

การแทนวัตถุประสงค์สามมิติโดยใช้การขึ้นรูปแบบปริภูมิโครงสร้างเซลล์



นาย ธีรุต จรัสสุริยงศ์

สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2548

ISBN 974-17-5453-1

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

3D OBJECT REPRESENTATION USING
CELLULAR STRUCTURED SPACE MODELING

Mr. Nat Charussuriyong



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2005

ISBN 974-17-5453-1

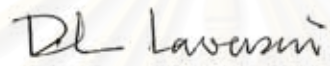
Thesis Title 3D Object Representation Using Cellular Structured Space
Modeling

By Mr. Nat Charussuriyong

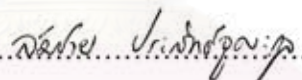
Field of Study Computer Engineering

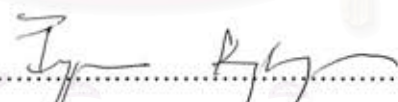
Thesis Advisor Pizzanu Kanongchaiyos, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial
Fulfillment of the Requirements for the Master's Degree



..... Dean of the Faculty of Engineering
(Professor Direk Lavansiri, Ph.D.)

THESIS COMMITTEE


..... Chairman
(Associate Professor Somchai Prasitjutrakul, Ph.D.)


..... Thesis Advisor
(Pizzanu Kanongchaiyos, Ph.D.)


..... Member
(Pinyo Jinuntuya)

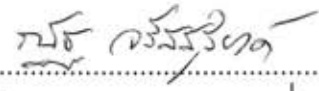


..... Member
(Atiwong Suchato, Ph.D.)

นาย ณัฐ จรัสสุริยงค์ : การแทนวัตถุสามมิติโดยใช้การขึ้นรูปแบบปริภูมิโครงสร้างเซลล์
(3D OBJECT REPRESENTATION USING CELLULAR STRUCTURED SPACE
MODELING) อาจารย์ที่ปรึกษา: ดร.พิเชษฐ คนองชัยยศ, 93 หน้า. ISBN 974-17-5453-1

การขึ้นรูปวัตถุสามมิติในงานทางด้านคอมพิวเตอร์กราฟิกส์นั้น โดยส่วนใหญ่แล้วมักจะเริ่มต้นจากปริภูมิเรขาคณิต ซึ่งวิธีการดังกล่าวนั้นสามารถแสดงผลให้ผลสังเกตเห็นได้โดยไม่มีความรู้สึกถึงข้อผิดพลาด อย่างไรก็ตามสิ่งที่ขาดหายไปคือคุณสมบัติที่สำคัญของวัตถุอย่าง ความสมนัยของวัตถุซึ่งต้องอาศัยการหาความเหมือนทางทอพอโลยีของวัตถุ อย่างไรก็ตามคุณสมบัติทางทอพอโลยีของวัตถุ ก็ยังไม่สามารถใช้พิจารณาความสมนัยของวัตถุได้อย่างมีประสิทธิภาพนัก ในงานวิจัยฉบับนี้ได้เสนอวิธีการขึ้นรูปวัตถุ โดยคงคุณสมบัติทางทอพอโลยีของวัตถุไว้ผ่านทาง การขึ้นรูปแบบปริภูมิโครงสร้างเซลล์ ซึ่งกำหนดให้เป็นปริภูมิที่อยู่ในระดับระหว่างปริภูมิเรขาคณิตและปริภูมิทอพอโลยีโดยเปรียบเทียบเหมือนการพิจารณาเป็นโครงสร้างที่ฝังในตัววัตถุ

การขึ้นรูปดังกล่าวสามารถขึ้นรูปโดยเริ่มต้นจากเซลล์ศูนย์มิติหรือจุด และอาศัยการประกอบเชิงอุปนัยในการสร้างปริภูมิโครงสร้างเซลล์ในมิติที่สูงขึ้นจนถึงมิติที่สามตามลำดับแบบจำลองสามมิติที่ได้จะถูกเก็บในรูปแบบโครงสร้างข้อมูลแบบตารางโครงสร้างเซลล์ซึ่งสามารถแปลงให้อยู่ในรูปของกราฟโครงสร้างเซลล์ได้

จากการทดลองนำแบบจำลองปริภูมิโครงสร้างเซลล์ไปใช้ในการขึ้นรูปวัตถุสามมิติพบว่าโครงสร้างของข้อมูลที่ได้มีความซับซ้อนของข้อมูลลดน้อยลงเมื่อเทียบกับรูปแบบการเก็บข้อมูลเชิงเรขาคณิตทั่วไป และเมื่อทดลองนำแบบจำลองที่นำเสนอไปเปรียบเทียบความคล้ายเพื่อหาความสมนัยของวัตถุ พบว่าสามารถจำแนกวัตถุที่มีความสมนัยได้ดีกว่าการใช้ข้อมูลเชิงเรขาคณิตหรือ ข้อมูลเชิงทอพอโลยีของแบบจำลองในการคำนวณ

ภาควิชา.... วิศวกรรมคอมพิวเตอร์.....ลายมือชื่อนิสิต..... 
สาขาวิชา...วิศวกรรมคอมพิวเตอร์.....ลายมือชื่ออาจารย์ที่ปรึกษา..... 
ปีการศึกษา2548.....

4770282921 : MAJOR COMPUTER ENGINEERING

KEY WORD: COMPUTER GRAPHICS MODELING / 3D OBJECT REPRESENTATION /
CELLULAR STRUCTURED SPACE / TOPOLOGICAL CHARACTERISTIC

NAT CHARUSSURIYONG: 3D OBJECT REPRESENTATION USING CELLULAR
STRUCTURED SPACE MODELING. THESIS ADVISOR: PIZZANU
KANONGCHAIYOS, Ph.D., 93 pp. ISBN 974-17-5453-1

Modeling methods for 3D models in computer graphics usually start from geometry level, such methods can generate result displaying well on screen with no dubiousness, but these methods lack essential properties such as shape equivalence indicated by using topological properties of the models. Unfortunately, even though these topological properties are concerned, the efficiency for considering shape equivalence is quite low. In this research, we propose a valid shape modeling using cellular structured space, an abstraction level between topological and geometry level, for representing 3D objects in computer graphics.

This proposed modeling method starts from creating set of zero dimensional cells or points and inductively compose a finite structure in each dimension to compose higher dimensional skeleton via attaching map until the maximum dimension of the models is three. The structure of this model is described by cellular structured table which can be conversed to graph structure called CSSGraph.

In experiment, 3D objects are constructed by using the proposed method and the result shows that the structure of the cellular structured space model is less complex than that of general geometrical model. Moreover, the similarity result shows that using the proposed model can categorize objects with shape equivalence more efficient than using only geometrical or topological data of the model.

Department.... Computer Engineering.... Student's signature.....

Field of study.... Computer Engineering...Advisor's signature.....

Academic year ...2005.....

Nat Charussuriyong

Pizanu

ACKNOWLEDGMENTS

I would like to thank first and foremost my thesis advisor, Pizzanu Kanongchaiyos, Ph.D. His wisdom and advice over the years have been invaluable to the development of this thesis. Without his patience, intellectual advices and invariable assistances, this thesis would not and could not have been done.

I would also like to extend a special thanks to my thesis committee, Associate Professor Somchai Prasitjutrakul, Ph.D, who provided me with algorithms and inspiration, Pinyo Jinuntuya, Atiwong Suchato, Ph.D. for their advantageous guidance and suggestions.

I wish to express my thanks to all 20th floor members especially my colleagues in computer graphic lab (CG Rangers) for their warm supports, generous helps, and truly relationships which make my experience through my graduate study filled with hilarity and happiness.

I would also like to express gratitude to the superbly professional staff, including administrator, technician and secretary at department of computer engineering, Chulalongkorn University for providing me the opportunity and facilities to complete this thesis. Special thanks are due to Lhong, for his beneficial perspective and unfailing comradeship; to Nok for his exceedingly generous and especially for his breakthrough in C++ programming.

Most of all, I would like to express my deep and abiding thanks to Meilan, my beloved, who believed in me and supports me in all my wishes, hopes and dreams. Without her encouragements, this thesis would be very hard to succeed.

Finally, I am particularly grateful to my parents for their attitude and behaviors inspired me to never give up in the pursuit of my entire dream, their embedding in me a love of learning and their firm belief that education is the best asset. Furthermore, their love, understanding and invaluable supports throughout my life and my study are impossible to repay.

TABLE OF CONTENTS

	Page
ABSTRACT (THAI)	iv
ABSTRACT (ENGLISH)	v
ACKNOWLEDGMENTS	vi
LIST OF FIGURES	ix
LIST OF TABLES	xii
CHAPTER 1 INTRODUCTION	1
1.1 Background and Statement of Problems	1
1.2 Objectives	4
1.3 Scopes of Study	4
1.4 Research Procedure	4
1.5 Expected Benefits	5
1.6 Thesis Structure	6
1.7 Publications	6
CHAPTER 2 THEORETICAL BACKGROUND AND RELATED WORKS	7
2.1 Theoretical Background	7
2.1.1 Abstraction hierarchy of invariant	7
2.1.2 Mathematical background	8
2.1.3 Object representation	10
2.1.3.1 Wire-frame models	10
2.1.3.2 Curve and surface models	11
2.1.3.3 Solid models	17
2.1.3.4 Fractal Models	21
2.1.3.5 Procedural and Grammar-based Models	22
2.1.4 Cellular Structured Space	22
2.1.5 Graph Theory	25

	Page
2.2 Related Works	29
2.2.1 Cellular Structured Space	29
2.2.2 Model Matching	31
CHAPTER 3 MODELING METHOD	36
3.1 Component of the Model	36
3.1.1 Point	36
3.1.2 Line	37
3.1.3 Surface	39
3.1.4 Volume	39
3.2 Object Construction	40
3.3 Data Structure of the Model	41
3.4 CSSGraph	44
3.5 Example	44
3.5.1 Implementation of CSS	51
3.5.2 Graph Conversion	54
3.5.3 Similarity Assessment using CSS Model	58
CHAPTER 4 IMPLEMENTATION AND RESULTS.....	59
4.1 Environment	59
4.2 Implmentation.....	59
4.3 Results	66
CHAPTER 5 CONCLUSION AND FUTURE WORK	72
5.1 Conclusion	72
5.2 Discussion.....	74
5.3 Future Work.....	77
REFERENCES	78
APPENDIX.....	83
BIOGRAPHY	93

LIST OF FIGURES

Figure	Page
1-1 A tetrahedron composed from different wires; closed and open.....	2
1-2 A boundary tetrahedron constructed from different types of plane	2
1-3 Two similar objects with different geometrical data	3
1-4 Cellular Structured Space in 0 - 3 dimension for tetrahedron	3
2-1 A simple and a complex wire-frame model	11
2-2 A data structure for wire-frame model	11
2-3 Simple polygon meshes	12
2-4 Example of pointer to vertex list representation	13
2-5 Example of pointer to edge list representation	14
2-6 Triangle fans and strips	14
2-7 A primitive cone	18
2-8 Sweep representation	19
2-9 Constructive Solid Geometry.....	20
2-10 Sierpinski triangle and Koch snowflake.....	21
2-11 Peano curve and Mandelbrot set.....	21
2-12 Lorenz attractor.....	22
2-13 A cellular structured space for a tetrahedron	24
2-14 A cellular structured space for a cone and a cylinder.....	25
3-1 Set of points.....	37
3-2 Lines with two boundary points	37
3-3 Lines with one boundary point	38
3-4 Two types of line used to compose a cylinder.....	38
3-5 Surface attached to 1-D skeleton.....	39
3-6 Union operation.....	37

3-7	Attaching operation	37
3-8	A cellular structured for a cylinder.....	44
3-9	X_1^0 and X_1^1 structure of tea cup.....	46
3-10	Construction of X_1^2 by attaching surfaces to 2D skeleton and X_1^3 structure of a tea cup	47
3-11	Modeling the goblet from three parts; A,B and C	48
3-12	X_1^1, X_1^2 and X_1^3 structure of a goblet	49
3-13	$X_2^3 = X_{2A}^3 \amalg X_{2B}^3 \amalg X_{2C}^3$	50
3-14	X_1^1, X_1^2 and X_1^3 structure of a tea cup	54
3-15	Graph representation of the cellular structured space for the teacup	56
3-16	Graph representation of the attaching objects	57
3-17	Illustration of the TDG matching algorithm	58
4-1	Nodes are 3D objects and edges are attaching relation	59
4-2	Modeling interface spitted into four views	60
4-3	Full screen perspective views.....	61
4-4	Translate 0D cell, which is a point.	61
4-5	Translate 1D cell, which is a line.....	62
4-6	Define attribute to each cell for similarity assessment.....	62
4-7	<i>TDG</i> for a tetrahedron.....	63
4-8	A 2D skeleton of cellular structured space cylinder.....	63
4-9	Adjust attribute for this 2D boundary circle such as radius.....	64
4-10	2D boundary circle with different radius.....	64
4-11	<i>TDG</i> for a cylinder.....	65
4-12	Calculating similarity between two objects	65
4-13	Object (a) is a circular surface.....	66
4-14	Object (b) is a triangular surface.....	66

4-15	Object (c) is a square surface.....	67
4-16	Object (a) is a tetrahedron.....	68
4-17	Object (b) is a cone.....	69
4-18	Object (c) is a cylinder.....	69
4-19	Cellular structured space model of a cube.....	71



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

LIST OF TABLES

Table	Page
3-1 Table for 1-dimensional skeleton.....	42
3-2 Table for 2-dimensional skeleton	42
3-3 Table for 3-dimensional skeleton.....	43
3-4 Table for 1-dimensional skeleton of the teacup.....	51
3-5 Table for 2-dimensional skeleton of the teacup.....	52
3-6 Table for 3-dimensional skeleton of the teacup.....	52
3-7 Cellular Structured Table for 3D Object.....	53
4-1 Result of the first matching experiment; $\beta, \alpha, \omega, c = 0.5$	67
4-2 Result of the second matching experiment; $\beta, \alpha, \omega, c = 0.5$	70



 สถาบันวิทยบริการ
 จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER I

INTRODUCTION

1.1 Background and Statement of Problems

Three major parts in computer graphics pipeline are modeling, deforming and rendering. Firstly, modeling is a general phrase, which is used to define a data structure to an object that we want to present. We collect information of the object by transforming the object into data structure such as set of points or vertex. Next, deforming make the object translated or rotated. Finally, rendering phrase, we concern about light, shadow and color that the object receives. There are two main types of rendering – photo realistic and non-photo realistic.

Modeling is one of the major parts in computer graphics pipeline. There are many applications for modeling objects are developed. Because the cost of modeling is expensive, several techniques have been proposed. However, each technique has been developed for some specific objects and usually has some unpredicted drawback when used for other objects.

Modeling in computer graphics usually starts from geometric level, which categorizes objects in general way more than presentation level. But both geometric level and presentation level cannot preserve some topological properties.

The problems occur when we try to retrieve some features from the objects, because of the lack of correct definition. For example, a wire frame tetrahedron modeling consists of six equivalent wires, divided into two types: open wire and closed wire, to compose a consistent wire frame tetrahedron.

When a wire frame tetrahedron with 3 open wires is composed, the meeting point at the vertex is missing. So the constructed wire frame tetrahedron has no vertex. Moreover, when a wire frame tetrahedron is composed, three meeting points at the one vertex are found. Therefore, a wire frame tetrahedron with 3 points at each vertex is constructed.



Figure 1-1. A tetrahedron composed from different wires; closed and open.

The solution is to use one closed wire and two open wires are proposed. The constructed tetrahedron looks correct on the screen, but the equivalence of wires is lost. The wire frame tetrahedron is also asymmetric and depends on the directions of the tetrahedron.

The same problem is found when boundary tetrahedron is modeled. A boundary tetrahedron is constructed with 4 equivalence triangle planes. The boundary tetrahedron has no edge line when open planes are used, while closed planes give double lines. The solution is to use a pair of open and closed planes to compose a boundary tetrahedron, but the plane equivalence is still missed.



Figure 1-2. A boundary tetrahedron constructed from different types of plane.

To sum up, in mathematics, a point and a line have no size, thus they cannot be displayed in display screen, while in computer graphics, a point and a line are defined by a pixel and a series of pixels to make them displayable. Different based-definitions are used with the same mathematical method to define objects in computer graphics, so we need more level of definition for solving the problem described above.

Furthermore, modeling also plays an important role in computer system; for example when the model is used in multimedia system, the efficiency of 3D data retrieval depends on the object representation. Several techniques for modeling 3D objects have been proposed such as polygonal meshes [1, 2], surface modeling such as implicit surface [3] [4] and parametric surface [1, 3], NURBS surface, Bézier

surface, subdivision surface [1, 2, 4, 5] However, these methods seem to lack of topological properties, which is important for similarity estimation.

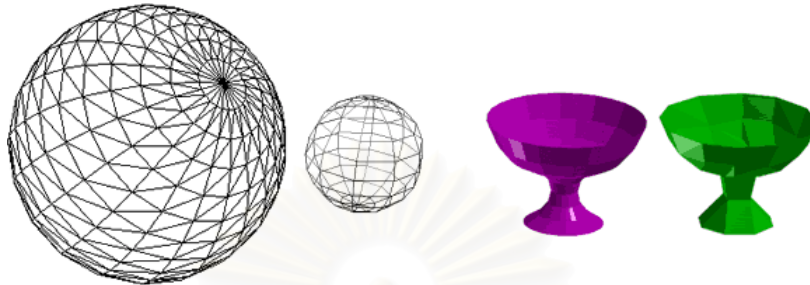


Figure 1-3. Two similar objects with different geometrical data.

The modeling of Reeb graph is proposed [6, 7] to reserve the topological properties. A method for fully automatic similarity estimation of 3D shapes using the Multi-resolutional Reeb graphs based on geodesic distance function is then proposed [8]. However, this method may not give precise estimation when objects are overly deformed in geometry.

In this research, we present a method for modeling object based on the definition level between Topology and Geometry called cellular structure space and finding the similarity between them. For modeling, the objects in n -dimensions are composed from structure in $n-1$ dimensions. For example, we composed a tetrahedron out of four triangular planes and wire frame tetrahedron as its skeleton.

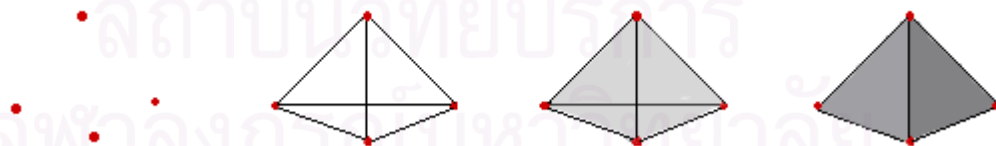


Figure 1-4. Cellular Structured Space in 0 - 3 dimension for tetrahedron.

1.2 Objectives

The objective of this study is to apply theoretical idea in cellular structured space into modeling method in 3D computer graphics by developing a modeling application using cellular structured space modeling concept. Once a model has been created, we propose a structure of the model by graph representation collected as a topological data together with geometrical data called cell properties such as color and position. Moreover, in our experiment, we use that structure for similarity assessment between two cellular structured space models.

1.3 Scope of the thesis

1. Apply theoretical idea in cellular structured space into modeling method for 3D computer graphics modeling.
2. Develop the three dimensional modeling system for composing cellular structured space model as well as function for finding similarity between two cellular structured space models.
3. This modeling application is applicable for creating object with at least one manifold in each dimensions.
4. Experiment on creating primitive models and develop the algorithm for similarity assessment between two cellular structured space models.
5. Models that can be assessed their similarity in this research are created by this modeling system.

1.4 Research Procedure

This section presents draft of the plan for the thesis.

1. Literature Review.
2. Specify the problem and objectives.
3. Research and study the previous work along with the advantages and disadvantages analysis.
4. Develop the method for modeling object by using cellular structured space.
5. Design experiment.

6. Experiment.
7. Validate and analyze the result from the experiment.
8. Publish scientific papers from the work and experiment.
9. Write the thesis.
10. State the conclusion, discussion and future works.

1.5 Expected Benefit

Using this modeling method, we can preserve some important topological properties of three dimensional objects that lack in other modeling methods and models created by using this method are valid shape models which conforms an equivalence relation each dimension. This research emphasizes on offering a new modeling method and implementing modeling system based on this concept. User can apply mathematical knowledge of cellular structured space or CW-complex to model three dimensional objects via this modeling application, so one can visualize the mathematical idea into screen.

The model created by this modeling method has two data types; geometrical data and topological data. The topological data of the model is represented by graph structure and can be used for similarity assessment between cellular structured space models and applicable for searching objects in three dimensional databases by using their cellular structured as a key for screening promising objects. Moreover, storing only the important structure into the database can reduce the size of database and reduce the redundant data of the fundamental objects also. In order to find the similarity, cellular structured space models are determined by concerning sub-structure in each dimension. By using graph structure to represent the model, we can assess similarity between two models by comparing their structure and award the similarity of two models if they have high matching nodes and edges. Moreover we concern attribute of arbitrary cell in each dimension, calculating their attribute difference is another feature to find their similarity.

1.6 Thesis Structure

This thesis is divided into 6 chapters, which are introduction, theory, related works, algorithm, implementation and results, conclusion, discussion and future works.

First chapter, introduction, provides problem statement, objectives, scope, research procedure, benefits, research structure and publications. Chapter 2 gives a brief description about related theory and some research on cellular structured space and previous model matching techniques are then discussed. In chapter 3, proposed modeling method, component of the model, example and similarity assessment between two models are presented. The implementation and results is shown in chapter 4, follow by the conclusion, discussion and future works in the final chapter.

1.7 Publication

1. Nat Charussuriyong and Pizzanu Kanongchaiyos. 2004. Modeling Object in Computer Graphics using Cellular Structure Space Modeling. The 1st Thailand Computer Science Conference (ThCSC 2004), December, Kasetsart University, Bangkok, Thailand.
2. Nat Charussuriyong and Pizzanu Kanongchaiyos. 2005. 3D Object Modeling Method for Multimedia using Cellular Structured Space. The 6th Thailand Computer Science Conference (ICCIMA 2005), December, University of Nevada, Las Vegas, USA.

CHAPTER II

THEORETICAL BACKGROUND AND RELATED WORKS

2.1 Theoretical Background

This section is divided into 5 parts which are definition of abstraction hierarchy of invariant, mathematical background for this thesis, a brief theory about object representation, cellular structured space and graph theory.

2.1.1 Abstraction hierarchy of invariant

Finding the invariants to model the objects in the real world develops scientific modeling. In mathematics, the objects are classified into equivalence classes as a disjoint union of the subsets of objects by an equivalence relation which represents a mathematical invariant. We use the same idea to classify objects in computer graphics, and the five level of abstraction are proposed.

First, *a presentation level*, in this level objects are classified by a view equivalence relation based on personal cognition. Second, *a geometry level*, objects in this level is classified by geometrical data or information model equivalence relation. Third, *a topology level*, topological equivalence relation is used to classify objects. Fourth, *a set level*, in this level objects are classified by a set theoretical equivalence relation. Finally, *a homotopy level*, homotopic function is used.

Recently, a new level called Cellular structured level has been proposed for improving model classification [9]. The abstraction hierarchy based on the equivalence relation in mathematics is proposed as following:

1. Homotopy Level
2. SET level
3. Topology level
4. Cellular structured level
5. Geometry level
6. Presentation level

2.1.2 Mathematical background

Set Theory

We summarize the ideas of set theory, and establish the basic notation and terminology using in the following sections [10].

A *set* is a collection of objects. The objects belonging to the set are the *elements* or *members* of the set. If an object a belongs to a set A , we express by notation $a \in A$. A set with no elements is called an *empty set*, ϕ , and all empty set are equal.

A set A is said to be a *subset* of a set B if every element of A is a member of B . We use the notation $A \subseteq B$, and also say that B contains A .

The set whose elements are sets is called a *collection* of sets. The set of all subsets of A is called the *power set* of A and denoted by $P(A)$ or 2^A . It is also called the *discrete topology* of A .

Given sets A and B , we can perform the *set theoretical operations* such as the union $A \cup B$, the intersection $A \cap B$, the difference $A - B$ or $A \setminus B$, and the complement of A , A^c . Given a collection \mathcal{A} of sets, the union of the elements of \mathcal{A} is defined by

$$\bigcup_{A \in \mathcal{A}} A = \{x \mid x \in A \text{ for at least one } A \in \mathcal{A}\},$$

the intersection of the elements of \mathcal{A} is defined by

$$\bigcap_{A \in \mathcal{A}} A = \{x \mid x \in A \text{ for every } A \in \mathcal{A}\},$$

If \mathcal{A} is an empty collection, then $\bigcup_{A \in \mathcal{A}} A = \phi$ and $\bigcap_{A \in \mathcal{A}} A = U$; U is a *universe*.

Functions

From a common view of function, it is a rule or equation that assigns a unique y-value to each possible x-value. It likes the machine that when the certain button is pushed the same result always happens. [10, 11].

A *rule of assignment* is a subset r of the Cartesian product $A \times B$ of two sets, having the property that each element of A appears as the first coordinate of at most one ordered pair belonging to r .

Given a rule of assignment r , the *domain* of r is defined to be the subset of A consisting of all first coordinates of elements of r , and the *image set* of r is defined as the subset of B consisting of all second coordinates of elements of r .

A *function* is a rule of assignment r , together with a set B that contains the image set of r . The domain A of the rule r is also called the *domain* of the function f ; the image set of r is also called the *image set* of f ; and the set B is called the *range* of f . If f is a function having domain A and range B , then write $f : A \rightarrow B$ and if $a \in A$ we call $f(a)$ the unique element of B the *image* of a under f .

If $f : A \rightarrow B$ be a function and $A_0 \subseteq A$. The *restriction* of f to A_0 is a function $g : A_0 \rightarrow B$ so that $f(x) = g(x)$ for all $x \in A_0$. We will use the notation $f|_{A_0}$ for the restriction of f to A_0 .

Given a function $f : A \rightarrow B$, there are three basic types of relationships:

1. The function f is *one-to-one* $\Leftrightarrow \forall x_1, x_2 \in A, [f(x_1) = f(x_2)] \Rightarrow [x_1 = x_2]$.
2. The function f is *onto* or *surjective* $\Leftrightarrow \forall y \in B, \exists x \in A, [f(x) = y]$.
3. The function f is *invertible* or *bijjective* $\Leftrightarrow f$ is one-to-one and onto.

Equivalence Relations

A *relation* on a set X is a subset R of the Cartesian product $X \times X$. We use xRy to mean that $(x, y) \in R$.

A relation R is *reflexive* if $\forall x \in X, xRx$, *symmetric* if $\forall x, y \in X, xRy \Rightarrow yRx$ and *transitive* if $\forall x, y, z \in X, (xRy \wedge yRz) \Rightarrow xRz$.

An *equivalence relation* on a set X is relation \sim on X having the three properties as following: *reflexive*, *symmetric* and *transitive*.

Given $x \in X$, an *equivalence class* of x is defined by $x / \sim = \{y \in X \mid x \sim y\}$.

Moreover, the set of all equivalence class, called the *quotient space* or *identification space* of X , is defined by $X / \sim = \{x / \sim \in 2^X \mid x \in X\} \subseteq 2^X$.

A *partition* of a set X is a collection of disjoint nonempty subset of X whose union is all in X . So we can partition a set X into *non-empty* and *disjoint* equivalence classes.

Topology

A *topological transformation* is a continuous transformation that can be continuously reversed or undone.

The *topology* is the study of properties of objects that endure when the objects are subjected to topological transformations. A *topology* on the set X is a collection τ of subsets of X having the following properties.

1. $X \in \tau$ and $\phi \in \tau$.
2. For an arbitrary index set N , $\forall_{i \in N} [U_i \in \tau \Rightarrow \bigcup_{i \in N} U_i \in \tau]$.
3. $\forall_{i, j \in N} [U_i, U_j \in \tau \Rightarrow U_i \cap U_j \in \tau]$.

We call (X, τ) a topological space with topology τ . You can see all definition in [10]. Two topological space (X, τ) and (Y, τ) are *topological equivalence* or *homeomorphic* if there is a topological transformation between them.

2.1.3 Object representation

We summarize existing object representation or modeling method. Many representations have been developed for displaying in computer graphics world. In object representation, there are wire-frame models, curve and surface models, solid models, fractal models, and procedural and grammar-based models.

2.1.3.1 Wire-frame models

Wire-frame models represent objects as a set of significant vertices connected by significant edges. The wire-frame models play an important role in many graphics application because of its inexpensive cost of drawing using world-to-screen rendering techniques. They are convenient for many tasks and could be draw very efficiently but they have a very unrealistic appearance.

In wire-frame models, the edges are described by specifying two ends points, since several edges may share some endpoints. We employ a data structure of wire-frame models by collecting vertices separately from the edges, which are described as two references to proper vertices as shown in Figure 2-2.

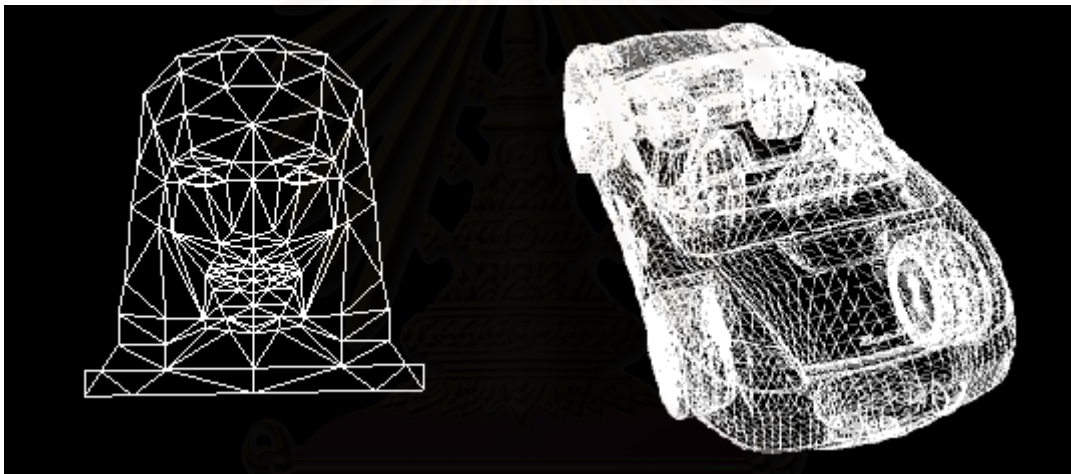


Figure 2-1. A simple and a complex wire-frame model.

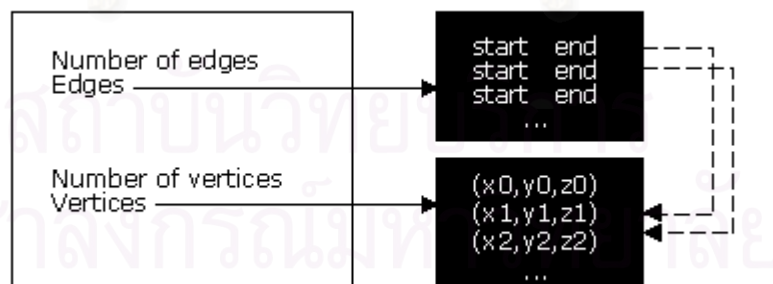


Figure 2-2. A data structure for wire-frame model.

2.1.3.2 Curve and surface models

We can use curves and surfaces to model existing and no preexisting objects. Thus, Curve and surface modeling is widely used in Computer graphics and CAD/CAM. Here, We represent some of the important ideas of the curve and surface modeling and briefly categorize this model into polygon meshed, nonparametric representation and parametric representation.

Polygon meshes

With a wire-frame model we specify a set of points in space and we connect various pairs of these points to form edges. This representation only suggests the actual shape and doesn't really look solid. We need a method to make an object look solid and to be able to color it's surface.

A polygon meshes is a collection of edges, vertices and polygons. An edge connects two vertices, and a polygon is a closed sequence of edges. An edge can be shared by adjacent polygons, and a vertex is shared by at least two edges.

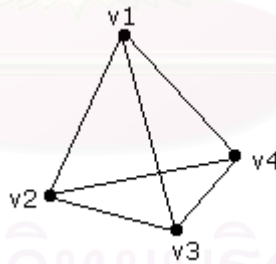


Figure 2-3. Simple polygon meshes.

Consider a polygon meshes in Figure 13; we have 4 vertices and 4 faces. A list of vertices for a polygon is created by looking at the polygon from the outside, listing the vertices in a counter-clockwise direction until a complete circle is made.

$$\begin{aligned} Polygon_1 &= \{v_1, v_2, v_3\} \\ Polygon_2 &= \{v_1, v_3, v_4\} \\ Polygon_3 &= \{v_1, v_2, v_4\} \\ Polygon_4 &= \{v_2, v_3, v_4\} \end{aligned}$$

This kind of mesh is known as a *polyhedron*, which is a closed mesh and all the faces are *planer*. A polyhedron can be used to represent almost any solid, but if the solid is highly curved, we must use a large number of faces to achieve the illusion of roundness and smoothness. We will discuss polyhedron more in section 2.3.3.3

There are three main polygon mesh representations in use: Explicit representation, Pointer to Vertex List and Pointer to Edge List.

1. **Explicit representation:** Each polygon is represented by a list of vertex coordinates.

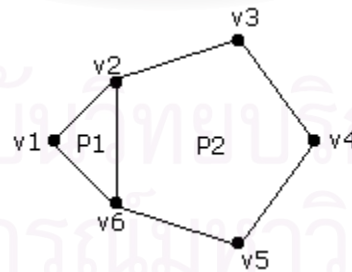
$$Face_1 = \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), \dots, (x_n, y_n, z_n)\}$$

The vertices are stored in counter clockwise order. There are edges between each pair of vertices and between the first and last vertices. This representation is not space efficient, because a shared vertex, will be stored many times.

2. **Pointer to Vertex List:** In this representation we have a vertex list, which contains all the vertices in the mesh.

$$VertexList = \{v_1 = (x_1, y_1, z_1), v_2 = (x_2, y_2, z_2), \dots, v_n = (x_n, y_n, z_n)\}$$

A polygon is defined as a list of indices or pointers into the vertex list.

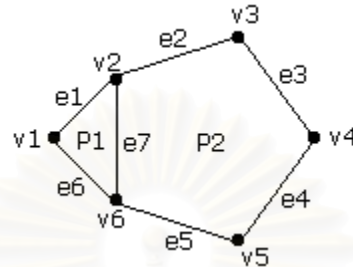


$$\begin{aligned} Face_1 &= \{v_2, v_1, v_6\} \\ Face_2 &= \{v_2, v_6, v_5, v_4, v_3\} \end{aligned}$$

Figure 2-4. Example of pointer to vertex list representation.

Each vertex is stored once. But each edge is stored twice and it is difficult to find polygons that share an edge.

3. **Pointer to Edge List:** We have a vertex list V , but we also have an edge list E , in which each edge occurs just once. Each edge is stored as a pair of pointers into the vertex list; each edge also contains pointers to the polygons, which share the edge. Polygons are defined as lists of pointer to the edge list.



$$\begin{aligned} \text{VertexList} &= \{v_1, v_2, v_3, v_4, v_5, v_6\} \\ \text{edge}_1 &= \{v_1, v_2, P_1, \lambda\} \\ \text{edge}_2 &= \{v_2, v_3, P_2, \lambda\} \\ \text{edge}_3 &= \{v_3, v_4, P_2, \lambda\} \\ \text{edge}_4 &= \{v_4, v_5, P_2, \lambda\} \\ \text{edge}_5 &= \{v_5, v_6, P_2, \lambda\} \\ \text{edge}_6 &= \{v_1, v_6, P_2, \lambda\} \\ \text{edge}_7 &= \{v_2, v_6, P_1, P_2\} \\ \text{Polygon}_1 &= \{e_1, e_6, e_7\} \\ \text{Polygon}_2 &= \{e_2, e_3, e_4, e_5, e_7\} \end{aligned}$$

Figure 2-5. Example of pointer to edge list representation.

Specialized graphics hardware often prefers particular arrangement of polygons to draw very fast. Many objects can be representation as triangle strips and triangle fans.



Figure 2-6. Triangle fans and strips.

Nonparametric Representation

The nonparametric representation describes objects in terms of the coordinates in the reference frame. We can represent both a surface and a three-dimensional curved line in nonparametric form with either of the two Cartesian functions as followed.

$$f(x, y, z) = 0 \text{ or } z = f(x, y)$$

The first one is called an *implicit* function, and the second one is called an *explicit* function.

Nonparametric representations are useful in describing objects within a given reference frame, but they have some disadvantages when used in graphics algorithm.

The most simple and widely use of nonparametric representations is Quadric surfaces. It has the implicit function of the form

$$f(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fzx + 2gx + 2hy + 2kz + C = 0$$

Nonparametric representation are useful in specialized applications and have been integrated into solid modeling system, because they are easy to compute the surface normal, test whether a point is on the surface, compute z given x and y and calculate intersections of one surface with another.

On the other hand, there are some disadvantages of nonparametric representation. In explicit function, $z = f(x, y)$, it is impossible to get multiple values of y for a single x and the definition of an explicit function is not rotationally invariant. In implicit function, $f(x, y, z) = 0$, the given equation may have more solutions than we want and if two implicit curve segments are joined together, their tangent directions may not agree at their join point.

Parametric Representation

The parametric representation describes objects according to the number of parameters needed to identify the objects' coordinate positions. The parametric representation for curves is the following function.

$$\vec{P}(u) = (x(u), y(u), z(u))$$

where each of the Cartesian coordinates is a function of u. In similar way, we can represent a surface by this two-parametric function.

$$\vec{P}(u, v) = (x(u, v), y(u, v), z(u, v))$$

Each of the Cartesian coordinates is a function of u and v .

One of the most popular method used parametric representation is spline. The spline representation is widely used for modeling curves and surfaces in both computer graphics and CAD because of the simplicity of their construction, their ease and accuracy of evaluation, and their capacity to approximate complex shapes through curve fitting and interactive curve design.

The term "spline" is used to refer to a wide class of functions that are used in applications requiring data interpolation. Splines may be used for interpolation or smoothing of either one-dimensional or multi-dimensional data. A spline curve or surface is defined with a set of control points and the boundary conditions on the spline sections. Lines connecting the sequence of control points form the control graph, and all control points are within the convex hull of a spline object. The boundary conditions can be specified using parametric or geometric derivatives, and most spline representations use parametric boundary conditions.

We can mathematically describe a curve or a surface with a piecewise cubic polynomial function whose first and second derivatives are continuous across the various curve sections. The general definition of the spline is presented as following.

A one-dimensional polynomial spline, $S(t)$, is an example of a piecewise function. In its most general form a polynomial spline, defined on an interval $[a,b]$, consists of polynomial pieces, $P_i(t)$, with each piece defined on one of a number of given subintervals

$$a = t_0 < t_1 < \dots < t_{k-2} < t_{k-1} = b$$

that is,

$$S(t) = P_0(t), t_0 \leq t < t_1$$

$$S(t) = P_1(t), t_1 \leq t < t_2$$

...

$$S(t) = P_{k-2}(t), t_{k-2} \leq t < t_{k-1}$$

Moreover, It is required that the polynomial pieces on the subintervals $[t_i, t_{i+1}), i = 0, \dots, k-2$ all have degree n ; and it is also required that two adjacent polynomial pieces $P_{i-1}(t) \in [t_{i-1}, t_i), P_i(t) \in [t_i, t_{i+1})$ connect with a specified loss of

continuity j_i so that $S(t) \in C^{n-j_i}$, $1 \leq j_i \leq n+1$ at t_i ; that is, the two pieces share common derivative values from derivative 0 (the function value) up through derivative $n-j_i$.

All piecewise polynomials $P_i(t)$, on the given interval $[a,b]$, with the given subintervals $[t_0,t_1), \dots, [t_{k-2},t_{k-1}]$, having the required degree n , and having the specified connectivities given by j_1, \dots, j_{k-2} at the interior knots t_1, \dots, t_{k-2} would be splines of a common type.

The given k points t_i are called knots. $S(t_i)$ is called a knot value and $(t_i, S(t_i))$ is called an internal control point. (t_0, \dots, t_{k-1}) is called the knot vector. If the knots are equidistantly distributed in the interval $[a,b]$ we say the spline is uniform otherwise we say it is non-uniform.

We can categorize the spline representation into two groups: interpolation splines and approximation splines. Interpolation splines connect all control points; approximation splines do not connect all control points. Interpolation splines include the Hermite, cardinal, and Kochanek-Bartels splines. B-splines, which include Bézier splines as a special case, are a versatile approximation representation, but they require the specification of a knot vector. Beta splines are generalizations of B-splines that are specified with geometric boundary conditions. And rational splines are formulated as the ratio of two spline representations. Rational splines can be used to describe quadrics, and they are invariant with respect to a perspective viewing transformation. A rational B-spline with a non-uniform knot vector is commonly known as a NURB.

In conclusion, modeling using spline representations allows for representing highly refined objects. This method provides some degree of experience for the models' creation, however the rendering process can be done in a very efficient and simple manner.

2.1.3.3 Solid models

A solid model is a representation of the geometry of a physical object. Solid models are used in many industries, from entertainment to engineering. They play a major role in computer-aided design (CAD) systems. The design process is usually

incremental. They may select models of simple shapes, such as cubes, spheres, cones or cylinders, specify their position, and orientation, and combine them to compose complicated solid models.

We review the most common schemes for representing solid models: primitive instancing, sweep representation, boundary representations, spatial-partitioning representations, and constructive solid geometry.

Primitive instancing

A primitive instancing is a parameterized 3D solid shape, which is prior, defined. It is usually represented by a type code followed by several properties. For example, a solid cone is represented by the 3-tuple ('cone', H, R), where the last two parameters are real numbers that define the dimensions of the cone, as shown in Figure 3.2.1.1. The only way to create a new kind of object is to write the new code that defines it. So, the domain of objects is very limited.

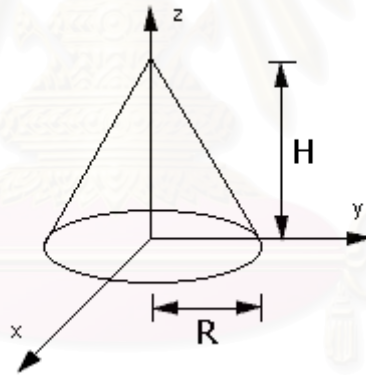


Figure 2-7. A primitive cone.

Sweep representations

Solid objects can be generated by sweeping a cross-section along a curve. For example, a cylinder can be considered as a circle swept along a line and a torus is a circle swept round a circle. The cross-section may be allowed to change in size as it is swept.

We perform a sweep by moving the shape along a path. At intervals along this path, we replicate the shape and draw a set of connecting line in the direction of the sweep to obtain the solid model.

The simplest form of sweep is that produced by sweeping a two-dimensional area along an axis perpendicular to its plane. Another common form of sweep is to rotate a two-dimensional profile around an axis. Another form of sweep representation sweeps a closed two-dimensional profile along a curve

Changing the size of the swept profile while it is swept along the curve can produce more complex objects. However, there can be difficulties to compose self-intersection surfaces. Therefore, the domain of objects that can be represented by sweeps is limited.

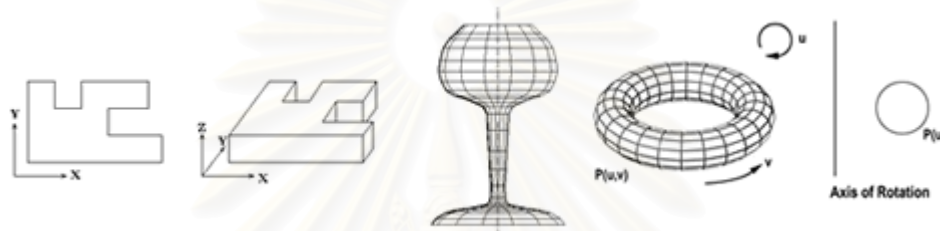


Figure 2-8. Sweep representation.

B-Rep

A boundary representation, B-Rep, of a solid object represents the object by its surface boundary. This representation usually consists of vertices, edges, and faces. The faces are typically planar, but may also be curved.

There are two types of information in a B-rep: topological and geometric. Topological information provides the relationships among vertices, edges and faces. In addition to connectivity, topological information also includes orientation of edges and faces. Geometric information is usually equations of the edges and faces.

Boundary representations incorporate a variety of data structures, including the winged-edge, quad-edge, vertex-edge, and face-edge data structures. Each data structure has its benefits and disadvantages, depending on the application. The tradeoff is between memory requirements and average access time for geometric data of interest.

One of the most popular methods is the winged-edge data structure. It is based on the observation that every edge has exactly one next edge and two previous edges in each of the two faces it appears. These are recorded as the wings of the edge.

Spatial-Partitioning Representation

In spatial-partition representation, a solid is decomposed into a collection of cells that are primitive than the original solid. There are many methods of decomposition. If the original solid is decomposed into identical and fixed-size cells, then this method is called spatial-occupancy enumeration. In 2-D, the primitive square cells are called *pixels*, picture elements, and in 3-D, cubical cells are called *voxels*. The decompositions become very large if we need more accuracy.

Hierarchical decompositions of spatial-occupancy enumeration with cells of varying sizes are designed to reduce storage requirements for three-dimensional objects, because small cells are used only where required for an accurate approximation. In 2D, called quadtrees, they are used in image processing. Quadtrees are the 2-D analogs of binary search trees, and can be used for spatial search.

The 3D extension of quadtrees is called *octrees*. Both quadtrees and octrees are special cases of *k-d trees*, which are studied in the theoretical computational geometry. Issues of completeness and validity are trivial for most spatial decompositions.

Octrees recursively divide space into eight at each step with three mutually perpendicular planes. On the other hand, A BSP trees, binary space-partitioning trees, recursively split space into a binary tree, whose leaves correspond to convex polyhedron. A selection of these leaf-cells defines the solid. BSP trees were designed originally to speed up hidden-surface removal in graphics.

Constructive Solid Geometry

In CSG objects are built up from primitive objects such as cubes, spheres, cylinders, cones etc. and combining them by logical set operations of union, intersection and difference.

A CSG model is described as a tree structure whose terminal nodes are primitive objects together with an appropriate transformation and other nodes are Boolean Set Operations.

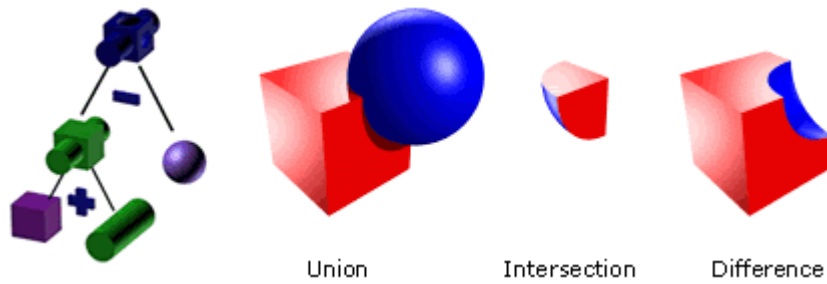


Figure 2-9. Constructive Solid Geometry.

2.1.3.4 Fractal Models

A fractal is a fragmented geometric shape that can be subdivided in parts and described by mathematical equation, each of which is a reduced-size copy of the whole. Fractals are generally self-similar and independent of scale.

There are many mathematical structures that are fractals; e.g. Sierpinski triangle, Koch snowflake, Peano curve, Mandelbrot set, and Lorenz attractor. Fractals also describe many real-world objects, such as clouds, terrains, mountains, turbulence, and coastlines, which do not correspond to simple geometric shapes.



Figure 2-10. Sierpinski triangle and Koch snowflake

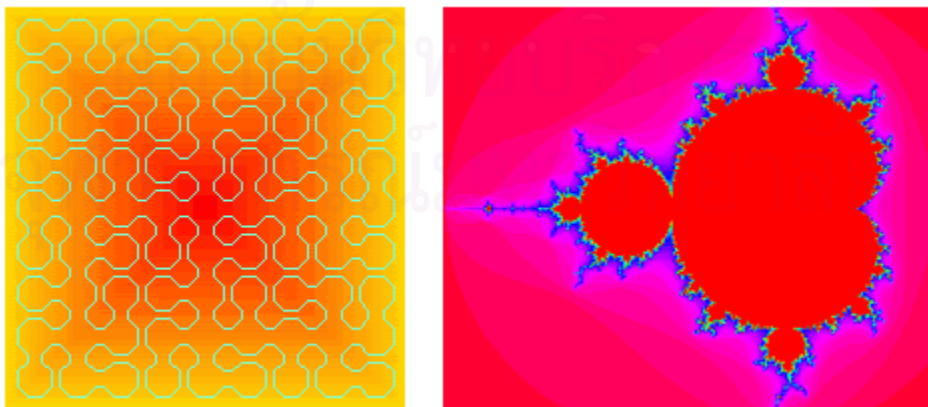


Figure 2-11. Peano curve and Mandelbrot set

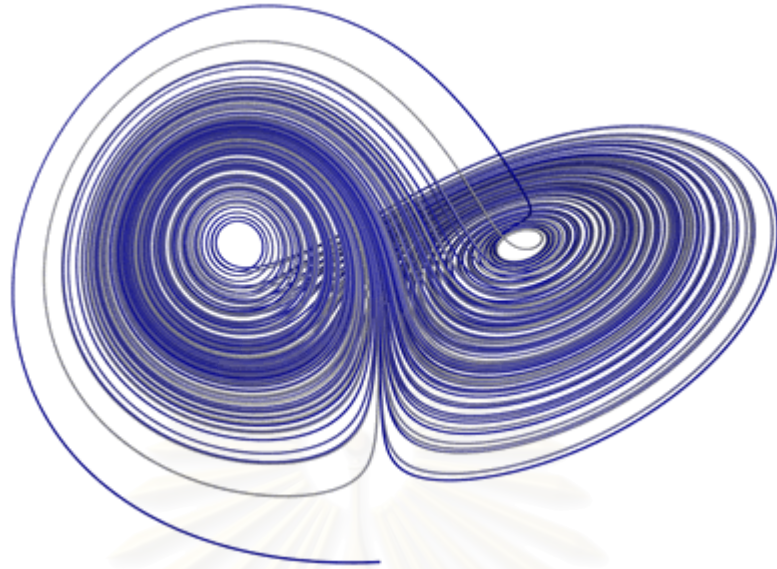


Figure 2-12. Lorenz attractor

2.1.3.5 Procedural and Grammar-based Models

A procedural model is similar to primitive instancing, but the model constructed by procedural does not need to consist of a collection of solids. Procedural models are able to interact with their environment, also interobject communication used to control the shape of objects defined by procedural.

A grammar-based model is a model generated from set of production rules that describe the shape of objects. This method, usually called *L-grammars* or *graftals*, is typically used to model plants.

2.1.4 Cellular Structured Space

Cellular structured space based on theory of cellular structure or cell complexes in algebraic topology [1] are proposed as followed. First, a cell is a topological space X that is topologically equivalent to an, n-dimensional open ball, $IntB^n$, and called an n-cell, e^n .

$$B^n = \{x \in R^N \mid |x| \leq 1\}, \text{ n-dimensional closed ball.}$$

$$IntB^n = \{x \in R^N \mid |x| < 1\}, \text{ n-dimensional open ball.}$$

$\partial B^n = B^n - \text{Int}B^n = S^{n-1}$, a boundary of a n-dimensional closed ball and for a topological space X , a continuous function F from B^n to X called characteristic map.

We can inductively compose a finite or infinite sequence of cells X^p that are sub spaces of X , called a filtration, such that X^0 is a set of point, 0-cell and

$$X^n \text{ covers } X^{n-1}.$$

$$X = \bigcup_{p \in \mathbb{Z}} X^p \text{ and } X^0 \subseteq X^1 \subseteq \dots \subseteq X^{p-1} \subseteq X^p \subseteq \dots \subseteq X.$$

The skeleton with a dimension at most p is called a p -skeleton. A space topologically equivalent to a filtration is called a filtration space. We can decompose objects into small parts and collect them in database.

We can compose a new cellular structured space Y by attaching an open n -cell, e^n , to the already composed topological space X , using a surjective and continuous map f called an attaching map.

For topological space X and Y , we attach X to Y , via attaching function f .

$$Y_f = Y \amalg_f X = Y \amalg X / \sim, \amalg \text{ is a disjoint union.}$$

The set of all equivalence class is denoted as X/\sim , and is called quotient space or identification space of X ,

$$X / \sim = \{x / \sim \in 2^x \mid x \in X\}$$

It divides the space into a disjoint union of subspace. We showed some examples of objects modeled by cellular structured space modeling, starting from 0-dimension structure called X^0 , consisting of 0-cell elements which are points, and then attach B^1 to X^0 to construct X^1 . For X^p , we can construct from attaching $\amalg_i B_i^p$ to X^{p-1} via attaching function $f : \amalg_i \partial B_i^p \rightarrow X^{p-1}$, so we define our cellular structured space model as:

$$X^p = X^{p-1} \amalg (\amalg_i B_i^p) / (x \sim f(x) \mid x \in \partial B_i^p)$$

A cellular structured space for a tetrahedron

We can construct a wire frame tetrahedron by composing X^1 , starting from X^0 that consists of 0-cell as the elements, via attaching 1-cells to X^0 . In this case, X^0 is a set of four vertex points; $X^0 = \{e_1^0, e_2^0, e_3^0, e_4^0\}$.

A tetrahedron has six edges, and we attach

$$\coprod_i B_i^1 = B_1^1 \amalg B_2^1 \amalg B_3^1 \amalg B_4^1 \amalg B_5^1 \amalg B_6^1$$

to X^0 via attaching function $f : \coprod_i \partial B_i^1 \rightarrow X^0$.

$$X^1 = X^0 \amalg_{f_i} (\coprod_i B_i^1) = X^0 \amalg (\coprod_i e_i^1), \text{ where } i = 1, 2, 3, 4, 5, 6.$$

$$f_1 : \partial B_1^1 \rightarrow \{e_1^0, e_2^0\} \quad f_2 : \partial B_2^1 \rightarrow \{e_2^0, e_3^0\} \quad f_3 : \partial B_3^1 \rightarrow \{e_3^0, e_4^0\}$$

$$f_4 : \partial B_4^1 \rightarrow \{e_4^0, e_1^0\} \quad f_5 : \partial B_5^1 \rightarrow \{e_2^0, e_4^0\} \quad f_6 : \partial B_6^1 \rightarrow \{e_1^0, e_3^0\}$$

Since a tetrahedron has four surfaces, a skeleton X^2 can be obtained by attaching 4 surfaces, $\coprod_i B_i^2 = B_1^2 \amalg B_2^2 \amalg B_3^2 \amalg B_4^2$, to X^1 via attaching function $g : \coprod_i \partial B_i^2 \rightarrow X^1$.

$$X^2 = X^1 \amalg_{g_i} (\coprod_i B_i^2) = X^1 \amalg (\coprod_i e_i^2), \text{ where } i = 1, 2, 3, 4.$$

$$g_1 : \partial B_1^2 \rightarrow \{e_1^1, e_2^1, e_6^1\} \quad g_2 : \partial B_2^2 \rightarrow \{e_2^1, e_3^1, e_5^1\}$$

$$g_3 : \partial B_3^2 \rightarrow \{e_1^1, e_3^1, e_4^1\} \quad g_4 : \partial B_4^2 \rightarrow \{e_1^1, e_2^1, e_4^1\}$$

Finally, a tetrahedron has one solid body so X^3 , consisting of a body and X^2 , is obtained by attaching B^3 to X^2 via attaching function

$$h : \partial B^3 \rightarrow X^2, X^3 = X^2 \amalg_h B^3 = X^2 \amalg e^3 \quad a$$

which expresses a solid model of a tetrahedron.

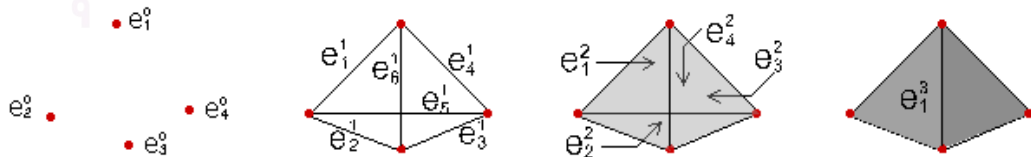


Figure 2-13. A cellular structured space for a tetrahedron.

A cellular structured space for a cone and a cylinder

We construct a cone, starting from $X^0 = \{e_1^0, e_2^0\}$, and a skeleton $X^1 = X^0 \amalg (\amalg_i e_i^1)$, where $i = 1, 2$, or $\{X^0, e_1^0, e_2^0\}$ and we construct $X^2 = \{X^1, e_1^2, e_2^2\}$ and $X^3 = \{X^2, e_1^3\}$ respectively.

Next, we construct a cylinder by the same method, which is described as following.

$$X^0 = \{e_1^0, e_2^0\}, X^1 = \{X^0, e_1^1, e_2^1, e_3^1\}, X^2 = \{X^1, e_1^2, e_2^2, e_3^2\} \text{ and } X^3 = \{X^2, e_1^3\}.$$

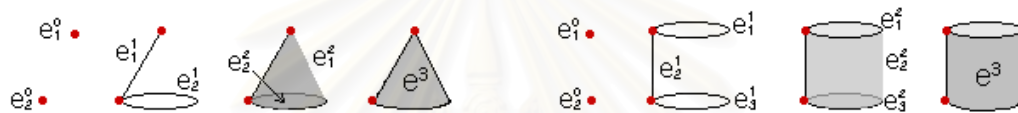


Figure 2-14. A cellular structured for a cone and a cylinder.

2.1.5 Graph Theory

Graph is one of the most abstract and powerful structures for modeling and describing structured objects. Its fundamental ideas were introduced by the great mathematician Leonard Euler in the eighteen-century. To solve the Königsberg bridge problem, he started a new field in mathematics called combinatorics. Graph theory is widely used in practical research in many fields. For instance, biology uses graph for representing complex biological systems. Moreover, problems in chemistry have been using graph theory for a long time. In chemistry, chemical compound have an intrinsic graph structure, with nodes representing various types of atoms and edges representing inter-atomic bonds. In social network, graph theory is used to represent relationship among members by different rules. The World Wide Web or WWW is an astonishing graph of which edges between nodes represent links between web pages. The Web graph provides the statistics and dynamic connectivity used to retrieve information from the Web. Furthermore, computer graphics and computer vision has long used graph theory to represent data structures. An example of this kind of application is the image application software, which can often identify interesting features and describe the relationship among these features in two given images.

In this research, we represent the structure of cellular structured space models by using graph and assess similarity between them by using graph similarity method. We define some basic graph terminology and definition in this section.

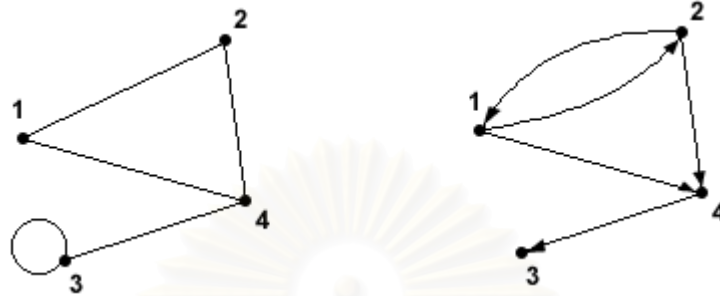


Figure 2-15. An undirected graph and a directed graph.

Graph

A graph is a 4-tuple $G = (V, E, \nu, \varepsilon)$, where

- V is a set of vertices.
- $E \subseteq V \times V$ is a set of edges.
- $\nu : V \rightarrow L_V$ is a function assigning labels to vertices.
- $\varepsilon : E \rightarrow L_E$ is a function assigning labels to edges.

A graph may be directed or undirected; in an undirected graph, for $u, v \in V$ then $(u, v) \in E \Rightarrow (v, u) \in E$.

Subgraph

A subgraph of $G = (V, E, \nu, \varepsilon)$ is a graph $S = (V_S, E_S, \nu_S, \varepsilon_S)$ such that

- $V_S \subseteq V$
- $E_S \subseteq E \cap (V_S \times V_S)$
- $\nu_S(v) = \begin{cases} \nu(v) & \text{if } v \in V_S \\ \text{undefined} & \text{otherwise} \end{cases}$
- $\varepsilon_S(e) = \begin{cases} \varepsilon(e) & \text{if } e \in E_S \\ \text{undefined} & \text{otherwise} \end{cases}$

Adjacency Matrix

The adjacency matrix representation of a graph $G = (V, E, \nu, \varepsilon)$, denoted by A_G , is a zero-one matrix with a 1 as the $(i, j)^{th}$ entry if vertices V_i and V_j are adjacent.

Graph isomorphism

A one-to-one and onto function $f : V \rightarrow V'$ is a graph isomorphism from a graph $G = (V, E, \nu, \varepsilon)$ to a graph $G' = (V', E', \nu', \varepsilon')$, denoted by $G \cong G'$, if

1. $\nu(v) = \nu'(f(v)) \quad \forall v \in V$
2. For any edge $e = (v_1, v_2) \in E$ there exists an edge $e' = (f(v_1), f(v_2)) \in E'$ such that $\varepsilon(e) = \varepsilon'(e')$, and for any $e' = (v'_1, v'_2) \in E'$ there exists an edge $e = (f^{-1}(v'_1), f^{-1}(v'_2)) \in E$ such that $\varepsilon'(e') = \varepsilon(e)$.

Moreover, if an isomorphism exists between two graphs, then they are structurally indistinguishable. In practice, if graphs G and G' are isomorphic, then there exists a permutation matrix, P , that will transform the adjacency matrix A_G into the adjacency matrix $B_G = PAP^T$.

Subgraph isomorphism

If $f : V \rightarrow V'$ is a graph isomorphism between G and G' , and G' is a subgraph of another graph G'' , then f is called a subgraph isomorphism from G to G'' .

Common subgraph and Maximum common subgraph

A graph \hat{G} is a common subgraph of two graphs G_1 and G_2 if there exist subgraphs $\hat{G}_1 \subseteq G_1$ and $\hat{G}_2 \subseteq G_2$ such that $\hat{G} \cong \hat{G}_1$ and $\hat{G} \cong \hat{G}_2$. We call \hat{G} a maximum common subgraph of G_1 and G_2 , if there exists no other common subgraph of G_1 and G_2 that has more nodes than \hat{G} .

Maximum common subgraph is a generalization of subgraph isomorphism. This measure of graph similarity compares two subject graphs by generating a third.

The maximum common subgraph of two graphs is the largest graph contained in both subject graphs.

It is beneficial to have a scalar measurement of the distance between two graphs, but graphs do not exist in a trivial metric space. However, the size of the maximum common subgraph of two graphs can be used to define a metric distance between them

The maximum common subgraph problem is known to be NP-complete. In fact, finding the maximum common subgraph between G_1 and G_2 is equivalent to finding a maximum clique in the product graph G_{AB} .

Bipartite graph

A graph $G = (V, E, \nu, \varepsilon)$ is called bipartite if its vertex set V can be partitioned into two disjoint nonempty sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex in V_2 .

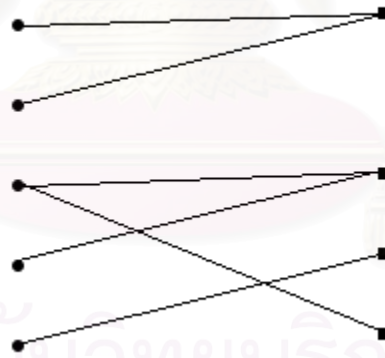


Figure 2-16. Bipartite graph.

2.2 Related Works

In this research, many related papers have been surveyed and categorized into two topics; cellular structured space and model matching. Since J.H.C. Whitehead introduced CW-complex on his publication in 1949, cellular structured space has not been used in the field outside mathematics because of its obscure benefit. Until Kunii recognized its advantages and recommended to use in Cyberworld, some researches have been developed in computer science for a few decade. Thus, in the next section of this chapter, the development of cellular structured space is summarized. Moreover, in the end of this chapter, we also summarized various techniques applied to different situations in model matching. We survey on a relevant techniques and classify into geometric based and topological based method.

2.2.1. Cellular Structured Space

Kunii has proposed on his work [9] about current graphics which lack of consistencies and suffer from invalid shape model. Moreover, this paper also point out the fundamental problem in shape modeling for computer graphics which has not been solved. Then, this work has initiated some issues to overcome the problem such as homotopy theoretical modeling.

In [45], Kunii has proposed an abstraction modeling called homotopy modeling for constructing cyberworlds. This paper claim that homotopy modeling based on novel space-time modeling with cellular spatial structures is suitable for cyberbusiness including electronic financing, electronic commerce, digital contents business which is predicted to be the biggest and most crucial business in the 21st century.

In 1999, Kunii [46, 47] proposed a method to design a valid computational shape modeling based on cellular structured space modeling. This work shows an example of simplest cases to compose higher dimensional cellular spaces inductively starting from 0-cells. Moreover, we looked at the case of a tetrahedron at the

beginning, let us go back to it as discussed previously to use it as the design specification of various shape models of a tetrahedron based on cell attachment

Kenji and Kunii [48] have introduced new theoretical tools, homotopy and cellular structured spaces for visualization. In this paper, they used an example of cup lifting to represent the idea of abstraction.

Hisada and Kunii [49] have done a research on the difficulty of representing object attachments in geometric modeling. They assume an assumption that the information of object attachment such as gluing and fusing can be represented based on the cellular models. Furthermore, they also showed a prototype implementation of the computer graphics of these cellular models, and proved that their theory can be implemented for computer-aided design and manufacturing.

In [50], Ohmori and Kunii have discussed the benefit of their model which combines homotopy and object-oriented modeling in shape modeling in the internet era. In this paper, shape modeling using homotopy is presented, implemented and instantiated via the example of tennis ball on flat slope.

In [51], Kunii has described how topological graphics is important for new computer graphics application. With the modern algebraic topology, especially homotopy theory and cellular spatial structures, topological graphics is developed and lays out the framework to interactive construct cyberworlds. Moreover, this paper also propose some design to cover wide ranges of real world application cases such as Topographic applications, Financial applications, human mental space modeling.

In [52], the adjunction space model presented as a Web information model is shown to have advantages of its generality overcoming many shortcomings of existing data models and Web information models. The researches on the implementations are underway quite well.

2.2.2. Model Matching

With recent developments in methods for modeling 3D objects, the need for retrieval of objects from large 3D models database has gained importance in graphics research. And, with current techniques that make the construction of 3D models much easier, we confront with an inundation of 3D models both on specific database and an internet. Owing to the popularity and high cost of 3D model creation, though they are much easier to create, there is an increasing demand for model sharing which is very significant in very different fields.

Cardone et al. [12] have discussed benefits of 3D model sharing in product design and manufacturing application such as cost estimation, group similar object, reusing previously design parts [12]. Over the last few years, the widespread of the World Wide Web has resulted in merged storage of mass information. To find the desired information, a number of different search engines have been established which require user to input specific simple text query and then it returns documents with matching content (e.g. Google, Yahoo, etc.). More recently, tools for searching 3D models have become significant part of some applications, including engineering, medical, architecture and entertainment. This moves development on 3D search engines [13] and multimedia search and retrieval [14] which cover an important trend in the multimedia content description.

There are a number of techniques to conduct 2D shapes or images search on model database. These methods mostly use geometric data such as silhouette or contour curve by calculating their shape curvature and Hausdorff distance [15], or color, texture and wavelet data of that image [16]. However, these techniques do not probably use in 3D models because of the difficulty in extending parameter to higher dimension such as boundary of the model.

As we turn our eye to 3D shapes, various researches for shape matching have been developed to perform similarity estimation. The selected surveys on shape similarity, for the researcher to get the picture of a current situation, are [12, 17, 18].

Recently, many developments have been devoted to the study of 3D models matching. They can be approximately categorized into two approaches: geometry-

based, and topology-based. There are a lot of researches have been done. The advantages and disadvantages of each method are as follow.

Geometry-based methods

Geometry-based methods use geometric properties in order to perform model comparison between objects. There are a number of techniques in this category. We start with methods using local feature and then discuss some global feature based method in the very next paragraph.

Cass [19] computes a local geometric feature from image data and matches 3D models in a database. This method has a trade-off between algorithmic complexity and geometric consistency, and probabilistic completeness. Another local feature based method is proposed by Körtgen et al. [20], which apply 3D shape contexts for 3D shape retrieval and matching. This method extend from 2D shape contexts firstly presented by Belongie et al. [21] with the idea that an equivalence class under a group of transformation is incomplete in context of visual system. This method is used to recognize 2D image by considering the set of vectors originating from a point to all other sample points on a model. The shape context of a point p on the shape is defined as a coarse histogram of the relative coordinates of the remaining surface points [21]. Local matching and global matching are used to match object. However, these techniques are restricted to some application and less efficient with respect to the other methods in the same category.

Keim [22] use the Maximum Included Volume (MIV) and the Minimum Surrounding Volume (MSV) to compose the geometry-based similarity search tree (GSS-tree) for indexing. The GSS-tree, in his research, is instantiated using two different hierarchical approximations: the cuboid similarity search tree (CSS-tree) and the octree similarity search tree (OSS-tree). In his experiment, the performance of the CSS-tree is better than the OSS-tree's, however the performance is up to the given data set. This method is used for searching similar 3D-volume objects and applicable for CAD and medical application.

Zhang and Chen [23] propose methods to calculate physical properties of the object known as global features including surface area, volume, moments and Fourier transform from a 2D or 3D mesh representation. The proposed method is very efficient and often used in mechanics application to find the principal axis of the model, but it still need some improvement on computational time if many coefficients of Fourier transform are simultaneous used.

Another method, using Fourier proposed by Vranić [24], converts spatial data into frequency data by using a coarse voxelization of 3D model as the input for the 3D Discrete Fourier Transform. The more efficient on frequency domain features, the higher computational complexity. We have to choose between this tradeoff. In fact, if the model has more than one major axis, the pose-normalization module may not be stable.

Osada et al. [25, 26] use a global feature as a search key for a 3D objects database. The main theme of this method is the idea of using random sampling to construct a continuous probability distribution and use it as a descriptor for 3D shape. This technique computes shape distributions of models using shape functions, which measure global geometric properties of the object, and compares these shape distributions, which measure properties based on distance, angle, area and volume measurements between random surface points, to find similarity. There are five different shape functions: A3, D1, D2, D3, and D4. In their conclusion, the D2 shape distribution function is most effective and suitable for representing features of shapes due to its robustness and efficiency. However, this type of search key cannot estimate local feature.

Vandeborre et al. [27] use linear combination of curvature index, distance index [25] and volume index [23] to compute a similarity between objects. This mixing method can improve weakness of the single descriptor methods. This hybrid method between local feature and global feature has some geometric transformation invariant and robustness to noise, but it also has some limitation in the case of the non-regular mesh is used.

Although these techniques are very efficient, they cannot perfectly match the similar model and are usually applied for model pre-categorization in an object recognition system. There are many other methods, using geometric data for model comparison. These methods consider surface properties such as: harmonic shape image [28] which is usually used in computer vision. Light Field Descriptor is proposed by Chen et al. [29], which has the main idea that two models are similar if they look similar formal viewing angle. And surface normal vector [30] represented by extended Gaussian images. However, these methods will not be reviewed here.

Topology-based methods

Unlike geometry-based methods, these methods use topological graphs as shape descriptors to perform similarity assessment. These methods consider the model skeletons for model matching. Biasotti [31] has summarized the recent topological techniques for shape understanding such as Morse theory and Reeb graphs. The selected reference papers for Morse theory and Reeb graph are [32, 33] and [34] respectively. These papers do not represent the model matching technique but they introduce some theoretical background for topological modeling and surface coding.

Sundar et al. [35] present an explicit skeletons matching method. The skeleton is used as a shape descriptor to encode geometric and topological information. This skeleton is extracted by applying a 3D thinning process after model voxelization. Thus, this method, as a result of voxelization, may subject to quantization error and high computational cost. Moreover, this method may easily mismatch models which represent different objects but have similar skeleton.

Chuang et al. [36] have discussed a well-known skeleton structure called medial axis model. The Medial Axis (MA) is the locus of the center of maximal balls contained within the object and the limit points of this locus. The maximal ball within an object is a ball contained in the objects but is not a proper subset of any other ball in the object. The medial axis transform (MAT) produced the medial axis together with the associated radius of the maximal ball around any given point on the medial

axis [37]. However, this 3D version of medial axis has a very high computational cost and is not robustness.

The Reeb graph, first defined by Reeb [38] is a structure determined by a continuous scalar function defined on an object. Lazarus et al. [39] define the geodesic distance from a source point on a surface and constructed the Reeb graph called a Level Set Diagram by extracting the isovalued contours. Reeb graphs have been used for 3D shape modeling [39], visualization, morphing [40] and terrain modeling [41]. Hilaga et al. [8] propose a topological matching method used geodesic distance to construct multi-resolution reeb graph (MRG) for articulated objects. Chen [42] proposes a 3D model retrieval system based on MRG. This MRG system work very effective in many cases, however it may still be easy to mismatch different models with similar skeleton and may be break down when dealing with high complexity object. So, Bespalov et al. [43] introduce an adjustment of Hilaga's method, which computes a scale-space decomposition of 3D models.

Yu et al. [44] propose a hybrid method to combine topology and geometry information for model matching. However, this method cannot handle highly deformable object because of the significantly change on the deformed model's center of mass.

CHAPTER III

CELLULAR STRUCTURED SPACE MODELING (CSS)

In this research, we show how to apply cellular structured space described in chapter 2 for modeling 3D objects in computer graphics, and conduct some experiment on similarity assessment between them. First, we have to create models from this concept by inductively compose from zero dimensional cell which is a set of points to three dimensional cell or solid. We collected geometrical data of each cell in each dimensions and define graph structure representing attachment between each cell to compose entire object as topological data of arbitrary object. Once we have the coarse structure, which will be weighted graph-based or some quotient space of models, we apply appropriate graph or sub-graph similarity algorithm. The cellular structured space model can be created by hand in the specific application which we have developed one for this research shown in experiment section or converted from B-rep model such as polygonal mesh, but the later method will not be discussed here.

3.1 Component of the model

In this section, we describe the component of the cellular structured space model which is point, line, surface, and volume. Moreover, we collect the attribute of each component for representing geometrical data of the model such as color and luminosity diffuse.

3.1.1 Point

Point is a topological space that is homeomorphic to a zero dimensional open ball. It is represented by a 0-cell, e^0 . All cellular structured space models are first created from zero dimensional skeleton X^0 . We define X^0 as a set of points and X^0 is a basis skeleton in inductive composing method for model construction. Point or 0-cell e^0 has attributes for representing geometrical data such as color and position. All points have no boundary so its child in CSSGraph is NULL. In figure 3-1, $X^0 = \{e_0^0, e_1^0, e_2^0, e_3^0\}$ is composed from a set of four points or zero dimensional open

ball, e^0 and each points has geometrical data (x, y, z) to represent its position in 3D view as well as its color.

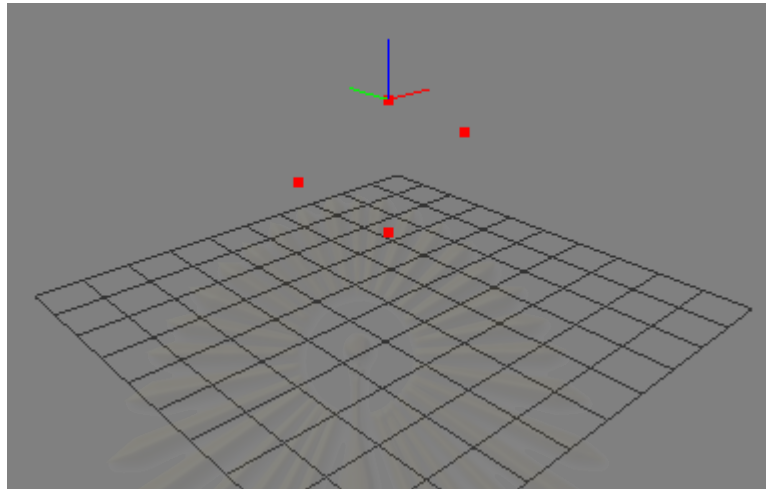


Figure 3-1. Set of points.

3.1.2 Line

Line is a topological space that is homeomorphic to a one dimensional open ball, $IntB^1 = \{x \in \mathfrak{R}, |x| < 1\}$. It is represented by a 1-cell, e^1 . All lines have one or two boundary points, thus we collect this relation as its topological structure represented in CSSGraph by node with one child or two children. In case of two boundary points, the line or one dimensional cell is represented by straight line or curve which can be implemented by using NURB.

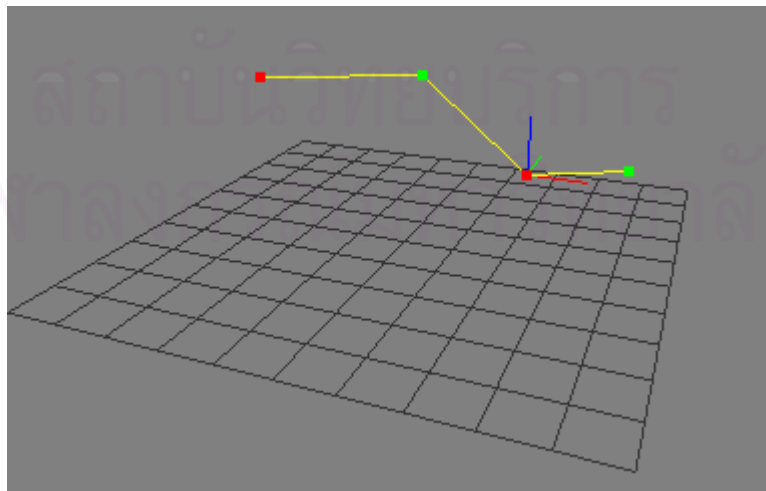


Figure 3-2. Lines with two boundary points.

In the other case, the line has only one boundary point. In this case the line is represented by a circle with attribute such as radius.

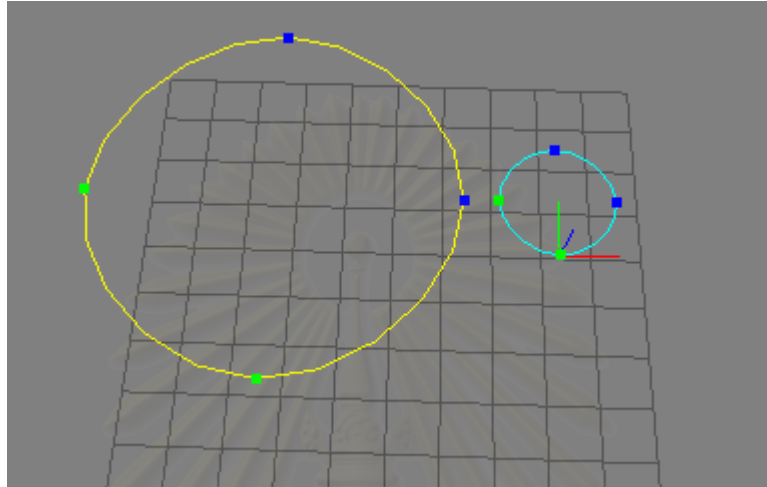


Figure 3-3. Lines with one boundary point.

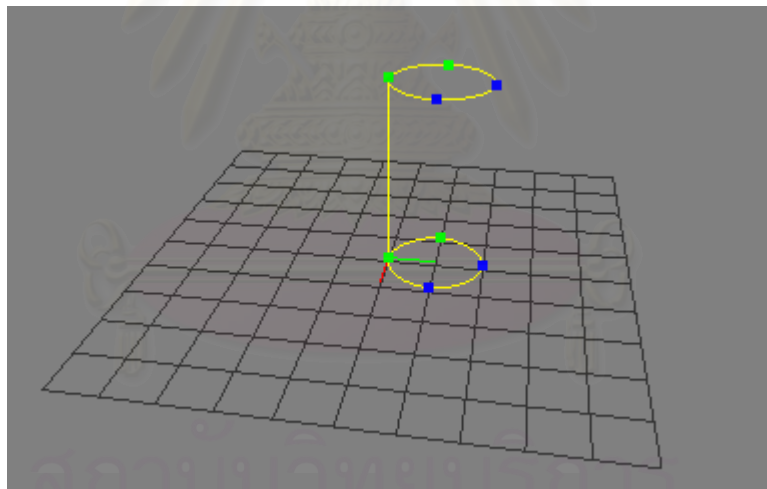


Figure 3-4. Two types of line used to compose a cylinder.

We attach lines or 1-cell, e^1 , to zero dimensional skeleton X^0 to construct $X^1 = \{X^0, e_0^1, e_1^1, e_2^1, \dots\}$ which is a one dimensional skeleton for the model. The geometrical data of lines are collected in attribute of 2-D cell node in CSSGraph such as weight and control point (if line is a NURB), curvature and radius.

3.1.3 Surface

Surface is a topological space that is homeomorphic to a two dimensional open ball, $IntB^2 = \{x \in \mathfrak{R}^2, \|x\| < 1\}$. It is represented by a 2-cell, e^2 . All surfaces have at least one line as its boundary and we collect this relation in term of graph to represent its topological data. We can compose surface by attaching two dimensional open ball to lines in X^1 . Basically, We use surface to construct boundary model or two dimensional skeleton X^2 of the model, for example, as a tetrahedron has 4 surfaces, a skeleton X^2 can be obtained using 4 surfaces and X^1 ; $X^2 = \{X^1, e_0^2, \dots, e_3^2\}$. Moreover, surfaces have properties or attributes for each cell such as color, material, texture, and luminosity diffuse. These properties are collected as geometrical data for the model.

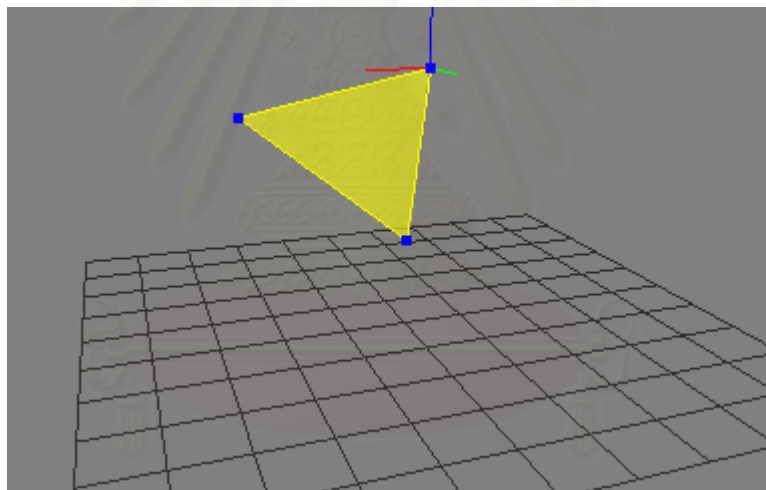


Figure 3-5. Surface attached to 1-D skeleton.

3.1.4 Volume

Volume or solid is a topological space that is homeomorphic to a three dimensional open ball, $IntB^3 = \{x \in \mathfrak{R}^3, \|x\| < 1\}$. It is represented by a 3-cell, e^3 . We attach volume to boundary model for constructing solid model.

3.2 Object Construction

We can inductively compose a cellular structured space model X with finite sequence of skeleton X^p where $X = \bigcup_{p \in \{0,1,2,3\}} X^p$ and $X^0 \subseteq X^1 \subseteq X^2 \subseteq X^3 \subseteq X$.

Using part X and Y to create the model, we have two choices of combining these two parts. First, we union X and Y . This operation is the same as in CSG. The other is attaching X to Y , via attaching function f and attaching operation \amalg , where $Y_f = Y \amalg_f X = Y \amalg X / \sim$, and \amalg is a disjoint union.

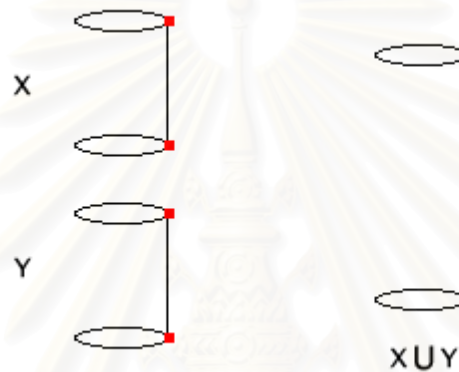


Figure 3-6. Union operation.

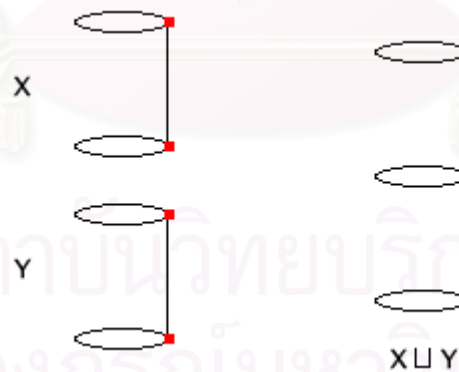


Figure 3-7. Attaching operation.

We can compose cellular structured space models by starting from 0-dimension structure called X^0 , consisting of 0-cell elements which are points, and then attach $\amalg_i B_i^1 = B_1^1 \amalg \dots \amalg B_m^1$ which is an attachment of 2-cell or edges

to $X^0 = \{e_1^0, \dots, e_n^0\}$ to construct X^1 which is a wire frame model. We use attaching function $f : \coprod_i \partial B_i^1 \rightarrow X^0$ to indicate which cells are attached.

Suppose that the model has exactly k surfaces, a skeleton X^2 can be obtained by attaching surfaces, $\coprod_i B_i^2 = B_1^2 \coprod \dots \coprod B_k^2$, to X^1 via attaching function $g : \coprod_i \partial B_i^2 \rightarrow X^1$.

Finally, we assume that the model has one solid body so X^3 , consisting of a body and X^2 , is obtained by attaching B^3 to X^2 via attaching function

$$h : \partial B^3 \rightarrow X^2, X^3 = X^2 \coprod_h B^3 = X^2 \coprod e^3$$

3.3 Data Structure of the Model

Data structure of cellular structured space model is presented in this section. The structure of CSS model is straightforward and we implement the structure of the abstract model by using the concept of Object Oriented. An object oriented programming helps to categorize the same kind of objects, which are homotopically, topologically or geometrically equivalent. In an object oriented programming, we first create the abstract class that define the overall structure of the object and then create the object as an instance of the abstract class.

To begin with, we define data structure of the cellular structured space model by create $Table(B_i^k) \subseteq \partial B_i^k \times X^{k-1}$ for each cell in each dimension. This table can be implemented as a class of object in specific dimension n and the data in that class are its properties and boundary information.

In 0-dimensional cell, it is a point, which has no boundary. This is an initial case; hence we created class, namely `vertex`, collecting data $(p_x, p_y, p_z) \in \mathfrak{R}^3$ and its properties such as color.

In 1-dimensional cell, it is a line and its boundary is a point or zero dimensional cells defined above. Thus, we defined class, called `line`, to collect its two boundary points, which can be the same point, and its line properties such as color and curvature. To illustrate, we have 2 boundary cells for each line segment and if we have n lines, then the table will look like this; $Table(B_{Ai}^1)$ for $i = 1, 2, 3, \dots, n$.

Table 3-1. Table for 1-dimensional skeleton.

Table(B_i¹)

∂B_i^1	X^0
x_1	e_1^0
x_2	e_2^0

This table is used for instancing the model, where x_1, x_2 are two end points being identified with vertex point. $x_1, x_2 \in \partial B_i^1$ for $i = 1, 2, 3, \dots, n$.

Likewise, surface is concerned as a 2-dimensional cell but its number of boundary is not constant. Therefore, we define class for surface, namely `surface`, with the vector of line and its properties, for example, 2-dimensional skeleton of a model with two surfaces; one is the triangle, the other is a circle.

Table 3-2. Table for 2-dimensional skeleton.

Table(B₁²)

∂B_1^2	X^1
λ_1	e_1^1
λ_2	e_2^1
λ_3	e_3^1

Table(B₂²)

∂B_2^2	X^1
λ_1	e_1^1

Finally, we create class collecting its boundary, which is a vector of surface, for 3-dimensional cell and we called this class `node`. When two 3D objects are created and attached, we create an edge between their nodes in 3-dimensional cell and the edge is a class that has a data such as two pointers to node and weight of this edge. For instance, X^3 has three boundaries, thus we create 1 table *Table(B_A³)* with three rows of boundaries.

Table 3-3. Table for 3-dimensional skeleton.

Table(B³)

∂B^3	X^2
π_1	e_1^2
π_2	e_2^2
π_3	e_3^2

We can combine all instance tables represented in each dimension into 1 table, namely CB-table. This table is used for showing situation and boundary relation between cell in k and $k + 1$ dimension of the model.

Now, we have an abstract model in the form of abstract instance table. But, the model has some properties such as shape, color and material. To provide these properties, instantiating abstract model should be done.

The abstract model that we created is in topology level which coordinate is insignificant. So, firstly, we have to set the coordinate system to the abstract model in this case we use Cartesian coordinate system. Then properties such as material, weight, color or coordinate are defined. Since each component is a manifold, the cell can be instantiated by a smooth parametric function. The following is an example of some properties for 0,1-dimensional cell. For finding structural similarity, these properties are not concerned as part of the features of two objects. On the other hand, if we have to assess geometric similarity, we can easily add these properties as part of distance function for finding structural and geometric similarity.

$$e_{A1}^0 \mapsto (x = 5, y = 0, z = 10) \quad e_{A2}^0 \mapsto (x = 7, y = 0, z = 10)$$

$$e_{A3}^0 \mapsto (x = 4, y = 0, z = 0) \quad e_{A4}^0 \mapsto (x = 6, y = 0, z = 0)$$

$$e_{A1}^1 \mapsto (x = r \cos 2\pi t, y = r \sin 2\pi t, z = 10, r = 5, 0 < t < 1)$$

$$e_{A2}^1 \mapsto (x = r \cos 2\pi t, y = r \sin 2\pi t, z = 10, r = 7, 0 < t < 1)$$

$$e_{A3}^1 \mapsto (x = 7 - 2t, y = 0, z = 10, 0 < t < 1)$$

3.4 CSSGraph

After we created Cell-Boundary table, CB-table, for each objects, we can represent them by graph structure called CSSGraph. The objective of this section is to present the graph representation of cellular structured space models and to illustrate the attachment between objects.

Basically, the cellular structured space model begins with defining 0-dimensional cell or set of points and incremental design is used to define the structure in higher dimensions. The $k+1$ -dimensional skeleton is composed from k -dimensional skeleton by attaching $k+1$ -D cell to that skeleton via attaching map from boundary of $k+1$ -D cell to k -D cell. Consequently, there is always a link or edge between $k+1$ and k -D cell; $k \in \{0,1,2\}$. This hierarchical structure of the cellular structured space model is presented. Such structure is often represented as directed acyclic graph (DAG), where nodes represent the cell in different dimension and edges represent its boundary relation where the terminal node is said to be a boundary of the source node.

We concern the object with one solid part, 3-dimensional cell. For general objects, we compose such objects by attaching different model with one solid part into complex objects with many solid parts. Thus, the rooted node of the DAG represented the cellular structured space model is unique for our model we concerned and stand for the 3-dimensional cell of the object.

3.5 Example

We have described the cellular structured space for some primitive models such as a tetrahedron, a cone and a cylinder. The cell in each dimension has to topological equivalent to an open ball in its dimension that is why a cylinder has three edges instead of two edges.

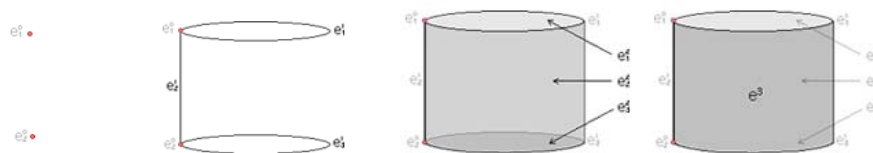


Figure 3-8. A cellular structured for a cylinder.

In this section, we show how to model and implement the concept of cellular structured space by modeling 3 kinds of cups having different shapes and structures. First, we start from 0-dimensional structure called X^0 , which are points, and we can display on screen as a pixel with three-color properties; red, green and blue components. Next, we attach B^1 to X^0 to construct X^1 , which look like the wire-frame model. The B^1 attached with X^0 by attaching function can be implemented by a line segment. We can instance this 1-dimensional cell by set the properties such as color, equation for this line section. It is trivial in case of a straight line, and we will use B-spline if the line is curve. In case of X^2 and X^3 , we use surface model for instancing X^2 and solid model for X^3 . We can inductively compose the skeleton of the model up to X^p , constructing from attaching $\coprod_i B_i^p$ to X^{p-1} via attaching function $f : \coprod_i \partial B_i^p \rightarrow X^{p-1}$. In section 2.1, we define our cellular structured space model as:

$$X^p = X^{p-1} \coprod (\coprod_i B_i^p) / (x \sim f(x) \mid x \in \partial B_i^p)$$

In the real world, we cannot recognize objects that have the dimension more than three. To represent the object, the maximum p used is three. So we have X^0, X^1, X^2 and X^3 structure and geometric data for each cell to represent the model in 3D world by convert to polygonal mesh. Next we will model a teacup and then model a goblet by attaching two cylinders to teacup model as the example.

We started from defining set of 0-dimensional cell, namely, X_1^0 . A skeleton X_1^0 of a teacup consists of 4 points, X_1^0 is $\{e_{A1}^0, e_{A2}^0, e_{A3}^0, e_{A4}^0\}$. A skeleton X_1^1 is obtained from X_1^0 ; $X_1^1 = X_1^0 \coprod_{F_{Ai}} (\coprod_i B_{Ai}^1)$, where $i \in \mathbb{N}$ and $F_{Ai} : \coprod_i \partial B_{Ai}^1 \rightarrow X_1^0$

$$\begin{aligned} \coprod_i B_{Ai}^1 &= B_{A1}^1 \coprod B_{A2}^1 \coprod B_{A3}^1 \coprod B_{A4}^1 \coprod B_{A5}^1 \coprod B_{A6}^1 \coprod B_{A7}^1 \\ F_{A1} : \partial B_{A1}^1 &\rightarrow \{e_{A1}^0\} & F_{A2} : \partial B_{A2}^1 &\rightarrow \{e_{A2}^0\} \\ F_{A3} : \partial B_{A3}^1 &\rightarrow \{e_{A1}^0, e_{A2}^0\} & F_{A4} : \partial B_{A4}^1 &\rightarrow \{e_{A1}^0, e_{A3}^0\} \\ F_{A5} : \partial B_{A5}^1 &\rightarrow \{e_{A2}^0, e_{A4}^0\} & F_{A6} : \partial B_{A6}^1 &\rightarrow \{e_{A3}^0\} \\ F_{A7} : \partial B_{A7}^1 &\rightarrow \{e_{A4}^0\} \end{aligned}$$

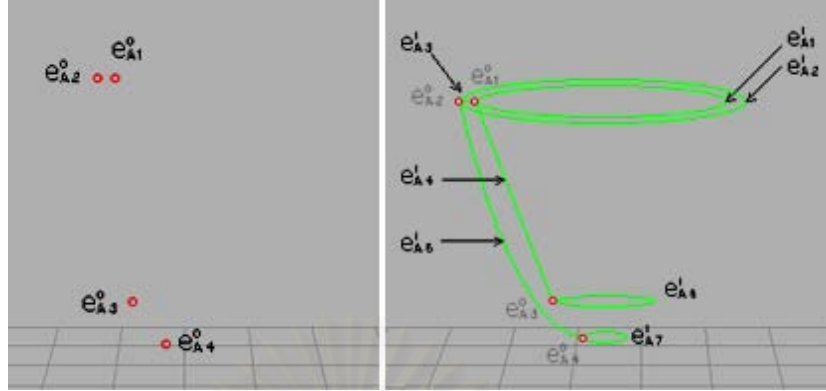


Figure 3-9. X_1^0 and X_1^1 structure of tea cup.

A skeleton X_1^2 is also obtained from X_1^1 by attaching ∂B_i^2 to X_1^1 ,

$$X_1^2 = X_1^1 \amalg_{G_{A_i}} \left(\amalg_i B_{A_i}^2 \right), \text{ where } i \in N \text{ and } G_{A_i} : \amalg_i \partial B_{A_i}^2 \rightarrow X_1^1$$

$$\amalg_i B_{A_i}^2 = B_{A_1}^2 \amalg B_{A_2}^2 \amalg B_{A_3}^2 \amalg B_{A_4}^2 \amalg B_{A_5}^2$$

$$G_{A_1} : \partial B_{A_1}^2 \rightarrow \{e_{A_1}^1, e_{A_2}^1, e_{A_3}^1\}$$

$$G_{A_2} : \partial B_{A_2}^2 \rightarrow \{e_{A_1}^1, e_{A_4}^1, e_{A_6}^1\}$$

$$G_{A_3} : \partial B_{A_3}^2 \rightarrow \{e_{A_2}^1, e_{A_5}^1, e_{A_7}^1\}$$

$$G_{A_4} : \partial B_{A_4}^2 \rightarrow \{e_{A_6}^1\}$$

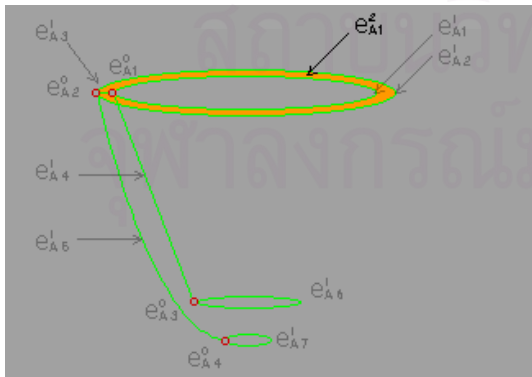
$$G_{A_5} : \partial B_{A_5}^2 \rightarrow \{e_{A_7}^1\}$$

We construct X_1^2 by attaching 2-cell, e_i^2 , to X_1^1 via attaching function G_{A_i} .

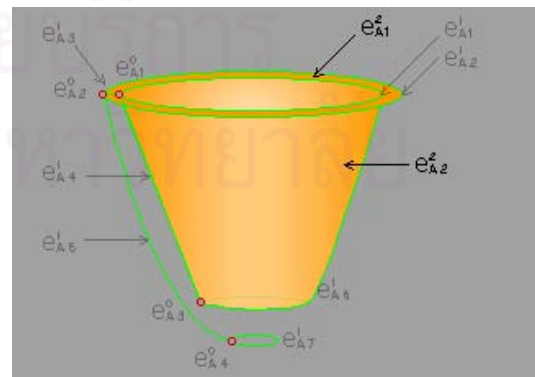
Finally, we attach e_i^3 to X_1^2 for constructing a solid model of a tea cup.

$$X_1^3 = X_1^2 \amalg_{H_A} B_A^3, \text{ where } H_A : \partial B_A^3 \rightarrow X_1^2$$

$$H_A : \partial B_A^3 \rightarrow \{e_{A_1}^2, e_{A_2}^2, e_{A_3}^2, e_{A_4}^2, e_{A_5}^2\}$$



(a)



(b)

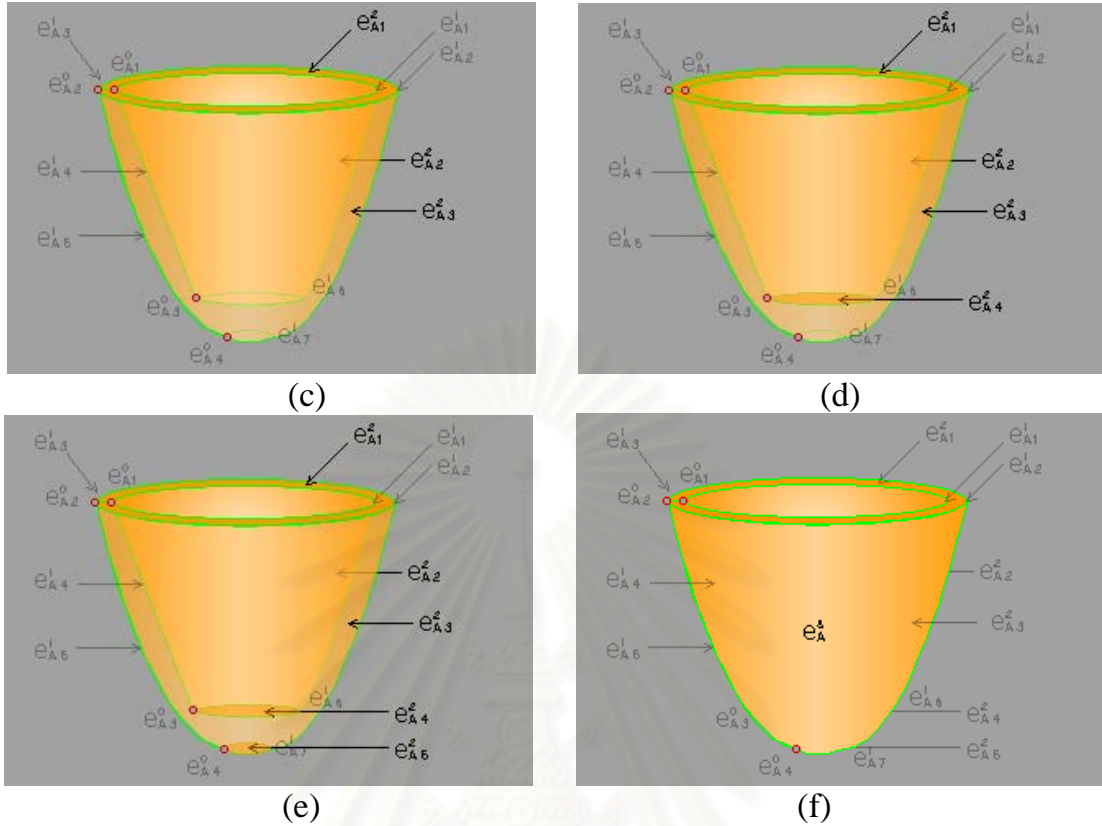


Figure 3-10. Construction of X_1^2 by attaching surfaces to 2D skeleton and X_1^3 structure of a teacup.

Next, a goblet is also represented by cellular structured space, X_2 . In this case, we started modeling the goblet from three parts, called A, B and C, and attached them together by attaching function to complete the goblet model. This model consists of 8 points and we used part A in the goblet from a teacup in the previous model.

$$X_2^0 = \{e_{A1}^0, e_{A2}^0, e_{A3}^0, e_{A4}^0, e_{B1}^0, e_{B2}^0, e_{C1}^0, e_{C2}^0\}$$

$$X_2^1 = X_2^0 \amalg_{F_{ij}} \left(\amalg_i B_{Ai}^1 \right) \amalg \left(\amalg_i B_{Bi}^1 \right) \amalg \left(\amalg_i B_{Ci}^1 \right), \text{ where } i \in \{A, B, C\} \text{ and}$$

$$F_{ij} : \amalg_i \partial B_{ij}^1 \rightarrow X_2^0$$

Part A: used the same function, $F_{A1}, F_{A2}, F_{A3}, F_{A4}, F_{A5}, F_{A6}$ and F_{A7} , as the teacup.

$$\text{Part B: } F_{B1} : \partial B_{B1}^1 \rightarrow \{e_{B1}^0\}, \quad F_{B2} : \partial B_{B2}^1 \rightarrow \{e_{B1}^0, e_{B2}^0\}, \quad F_{B3} : \partial B_{B3}^1 \rightarrow \{e_{B2}^0\}$$

$$\text{Part C: } F_{C1} : \partial B_{C1}^1 \rightarrow \{e_{C1}^0\}, \quad F_{C2} : \partial B_{C2}^1 \rightarrow \{e_{C1}^0, e_{C2}^0\}, \quad F_{C3} : \partial B_{C3}^1 \rightarrow \{e_{C2}^0\}$$

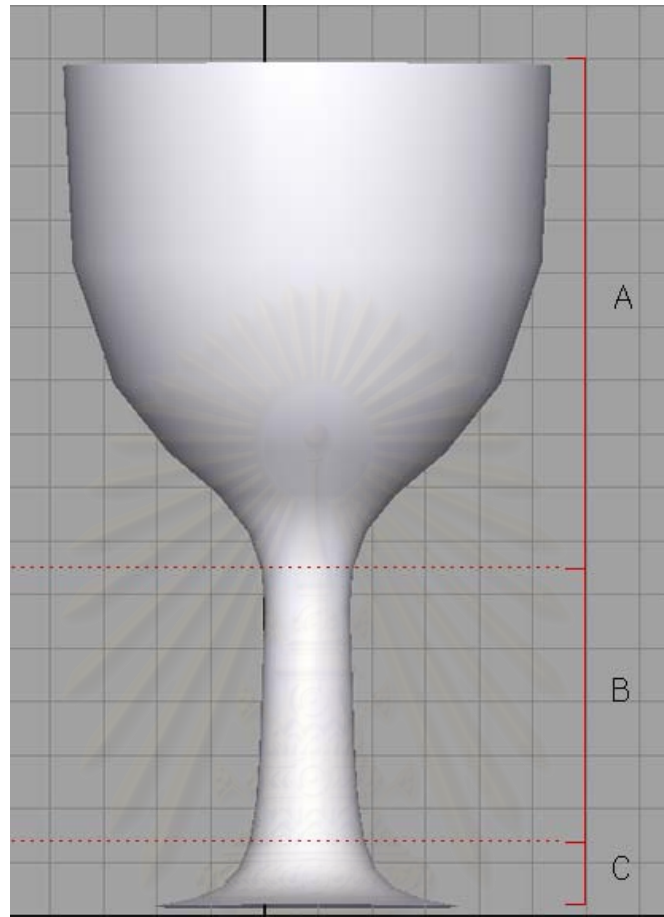


Figure 3-11. Modeling the goblet from three parts; A,B and C.

$$X_2^2 = X_2^1 \amalg_{G_{ij}} \left(\amalg_i B_{Ai}^2 \right) \amalg \left(\amalg_i B_{Bi}^2 \right) \amalg \left(\amalg_i B_{Ci}^2 \right), \text{ where } i \in \{A, B, C\} \text{ and}$$

$$G_{ij} : \amalg_i \partial B_{ij}^2 \rightarrow X_2^1$$

Part A: used the same function, $G_{A1}, G_{A2}, G_{A3}, G_{A4}$ and G_{A5} , as the teacup.

$$\text{Part B: } G_{B1} : \partial B_{B1}^2 \rightarrow \{e_{B1}^1\} \quad G_{B2} : \partial B_{B2}^2 \rightarrow \{e_{B1}^1, e_{B2}^1, e_{B3}^1\} \quad G_{B3} : \partial B_{B3}^2 \rightarrow \{e_{B3}^1\}$$

$$\text{Part C: } G_{C1} : \partial B_{C1}^2 \rightarrow \{e_{C1}^1\} \quad G_{C2} : \partial B_{C2}^2 \rightarrow \{e_{C1}^1, e_{C2}^1, e_{C3}^1\} \quad G_{C3} : \partial B_{C3}^2 \rightarrow \{e_{C3}^1\}$$

$$X_{2A}^3 = X_{2A}^2 \amalg_{H_A} B_A^3, \text{ and } H_A : \partial B_A^3 \rightarrow X_{2A}^2$$

$$X_{2B}^3 = X_{2B}^2 \amalg_{H_B} B_B^3, \text{ and } H_B : \partial B_B^3 \rightarrow X_{2B}^2$$

$$X_{2C}^3 = X_{2C}^2 \amalg_{H_C} B_C^3, \text{ and } H_C : \partial B_C^3 \rightarrow X_{2C}^2$$

$$X_2^3 = X_{2A}^3 \amalg X_{2B}^3 \amalg X_{2C}^3$$

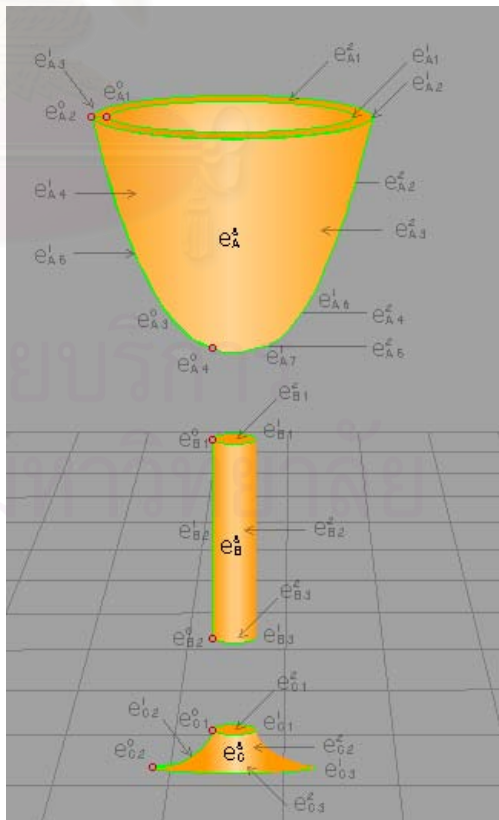
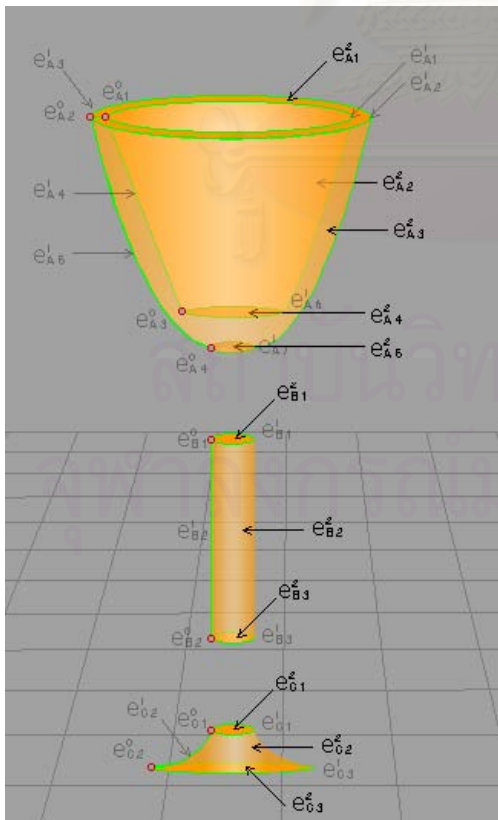
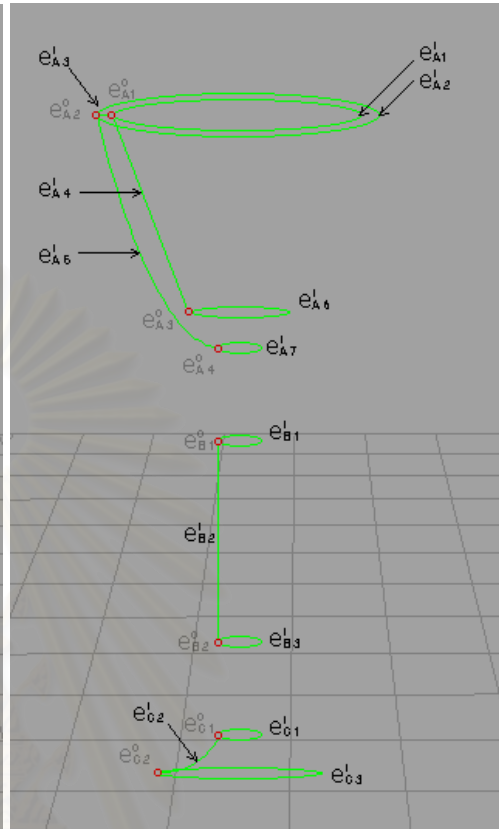
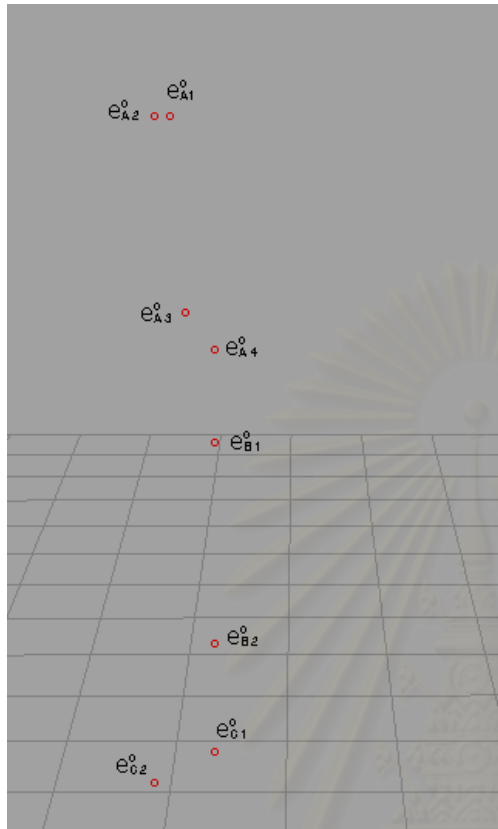
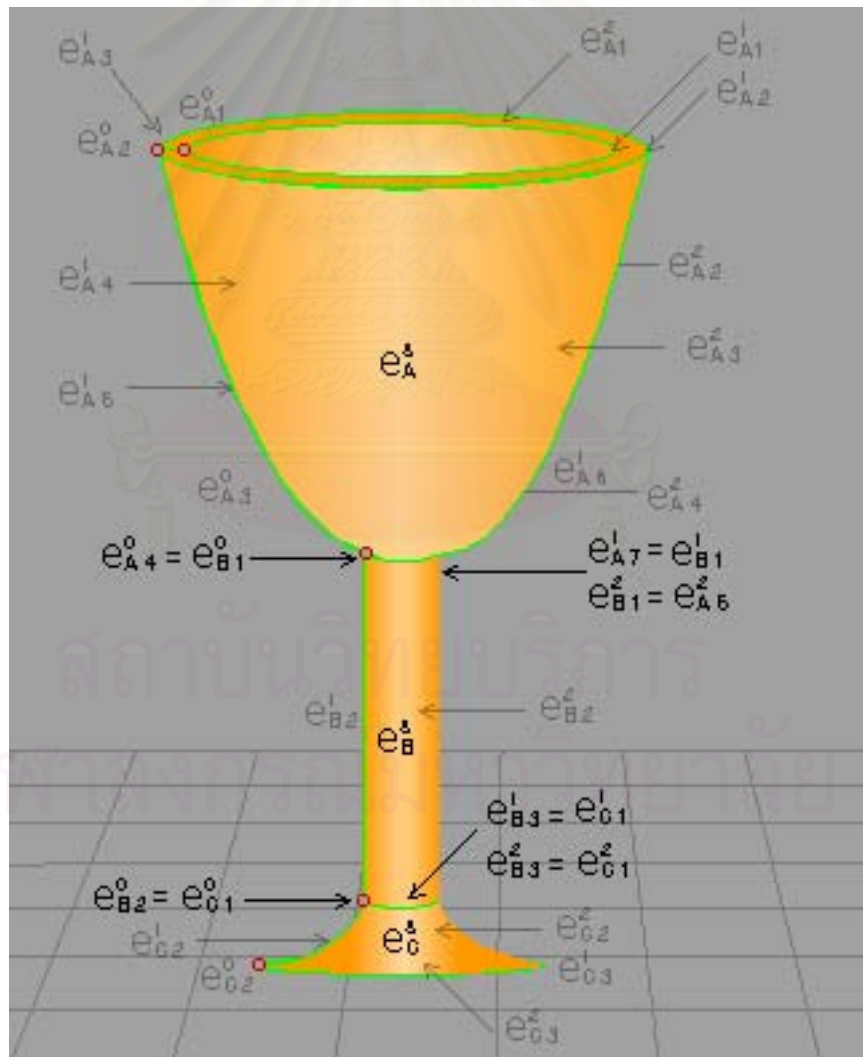


Figure 3-12. X_2^0, X_2^1, X_2^2 and X_2^3 structure of the goblet.

Then, we combine A, B and C into one goblet by attaching them together via setting $e_{A4}^0 = e_{B1}^0$ (the point at the bottom of the teacup, part A, and the point at the top of cylinder B) and $e_{B2}^0 = e_{C1}^0$ (the point at the bottom of the cylinder B, and the point at the top of cylinder C)

This attaches X_2^1 together, setting $F_{A7} \cong F_{B1}$ and $F_{B3} \cong F_{C1}$, thus we will get $e_{A7}^1 = e_{B1}^1$ and $e_{B3}^1 = e_{C1}^1$.

This also attaches X_2^2 together, setting $G_{A5} \cong G_{B1}$ and $G_{B3} \cong G_{C1}$, therefore we have $e_{A5}^2 = e_{B1}^2$ and $e_{B3}^2 = e_{C1}^2$.



Figure

3-13.

$$X_2^3 = X_{2A}^3 \amalg X_{2B}^3 \amalg X_{2C}^3$$

3.5.1 Implementation of CSS

We implement the abstract model for teacup described above. First, we implement the teacup X_1^3 designed above by create $Table(B_i^k) \subseteq \partial B_i^k \times X^{k-1}$ for each cell in each dimension as proposed in section 3.3.

In 0-dimensional cell, it is a point, which has no boundary. We have a one dimensional skeleton consist of 4 points.

In 1-dimensional cell, it is a line and its boundary is a point or zero dimensional cells defined above. For this teacup, we have 7 tables for each line segment; $Table(B_{A_i}^1)$ for $i = 1,2,3,4,5,6,7$.

Table 3-4. Table for 1-dimensional skeleton of the teacup.

$Table(B_{A_1}^1)$	$Table(B_{A_2}^1)$	$Table(B_{A_3}^1)$	$Table(B_{A_4}^1)$
$\partial B_{A_1}^1$	$\partial B_{A_2}^1$	$\partial B_{A_3}^1$	$\partial B_{A_4}^1$
X^0	X^0	X^0	X^0
x_1	x_1	x_1	x_1
$e_{A_1}^0$	$e_{A_2}^0$	$e_{A_1}^0$	$e_{A_1}^0$
x_2	x_2	x_2	x_2
$e_{A_1}^0$	$e_{A_2}^0$	$e_{A_2}^0$	$e_{A_3}^0$
$Table(B_{A_5}^1)$	$Table(B_{A_6}^1)$	$Table(B_{A_7}^1)$	
$\partial B_{A_5}^1$	$\partial B_{A_6}^1$	$\partial B_{A_7}^1$	
X^0	X^0	X^0	
x_1	x_1	x_1	
$e_{A_2}^0$	$e_{A_3}^0$	$e_{A_4}^0$	
x_2	x_2	x_2	
$e_{A_4}^0$	$e_{A_3}^0$	$e_{A_4}^0$	

These tables are used for instantiating the model, where x_1, x_2 are two end points being identified with vertex point. $x_1, x_2 \in \partial B_{A_i}^1$ for $i = 1,2,3,4,5,6,7$.

Likewise, surface is concerned as a 2-dimensional cell but its number of boundary is not constant. For X_1^2 , 2-dimensional skeleton of this teacup, we create 5 tables for each surface; $Table(B_{A_i}^2)$ for $i = 1,2,3,4,5$.

Table 3-5. Table for 2-dimensional skeleton of the teacup.

$Table(B_{A1}^2)$		$Table(B_{A2}^2)$		$Table(B_{A3}^2)$		$Table(B_{A4}^2)$		$Table(B_{A5}^2)$	
∂B_{A1}^2	X^1	∂B_{A2}^2	X^1	∂B_{A3}^2	X^1	∂B_{A4}^2	X^1	∂B_{A5}^2	X^1
λ_1	e_{A1}^1	λ_1	e_{A1}^1	λ_1	e_{A2}^1	λ_1	e_{A6}^1	λ_1	e_{A7}^1
λ_2	e_{A2}^1	λ_2	e_{A4}^1	λ_2	e_{A5}^1				
λ_3	e_{A3}^1	λ_3	e_{A6}^1	λ_3	e_{A7}^1				

Finally, we create table for three dimensional skeleton of this teacup, namely, X_1^3 , which has five boundaries. Therefore, we create 1 table $Table(B_A^3)$.

Table 3-6. Table for 3-dimensional skeleton of the teacup.

$Table(B_A^3)$	
∂B_A^3	X^2
π_1	e_{A1}^2
π_2	e_{A2}^2
π_3	e_{A3}^2
π_4	e_{A4}^2
π_5	e_{A5}^2

We combine all instance tables represented in each dimension to form CB-table. This table for the teacup is composed of one three-dimensional cell, five two-dimensional cells, and seven one-dimensional cells as shown in the following table.

Table 3-7. Cellular Structured Table for 3D Object.

Cell	Boundary				
e_A^3	e_{A1}^2	e_{A2}^2	e_{A3}^2	e_{A4}^2	e_{A5}^2
e_{A1}^2	e_{A1}^1		e_{A2}^1		e_{A3}^1
e_{A2}^2	e_{A1}^1		e_{A4}^1		e_{A6}^1
e_{A3}^2	e_{A2}^1		e_{A5}^1		e_{A7}^1
e_{A4}^2	e_{A6}^1				
e_{A5}^2	e_{A7}^1				
e_{A1}^1	e_{A1}^0				
e_{A2}^1	e_{A2}^0				
e_{A3}^1	e_{A1}^0		e_{A2}^0		
e_{A4}^1	e_{A1}^0		e_{A3}^0		
e_{A5}^1	e_{A2}^0		e_{A4}^0		
e_{A6}^1	e_{A3}^0				
e_{A7}^1	e_{A4}^0				

This table is created from the teacup composed from previous section.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

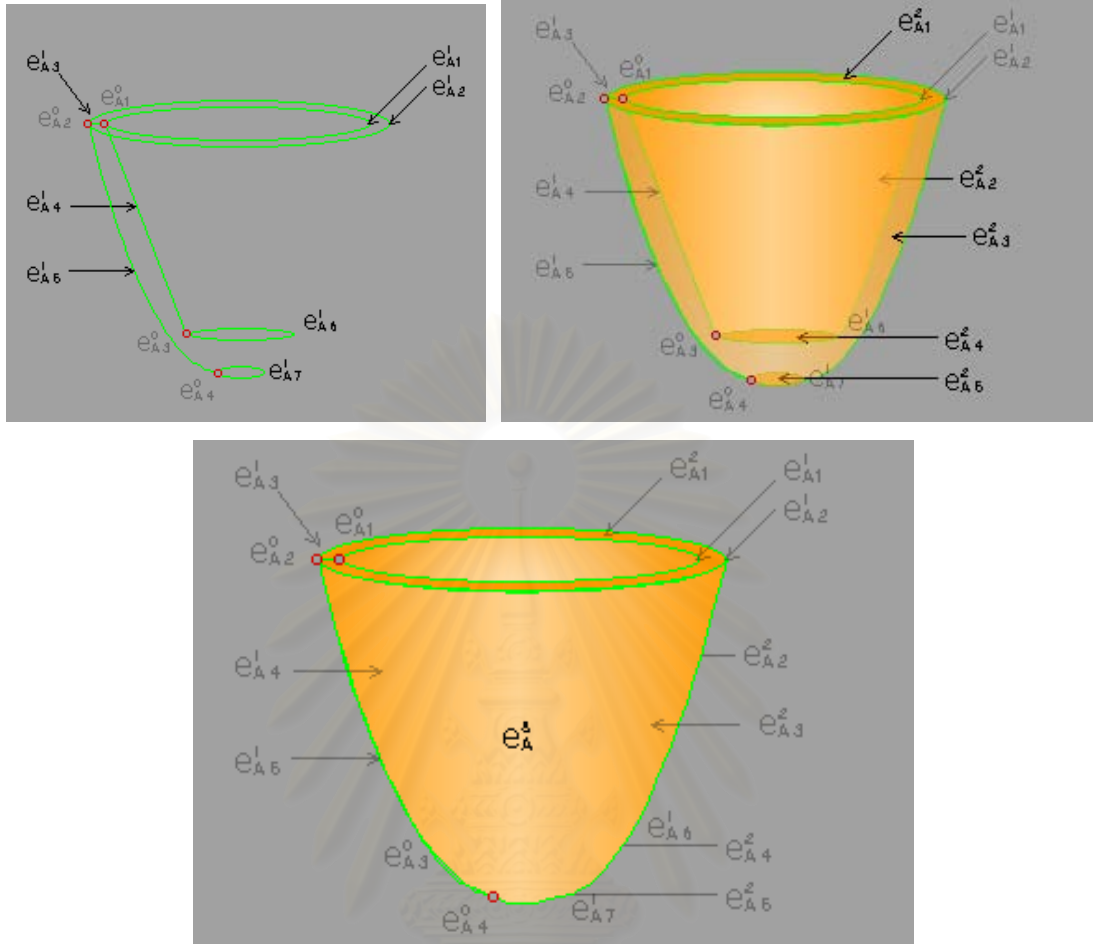
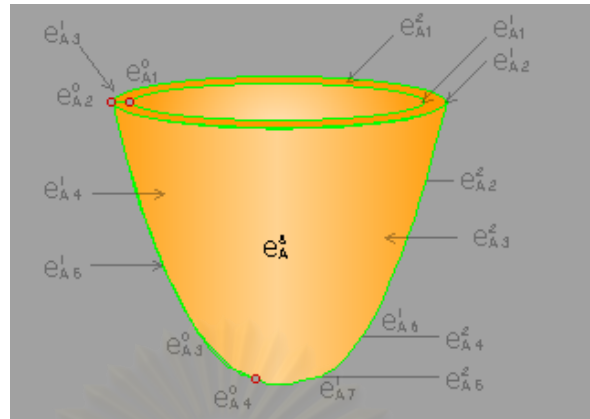


Figure 3-14. X_1^1, X_1^2 and X_1^3 structure of a tea cup.

3.5.2 Graph Conversion

Next, we show the graph representation of the cellular structured space model. Firstly, we create the rooted node, e_A^3 , which has four boundaries which are surface or 2-dimensional cell; $e_{A1}^2, e_{A2}^2, e_{A3}^2, e_{A4}^2$ and e_{A5}^2 , and these cell formed nodes in the lower level also have edges to their boundaries; $e_{A1}^1, e_{A2}^1, e_{A3}^1, e_{A4}^1, e_{A5}^1, e_{A6}^1$ and e_{A7}^1 . Other nodes and edges are constructed in the same manner.



Cell	Boundary				
e_A^3	e_{A1}^2	e_{A2}^2	e_{A3}^2	e_{A4}^2	e_{A5}^2
e_{A1}^2	e_{A1}^1		e_{A2}^1		e_{A3}^1
e_{A2}^2	e_{A1}^1		e_{A4}^1		e_{A6}^1
e_{A3}^2	e_{A2}^1		e_{A5}^1		e_{A7}^1
e_{A4}^2	e_{A6}^1				
e_{A5}^2	e_{A7}^1				
e_{A1}^1	e_{A1}^0				
e_{A2}^1	e_{A2}^0				
e_{A3}^1	e_{A1}^0		e_{A2}^0		
e_{A4}^1	e_{A1}^0		e_{A3}^0		
e_{A5}^1	e_{A2}^0		e_{A4}^0		
e_{A6}^1	e_{A3}^0				
e_{A7}^1	e_{A4}^0				

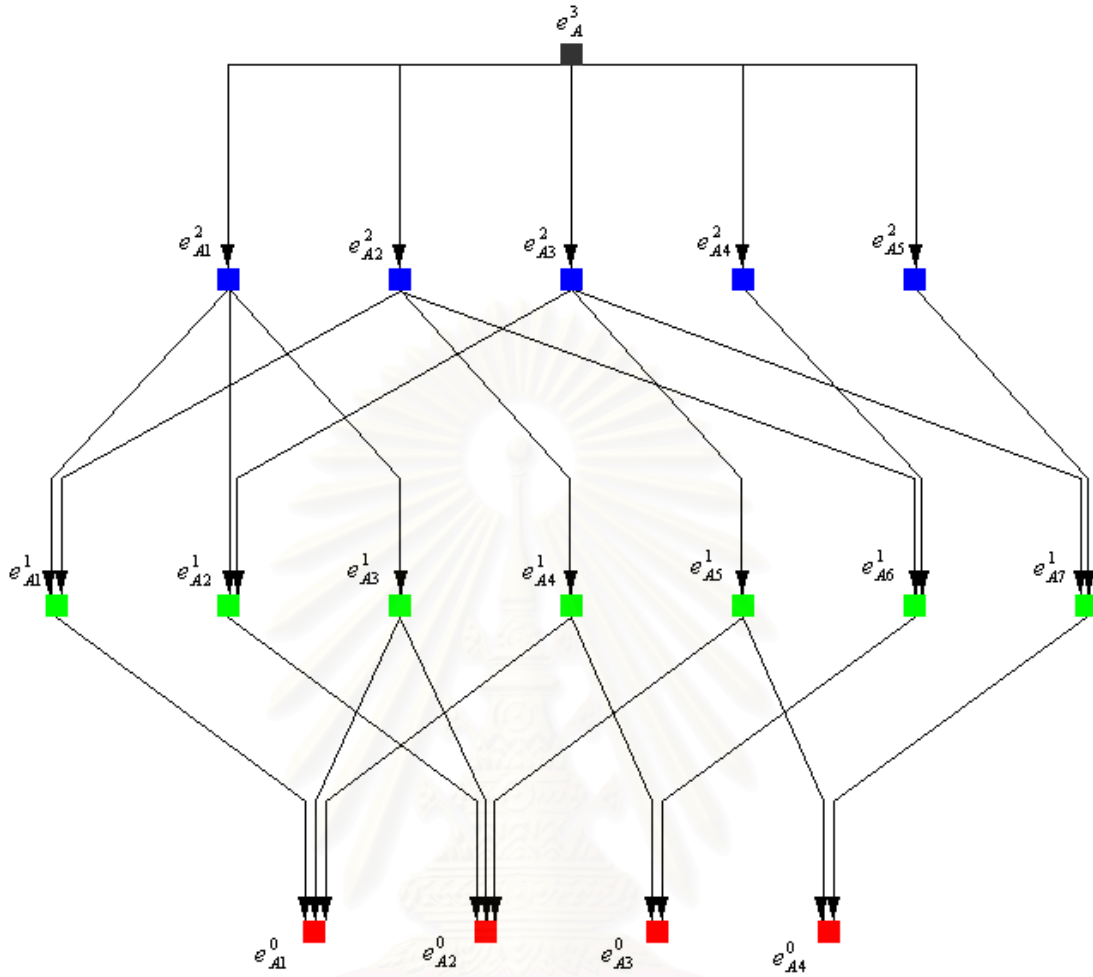


Figure 3-15. Graph representation of the cellular structured space for the teacup.

We define a directed graph created from cellular structured space which have one three dimensional cell, Three-Dimensional Graph, $TDG = (V, E, \nu, \varepsilon)$; where V is set of cells in the model; E is a set of edges where $\forall u, v \in V (u, v) \in E \Rightarrow v$ is a boundary of u . $\nu(\cdot)$ is a function define value of cell properties.

Next, we create graph, called Object Graph, $OBG = (V', E', \nu', \varepsilon')$ for representing the structure of attaching between TDG to compose objects; V' is a set of pointer to rooted node in TDG which attach to this object; E' is a set of edges where $\forall u, v \in V' (u, v) \in E' \Rightarrow$ object v and u are attached to each other; $\varepsilon'(\cdot)$ is a weight of the edge used for finding number of attaching cell between objects; $\nu'(\cdot)$ is function to locate cells which attach to the other object.

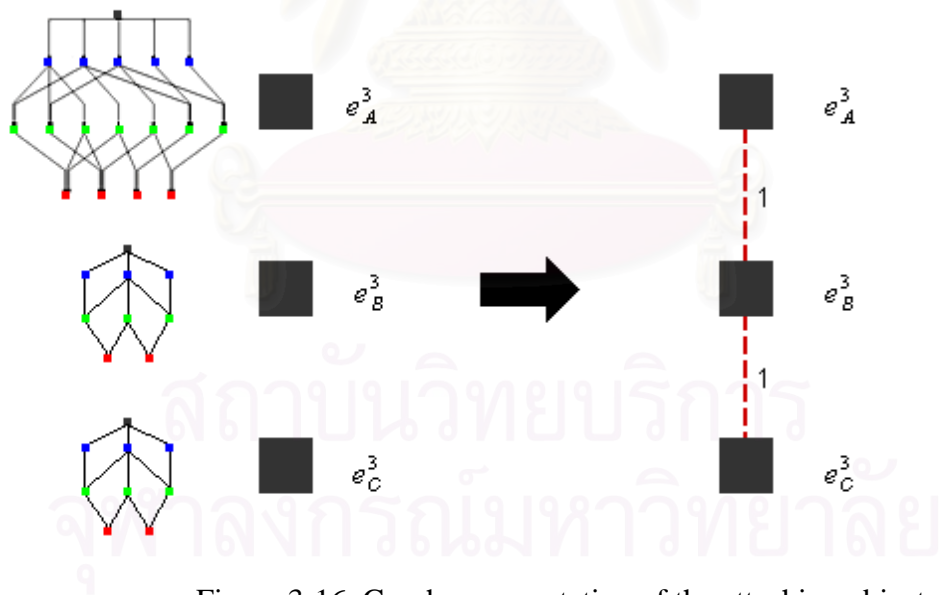
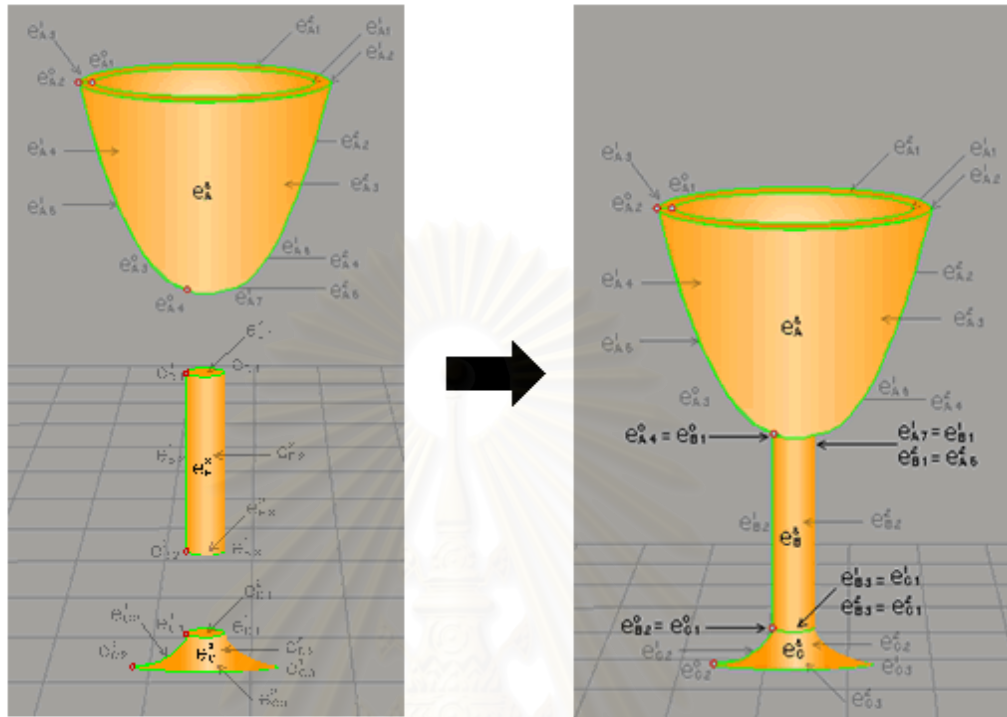


Figure 3-16. Graph representation of the attaching objects.

3.5.3 Similarity Assessment using CSS Model

We compose *TDG* to represent individual cellular structured space object with one 3D cell. *OBG* is a graph of cellular structured space object composed from attaching objects with one 3D cell together with edges in this graph shown how objects attach to each other. We use these graphs for finding similarity between two objects. We assess similarity between two *TDGs* and *OBGs* by algorithm described in the appendix. We match cell in the same dimension by finding the maximum cardinality and minimum weight matching in each dimension. The illustration of *TDG* matching is shown below.

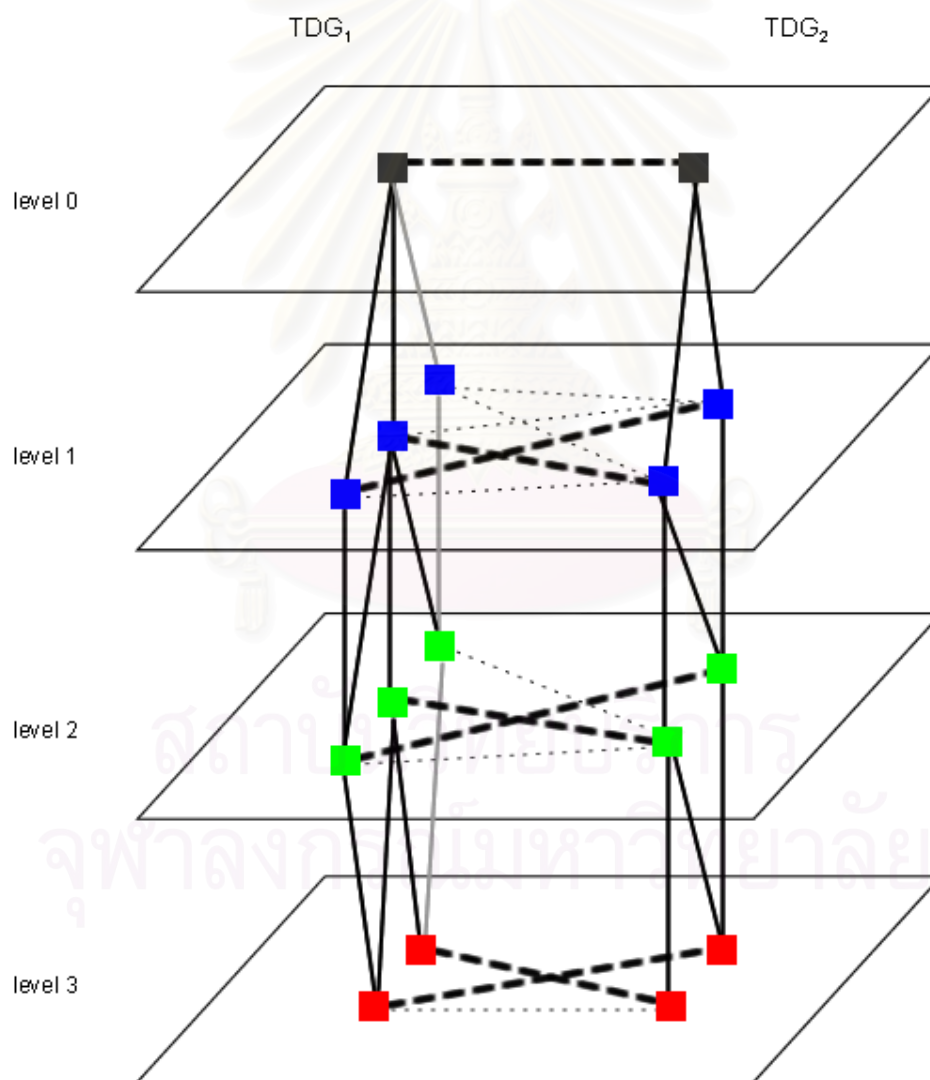


Figure 3-17. Illustration of the TDG matching algorithm.

CHAPTER IV

IMPLEMENTATION AND RESULTS

4.1 Environment

The proposed algorithm has been implemented in C++ with OpenGL library and MFC. The tests were performed on a Notebook with: An Intel Pentium4 M 2.0 GHz processor, RAM 1 GB, and hard disk 100 GB.

4.2 Implementation

We develop the three dimensional modeling application for composing cellular structured space model. This application can be used to model simple objects and attach them to form a complex one, and also converting structure of cellular structured space model into graph structure described in chapter 4.

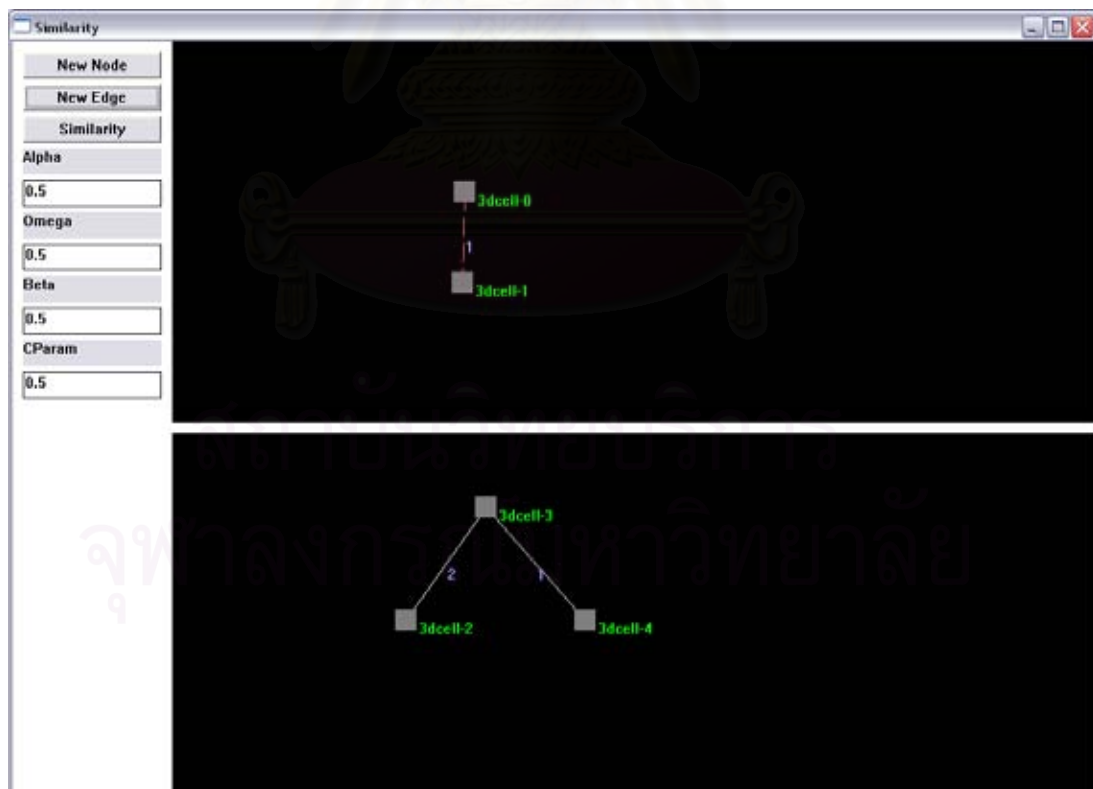


Figure 4-1. Nodes are 3D objects and edges are attaching relation.

This application composes objects by using concept of cellular structured space described in chapter 2. We incrementally construct skeleton in each dimension starting from zero dimension. Once we have zero dimension skeleton which is a set of points, we select two end points to form a line or a circle.

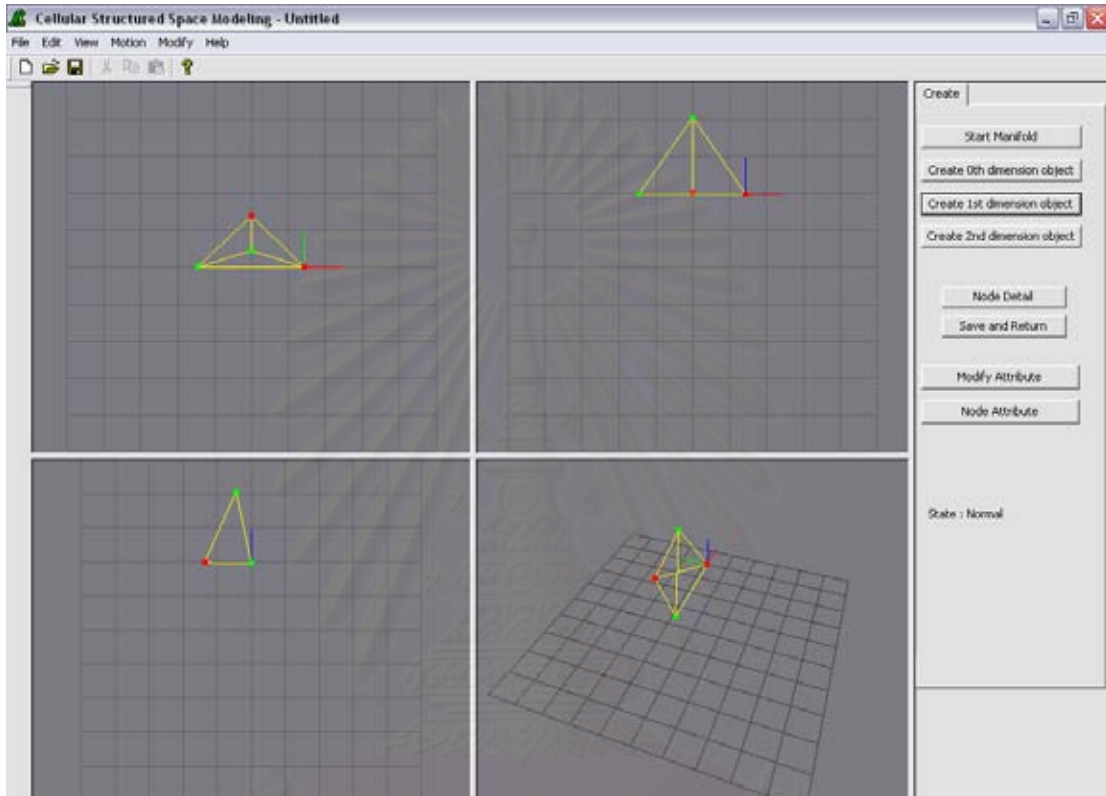


Figure 4-2. Modeling interface splitted into four views.

Moreover, we can edit, view, translate and attach objects by graph interface. We can select the perspective view or other views and enlarge to full screen.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

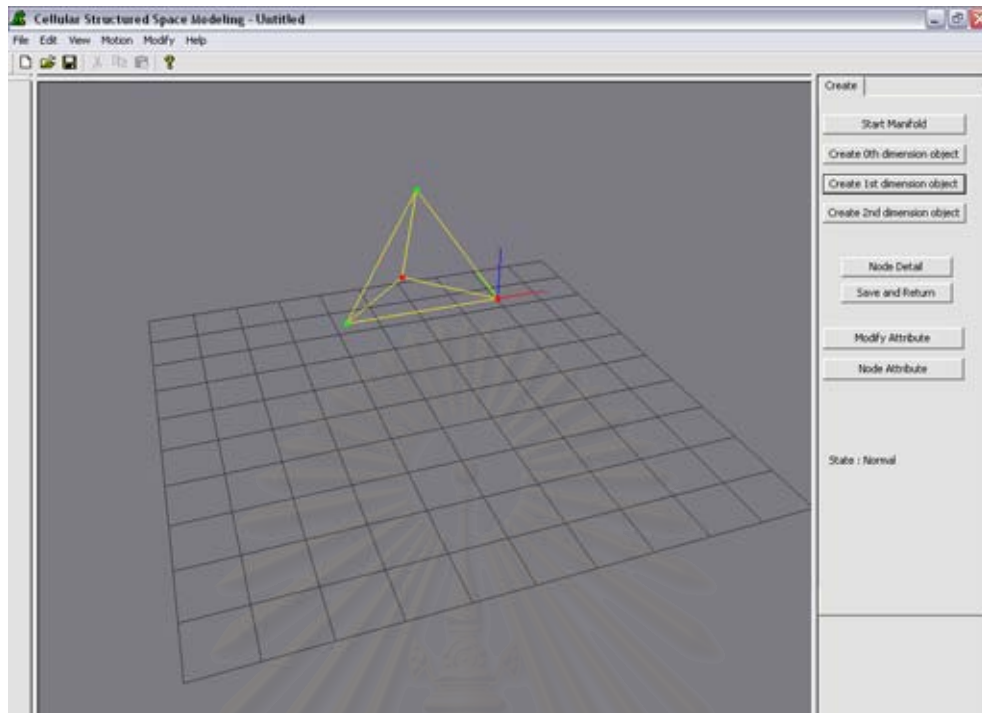


Figure 4-3. Full screen perspective views.

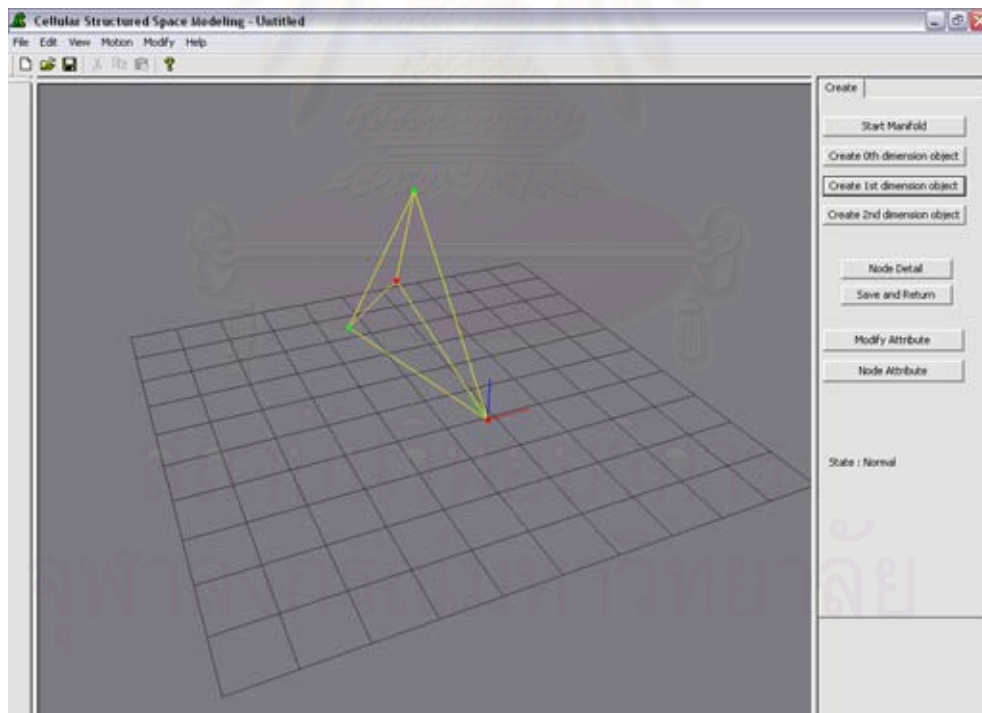


Figure 4-4. Translate 0D cell, which is a point.

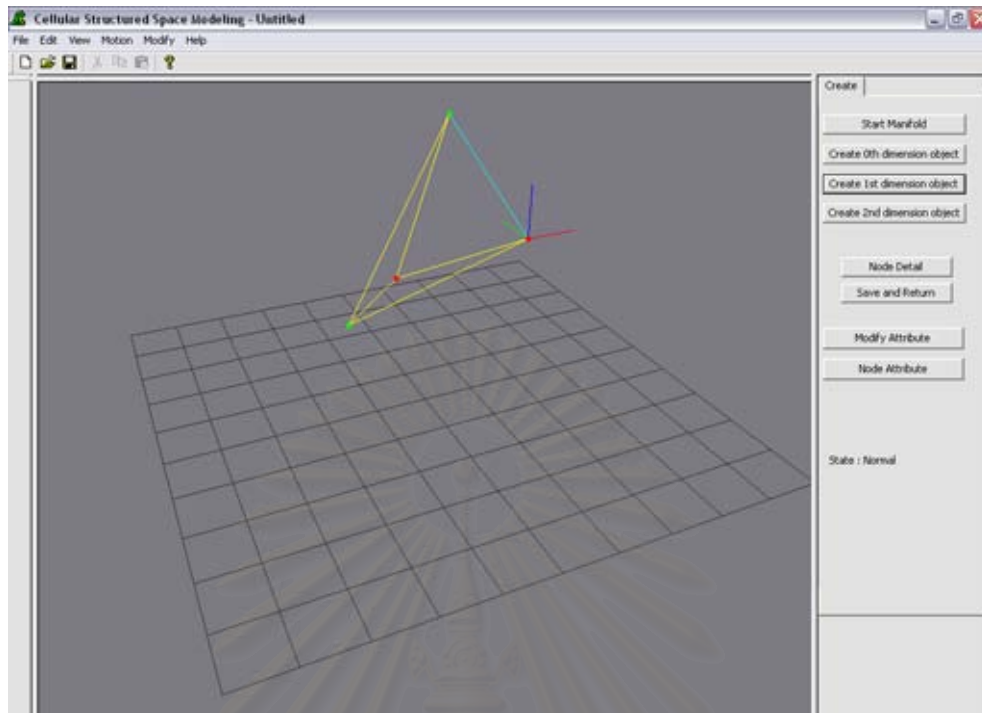


Figure 4-5. Translate 1D cell, which is a line.

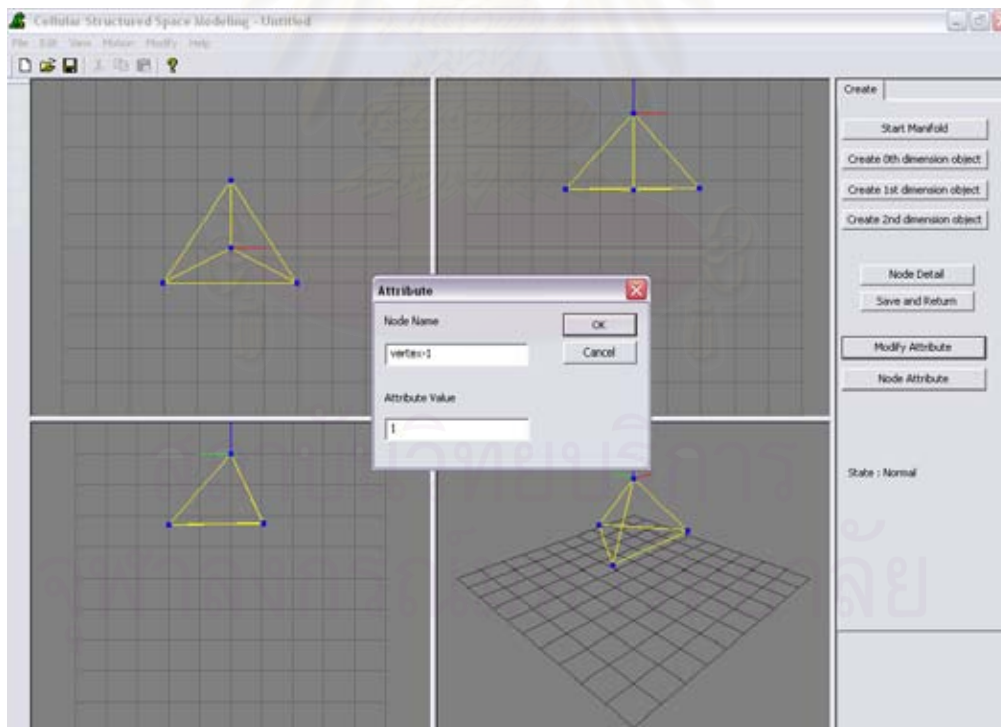


Figure 4-6. Define attribute to each cell for similarity assessment.

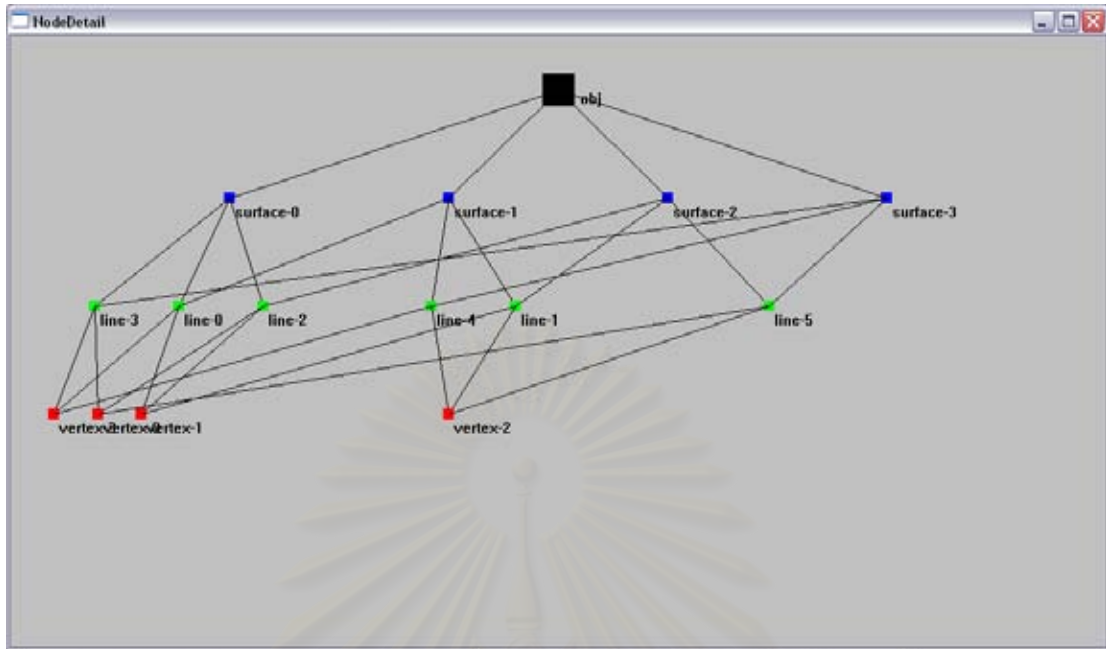


Figure 4-7. TDG for a tetrahedron.

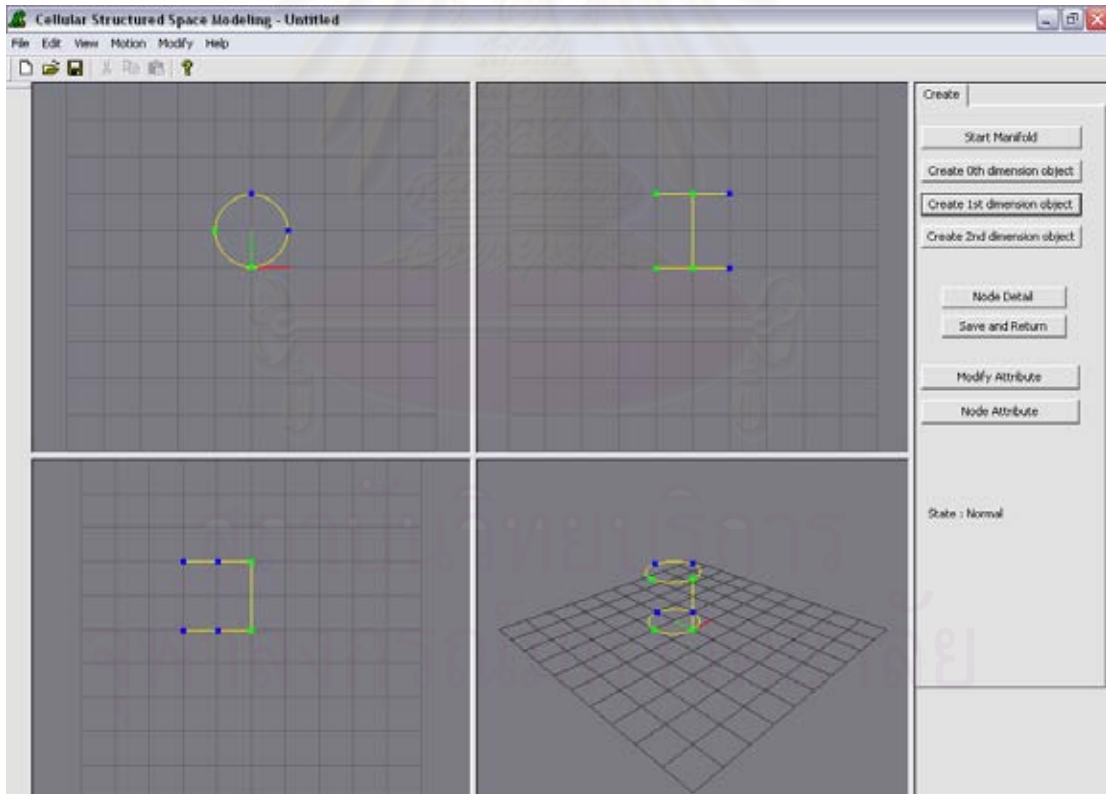


Figure 4-8. A 2D skeleton of cellular structured space cylinder.

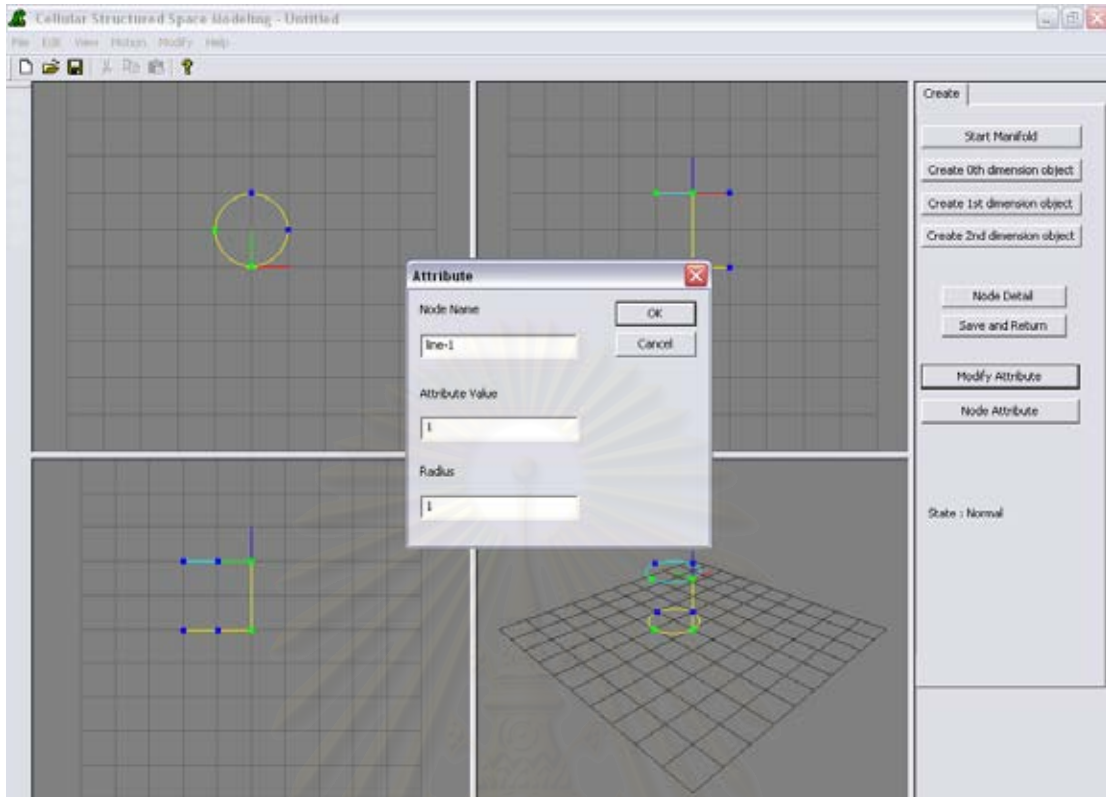


Figure 4-9. Adjust attribute for this 2D boundary circle such as radius.

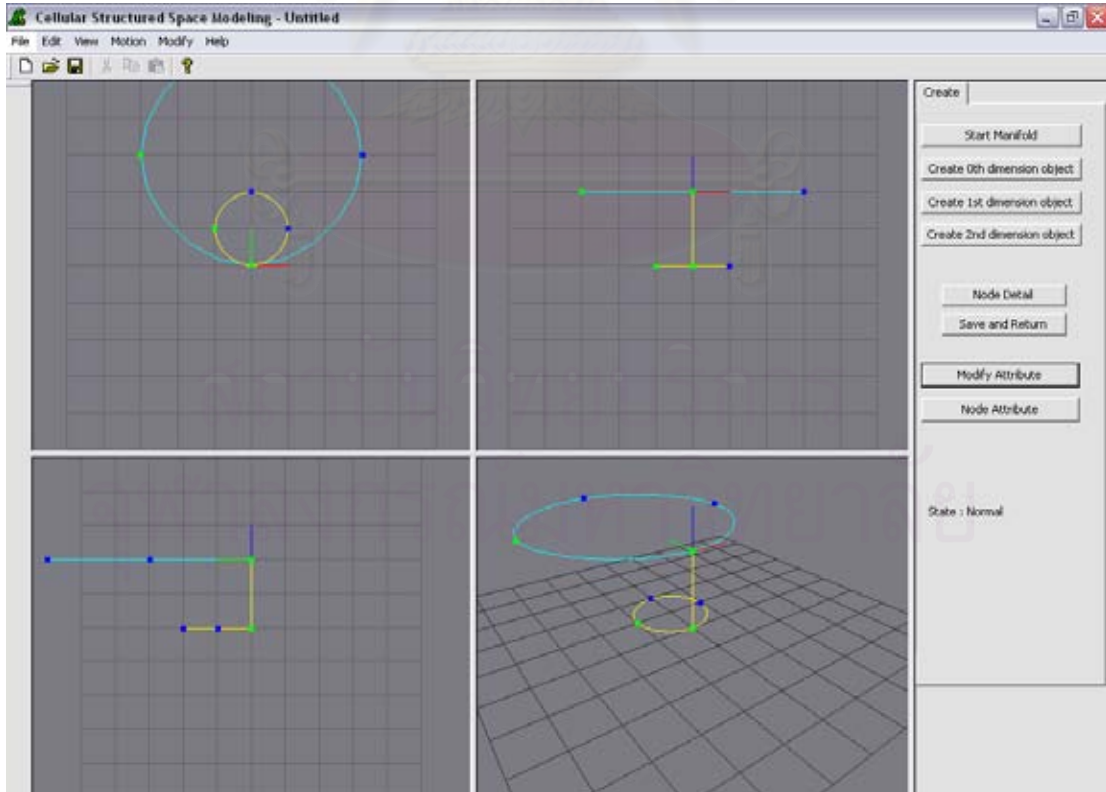


Figure 4-10. 2D boundary circle with different radius.

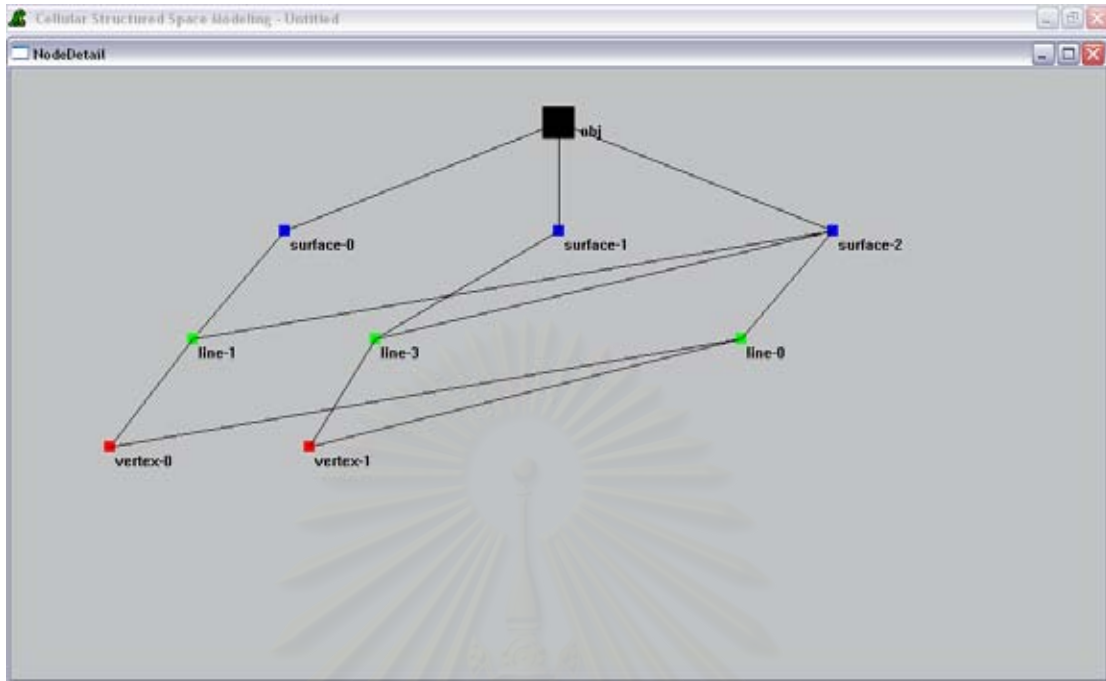


Figure 4-11. TDG for a cylinder.

After we create objects using this application, we can assess their similarity by using algorithm described in appendix.

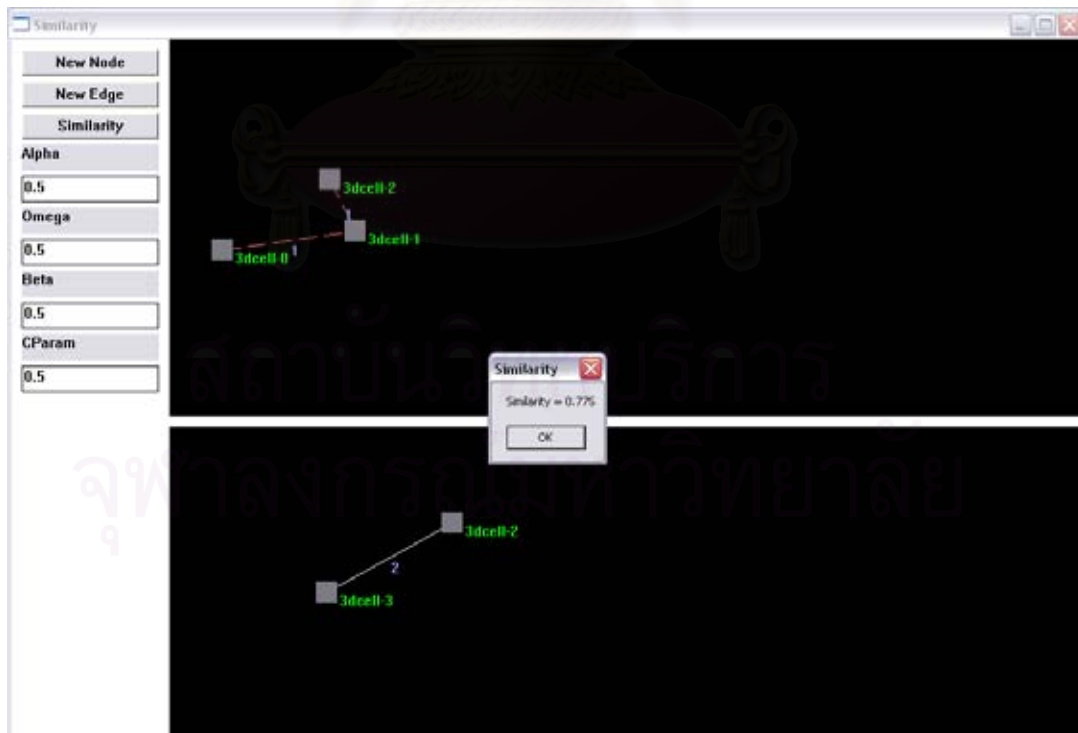


Figure 4-12. Calculating similarity between two objects.

4.3 Results

We implement the modeling system in the last section and show the validity of cellular structured space modeling technique in real world application. In this section we compute the similarity between two cellular structured space models composed from our system. In order to test the feasibility of the similarity assessment algorithm on cellular structured space models, we conducted and tested an experiment on the following objects.

Firstly, we find similarity between set of surface primitives; circle, triangle and square.

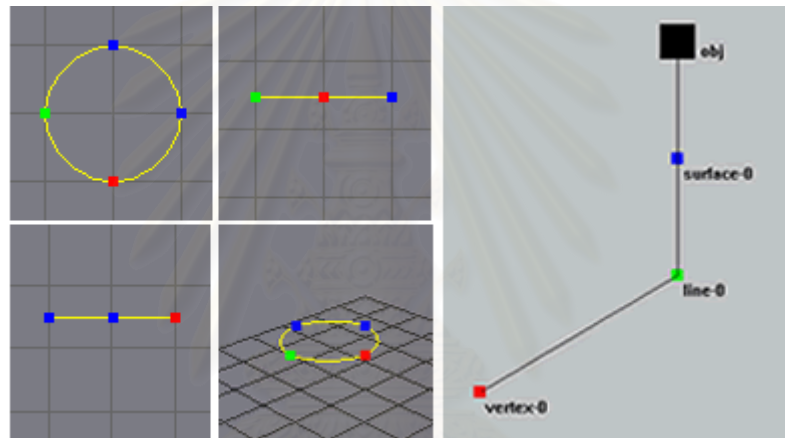


Figure 4-13. Object (a) is a circular surface.

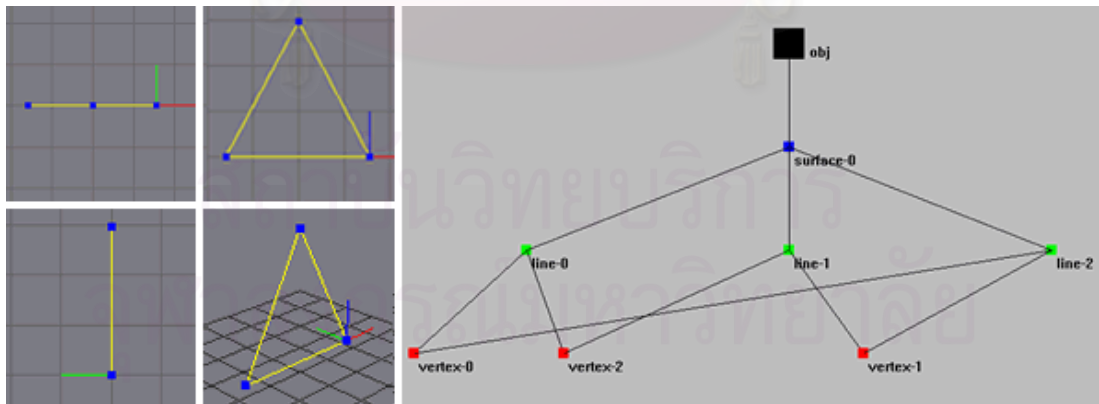


Figure 4-14. Object (b) is a triangular surface.

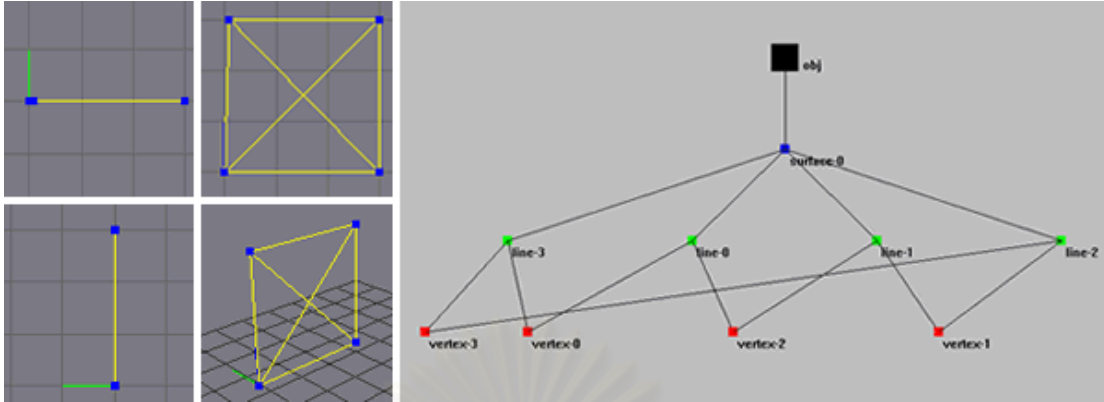


Figure 4-15. Object (c) is a square surface.

Table 4-1. Result of the first matching experiment; $\beta, \alpha, \omega, c = 0.5$.

	a	b	c
a	1.00	0.752778	0.724679
b	0.752778	1.00	0.924038
c	0.724679	0.924038	1.00

The similarity between the same object is always 1 as shown in diagonal line of Table 4-1. Triangle is more similar to square than circle.

Matched nodes between (a) circle and (b) triangle are 4/4 in (a) and 4/8 in (b), so the node similarity is 0.75, and matched edges between (a) and (b) is 3/3 and 3/10 respectively; the edge similarity is 0.65. Moreover, the degree fraction is 0.611111 between (a) and (b).

Matched nodes between (a) circle and (c) square are 4/4 in (a) and 4/10 in (c), so the node similarity is 0.7, and matched edges between (a) and (c) is 3/3 and 3/13 respectively; the edge similarity is 0.615385, and the degree fraction is 0.583333 between (a) and (c).

Finally, matched nodes between (b) triangle and (c) square are 8/8 in (b) and 8/10 in (c), so the node similarity is 0.9, and matched edges between (b) and (c) is 10/10 and 9/13 respectively; the edge similarity is 0.846154, and the degree fraction is 0.95 between (b) and (c).

Secondly, we conduct a similarity assessment experiment in the set of three dimensional primitives; tetrahedron, cone and cylinder.

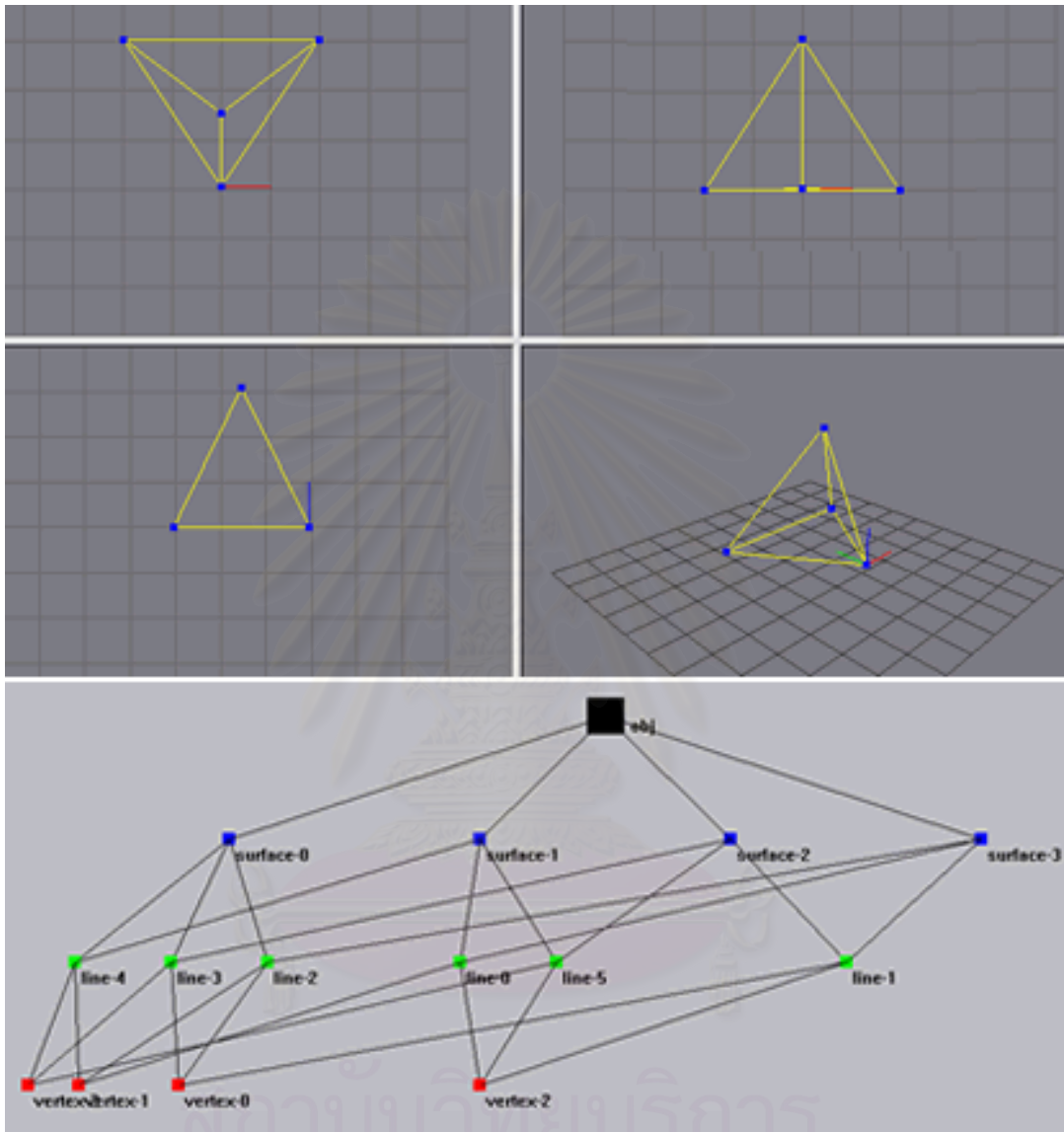


Figure 4-16. Object (a) is a tetrahedron.

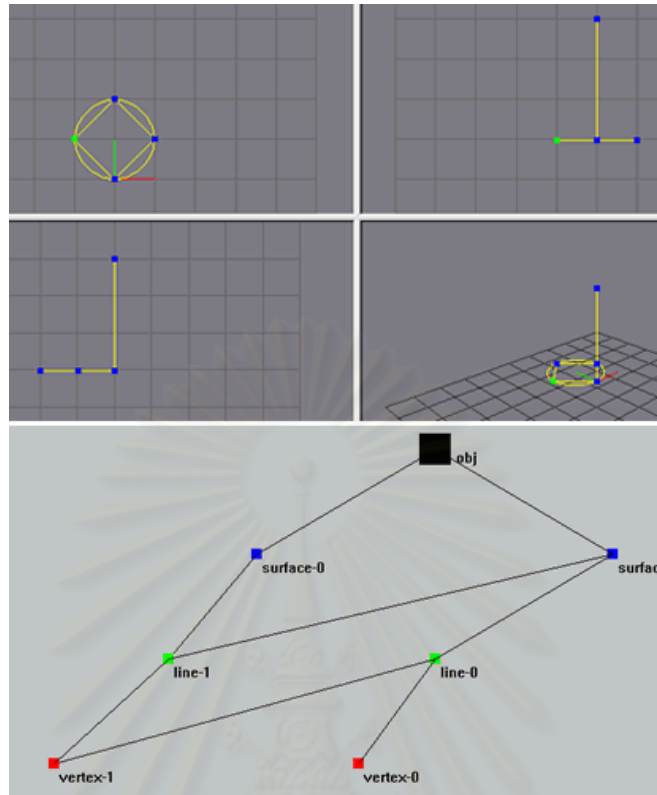


Figure 4-17. Object (b) is a cone.

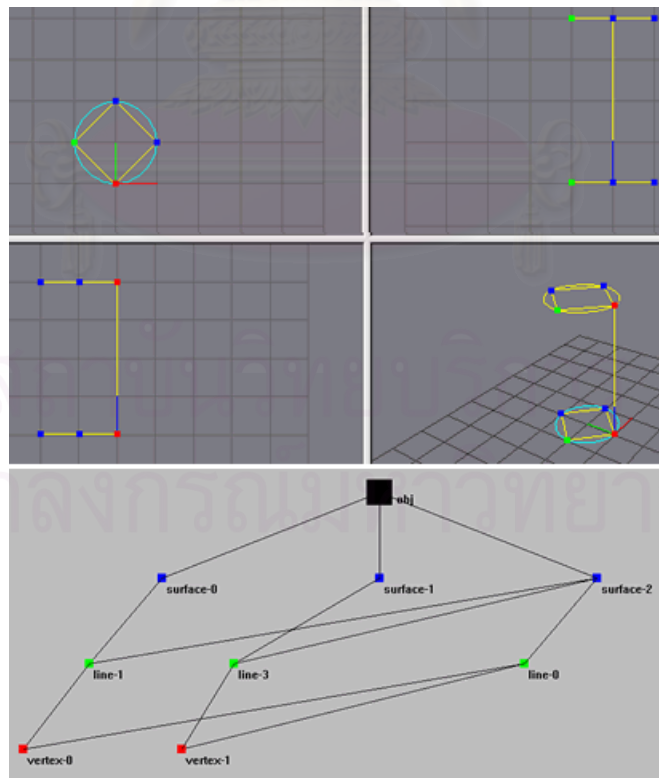


Figure 4-18. Object (c) is a cylinder.

Table 4-2. Result of the second matching experiment; $\beta, \alpha, \omega, c = 0.5$.

	a	b	c
a	1.00	0.744048	0.78631
b	0.744048	1.00	0.843056
c	0.78631	0.843056	1.00

The similarity between the same object is always 1 as shown in diagonal line of Table 4-2. Similarity value between cone and cylinder is more than any pairs, because its matched nodes and edges are maximum and degree fraction is high.

Matched nodes between (a) tetrahedron and (b) cone are 7/15 in (a) and 7/7 in (b), so the node similarity is 0.733333, and matched edges between (a) and (b) is 8/28 and 8/8 respectively; the edge similarity is 0.642857. Moreover, the degree fraction is 0.6 between (a) and (b).

Matched nodes between (a) tetrahedron and (c) cylinder are 9/15 in (a) and 9/9 in (c), so the node similarity is 0.8, and matched edges between (a) and (c) is 12/28 and 12/12 respectively; the edge similarity is 0.714286, and the degree fraction is 0.630952 between (a) and (c).

Finally, matched nodes between (b) cone and (c) cylinder are 7/7 in (b) and 7/9 in (c), so the node similarity is 0.888889, and matched edges between (b) and (c) is 8/8 and 6/12 respectively; the edge similarity is 0.75, and the degree fraction is 0.733333 between (b) and (c).

Moreover, we calculate the similarity between tetrahedron and cube and the result show that tetrahedron is similar to cube more than cone and cylinder.

Matched nodes between tetrahedron and cube are 15/15 in tetrahedron and 15/27 in cube, so the node similarity is 0.777778, and matched edges between tetrahedron and cube is 28/28 and 20/54 respectively; the edge similarity is 0.685185, and the degree fraction is 0.878788 between tetrahedron and cube

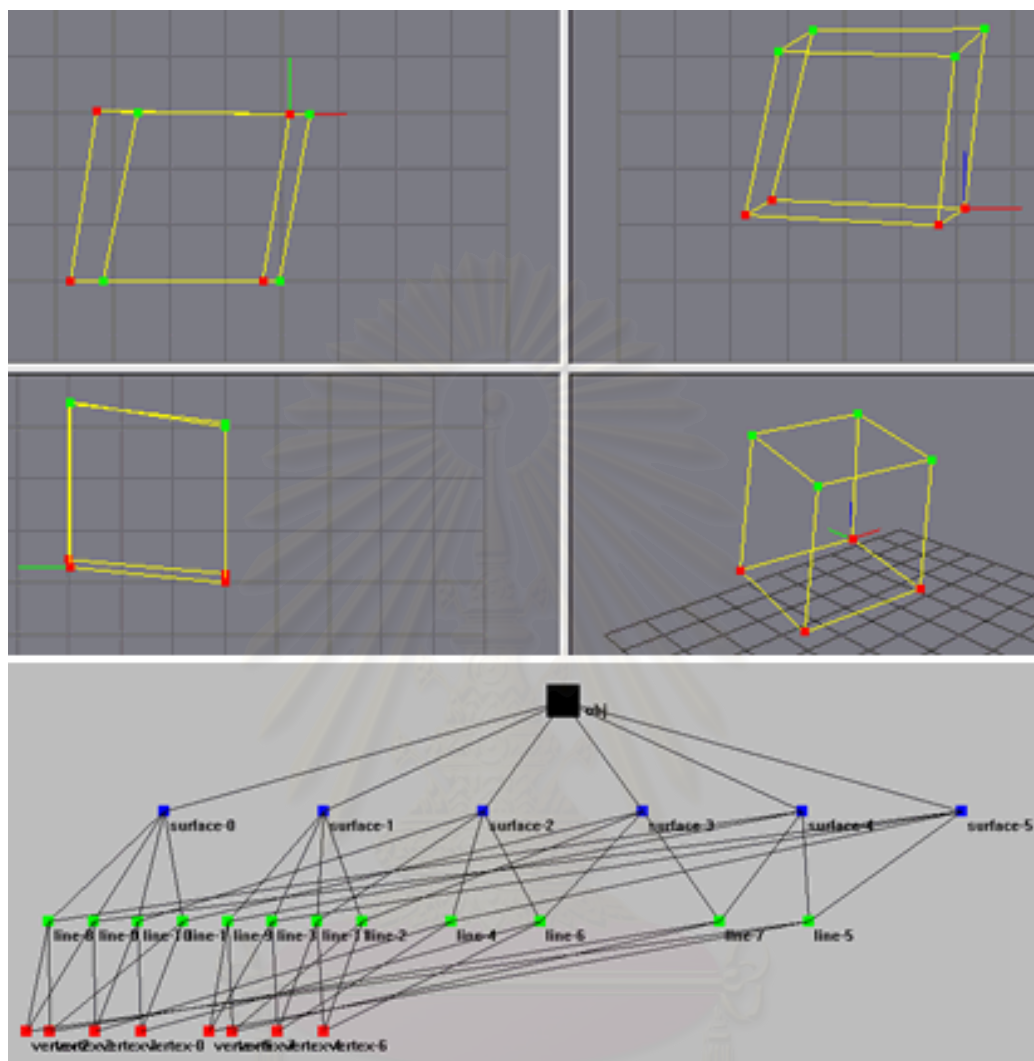


Figure 4-19. Cellular structured space model of a cube.

Next, we compare two cylinders with different attribute value in each dimensions; the attribute value in this research is uniform, range from 0-4, and concerned as a geometrical properties of the arbitrary cell.

We set the attribute of all cells in the first object to 4 and the second to 1, thus the similarity should be decrease and less than 1. The result shows that all nodes and edges are completely matched, degree fraction is also maximum, but the attribute difference is $5.4/9 = 0.6$; the attribute similarity is 0.4, and the similarity is 0.85. Furthermore, if we set $\omega = 0$, the similarity between the two cylinders with different attribute value in each dimensions is 0.7.

CHAPTER V

CONCLUSION, DISCUSSION AND FUTURE WORK

This research emphasizes on offering an idea, possibilities, algorithm and the experimental results in order to show a validity of using cellular structured space for modeling objects in computer graphics and finding their similarity as an example. The conclusion and discussion of this research in term of modeling method as well as future work are discussed in the following sections.

5.1 Conclusion

We propose a valid shape modeling using topology features for checking validity of the model called cellular structured space, starting from inductively compose a finite structure via attaching map. By this method, invariant and some topological properties are preserved and used for finding similarity of the 3D objects.

The experimental result obviously shows that we can compose the object using cellular structured space modeling to divide it into smaller cells. We can attach or detach the small cells, while topological properties of the model still remain. Then, we can easily define the new object using other objects.

It can reduce the redundancy of geometrical data. Comparing with several previous methods, we know that the redundancy is caused by the definition of fundamental objects. Therefore, further data collection is necessary to distinguish the difference. With the cellular structured space modeling, the structure of the object is defined as a collection of non-redundant attached cells in different dimensions.

Proposed modeling method offers more variety of shape. It can construct both fundamental and complicated objects, while other methods such as “Primitive instancing” and “sweeps”, restrict the shape of object by types of primitives.

Moreover, results of operation to cellular structured are always in cellular structured space. This reserves the validity of the model

We can model three-dimensional objects using cellular structured space concept starting from set of points, zero-dimensional cell, and construct a complete object by incrementally method via attaching higher dimensional cell to skeleton that already composed. Moreover, we have created the modeling application based on cellular structured space to show the concept and possibility of modeling object in the real world. Furthermore, we also created interface to find similarity between models created by this application.

The results gave in previous chapter show that the proposed algorithm can efficiently assess the similarity between cellular structured space models. However, the algorithm still needs improvement in some parts such as model partitioning before finding similarity, finding maximum cardinality minimum weight matching, finding maximum common subgraph, and determining whether selected lines can form a surface. In the experiment, we assume that models are partitioned by user in modeling process.

The similarity measure presented in this research is based on comparing two graphs structure created from cellular structured space of models. Using these graphs, their similarity is not only computed based on internal structure of graphs but also takes into account node and edge attribute and attaching between objects in the graph are concerned. In addition, we can set the value of ω, α or β up to specific similarity assessment purpose. α is a parameter in similarity function that weight between internal and external structure. If we desire to find only substructure in efficient way, we had better set α to zero and β to one. ω is a parameter for finding similarity in two *TDGs*, and weight between topology and attribute value of the two graphs.

Using this similarity measure, we achieved exactly what we expected. It is possible to compare three-dimensional computer graphic models on the basis of their cellular structured space graphs and their attaching graphs. Hence, assumptions about the similarity of the model's global structure can be made. However, the approach does not suit for a comparison of geometry features of the models.

5.2 Discussion

This modeling system can create and edit cellular structured space model; together with, displaying and collecting data structure of the composed cellular structured space model by using graph structure; *TDGs* and *OBG*. Moreover, we assess the similarity between two cellular structured space models by comparing these graphs. In this application, there are some limitations in term of the input and rendering or displaying three dimensional objects.

At this time, input objects loaded into program are cellular structured space object only. Converting general three dimensional objects to cellular structured space objects is our future work. Consequently, for finding similarity between two objects by using this proposed method, input objects have to be cellular structured space object composed by user either from our modeling application or type in text file and load into system.

Another limitation of our modeling system is rendering the surface. Composing surfaces, 2D cells, require more additional work, because we construct surface from arbitrary number of lines, 1D skeleton. We have to determine whether the set of selected lines can form the surface; thus, the editor neglect displaying surface in the editor view.

The interface of the modeling system is very suit for modeler, because we use the same lay out as some famous modeling applications such as Maya. This modeling method need some background knowledge about cellular structured space concept to model object correctly, however we can overcome this problem by including input procedure that convert three dimensional wire frame or mesh objects into cellular structured space model.

This system is applicable for searching and matching models in database by using our proposed graph structure of cellular structured space models as a search key with some minor changes.

Our similarity assessment algorithm based on finding similarity of two types of graphs. First, we have to assess similarity between two *TDGs* which is a hierarchical directed graph with four levels. We apply maximum cardinality minimum weight matching for bipartite graph to assess their similarity. The other is

the similarity between two objects that have object graph structure which show how *TDGs* are attached to form *OBG*. In this case, our algorithm is based on finding maximum common subgraph of two *OBGs*. We will discuss about our proposed strategy in this section.

We can improve the similarity assessment by using different and complex algorithm for finding maximum cardinality minimum weight matching in graph. The classic method is Edmond' blossom algorithm which can be implemented by integer programming methodology.

Graph structure in our research is collected as simple array with reference to its parent and child. Sometimes we can speed up an algorithm by using an efficient data structure that support the primitive operations used by the algorithm. In our case, the priority queue may be used.

Our problem is due to the similarity of graphs, which is generally referred to graph matching. Early approaches were restricted to find graph or subgraph isomorphism between two graphs. Graph or subgraph isomorphism are beneficial to find whether two objects are identical or one object is part of the other. Both graph and subgraph isomorphism are limited in their use because real world identical objects are usually not exactly match. Consequently, we use the maximum common subgraph of two graphs as a main theme for similarity measure.

Not only maximum common subgraph but also minimum common supergraph can be used. There are some researches showing that finding minimum common supergraph is the same as finding maximum common subgraph. Consequently, we choose only one of them to represent the similarity. In contrast, we will not use edit distance to measure similarity in our research. The main difficulty when applying edit distance methods to the three dimensional objects similarity problem, is that there is no clear meaning for the edit operations. This problem mean that the similarity measure based on graph edit distance will return a value with little physical significance. In our case, maximum common subgraph is suitable for being a part of similarity assessment for cellular structured space model.

There are numerous algorithms to determine the maximum common subgraph, however, two main algorithms are considered to be the fundamental. All other

algorithms follow the same idea, but have minor changes to enhance performance due to the specific application field.

The first algorithm was proposed by McGregor [54], searching for the maximum common subgraph by finding all common subgraphs of the two given graphs and then choose the graphs with the largest size. The second algorithm by Durand and Pasari is well known for reduction of the maximum common subgraph problem to maximum clique problem [55]. We follow the first algorithm because our *OBGs* usually have low edge-density which convenient search for maximum common subgraph by finding all the subgraphs of he two given graphs. Moreover, we can improve performance of our maximum common subgraph algorithm by using efficient pruning technique to delete node that no need to traverse.

Similarity function $SIM(TDG_1, TDG_2)$ use four parameters to calculate similarity between two *TDGs*; p_{node} and p_{edge} are used to assess how many nodes and edges are matched, p_{degree} and p_{attr} are used to find cell properties and attribute similarity. We let user to set ω to weight between node-edge and attribute similarity.

In our implementation, we set the node attribute to constant value with range four. Consequently, an attribute function are still needed to improve he algorithm by defining suitable attribute value to different cell type.

Similarity function $OBSim(OBG_1, OBG_2)$ use s_{ext} and s_{int} parameters to determine similarity between two *OBGs*. s_{ext} assess the ratio of matched nodes and edges to the graph size. s_{int} calculate the similarity of their matching node compare with the node that they should match and the different in attaching function as an edge error; where β is a weight defined by user for arbitrary weight edge or node similarity and α is a weight defined by user for arbitrary weight internal or external structure similarity.

5.3 Future Work

The application will be enhanced to convert objects from three dimensional mesh to cellular structured space and can edit these objects by this application. The presented algorithm could be further improved in the future by making use of; for example, a priority queue structure in maximum cardinality minimum weight matching. Furthermore, defining the primitive object may be useful for our approach.

In addition, it would be interesting to consider the distance in geometric space. This would increase the performance of the algorithm for simple geometry models. In this context, a threshold could be introduced to achieve more exact results. Likewise, it would be possible to include line vector methods that could provide a preprocessing of the models as a key search in database. In order to achieve results that are similar not only in their external structure but also in their surface details, other information would have to be included. This could, for example, be achieved by using a suitable attribute function.

The improvement of the maximum common subgraph and maximum cardinality minimum weight matching algorithm can be done in the near future because these two problems are prevalent in many researches.

REFERENCES

- [1] D. Harn and M. P. Baker. *Computer Graphics*. (n.p.) Prentice-hall International, 1994.
- [2] J. D. Foley, A. v. Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics Principle and Practice*. second ed. (n.p.) Addison-Wesley Publishing, 1997.
- [3] P. J. Schneider and D. H. Eberly. *Geometric Tools for Computer Graphics*. (n.p.) 2003.
- [4] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, vol. 153: Springer-Verlag New York, Inc., 2003.
- [5] L. Piegl and W. Tiller. *Te NURBS Book*. (n.p.) Springer, 1997.
- [6] Y. Shinagawa and T. L. Kunii. "Constructing Reeb Graph Automatically from Cross Sections," *IEEE Computer Graphics and Applications*, pp. 44-51, 1991.
- [7] T. L. Kunii. "Algebraic Topological Modeling for Cyberworld Design," presented at Second International Conference on Cyberworlds (CW'03) 2003.
- [8] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. "Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes," presented at Proc. ACM SIGGRAPH, 2001.
- [9] T. L. Kunii. "Graphics with Shape Property Inheritance," presented at Sixth Pacific Conference on Computer Graphics and Applications (PG'98), 1998.
- [10] J. R. Munkres. *Topology*, second ed. (n.p.) Prentice Hall, 2000.
- [11] M. L. O'Leary. *The Structure of Proof With Logic and Set Theory*. (n.p.), 2002.

- [12] A. Cardone, S. K. Gupta, and M. Karnik. "A Survey of Shape Similarity Assessment Algorithms for Product Design and Manufacturing Applications," *Journal of Computing and Information Science in Engineering*, vol. 3, pp. 109-118, 2003.
- [13] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. "A Search Engine for 3D Models," *ACM Transactions on Graphics (TOG)*, vol. 22, pp. 83 - 105 2003.
- [14] S.-F. Chang, Q. Huang, T. Huang, A. Puri, and B. Shahraray. "Multimedia Search and Retrieval," in *Advances in Multimedia: Systems, Standards, and Networks*, 1999.
- [15] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge. "Comparing Images Using the Hausdorff Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 850-863, 1993.
- [16] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. "Fast Multiresolution Image Querying.," presented at International Conference on Computer Graphics and Interactive Techniques 1995.
- [17] J. W. H. Tangelder and R. C. Veltkamp. "A Survey of Content Based 3D Shape Retrieval Methods," presented at International Conference on Shape Modeling and Applications 2004 (SMI'04) 2004.
- [18] R. C. Veltkamp and M. Hagedoorn. "State of the Art in Shape Matching," presented at Technical Report UU-CS-1999-27, 1999.
- [19] T. A. Cass. "Robust Affine Structure Matching for 3D Object Recognition," presented at IEEE Trans. PAMI, 1998.
- [20] M. Kortgen, G. J. Park, M. Novotni, and R. Klein. "3D Shape Matching with 3D Shape Context," presented at The 7th Central European Seminar on Computer Graphics, 2003.
- [21] S. Belongie, J. Malik, and J. Puzicha. "Shape Matching and Object Recognition using Shape Contexts," presented at PAMI, 2002.

- [22] D. Keim. "Efficient Geometry-based Similarity Search of 3D Spatial Databases," presented at Proc. ACM SIGMOD,, 1999.
- [23] C. Zhang and T. Chen. "Efficient Feature Extraction for 2D/3D Objects in Mesh Representation," presented at Proc. IEEE ICIP, 2001.
- [24] D. Vranic and D. Saupe. "3D Shape Descriptor based on 3D Fourier Transform," presented at Proc. EURASIP Conf. on Digital Signal Processing for Multimedia Communications and Services, 2001.
- [25] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. "Matching 3D Models with Shape Distributions," presented at Proc. Shape Modeling, 2001.
- [26] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. "Shape Distributions," presented at ACM Transactions on Graphics, 2002.
- [27] J. Vandeborre, V. Couillet, and M. Daoudi. "A Practical Approach for 3D Model Indexing by combining Local and Global Invariants," presented at Proc. Int'l Symp. on 3D Data Processing Visualization and Transmission (3DPVT'02), 2002.
- [28] D. Zhang and M. Hebert. "Harmonic Maps and Their Applications in Surface Matching," *Proc. IEEE CVPR*,, vol. 2, 1999.
- [29] D. Y. Chen, X. P. Tian, Y. T. Shen, and M. Ouhyong. "On Visual Similarity Based 3D Model Retrieval," presented at Eurographics 2003, 2003.
- [30] B. Horn. "Extended Gaussian images," presented at Proc. IEEE, 1984.
- [31] S. Biasotti. "Topological Techniques for Shape Understanding," Central European Seminar on Computer Graphics, 2001.
- [32] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien. "Surface Codings Based on Morse Theory," *IEEE Computer Graphics and Applications*, vol. 11, pp. 66-78, 1991.
- [33] J. C. Hart. "Morse Theory for Implicit Surface Modeling," presented at Mathematical Visualization, 1997.

- [34] S. Takahashi, Y. Shinagawa, and T. L. Kunii. "A Feature-based Approach for Smooth Surfaces," presented at Solid Modeling, Atlanta GA USA, 1997.
- [35] H. Sunder, D. Silver, N. Gagvani, and S. Dickinson. "Skeleton Based Shape Matching and Retrieval," presented at Proc. Shape Modeling and Apps., 2003.
- [36] J.-H. Chuang, C.-H. Tsai, and M.-C. Ko. "Skeletonization of Three-Dimensional Object using Generalized Potential Field," presented at IEEE Transaction on Pattern Analysis and Machine Intelligence, 2000.
- [37] P. Kanongchaiyos. "A Study of Transformation of 3D Objects Using Topological Analysis by Reeb Graph-based Models," in *The Graduate School of Science*, vol. Doctoral: The University of Tokyo, 2003.
- [38] G. Reeb. "On the Singular Points of a Completely Integrable Pfaff Form or of a Numerical Function," presented at Comptes Rendus Acad. Sciences Paris, 1946.
- [39] F. Lazarus and A. Verroust. "Level Set Diagrams of Polyhedral Objects," presented at Proc. ACM Solid Modeling and Applications, 1999.
- [40] P. Kanongchaiyos, T. Nishita, S. Yoshihisa, and T. L. Kunii. "Topological Morphing using Reeb Graphs," presented at Proceedings of the First International Symposium on Cyber Worlds(CW'02), 2002.
- [41] T. L. Kunii. "Topological Graphics," presented at Proceedings of Spring Conference on Computer Graphics 2001 (SCCG 2001), 2001.
- [42] D.-Y. Chen and M. Ouhyong. "A 3D Object Retrieval System Based on Multi-Resolutional Reeb Graph."
- [43] D. Bepalov, A. Shokoufandeh, W. C. Regli, and W. Sun. "Scale-Space Representation of 3D Models and Topological Matching," presented at Solid Modeling, Seattle Washington USA, 2003.

- [44] M. Yu, I. Atmosukarto, W. K. Leow, Z. Huang, and R. Xu. "3D Model Retrieval with Morphing-Based Geometric and Topological Feature Maps," presented at IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03), 2003.
- [45] T. L. Kunii. "Homotopy Modeling as World Modeling," presented at Proceeding of Computer Graphics International 1999 (CGI99), 1999.
- [46] T. L. Kunii. "Computational Shape Modeling: Valid vs. Invalid," presented at Proceedings of International Conference on Shape Modeling and Application '99 (SMI99), 1999.
- [47] T. L. Kunii. "Valid Computational Shape Modeling: Design and Implementation," *International Journal of Shape Modeling*, pp. 123 - 133, 1999.
- [48] K. Ohmori and T. L. Kunii. "A Homotopy Model for Cup Lifting," presented at Proceeding of Computer Graphics International 2000 (CGI'00), 2000.
- [49] T. L. Kunii. "Implementation of Object Attachments by Cellular Modeling," presented at Proceedings of CG International 2001 (CGI'01), 2001.
- [50] K. Ohmori and T. L. Kunii. "Shape Modeling Using Homotopy," presented at International Conference on Shape Modeling & Applications, 2001.
- [51] T. L. Kunii. "Topological Graphics," presented at 17th Spring Conference on Computer Graphics (SCCG '01), 2001.
- [52] T. L. Kunii. "Web Information Modeling: The Adjunction Space Model," presented at 2nd International Workshop on Database in Networked Information System (DNIS 2002), 2002.



APPENDIX

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Maximum cardinality minimum weighted bipartite matching

Given two graph G_1 and G_2 , $H = (G_1, G_2, E)$ is a weighted bipartite graph with weight matrix $W = [w_{u,v}]$ of size $|G_1| \times |G_2|$ if, for all edges of the form $(u, v) \in E, u \in G_1, v \in G_2$ and (u, v) has an associated weight = $w_{u,v}$ or $W[u, v]$. Solving the maximum cardinality minimum weight matching in H solves an optimization problem which tries to minimize total edge weight on one hand while trying to maximize total number of matched nodes in the matching matrix on the other hand. The input consists of a undirected graph $H = (G_1, G_2, E)$. The vertices represent nodes or objects in our case, and each edge represents the possibility that its endpoints are matched. A matching M is a subset of the edges such that no two edges in M share a vertex. We first introduce the maximum cardinality matching in bipartite graphs. In this problem, the vertices are partitioned into two set, and an edge can only join between this two sets. We look for a matching with the maximum cardinality.

Given a matching M in a bipartite graph $H = (G_1, G_2, E)$, a simple path in H is called a augmenting path with respect to matching M if its two endpoints are both unmatched, and its edge are alternatively in $E - M$ and in M .

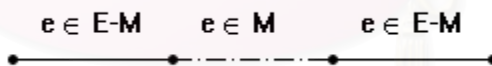


Figure A-1. Alternating path.

A matching M is not maximum if and only if there exists an augmenting path relative to M . The following algorithm use this truth to find a maximum cardinality matching for bipartite graphs.

1. M is an arbitrary matching. All nodes are unlabeled and unscanned.
2. If no nodes are exposed, a node v is said to be exposed relative to M if no edges of M meets v , and unlabeled, the current matching is maximum. Otherwise choose an exposed and unlabeled node r . Label it $(E, -)$.

3. Choose a labeled and unscanned node i .
 - 3.1. If it is even, let $J = \{j \in V : j \text{ is an unlabeled neighbor of } i\}$. Label all $j \in J$ with (O, i) . Node i is scanned; go to step 4.
 - 3.2. If i is odd and exposed, go to step 5.
 - 3.3. If i is odd and not exposed, label the node joined to i by a matching edge (E, i) . Node i is scanned; go to step 4.
4. If there is a labeled and unscanned node, go to step 3; otherwise go to step 2.
5. Use the second components of the labels to identify the augmenting path from node r to i . Remove all labels, update the matching, and return to step 2.

For the matching M . Its total weight of matching M is defined as

$$w(M) = \sum_{e \in M} w(e)$$

Let M' be a set of edges, then an incremental weight $\Delta M'$ is defined as the total weight of the unmatched edges in M' minus the total weight of the matched edges in M' .

$$\Delta M' = w(M' - M) - w(M' \cap M)$$

We find the augmenting path p with respect to M , which increase the net change in the weight of the matching after augmenting p .

We can use an algorithm described above to find a maximum cardinality minimum weight matching. Initially, we change the minimum weight matching problem to maximum weight matching problem by finding the maximum weight in M ; w_{\max} , and set new weight to W as followed;

$$w'(e) = w_{\max} - w(e).$$

At each iteration, the matching M is increased by finding an augmenting path of maximum incremental weight. We can search for all possible alternating paths from unmatched vertices simultaneously in a breath-first manner.

Moreover, we can use algorithm for solving assignment problem to find maximum weight matching for bipartite graph (also minimum weight matching). This algorithm, namely Munkres' Assignment Algorithm or Hungarian Algorithm, describes to the manual manipulation of a two-dimensional matrix by starring and priming zeros and by covering and uncovering rows and columns. This algorithm is an another choice for finding maximum weight matching in graph.

1. Create an $n \times m$ matrix called the cost matrix in which each element represents the cost of assignment. Rotate the matrix so that there are at least as many rows as columns and let $k = \min(n, m)$.
2. For each row of the matrix, find the smallest element and subtract it from every element in its row. Go to Step 3.
3. Find a zero in the resulting matrix. If there is no starred zero in its row or column, star Z . Repeat for each element in the matrix. Go to Step 4.
4. Cover each column containing a starred zero. If K columns are covered, the starred zeros describe a complete set of unique assignments. In this case, Go to step 8, otherwise, Go to Step 5.
5. Find a noncovered zero and prime it. If there is no starred zero in the row containing this primed zero, Go to Step 6. Otherwise, cover this row and uncover the column containing the starred zero. Continue in this manner until there are no uncovered zeros left. Save the smallest uncovered value and Go to Step 7.
6. Construct a series of alternating primed and starred zeros as follows. Let Z_0 represent the uncovered primed zero found in Step 5. Let Z_1 denote the starred zero in the column of Z_0 . Let Z_2 denote the primed zero in the row of Z_1 . Continue until the series terminates at a primed zero that has no starred zero in its column. Unstar each starred zero of the series, star each primed zero of the series, erase all primes and uncover every line in the matrix. Return to Step 4.

7. Add the value found in Step 5 to every element of each covered row, and subtract it from every element of each uncovered column. Return to Step 5 without altering any stars, primes, or covered lines.
8. Assignment pairs are indicated by the positions of the starred zeros in the cost matrix. If $C(i, j)$ is a starred zero, then the element associated with row i is assigned to the element associated with column j .

Similarity between TDGs

In this section, we describe an algorithm, which find the similarity between two TDGs; $TDG_1 = (V_1, E_1, f_{v_1}, f_{e_1})$ and $TDG_2 = (V_2, E_2, f_{v_2}, f_{e_2})$. The main theme of this algorithm is to find a maximum cardinality minimum weight matching between vertices of TDG_1 and TDG_2 which lie in the same dimension and the similarity function, $SIM(TDG_1, TDG_2)$, which is calculated from a real-valued function, sim_{top}, sim_{mat} , defined on the set of all partial mappings matrix, M . Our error function consists of two main parts with respect to any partial mapping matrix. On defining the similarity function, we would like to increase similarity in term of their topology and node attribute, and we would like to decrease similarity the more they exclude nodes from the two TDGs.

$$\begin{aligned}
 SIM(TDG_1, TDG_2) &= \omega sim_{top} + (1 - \omega) sim_{mat} \\
 sim_{top} &= (p_{node} + p_{edge}) / 2, \quad sim_{mat} = (p_{degree} + p_{attr}) / 2 \\
 p_{node} &= \frac{1}{2} \left(\frac{\sum_{u \in V_1} \sum_{v \in V_2} M[u, v]}{(|V_1| + |V_2|)} \right), \quad p_{edge} = \frac{1}{2} \left(\frac{edge_{mat}^1}{|E_1|} + \frac{edge_{mat}^2}{|E_2|} \right) \\
 edge_{mat}^k &= \# \{ (u, v) \in E_k \mid \forall_i (u, i) = 1 \text{ in } M \Rightarrow \exists_j (v, j) = 1 \text{ in } M \} \\
 p_{degree} &= \sum_{i=0}^3 \left(\frac{\sum_{u \in V_1, v \in V_2 \text{ in level } i} M[u, v] \left(\frac{\min(d^+(u), d^+(v))}{\max(d^+(u), d^+(v))} \right)}{\text{number of node in level } i} \right)
 \end{aligned}$$

$$P_{attr} = \sum_{i=0}^3 \left(\frac{\sum_{u \in V_1, v \in V_2 \text{ in level } i} M[u, v] |f_{v_1}(u) - f_{v_2}(v)|}{\text{range of attribute value in level } i} \right)$$

where $0 \leq \omega \leq 1$ is a weight defined by user for arbitrary weight topology or attribute similarity; default value of ω is 0.5 and M is partial mappings matrix with maximum cardinality, minimum weight matching between TDG_1 and TDG_2 . We define the partial mapping matrix M , between TDG_1 and TDG_2 , $|V_1| \times |V_2|$, $\{0,1\}$ matrix as follows:

$$M[u, v] = \begin{cases} 1 & \text{if } u \in TDG_1, v \in TDG_2, u \text{ match } v \\ 0 & \text{Otherwise} \end{cases}$$

Obviously, in the case of perfect similarity, $SIM(TDG_1, TDG_2) = 1$, while $SIM(TDG_1, TDG_2) = 0$ if there is no match.

p_{node} is a ratio of matched nodes to number of node in TDG_1 and TDG_2 .

p_{edge} is a ratio of matched edges to number of edge in TDG_1 and TDG_2 .

p_{degree} is a ratio between degree of matched nodes to in TDG_1 and TDG_2 .

p_{attr} is a ratio of attribute difference value of matched nodes to attribute range value in TDG_1 and TDG_2 in level i .

Solving for a maximum cardinality, minimum weight matching involves choosing a subset of the edges in the bipartite graph which provide a one-to-one mapping, whose sum edge weights (distance) is small, and whose cardinality is high

$TDGs$ can categorize cells or vertices into 4 levels; 3D cell in level 0, 2D cells in level 1, 1D cells in level 2, and 0D cell in level 3. The similarity between $TDGs$ is determined by the algorithm implemented from the following steps.

1. Match 3D cell at level 0 in TDG_1 and TDG_2 , namely M_0 .
2. Algorithm starts with the 2-dimensional cells in level $i = 1$.
3. In phase i , the algorithm considers the $3 - i$ D cells or vertices in level i .

4. Let U_i and V_i is the set of $3-i$ D cells or vertices of level i in TDG_1 and TDG_2 respectively.
5. Construct a weighted bipartite graph $G = (U_i, V_i, E)$ of which (u, v) is an edge of G if they have at least one matched parent in level $i-1$; basically, there is a parent $p(u)$ of u and $p(v)$ of v such that $(p(u), p(v)) \in M_{i-1}$.
6. We define weight of the edge (u, v) $u \in U_i, v \in V_i$ to be $|d^+(u) - d^+(v)| + |f_{v_1}(u) - f_{v_2}(v)|$ where $d^+(\cdot)$ is the number of edge pointed to child node in level $i+1$ and $f_{v_j}(\cdot)$ is value of cell properties or attribute in TDG_j . Note that if $i=3$ then $\forall u \in U_i, V_i$ $d^+(u) = 0$, because vertex has no boundary, and if the weight is equal who choose the matching due to the edge ratio, $\frac{d^+(u)}{d^+(v)}, d^+(u) \leq d^+(v)$.
7. We find a maximum cardinality, minimum weight matching M_i in G .
8. If $i < 3$, move to next level (go to step 3 and $i++$). The algorithm terminates after 3 phases.
9. We compute the partial mapping M from the union of all M_i values for $i = 0, \dots, 3$.
10. Finding similarity between TDG_1 and TDG_2 by $SIM(TDG_1, TDG_2)$ and the partial mapping M .

$$SIM(TDG_1, TDG_2) = \omega sim_{top} + (1 - \omega) sim_{mat}$$

$$sim_{top} = (p_{node} + p_{edge}) / 2, sim_{mat} = (p_{degree} + p_{attr}) / 2$$

$$p_{node} = \frac{1}{2} \left(\frac{\sum_{u \in V_1} \sum_{v \in V_2} M[u, v]}{(|V_1| + |V_2|)} \right), p_{edge} = \frac{1}{2} \left(\frac{edge_{mat}^1}{|E_1|} + \frac{edge_{mat}^2}{|E_2|} \right)$$

$$edge_{mat}^k = \# \{ (u, v) \in E_k \mid \forall_i (u, i) = 1 \text{ in } M \Rightarrow \exists_j (v, j) = 1 \text{ in } M \}$$

$$p_{degree} = \sum_{i=0}^3 \left(\frac{\sum_{u \in V_1, v \in V_2 \text{ in level } i} M[u, v] \left(\frac{\min(d^+(u), d^+(v))}{\max(d^+(u), d^+(v))} \right)}{\text{number of node in level } i} \right)$$

$$P_{attr} = \sum_{i=0}^3 \left(\frac{\sum_{u \in V_1, v \in V_2 \text{ in level } i} M[u, v] |f_{v_1}(u) - f_{v_2}(v)|}{\text{range of attribute value in level } i} \right)$$

Similarity between OBGs

The similarity between objects after they are already attached can perform by determining the maximum common subgraph of two *OBGs* with four concerns;

- Maximum node
- Edge attribute and minimum degree difference
- Maximum $SIM(TDG_1, TDG_2)$ for each *TDG* matched

In this section we present an algorithm to determine the similarity between two objects composed from cellular structured space method. Inputs are two objects with their *OBG* of which node is a pointer to its *TDGs* and edge is a number of attachment to this node, $OBSim(OBG_1, OBG_2)$.

1. First, we create matrix S of size $|V_{OBG_1}| \times |V_{OBG_2}|$ where each elements defined as follow:

$$S[u, v] = SIM(u, v) \quad \forall u \in OBG_1, v \in OBG_2$$

2. Starting from one node from each graph.
3. We attempt to enlarge the current common subgraph called current state, by adding feasible matching pairs, which are connected to the current common subgraph.
4. If there are multiple feasible pairs exist in the current state, we follow one pair and save the decision state for backtracking.
5. The subgraph detection ends when there is no more pairs are feasible for each state, and every state has been backtracked.
6. We select the common subgraph with the maximum node, maximum SIM value between matching node, and minimum edge error respectively.
7. We call the maximum common subgraph $\hat{G} = (\hat{V}, \hat{E}, \hat{v}, \hat{\varepsilon})$
8. We calculate the similarity between two *OBGs* by concerning external and internal structure similarity;

8.1. For external structure similarity, we calculate the average ratio between number of node and edge in \hat{G} to those in OBG_1 and OBG_2 as followed;

$$s_{ext} = \frac{1}{2} \left(\frac{c|\hat{V}| + (1-c)|\hat{E}|}{c|V_{OBG_1}| + (1-c)|E_{OBG_1}|} + \frac{c|\hat{V}| + (1-c)|\hat{E}|}{c|V_{OBG_2}| + (1-c)|E_{OBG_2}|} \right) \quad 0 \leq c \leq 1$$

8.2. For internal structure similarity, we calculate similarity between matching node and different in edge attribute.

$$s_{int} = \beta \left(\frac{1}{|\hat{V}|} \left(\sum_{u \in \hat{V}} SIM(f_1^{-1}(u), f_2^{-1}(u)) \right) \right) + (1-\beta) \left(1 - \frac{\sum_{e \in \hat{E}} |\varepsilon_{OBG_1}(f_1^{-1}(e)) - \varepsilon_{OBG_2}(f_2^{-1}(e))|}{\sum_{e \in \hat{E}} \max\{\varepsilon_{OBG_1}(f_1^{-1}(e)), \varepsilon_{OBG_2}(f_2^{-1}(e))\}} \right)$$

f_1, f_2 is the subgraph isomorphism between \hat{G} and OBG_1 , and \hat{G} and OBG_2 respectively.

$0 \leq \beta \leq 1$ is a weight defined by user for arbitrary weight edge or node similarity; default value of β is 0.5

9. Finally, we assess the similarity between two objects by

$$OBSim(OBG_1, OBG_2) = \alpha s_{ext} + (1-\alpha) s_{int} \quad ; 0 \leq \alpha \leq 1$$

where α is a weight defined by user for arbitrary weight internal or external structure similarity; default value of α is 0.5

Complexity

The complexity of $SIM(TDG_1, TDG_2)$ is determined from the following steps. Given $|TDG_1| = u, |TDG_2| = v$. All $TDGs$ have four levels and the first level have one node and always matched. We assume that nodes in each level is equal, so we create bipartite graph $H = (TDG_1, TDG_2, E)$ at each level where number of nodes is $\frac{(u+v)}{3}$ and number of edges is $|E| = w$. First we compose a bipartite graph at each level and finding its weight from its attribute and degree to its children; $\Theta\left(\frac{uv}{9}\right)$. Next, we find

maximum cardinality minimum weight bipartite matching; the complexity is due to the implemented algorithm. $O((u+v)^2 w)$ by using IP, if we implement Edmonds' algorithm, $O((u+v)^2 \sqrt{(u+v) \log \log (u+v)})$, using the scaling algorithms of Gabow, Gomans and Williamson. Once we have this matching matrix, we calculate their similarity between two graphs. Complexity is $\Theta\left(\frac{uv}{9}\right)$ for finding p_{node} , p_{edge} , p_{degree} , and p_{attr} , $\Theta(1)$ for sim_{top} , sim_{mat} and $SIM(TDG_1, TDG_2)$. Consequently, in this method the complexity is bounded by maximum cardinality minimum weight bipartite matching procedure.

For $OBSim(OBG_1, OBG_2)$, object similarity, the complexity corresponds to the following steps; Given $|OBG_1| = n, |OBG_2| = m$. First, we create matrix S of size $|V_{OBG_1}| \times |V_{OBG_2}|$ where each elements defined 3D node similarity, $SIM(TDG_1, TDG_2)$, thus the complexity is $\Theta(nm)$. Next, the complexity of finding maximum common subgraph is, $O\left(n! \left(\frac{1}{(n-m)!} + \frac{1}{(n-m+1)!} + \frac{1}{(n-m+2)!} + \dots + \frac{1}{(n-1)!} \right)\right)$ which can be approximated by $O(e \cdot n!)$, implemented by using stack and state collecting matching matrix. For calculating the similarity between two $OBGs$ by concerning external and internal structure similarity, s_{ext} and s_{int} , the complexity is $\Theta(nm)$ and $\Theta(1)$ for assess the similarity between two objects by $OBSim(OBG_1, OBG_2)$. Obviously, the most computing time is dedicated to find maximum common subgraph. Unfortunately, at this present, there is no polynomial time algorithm exists. Comparing with the complexity if we implement by using maximum clique detection, the complexity is $O\left(\frac{(n+1)!}{(n-m+1)!}\right) \approx O(n \cdot n!)$.

BIOGRAPHY

Nat Charussuriyong was born in Bangkok, Thailand, on March 8, 1983. He received the bachelor of engineering (with first class honor) in computer engineering from Chulalongkorn University, Bangkok, Thailand, in 2003. He also a cofounder of First Vision Advantage Co.,Ltd. His research interests include computer graphics, shape representation and topology.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย