mapping technique, and the Ray Casting algorithm.

Shear-Warp --- P. Lacroute and M. Levoy, 1994 [33] and Shin Yi Yen, 1996 [34]

Shear-Warp considers the volume as a stack of 2D slices that parallel to the face of the volum. The algorithm factors the viewing transformation into three components: a 3D shear parallel to the data slices, a projection to form an intermediate but distorted image, and a 2D warp to form the undistorted final image. 3D shear transforms the volume into an intermediate coordinate system, for which there is a simple mapping from the object coordinate system that allows for efficient projection. In the intermediate coordinate system, called the sheared object space, all viewing rays are parallel to the principal viewing axis, which is defined as the main axis in object space. This axis is the most parallel to the viewing direction. The volume data is projected in the sheared-object space, forming a distorted intermediate image. It is efficient to project in the sheared-object space because the transformation applied to each slice for parallel projections consists of

only a translation. The final 2D warp to produce the undistorted image is accomplished by using a general purpose affined image warp with a bilinear filter. This part of the computation is relatively inexpensive because the 2D image is small compared to the size of the volume.

Figure 1.5 illustrates the transformation of the parallel projection from object space to sheared object space. The volume is sampled on a rectilinear grid. The horizontal lines in the figure represent slices of the volume data viewed in cross-section. After transformation, the volume data have been sheared parallel to the set of slices that is most perpendicular to the viewing direction, and the viewing rays are perpendicular to the slices. The term “perspective transformation” implies that each slice must be scaled as well as sheared, as shown schematically in Figure 1.6.

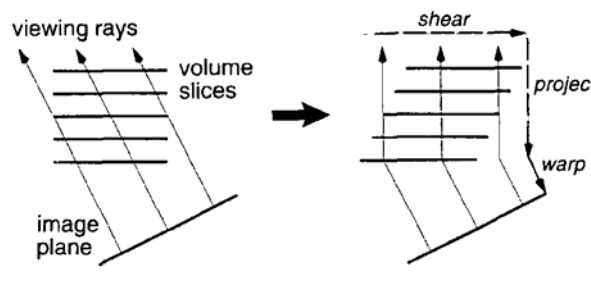


Fig. 1.5 Transformation from Object Space to Sheared Object Space

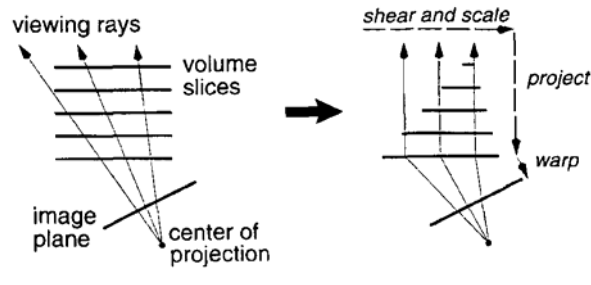


Fig. 1.6 Transformation from Object Space to Sheared Object Space

Shin Yi Yen [34] extended the shear warp method by presenting a fast-sliding, thin-slab volume visualization method. This method renders only those portions of the data within the acquired volume that lie between a set of parallel clipping planes oriented perpendicular to the viewing direction.

The Shear-Warp algorithm has its limitations. First, the sampling rate on the z-axis is

between 1 mm and 1.73 mm according to viewpoint, which is inadequate for observing thin volume structures. Second, the pre-classification partially blurs the intermediate image, which is increased by the final re-sampling step. Finally, artifacts due to the bilinear interpolation occur when the viewing angle is close to 45° . Thus, the global quality provided by the original implementation is poor.

3D Texture Mapping ---- Frank D. et al, 1998 [35] and Allen Van Gelder, et al, 1996 [36]

The 3D Texture Mapping first computes the quantized gradient index and material classification of each voxel in volume. A voxel may be classified as either reflecting or ambient, depending on the client-supplied gradient-magnitude threshold. The result of this step is an index for each voxel into the lookup table. With the pre-assigned look-up table index of each voxel, 3D texture maps are filled with pre-computed color values. These texture maps are then processed in the back-to-front visibility order of the partitions of the volume that they represent. In 3D texture mapping, each polygon vertex is given a point in the texture space, and the graphics system maps values from the texture map onto the polygon surface by interpolating texture coordinates.

Figure 1.7 shows the current architecture in which density and gradients are initially loaded into the texture memory and then re-sampled by the texture hardware along rays cast through the volume. The sample data for each ray (or slice) is then transferred to a buffer in main memory and shaded by the CPU. The shaded samples along a ray are composed and the final pixels are moved to the frame buffer for display. Alternatively, within the same architecture, the shaded voxels can be composed by the frame buffer.

The major advantage of the 3D Texture Mapping is that after the original data is converted into 3D texture maps, the texture hardware can perform the rendering and composing of squares very quickly. The approach is limited, however, to binary classification and diffuse shading. Although real-time rendering rates can be implemented, they do not exceed 2 frames per second for 256^3 volume due to the required high

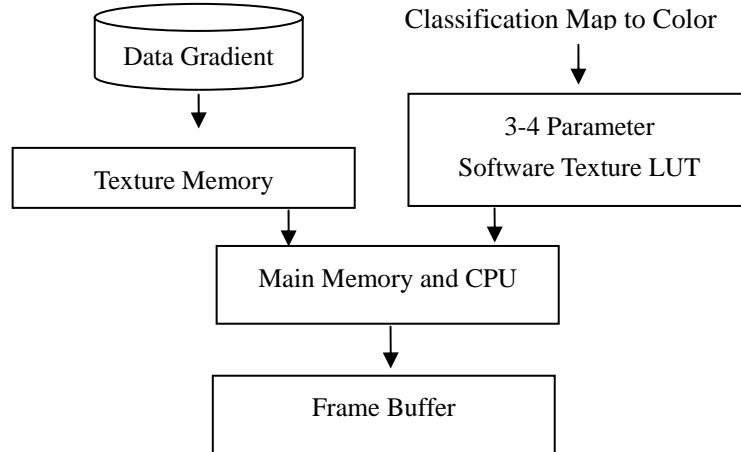


Fig. 1.7 the Architecture for 3D Texture Mapping based Volume Rendering

memory bandwidth. Furthermore, the large distance between the extracted slices can miss small details or produce artifacts.

Ray Casting method - B. M. et al [37], Farrell, E. J. [38] and Hohne, K. H. et al [39]

The volumetric Ray Casting algorithm sends a ray into the scene for each pixel on the object (See (a) in Figure 1.8). Starting at the point where the ray enters the volume (See (b) in Figure 1.8), the ray is followed while sampling the volume at constant distances (See (d) in Figure 1.8). It accumulates (composites) the colors and opacities of these sample values. The ray is no longer followed when the value cannot change significantly. That is, when it has accumulated an opaque color or when it is no longer inside the volume (See (c) in Figure 1.8).

There are two classes of Ray Casting strategies when scanning the volume. The first class consists of object-space oriented methods, which scan along lines or columns of the

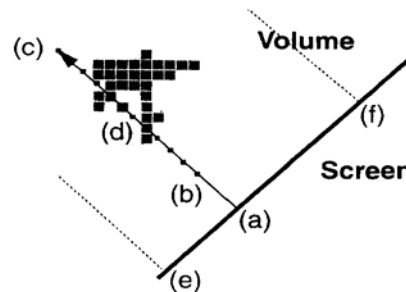


Fig. 1.8 the Principles of Ray Casting

3D-array and project a chosen aspect onto an image plane in the direction of view when a pure surface display of a single object is required. The second class consist of image- space oriented methods, which scan the image volume along the viewing direction to visualize translucency and other volumetric properties.

Farrel [38] used Ray Casting to find the 3D surface. Rather than shade the image with a gray scale, however, he used hue lightness to display the surface. In another Ray Casting method used after the surface along a ray is located, Hohne et al [39] calculated the gradient along the surface, scale by an “appropriate” value and generated gray scales for the image.

Ray Casting is a good way to produce quality images because trilinear interpolation can easily be implemented [40, 41]. In addition, the incoherency between the rays reduces greatly the staircase artifacts visible in algorithms extracting 2D planes. The limitation of this algorithm, however, is that, because it is a pixel-by-pixel approach, it is naturally slow. Every time the ray steps forward within the volume, eight samples have to be loaded before performing trilinear interpolation. This creates cache misses because the samples cannot be stored in memory order. Furthermore, other rays rerunning the same cell may not take advantage of the preloaded data in the cache because the cache lines are often replaced by other data. The second drawback is the difficulty in skipping empty regions of the volume, especially when interactive classification is needed.

In short, the limitation of Planar Contours is that the connected contour algorithms throw away the inter-slice connectivity that exists in the original data. Although a faster alternative to the Marching Cubes algorithm for the generation of an estimated iso-value surface in volumetric data, the Double-Time Cubes algorithm is limited in that it computes a less-detailed representation, resulting in a rougher approximation composed of fewer triangles. Regarding quality, Shear-Warp algorithm also has its drawbacks; the sampling rate on the z-axis is definitely not enough for the observation of thin volume structures, and the global quality provided by the original implementation is poor. Ray

Casting algorithm is naturally slow because it is a pixel-by-pixel approach. Every time the ray steps forward within the volume, eight samples have to be loaded before performing trilinear interpolation. Although Direct Volume Rendering can use volumetric data directly, it does not build a 3D model of the surface. Marching Cubes, however, exploits the property of simple rendering, manipulation. Planar Contour has high resolution.

1.3 SolidWorks

In the engineering applications fields, SolidWorks software, which is developed by SolidWorks Corporation for fully 3D integrated mechanical design software, has the best-in-class capability to re-recognize different types of features, combine multiple features inside the new feature tree, and recognize draft and rib features. Figure 1.9 shows two SolidWorks 3D models in mechanical design applications.

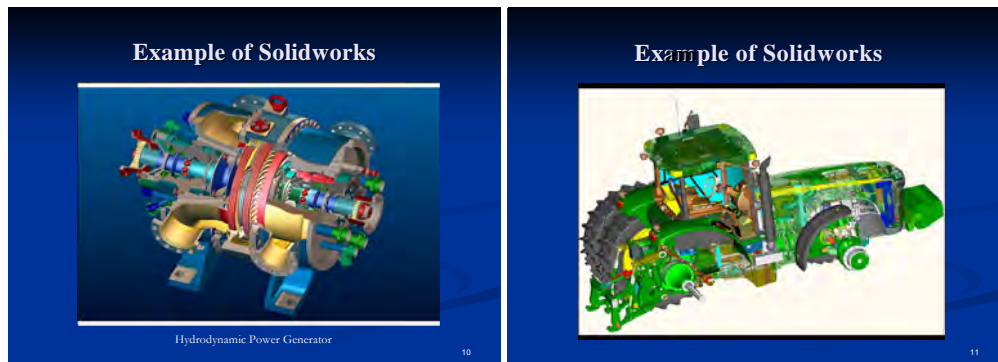


Fig. 1.9 Examples of SolidWorks 3D Models

SolidWorks can output CAD files in a format suitable for medical device design: *step*, *iges*, *sldprt*, *stl*, *fea*, *cfld*, *obj* ... etc. There are 3 basic types of 3D files: the first, surface/graphics files, like *stl*, can be visualized and used for rapid prototyping manufacturing. Second, solid files, like *iges* and *step*, are made of curves and surfaces. These files can be imported into CAD and manipulated. The third, parametric files are composed of solid geometric shapes such as spheres, cones, cubes, etc. These files are native to CAD packages and provide more maneuverability than any of the other file types.

SolidWorks has a drive into the medical device technology market. Its aim is to underline the considerable advantages that medical equipment manufacturers can derive by building their design strategy around advanced 3D modeling technology from initial product development to data management, stress analysis on new designs, and 3D printer for creating rapid prototypes manufacturing.

1.4 Motivation

Computer-based 3D visualization techniques have already had a large impact on the field of medicine. The generation of 3D human anatomical organ from CT or MRI images has many applications in biomedical field. At present, 3D-DOCTOR is an advanced 3D imaging software for visualization, surface modeling and volume rendering, 3D measurement of CT, MRI, microscopy and other 3D images, which exports 3D models to STL for rapid prototyping machines.

Relatively little research has been done, however, in either 3D biomedical image processing or computation biomechanical modeling in SolidWorks environment where the 3D physical model of MRP is created from CT or MRI images for biomedical rapid design and manufacturing. SolidWorks possesses 3D image segmentation, feature extraction, surface generation, and volume rendering. The 3D display and image handling requirements are handled very efficiently in SolidWorks and interactive 3D graphics display and animation requirements become not only possible, but also practical. For example, medical doctors will be able to do all their visualization and analysis from their own desktop computer and eventually from remote locations with just a laptop and on-line access to the Internet.

SolidWorks is fully focused on design, which streamlines the entire design process. Change dimensions, relationships, and geometry at any time or roll back and reorder features easily. Full constraint of biomedical models and assemblies is unnecessary. Design data is 100% editable, and relationships between biomedical models, assemblies,

and drawings always stay up-to-date. With SolidWorks, Designer can reference other biomedical models directly and maintain relationships when creating new biomedical model.

SolidWorks provides not only ease of use, powerful contact modeling, speed, accuracy, and reliability, but also enables immediate access to the most effective rapid prototyping technologies in the industry, such as Print3D, including stereolithography (SLA), fused deposition modeling (FDM), selective laser sintering, and rapid injection molding. With the development of MRP science and technology, surgeons worldwide are using these models to enhance preoperative planning for complex reconstructive surgeries and custom implant design. Applying RP technologies to the medical field, however, differs radically from its application in manufacturing environments. In manufacturing, models are planned and conceived entirely on the computer screen, and then converted to physical reality. In bio-medical applications, by contrast of interest the objects usually already exist physically. Therefore, building medical models essentially involves reverse engineering, starting with acquiring data, such as a stack of CT or MRI cross-sectional images. Prior to model building, these highly complex data need extensive preprocessing to provide a format that a CAD program can utilize. It is difficult and complex process to transfer CT or MRI data to a RP system directly. Therefore, the challenge is to create a 3D model of human anatomical organs and structure from CT or MRI data that is both accurate and beneficial for biomedical applications. This research, in which 3D reconstruction is created and rapid prototyping is built from CT and MRI image in SolidWorks, is put forward for biomedical rapid design and manufacturing.

1.5 Objective

The objectives of this project are three-field. The first objective is to develop an interactive software tool that can create 3D anatomical models from the original cross-sectional CT scan or MRI data, based on Marching Cubes algorithm and Planar Contour

method. Visual Basic computer language in the SolidWorks environment will be used to develop the software tool. The second objective is to convert 3D images into RP data. The third filed objective is to evaluate the systems performance companion with the actual set of CT or MRI scans shown with the 3D reconstruction and MRP.

1.6 Frame of Thesis

The thesis is organized as follows:

This chapter is divided into three parts. Firstly, 3D reconstruction techniques are surveyed. Two basic approaches are discussed: iso-surfacing reconstruction and direct volume rendering. In the second part, SolidWorks is surveyed. In third section, motivation, objective, and frame of thesis are performed.

Chapter two discusses CT and MRI images and transformation CT and MRI image into digital binary matrix by image processing.

Chapter three and four constitute a major component of the thesis. The discussion is Marching Cubes algorithm and strategy of 3D reconstruction, library feature establishment, and the software programming in SolidWorks environment developed by Visual Basic language.

Chapter five describes the actual program development with the CT and MRI image. Then, it provides the 3D model observer study, test and discussion.

Chapter six is divided into two parts to present 3D reconstruction by using Planar Contours method and the results of knee and teeth solid models.

Chapter seven begin with a general discussion of the principle and application in medical rapid prototyping. It then uses SolidWorks interface to convert 3D model data into the medical rapid prototyping and manufacturing. Final, the chapter finishes the discussion of the result of MRP application and test.

Figure 1.10 shows the produce and main milestones of the thesis.

In this thesis, an interactive software tool has been developed to create 3D models of

anatomical organs or other structures from 2D medical data (CT or MRI). Marching Cubes algorithm and Planar Contour method in SolidWorks® environment were used in which a 3D model is converted into STL file data for MRP manufacturing. The research includes transferring CT and MRI images into digital binary matrixes, entering digital binary matrixes into SolidWorks environment, building feature library for 3D reconstruction, creating medical rapid prototyping models, and providing biomedical rapid design and manufacturing. CT and MRI images processing is obtained by capturing the patient scan data, converting the image format, extracting the gray scale of bone image, and transferring CT and MRI image into digital binary matrixes. 3D reconstruction is created by edge configuration generation and triangulated cube configuration generation in Marching Cubes algorithm and by capturing section contour points from medical image per slice, creating B-spline curve with the control points in each layer, producing solid model construction in Planar Contours method. Medical rapid prototyping models are performed in SolidWorks, including three views or any combination of views, for biomedical rapid designing and manufacturing according to the biomedical needs. Layered manufacturing techniques are used for producing parts of arbitrary complexity.

This effort is the first to develop image processing 3D visualization in SolidWorks Application Programming Interface (API) using Visual Basic Language. Rapid Prototyping data from 3D models is created simultaneously. The system performance is tested using truth CT and MRI data, and RP example models of teeth and knee joint were manufactured for MRP manufacturing, which helps the surgeon to prepare the operations in close detail, produce biomedical implants for organs replacement, provide engineering testing, and perform various bio-mechanic simulations.



Scan Image

Data Processing

- 1) *Convert the image format*
- 2) *Obtain the patient scan data.*
- 3) *Invert the colors.*
- 4) *Extract the bone image*
- 5) *Convert the image to a digital binary image matrix*

3D Representation (Marching Cubes)

- 1) *Cube intersections.*
- 2) *Edge configuration generation (linear interpolation)*
- 3) *Triangulated Cube configuration generation*
- 4) *Cube Index*
- 5) *Combination*

3D Representation (Planar Contour)

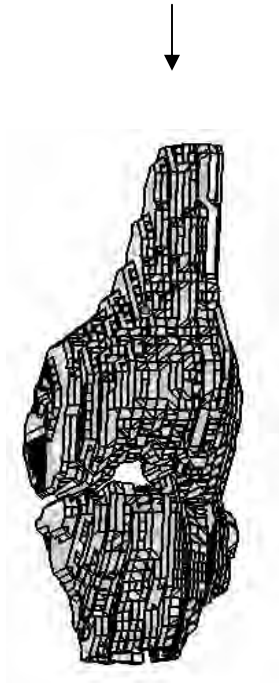
- 1) *Capturing section contour points*
- 2) *Creating B-spline curve*
- 3) *producing solid model construction*

Implementation

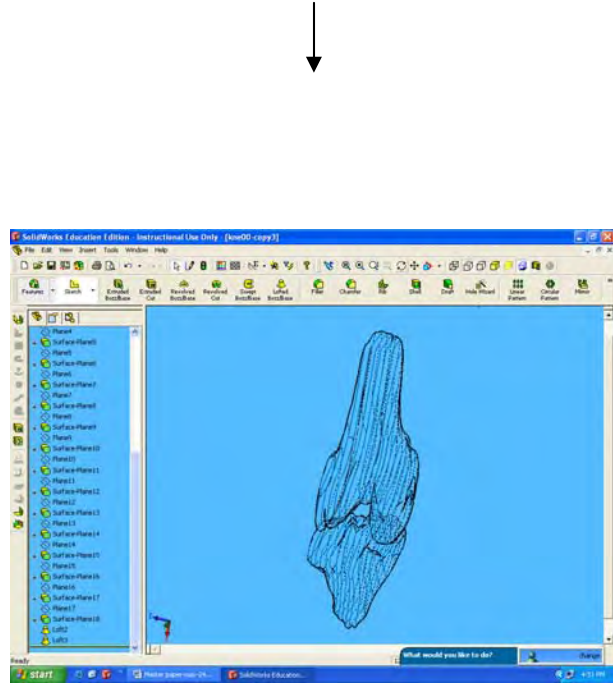
- 1) *Database- File List, Feature Library, Library Feature Type List, Part list*
- 2) *Create the Macro File*
- 3) *Read the data from the digital binary image matrix txt-file for data input*
- 4) *Calculate Cube Index for recognizing which kind of triangulated cube configurations joins the 3D model*
- 5) *Extract the triangulated cube configurations selected for 3D reconstruction*

MRP Manufacturing

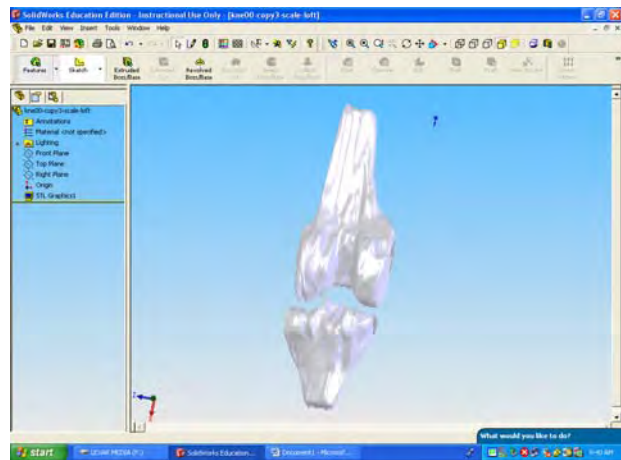
- 1) *Output 3D STL file by SolidWorks*
- 2) *Rapid Prototyping manufacturing*



3D Model by Marching Cubes



3D Model by Planar Contour



3D STL File for Rapid Prototyping

Fig. 1.10 the Produce and Main Milestones of the Thesis

Chapter 2 CT and MRI Digital Image Processing

Since the amount of data is too large to be understood in its raw form, it is essential for the algorithm to use image-processing techniques to filter the original data for 3D reconstruction. In addition, because scan image data that are image model cannot be read directly in SolidWorks, the image data must be processed as the digital binary matrix for 3D modeling. This chapter discusses CT and MRI image and performs image processing from CT and MRI image to digital binary matrixes.

2.1 CT and MRI Image

In medical imaging, the two most common systems used in acquiring detailed anatomical information are Computer Tomography (CT) and Magnetic Resonance Imaging (MRI). The key feature of the imaging technologies is their ability to provide detailed information about the anatomical structure and abnormalities.

CT uses a number of thin, rotation X-ray beams and computer technology to slice 2D images or slice planes to create detailed cross-sectional images of objects. It is fast, patient friendly, and has the unique ability to image a combination of bone and soft tissue. On the other hand, MRI images are obtained by varying the number and sequence of pulsed radio frequency field in order to take advantage of magnetic relaxation properties of hard and soft tissues. Specifically a strong magnetic field is generated to cause atoms inside the body to become aligned. After alignment, a radio wave is issued to “excite” the atoms. Once the radio signal is turned off, the atoms give off a small characteristic signal. Those signals are then measured with a sensitive antenna called an MRI coil. This process is repeated many times until enough measurements are detected to create a series of detailed images. MRI does not use any ionizing radiation, and can create images of almost any body part oriented in any direction. Figure 2.1 shows a MRI example of a

head.

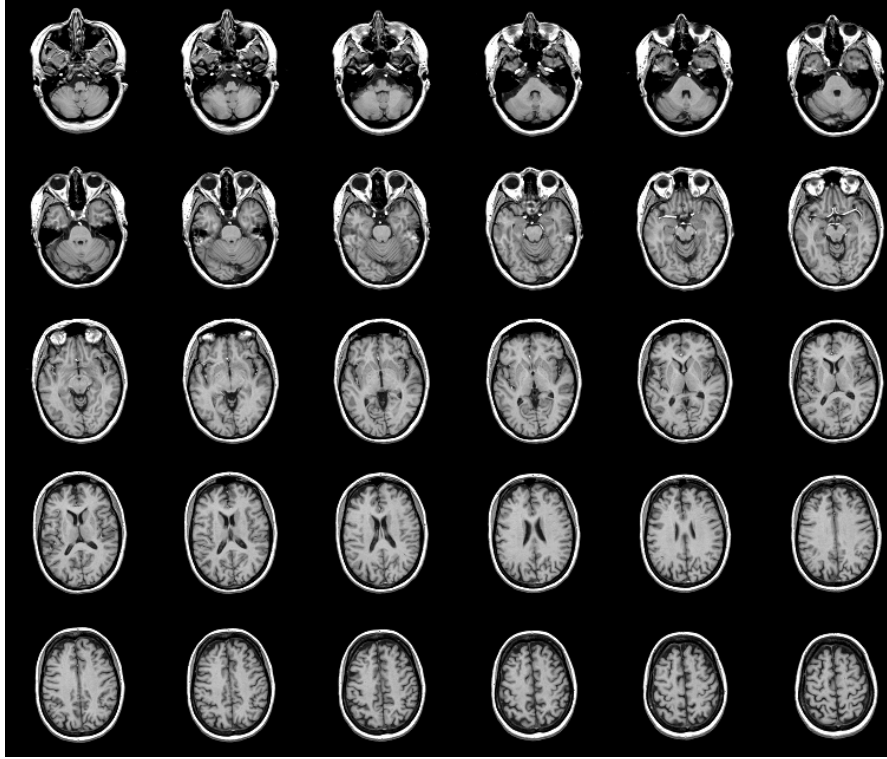


Fig. 2.1 Medical Image of Skull

CTs and MRIs differ in at least two key aspects: (1) CT data are most suitable for modeling bone structures. MRI data are best suited for modeling of soft tissues. (2) CTs sequentially records 20 slices within the measurement volume. MRIs measure the density of a specific nucleus and is volumetric (i. e. interrogation of the entire body within the measurement volume is done simultaneously).

CT and MRI represent the finest resolution capability available in diagnostic systems achieving volumetric resolutions. The information from each plane can be put together to provide a volumetric image of the structure as well as the size and location of anatomical structures [42, 43]. In order to minimize the patient's exposure to radiation, in most real data sets, the distance between two layers is greater. Typical resolutions of the 2D-slices are 256×256 , 512×512 or 1024×1024 . Each pixel possesses an information depth of eight-bits.

In this thesis, CT-Scan data consist of axial scan images of the entire anatomical organs taken at 1 mm intervals at a resolution of 512 x 512 pixels, in which each pixel is made up of 8-bits, 12-bits, or 16-bits of 256gray scale.

2.2 Image Processing

The slice data of original medical imaging are structured point data for which the topology and geometry of the data are implicitly known, and only require dimensions, an origin, and aspect ratio. An image may be defined as a 2D function, $f(x, y)$, where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates (x_0, y_0) is called the *gray level* (ℓ) of the image at that point. That is $\ell = f(x_0, y_0)$, ℓ lies in the range $L_{\min} \leq \ell \leq L_{\max}$. The interval $[L_{\min}, L_{\max}]$ is called the *gray scale*. Common practice is to shift this interval numerically to the interval $[0, L-1]$, where $\ell = 0$ is considered black and $\ell = L-1$ is considered white on the gray scale. All intermediate values are shades of gray varying from black to white. When x, y , and the amplitude values of f are all finite and discrete quantities, the image is called a *digital image*. A digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as *pixels*.

Spatial domain processes will be denoted by the expression $g(x, y) = T[f(x, y)]$, where $f(x, y)$ is the input image, $g(x, y)$ is the processed image. T is an operator on f , defined over some neighborhood of (x, y) . When the neighborhood is size 1×1 (that is a single pixel), $g(x, y)$ depends on the value of $f(x, y)$, and T becomes a *gray-level transformation function* of the form $s = T(r)$. The effect of this transformation would be to produce an image of higher contrast than the original by darkening the levels below

m and brightening the levels above m in the original image. In the limiting case, $T(r)$ produces a two-level (binary) image. A mapping of this form is called a *thresholding* function (see Figure 2.2).

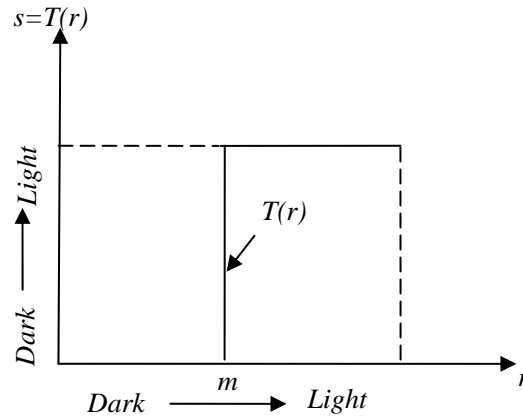


Fig. 2.2 Gray-level Thresholding functions

A pixel p at coordinates (x, y) has four *horizontal* and *vertical* neighbors whose coordinates are given by $(x + 1, y)$, $(x - 1, y)$, $(x, y + 1)$, $(x, y - 1)$. This set of pixels, called the *4-neighbors* of p , is denoted by $N_4(p)$. Each pixel is a unit distance from (x, y) , and some of the neighbors of p lie outside the digital image if (x, y) is on the border of the image. The four diagonal neighbors of p have coordinates $(x + 1, y + 1)$, $(x + 1, y - 1)$, $(x - 1, y + 1)$, $(x - 1, y - 1)$ and are denoted by $N_d(p)$. These points, together with the 4-neighbors, are called 8-neighbors of p , denoted by $N_8(p)$.

To establish if two pixels are connected, it must be determined if they are neighbors and if their gray levels satisfy a specified criterion of similarity. In a binary image with values 0 and 1, two pixels may be 4-neighbors, but they are said to be connected only if they have the same value. Let V be the set of gray-level values used to define adjacency. In a binary image, $V = \{1\}$ if we are referring to adjacency of pixels with value 1. We

consider three types of adjacency: (a) 4-adjacency. Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$. (b) 8-adjacency. Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$. (c) m-adjacency (mixed adjacency). Two pixels p and q with values from V are m-adjacent if (1) q is in the set $N_4(p)$, or (2) q is in the set $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V [44, 45] (see Figure 2.3).

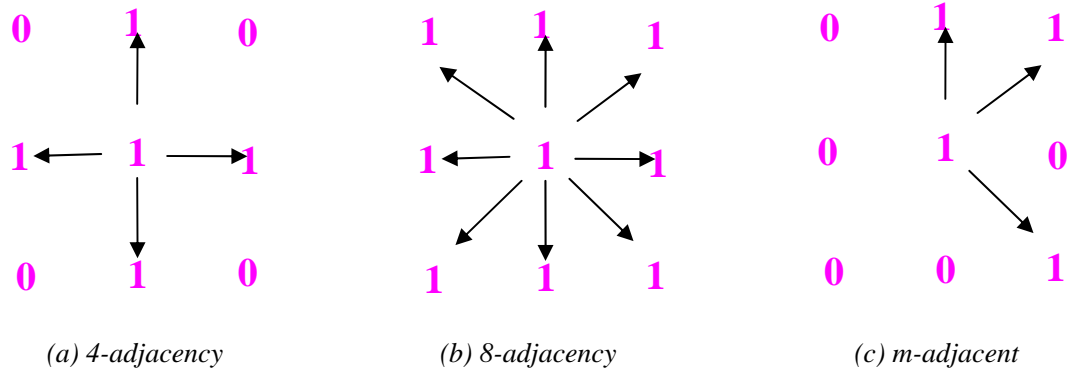


Fig. 2.3 Arrangement of pixels

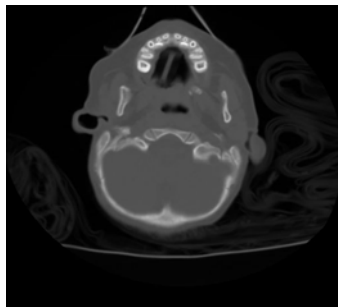
Because SolidWorks cannot read Scan image data directly, the image data must be processed as the digital binary matrix. In this research, Scan data are stored as each slice per file. Firstly, Scan image data is converted into *scan.jpg* file format by using XnViewer, software that both handles images and reduces the image storage and handling size. The *scan.jpg* file, then, is inputted into Matlab. In Matlab the Scan image is processed into **digital binary matrixes** by identifying the image's pixel color numbers, given the color map, picking exact locations of the images, and picking exact points corresponding to the component material. The detailed steps are as the follows:

1. Obtain the patient scan data. The images are captured and scanned. The model will be more accurate if the images are taken directly from the CT or MRI machine as data file in computer.

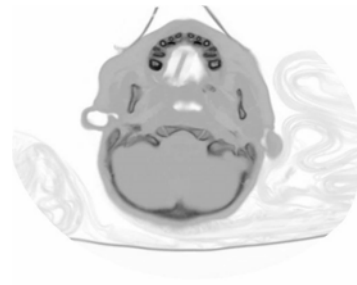
2. Convert the image format. If the images are not in *jpg* format, use XnViewer to

convert them.

3. *Invert the colors.* The original image shown in Figure 2.4 (a) is with black background and the scanned image is represented in white and gray tones. If the colors are inverted in Microsoft Paint, the image is clearer because white and gray tones are changed into black. For more specific properties of the image, it is better to leave the image with its original colors and making a very specific image processing. To improve the process, the inverted image (see Figure 2.4 (b)) is used.



(a) Original Image



(b) Inverted Image

Fig. 2.4 CT Scan Image of a Head

4. *Open a numeric matrix of the image in Matlab.* To do this, the *jpg* image is imported into Matlab. Matlab will import the *jpg* image as a matrix with the same dimensions as the image's number of pixels; Matlab identifies each pixel by its corresponding number in the given color map. The Figure 2.4 (b) shows the inverted image imported by Matlab with more specific details than the original one.

5. *Clean the image from the unwanted parts.* A program has been developed in Matlab to scan the matrix pixel by pixel. A gray-scale map is assigned in Matlab, in which 0 is black that is wanted colors and 255 is white. The image colors that are smaller than 0.005 are captured, the other are deleted. The cleaned image is created (see Figure 2.5).

6. *Convert image into digital binary matrix.* The objective of this step is to make a “boss” that will tell SolidWorks where to draw, and where not to draw. This program

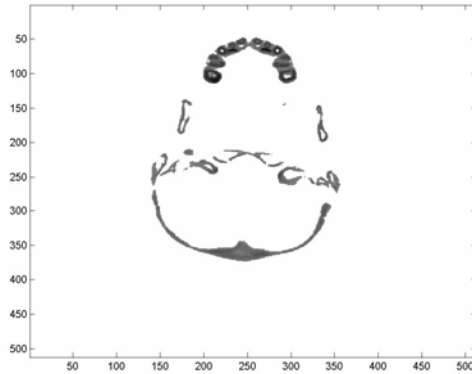


Fig. 2.5 Inverted Cleaned Image

converts every pixel that is not in part of the solid model to 0. In contrast, all remaining pixels are created and identified as 1. Thus, SolidWorks sketches a “keypoint” when it finds a value of 1, and nothing when it finds a value of 0. The resulting matrix is saved as *digital binary matrix file* (See Figure 2.6).

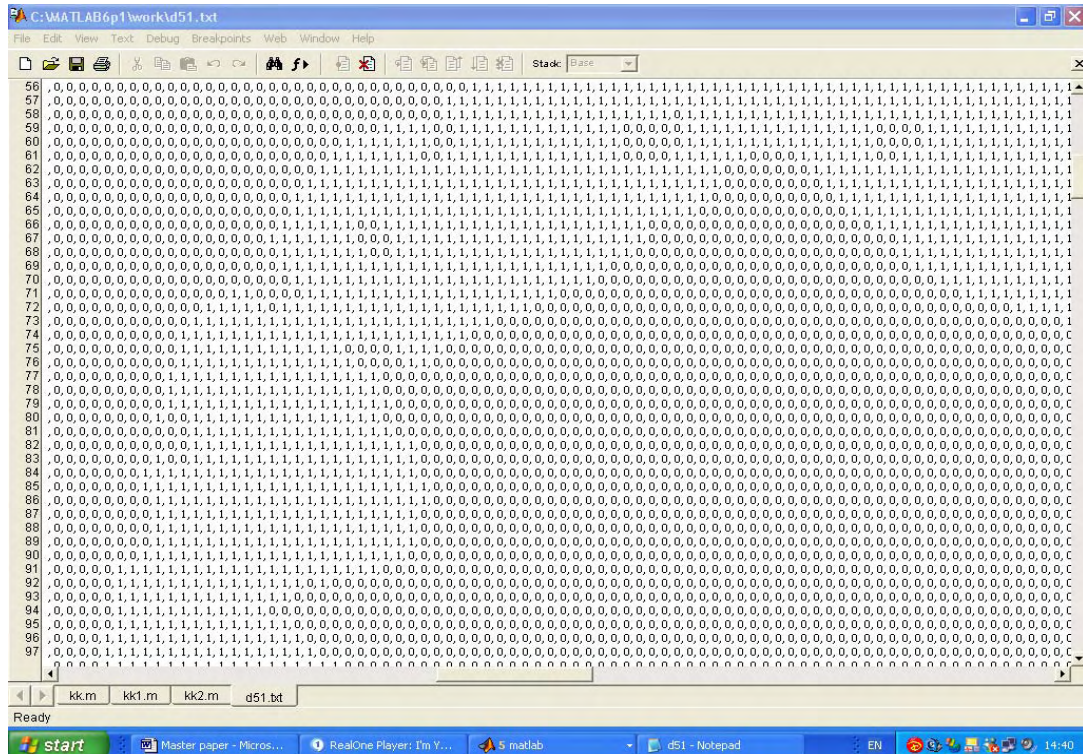


Fig. 2.6 Digital Binary Matrix File

A digital binary matrix is defined by the function $f(V)$, also called the *characteristic*

function, whose domain is the set of all voxels V and whose range is the set $\{0, 1\}$. The set of voxels $S = \{V \mid f(V) = 1\}$ is referred to as the *object* and the set $\underline{S} = \{V \mid f(V) = 0\}$ is referred to as the *background*. In applications such as space planning, the *characteristic function* f is specified by a 3D binary array with value 1 representing *full* and value 0 representing *void*. If Q is a point set in 3D Euclidean space, then $f(V) = 1$ if the points of V have a nonempty intersection with the points of Q , and $f(V) = 0$ otherwise [44, 45].

If *characteristic function* represents the value of the image at V , and D is the range of $f(V)$, then the thresholding operation is defined by the *characteristic function*

$$f(V) = \begin{cases} 1, & f(V) \in D' \subseteq D, \\ 0, & \text{otherwise} \end{cases} \quad (2.1)$$

Thresholding is effective when there is high contrast between the object and background values, and little clutter. **The digital binary matrixes**, finally, are saved in Matlab as *scan.txt* file so that the SolidWorks developed in Visual Basic language reads the input data.

Chapter 3 Marching Cubes Algorithm

Transferring the *Digital Binary Matrixes* to the 3D model will require the application of a 3D image-handling algorithm. In the comparison of the known approaches, the *Marching Cubes algorithm* is chosen because its triangle model it produces can be displayed on the SolidWorks graphics systems by using standard rendering algorithms. In addition, data information from *the digital binary matrixes* can be used to derive inter-slice connectivity. The algorithm can use the table of the edge intersections to describe how a surface is cut through each cube, and the Library Feature to display how the 3D model is created. These procedures are achieved with ease, convenience and rapidity with SolidWorks.

3.1 Marching Cubes Algorithm

Marching Cubes algorithm is to subdivide space into a series of small cubes created from eight pixels and four each from two adjacent slices, to march through each of the cubes, to test the corner points, and to replace the cube with appropriate set of polygons [26]. Major components of this algorithm are deciding how to define the edge configuration in 2D and triangulated cube configuration in 3D. There are some basic conceptions need to be defined before determining them.

Cube (Voxel) --- the volume defined by eight neighboring vertexes

Vertex --- the pixel values at the eight corner points of the cube

Isosurface --- all points within the cube with equal property

Face --- one of the six sides of a cube

Edge--- one of the four rims of a face

(See Figure 3.1)

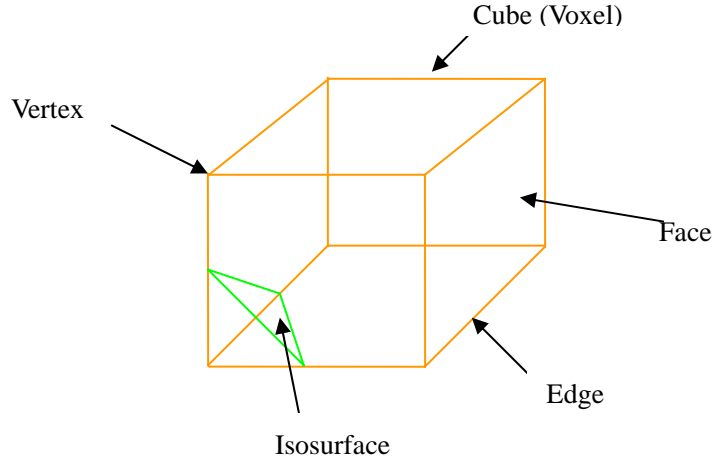


Fig. 3.1 Definitions in Marching Cubes Algorithm

Each vertex is classified as either being inside or outside the isosurface. **0** in *the digital binary matrixes* indicates that the vertex values are outside the isosurface; **1** in *the digital binary matrixes*, by contrast, indicates for the vertex values are inside the isosurface (See Figure 3.2).

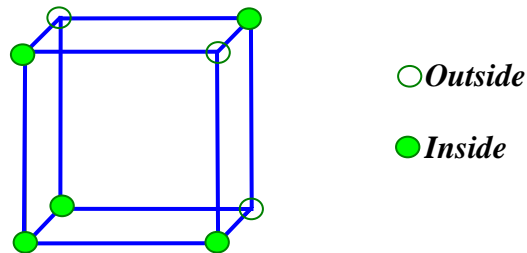


Fig. 3.2 Vertex Classifications

Vertex Index of eight (0-7) vertices and **Edge Index** of twelve (0-11) edges of each cube are indexed as Figure 3.3.

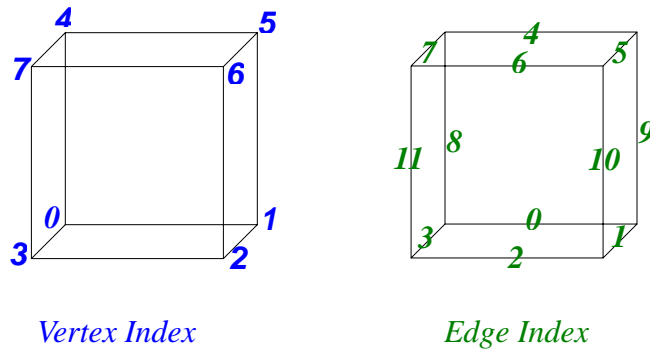


Fig. 3.3 Vertex Index and Edge Index

Whether a vertex value is inside or outside the isosurface is determined by the 8-bit **Cube Index** (See Figure 3.4). If only the 0 bit value of eight-bit number is 1, which means only vertex 0 is inside the isosurface, then the cube index equals $2^0 = 1$ (00000001). Similarly, if only 1 bit value of eight-bit number is 1, which means only vertex 1 is inside the isosurface, and then the cube index equals $2^1 = 2$ (00000010). The largest number of possible combination is 256 because there are $2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 = 256$.

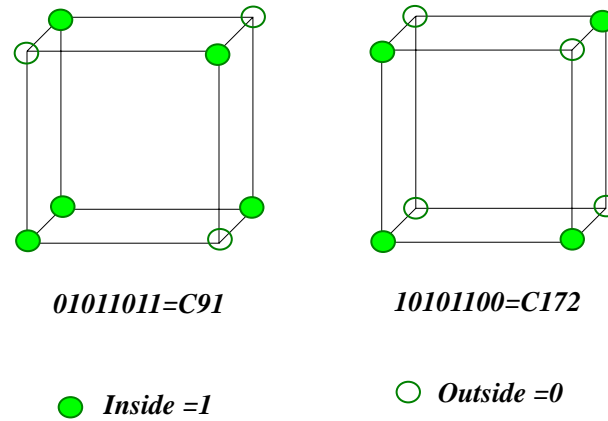


Fig. 3.4 Cube Index

The algorithm determines how the Edge Configuration and the Triangulated Cube Configuration are generated, and then moves to the next cube until the whole object is marched through to create the 3D model.

3.2 Edge Configuration Generation

There are six faces in each cube. Using the follow criteria decides where the isocontour intersects each face. With four vertices and two states in each face, inside and outside each isosurface, 16 cases of edge configuration ($2^4 = 16$) occur when the isocontour intersects the face. The binary “1” indicates insider isocontour and “0” shows

outside the isocontour. 16 cases can be reduced to 6 cases by the rotation and mirror of each face (See Figure 3.5).

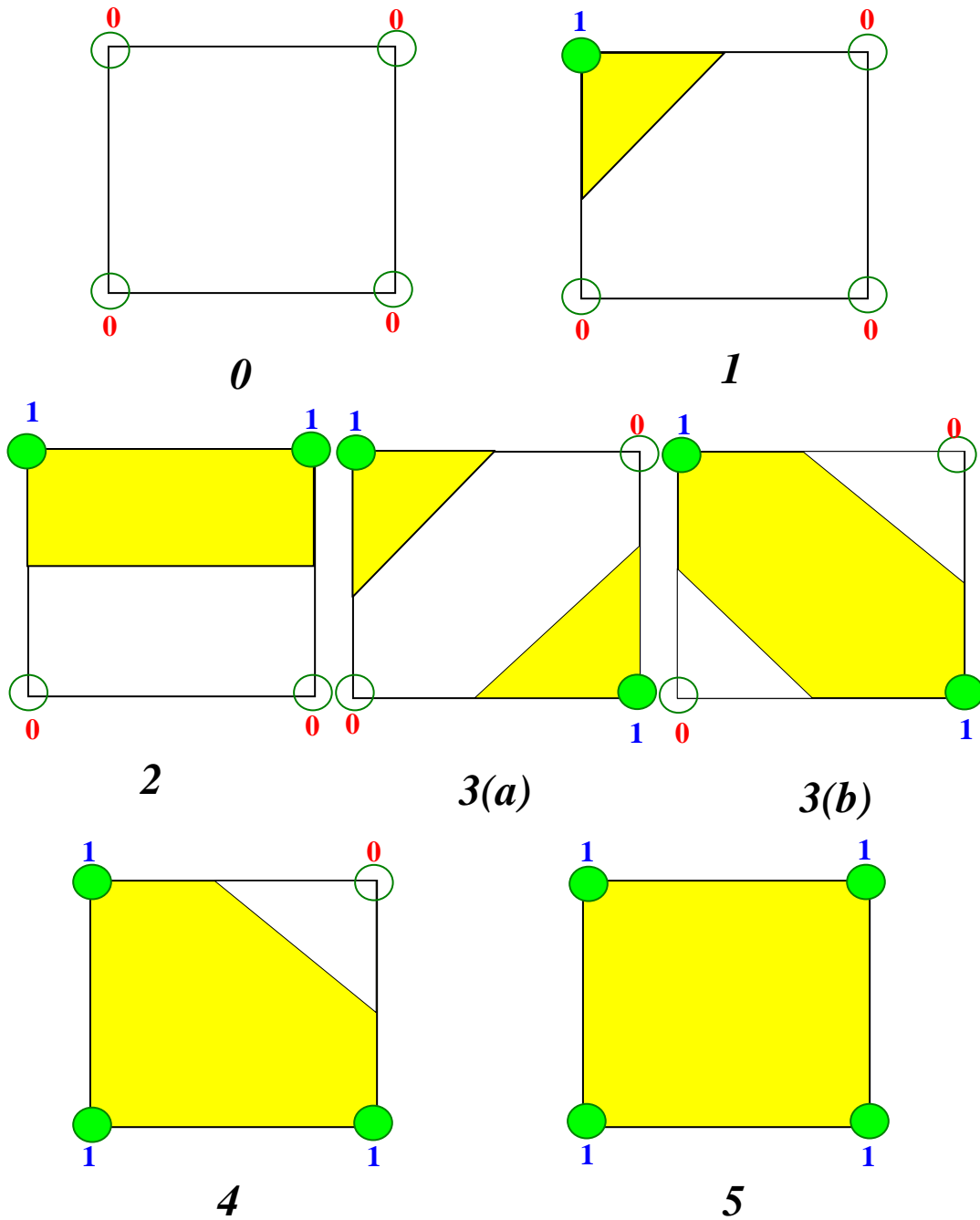


Fig. 3.5 Edge Configuration Generation

When no intersections occur, the edge configuration that no edge is cut generates (configuration 0 and 5).

When one intersection occurs, the edge configuration is produced where two edges are cut (configuration 1, 2, 4). The yellow shaded part is the inside isocontour.

When two intersections occur (See configuration 3 (a) and (b)), the edge configuration in which four edges are cut appears and the face ambiguity arises, which will be expounded in section 3.3.

The fractional distance from the vertex is computed using **linear interpolation**. If a function is defined at integer parameter values, one can define it for intermediate parameter values using linear interpolation, (i.e. connecting the values at integers with straight line segments) [46]. Figure 3.6 shows the interpolation in the 1D case between i^{th}

and $(i+1)^{th}$ point. The line segments of the line between i^{th} and $(i+1)^{th}$ point can be expressed by linear interpolation as $\frac{u-i}{f(u)-f(i)} = \frac{(i+1)-i}{f(i+1)-f(i)}$,

then $f(u) = f(i)(1-(u-i)) + f(i+1)(u-i)$, for $i \leq u < i+1$. By summing these equations for all i the equation of the complete interpolating linear function is:

$$f(u) = f(i)(1-(u-i)) + f(i+1)(u-i) \quad (3.1)$$

Where $f(i)$ only influences the intervals $[i-1, i]$ and $[i, i+1]$. The equation is written as:

$$f(u) = \sum_i f(i)B(u-i) \quad (3.2)$$

where $B_i(u) = 1-|u|$, for $-1 < u < 1$, and zero otherwise. The functions $B_i(u) = B(u-i)$ can be regarded as the basis functions; the complete interpolation is the sum of basis functions weighted by function values at integers.

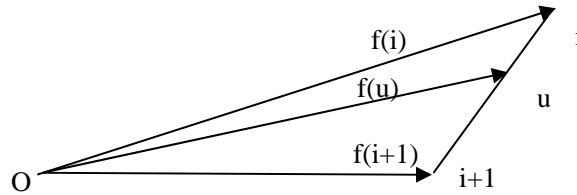


Fig. 3.6 Interpolation in 1D Case between i^{th} and $(i+1)^{th}$ Point

In 2D case, the values are defined at points (i, j) in the plane with integer coordinates. The simplest way to extend linear interpolation to 2D is to use $B(u - i)B(v - j)$ as the bases functions:

$$f(u, v) = \sum_{i, j} f(i, j)B(u - i)B(v - j) \quad (3.3)$$

This is also known as bilinear interpolation [47].

3.3 Triangulated Cube Configuration Generation

After creating Edge Configuration, Triangulated Cube Configuration is generated by utilizing trilinear interpolation, where the isosurface can intersect the cube according to the above criteria of edge configuration generation. Similarly, 3D is the product of three 1D basis functions:

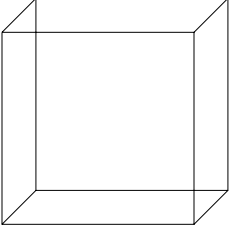
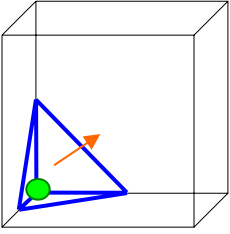
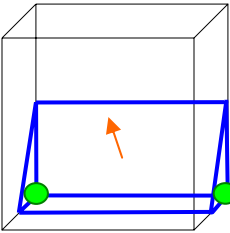
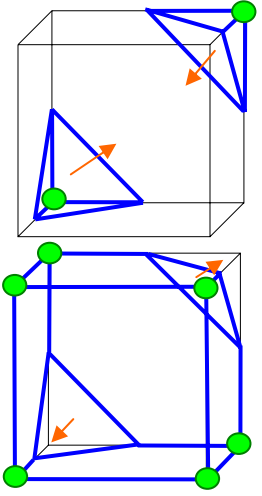
$$f(u, v, w) = \sum_{i, j, k} f(i, j, k)B(u - i)B(v - j)B(w - k) \quad (3.4)$$

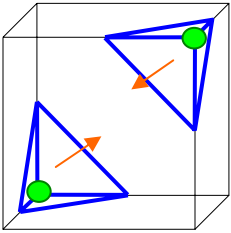
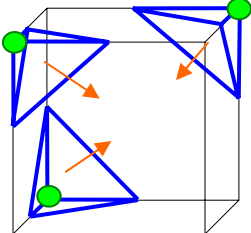
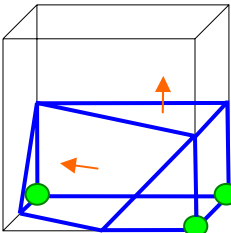
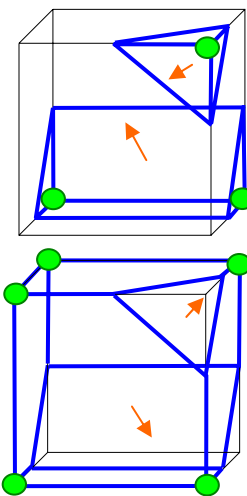
Usually $B_{i, j, k}(u, v, w)$ replaces $B(u - i)B(v - j)B(w - k)$.

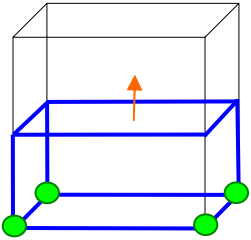
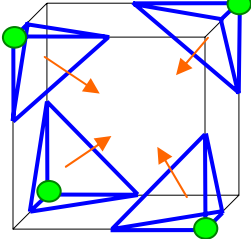
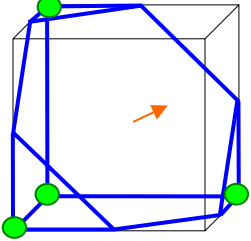
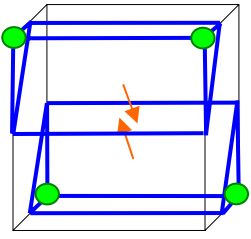
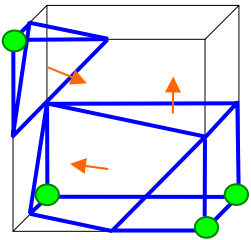
It is important to note that the level set of $f(u, v, w)$ is not a piece-wise linear surface: the equation $f(u, v, w) = 0$ is not linear (it includes terms uv, uw, vw and uvw). Marching Cubes algorithm can compute a piece-wise linear approximation to this set of the trilinear interpolation. The resulting meshes are usually of somewhat better quality and have lower triangle count [47].

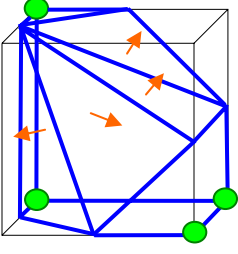
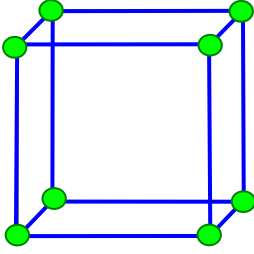
Because there are eight vertices and two states, (inside and outside isosurface), in each cube there are only $2^8 = 256$ ways of triangulated cube configuration that the isosurface can intersect the cube, according to the above criteria of the edge configuration generation. To simplify the algorithm, these 256 cases can be reduced to 15 patterns by rotation, mirroring, and inversion. Table 3.1 shows the triangulation for the 15 patterns.

Table 3.1 Triangulated Cube Configuration

Triangulated Cube Configuration	Pattern	Cube Index (Number)
	0	C0 (1)
	1	Rotation: C1, C2, C4, C8, C16, C32, C64, C128 (8) Inverse: C127, C223, C239, C191, C247, C251, C253, C254 (8)
	2	Rotation: C12, C9, C3, C6, C192, C144, C48, C96, C17, C34, C68, C136 (12) Inverse: C63, C111, C119, C187, C159, C207, C221, C238, C243, C246, C249, C252 (12)
	3(a) 3(b)	Rotation: C72, C36, C18, C129, C132, C66, C33, C24, C5, C10, C80, C160 (12) Rotation: C95, C126, C175, C183, C189, C219, C222, C231, C123, C237, C245, C250 (12)

	4	<p>Rotation: $C_{40}, C_{65}, C_{130}, C_{20}$ (4)</p> <p>Inverse: $C_{190}, C_{215}, C_{235}, C_{125}$ (4)</p>
	5	<p>Rotation: $C_{164}, C_{88}, C_{161}, C_{82}, C_{26}, C_{37}, C_{74}, C_{133}$ (8)</p> <p>Inverse: $C_{173}, C_{181}, C_{218}, C_{229}, C_{91}, C_{94}, C_{122}, C_{167}$ (8)</p>
	6	<p>Rotation: $C_7, C_{11}, C_{14}, C_{13}, C_{112}, C_{176}, C_{224}, C_{208}, C_{98}, C_{196}, C_{152}, C_{49}, C_{19}, C_{25}, C_{35}, C_{38}, C_{50}, C_{70}, C_{76}, C_{100}, C_{137}, C_{140}, C_{145}, C_{200}$ (24)</p> <p>Inverse: $C_{31}, C_{47}, C_{55}, C_{59}, C_{79}, C_{103}, C_{115}, C_{110}, C_{118}, C_{143}, C_{155}, C_{157}, C_{179}, C_{185}, C_{205}, C_{206}, C_{217}, C_{220}, C_{230}, C_{236}, C_{241}, C_{242}, C_{244}, C_{248}$ (24)</p>
	7(a) 7(b)	<p>Rotation: $C_{44}, C_{73}, C_{131}, C_{22}, C_{194}, C_{148}, C_{56}, C_{97}, C_{81}, C_{162}, C_{84}, C_{168}, C_{52}, C_{67}, C_{69}, C_{104}, C_{134}, C_{138}, C_{146}, C_{193}$ (24)</p> <p>Rotation: $C_{61}, C_{62}, C_{87}, C_{93}, C_{107}, C_{109}, C_{117}, C_{121}, C_{124}, C_{151}, C_{158}, C_{171}, C_{174}, C_{182}, C_{186}, C_{188}, C_{199}, C_{203}, C_{211}, C_{233}, C_{234}, C_{213}, C_{214}, C_{227}$ (24)</p>

	8	<p>Rotation: C_{15}, C_{102}, C_{51} (3)</p> <p>Inverse: $C_{153}, C_{204}, C_{240}$ (3)</p>
	9	<p>Rotation: C_{90}, C_{165} (2)</p>
	10	<p>Rotation: $C_{27}, C_{39}, C_{78}, C_{141}$ (4)</p> <p>Inverse: $C_{114}, C_{177}, C_{216}, C_{228}$ (4)</p>
	11	<p>Rotation: C_{170}, C_{60}, C_{105} (3)</p> <p>Inverse: C_{85}, C_{195}, C_{150} (3)</p>
	12	<p>Rotation/Inverse: $C_{135}, C_{75}, C_{30}, C_{45}, C_{120}, C_{180}, C_{225}, C_{210}, C_{53}, C_{58}, C_{83}, C_{86}, C_{89}, 92, C_{101}, C_{106}, C_{149}, C_{154}, C_{163}, C_{166}, C_{169}, C_{172}, C_{197}, C_{202}$ (24)</p>

	13	<p>Rotation: C23, C46, C29, C54, C57, C71, C77, C99, C108, C116, C113, C43 (12)</p> <p>Inverse: CC232, C209, C226, C201, C198, C184, C178, C156, C147, C142, C139, C212 (12)</p>
	14	<p style="text-align: center;">C255 (1)</p>

The **arrow** denotes the surface normal of the relevant triangles and points to the outside triangulated cube configuration. The green vertex means it is inside the isosurface and its value is **1**. The simple **Pattern 0** occurs if all vertex values are outside the isosurface and produces no triangulated cube configuration. By contrast, **Pattern 15** occurs if all vertex values are inside the selected object and produce a cube. The **Pattern 1** occurs if the surface separates one vertex from the other seven, resulting in one of the triangulated cube configurations defined by the three edge intersections. Other patterns produce multiple triangulated cube configurations. Permutation of these 15 basic patterns, using complementary and rotational symmetry, produces the 256 triangulated cube configurations.

In some cases, there is more than one topologically distinct way to construct triangulated configurations from the points on edges. In 2D, when two intersections occur, the edge configuration appears in which four edges are cut and the face 2D ambiguity arises (See Figure 3.7: configuration (a) and (b) in 2D). This problem is resolved by deciding between different cases, and depending on whether the center of the cube in 3D is inside or outside the triangulated configuration cube determines configuration (a) or (b).

If the center of the cube is outside the triangulated configuration cube, the face's center in 2D is not connected to the inside isocontour (See Figure 3.7: configuration (a)), which produces the triangulated configuration cube (a) in 3D. By contrast, the face's center in 2D is connected inside it (See Figure 3.7 configuration (b)), which causes the triangulated configuration cube (b) in 3D [47].

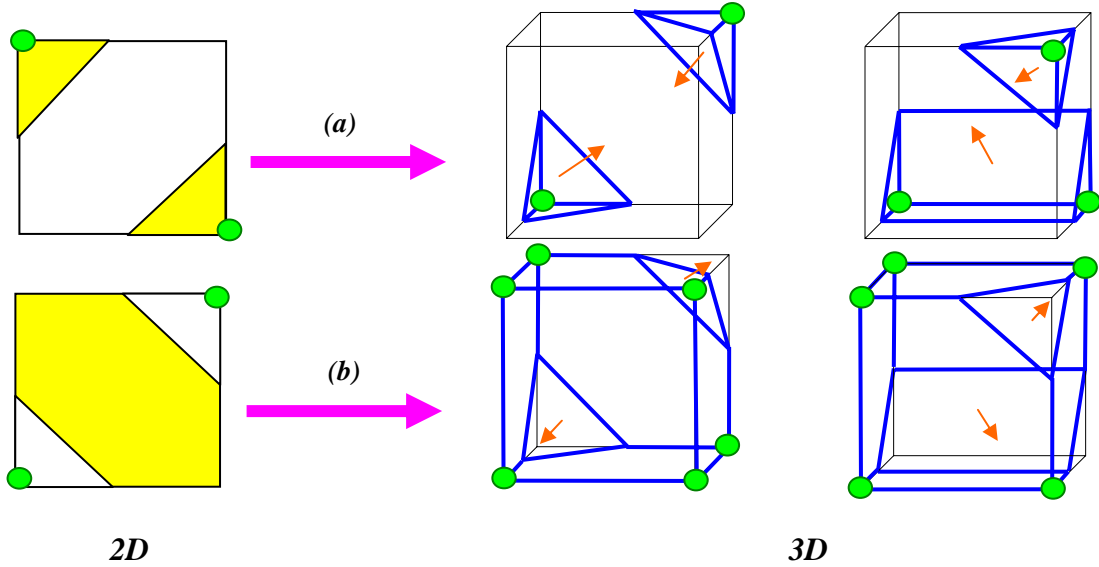


Fig. 3.7 2D and 3D Ambiguity

The transformation progress for 3D reconstruction requires **Edge configuration generation** and **Triangulated Cube configuration generation**. Edge configuration generation is based on the isosurface and cube intersections. Triangulated cube configuration generation is based on Edge configuration. The **Cube Index** values represent 256 cases of Triangulated cube configuration combinations. By connecting triangulated cubes from all cubes, a 3D model is represented.

Chapter 4 Feature Library and Database Library

3D model is reconstructed in SolidWorks Application Programming Interface (API) and the programs are developed by *Visual Basic*. In 3D reconstruction, the first technical problem to be solved is what kinds of database are stored in memory. For research, there are two types of library: **Feature Library** and **Database** for data exchange. Building feature library is to speed up process rates of 3D models, where 170 library features are created in SolidWorks depending on the different types. Database creation is to organize the program reasonably for highly running rates, in which there are three types such as List of Files, List of Library Features, and List of Parts.

4.1 Feature Library

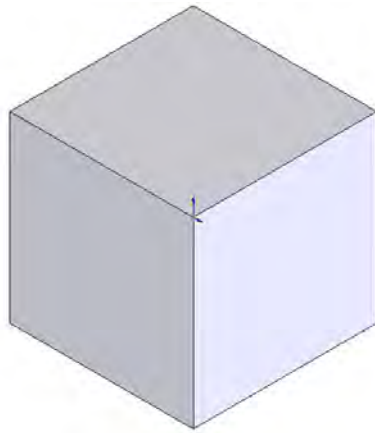
Feature elements are the smallest elements that are used to reconstruct 3D models. Feature Library is in charge of the 256 triangulated cube configurations that are saved in Feature Library of SolidWorks. They are matched by **Cube Index** to join the 3D reconstruction.

4.1.1 Building Feature Library

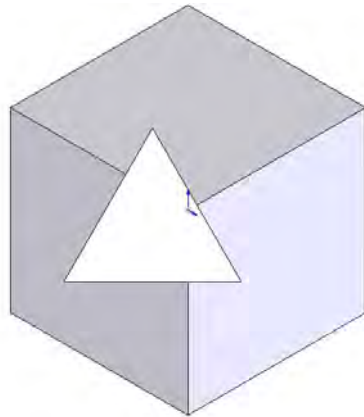
To create a feature library, a *base feature*, which is either the first solid feature, is first created. The triangulated cube configuration features included in the library feature on the base are then produced separately. Feature Library file has the *.*sldlfp* extension. Building feature library includes following three steps:

- Open a new part, sketch a profile, and create a base feature.
- Create the features of 256 cases in the feature library separately (See Appendix B).
- From the *Save as type* list, select *Lib Feat Part (*.sldlfp)*. Enter a name and save it.

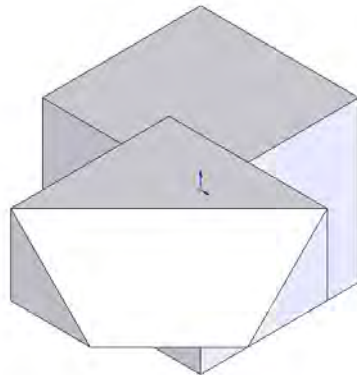
Figure 4.1 show the Base Feature, Library Feature of *C1*, and *C25*



Base Feature



*Base Feature and
Library Feature of C1*



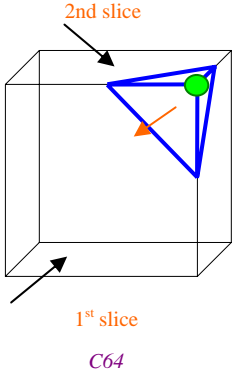
*Base Feature and
Library Feature of C25*

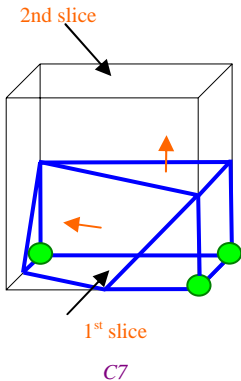
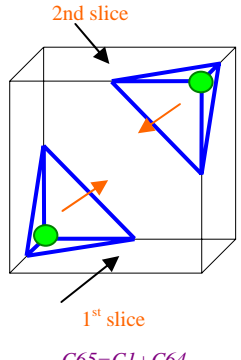
Figure 4.1 Establishment of Base Feature and Feature Library

4.1.2 Type of Feature Library

Because SolidWorks software requires that the library feature must touch the Base Feature when creating the feature library, the triangulated cube configurations are classified as three types signed as “-7”, “-66”, and “-8” and the numbers are used to indicate types of feature library. Those integers represent only marks, which may be arbitrary values. “-7” describes the triangulated cube configurations that cannot attach the base features. During 3D reconstruction, they return to the right position. “-66” expresses the triangulated cube configurations that are individual bodies and touch the base feature directly. The library features are picked up directly without any processing. Type “-8” is the combination of “-7” and “-66”. The library features are combined depending on the triangulated cube configurations in 3D representation (See Table 4.1).

Table 4.1 Feature Library

<i>Type</i>	<i>Triangulated Cube Configurations</i>	<i>Library Feature</i>	<i>Cube Index (Number)</i>
-7			C192, C144, C48, C96, C240, C112, C176, C224, C208, C16, C32, C64, C128 (13)

-66	 <p style="text-align: center;">$C7$</p>		<p><i>All of them except Cube Indexes of -7 and -8</i> (157)</p>		
-8	 <p style="text-align: center;">$C65=C1+C64$</p>		<div><div>$C5(C1+C4)$</div><div>$C20(C4+C16)$</div><div>$C24(C8+C16)$</div><div>$C30(C14+C16)$</div><div>$C37(C5+C32)$</div><div>$C42(C34+C8)$</div><div>$C52(C4+C48)$</div><div>$C58(C50+C8)$</div><div>$C66(C2+C64)$</div><div>$C72(C8+C64)$</div><div>$C75(C11+C64)$</div><div>$C82(C2+C80)$</div><div>$C86(C70+C16)$</div><div>$C90(C10+C80)$</div><div>$C94(C78+C16)$</div><div>$C104(C8+C96)$</div><div>$C120(C8+C112)$</div><div>$C130(C2+C128)$</div><div>$C133(C5+C128)$</div><div>$C138(C136+C2)$</div><div>$C149(C145+C4)$</div><div>$C160(C128+C32)$</div><div>$C163(C35+C128)$</div><div>$C166(C38+C128)$</div><div>$C169(C137+C32)$</div><div>$C180(C4+C176)$</div><div>$C194(C2+C192)$</div><div>$C202(C200+C2)$</div><div>$C225(C1+C224)$</div></div> <div><div>$C10(C8+C2)$</div><div>$C21(C17+C4)$</div><div>$C26(C10+C16)$</div><div>$C33(C1+C32)$</div><div>$C40(C8+C32)$</div><div>$C44(C12+C32)$</div><div>$C53(C49+C4)$</div><div>$C60(C12+C48)$</div><div>$C67(C3+C64)$</div><div>$C73(C9+C64)$</div><div>$C80(C16+C64)$</div><div>$C83(C19+C64)$</div><div>$C88(C8+C80)$</div><div>$C91(C27+C64)$</div><div>$C97(C1+C96)$</div><div>$C105(C9+C96)$</div><div>$C122(C114+C8)$</div><div>$C131(C3+C128)$</div><div>$C134(C6+C128)$</div><div>$C146(C2+C144)$</div><div>$C150(C6+C144)$</div><div>$C161(C1+C160)$</div><div>$C164(C4+C160)$</div><div>$C167(C39+C128)$</div><div>$C172(C140+C32)$</div><div>$C181(C177+C4)$</div><div>$C195(C3+C192)$</div><div>$C210(C2+C208)$</div><div>$C229(C228+C1)$</div></div> <div><div>$C18(C2+C16)$</div><div>$C22(C6+C16)$</div><div>$C28(C12+C16)$</div><div>$C36(C32+C4)$</div><div>$C41(C9+C32)$</div><div>$C45(C13+C32)$</div><div>$C56(C8+C48)$</div><div>$C65(C1+C64)$</div><div>$C69(C68+C1)$</div><div>$C74(C10+C64)$</div><div>$C81(C17+C64)$</div><div>$C84(C68+C16)$</div><div>$C89(C25+C64)$</div><div>$C92(C76+C16)$</div><div>$C101(C100+C1)$</div><div>$C106(C98+C8)$</div><div>$C129(C1+C128)$</div><div>$C132(C4+C128)$</div><div>$C135(C7+C128)$</div><div>$C148(C4+C144)$</div><div>$C154(C152+C2)$</div><div>$C162(C34+C128)$</div><div>$C165(C5+C160)$</div><div>$C168(C136+C32)$</div><div>$C173(C141+C32)$</div><div>$C193(C1+C192)$</div><div>$C197(C196+C1)$</div><div>$C218(C216+C2)$</div><div></div></div> <div>(86)</div>		

4.2 Database

Database is built for organizing the program reasonably in order to have a fast running rate. Database consists of List of Files, List of Library Features, and List of Parts that join the data exchange in programming.

List of Files --- including the CT scan data files. CT scan data are saved slice by slice as “CT*.txt” where the data structure is *the digital binary matrixes*. The file component is as the follow:

File name: *filelist.txt*

```
C:\Documents and Settings\My Documents\Solidapplication\CT1.txt
C:\Documents and Settings\My Documents\Solidapplication\CT2.txt
C:\Documents and Settings\My Documents\Solidapplication\CT3.txt
C:\Documents and Settings\My Documents\Solidapplication\CT4.txt
C:\Documents and Settings\My Documents\Solidapplication\CT5.txt
C:\Documents and Settings\My Documents\Solidapplication\CT6.txt
C:\Documents and Settings\My Documents\Solidapplication\CT7.txt
```

.....

```
C:\Documents and Settings\My Documents\Solidapplication\CT256.txt
```

List of Library Features--- showing an array (13-column, 256-row) where each column represents one type of three types in feature library and each row shows the different cases following the different **Cube Index** values. Integer after **-8** expresses which triangulated cube configuration of “**-7**” and “**-66**” consists of the library feature. **-1** shows that no library feature is used for 3D reconstruction. List of Library Features is shown in follow:

File name: *triLFCtable.txt*

```
-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-8,1,-66,-8,4,-66,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
```

-8,8,-66,-8,2,-66,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-66,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-7,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1

List of Parts --- describing 256 triangulated cube configurations in feature library in order to match the **Cube Index** for extracting the relative library feature. The format is as follows, where “*-cube li.sldlfp” file stands for each Library Feature saved in SolidWorks as *Library Feature* format.

File name: *partlist.txt*

"C:\Documents and Settings\My Documents\Solidapplication\LFC\0-cube li.sldlfp"
"C:\Documents and Settings\My Documents\Solidapplication\LFC\1-cube li.sldlfp"
"C:\Documents and Settings\My Documents\Solidapplication\LFC\2-cube li.sldlfp"
"C:\Documents and Settings\My Documents\Solidapplication\LFC\3-cube li.sldlfp"
"C:\Documents and Settings\My Documents\Solidapplication\LFC\4-cube li.sldlfp"
"C:\Documents and Settings\My Documents\Solidapplication\LFC\5-cube li.sldlfp"
"C:\Documents and Settings\My Documents\Solidapplication\LFC\6-cube li.sldlfp"
.....
"C:\Documents and Settings\My Documents\Solidapplication\LFC\256-cube li.sldlfp"

Chapter 5 3D Reconstruction by Marching Cubes

To evaluate the performance of the Marching Cubes algorithm, programming implementation is developed in SolidWorks by using Visual Basic language. The flow chart of program is shown in the follow section. The program flows mainly include creation of macro file that is main program for activating the link between SolidWorks and Visual Basic, image input that is to transfer CT and MRI image into digital binary matrixes, data capture where digital binary matrixes are read into SolidWorks, and 3D modeling that are created by calculating Cube Index to pick up triangulated cube configuration from feature library.

5.1 Programming Development

5.1.1 Flow Chart

In programming, firstly, a CT image is input and converted into digital binary matrixes. The Library and Cube Index are set up to achieve the database exchange. The Cube Index then matches the data that are extracted from the library to find the edge intersection by interpolation, and Library Feature is inserted into the 3D model as this process continue to build each individual feature. Finally, the 3D model is created. Further, the reconstructed models can be imported into RP software for biomechanical manufacturing. In each step, two CT slices are kept in memory. Each cube is processed through two slices. The following Figure 5.1 shows the flow chart for 3D reconstruction.

5.1.2 Creation of Macro File

The SolidWorks application environment is activated with Visual Basic language by creating macro file. The purpose of creating a Macro file is to activate the link between SolidWorks and Visual Basic automatically and correctly. Creation of Macro File includes the follow steps:

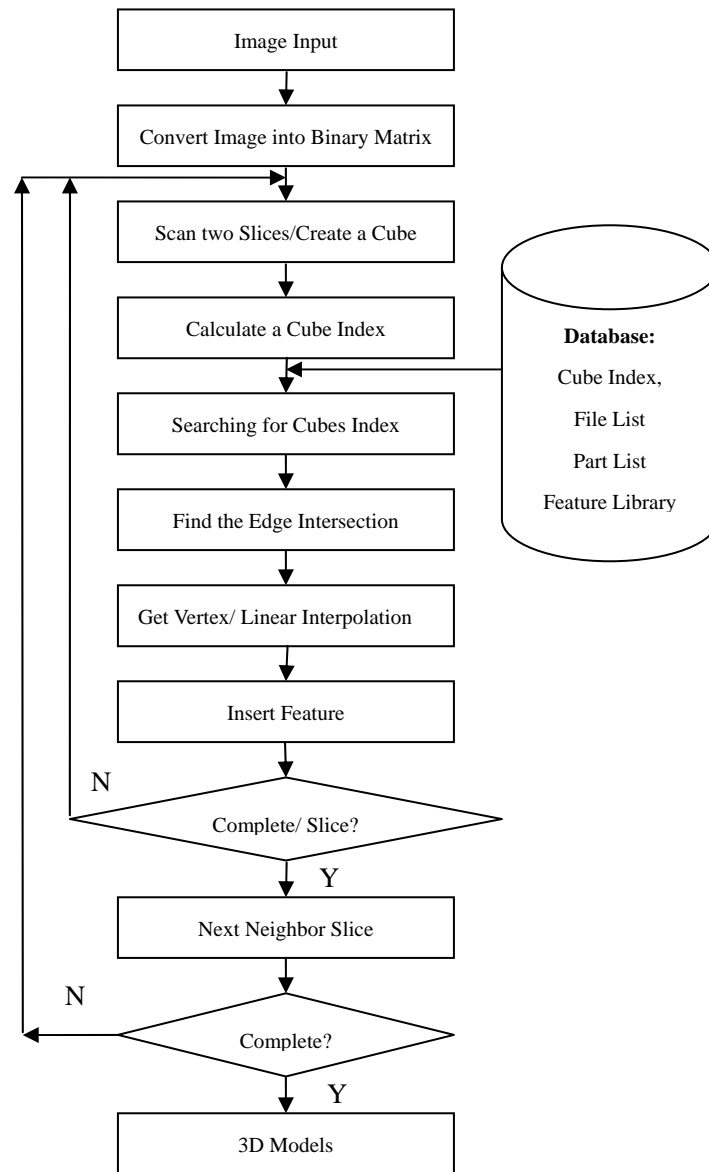


Fig. 5.1 Flow Chart of 3D Reconstruction

- (1) **Open Visual Basic**, click the **New/Project** to create the **Project**, build a new **Form**, and then save the **scan.vbp Project file**, the **scan.frm Form file**, and **scan.exe Active file**
- (2) **Enter SolidWorks environment**, select **Tool/Macro** to record the Project to be created by Visual Basic language, and save **scan.swp Macro file**

(3) Write the **Macro code** in Edit by selecting the **Tool/Marco/Edit**:

File name: *list1.swp*

Sub main()

MyAppID=Shell ("C:\my.....\scan.exe", 1)

AppActivate MyAppID

End Sub

(4) **Build** the **Macro Button** in SolidWorks by opening the **Tool/Custom/Command/Macro** to drag the picture to main menu, then click Ok

5.1.3 Image Transformation

Matlab will import the image as a matrix with the same dimensions as the image's number of pixels and run the following program to test the procedure. The following program removes all that is not bone from the image, and save the cleaned image as *CLA51.jpeg* .

File name: *kk.msn*

figure;

colormap (gray);

image;

pause

for I = 1:512

for j = 1:512

for k = 1:3

A = A51(i,j,k);

B = (double(A))/255;

if (b<.5098)

CLA51(i,j,k) =B;

else

CLA51 (i,j,k) =1;

end

end

end

end

figure;

colormap(gray);

```
image(CLA51);
saveas(gcf, 'CLA51', 'jpeg');
```

The program will then leave only the bones in the image. The resulting matrix is saved as *CL_A51.txt* file.

File name: *kk1.msn*

```
for I=1:512
    for J=1:512
        A=A51(I,J,1);
        B=A51(I,J,2);
        C=A51(I,J,3);
        if(A==B==C)
            CL_A51(I,J)=B;
        elseif((A<B)&(A<C))
            CL_A51(I,J)=A;
        elseif((B<A)&(B<C))
            CL_A51(I,J)=B;
        else
            CL_A51(I,J)=C;
        end
    end
end

for I=1:512
    for J=1:512
        CL=CL_A51(I,J);
        if(CL<128)
            CW(I,J)=1;
        else
            CW(I,J)=0;
        end
    end
end

csvwrite('CL_A51.txt',CW);
```

5.1.4 Reading Database

The aim is to enhance the running speed where the CT-Scan *binary digital image*

matrix database is read into SolidWorks by Visual Basic language to achieve the data input.

To read data from a sequential file, first open the file using:

Open SeqFileName For Input As #N

where N is an integer file number and SeqFileName is a complete file path. The file is closed using:

Close N

The Input statement is used to read in data from a sequential file. The format is:

Input #N, [data list]

The data names in the list are separated by commas. If no data are listed, the current line in the file N is skipped.

Note data must be read in exactly the same manner as they were written. So, using the data A, B, C, and D, the appropriate statements are:

Input #1, A, B, C

Input #1, D

These two lines read the data A, B, and C from the first line in the file and D from the second line. It doesn't matter whether the data was originally written to the file using Write or Print (i.e. commas are ignored) [VB help function].

The programming shows the file data input in SolidWorks developed by Visual Basic

File name: *list1.frm*

filename = "C:\Documents and Settings\jiman\My Documents\Solidapplication\filelist26.txt"

Open filename For Input As #1

Do While Not EOF(1)

For i = 1 To 7

Line Input #1, AA

filenamee(i) = AA

Next i

Loop

Close #1

5.1.5 Determining Cube Index

The goal is to recognize which triangulated cube configurations join the 3D model by calculating the **Cube Index** value for searching/matching the **File List**, **Part list**, and **Library Feature Type List** for 3D reconstruction. Creating cube index number by Visual Basic language is the follow program that has been developed.

File name: *list1.frm*

```
If grid(0)=1 Then Cubeindex0=1
    Else Cubeindex0=0
End If
If grid(1)=1 Then Cubeindex1=2
    Else Cubeindex1=0
End If
.....
If grid(7)=1 Then Cubeindex7=128
    Else Cubeindex7=0
End If
Cubeindex00=0
Cubeindex=Cubeindex0+ Cubeindex1+.....+ Cubeindex7+ Cubeindex00
```

5.1.6 Display of 3D Reconstruction Model

With the **Cube Index** value matching **File List**, **Part list**, and **Library Feature Type List**, the searched triangulated cube configurations is extracted from Feature Library and displayed in SolidWorks for 3D reconstruction. 3D representation undergoes different processes, depending on the different styles of library feature “-7”, “-66”, and “-8”. If the value from **Library Feature Type List** that is matched by **Cube Index** value is -66, the library feature is extracted directly and inserted into the reference plane of **Base Feature** by using *Part.InsertLibraryFeature (partdrawname)* function of *SolidWorks API*. If the value is -7, the library feature is extracted and inserted the reference plane that is located in the middle of two slices. For -8, the image processing experiences the combination of library features, then these are extracted and inserted into the related reference plane.

File name: list1.frm

```

If triLFCTable(cubeindex)(i1) <> -1 And triLFCTable(cubeindex)(i1) = -66 Then
    partdrawname = partname(cubeindex)
    Part.SelectByID newPlaneName, "PLANE", 0, 0, 0
    Part.InsertLibraryFeature (partdrawname)
    .....
If triLFCTable(cubeindex)(i1) <> -1 And triLFCTable(cubeindex)(i1) = -7 Then
    partdrawname = partname(cubeindex)
    Part.SelectByID newPlaneName0, "PLANE", 0, 0, 0
    Part.InsertLibraryFeature (partdrawname)
Else
    .....
If triLFCTable(cubeindex)(i1) <> -1 And triLFCTable(cubeindex)(i1) = -8 Then
    i1 = i1 + 1
    cubeindex1 = triLFCTable(cubeindex)(i1)
    i1 = i1 + 1
    If triLFCTable(cubeindex)(i1) = -66 Then
        .....
    Else
End If
    If triLFCTable(cubeindex)(i1) = -7 Then
        .....
    Else
End If

```

3D reconstruction is displayed in SolidWorks by repeating this process recursively.

Table 5.1 shows file names and functions of program development.

Table5.1 File Names and Functions

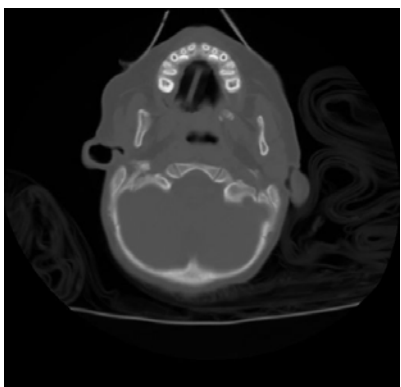
Image Processing			Library		3D Reconstruction	
Cleaning image		<i>kk.msn</i>	List of files	<i>filelist.txt</i>	Macro file	<i>list1.swp</i>
Digital Binary Matrix		<i>kk1.msn</i>	List of parts	<i>partlist.txt</i>	Active file	<i>List1.exe</i>
			List of Feature Library	<i>triLFCTable.txt</i>	Project	<i>list1.vbp</i>
			Feature Library	<i>*-cube li.sldlfp</i>	Form	<i>list1.frm</i>

5.2 Results and Discussions

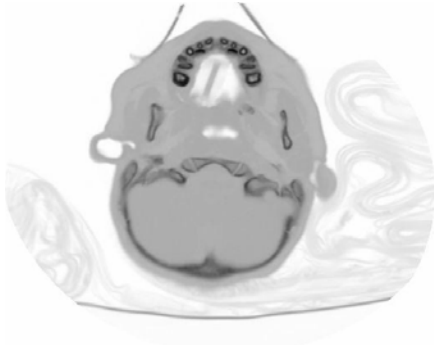
In order to evaluate the performance of the interface software tool in medical

situations, a set of actual clinical data is used. The test sample is a “tooth” dataset generated from CT scan of a human head. The image is grayscale color and contains 512×512 pixels and 18 slices with 1mm interval. Figure 5.2 shows the results of the transition from CT scans to 3D model the tooth. Figure 5.3 shows the result of 3D reconstruction of the knee joint and describe the original CT scan information, such as 18 slices and 512×512 pixels per slice. The processes cover conversion of the CT scan image to the digital binary matrixes (including: inverted image and cleared image), 3D reconstruction from edge configuration generation to triangulated cube configuration generation. All 3D models are visualized on the computer screen in lateral, frontal, oblique lateral and axial views with SolidWorks.

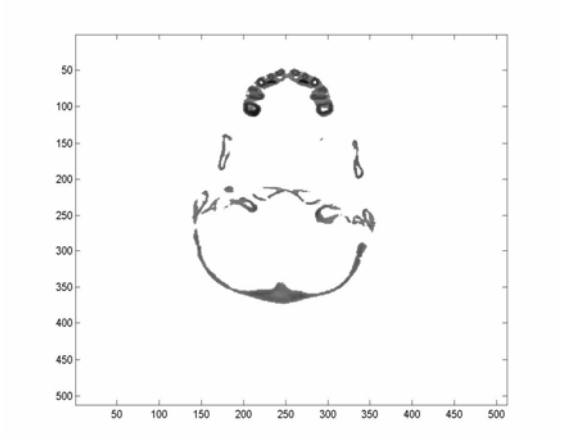
As described previously, the digital binary matrixes can be created from a CT scan image. Obtaining the digital binary matrixes is three-step processes. In the first step, the CT scan image must be converted to a JPG format and inverted in XnView. Then, the inverted image must be converted to an eight-bit gray scale image using a conversion method to compute each eight-bit gray pixel to replace each old pixel. Finally, these eight-bit pixels are transformed into the digital binary matrixes by computing thresholding function in Matlab. The test is conducted to evaluate feasibility of the conversion technique. On the other hand, the results of 3D reconstruction in SolidWorks and Rapid Prototyping from the digital binary matrixes are important to the success of the software. The process undergoes edge configuration generation and triangulated cube configuration generation. In the edge configuration generation, each face is constructed from four corner points and interpolated to produce one or two intersections. There are six unique cases. All of the interpolated lines can be used to define an isosurface representing the locations of the isovalue in the image plane. The triangulated cube configuration is obtained using the Marching Cubes algorithm, which can be used to create a 3D model in SolidWorks.



Original Image

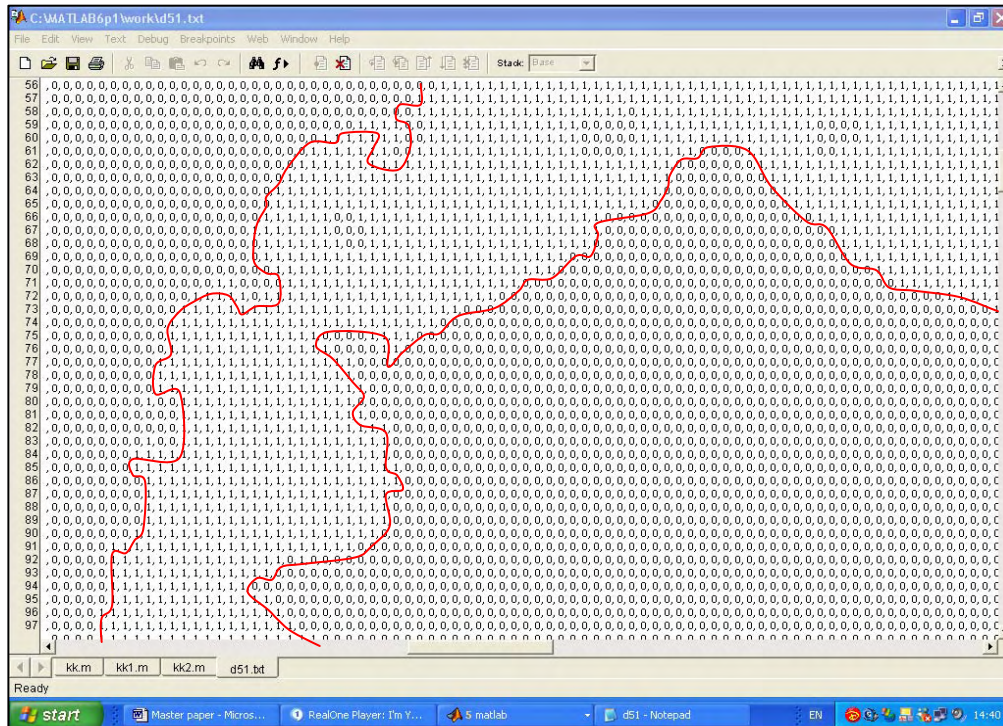


Inverted Image



Cleaned Image

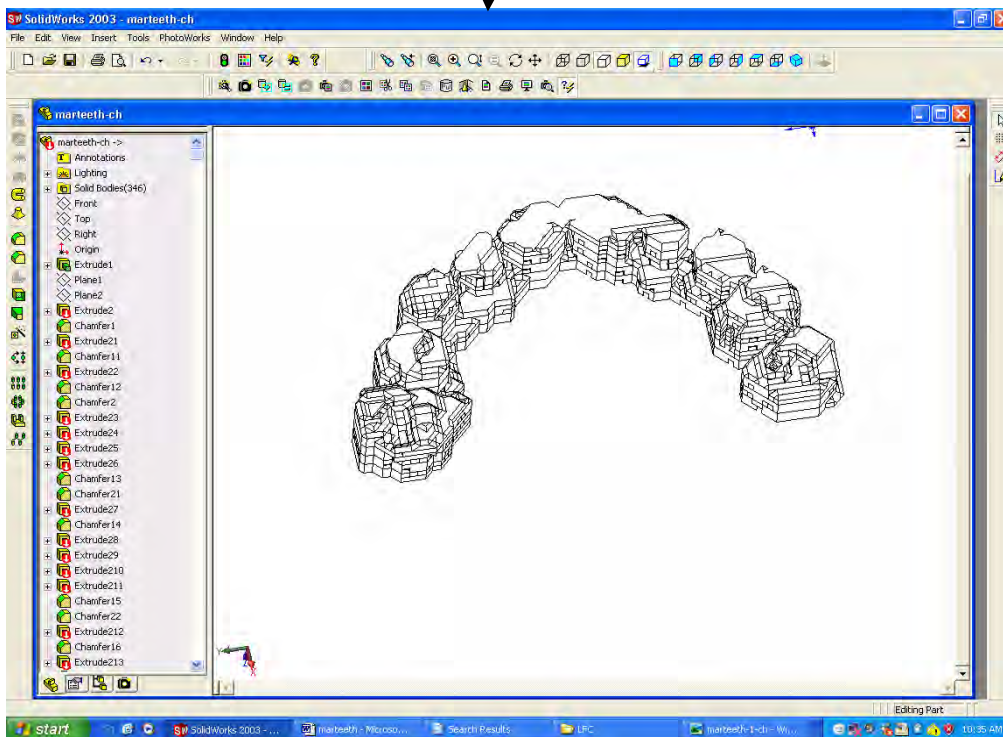




Digital Binary Matrixes

Edge configuration generation

Triangulated Cube configuration generation



3D Reconstruction

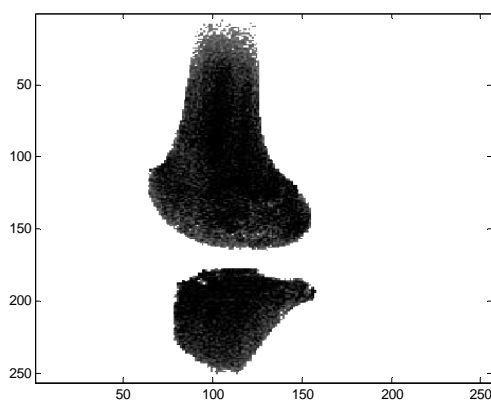
Fig. 5.2 Result from CT Scan to 3D Model for Teeth



Original Image



Inverted Image



Cleaned Image



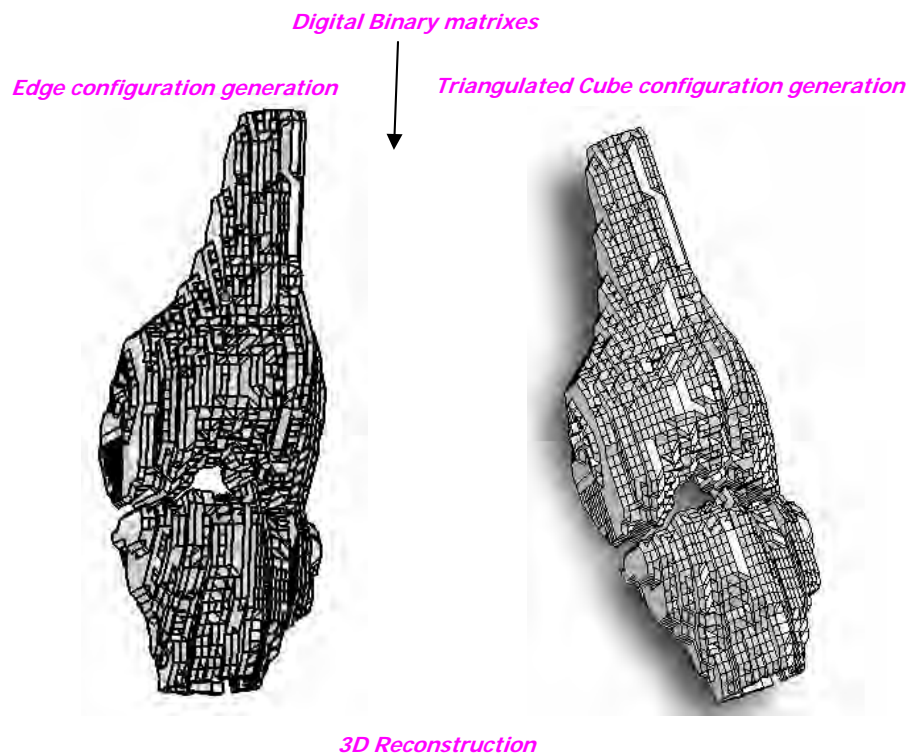
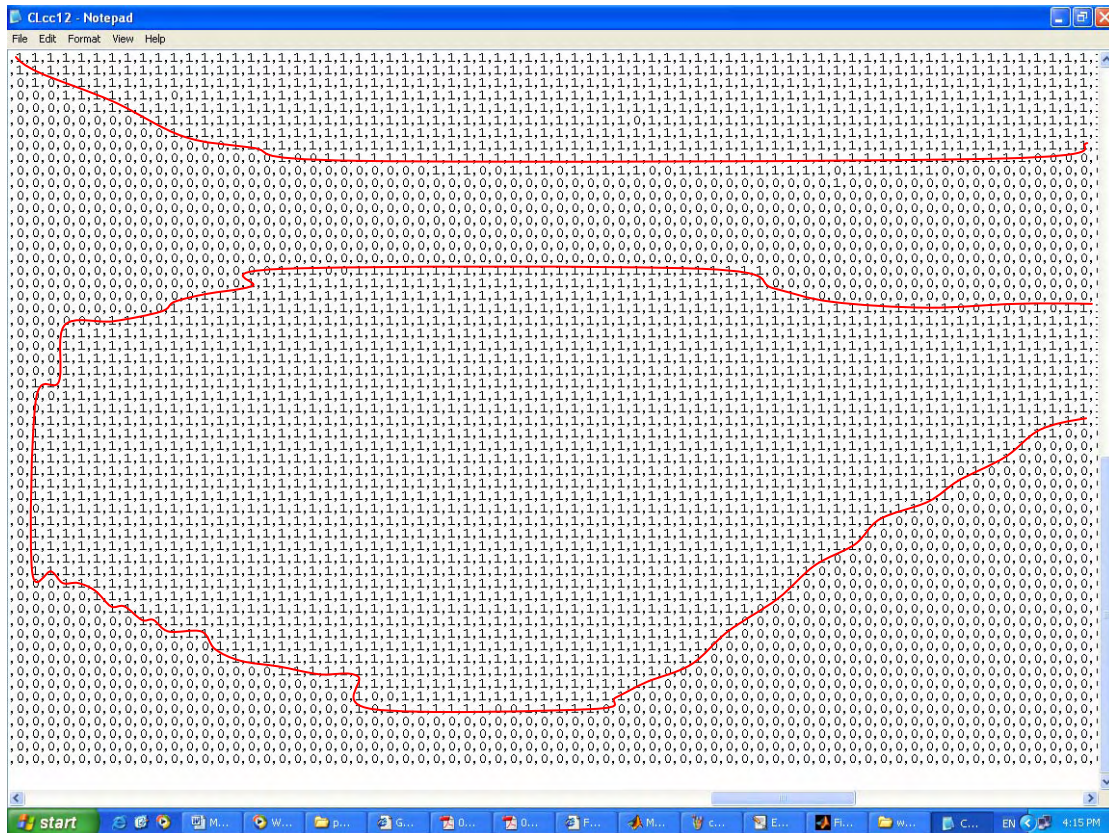


Fig. 5.3 Result from CT Scan to 3D Model for Knee

From the result, it can be observed that the program developed in this research project can be used to construct 3D model based on 2D CT scan images, which can then be used to machine a physical model. The pixels of this CT scan image are converted directly to the digital binary matrixes. The vertices-based data are interpolated by means of the Marching Cube algorithm, which converts the cube model into the triangulated cube configuration model by piecewise linear approximation. The linear interpolation techniques are considered as the most suitable, since the triangulated cube configuration representation is a kind of linear approximation.

On other hand, stair-casing effect is highly evidenced in the models and this could be explained by the fact that large layer thickness was required to reduce the time for producing such large models. The surface is rough and lacks fine details, particularly around the models region. The tests are initial steps and the results are very limited. In future, the results will need to be improved further.

In short, the results illustrate that:

- 1) The software tool can produce solid geometric models from CT scan data and MRP manufacturing.
- 2) Cube size has an impact on 3D reconstruction. The smaller the object is divided, the higher the solution of 3D reconstruction could be built.
- 3) Using the library feature in SolidWorks increases the program performance speed.
- 4) In the real working environment, the surgeon can rotate, zoom in/out, or change the 3D models, manipulation that can assist in pre-surgical planning.

The following some problems have been raised:

- 1) The cube size cannot be changed, which affects the quality of 3D reconstruction.
- 2) The exact isosurface has a staircase appearance which is preserved by use of face shoulder and interior points. Therefore, the 3D model is not smooth and the resolution is too slow.

- 3) The 3D model resolution needs to be improved by using interpolate method for obtaining smaller size of cubes.

Chapter 6 3D Reconstruction by Planar Contours

As mentioned above, 3D reconstruction models are very rough and limited, which takes time to improve the quality of 3D models. To get better 3D medical models in SolidWorks, Planar Contours, which use contour lines to approximate complex surfaces, will be employed for computer modeling and rapid prototyping. The initial planar contours method is to connect these contours by a triangulation in 3D space. The triangulation process is complicated by the occurrence of multiple contours on a data slice. In this research, planar contours method constructs 3D models by using *loft* command in SolidWorks, which saves the time and improves the quality.

The key steps in Planar Contours approach include capturing section contour points from medical image per slice, creating B-spline curve with the control points in each layer, and producing solid model construction.

6.1 Section Contour points Capture

Digital binary matrixes, slice by slice, were got by image processing in chapter 2, which were then brought into the SolidWorks environment for digitizing the section contour points using input and searching function. Figure 6.1 shows the reasoning flow chart.

Note that each section image was imported into SolidWorks and placed on corresponding sketch plane. A set of sketch planes parallel to a reference plane was created in SolidWorks. The distance between two adjacent sketch planes was set to be identical to the distance between two corresponding sections obtained from adjacent CT or MRI slices. These section contour points for teeth per slice from CT scan of a head were created along reasoning flow chart, as shown in Figure 6.2.

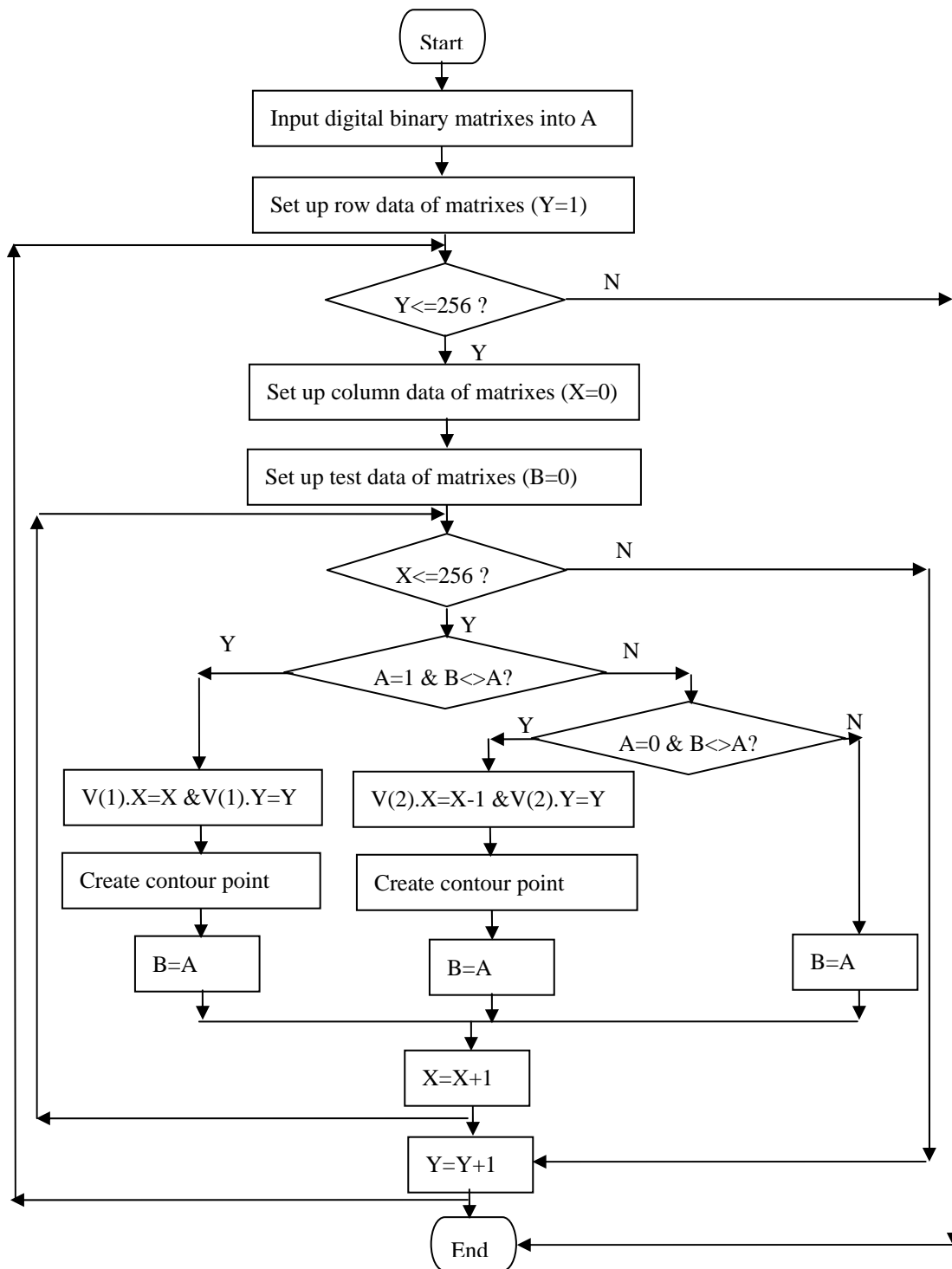


Fig. 6.1 Section Contour Points Capture Flow Chart

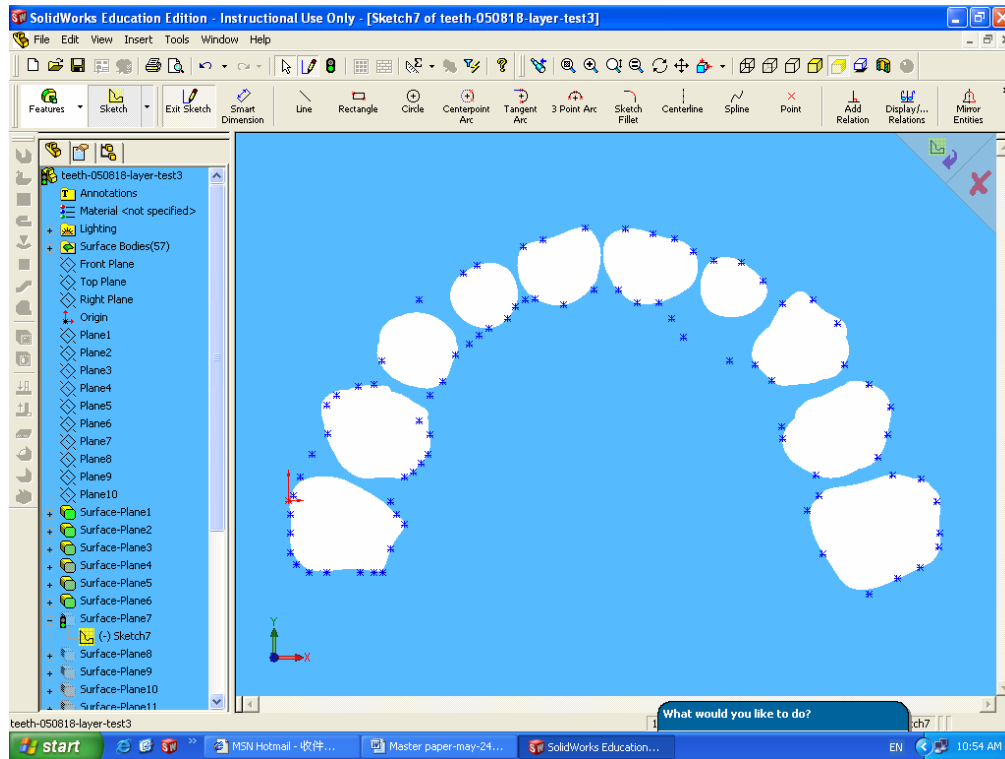


Fig. 6.2 Section Contour Points Capture in SolidWorks

6.2 B-Spline Curve Creation

The second step was to creation of splines, with the *fit spline* command in SolidWorks. This digitization was conducted by properly marking points along the exterior contours and using the point option of the spline mode in SolidWorks. Handles appeared at each spline point with arrows that controlled the vector leaving that point. Technically known as Bezier (B-spline) handles, these little arrows shape the B-spline curve. B-spline curves were formed in SolidWorks using the curve fitting technique, which employed the least square fitting for discrete points measured on a pre-selected section of an object. The best fitting curve can be obtained by minimizing the distance sum between the curve and the geometric points, as shown in Figure 6.3. The shape of the B-spline curve depends on the directions of tangent vectors. Spline can have as few as two points and can specify tangency at the end points. B-spline curves are used to provide

accurate interpolation of the intersection data points. To achieve higher accuracy, a larger number of linear segments are needed for the approximation.

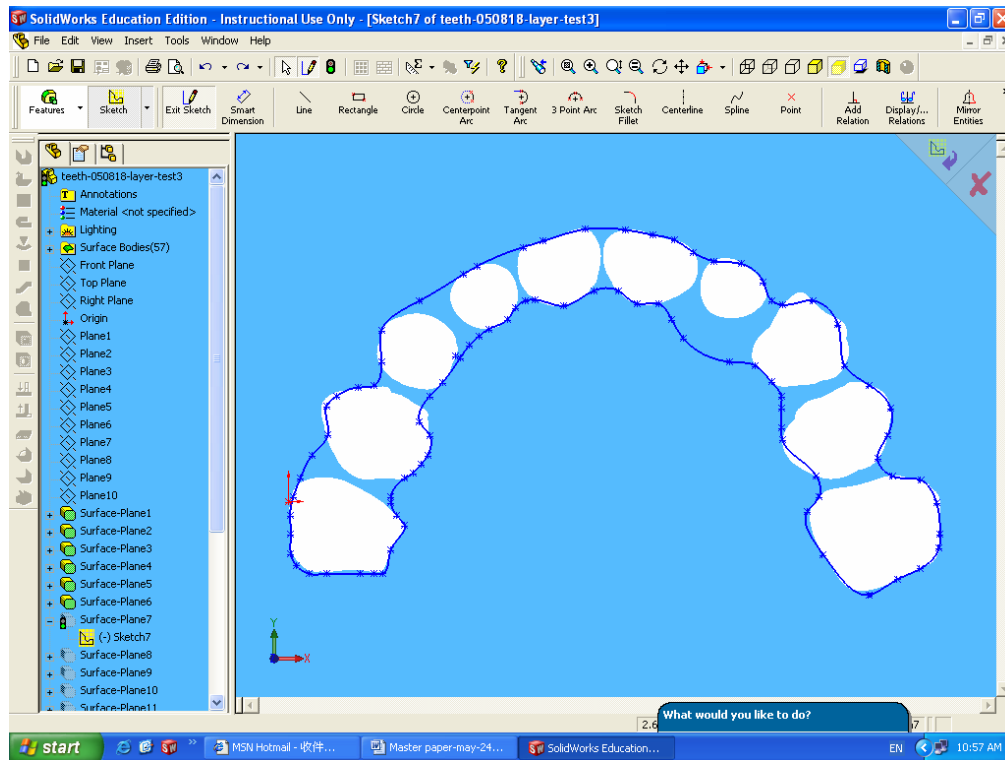


Fig. 6.3 B-spline Curve Creation

6.3 Solid Model Construction

After B-spline creation, the solid model was created using the *loft* features in SolidWorks. The loft feature created a solid model (see Figure 6.5) by connecting multiple closed curves on parallel planes (see Figure 6.4). The guide curves were selected along the loft direction to enhance the smoothness of the loft features. Solid Model was rendered using *shade* command in SolidWorks (see Figure 6.6). Consequently, 3D reconstruction model created by Planar Contour method is better than that produced by Marching Cube algorithm. The loft solid model created in SolidWorks is smoother and finer surface, which was helpful and convenient for further biomedical rapid design and manufacture (see section 6.3).

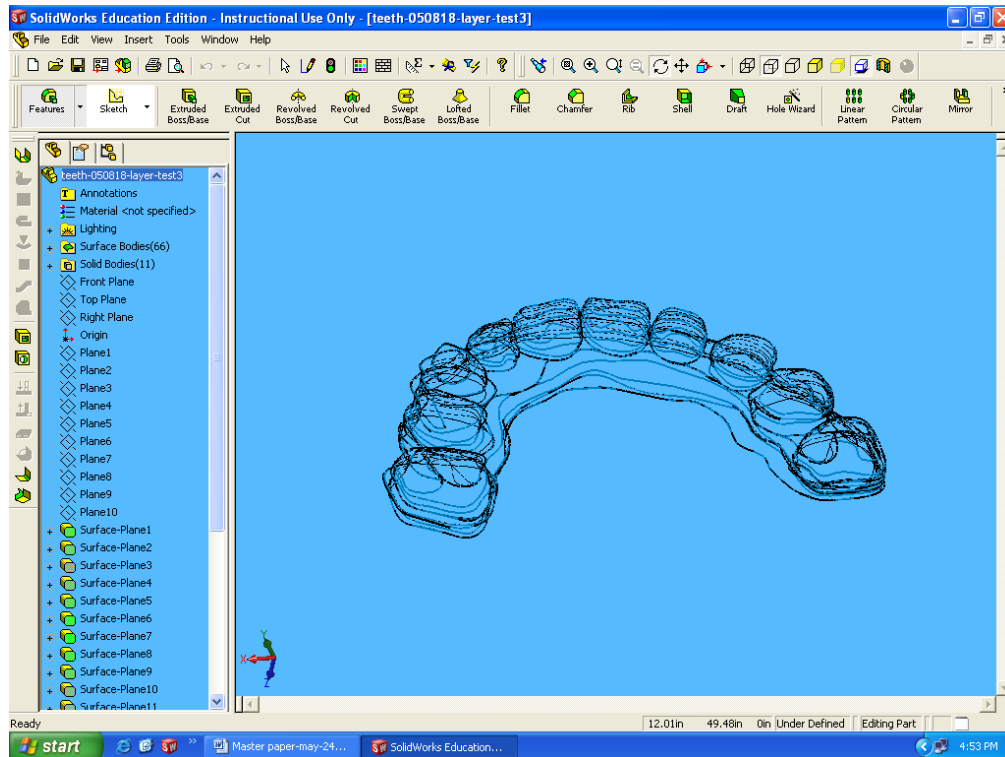


Fig. 6.4 18 Layers Closed Curves Selected for a Loft Feature

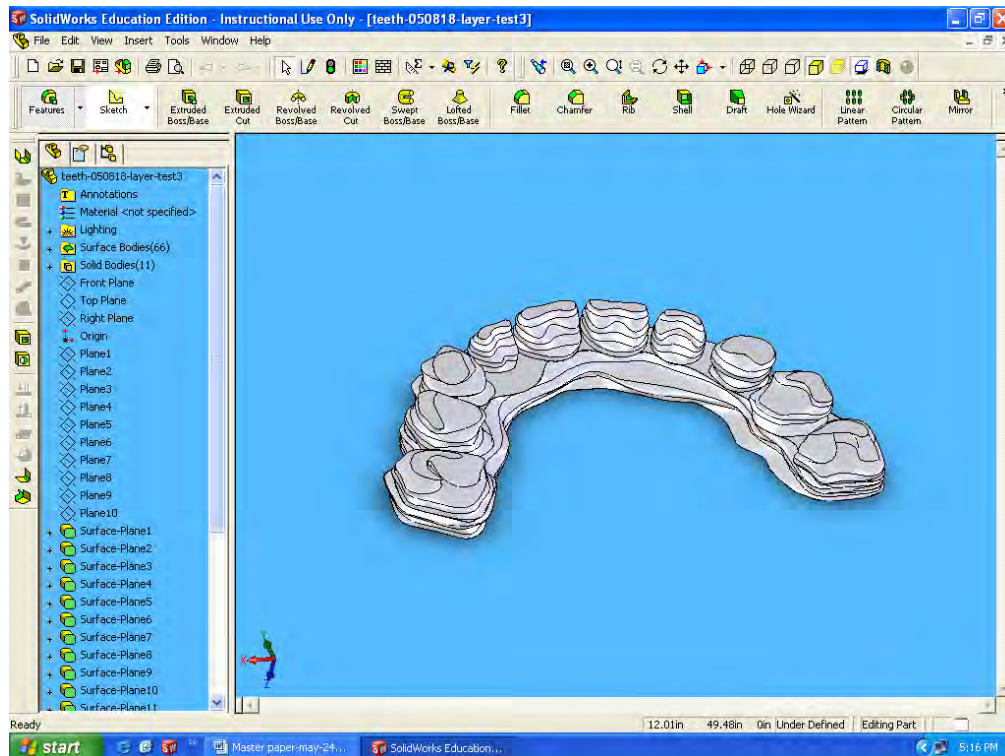


Fig. 6.5 Solid Model by Planar Contours Method

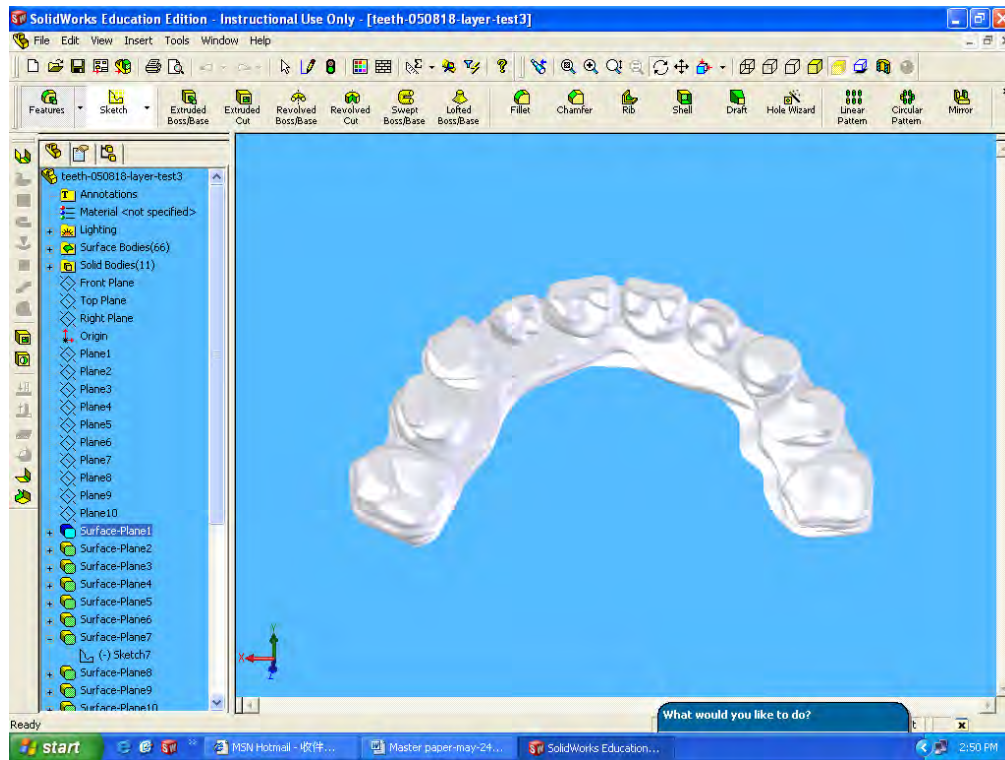


Fig. 6.6 Solid Model rendered in SolidWorks

6.4 Solid Model Result of Knee Joint

Solid model of knee joint was created along Planar Contours method. Figure 6.7 (a-e) shows the result through capturing contour points, creating B-spline curves, lofting features, building solid models, and rendering solid model. The result shows the better resolution for solid models.

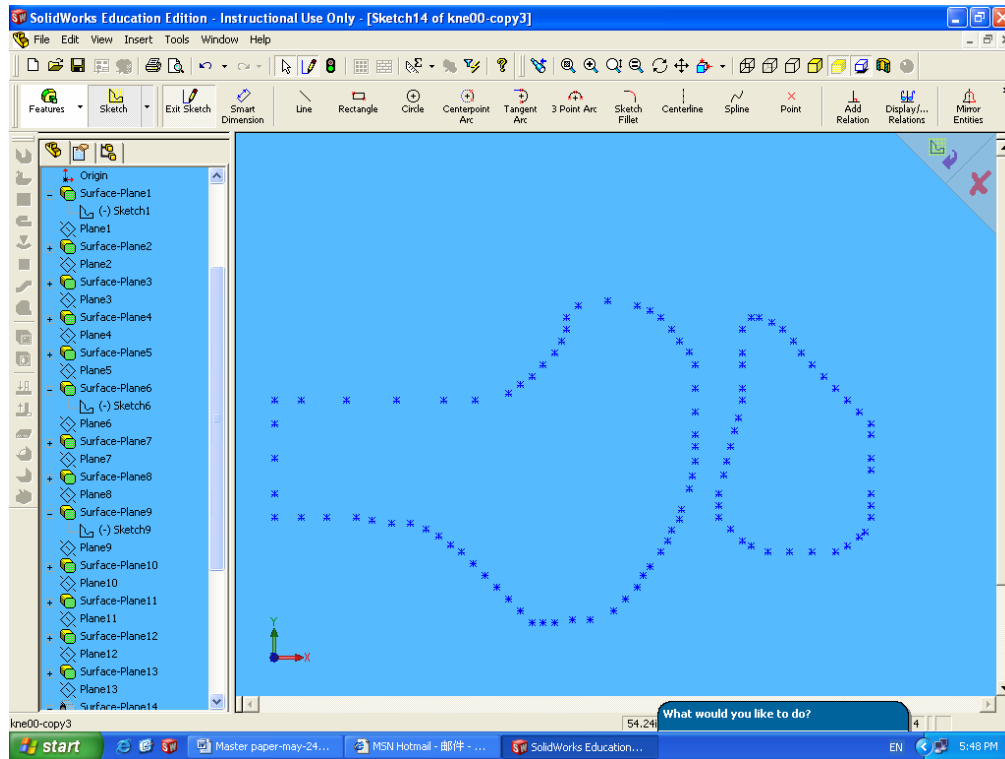


Fig. 6.7 (a) Capturing Contour Points

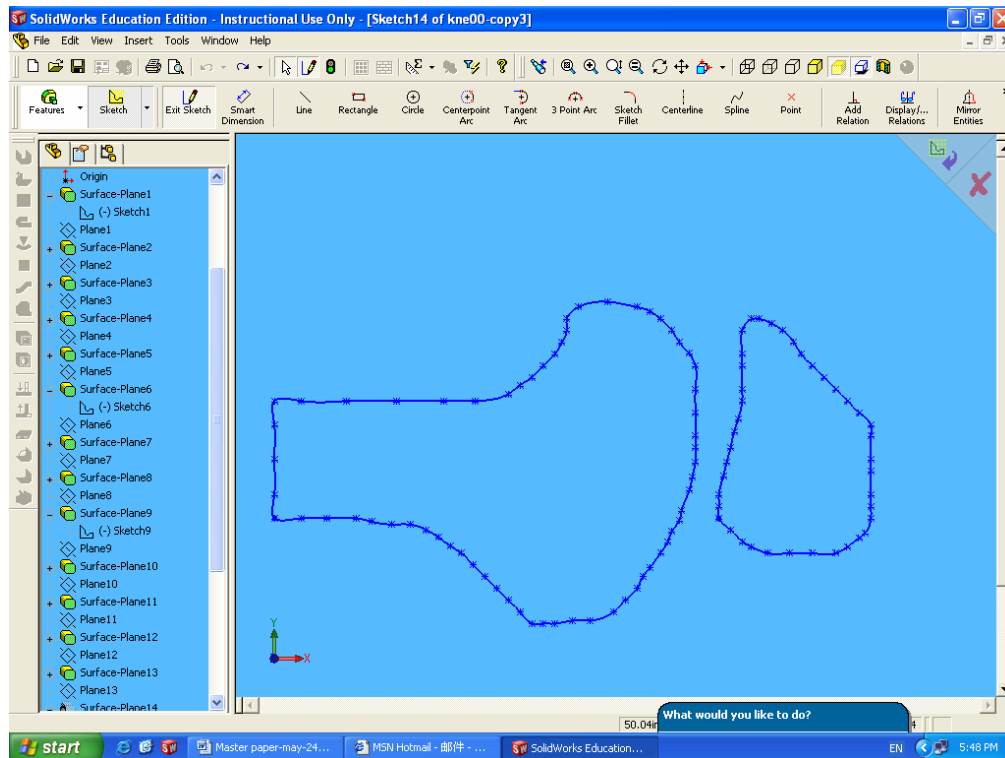


Fig. 6.7 (b) Creating B-spline Curves

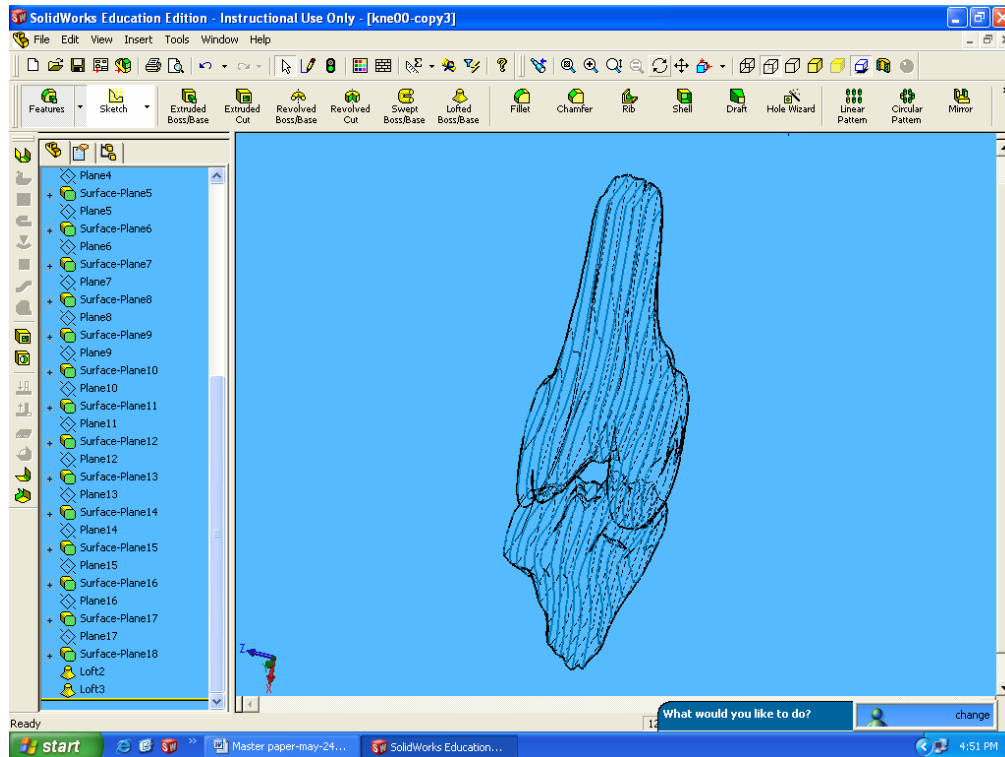


Fig. 6.7 (c) lofting features

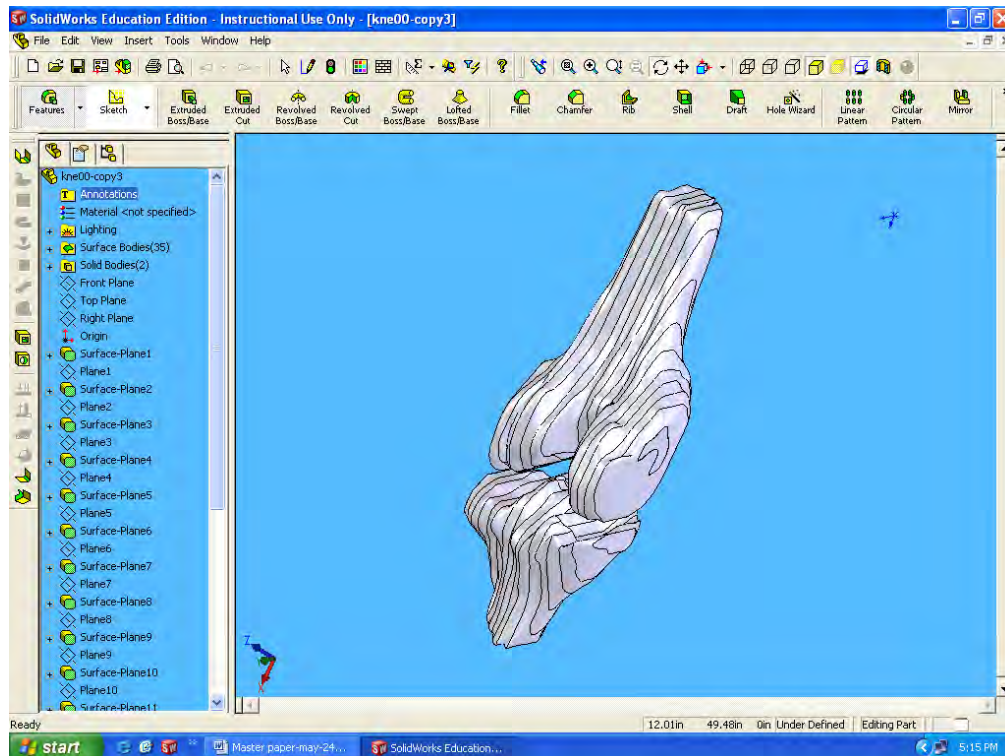


Fig. 6.7 (d) Building Solid Models

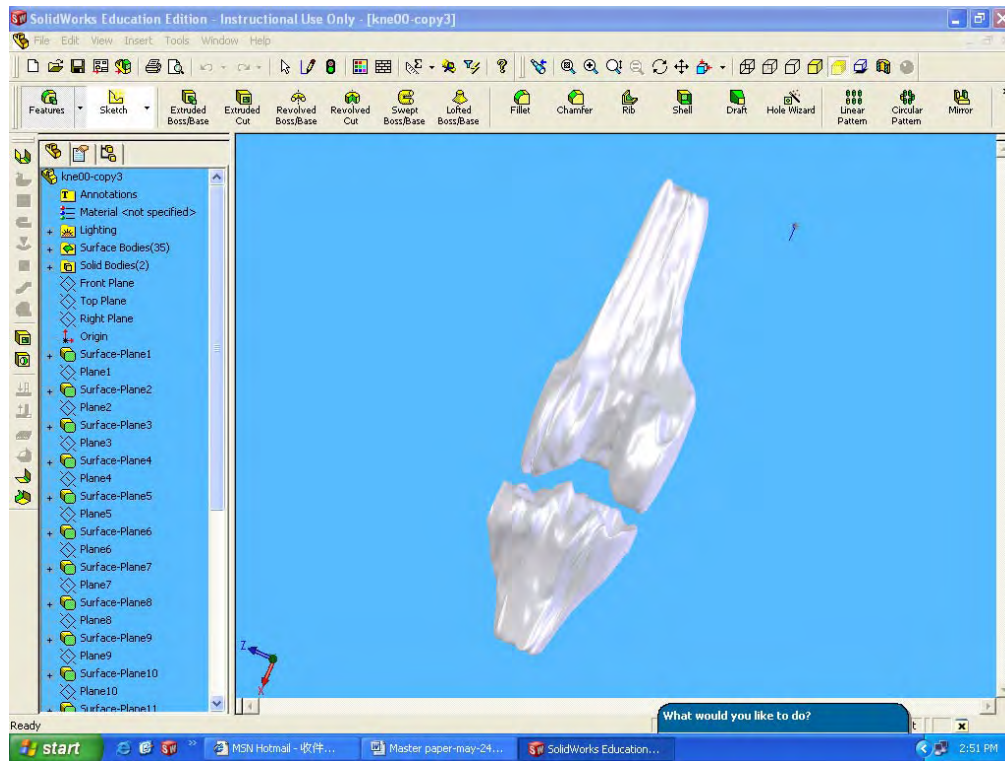


Fig. 6.7 (e) Rendering Solid Models

Chapter 7 Biomedical Rapid Prototyping

In medical industry, the use of Rapid Prototyping (RP) technology can improve services to patients such as surgical planning and implant designs. A precise of RP model facilitates the pre-operative planning of an optimal surgical approach. The reliability and accuracy of an RP model in surgical application allow surgeons to evaluate and select correct or appropriate implant approaches prior to operating the patient. In chapter, RP technique is based on layered manufacturing described in the following section. The second section discusses application of biomedical RP such as surgical planning and implant. The last section in chapter performs biomedical rapid design and manufacturing.

7.1 Rapid Prototyping Technologies

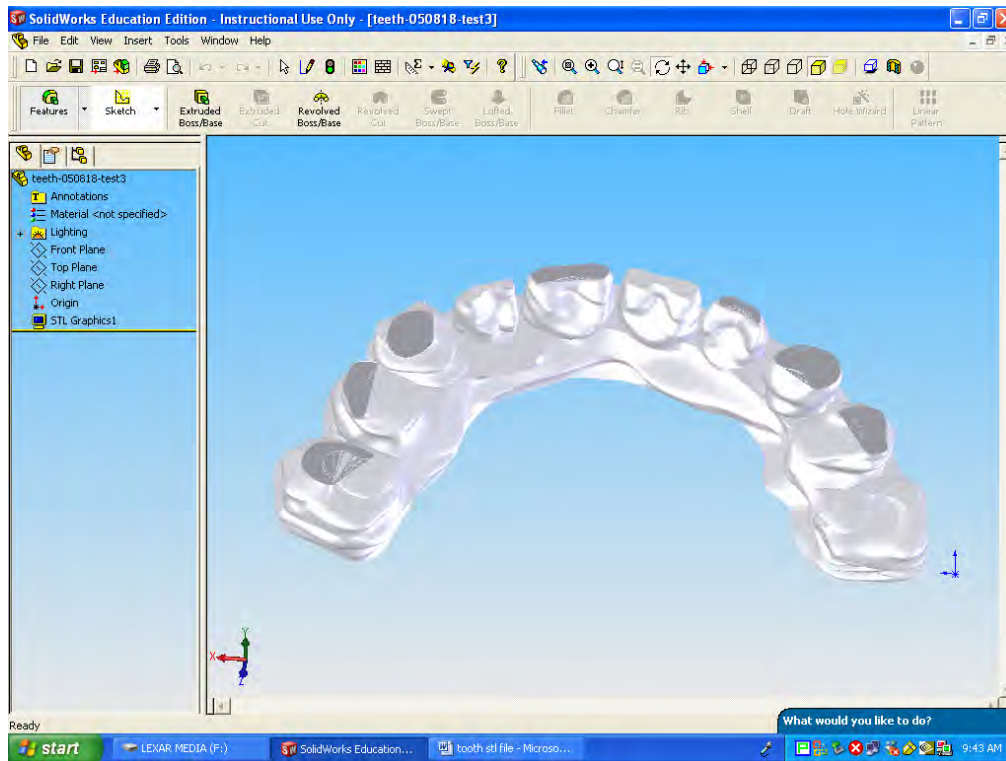
RP is the technique of manufacturing prototypes from complex 3D datasets, where all are based on layered manufacturing. The concept of layered manufacturing is the basis of all mainstream rapid prototyping processes. The idea behind layered manufacturing is that it is easier to build a series of 2D models then it is to build a single 3D model. Any solid or surface geometry can be interpolated to generate a series of 2D cross sections called “slices”. These slices are generated at evenly spaced intervals along the z-axis direction of the geometry. Each layer is then constructed sequentially to produce a model. This technology is fast developing and is more than competitive with traditional model building techniques, considering time and degree of detail.

Designing an interface to import the 3D images into RP format for manufacturing will include the following:

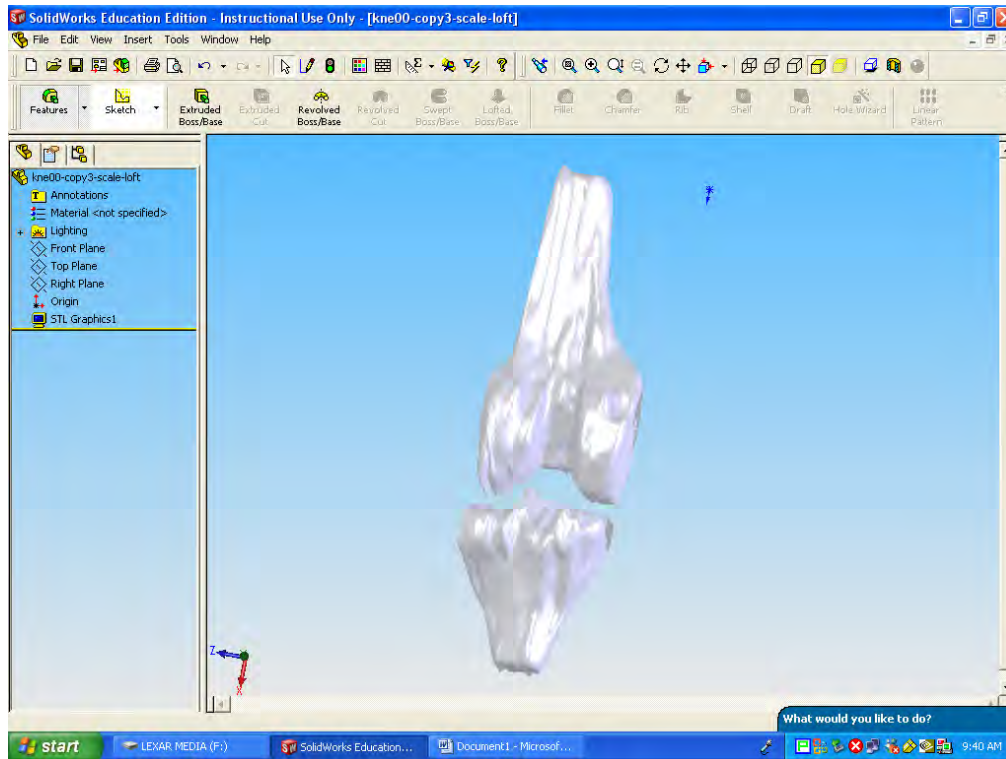
1. The RP interface accepts data in a format that accurately describes the surface of the anatomical organs.
2. The RP interface software “slices” the model data into very thin sections.

3. The RP system then builds the model slice upon slice.

To enable the slicing procedure, the Stereolithography (STL) geometry file of the 3D model has to be generated that is surface-mesh created with triangular elements. This process is done directly through SolidWorks. The STL format is standard for RP technology. The STL file contains an array of independent 3D triangles representing the model surface. Each triangle contains 3D points and one 3D normal vector. The STL file is an approximate representation. Only flat surfaces can be represented perfectly. All curved surfaces are approximated using a chord height criterion [49, 50]. Figure 7.1 (a) and (b) shows the teeth and knee STL files created by 3D reconstruction models in SolidWorks.



(a) *Teeth STL File*



(b) Knee STL File

Fig. 7.1 RP Models

Some of most commonly available systems are: Fused Deposition Modeling (FDM) [51], Stereolithography (SLA) [52], Selective Laser Sintering (SLS) [53], Sanders Prototyping Technology, and Z Corporation Fabrication Machine [54]. The FDM process from Stratasys constructs each layer with a path of heated extruded plastic [55]. The SLA process from 3D systems traces each cross section in a shallow pool of photocuring polymer using a laser [56, 57]. The SLS process from DTM constructs each layer by fusing powder along a path with a high powered laser. Z Corporation Fabrication Machine constructs layers using a modified inkjet print head to spray binder in powder such as starch or plaster [58-60].

Material development lagged behind other aspects of the industry. It has the following basic requirements: 1) exist in both liquid and solid state; 2) have a low viscosity in the liquid state; 3) and adhere strongly in the solid state. A phase or chemical change is used to change from a liquid state to the solid state. The resultant energy,

however, cannot be enough to melt or warp either material in its solid state. A high viscosity liquid will not fill small cavities, but will trap air bubbles within it. Poor surface quality will result in if the machined surface intersects a bubble.

Medical Rapid Prototyping is the production of medical models using rapid prototyping methods. The selection of a particular process will depend on the medical model application. With SLA, any anatomical object, regardless of its complexity, can be built automatically from its “CAD” file without the need for tools or manual interference.

A brief overview of the major RP technologies available today is given below.

Stereolithography (SLA)

Depending on the technology used, the 2D files are used to guide a laser beam in cutting sheets into the equivalent solid layers used in the SLA technique. The SLA system consists of an Ultra-violet Laser, a vat of photo-curable liquid resin, and a controlling system. A platform is lowered into the resin (via an elevator system), such that the surface of the platform is a layer-thickness below the surface of the resin. The laser beam receives its path instructions for each slice to trace the boundaries and fill in a 2D cross section of the model, solidifying the resin wherever it touches. Once a layer is complete, the platform descends a layer thickness, resin flows over the first layer, and the next layer is built. This process continues until the model is complete [56]. Figure 7.2 shows the sequence of steps for producing a SLA layer.

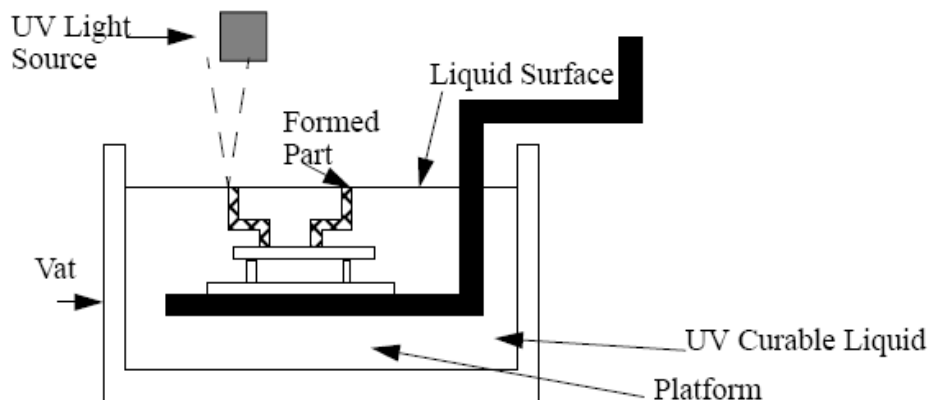


Figure 7.2 Process of RP for SLA Technique

The materials used by SLA equipment are epoxy-based resins that offer strong, durable, and accurate models. It is ideal for form, fit, and function testing, as well as for visual aids and patterns for tooling. In many cases, SLA is capable of reproducing snap fits. In general SLA materials have a low heat tolerance with typical heat deflection temperatures are around 110–120° F. Standard tolerances are $\pm 0.005''$ for the first inch, and $\pm 0.002''$ in most parts and features. These characteristics make SLA an excellent all round choice for prototypes. The advantages of SLA process are high accuracy , very good surface finish, semi-transparent material and moderate strength.

Selective Laser Sintering (SLS)

Laser Sintering uses fine powders of a wide range of materials to be treated in nitrogen atmosphere. The powder is heated up to a temperature just below the melting point of the specific material, which usually will take a couple of hours. A roller spreads the powder on the building platform. The laser beam then selectively melts the powder and bonds it. As the power is already heated, the laser needs to elevate the temperature slightly to cause sintering. The temperature gradients in the part remain small. The platform moves down incrementally and the process starts again, until the prototype is finished. Subsequently the building chamber piston raises completely to deliver the part. Excess powder is brushed away and final manual finishing may be carried out. Figure 7.3 shows the process of selective laser sintering. No supports are required with this method because overhangs and undercuts are supported by the solid powder bed. The advantages of SLS process are that it produces a large model, which is durable, functional, fast, and can be finished and painted [54, 57].

SLA vs. SLS: A Summarized Comparison

Material Properties: The SLA (stereolithography) process is limited to photosensitive resins which are typically brittle. The SLS process can utilize polymer powders that, when sintered, approximate thermoplastics quite well.

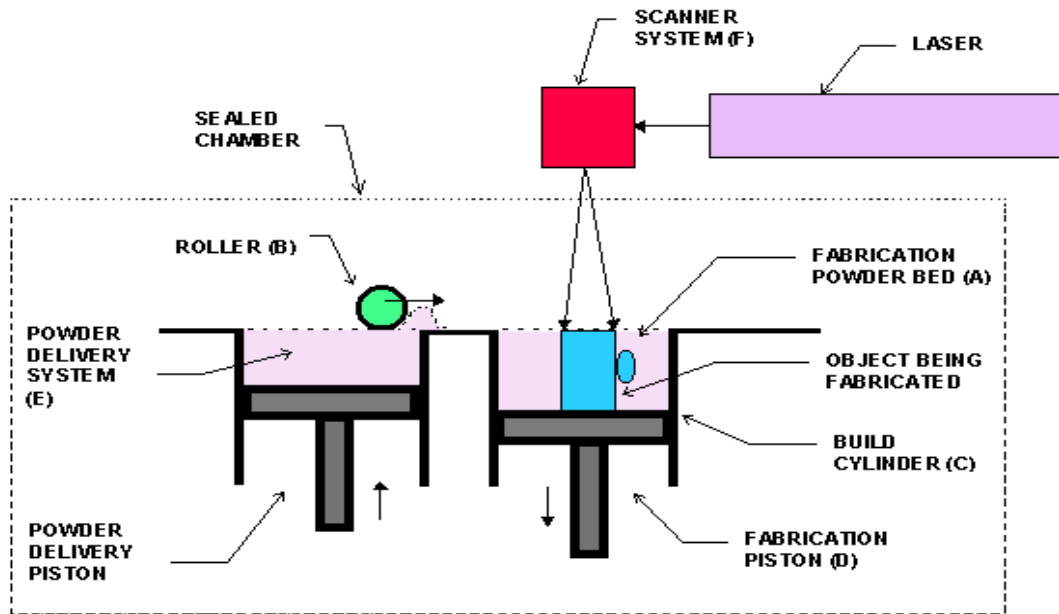


Fig. 7.3 the Process of Selective Laser Sintering

Surface Finish: The surface of an SLS part is powdery, like the base material whose particles are fused together without complete melting. The smoother surface of an SLA part typically wins over SLS when an appearance model is desired. In addition, if the temperature of uncured SLS powder gets too high, excess fused material can collect on the part surface. This can be difficult to control since there are so many variables in the SLS process. In general, SLA is a better process where fine, accurate detail is required. However, a varnish-like coating can be applied to SLS parts to seal and strengthen them.

Dimensional Accuracy: SLA is more accurate immediately after completion of the model, but SLS is less prone to residual stresses that are caused by long-term curing and environmental factors. Both SLS and SLA suffer from inaccuracy in the z-direction (neither has a milling step), but SLS is less predictable because of the variety of materials and process parameters. The temperature dependence of the SLS process can sometimes result in excess material fusing to the surface of the model, and the thicker layers and

variation in the process can result in more z inaccuracy. SLA parts suffer from the "trapped volume" problem in which cups in the structure that hold fluid cause inaccuracies. SLS parts do not have this problem.

Support Structures: SLA parts typically need support structures during the build. SLS parts, because of the supporting powder, sometimes do not need any support, but this depends upon part configuration. Marks left after removal of support structures for parts cause dimensional inaccuracies and cosmetic blemishes.

Machining Properties: In general, SLA materials are brittle and difficult to machine. SLS thermoplastic-like materials are easily machined.

Size: SLS and SLA parts can be made the same size, but if sectioning of a part is required, SLS parts are easier to bond.

Fused Deposition Modeling (FDM)

In FDM, the material is prepared in filament form, available on spools and in many different colors. This filament is heated in a nozzle, which moves in the X and Y directions according to the CAD data. The molten material is thus added layer by layer. A second filament is used equally to build up the necessary supports (See Figure 7.4). The supports are easily removable using a water-based solution that dissolves them and leaves the model with smooth surfaces. FDM is a free-form fabrication technology. Because it uses high strength ABS plastic, it is the favored technology for prototyping plastic parts requiring strength. The advantages of FDM process are that it is high strength, cost-effective, and waterproof, and that it can accommodate ABS material and multiple material colors [54, 61].

Z Corporation 3D Printing

Z Corporation 3D Printing uses inkjet printer technology to print fine patterns of glue onto a smooth bed of plaster powder. First, the 3D Printer spreads a thin layer of powder. Second, an ink-jet print head prints a binder in the cross-section of the part being created.

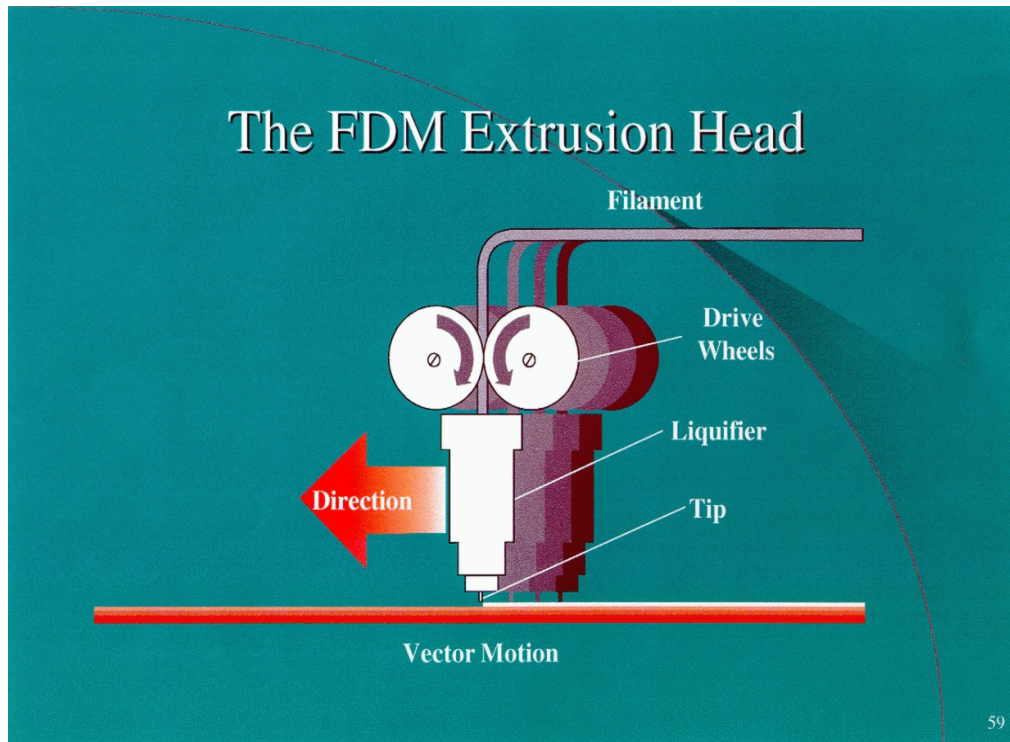


Fig. 7.4 the Process Fused Deposition Modeling

Third, the build piston drops down, making room for the next layer, and the process is repeated. Once the part is finished, it is surrounded and supported by loose powder, which is then shaken loose from the finished part. It is especially good at making parts that have hard-to-reach cavities, as the scrap can be poured and vacuumed out. Figure 7.5 shows how the Z-Corp 3D Printer fabricates solid parts from layers of powder [58, 60].

No support structure in Z Corporation 3D Printer means that parts can be made with very complex geometries, and when complete, simply “blow out” the powder. Inkjet technology makes this a very high-speed option. Z Corporation 3D Printers enable surgeons to rapidly produce inexpensive 3D models to obtain better case information to reduce operating time, enhance patient and physician communication, and improve patient outcomes. The ability to use models for pre-surgical planning reduces operating room time, lowers cost, and enhances patient outcomes by minimizing incision sizes, reducing recovery time and allowing for procedure rehearsals. Z Corporation models

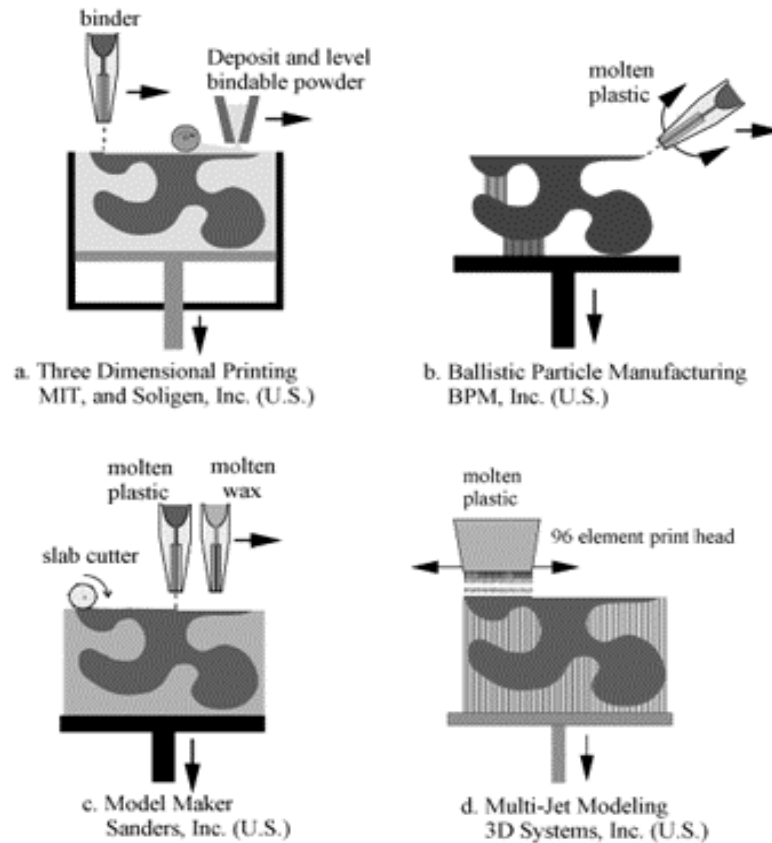


Fig. 7.5 Z Corp. 3D Printer

permit the world's leading implant manufacturers to fabricate custom implants rapidly and cost effectively for the ultimate in performance. Therefore, Z Corporation 3D Printer is used in this study to enable the slicing procedure. Figure 7.6 shows the Z Corporation 3D Printer in making a 3D solid model.

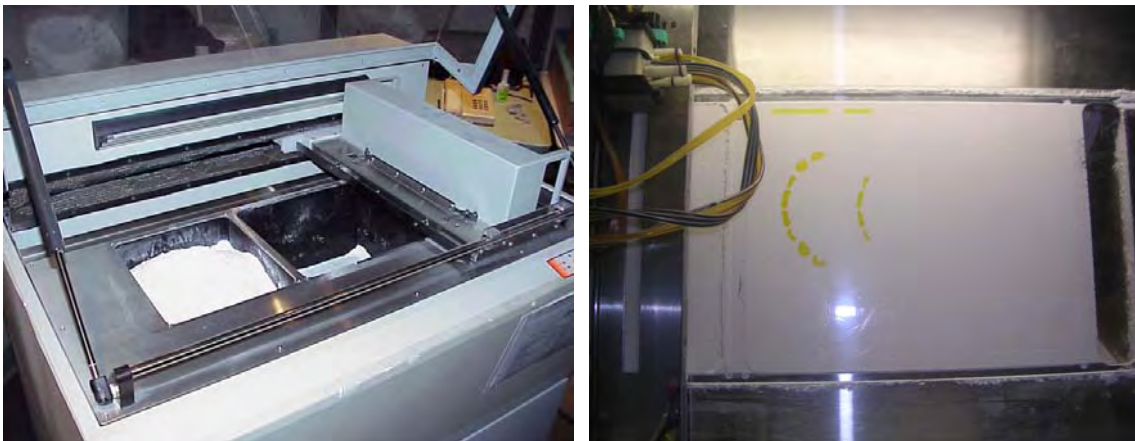


Fig. 7.6 Z Corporation 3D Printer in Making 3D Solid Model

7.2 RP Application in Biomedical Field

The application of RP processes to medicine appears very promising. A direct interface between the three dimensional reconstruction models and RP allows the development of physical, real 3D models of any anatomical structures [61, 62]. Such models are used for purposes of visualization and communication, surgical rehearsal, and custom implant preparation.

7.2.1 Surgical Planning

3D models are used by surgeons for more accurate surgical planning and better diagnostic methods. This planning reduces the risk to the patient, owing to the shortened time of surgical procedures, and is less expensive than the alternatives. For example, the human spine needs to be created from CT and MRI-data in order to correct a deformity. The surgeon needs 3D models to plan and rehearse the procedures, because sensitive areas are involved, that take susceptible to severe damage. The case is Torticollis, which involves the cervical portion of the spine where a vertebra is out of alignment. This causes the neck to be mis-aligned and can result in damage to the spinal cord. The surgeon needs to know the exact location of the spinal cord compared to the vertebrae and how far the vertebra has to be moved. By using information obtained only from the CT-and MRI-images, planning the procedure is difficult and the risk is substantial. Providing a physical model of the actual case will ease the pre-surgical planning and decrease the risks involved in such a delicate procedure. Another case is Scoliosis, which involves lateral curvature of the vertebrae portion of the column due to uneven growth of the vertebrae. The surgery, which involves partial removal of the vertebra, involves high risk to the spinal cord. A 3D physical model of the patient's anatomy will help the surgeon to plan the procedure and minimize the risk involved (See Figure 7.7).

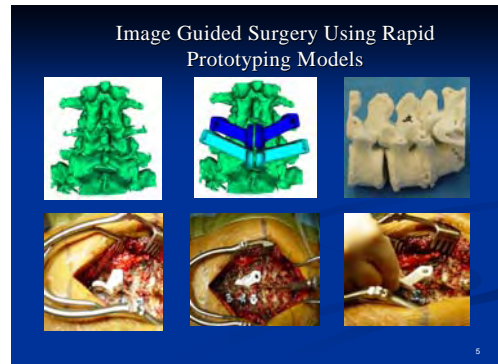


Fig. 7.7 RP model for Surgery Planning

7.2.2 Anatomical Implants

The implant would be made to fit exactly the patient's requirements in terms of shape, performance, and integration into existing structures within the body, using data collected by non-intrusive scans. RP is well suited to produce biomedical implants for bone replacement. 3D printing is an especially appropriate technique to generate complex porous ceramic matrices directly for biomedical applications. Anatomical information obtained from the patient is used to design and optimize the implant for a target defect. The use of RP allows 3D physical model to be created immediately, directly, and automatically from a 3D model. It works by breaking down a 3D model into 2D sections, which are built up layer by layer by high tech machines.

For example, the reason for customizing a knee implant is that the current procedure uses standard size implants, which cause the implant to loosen over time due lack of perfect fitting. However, using the 3D model created from CT-data to design a customized knee implant will fit perfect. Moreover, a stress analysis of the knee components can be conducted at different, angles, positions, and load distributions, which can vary due to different activities such as walking or running. Once the design is completed, the CT data for the subject can be used to prototype the femur, tibia and the corresponding customized implants, which would enable the precise manufacture of customized implant components (See Figure 7.8).

Therefore, the application of 3D reconstruction and RP techniques to medicine is an

invaluable contribution by engineering technology. The ability to produce 3D physical models directly from the scanned data promises to be the way of the future in medicine.



Fig. 7.8 RP model for Implant

7.3 Biomedical Rapid Design and Manufacturing

Biomedical design work is closely related to sculptural work. The human body does not have sharp corners or edges, making it were necessary to select CAD software that is versatile enough to give the model an irregular shape. The wide variety of modeling capability offered by SolidWorks software makes it suitable for biomedical design. SolidWorks software can provide an interface of the widest number of data translation formats of any CAD solution. It provides easy use of the CAD data, such as the above 3D reconstruction model data (see chapter 6), for maintaining in the original drawing while designing the medical model according to the biomedical needs. It allows the file to be checked and repaired for conceptual design, detailed design, and analysis. Using such an interface, image-based medical design becomes a reality.

SolidWorks capabilities provide for standard view drawings, including three views or any combination of views, which are automatically generated from the model or assembly with bill-of-materials included. Figure 7.9 shows the design of 3D medical

model created by planar contour method (see chapter 6) in SolidWorks. Figure 7.10 displays the 2D view of 3D model.



Fig. 7.9 Medical Design of 3D Reconstruction Model

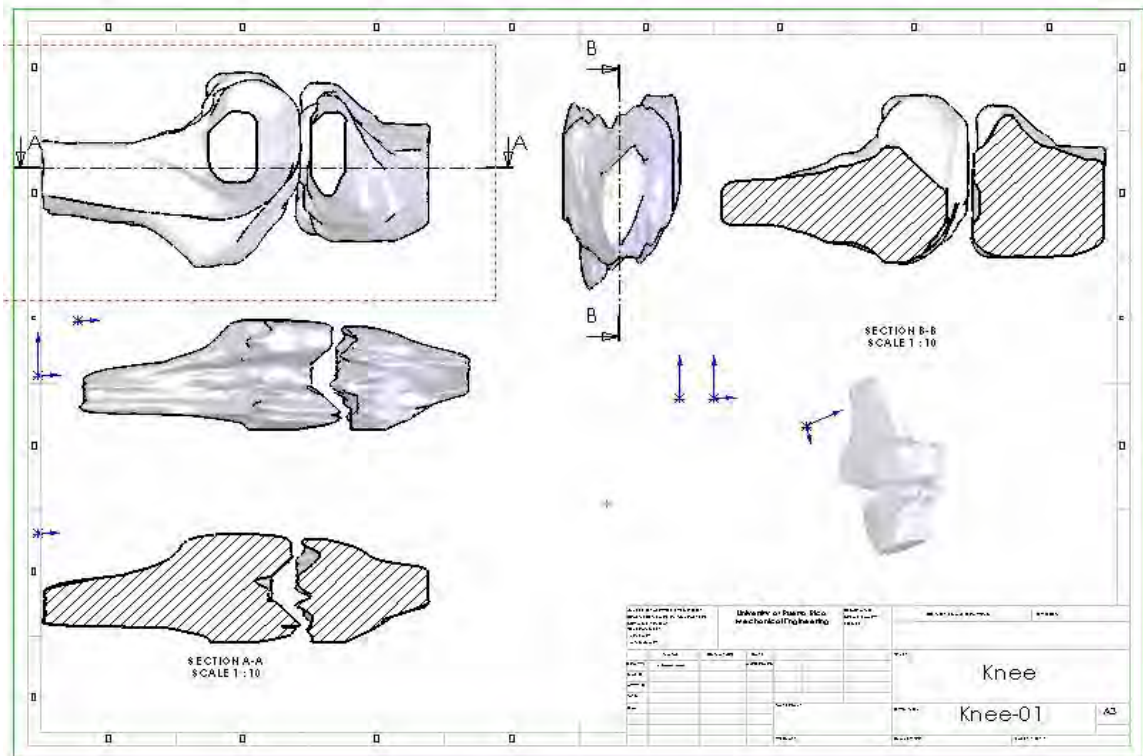


Fig. 7.10 2D View of 3D Model

3-D models of medical images are helpful tools used in surgical planning and as teaching aids. Currently, there is no cost-effective method of constructing these 3-D models from CT or MRI scans because scans are not homogenous. Creating biomedical models of human organs and regions can help surgeons to plan their surgeries. Implants and cut placements can be made and planned before cutting performing surgery. This minimizes both surgery time and cost. Biomedical rapid design, together with medical images, could be used to build accurate composite models of human organs and other structure in SolidWorks. This mechanically accurate model will allow for testing the efficiency of surgical devices on a more realistic model. The ability to model mechanical properties of complex structures could have many important implications. Specifically in the medical field, accurate models could assist in biomedical product testing and research.

Additional views can be easily added, including new breakout section views to display the detail drawing, such as the layer structures, the connection section, and the size value. Thus, it is convenient for medical manufacturing. Rapid prototyping is a manufacturing technique that creates various 3-D geometries by means of layer-by-layer construction. These layered manufacturing techniques can reduce the need for costly tooling, it is clear that it will make more sense in some situations than in others. The complex geometry cannot easily be made by conventional manufacturing techniques. Because layered manufacturing processes can produce parts of arbitrary complexity without the use of any fixed tooling, the processes will have a significant impact on the way product manufacturing. These manufacturing processes will actually allow designers to merge multiple parts together, thus requiring considerably less assembly. In the case of extremely small assemblies, it will even be possible to fabricate pre-assembled devices that require no assembly at all. For example, more knee joint replacement surgeries are performed than any other joint, so the joint component of the implant design is displayed in the section views for accurate biomedical manufacturing.

Many of rapid prototyping techniques such as SLA and FDM are primarily designed

to manufacture plastic parts. Using biomedical rapid design layer by layer, layered manufacturing can be modified for direct production of functional metal and ceramic parts. Figure 7.11 shows the knee implant manufactured with 3D model by layer by layer (a), bolt joint (b), and joint holes (c). Figure 7.12 (a), (b), and (c) show the results of 3D MRP manufacturing.

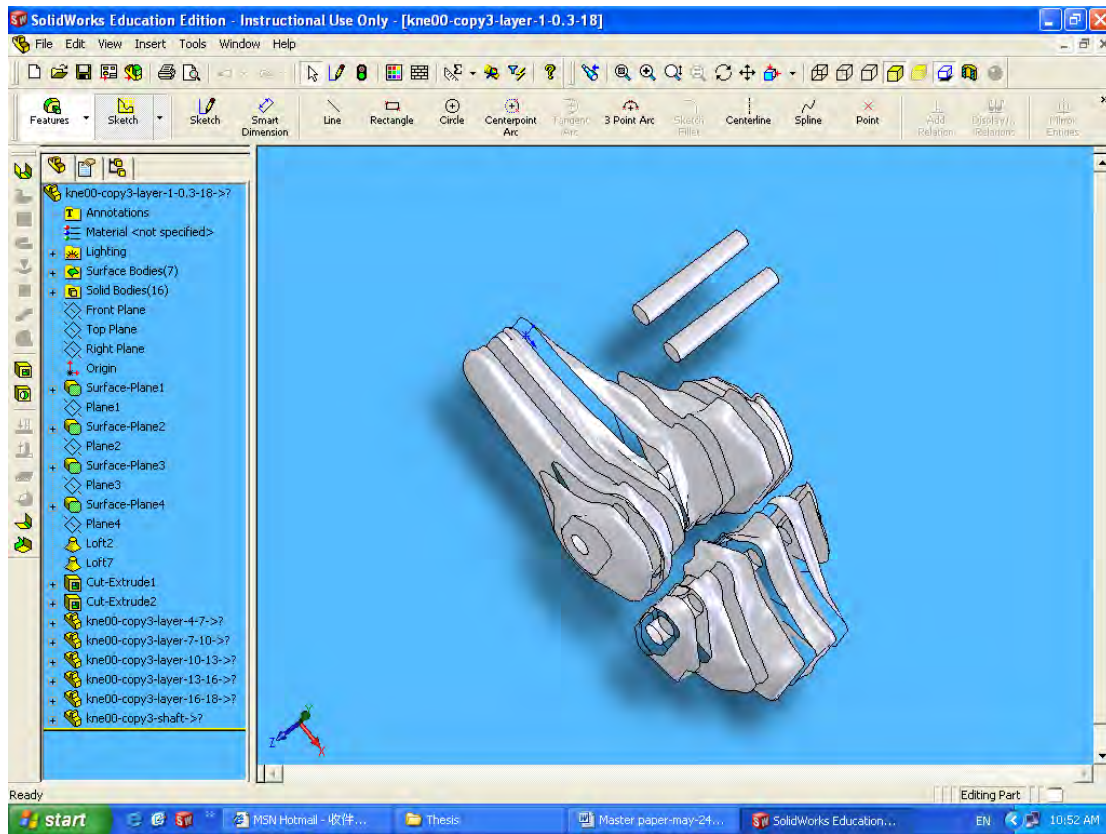
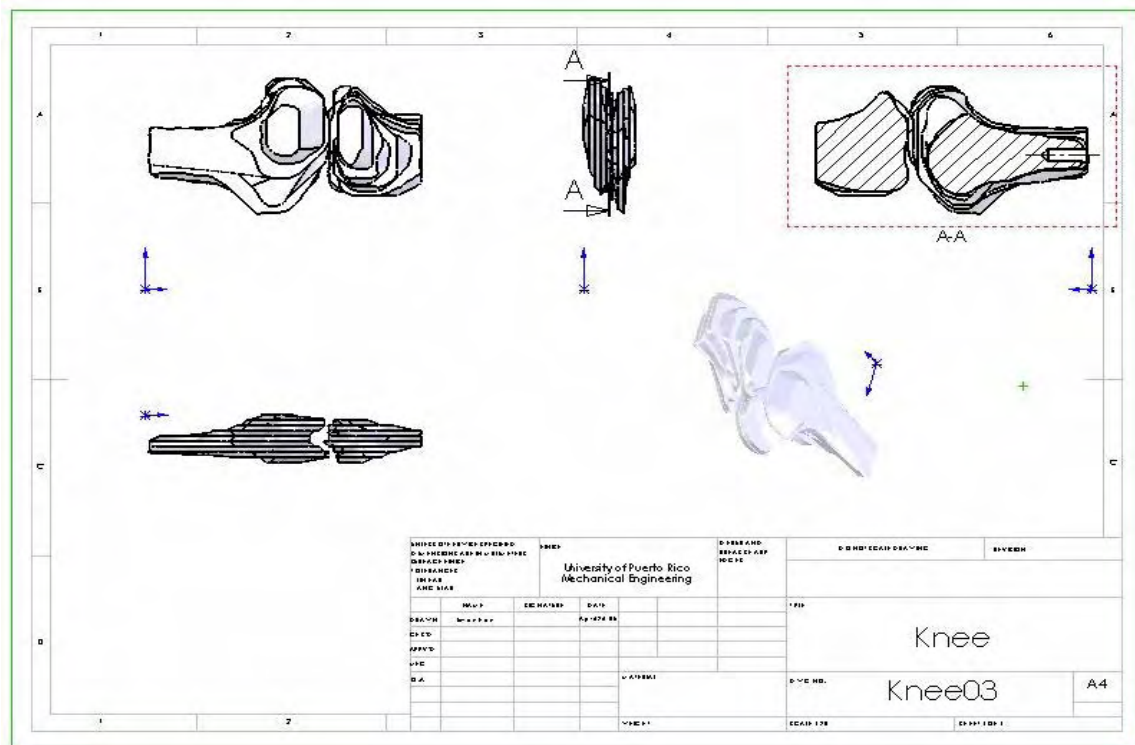
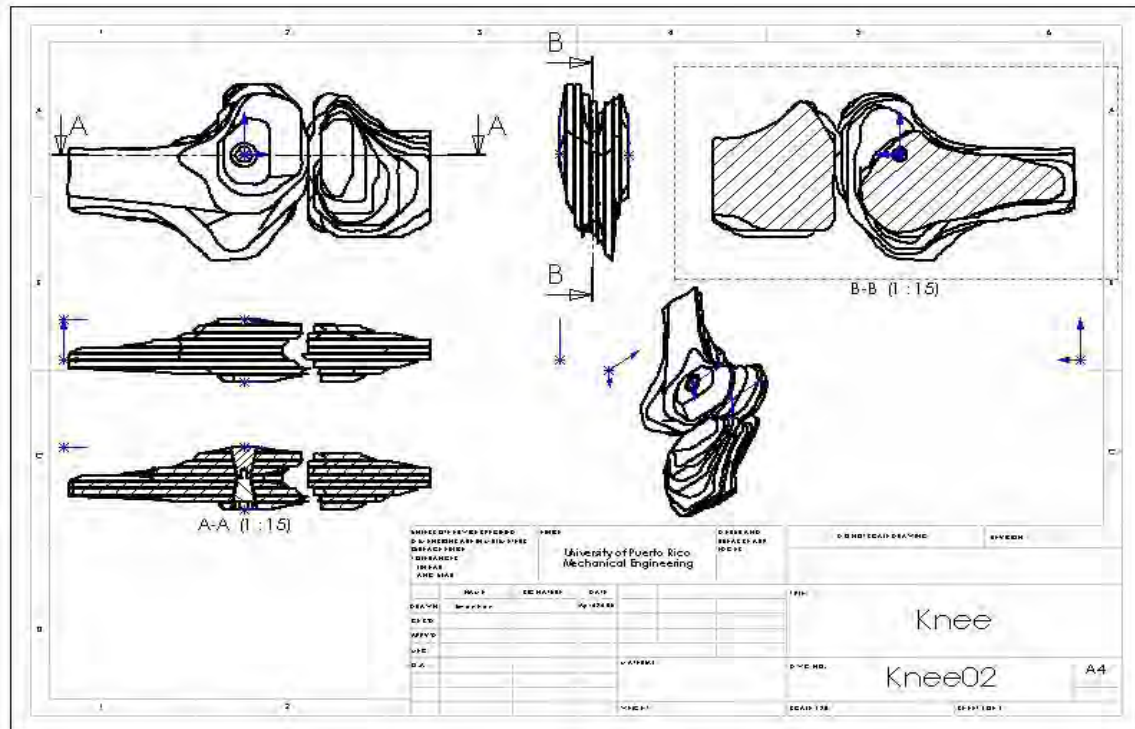


Fig.7.11 (a) 3D Model in Layer by Layer



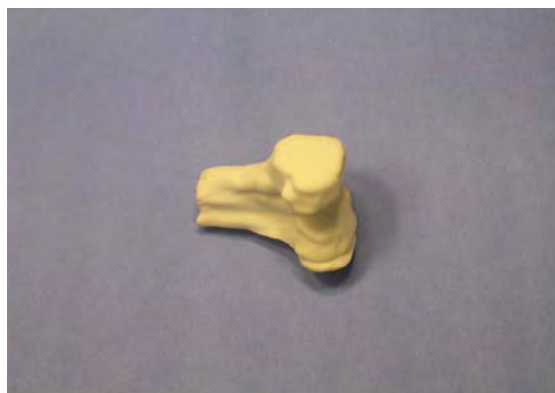
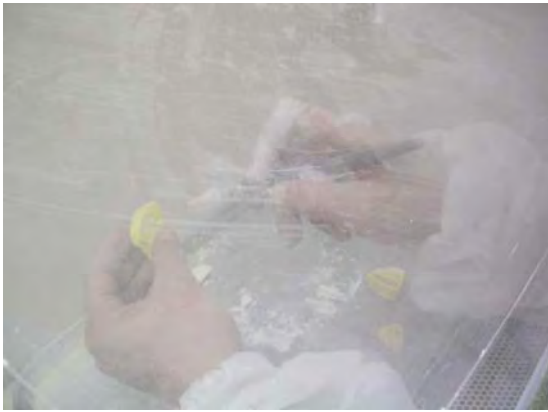


Fig. 7.12 (a) 3D Physical Model of Knee Joint



(b) 3D Physical Model of Knee Joint layer by layer



(c) 3D Physical Model of Teeth

For biomedical design, full associative assemblies are critical for effectively using bottom-up and top-down assembly design techniques. In SolidWorks, associative assemblies guarantee that all elements of a model are electronically associated or connected, including assembly models, components, drawings, details, and bills of materials. Thus, when a change is made to any SolidWorks file, the change is automatically made in all associated files. Therefore, the 3D reconstruction model generated by SolidWorks software can be designed and managed concurrently. This integration of technologies, such as biomedical imaging, design, and manufacturing, plays an important role in medicine. Layered manufacturing techniques can further be used to build electromechanical components for sensor application.

Chapter 8 Conclusions and Future Works

8.1 Conclusions

The ability to produce physical models from the scanned data in SolidWorks is an important contribution from engineering technology to the medicine. Biomedical rapid design and manufacturing in SolidWorks could help surgery to plan, manage, and manufacture concurrently. Because 3D reconstruction is performed in SolidWorks, the fields of Finite Element Analysis, MENS, and Mechanical Engineering can be combined with the areas of surgical planning and implantation. The combination of 3D reconstruction and Rapid Prototyping will have a significant impact on biomedical engineering and surgery.

The results of this research are the first step towards 3D reconstruction from original CT scan data. The manufacturing of Medical Rapid Prototyping will serve as the initial clinical study. The true advantages of 3D reconstruction in SolidWorks have yet to be determined through long-term study and clinic application. It is my opinion that 3D reconstruction in SolidWorks can provide STL format data for Medical Rapid Prototyping manufacturing to help plan implant surgeries because SolidWorks can export STL files for direct reading by Rapid Prototyping machine. The digital binary matrixes which are created are chosen to lie on the real CT scan or MRI image data and to be optimal in terms of interpolation accuracy. The dimension of 3D physical model can be proved to be correct depending on whether or not the scale is 1 comparing with the real sizes of CT scan or MRI image.

This thesis has certain key features:

- It uses the color gray level to process CT scan images into the digital binary matrix using MATLAB.
- Because all data are built in a database that is outside of main memory, the data

feature can be changed depending on the real requirement. The processing time can also be reduced.

- It uses the Library Feature for 3D reconstruction in SolidWorks for increasing the running speed.
- Cube processing takes place with linear interpolation in 3D.
- Higher accuracy in rendering produces for application such as MRP manufacturing by use of loft in SolidWorks with Planar contours.
- A 3D physical model for Medical Rapid Prototyping can be created directly from SolidWorks.

8.2 Future Works

This research has the limitations:

- Cube size cannot be changed automatically according to the real requirement in Marching Cubes.
- The surface of 3D model exist the staircase and is very rough.
- Resolution of 3D model is very slow.

Further refinements will be necessary in the follows respects:

- 1) To enhance the quality of 3D models, the cube size should be automatically changeable, depending on the organ to be modeled, by using the driving dimension.
- 2) 3D models that are created by using Marching Cubes algorithm need to be smoothed further using rendering technologies because the result of 3D models are rough.
- 3) Because size of 3D models are large, resolution of 3D models should be improved according to the interpolation that is most closely approximates the exact value, such as bilinear or trilinear interpolation.

Reference

- [1] Ashley, S., “Rapid Prototyping is Coming of Age”, *Mechanical Engineering*, 1995, 117(7): 62-68.
- [2] McGurk, M., Potamianos, P., Amis, A. A., and Goodger, N. M., “Rapid Prototyping Techniques for Anatomical Modeling in Medicine”, *Annals of the Royal College of Surgeons of England*, 1997, 79: 169-174.
- [3] Zonneveld, F. W., “Progress in Clinical Radiology: Decade of Clinical Three-Dimensional Imaging: A Review, Part III, Image Analysis and Interaction, Display Options, and Physical Models”, *Investigative Radiology*, 1994, 29(7): 716-725.
- [4] R. M. Koch, et. al. “A Framework for Facial Surgery Simulation”, *ETH Zurich, CS Technical Report #326, Institute of Scientific Computing*, June 18, 1999
- [5] Jianping Li, Pan Agthoklis, “a Case Study of Isosurface Generation in 3D Visualization”, *IEEE Pac. Rim’93*: 622-625
- [6] Musik Kwon, Chang-Su Kim, Kyoung Mu Lee, and Sang Uk Lee, “Progressive encoding of binary voxel models using pyramidal decomposition”, *J. Vis. Commun. Image R.*, 15 (2004):44-64
- [7] Carsten Maple, “Geometric Designing and Space Planning Using the Marching Squares and Marching Cube Algorithms”, *Proceedings of the 2003 International Conference on Geometric Modeling and Graphics (GMAG’03)*
- [8] Gregory M. Nielson, Adam Huang, Steve Sylvester, “Approximating Normals for Marching Cubes applied to Locally Supported Isosurfaces”, *IEEE Visualization*, 2002, Oct, 27-Nov, 1: 459-466
- [9] Fabien Vivodtzev, Lars Linsen, Georges-Pierre Bonneau, Bernd Hamann, Kenneth I. Joy, and Bruno A. Olshausen. “Hierarchical Isosurface Segmentation Based on Discrete Curvature”, *Proceedings of the Symposium on Data Visualization*, May 2003, 249-303
- [10] Armin Kanitsar, Dominik Fleschmann, Rainer Wegenkittl, and Petr Felkel. “CPR-

Curved Planar Reformation”, *Proceedings of the Conference on Visualization*, October 2002

- [11] Yongjie Zhang, Chandrajit Bajaj, Bong-Soo Sohn. “Adaptive and Quality 3D Meshing from Imaging Data”, *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, June 2003: 286-291
- [12] Kunio Nakamura. “Manual Registration of 3D Rendered Magnetic Resonance Images by Marching Cube Surface Construction Algorithm”, The Cleveland Clinic Foundation and Case Western Reserve University
- [13] Stefan Futterling, Reinhard Klein, Wolfgang Straber, and Heiner Weber. “Automated Finite Element Modeling of a Human Mandible with Dental Implants”, University Tübingen, Germany
- [14] Erwin Keeve, et. al., “Deformable Modeling of Facial Tissue for Craniofacial Surgery Simulation”, *Computer Aided Surgery*, 1998:34-45
- [15] McGurk, M., Aimis, A. A., Potamianos, P., and Goodger, N. M., “Rapid Prototyping Techniques For Anatomical Modeling In Medical”, *Ann. Royal Coll. Surgery Engl.* 1997, 79: 167-174.
- [16] Ashley, S., “Rapid Prototyping For Artificial Body Parts”, *Mechanical Engineering*, May 1993, 155(5): 50-53.
- [17] Artzy, E., Frieder, G., and Herman, G. T. “The Theory, Design, Implementation and Evaluation of a Three-Dimensional Surface Detection Algorithm”, *Computer Graphics and Image Processing* 15, 1 (January 1981), 1-24.
- [18] Ye Duan, Liu Yang, Hong Qin, and Dimitris Samaras. “Shape Reconstruction from 3D and 2D Data Using PDE-Based Deformable Surfaces”.
- [19] Natraj Lyer, et al. “A Reconfigurable 3D Engineering Shape Search System Part I: Shape Representation”, *Proceedings of DETC’03 ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference* Chicago, Illinois, USA, September 2-6, 2003, DETC2003/CIE-48180.

- [20] E. Keppel, “Approximating Complex Surfaces by Triangulation of Contour Lines”, *IBM J. Research and Development*, 1975, 19: 2-11
- [21] H. Funchs, Z. M. Kedem, and S. P. Uselton, “Optimal Surface Reconstruction from Planar Contours”, *Communications of the ACM*, 1977, 20: 693-702
- [22] H. Christiansen and T. Sederberg, “Conversion of Complex Contour Line Definitions into Polygonal Element Mosaics”, *Computer Graphics*, 1978, 13: 187-192
- [23] F. Preparata and M. Shamos, “Computational Geometry”, *An Introduction*. Springer Verlag, 1985
- [24] J. Boissonnat, “Shape Reconstruction from Planar Cross Sections”, *Computer Vision, Graphics and Image Processing*, 1988, 44: 1-29
- [25] Tamal K. Dey, Joachim Giesen, and James Hudson. Delaunay Based Shape Reconstruction from Large Data. *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, October 2001
- [26] William E. Lorensen, Harvey E. Cline. “Marching Cubes: a High Resolution 3D Surface Construction Algorithm”, *Computer Graphics*, July 1987, v21, Number 4: 163-169
- [27] Raj Shekhar, Elias Fayyad, Roni Yagel, J. Fredrick Cornhill, “Octree-Based Deximation of Marching Cubes Surfaces”, *IEEE*, 1996: 335-499
- [28] Chin-Feng Lin, Don-Lin Yang, and Yeh-Ching Chung, “a Marching Voxels Method for Surface Rendering of Volume Data”, *IEEE*, 2001: 306-313
- [29] David C. Banks, Stephen Linton, “Counting Cases in Marchign Cubes: Toward a Generic Algorithm for Producing Substitopes”, *IEE Visualization*, 2003: 51-58
- [30] Denis Dion Jr., Denis Laurendeau, Louis Borgeat, “3D Triangular Mesh Matching Through a Sequence of Registered 2D and 3D Images”, *IEEE*, 2000: 977-980
- [31] Adriano Lopes and Ken Brodlie. “Improving the Robustness and Accuracy of the Marching Cubes Algorithm for Isosurfacing”, *IEEE Transactions on Visualization*

and Computer Graphics, January- March 2003 vol. 9, No. 1; 16-29

- [32] Dennis J. Bouvier. “Double-Time Cubes: a Fast 3D Surface Construction Algorithm for Volume Visualization”, University of Arkansas
- [33] P. Lacroute and M. Levoy. “Fast Volume Rendering Using a Shear-warp Factorization of the Viewing Transform”, *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH'94), Orlando, 1994, 451-459
- [34] Shin Yi Yen. “Fast Sliding Thin Slab Volume Visualization” *Proceedings of the 1996 Symposium on Volume Visualization*, October 1996, 79-86
- [35] Frank Dachille, Kevin Kreeger, Baoquan Chen, Ingmar Bitter and Arie Kaufman. “High-Quality Volume Rendering Using Texture Mapping Hardware” *Workshop on Graphics Hardware Lisbon Portugal*, 1998, 1-58113-097-x/98/8: 69-76
- [36] Allen Van Gelder and Kwansik Kim. “Direct Volume Rendering with Shading via Three-Dimensional Textures”, *1996 IEEE*, 0-7803-3708-5/96: 23-98
- [37] Benjamain Mora, Jean-Pierre Jessel, and Rene Caubet. “A New Object-Order Ray-Casting Algorithm”, University Paul Sabatier, 31062 Toulouse, France
- [38] Farrell, E. J. “Color Display and Interactive Interpretation of Three-Dimensional Data”, *IBM J. Res. Develop*, July 1983, 27, 4: 356-366
- [39] Hohne, K. H. and Bernstein, R. Shading 3D Images from CT Using Gray-Level Gradients. *IEEE Trans. On Medical Imaging*, March 1986, M 1-5: 45-47
- [40] J. Wang, V. M. Gharpuray, and R. L. Dooley. “Automated 3D Reconstruction of 2D Medical Images: Application to Biomechical Modeling”, Presented at the *Twenty-First Annual Meeting of the American Society of Biomechanics*
- [41] MGP Cavalcanti, A Ruprech, and MW Vannier. “3D Volume Rendering Using Multislice CT for Dental Implant”, *Dentomaxillofacial Radiology*, 2002, 31: 218-228
- [42] Pommert, J.K. et al., “Three Dimensional Imaging In Medicine: Method And Applications”, *Computer Integrated Surgery* (Eds R. H. Taylor et al.), 1996, Ch. 9:

155-174.

- [43] Udupa, J. K., and Goncalves, R. J., "Imaging Transforms For Volume Visualisation", *Computer Integrated Surgery* (Eds R. H. Taylor et al.), 1996, Ch. 3: 33-57.
- [44] Gonzalez Woods, "Digital Image Processing", *Addison Wesley*, 2002
- [45] M. Sonka, "Image Processing, Analysis and Machine Vision", *Ed. PWS Publishing*, 1999
- [46] Ritter L., Lievin M., Sader R., Zeilhofer H-F., Keeve E., "Fast Generation of 3D Bone Model for Craniofacial Surgical Planning: an Interactive Approach", *CARS/Springer*, 2002:1-6
- [47] G22.3033-002: Topics in Computer Graphics: Lecture #10, Geometric Modeling, *New York University*
- [48] Gokul Varadhan, et, al. "Topology Preserving Surface Extraction Using Adaptive Subdivision", *Eurographics Symposium on Geometry Processing*, 2004
- [49] Fadel, G.M., and Kirschman, C., "Accuracy Issues in CAD to RP Translations," *Rapid Prototyping Journal*, 1996, 2(2), pp.4-17.
- [50] Dolenc, A., and Makela, I., "Rapid Prototyping from a Computer Scientist's Point of View," *Rapid Prototyping Journal*, 1996, 2(2), PP.18-25
- [51] Taha, F. and Wouters, K., "First French Workshop on Medical AppliCATION OF Prototyping Techniques", *Phidias*, 1998, December: No. 1, 2-3.
- [52] Thoms, S. W., "Optimising Composite Part Design and Manufacturing Using Stereolithography Tooling, E-Systems", *Third International Conference on Rapid Prototyping*, 1992, June 7-10: 267-279.
- [53] Jacobs, P. M., "Research Developments in Rapid Prototyping", *Proc inst. Mech. Engrs* , 1995, Vol. 209: 261-266.
- [54] Sara McMains, et, al. "The Evolution of a Layered Manufacturing Interchange Format", *Proceedings of DETC'02 ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference* Chicago,

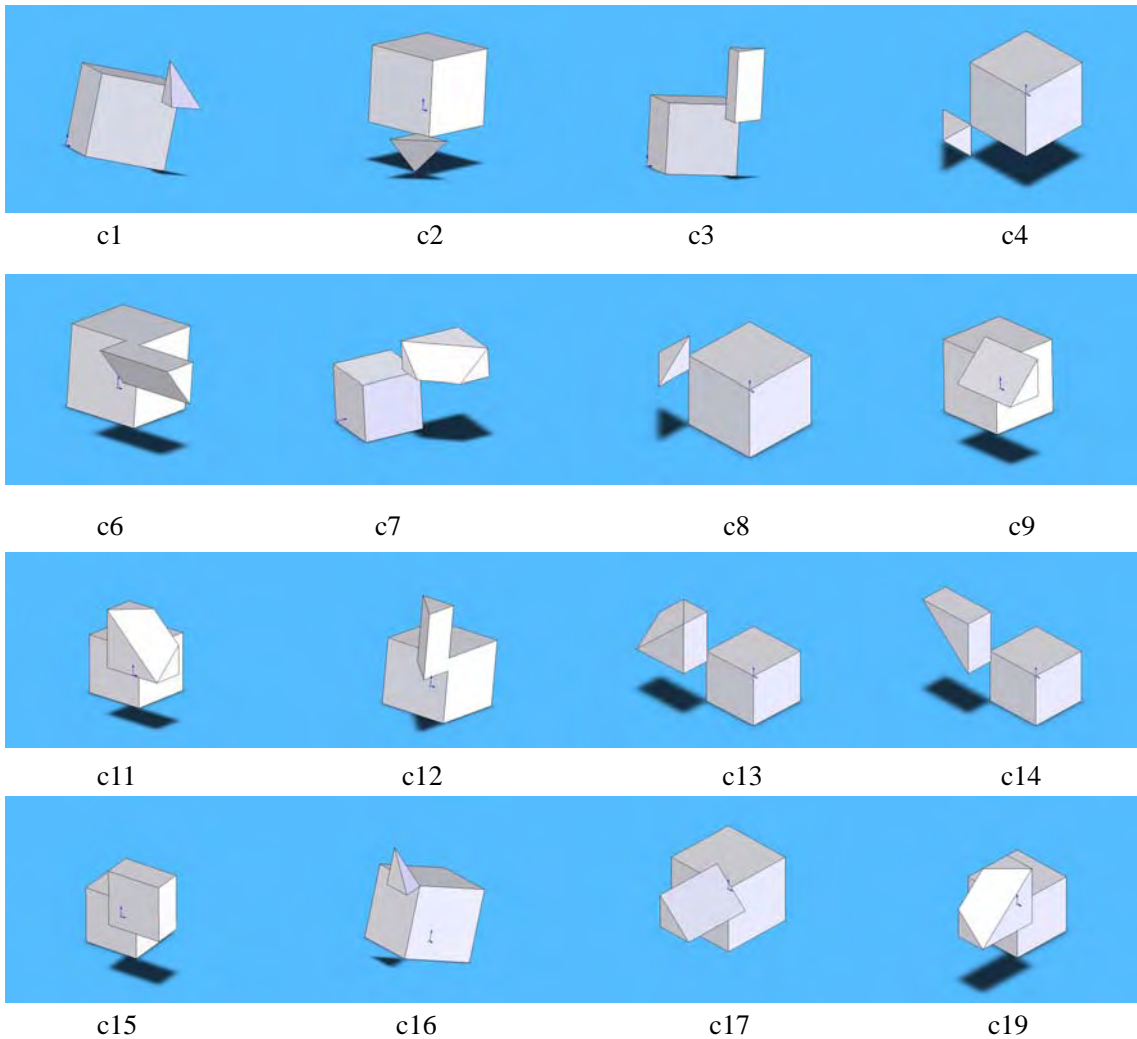
Illinois, USA, September 29, 2002, Montreal, Quebec, Canada, DETC2003/DAC - 34136

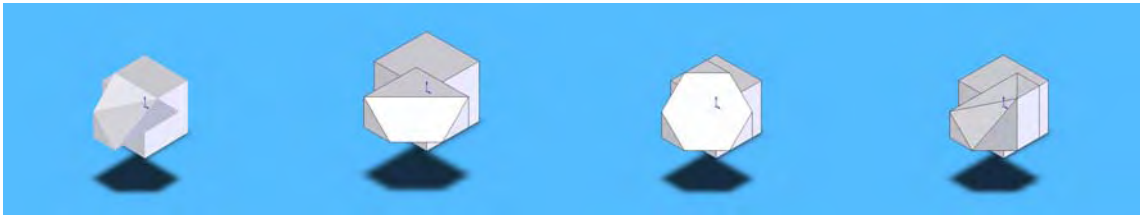
- [55] Stratasys, Inc., "Stratasys Web Site," <http://www.stratasys.com>, Accessed September 15, 2003, Stratasys, Inc., Eden Prairie, MN.
- [56] Kulkarni, R.B., and Manners, C. R., "Stereolithographic Process of Making a Three-Dimensional Object," United States Patent Number 6558606, 2003, United States Patent Office, Alexandria, VA.
- [57] 3D Systems, Inc., "3D Systems Web Site," [http:// www.3dsystems.com](http://www.3dsystems.com), Accessed September 15, 2003, 3D Systems, Inc., Valencia, CA.
- [58] Huffman, R. W., "3-D Printing: Solid Model Fabrication with at Touch of a Button," *The Technology Teacher*, November, 2001, pp.18-22
- [59] Bredt, J. F., Anderson, T. C., and Russel, D. B., "Three Dimensional Material System and Method", United States Patent Number 6610429, 2003, United States Patent Office, Alexandria, VA.
- [60] Z Corporation, "Z Corporation Web Site," [http:// www.zcorp.com](http://www.zcorp.com), Accessed September 15, 2003, Z Corporation, Burlington, MA.
- [61] Yasser A. Hosni. "Advances in the Mechanical Application of Rapid Prototyping", University of Central Florida.
- [62] P. Hering, E. Keeve, H. Seitz, C. Tille, "Ergonomics in Communications", *Computer Aided Surgery*, 2_3_1-9

Appendix A: Feature Library (File name: *-cube li.sldlfp)

Feature elements are the smallest elements that are used to reconstruct 3D models. Feature Library is in charge of the 256 triangulated cube configurations that are saved in Feature Library of SolidWorks. They are matched by **Cube Index** to join the 3D reconstruction.

To create a feature library, a *base feature*, which is either the first solid feature, is first created. The triangulated cube configuration features included in the library feature on the base are then produced separately. Feature Library file has the *.sldlfp extension.





c23

c25

c27

c29

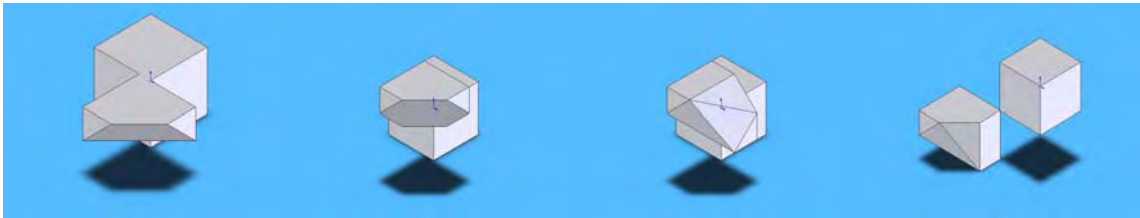


c31

c32

c34

c35

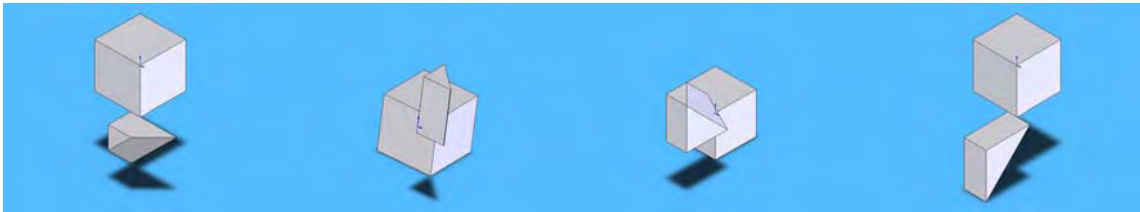


c38

c39

c43

c46

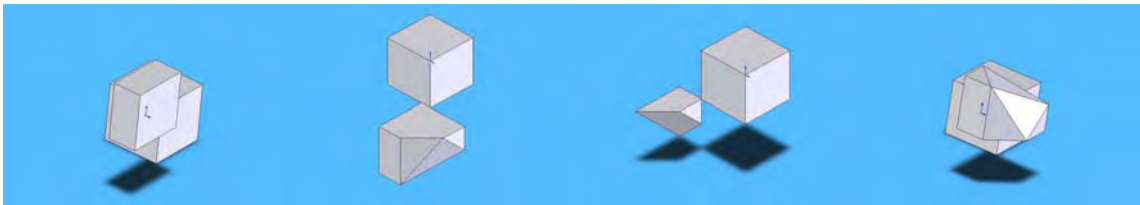


c47

c48

c49

c50

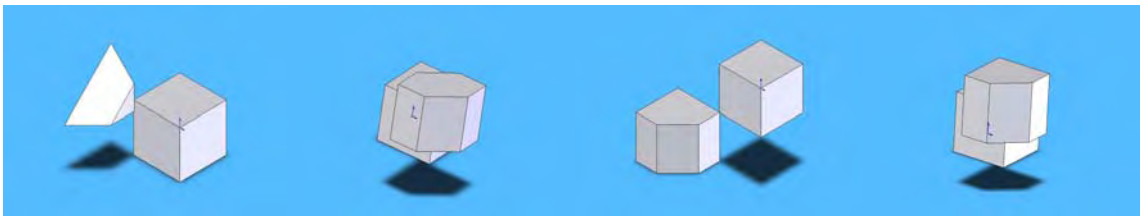


c51

c54

c55

c57

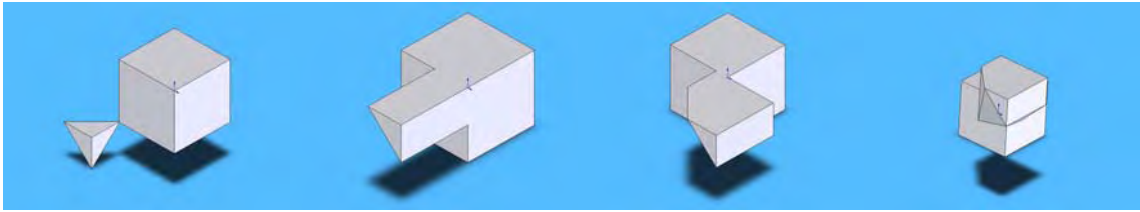


c59

c61

c62

c63

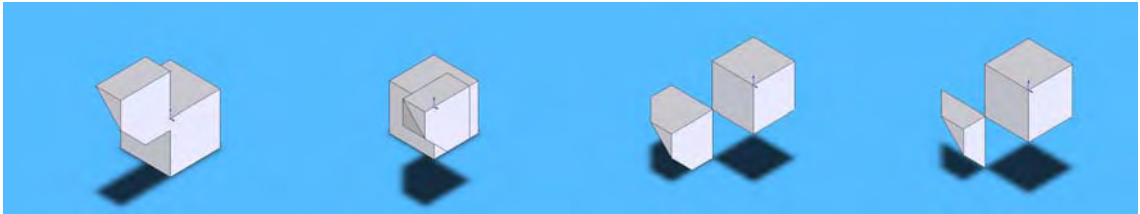


c64

c68

c70

c71



c76

c77

c78

c79

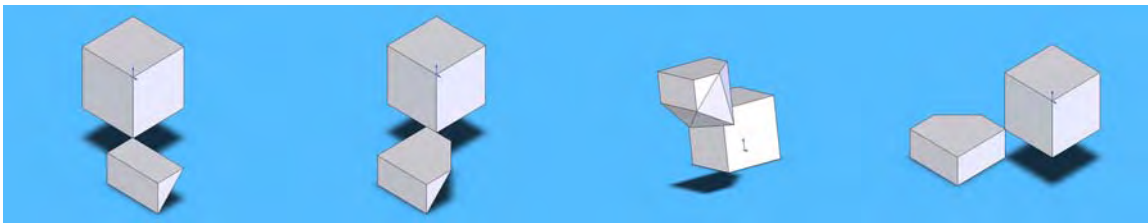


c85

c87

c93

c95

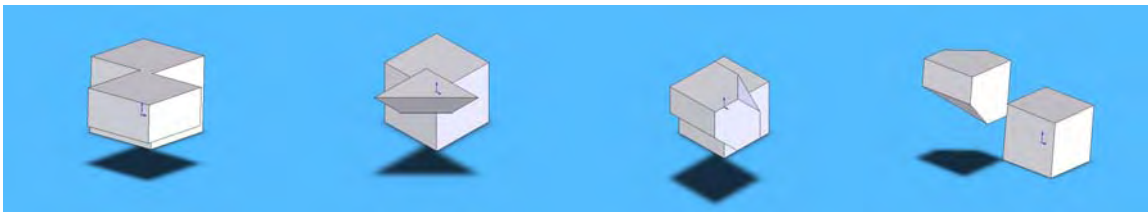


c96

c98

c99

c100

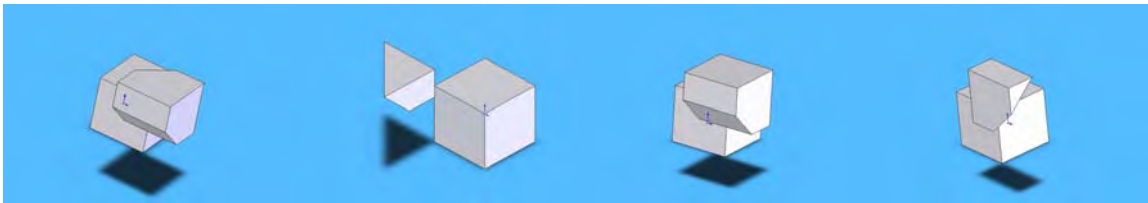


c102

c103

c107

c108



c109

c110

c111

c112

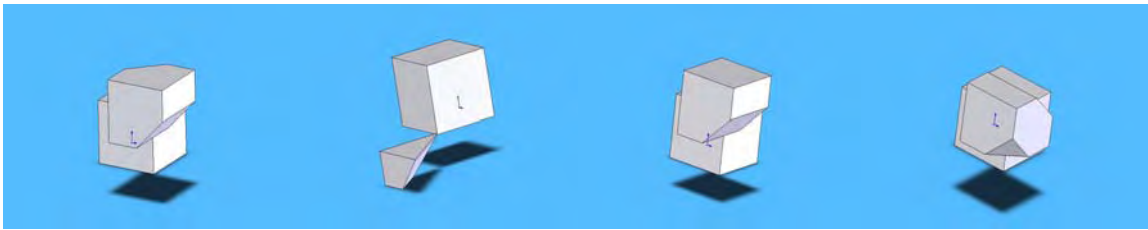


c113

c114

c115

c116

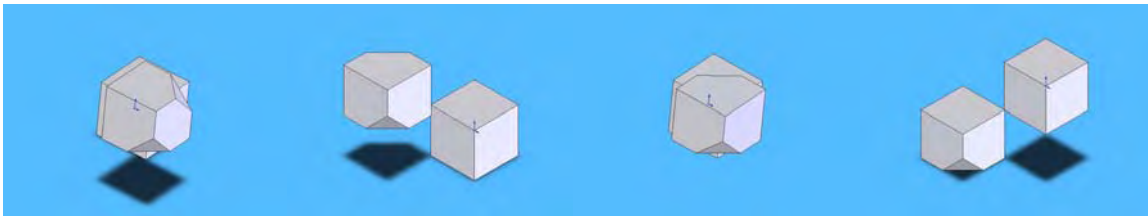


c117

c118

c119

c121



c123

c124

c125

c126

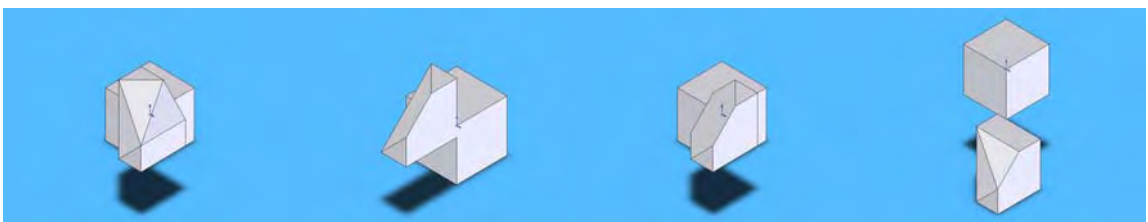


c127

c128

c136

c137



c139

c140

c141

c142

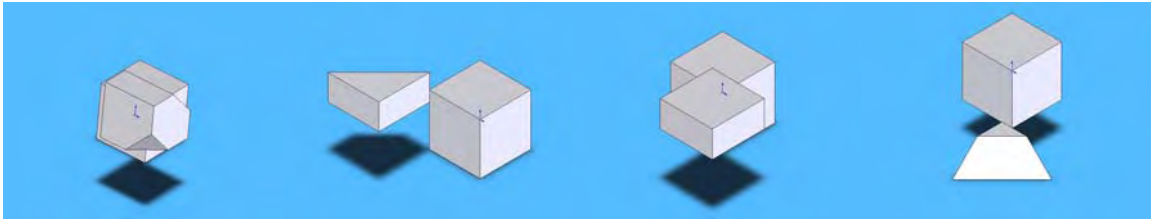


c143

c144

c145

c147

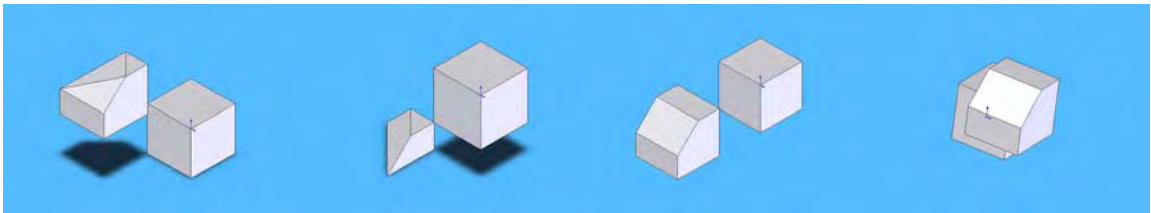


c151

c152

c153

c155

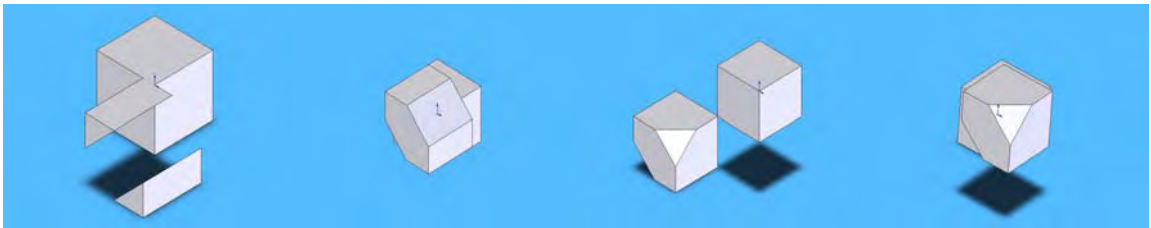


c156

c157

c158

c159



c170

c171

c174

c175



c176

c177

c178

c179



c182

c183

c184

c185

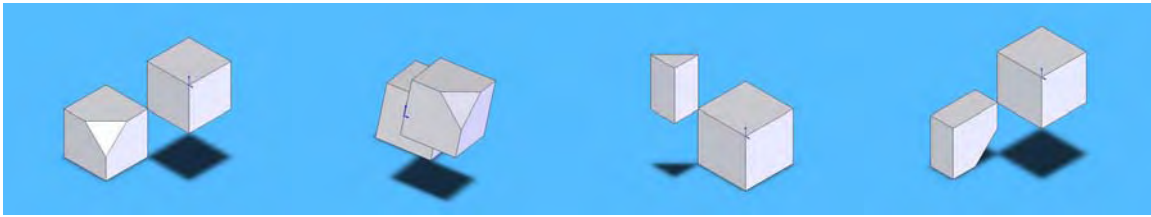


c186

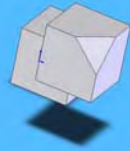
c187

c188

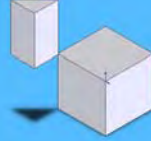
c189



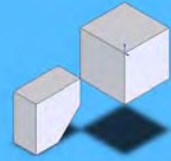
c190



c191



c192



c196



c198



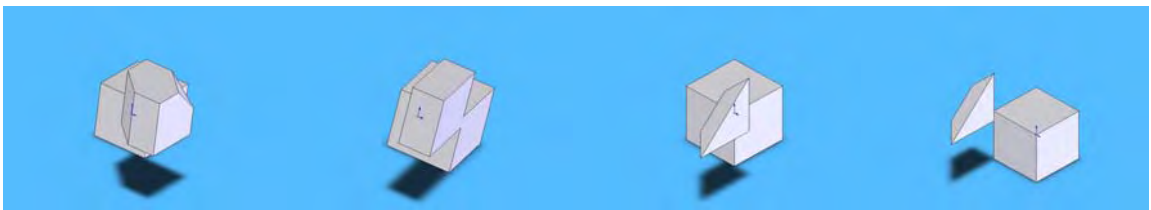
c199



c200



c201



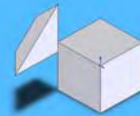
c203



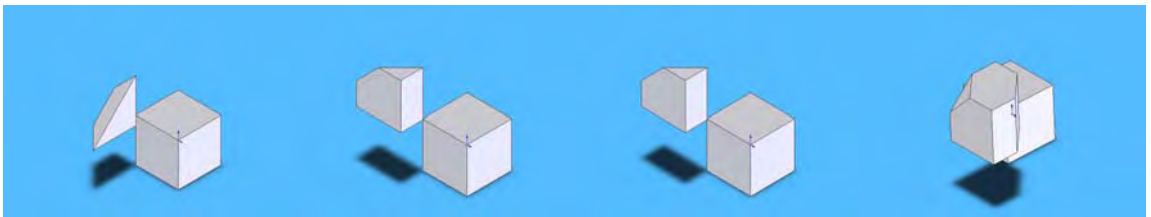
c204



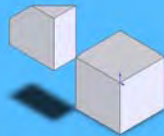
c205



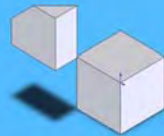
c206



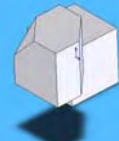
c207



c208



c209



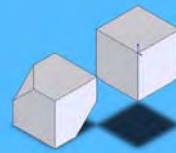
c211



c212



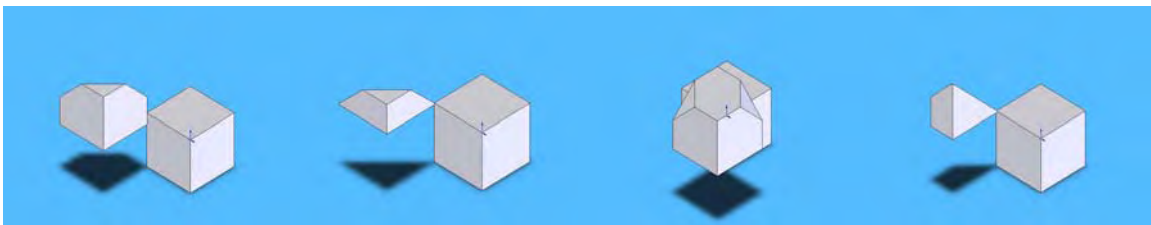
c213



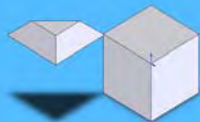
c214



c215



c216



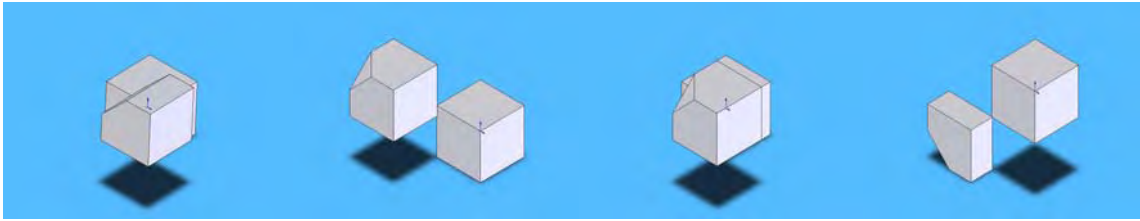
c217



c219



c220

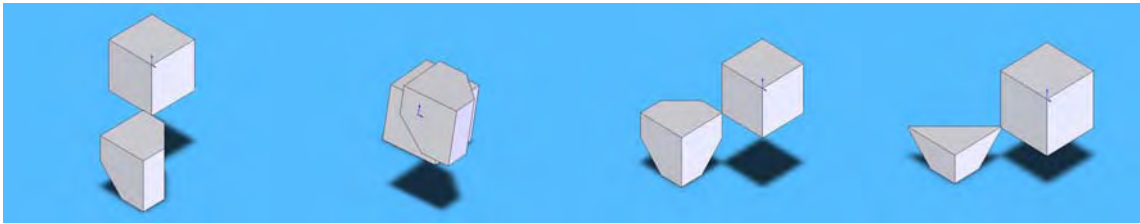


c221

c222

c223

c224

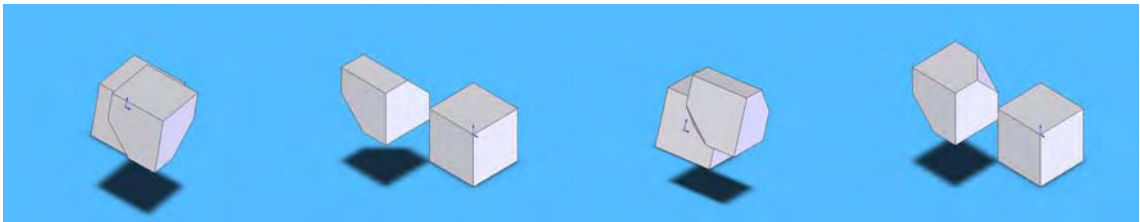


c226

c227

c228

c230

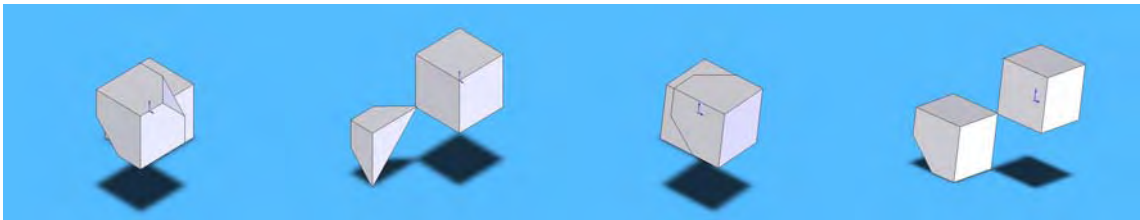


c231

c232

c233

c234



c235

c236

c237

c238



c239

c240

c241

c243

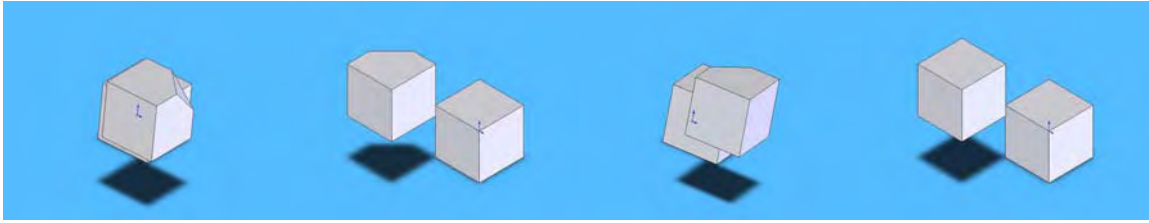


c245

c246

c249

c250

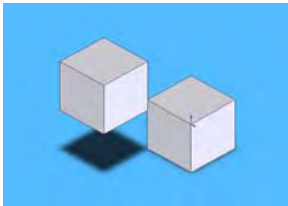


c251

c252

c253

c254



c255

Appendix B: Programming Code

File name: 3D model

Option Explicit

Private Type POINTAPI

X As Double

Y As Double

Z As Double

End Type

'Definitions in programming

Private Sub Command_Click()

Dim i As Integer, j As Integer, k As Integer

Dim II As Integer

Static MatrixF1(1 To 21, 1 To 21, 0 To 20) As Double

Static MatrixF2(1 To 21, 1 To 21, 0 To 21) As Double

Static triLFCTable(0 To 255) As Variant

Static partlistTable(0 To 255) As Variant

Dim grid(0 To 7) As Variant

Dim cubeindex0 As Integer

Dim cubeindex1 As Integer

Dim cubeindex2 As Integer

Dim cubeindex3 As Integer

Dim cubeindex4 As Integer

Dim cubeindex5 As Integer

Dim cubeindex6 As Integer

Dim cubeindex7 As Integer

Dim cubeindex00 As Integer

Dim cubeindex As Integer

Dim i1 As Integer

Dim swApp As Object

Dim filename As String

Dim filename1 As String

Dim filename2 As String

Dim filenameee(1 To 20) As String

Dim partlistname As String

Dim partdrawname As String

Dim A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12 As Long

Dim Part As Object

Dim V(0 To 7) As POINTAPI

Dim VV(0 To 11) As POINTAPI

Dim RR(1 To 2) As POINTAPI

Dim selMgr As Object

Dim Feature As Object

Dim theDispDimen As Object

Dim theDimen As Object

Dim thevalue As Variant

Dim pp As Long

Dim kk As Long

Dim planeName, newPlaneName, newPlaneName0 As String

Dim planeFeature As Object

Dim planeCount As Integer

Dim partdrawname0 As Variant

Dim partname(0 To 255) As String

Dim Sketch As Variant

Dim size As Long

Dim retval As Long

Set swApp = CreateObject("SldWorks.Application")

Set Part = swApp.Newpart()

Set selMgr = Part.SelectionManager

'size = 0

'retval = Part.SetFeatureManagerWidth(size)

'read CT file to produce MatrixF1, MatrixF2...

filename = "C:\Documents and Settings\jiman\My Documents\Solidapplication\LFC\filelist.txt"

Open filename For Input As #1

Do While Not EOF(1)

For i = 1 To 20

Line Input #1, AA

filenamee(i) = AA

Next i

Loop

Close #1

For II = 1 To 19

filename1 = filenamee(II)

Open filename1 For Input As #2

Do While Not EOF(2)

For i = 1 To 20

For j = 1 To 20

Input #2, A

MatrixF1(i, j, II - 1) = A

Next j

Next i

Loop

Close #2

filename2 = filenamee(II + 1)

Open filename2 For Input As #3

Do While Not EOF(3)

For i = 1 To 20

For j = 1 To 20

Input #3, A

MatrixF2(i, j, II) = A

'swApp.SendMsgToUser I

'swApp.SendMsgToUser J

Next j

Next i

Loop

Close #3

Next II

' Build Feature Library

partdrawname0 = "C:\Documents and Settings\jiman\My Documents\Solidapplication\partlist.txt"

Open partdrawname0 For Input As #1

Do While Not EOF(1)

For i = 0 To 255

Line Input #1, AA

partname(i) = AA

Next i

Loop

Close #1

Part.SketchRectangle -0.003, 0, 0, 0, 0.003, 0, 1

Part.FeatureExtrusion 1, 0, 0, 0, 0, 0.003, 0, 0, 0, 0, 0.01745329251994, 0.01745329251994, 0, 0

Part.AndSelectByID "", "SKECTCHPOINT", 0, 0.003, 0.003

Part.AndSelectByID "", "SKECTCHPOINT", -0.003, 0.003, 0.003

Part.AndSelectByID "", "SKECTCHPOINT", -0.003, 0, 0.003

Part.CreatePlaneThru3Points

Part.BlankRefGeom

kk = 2

pp = 0

newPlaneName = "Plane1"

Sketch = "Sketch2"

'read vetex of cube to setup grid0=..., grid1=..., grid7=...

For k = 0 To 14

Part.SelectByID newPlaneName, "PLANE", 0, 0, 0

Part.CreatePlaneAtOffset 0.0005, 0

Part.BlankRefGeom

newPlaneName0 = "Plane" & kk

kk = kk + 1

For i = 1 To 19

For j = 1 To 19

```

grid(0) = MatrixF1(i, j, k)
grid(1) = MatrixF1(i, j + 1, k)
grid(2) = MatrixF1(i + 1, j + 1, k)
grid(3) = MatrixF1(i + 1, j, k)
grid(4) = MatrixF2(i, j, k + 1)
grid(5) = MatrixF2(i, j + 1, k + 1)
grid(6) = MatrixF2(i + 1, j + 1, k + 1)
grid(7) = MatrixF2(i + 1, j, k + 1)

```

```

V(0).X = i: V(0).Y = j: V(0).Z = k
V(1).X = i: V(1).Y = j + 0.01: V(1).Z = k
V(2).X = i + 0.01: V(2).Y = j + 0.01: V(2).Z = k
V(3).X = i + 0.01: V(3).Y = j: V(3).Z = k
V(4).X = i: V(4).Y = j: V(4).Z = k + 0.01
V(5).X = i: V(5).Y = j + 0.01: V(5).Z = k + 0.01
V(6).X = i + 0.01: V(6).Y = j + 0.01: V(6).Z = k + 0.01
V(7).X = i + 0.01: V(7).Y = j: V(7).Z = k + 0.01

```

'create cubeindex number

```

If grid(0) = 1 Then
cubeindex0 = 1
Else: cubeindex0 = 0
End If

```

```

If grid(1) = 1 Then
cubeindex1 = 2
Else: cubeindex1 = 0
End If

```

```

If grid(2) = 1 Then
cubeindex2 = 4
Else: cubeindex2 = 0
End If

```

```

If grid(3) = 1 Then
cubeindex3 = 8

```

Else: cubeindex3 = 0

End If

If grid(4) = 1 Then

cubeindex4 = 16

Else: cubeindex4 = 0

End If

If grid(5) = 1 Then

cubeindex5 = 32

Else: cubeindex5 = 0

End If

If grid(6) = 1 Then

cubeindex6 = 64

Else: cubeindex6 = 0

End If

If grid(7) = 1 Then

cubeindex7 = 128

Else: cubeindex7 = 0

End If

cubeindex00 = 0

*cubeindex = cubeindex0 + cubeindex1 + cubeindex2 + cubeindex3 + cubeindex4 + cubeindex5 +
cubeindex6 + cubeindex7 + cubeindex00*

filename = "C:\Documents and Settings\jiman\My Documents\Solidapplication\LFCT\typelist.txt"

Open filename For Input As #1 ' Open file for input.

N = 0

Do While Not EOF(1) ' Loop until end of file.

Input #1, A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12 ' Read data into two variables.

triLFCTablee1(0) = A0

triLFCTablee1(1) = A1

triLFCTablee1(2) = A2

triLFCTablee1(3) = A3

triLFCTablee1(4) = A4

triLFCTablee1(5) = A5

```

triLFCTablee1(6) = A6
triLFCTablee1(7) = A7
triLFCTablee1(8) = A8
triLFCTablee1(9) = A9
triLFCTablee1(10) = A10
triLFCTablee1(11) = A11
triLFCTablee1(12) = A12

triLFCTable(N) = triLFCTablee1()
N = N + 1
Loop
Close #1 ' Close file.

'create triangulated cube configuration
If cubeindex <> 0 Then
    i1 = 0
    Do While i1 <= 9

        If triLFCTable(cubeindex)(i1) <> -1 And triLFCTable(cubeindex)(i1) = -66 Then
            partdrawname = partname(cubeindex)

            Part.SelectByID newPlaneName, "PLANE", 0, 0, 0
            Part.InsertLibraryFeature (partdrawname)
            Part.DissolveLibraryFeature
            Part.SelectByID Sketch, "SKETCH", 0, 0, 0

            Set Feature = Part.SelectionManager.GetSelectedObject3(1)
            If (Feature Is Nothing) Then
                Exit Sub
            End If
            Set theDispDimen = Feature.GetFirstDisplayDimension

            While (Not theDispDimen Is Nothing)
                Set theDimen = theDispDimen.GetDimension

                If theDimen.Name = "D1" Then
                    thevalue = theDimen.Value
                    If thevalue = 0.5 Then
                        theDimen.SetValue2 thevalue + V(0).X + 0.001, 0
                    Else
                        If thevalue = 0.01 Then

```



```

    theDimen.SetValue2 thevalue + 0.99 + V(0).X + 0.001, 0
Else
    If thevalue = 0.98 Then
        theDimen.SetValue2 thevalue - 0.48 + V(0).X + 0.001, 0
    Else
        If thevalue = 0.02 Then
            theDimen.SetValue2 thevalue + 0.48 + V(0).X + 0.001, 0
        Else
            theDimen.SetValue2 thevalue - thevalue + V(0).X + 0.001, 0
        End If
    End If
End If
End If
theDimen.Name = "DDD1"
Else
    If theDimen.Name = "D2" Then
        thevalue = theDimen.Value
        If thevalue = 0.5 Then
            theDimen.SetValue2 thevalue + V(0).Y + 0.001, 0
        Else
            If thevalue = 0.01 Then
                theDimen.SetValue2 thevalue + 0.99 + V(0).Y + 0.001, 0
            Else
                If thevalue = 0.98 Then
                    theDimen.SetValue2 thevalue - 0.48 + V(0).Y + 0.001, 0
                Else
                    If thevalue = 0.02 Then
                        theDimen.SetValue2 thevalue + 0.48 + V(0).Y + 0.001, 0
                    Else
                        theDimen.SetValue2 thevalue - thevalue + V(0).Y + 0.001, 0
                    End If
                End If
            End If
        End If
    End If
theDimen.Name = "DDD2"
End If
End If
Set theDispDimen = Feature.GetNextDisplayDimension(theDispDimen)
Part.ClearSelection
Wend
Part.EditRebuild

```

```

    pp = pp + 1
    Sketch = "Sketch2" & pp
Else
End If

If triLFCTable(cubeindex)(i1) <> -1 And triLFCTable(cubeindex)(i1) = -7 Then
    partdrawname = partname(cubeindex)
    Part.SelectByID newPlaneName0, "PLANE", 0, 0, 0
    Part.InsertLibraryFeature (partdrawname)
    Part.DissolveLibraryFeature
    Part.SelectByID Sketch, "SKETCH", 0, 0, 0

    Set Feature = Part.SelectionManager.GetSelectedObject3(1)
    If (Feature Is Nothing) Then
        Exit Sub
    End If
    Set theDispDimen = Feature.GetFirstDisplayDimension

    While (Not theDispDimen Is Nothing)
        Set theDimen = theDispDimen.GetDimension

        If theDimen.Name = "D1" Then
            thevalue = theDimen.Value
            If thevalue = 0.5 Then
                theDimen.SetValue2 thevalue + V(0).X + 0.001, 0
            Else
                If thevalue = 0.01 Then
                    theDimen.SetValue2 thevalue + 0.99 + V(0).X + 0.001, 0
                Else
                    If thevalue = 0.98 Then
                        theDimen.SetValue2 thevalue - 0.48 + V(0).X + 0.001, 0
                    Else
                        If thevalue = 0.02 Then
                            theDimen.SetValue2 thevalue + 0.48 + V(0).X + 0.001, 0
                        Else
                            theDimen.SetValue2 thevalue - thevalue + V(0).X + 0.001, 0
                        End If
                    End If
                End If
            End If
            theDimen.Name = "DDD1"

```

```

Else
If theDimen.Name = "D2" Then
thevalue = theDimen.Value
If thevalue = 0.5 Then
theDimen.SetValue2 thevalue + V(0).Y + 0.001, 0
Else
If thevalue = 0.01 Then
theDimen.SetValue2 thevalue + 0.99 + V(0).Y + 0.001, 0
Else
If thevalue = 0.98 Then
theDimen.SetValue2 thevalue - 0.48 + V(0).Y + 0.001, 0
Else
If thevalue = 0.02 Then
theDimen.SetValue2 thevalue + 0.48 + V(0).Y + 0.001, 0
Else
theDimen.SetValue2 thevalue - thevalue + V(0).Y + 0.001, 0
End If
End If
End If
End If
theDimen.Name = "DDD2"
End If
End If
Set theDispDimen = Feature.GetNextDisplayDimension(theDispDimen)
Part.ClearSelection
Wend
Part.EditRebuild
pp = pp + 1
Sketch = "Sketch2" & pp

Else
End If

If triLFCTable(cubeindex)(i1) <> -1 And triLFCTable(cubeindex)(i1) = -8 Then
i1 = i1 + 1
cubeindex1 = triLFCTable(cubeindex)(i1)
i1 = i1 + 1

```

```

If triLFCTable(cubeindex)(i1) = -66 Then

    partdrawname = partname(cubeindex1)

    Part.SelectByID newPlaneName, "PLANE", 0, 0, 0
    Part.InsertLibraryFeature (partdrawname)
    Part.DissolveLibraryFeature
    Part.SelectByID Sketch, "SKETCH", 0, 0, 0

    Set Feature = Part.SelectionManager.GetSelectedObject3(1)
    If (Feature Is Nothing) Then
        Exit Sub
    End If
    Set theDispDimen = Feature.GetFirstDisplayDimension

    While (Not theDispDimen Is Nothing)
        Set theDimen = theDispDimen.GetDimension

        If theDimen.Name = "D1" Then
            thevalue = theDimen.Value
            If thevalue = 0.5 Then
                theDimen.SetValue2 thevalue + V(0).X + 0.001, 0
            Else
                If thevalue = 0.01 Then
                    theDimen.SetValue2 thevalue + 0.99 + V(0).X + 0.001, 0
                Else
                    If thevalue = 0.98 Then
                        theDimen.SetValue2 thevalue - 0.48 + V(0).X + 0.001, 0
                    Else
                        If thevalue = 0.02 Then
                            theDimen.SetValue2 thevalue + 0.48 + V(0).X + 0.001, 0
                        Else
                            theDimen.SetValue2 thevalue - thevalue + V(0).X + 0.001, 0
                        End If
                    End If
                End If
            End If
            theDimen.Name = "DDD1"
        Else
            If theDimen.Name = "D2" Then
                thevalue = theDimen.Value
            End If
        End If
    End While

```

```

If thevalue = 0.5 Then
    theDimen.SetValue2 thevalue + V(0).Y + 0.001, 0
Else
    If thevalue = 0.01 Then
        theDimen.SetValue2 thevalue + 0.99 + V(0).Y + 0.001, 0
    Else
        If thevalue = 0.98 Then
            theDimen.SetValue2 thevalue - 0.48 + V(0).Y + 0.001, 0
        Else
            If thevalue = 0.02 Then
                theDimen.SetValue2 thevalue + 0.48 + V(0).Y + 0.001, 0
            Else
                theDimen.SetValue2 thevalue - thevalue + V(0).Y + 0.001, 0
            End If
        End If
    End If
End If
theDimen.Name = "DDD2"
End If
End If
Set theDispDimen = Feature.GetNextDisplayDimension(theDispDimen)
Part.ClearSelection
Wend
Part.EditRebuild
pp = pp + 1
Sketch = "Sketch2" & pp
Else
End If

If triLFCTable(cubeindex)(i1) = -7 Then

    partdrawname = partname(cubeindex1)
    Part.SelectByID newPlaneName0, "PLANE", 0, 0, 0
    Part.InsertLibraryFeature (partdrawname)
    Part.DissolveLibraryFeature
    Part.SelectByID Sketch, "SKETCH", 0, 0, 0

    Set Feature = Part.SelectionManager.GetSelectedObject3(1)
    If (Feature Is Nothing) Then
        Exit Sub
    End If

```

Set theDispDimen = Feature.GetFirstDisplayDimension

While (Not theDispDimen Is Nothing)

Set theDimen = theDispDimen.GetDimension

If theDimen.Name = "D1" Then

thevalue = theDimen.Value

If thevalue = 0.5 Then

theDimen.SetValue2 thevalue + V(0).X + 0.001, 0

Else

If thevalue = 0.01 Then

theDimen.SetValue2 thevalue + 0.99 + V(0).X + 0.001, 0

Else

If thevalue = 0.98 Then

theDimen.SetValue2 thevalue - 0.48 + V(0).X + 0.001, 0

Else

If thevalue = 0.02 Then

theDimen.SetValue2 thevalue + 0.48 + V(0).X + 0.001, 0

Else

theDimen.SetValue2 thevalue - thevalue + V(0).X + 0.001, 0

End If

End If

End If

End If

theDimen.Name = "DDD1"

Else

If theDimen.Name = "D2" Then

thevalue = theDimen.Value

If thevalue = 0.5 Then

theDimen.SetValue2 thevalue + V(0).Y + 0.001, 0

Else

If thevalue = 0.01 Then

theDimen.SetValue2 thevalue + 0.99 + V(0).Y + 0.001, 0

Else

If thevalue = 0.98 Then

theDimen.SetValue2 thevalue - 0.48 + V(0).Y + 0.001, 0

Else

If thevalue = 0.02 Then

theDimen.SetValue2 thevalue + 0.48 + V(0).Y + 0.001, 0

Else

theDimen.SetValue2 thevalue - thevalue + V(0).Y + 0.001, 0

```

    End If
    End If
    End If
    End If
    theDimen.Name = "DDD2"
    End If
    End If
    Set theDispDimen = Feature.GetNextDisplayDimension(theDispDimen)
    Part.ClearSelection
    Wend
    Part.EditRebuild
    pp = pp + 1
    Sketch = "Sketch2" & pp

Else
End If

Else
End If

il = il + 1

Loop
Else
End If
Next j
Next i

Part.SelectByID newPlaneName, "PLANE", 0, 0, 0
    Set Feature = Part.SelectionManager.GetSelectedObject3(1)
    If (Feature Is Nothing) Then
        Exit Sub
    End If
    Set theDispDimen = Feature.GetFirstDisplayDimension
    While (Not theDispDimen Is Nothing)
        Set theDimen = theDispDimen.GetDimension
        Part.ClearSelection

```

```

    thevalue = theDimen.Value
    Part.ClearSelection
    theDimen.SetValue2 1, 0
    Set theDispDimen = Feature.GetNextDisplayDimension(theDispDimen)
    Part.ClearSelection
    Wend
    Part.EditRebuild

Part.SelectByID newPlaneName0, "PLANE", 0, 0, 0
    Set Feature = Part.SelectionManager.GetSelectedObject3(1)
    If (Feature Is Nothing) Then
        Exit Sub
    End If
    Set theDispDimen = Feature.GetFirstDisplayDimension
    While (Not theDispDimen Is Nothing)
        Set theDimen = theDispDimen.GetDimension
        Part.ClearSelection
        thevalue = theDimen.Value
        Part.ClearSelection
        theDimen.SetValue2 0.5, 0
        Set theDispDimen = Feature.GetNextDisplayDimension(theDispDimen)
        Part.ClearSelection
    Wend
    Part.EditRebuild

Part.SelectByID newPlaneName, "PLANE", 0, 0, 0
    Part.CreatePlaneAtOffset 0.001, 0
    Part.BlankRefGeom
    newPlaneName = "Plane" & kk
    kk = kk + 1

Next k
Part.ViewZoomtofit2
End Sub

```

Planar Contour Points Creation

```

Private Sub Command_Click()
    Dim swApp As Object
    Dim part As Object
    Dim A As Integer
    Dim B As Integer

```



```

Dim X As Integer
Dim Y As Integer

Dim filename As String
Dim V(1 To 3) As POINTAPI
Set swApp = CreateObject("SldWorks.Application")
Set part = swApp.Newpart()

filename = "E:\Solidapplication\LFC\test.txt"
Open filename For Input As #1
Y = 1
Do While Y <= 256
X = 0
B = 0
Do While X + 1 <= 256
Input #1, A
If A = 1 And B <> A Then
V(1).X = X: V(1).Y = Y
part.CreatePoint2 V(1).X, V(1).Y, 0#
B = A
Else
If A = 0 And B <> A Then
V(2).X = X - 1: V(2).Y = Y
part.CreatePoint2 V(2).X, V(2).Y, 0#
part.ViewZoomtofit2
B = A
Else
B = A
End If
End If
X = X + 1
Loop
Y = Y + 1
Loop
Close 1
part.SaveAs2 "E:\LFC\Solidapplication\test01.sldprt", 0, False, False
End Sub

```