MODELING 3D OBJECTS WITH FREE-FORM SURFACES USING 2D SKETCHES

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Emre Akatürk

September, 2011

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Tolga Çapın(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Uğur Güdükbay

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Ahmet Oğuz Akyüz

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

MODELING 3D OBJECTS WITH FREE-FORM SURFACES USING 2D SKETCHES

Emre Akatürk
M.S. in Computer Engineering
Supervisor: Asst. Prof. Dr. Tolga Çapın
September, 2011

Using sketches for 3D modelling is a popular research area, which is expected since using 2D sketches feels natural to most of the artists. Many techniques have been proposed to enable an intuitive and competent tool for 3D object creation. In the light of the previous research in this area, we designed a system that enables creation of 3D free-form objects with details. Our system aims to enable users to easily create simple free-form objects using strokes and perturb their surfaces using sketches that provide contours of details and shading information. We provide the user with the ability to create a 3D simple object just by drawing its silhouette. We take this stroke input and create a simple 3D object. Then we allow the user to shade the parts of the 2D silhouette drawn before. We take the shading information and use shape from shading techniques to create a height map and apply the height map on the surface of the object to construct a perturbed surface for the previously created mesh. With our system, it is possible to create and modify 3D meshes easily and intuitively.

Keywords: 3D modeling, sketching, shape from shading.

ÖZET

2 BOYUTLU ESKİZLERDEN 3 BOYUTLU DÜZENSİZ YÜZEYLİ NESNELER MODELLEME

Emre Akatürk
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Y. Doç. Dr. Tolga Çapın
Eylül, 2011

İki boyutlu eskizlerin kullanımının, pek çok sanatçının kendisini doğal hissetmesini sağladığını düşünürsek, üç boyutlu modellemelerde eskiz kullanımının neden bu kadar rağbet gören bir araştırma alanı olduğunu anlayabiliriz. Üç boyutlu nesne yaratabilmek için; kullanımı kolay, aynı zamanda yetkin bir araca olanak sağlaması amacıyla pek çok teknik sunulmuştur. Bu tez de, bu alanda daha önce yapılmış olan araştırmaların ışığında, detaylı üç boyutlu nesne tasarımına olanak sağlayacak bir sistem geliştirmeyi amaçlamaktadır. Tezde kullanılan sistemin amacı, kullanıcıların bazı çizgilerle pürüzsüz basit nesneler yaratabilmesini ve gölgelendirme bilgisi taşıyan eskizler aracılığıyla, yaratılan nesnenin yüzeyini pürüzlü hale dönüştürebilmesini sağlamaktır. Böylelikle, sadece basit bir silüet çizimiyle kullanıcılara, üç boyutlu pürüzsüz basit bir nesne yaratma olanağı yaratılmıştır. Sistemin işleyişi şu şekilde özetlenebilir: Sistem, çizgi girdisini alır ve basit üç boyutlu bir nesne yaratır. Ardından, kullanıcının daha önceden çizmiş olduğu iki boyutlu silüetin istediği kısımlarını gölgelendirmesine olanak sağlanır. Burada, yükseklik haritasının oluşumunda kullanılan gölgelendirme bilgisini ve gölgelendirmenin şeklini alır ve daha önceden oluşturulmuş örgü üzerinde pürüzlü bir yüzey oluşturmak için pürüzsüz nesnenin yüzeyine gölgelendirme haritasını uygular. Tezde kullanlan sistem sayesinde, kolayca üç boyutlu örgüler yaratılabilir ve üzerinde değişikler yapılabilir.

Anahtar sözcükler: 3 boyutlu modelleme, eskiz, tonlama kullanarak şekil çıkarma.

Acknowledgement

I would like to express my gratitude to Dr. Tolga Çapın, from whom I have learned a lot, due to his supervision, suggestions, and support during this research.

I am also indebted to Dr. Uğur Güdükbay and Dr. Ahmet Oğuz Akyüz for showing keen interest to the subject matter and accepting to read and review this thesis.

I would like to thank to my colleagues from office, for their comments and reviews.

I am grateful to Denizhan Güçer, for his continuous support and patience.

I want to express my gratitude to my grandfather, Erol Karapınar who has encouraged and supported me during my education.

I am also grateful to my father for his guidance and I would like to thank him for the faith he put in me.

My mother has provided assistance in numerous ways during my work in this thesis. I am grateful for her help and support in my worst days.

Finally, I would like to express my gratitude to Elif Erdoğan for her endless support, her help and understanding during my work on this thesis.

Contents

List of Figures

Chapter 1

Introduction

3D object modeling is a major research area in the computer graphics. As the powerful commercial modeling tools such as Maya [52] and 3D Studio Max [53] has been available to everyone and the power of the computer aided design tools have been discovered, many design related professionals such as engineers and architects utilize 3D modeling tools. The 3D modeling is also very popular (as expected) in its inevitable usage in gaming and 3D animation film industry. Most of the commercial and powerful 3D design tools today employ window, icon, menu, pointer (WIMP) interface [13]. This kind of interaction proved to be useful, and 3D modeling tools that employ such an interaction method are able to construct very detailed and realistic 3D models.

The 3D design tools that utilize the WIMP interface are very powerful but creation of 3D meshes with these systems require tedious work and much experience. In order to remedy this, researchers have proposed a different interface, one that comes most natural to artists and designers, a sketch based interface. The idea that leads researchers to employ such an interface is that the traditional way to design and express ideas is done through sketching and one might extract ideas and understand the mental process under the sketch. Extending this idea to 3D modeling interfaces, researchers aimed to reconstruct 3D models by mimicking the cognitive process of human visual recognition system and by trying to understand the clues about the designer's mental process that the designer provides to

the system when designing an object by sketching it.

The ultimate goal of sketch based modeling interfaces is briefly to provide an easy and intuitive modeling method that uses sketch input and to provide an interface that is as powerful as the WIMP interfaces. This goal is far from becoming true due to the complex nature of the 3D shape recognition process, however. This complexity arises mostly because of the lack of the depth information in sketches and in most of the interfaces that allow the user to provide sketch input to the computer.

Research in sketch based modeling interfaces today consists of systems that are scattered among different approaches to the sketch interpretation problem. Many different approaches have been proposed and none of these approaches proved nor aimed to be a final and complete method for sketch based modeling systems, which is only normal since the field is relatively new. The results of research are promising, however. Many incomplete but useful methods that provide satisfactory results have been proposed. The current state of the research seems to become a collection of methods that employ different approaches that are able to address specific problems that sketch based modeling interfaces try to solve.

Observing the current state of the research, we try to address such a specific problem that has not been completely solved. The modeling of 3D objects is a problem of not only describing the overall structure of the object, but also a problem of depicting the detailed surface structure of the object. This is required in some systems more than others, especially in applications where the details are important such as modeling of 3D characters and objects in gaming and 3D animation film industry. The details of an object are especially important when modeling objects with perturbed free-form surfaces such as an almond or fish. These kind of objects have patterned or randomly placed perturbations on their surfaces.

Some of the current sketch based systems provide functionalities that allow such perturbations to be applied to the surface of an object such as Mudbox and ZBrush [54] [51]. These systems employ a depth painting method, which

allows the user to paint depth values on vertices using a brush. Some systems proposed to use shaded images [9] to construct shape of an object or to employ 2D painting interfaces to deform or manually correct mistakes on a pre existing 3D mesh [8]. The approach that is used in such systems inspired us to use shading data provided by the user to serve our purpose. Since these image based techniques are inadequate for fully 3D object creation purposes, we propose using such a technique as a modifier to a 3D object that is created by a purely sketch based 3D object creation method.

The main idea behind our work is that the perturbations and free-form structure of a surface can be easily described by sketch input, with a traditional method named shading which is used for describing the depth values of a surface. We have observed that an artist provides shading input to depict the depth values of a surface, and this input can prove to be most valuable when reconstructing the surface of an object. We combine this idea with a 3D free-form object creation approach that only requires 2D silhouette information provided by the user. Our 3D object creation method uses sweep lines to construct surfaces which are similar to the rotational sweep surfaces used in Cherlin et al.'s work [3] which is explained in more detail in Chapter 2. The 3D object creation method employed in our system, when combined with a depth modification system, provides an intuitive and simple interface based solely on 2D sketch input.

The contributions of the thesis are summarized as follows:

1. We present a novel approach for creating fully 3D objects with free-form surfaces by only utilizing sketch input. We combine a 3D mesh creation system that is similar to the the rotational sweep surface technique proposed by Cherlin et al. [3] and combine it with a image based depth modification technique. This combination provides an object with desired geometric perturbations applied to its surface. The overall shape of the object is determined by the object creation technique and the surface geometry is determined by the height field that our image based system constructs according to the shading input.

2. We test regular triangular mesh structure for our image based technique. We use a regular triangular mesh to represent the geometry of the objects. The triangles of our mesh have a predefined edge length and are formed to fit inside vertices that are distributed uniformly through the area inside the borders of the mesh. In order to apply the appropriate perturbation value to a vertex in the mesh, we simply find the appropriate height value from the height map. When we calculate the perturbation value, we also add height values of adjacent elements in the height map which is constructed using user's shading input. Then we apply the perturbation value to that vertex. When all vertices are perturbed in this manner, we obtain the desired surface geometry on the 3D final mesh.

We have observed that the usage of our system is easy and intuitive and users are able to model simple meshes with randomly placed or patterned perturbations with it. The limited scope of objects that can be modeled with our 3D object creation algorithm is a limitation, however. Also, the image based method we employ creates a fidelity problem when a curve is to be drawn on a 3D object such as the curves of a basketball. Thus our system is suitable for fast creation of 3D objects with detailed free-form surfaces.

The organization of the thesis is as follows: In Chapter 2 we briefly explain several different methods that discuss solutions to mesh creation problem and provide the background of the subject. Then in Chapter 3, we divide our system into sub-systems and explain each one in detail. In Chapter 4, we discuss different aspects and show results to the reader and finally in Chapter 5 we present conclusion for our work.

Chapter 2

Background and Related Work

There has been a lot of research in sketch based modeling systems. In this chapter, we explore different attempts that have been made in this research area. Most of the sketch based modeling systems include different sketch input acquisition methods [13]. Therefore, sketch input acquisition techniques is explained in Section 2.1. In the rest of the chapter, we explain different sketch based modeling methods. We used in this chapter a division that is proposed by Cook et al. [14]. The research and different sketch-based modeling techniques are divided into 7 different parts and explained in the rest of this chapter in the following order: Gesture Created Primitives, Reconstruction, Height Fields and Shape-from-shading, Deformation and Sculpture, Blobby Inflation, Contour Curves and Drawing Surfaces and Stroke Based Constructions.

2.1 Sketch Input

One of the several concerns of sketch based modeling interfaces is sketch acquisition. Since user interaction is a fundamental issue for sketch based systems and many contemporary systems aim for easier user interaction, there is a lot of research in this area. There are several problems faced when user interacts with the system, and so there have been several different approaches to each of these

problems.

A major aspect about user input is the sketch acquisition device that users utilize to interact with the system. Several hardware choices are available, starting with the mouse and extending all the way to more recently proposed devices such as haptic devices [41]. Among all the computer interaction devices, especially those that are used in sketch based systems, one of the most ubiquitous devices is the mouse. Although the mouse is a familiar device for many users, it is hard for most users to successfully draw accurate shapes with the mouse. Tablet devices provide easier interaction to which most artists are used to, due to its similarity with the traditional pen and paper sketching. Some tablet devices employ the pressure data and orientation of the pen to amplify the expressiveness of the interaction device. There have also been different solutions to this problem, such as virtual reality devices or haptic devices [42] [41].

Another concern is sampling of the input data. In most cases when the user input is received, it is sampled. Since the real input provided by the user is continuous and sketch based systems receive the input in a discrete manner, the sampling process can be problematic. One problem this restriction creates is that the spatial distance between the consecutive sampled input points provided by the user varies. Users tend to draw some part of the sketch faster than others. This causes the faster drawn parts to have adjacent input points with larger spacing between them.

The sampled input is sometimes stored in structures called strokes. A stroke is defined as a sequence of sampled point input that the user provides to the system. A stroke starts with the user putting down his pen and ends when the pen is up. All points sampled between these two actions are used to construct the stroke.

Some of the contemporary systems use image based inputs. This approach is mostly used in systems that utilize image based approaches such as shape from shading (SFS) [8] [9].

Using the sampled input points with varying spacing may produce undesired

results. Contemporary sketch based systems resample the input data for the sake of constructing the proper input desired by the user. There are several approaches to this problem.

One of the approaches proposed to solve the problem mentioned above is the minimax method [11]. This method minimizes the maximum distance between the approximating line and the points that represent the polygonal curve. Saykol et al. use another approach to approximate polygons [12]. In their work, they take polygon points as input and find importance level of these points, importance being determined by vertex velocity and acceleration, where vertex velocity is the rate of change of distance per angle and vertex acceleration is the rate of change of velocity per angle. Then using these vertex velocity and acceleration values, vertices with the highest importance level are found and used in the resulting polygon.

There are other approaches to this problem which produces rough approximations. For example, Igarashi et al. proposed a simpler solution in their system Teddy [1]. In Teddy, the first and the last input point provided by the user is connected, and the result is checked. If the resulting shape is a 2D closed polygon, the system resamples the input points so that the adjacent points are equidistant. This approach aims the resampled points to form vertices of a 2D polygon that has edges with equal length, but it does not guarantee the result to be precise approximation.

Fitting the input provided by the user to other forms such as curves or lines has been employed by many researchers in their work. Some systems benefit from line or curve fitting techniques since lines or curves are simpler to analyse in some cases. There are also systems that use curves or lines to construct the object to be modelled.

There are several systems that use curve fitting to input strokes. For example in their work, Kara et al. use curve fitting to construct B-splines using a least squares curve fitting algorithm [7]. These curves are used in the model creation process where the user draws a wireframe model of the intended object. There are also other approaches to this problem. For example, Eggli et al. used least

squares curve fitting algorithms to construct B-splines [4] and Cherlin et al. used a reverse Chaikin subdivision technique to construct B-spline curves [3].

Some methods use both linear and curve representations. Such a work is proposed by Sezgin et al [10]. In their work, they propose a system that distinguishes line segments from curves from a drawing. The system finds straight line segments in the stroke and then it discretely approximates Bezier curves from the curvy portions of the drawing. The resulting image consists of curves and line segments.

Line and curve fitting is mostly used in applications where details and precision is most important such as engineering design systems. On the other hand, free form sketches have gained increased attention in the past decades. These differ from engineering design applications in that they provide more freedom to the user but lack the precision the engineering design applications offer. It is generally observed that people tend to draw several strokes before they get the final shape of the sketched object [14]. These lines depict the shape of the drawn object together. Pointing this issue, a sketching technique allows the user to draw many lines to describe the overall shape of an object and then use all of them to get a final result. This method of interaction is generally known as oversketching [14].

A number of systems use the idea of oversketching to allow the user to edit a line created before. For example, Fleisch et al. have proposed a system where the user draws an editing curve and the previously drawn curve is modified according to this curve [6]. Additionally they allow the user to provide several parameters. One parameter defines the effect of the overdrawn curve. The user is able to select how much the replacement curve will affect the original one. This parameter allows the user to completely change the original curve into the editing curve when it is 1, not to change it at all when it is 0. Thus, any value between 0 and 1 produces a curve between the original and the editing curve. In order to avoid breaks where the editing starts and ends on the original curve, the system smooths the curve around the starting and ending areas in order to make the transition. This transition interval size is also changeable by the user, where the

number of points is the parameter provided by the user.

Several approaches use techniques to find one curve that depicts the shape of the oversketched curve drawn by the user. One such system is proposed by Pusch et al. where the overall sketch is subdivided into boxes [5]. The subdivision ends when all boxes contain strokes that has roughly the same direction. These boxes are ordered so that internal strokes compose a complete curve. Then the points are fitted to a B-spline curve with the utilization of the ordering found using a reverse Chaikin subdivision technique.

Another method that uses oversketching input to form curves is proposed by Henzen et al. [2]. In their work, they proposed a system where the user is able to overdraw a line and the lines begin to fade in time. The final shape of the curve is formed by taking into account the most intensely coloured part of the overdrawn curves. This system allows the user to draw many lines and the final curve is constructed according to the part which is more saturated.

2.2 Sketch based Modeling Methods

After the the sketch input acquisition and re sampling is done, the next step is to interpret the sketch input and create 3D models.

2.2.1 Primitives Created using Gestures

An early approach to sketch based modeling was the creation of simple objects such as cubes or cylinders using gestures. The motivation behind some of these systems was to create complex objects by combining or applying other boolean operations with several simple objects. Several methods were proposed to create an unambiguous interface that allows the user to easily create simple objects with gestures.

One early system named SKETCH that uses gestures to create simple objects

was proposed by Zeleznik et al. [15] In SKETCH, users are able to draw several lines that combine into gestures. These gestures are recognized by the system and the recognized 3D object is created. For example, drawing an edge and two lines that are perpendicular to the edge and that end at any point on the drawn edge results in a 3D cuboid object and lengths of its sides are determined by the lengths of the drawn lines. There are also other objects that are not defined by their edges. An object of revolution, for example, can be created by drawing its profile and axis. There are several objects that can be created by SKETCH system, which are: cones, cylinders, spheres, objects of revolution, prisms, extrusions, ducts and superquadrics.

A similar system that uses gestural recognition is CIGRO which is proposed by Contero et al. [43]. CIGRO is capable of creating 3D objects using a small instruction set which includes gestures like adding/removing an edge, adding auxiliary edges and gestural commands such as move copy and delete. In CIGRO, user first draws a number of auxiliary lines that depicts the overall shape of the object. Then the user draws the real edges of the 3D object within the auxiliary lines. The real lines are differentiated with the auxiliary lines with the pressure information retrieved by the tablet device. The recognizer used in CIGRO is able to recognize elemental geometric forms such as triangles, rectangles, circles and ellipses.

Although SKETCH and SKETCH like systems shows that objects can be created using gestures with ease, the scope of such systems is very limited. Although more gestures can be added to overcome this, recognition of gestures gets harder as gesture library gets larger. Several researchers have designed suggestive systems. The key idea of these systems is to inform the user of the possible results and allow him to select one, eliminating the ambiguity of the interpretation of the drawn shape. Such systems called GIDES and GIDES++, have been designed by Pereira et al. [16] [17]. In these systems, the user draws gestures and when the reconstruction process is complete, the system interactively suggests several options in a small window. When the suggestions (called expectation lists by author) are viewed, the user may or may not choose one. In the latter option, the user continues drawing in order to create a different and maybe a more complex

object. In these systems, the users are also able to apply editing operations using gestures, which also has suggestive feedback to the user.

Although these systems prove successful creation of some objects, their scope is limited even with the utilization of expectation lists. Moreover, a gesture based system restricts the user by forcing their input to predefined shapes and does not provide the freedom and expressiveness of traditional sketching.

2.2.2 Reconstruction

There are a number of studies that aim to interpret the users' intention without using gestures. These systems evolved in order to remedy the limited scope and indirect interaction style of gesture based systems and to build systems that do not interrupt the sketching experience of the user. These systems also have to deal with the ambiguity of the 2D sketches.

Early researchers have tried to solve the ambiguity problem by identifying lines. Identifying a line here means differentiating lines so that when a 3D object is formed, the effect of the line is determined. This may be a tedious or even impossible job if the scope of objects that the system tries to identify is large. There are some methods that are used to differentiate (or label) lines this way.A number of solutions use a method called Huffman-Clowes line labelling [18] [19]. The key point in Huffman-Clowes line labelling system is that the scope of the system is limited to trihedral planar objects. The system labels every line in the 2D object. There are three different labels. Each line, being an edge of the 3D object to be created are either a convex, concave or occlusion edge. These labels mean that when the 3D object is constructed, from the viewers perspective, convex edges will be closer to the screen than its adjacent edges and concave edges will be farther away. Occlusion edges are edges that form the silhouette of the 3D object from the user's perspective.

One system that uses Huffman-Clowes line labelling is proposed by Grimstead and Martin. [20] In their system, an incremental line labeller finds possible line

labellings that can be produced as the user draws. If the result is not what it is intended to be, then results derived from alternative labellings are shown. Then the system produces the 3D object with the selected line labelling by producing both visible and hidden faces from the labelled edges. Although this approach is capable of producing 3D objects and deal with the ambiguity, the system is very complex and its scope is limited to trihedral objects. The intervention of the user is also still required to resolve the ambiguity of the sketch.

In their system, Stilson et al. [21] have proposed a system that reconstructs a 3D object from line drawings that are drawn onto a 3D model. The motivation behind this system is to provide a system for architectural design by combining 2D sketching and 3D environments. The perspective information and pre-existing geometry of the 3D environment is used to interpret the 2D line drawings.

Another approach for reconstruction systems is proposed by Lipson et al. [22]. The main motivation behind their work is to emulate the human interpretation of 2D sketches. They claim that humans' interpretation of 2D objects is based on their visual experience. Building a system based on this claim, they have tried to emulate the 2D-3D correlation by employing the correspondence information between 3D objects and their 2D projections. Using this information, they produce a probability function for the candidate 3D objects that is later used to find a resulting 3D object. As the system produces correct 3D objects, the results are mostly rough objects as Lipson et al. indicates, which makes the system unusable for applications that require precision.

A different reconstruction approach is proposed by Piquer et al. which reconstructs 3D polyhedral objects using their symmetric properties [45]. To create an object, the user draws a 3D polyhedral object from straight lines. Then the system fins a symmetry plane from the sketch and creates a new coordinate system that the authors call the symmetry system according to the symmetry plane found previously. Using the symmetry axis, the symmetry conditions are found and the 3D object is formed according to these conditions.

Since sketching systems based on reconstruction are suitable for engineering design and some objects that are constructed for engineering purposes may need

analysis of their physical properties, a system that incorporates these two features would be practical. With this motivation, Masry et al. developed a system that incorporates reconstruction based systems with analysers that provides analysis of the structural properties of the created objects [44]. Their approach has two parts: The first part is the reconstruction. The system allows reconstruction of an object that is composed of straight lines and planar curves. The second part is the analysis. In order to analyse the physical properties of the object created, the authors use finite element analysis technique.

One recent work, that reconstructs shapes from 2D silhouettes is proposed by Rivers et al [48]. In their work, the authors proposed a system where the user draws 2D silhouettes of a 3D object from top, side and front views. The system automatically constructs 3D shape of an object whenever the user draws different silhouette sketches of an object interactively. For example if the user draws triangles for the front and side views and a square for the top view, the system generates a 3D pyramid object. The creation of the models is achieved by boolean operations that are applied on the silhouette cylinders of the 2D sketches where the silhouette cylinder is defined by the infinite extension of a silhouette in the view direction. The resulting object is defined by the intersection of these silhouette cylinders.

2.2.3 Height Fields and SFS

One key feature of traditional pen and paper sketching is shading. Artists use the shading information to describe the shape of the object they are drawing. A number of researchers have proposed methods that utilize the shading information provided by the user to extract depth information.

Rushmeier et al. have proposed a system where users can edit the 3D geometry of an object by modifying images rendered from the 3D object with a 2D paint program or by using images from a photo or another model [8]. Using their system, the user is able to edit the surface of a 3D object by manipulating an image of the object with 2D paint operations such as cut, paste, paint, sharpen

and blur. Then the user is asked to edit the diffuse reflectance map in order to finalize the editing operation. The system also provides a solution for making major changes to a 3D model. In this editing scheme, the user provides an image of another 3D model or a real photograph and fits the image to the area to be edited. The grayscale changes are applied to the mesh after the grayscale image is inputted to a shape-from-shading algorithm. Then the resulting depth map is applied to the 3D object.

Another approach that uses shading to extract depth information is proposed by Kerautret et al. [9]. The authors mention that the current shape from shading approaches are not robust and are not able to provide a unique solution. In order to remedy this, they use several shaded images of the same object each provided by the user to construct a unique interpretation of the input. The images provided by the user contain the contour of the drawn object as well as the shading of the object under different lighting conditions. Although this system provides an intuitive method for creation of objects, the resulting objects are $2\frac{1}{2}$D, which means their depth extends in one direction.

The methods based on solely height fields and shape-from-shading provide intuitive model creation and modification but creation of fully 3D, complex objects from scratch is still not possible with these methods. A commercial product named ZBrush allows creation of fully 3D detailed objects, by using a depth painting interface incorporated with modeling methods [51]. Using ZBrush, the users can interactively alter the surface geometry of an object by painting the surface of an object. The results of the ZBrush shows that incorporating the depth information with another technique provides better results.

2.2.4 Deformation and Sculpture

A number of studies utilize a more general form of editing in the sense that the whole object or a part of it is deformed in a more direct manner. In these systems, rather than using images to edit objects, the user deforms the object directly.

Most of these deformation techniques use interfaces that allow the user to sculpt or carve minor details onto the surface of an object. One such system proposed by Frisken et al. [34] utilizes Adaptively Sampled Distance Fields (ADFs) to allow local deformations. Their implementation of ADF provides a representation for volumetric data and allows carving fine details. A distance field is basically a scalar field that specifies the minimum distance to a shape. The motivation behind the employment of such a representation is that the storage method is adaptive and sampling of the distance field is less where the local detail is low. The system allows fine carving of the object with efficient sampling rates since the method is adaptive. The user carves an object by moving the carving tool on the object's surface.

Other surface representations are used for the deformation of the objects. For example, Bærentzen et al. proposed using the level set method for such purposes [35]. Using the level set method, the authors aimed to provide a generic technique for volumetric deformation. The level set method is briefly used to compute the evolution of surfaces that may expand or contract [35]. In their system, authors provide sculpting operations such as addition or removing of volumes and smoothing.

There are other attempts that have used global deformations on the objects. Wyvill et al. have proposed a system where models are defined by skeletal implicit surfaces [33]. The authors propose a structure called Blob tree, which is a hierarchical structure that consists of models on which warping, blending and boolean operations can be applied. The users are able to apply boolean operations globally to implicit surface models and the hierarchical structure allows arbitrary compositions of these deformed objects.

Some proposed different techniques for modeling systems using deformation. One such attempt was made by Lawrence et al. where the user is able to deform and model an object with painting on its surface [36]. The work aims to provide a direct modeling and deformation interface. To deform a surface, the user paints over the surface and the system interactively provides a volumetric addition on the surface that is painted. The system also allows the user to select different

paints, changing the effect of the deformation, where the surface is propagated towards the surface normals in one selection and in other the surface is propagated in a constant direction.

One other approach was proposed by Singh et al. which allows the user to employ a structure called wires in order to deform an object [37]. Their deformation technique is similar to armatures used by sculptures, where wires define the shape of the object. Their system briefly defines a number of wires on an object and allows the user to manipulate the wires in order to deform the object.

These deformation and sculpture techniques are mostly intuitive mostly because of its resemblance to sculpture techniques used by artists, but most of these systems are not practical when it comes to create an object from scratch. These techniques are better used as a supporting tool for model creation techniques. One such approach is proposed by Draper et al. in their system Freddy [46]. The authors proposed utilization of gesture recognized free-form-deformations on objects that are created using the 3D object creation employed in Teddy [1]. To deform an object, the user provides the system with gestural inputs. Then the gesture is recognized and the FFD lattice is displaced according to the input. The user can bend, twist, stretch or squash an object with different gestures. Kho et al. also proposed a system where the user deforms the object using a few curves [47]. In their system, the user provides two curves in order to deform a 3D object. The user first draws a reference curve as it was the skeletal description of the part of the 3D object to be deformed. The curve is projected onto the 3D space and the surface to be deformed according to the curve is calculated. Then the user draws a second curve and the surfaces of the object that is previously associated with the reference curve are displaced according to the second curve. The final deformed object provides a result such that the two curves are the bone structure of the object and the object is bent according to it.

2.2.5 Blobby Inflation

While there is a lot of research on sketch based modeling systems which try to increase the user's control over the resulting mesh, there are also some researchers that have proposed solutions for creation of smaller set of objects by decreasing the user's control over the resulting object.

Igarashi et al. has proposed such a system named Teddy, where the user's sketch input is taken and a final 3D object is created [1]. The 3D object creation method used by Teddy is called inflation, and as its name indicates the 2D silhouette provided to the system produces a resulting object so that it seems like it has been inflated. In to create an object in Teddy, the user provides the system with 2D free form strokes, which the system interprets as the silhouette for a object and then the system constructs a 3D polygonal object based on the given data. From the 2D strokes, the system creates a closed planar polygon by connecting the start and end points of the stroke. Then the system creates a spine for the 2D polygon. Here, spine is defined as the structure that describes the skeleton of the mesh and every point in the spine is equidistant from the edges. The spine is created by finding the chordal axis of the 2D polygon [24]. In order to find the chordal axis, the system triangulates the 2D closed polygon using constrained delaunay triangulation. After the triangulation, midpoints of internal edges of the triangulated 2D polygon are connected to create the cordial axis. Then the cordial axis is pruned to construct the final shape of the spine. After the spine is created, the triangles of the 2D polygon are divided by the spine and resulting polygons are triangulated. Then the vertices of the spine are elevated proportional to their distance between the vertex and edges. After the spine is elevated, a 3D mesh that covers the elevated spine edges and the external edges is formed. The final shape of the 3D mesh is constructed so that they form a quarter ovals between spine vertices and external edge vertices.

The inflation method that is employed by Teddy has its benefits. Although Teddy provides little control to the user on the final shape and the system is only able to produce rotund objects, Teddy provides a well defined, intuitive and easy to use technique to create 3D meshes. Although this system has benefits,

its aim is to construct approximate objects without precise details and is unable to create complex meshes.

The idea of using 2D sketch silhouettes to from 3D rotund objects has been used by other researchers who have been inspired by the approach used in Teddy. Karpenko et al., for example, have proposed using variational implicit surfaces in their work [25]. The variational implicit surfaces are proposed by Turk and O'Brien [26]. In their system, they allow the user to draw the silhouette of an object and the system inflates the outline of the object drawn and a 3D object is created. They also support drawing of additional shapes which overlap with the previously inflated objects. The inflation algorithm produces 3D objects so that the shape varies according to the width of the 2D shape drawn by the user, meaning that thin shapes produce cylindrical objects whereas circular shapes produces fatter 3D shapes. They also support creation of hierarchical shapes. The hierarchy is constructed according to the overlapping regions of the 3D object that each drawn shape produces and the previously created objects.

Tai et al. have noted that the technique used by Karpenko et al. requires high computational cost and proposed a technique that constructs 3D rotund objects from 2D shapes using convolution surfaces [27]. Their method extracts the skeleton of the 2D shape and a rotund generic convolution surface is created for each skeletal line segment. Their skeleton finding algorithm aims to find an approximate medial axis which is defined as the locus of the center of maximal circles inside the 2D shape. The 3D mesh is obtained by convolving the skeleton.

Schmidt et al. have proposed a free form modeling system that provides many features in order to create a fully capable free form modeling system [28]. There are several modeling operations that the system provides. The system provides blobby inflation from 2D shapes, sweep surfaces that are created with linear sweeps and surfaces of revolution, cutting, blending operations. The system also provides surface drawings to be applied to a created 3D object, where the strokes are used to modify the surface of the 3D object.

The skeletal structure that is employed by some of the systems that use blobby inflation can be used for other purposes as well. In their recent work, Yang et al.

proposed a system called Life-Sketch that constructs 3D models from 2D sketches and extracts its skeleton to be used for animation purposes [50]. The authors used Teddy's method of inflation to create 3D objects from 2D silhouettes. Then a skeleton extraction algorithm finds the skeleton of the object using the chordal axis which is created as a middle step of Teddy's inflation algorithm. When the skeleton is found, the user can animate the object by rotating the object's bones around its skeletal joints using a keyboard.

Most of the systems that utilize blobby inflation techniques are powerful tools in the sense that they allow creating simple objects with ease. Blobby inflation techniques provide an intuitive way of 3D simple object creation. Furthermore, inflation methods achieved what most sketch based systems aimed, creating 3D object from 2D sketches directly. The shortcomings of these techniques are the limited scope of objects that these systems are capable of creating and the lack of fine details.

2.2.6 Contour Curves and Drawing Surfaces

Sketching activity mostly involves drawing of curves to depict the overall shape. Since the contour of the object depicts much about the object to be designed, some researchers proposed modeling methods that is based on drawing curves. Although, since drawing 2D curves on a 2D plane is not enough to describe 3D surfaces, some sketch based systems evolved to use 2D curves to construct 3D curves or to create 3D curves directly.

An early work that is proposed by Cohen et al. proposes a technique that allows the user to draw 3D curves using 2D sketches [38]. After drawing 2D curves, the user provides a second curve that the authors call the shadow curve. As its name indicates, the shadow curve is the projection of the initial curve on a plane. The depth information and the 3D structure of the final curve is extracted from the shadow curve.

Some researchers used this modeling with 3D curves idea to enhance specific

methods of 3D object modeling. Grossman et al., for example, proposed a method that is inspired by a design method called "tape drawing" [39]. Tape drawing technique is used in automobile industry, where the artist draws the concept sketches of a car to large, 1 to 1 scale black photographic tapes. The system also uses large displays to provide a similar environment. The user interacts with the system by drawing 2D profile curves and these curves are then used in creation of 3D models. The 2D profile curves, drawn by user are constructed on 2D planes. The view of the system is a cuboid that includes the 2D drawing planes. These planes are shown to the user within the cuboid that contains these 2D planes as parallel surfaces. This display method is employed in order to help the user understand the relation between the drawn 2D curves and the underlying 3D object.

Some approaches have used similar techniques that allow creation of 3D objects with 2D curves with a suggestive interface. The work proposed by Tsang et al., for example, uses image guided sketching on 2D planes on a 3D environment [40]. In the system, the user draws curves on 2D planes that has 2D images of an object that is similar to the one which is being created. The user draws curves on these 2D planes, and the system guides the user by attracting the drawn curves to the curves in the image. Suggestions of pre-existing and user created shapes is also available when the system matches the user input with one that is held at a database of shapes. The user is able to draw on orthographic 2D planes from three different viewpoints: top, side and front. All the 2D planes and 2D curves drawn on these planes are shown within a 3D cuboid volume.

2.2.7 Stroke Based Constructions

There have been attempts to construct 3D objects from 2D strokes by fitting surfaces to input curves. For example, Wesche and Seidel have proposed FreeDrawer, a system where the user is required to provide the system with several strokes that describe the object [29]. The user needs to provide the overall description of the object as a network of curves. The system works in a 3D environment in which the user is able to construct 3D shapes by drawing 2D/3D curves and

creating surfaces between closed loops of curves. The system is best used by users with drawing skills, as it is also noted by the authors [29]. Michalik et al. have proposed another system that allows modification of B-spline surfaces by drawing 3D curves on the surface [30]. In their work, to create a new 3D object, the user draws curves which are projected onto planes. Using these curves, B-spline surfaces are created using a constraint based approach. This system is powerful but has high computation cost and the run time increases rapidly with large examples, as its authors note [30].

Although curve sketching is a powerful design tool, the need for drawing skills and the complexity of the object creation make these systems hard to use especially for users without an artistic background. A number of approaches have constrained the scope of the objects by sacrificing the freedom of the user but increasing the intuitiveness and expressiveness of the method. Levet et al. have proposed such an object creation method that uses a similar approach to inflation employed in Teddy [31]. The system requires the silhouette of the object and a profile curve. The silhouette is used to inflate the shape with a method similar to Teddy but the shape of the object depends on a profile curve which is also proposed by the user. The system elevates the vertices of the 2D shape according to the profile curve provided by the user which is in contrast to Teddy where the system inflates the 2D shape according to a circular profile curve.

Another approach, proposed by Cherlin et al. aims users to design 3D objects with a few strokes [3]. Their approach is based on a method used in traditional pen and paper drawing, which the authors call the spiral method, in which the artist draws the silhouette of the shape and then draws spiral curves to describe the shape of the object. In the system, creation of the 3D objects is done in a similar manner: The user first draws the silhouette of the object using two or more strokes, which are called the constructive curves. Then the midpoints of these two constructive curves are used to form another curve, which is used as a center curve to construct the surface of the object. Finally, for all points of the center curve, a circle that has a center at the center curve and is passing through the two constructive curves is created to form the final shape of the object. A surface that is created by this method is called a rotational blending

surface. Using this method, the system offers an easy to use, intuitive method for creating 3D objects which requires users to use only a few strokes to depict the object they want to create. Another creation method used by the system proposed by Cherlin et al. is called the cross sectional blending surfaces. In this method, which is similar to the rotational blending surface method, the user again draws two constructive curves. Then he draws a curve between these two constructive curves. The resulting surface between the two constructive curves is generated by using this curve rather than a circle. This method allows users to create thin, non circular objects with an arbitrary cross section.

A recent work has been proposed by Stiver et al. [49] where the system uses a stroke based construction system that is similar to what is used by Cherlin et al. [3]. The authors proposed a method for cloud modeling, where the user models the general form of clouds by strokes. Then using rotational or cross sectional blending surface techniques, an initial mesh is created. Then the mesh is filled with volumetric particles. These particles provide a noisy look on the boundaries of the cloud and makes it seem like one.

Stroke based systems are powerful and provide easy and intuitive way of 3D object creation methods, where the user is able to express the shape of the object with simple curves. Whereas some of these systems can be used to construct complex objects in detail and provide freedom to the user, some can be used to construct simpler objects by limiting the scope. In both cases, since the strokes are used directly to form objects, these systems are very intuitive.

Most of the approaches explained in this chapter have both strong and weak aspects and by merging two different methods, better solutions can be produced. Our system combines two of the above mentioned approaches, to provide a solution for a very specific problem. Our system is inspired by two different limitations that we observed in the contemporary sketch based modeling solutions. We observed that blobby inflation methods are capable of producing 3D free-form objects from simple 2D silhouettes but they are mostly incapable for addition of surface texture details. We also observed that the shape from shading methods are incapable of creating 3D objects from scratch but can be utilized to form

detailed surfaces. In our work, we combine these two approaches and create 3D objects with detailed free-form surfaces.

Chapter 3

Method Description

3.1 Overall System Description

Our system takes two separate sketch inputs from the user. The first input is the silhouette, which is used in creating of the 3D object mesh; and the second input is the shading, which is used to create the height field. The two results are then combined by applying the height field on the 3D mesh to obtain the final result. The overall framework of the system is shown in Figure 3.1.

The system briefly operates in the following order: First, the user provides the system with a drawing of the 2D silhouette of the 3D object to be created. The silhouette input is resampled in order to create a smaller set of input points that carries roughly the same information. Then the 3D object creation step constructs a 3D object from the silhouette input. Then user provides the system with shading strokes that depicts the surface of the final 3D object. The user inputs the shading data by drawing strokes inside the 2D silhouette that was previously drawn. Therefore, the user sees every line and shading that is drawn as it were in pen and paper sketching. Then a height field is constructed using this shading data and finally the height field is applied to the 3D object to produce the final mesh. The overall procedure can be seen in Figure 3.2.

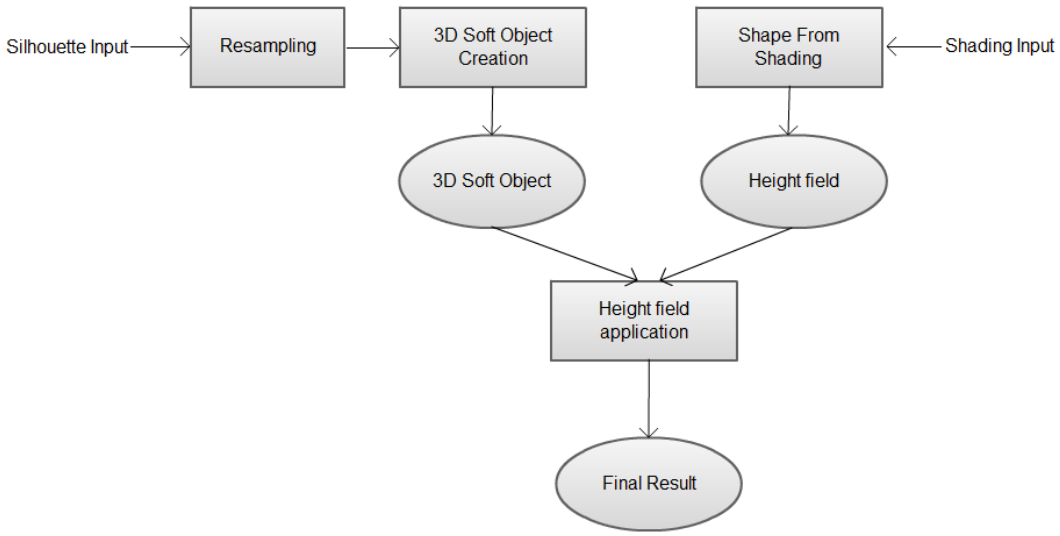


Figure 3.1: The Proposed Framework

3.2 System Description

This section aims to provide further explanation of the different parts of the system. In Section 3.2.1, storage and resampling of the silhouette input is discussed. The creation of the 3D object is explained in Section 3.2.2 in detail. In Section 3.2.3, creation of the height field is discussed and finally in Section 3.2.3, we describe the application of the height field on the 3D object.

3.2.1 Receiving and Resampling the Silhouette Input

Our system is compatible with 2D input devices. Using a 2D input device such as a mouse or a tablet device, a user without drawing skills is able to use our system to its full extent. When the user starts to draw the silhouette of the object, the input is sampled and is stored as dense points. The points are stored in a list structure. The list's order is important; it is ordered according to the time the sampled point is received by the system. After the sampling of the silhouette input is finished, the input is resampled. The aim of the resampling is to find a rough estimate of the strokes provided by the user with fewer point samples, since

1. The user draws the silhouette of the object to be created
2. The 2D silhouette input is resampled
3. The 3D object is created
4. The user provides the shading strokes by drawing strokes inside the silhouette
5. A height field is constructed according to the shading input
6. The height field is applied to the mesh of the 3D object

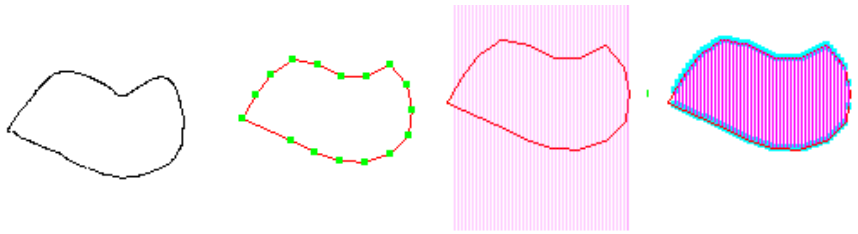
Figure 3.2: Overall procedure

the future calculations will require edge intersection test between sweep lines and the planar input polygon (Section 3.2.2). We employ a resampling method that is similar to the technique that was used in Teddy [1]. The ordered list of sampled input points are held in $P = \{p_0, p_1, p_2, ..., p_n\}$ where P denotes the list structure and p_i are the individual points where $i = 0, 1, ..., n$. The resampling algorithm is given in Figure 3.3.

```
   Input: P = {p_1, p_2, ..., p_n}, input points
   Output: R = {r_1, r_2, ..., r_m}, resampled points
[1]   R ← p_1
[2]   for p_i, i = 1 to n − 1, do
[3]       if distance(p_i, p_{i+1}) is bigger than or equal to IRC
[4]           R ← R + p_{i+1}
```

Figure 3.3: Input Resampling Algorithm

R stands for the list of resampled points and *IRC* is the Input Point Resampling Constant. After the resampling is over, a smaller number of input points is kept and the distances between adjacent points are close to *IRC*.



(a) Raw input (b) Resampled input (c) Sweep lines (d) Slabs

Figure 3.4: Different steps of our 2D silhouette processing algorithm.

3.2.2 3D Object Creation

After the input is resampled, a 2D closed polygon that represents the resampled input is constructed. Once we have the 2D closed polygon, we aim to create the 3D object's triangular mesh structure. During the triangulation, we elevate each of the vertices according to our vector elevation equation. The steps of this procedure can be seen in Figure 3.5.

---

1. Construct a 2D closed polygon from resampled points.
2. Determine whether the sweep lines should be parallel to x or y axis
3. Find the intersection points of each sweep line and edges of the polygon
4. Construct slabs by combining every two consecutive slab edge
6. Triangulate slabs to create the triangular mesh structure.
7. Elevate the resulting triangle points' z value so that each slab edge becomes an arc.

Figure 3.5: 3D Object Creation Procedure

In order to form the edges of the 2D closed polygon, adjacent vertices of the resampled input points are connected. Since the resampled input points are

ordered according to the time they were received, by connecting each point with the next one we are able to create the closed 2D polygon. The edge creation algorithm is given in Figure 3.6.

Input: $R = \{r_1, r_2, ..., r_m\}$, resampled input
Output: $E = \{(r_1, r_2), ..., (r_{m-1}, r_m), (r_m, r_1)\}$, polygon edges
[1] for r_i, $i = 1$ to m, do
[2] if i is equal to m
[3] $E \leftarrow E+(r_i, r_1)$
[4] else
[5] $E \leftarrow E+(r_i, r_{i+1})$

Figure 3.6: Silhouette Edge Creation Algorithm

When the 2D polygon is formed by connecting the resampled vertices, we determine whether the sweep lines are parallel to the x or y axis. To achieve this, we first find the largest and the smallest x and y values among the input points. After these points are found, we calculate the distance between the largest and smallest x and y values. We will call these calculated values $d_{max,x}$ and $d_{max,y}$ from now on. Then we compare the $d_{max,x}$ and $d_{max,y}$. If $d_{max,x}$ is bigger, we use sweep lines that are parallel to y axis. If $d_{max,y}$ is bigger, we use sweep lines that are parallel to x axis.

After the sweep axis is selected, we generate sweep lines and for each sweep line we find the intersection points between the sweep lines and polygon edges. If the lines are selected to be parallel to the y axis, the first sweep line starts with the same x value with the point within the resampled input that has the smallest x value. If the sweep lines are selected to be parallel to x axis, it intersects the point with the smallest y value. The sweep lines are generated so that each consecutive sweep line has a predefined spacing in between. This predefined length is called Sweep Line Spacing (SLS) . The equations of each sweep line are shown in the below equations:

$$y = y_{min} + i \times SLS \tag{3.1}$$

is used if sweep lines are selected to be parallel to x axis,

$$x = x_{min} + i \times SLS \tag{3.2}$$

is used if sweep lines are selected to be parallel to y axis.

In both equations SLS is the predefined measure that determines the resolution of the mesh to be created since there will be no more subdivision between the sweep lines and $i = 0, 1, ..., n$ where n makes the result of the equation equal to y_{max} or x_{max}. The next step is to find the intersection points between the polygon edges and the sweep lines (Figure 3.4c) and to create slab edges which are formed by connecting the two intersection points that are found for each sweep line (Figure 3.4d). The complete line generation and intersection finding algorithm is given in Figure 3.7.

```
   Input: E = {(r1, r2), ....., (rm−1, rm), (rm, r1)}, polygon edges
   Output: S = {(s1, t1), (s2, t2), ....., (sk, tk)}, slab edges
[1]   if sweep lines are parallel to the y axis
[2]      for y = ymin to ymax, increment y by SLS
[3]         a ← 0
[4]         for e starts with first element in E, until the last element of E
[5]            if y is between the y values of e
[6]               t ← intersection point of e and the current sweep line
[7]               a ← a + 1
[8]               if a is 1
[9]                  s ← t
[10]              else if a is 2
[11]                 S ← S+(s, t)
```

Figure 3.7: Sweep Line Generation and Intersection Finding Algorithm

This algorithm gives us slab edges. Any two consecutive slab edges form a slab with a predefined length (SLS) between them. Slabs are basically the polygons that are formed by connecting the two consecutive and parallel slab edges. They can be considered as thin plates that are constructed by dividing the polygon

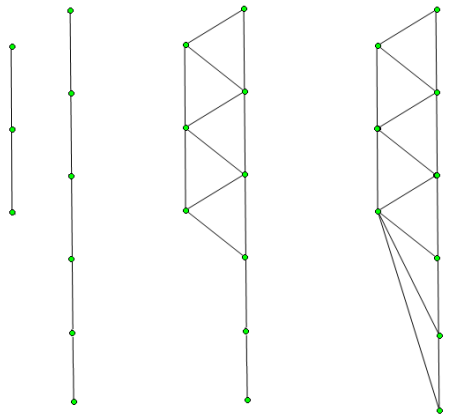


Figure 3.8: Slab Triangulation

with parallel lines, therefore when all slabs are considered, they form the polygon itself.

The rest of the 3D object creation process can be explained briefly with the following steps: When we have the slabs defined, we triangulate each slab. During the triangulation, we create points that are on the slab edges. When creating these points, an elevation offset for each point is calculated and applied to the point so that each slab forms an arc. The result of the triangulation with the elevated of vertices, is the triangular 3D object's mesh.

The aim of the triangulation algorithm we employed is to construct triangles with one of their edges, which we will call base edge from now on, on one of the slab edges and one point on the other slab edge. Each triangles base edge's length is determined by a predefined constant, Triangulation Edge Constant (TEC). The triangles are formed so that they fill the whole area inside a slab. The triangulation process is shown in Figure 3.8. In order to triangulate the slab edges, we first find equidistant points for the two slab edges that form the slab to be triangulated. The point finding algorithm for a slab edge is shown in Figure 3.9.

$$z = \sqrt{l^2 - ((x - m_x)^2 + (y - m_y)^2)} \times IC \tag{3.3}$$

```
   Input: (s, t), a slab edge
   Output: K = {(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_u, y_u, z_u), }, equidistant points found
[1]   if s.x − t.x is 0
[2]       dx ← 0
[3]       dy ← TEC
[4]   if s.y − t.y is equal to 0
[5]       dx ← TEC
[6]       dy ← 0
[7]   for x = t.x to s.x and
      y = s.y to t.y
      increment x by dx, increment y by dy
[8]       if x is bigger than s.x
[9]           x ← s.x
[10]      if y is bigger than s.y
[11]          y ← s.y
[12]      calculate corresponding z value according to Equation 3.3
[13]      K ← (x, y, z)
[14]      if x is equal to s.x and y is equal to s.y
[15]          break
```

Figure 3.9: Point Finding Algorithm

In Equation 3.3, l is the distance between the midpoint of the slab edge, x and y are x, y coordinates of the point and m_x, m_y are midpoint's x and y coordinates. IC stands for inflation constant and is used to control the flatness of the object created. If IC is 0, then the object is a flat surface, if IC is 1 then the resulting object has circular cross sections. This equation is used in order to create semi elliptic lines from slab edges, when the vertices are elevated. The z values of vertices of the slabs are calculated according to this equation, so that when the slabs will be constructed, they are shaped as a thin plate that is bent so that it forms an arc. A graphical description that shows the 3D form of a slab after the triangulation of slab edges with the height values of its vertices calculated using Equation 3.3 is given in Figure 3.10.

After points to be used in triangulation are created from both slab edges, we find the slab edge with the minimum number of points between the two edges that

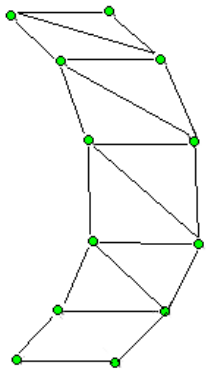


Figure 3.10: 3D Slab

form the slab. When we determine the edge with the smaller number of points, we use the algorithm shown in Figure 3.12 to create the triangles as shown in Figure 3.8. Note that for simplicity, Figure 3.8 does not show the result of the elevation of vertices. Figure 3.11 shows a wireframe model of a 3D mesh generated by our algorithm.

As a result of this procedure, we finish triangulating the slabs. When all slabs are triangulated, we have the half of the triangular mesh representing the 3D object. To create the other half, we just create a mirror image of the first half of the mesh. For simplicity, we build our mesh on $z = 0$ plane, so that the mirror image of any vertex with coordinates *x, y, z* in the mesh is a point with coordinates *x, y, -z*.

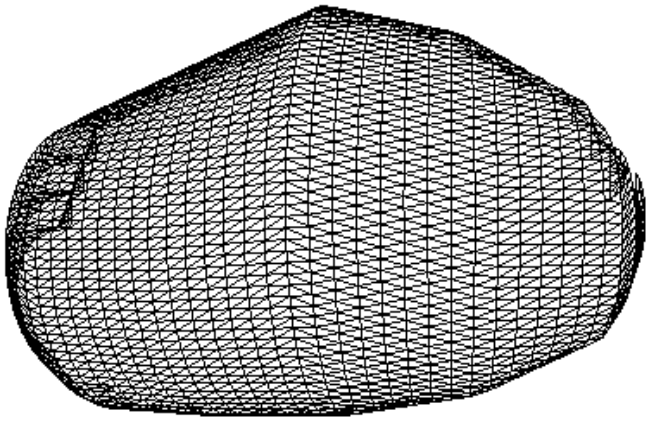


Figure 3.11: Wireframe 3D mesh

Input: $A = \{(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_i, y_i, z_i)\}$, finite point set
Input: $B = \{(\grave{x}_1, \grave{y}_1, \grave{z}_1), (\grave{x}_2, \grave{y}_2, \grave{z}_2), ..., (\grave{x}_j, \grave{y}_j, \grave{z}_j)\}$, finite point set
Assume $i < j$

```
[1]   p1 ← B.first
[2]   for p0 ← A.first to A.last
[3]       if p1 is B.end
[3]           break
[3]       createTriangle(p0, p1, p1.next)
[4]       if p0 is not A.last
[5]           createTriangle(p0, p0.next, p1.next)
[6]       p1 ← p1.next
[7]       if p0 is A.last
[8]           while p1 is not B.end
[9]               createTriangle(p0, p1, p1.next)
[10]              p1 ← p1.next
```

Figure 3.12: Triangulation Algorithm

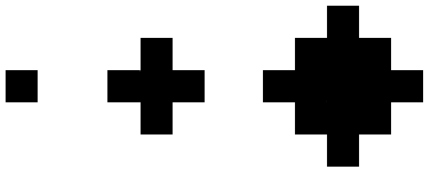


Figure 3.13: Brush sizes

3.2.3 Construction of the Heightfield

The shading strokes are stored as image pixels in order to make the transition of shading input to SFS input easier. Since the SFS input is image based, we use the shading data directly as an input to the SFS algorithm. We define a sketchpad, which is the area on the screen that user is able to draw onto. We get the input as it is drawn to the screen, meaning that whenever the pen of the tablet device touches the tablet or mouse's first button is pressed, we get the pixel coordinate information of the cursor whenever the input event is received. If the stroke input is continuous, then each time the stroke is sampled, each sampled pixel coordinate information is received. We store the shading information in a 2D array, where each element corresponds to a pixel on the screen. When the corresponding pixel in the screen is drawn the corresponding element in the 2D array is set.

In order to provide an easier form of interaction, we provide different brush sizes for the user. As the brush size gets bigger, the pixels that are set are increased. The three different sizes for the brush are shown in Figure 3.13. As it can be seen in the figure, the brushes are designed so that they draw circular points.

Once the shading input is done, our system requires the user to inform the system that the shading is over. When user finishes shading and notifies the system, the image based shading data is used as input to the SFS step. We use Tsai et al.'s SFS algorithm [32] for this step. Since the input is already a 2D array of pixels, it can be utilized as an image. We input the image data directly into the SFS algorithm and obtain the result as a height field.

Tsai et al.'s SFS algorithm employs a linear approach, meaning to find a surface shape from shading, the authors linearly approximate the reflectance function. In order to achieve this, the authors approximate the surface normal in a discrete manner. Then the authors linearize the reflectance function in depth. The overall procedure of Tsai et al.'s method is given in Figure 3.14. Refer to [32] for details of this algorithm.

---

1. The surface normal is approximated in a discrete manner.
2. Using these approximations with the reflectance function, the authors are able to linearly approximate a depth map using a Taylor series expansion for a fixed point.
3. Then for each point, the authors form a linear system.
4. Using the Jacobi iterative method, the linear system is solved.

Figure 3.14: Shape-from-shading method. Courtesy of Tsai et al. [32]

As a result of this process, we obtain a height field that contains depth perturbation values that are calculated from the shaded pixels. Since the user input is image based, each element in the height field contains depth information corresponding to the pixel that has its indices as coordinates. In other words, we can obtain the depth perturbation value of a point in the 3D mesh by giving its x and y coordinates to the height field as indices and obtaining the corresponding depth value.

3.2.4 Application of the Heightfield on the 3D Mesh

After the 3D object mesh is constructed, as the next step, we apply the height field. Each element in the height field has values between 0 and 1, indicating the

height value to be applied. In order to apply the height values to the mesh vertices, we calculate a depth elevation value from the information we acquire from the height field. For each vertex, we first find the nearest 9 depth perturbation values from the height field. We use 9 values, one of them being the height value that corresponds to the vertex itself and 8 other height values that belongs to the adjacent elements in the height field. In order to get the depth perturbation values, we simply floor the x and y coordinates of the vertex and we employ the following equation to calculate the depth perturbation value where Δz is the depth perturbation value of the vertex with coordinates x, y.

$$\Delta z = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \frac{heightField(x+i, y+j)}{9} \tag{3.4}$$

Chapter 4

Results and Discussion

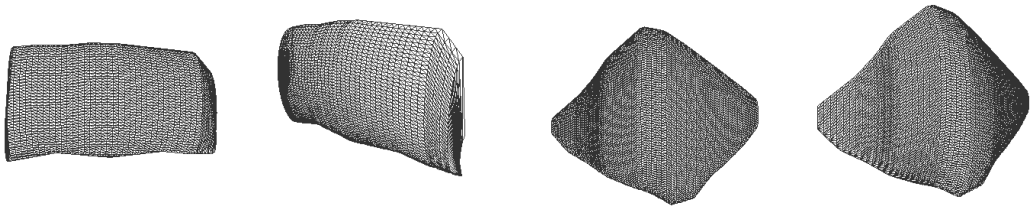
4.1 3D Object Types

Here we provide 3D free-form object meshes that are constructed using our 3D object creation system. The input is represented as a 2D closed polygon after resampling, so we evaluate different input examples which form different types of polygons after resampling. Our system is able to construct 3D objects from monotone polygons, which means the input polygon P has exactly two intersection points with every line orthogonal to a line L [55]. This obvious limitation origins from the sweep method we employ for the creation of 3D shapes.

We will provide several example meshes that are created using our object creation method. We present several wireframe object meshes that are constructed from 2D convex and concave polygon inputs. Note that these objects have been created using silhouette input only: The meshes presented here are only 3D objects without any height fields applied. In all of the examples presented here, Sweep Line Spacing (SLS) is 1, Inflation Constant (IC) is 0.2 and Input Resampling Constant (IRC) is 10. Only half of each object is visible in these images to provide clearer images of the objects. The other half of the objects can be predicted easily since it is the mirror image of the part shown.

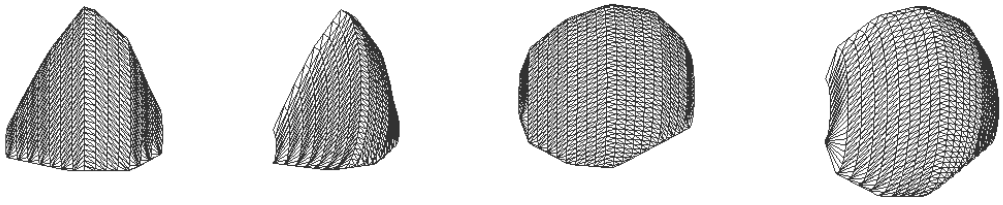
(a) (b) (c) (d)

Figure 4.1: Sketches of 3D objects constructed with convex polygon inputs



(a) Rectangular object (b) Diamond shaped object

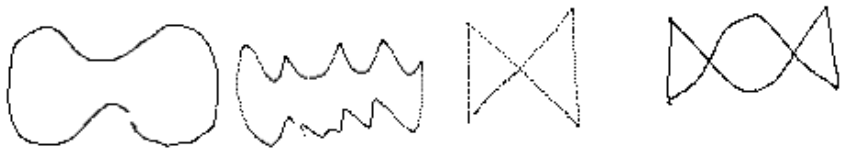
Figure 4.2: Different wireframe examples of 3D objects constructed with rectangular convex polygon inputs



(a) Triangle shaped object (b) Circular object

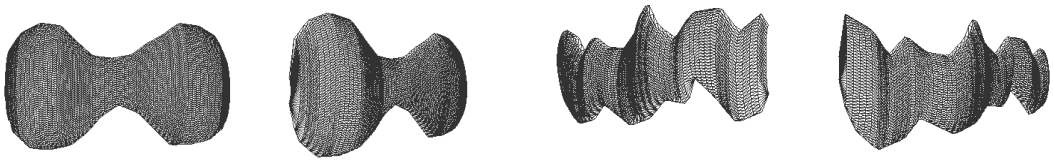
Figure 4.3: Different wireframe examples of 3D objects constructed with convex polygon inputs

All of the examples in Figures 4.2 and 4.3 are created by providing simple silhouette strokes. All of these objects are created using sketches of convex polygons. The objects are fairly simple and the results are intuitive. The small inflation constant makes objects flat while a higher value would make them more round.



(a) (b) (c) (d)

Figure 4.4: Sketches of 3D objects constructed with concave polygon inputs

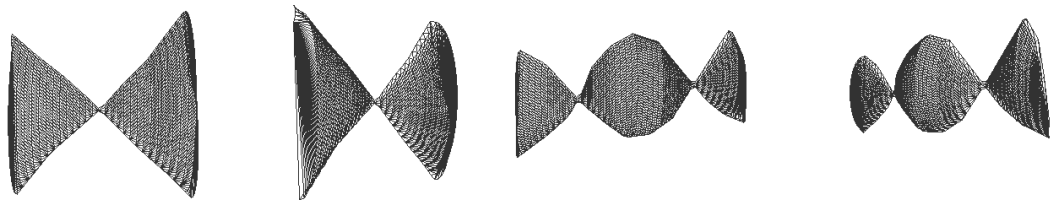


(a) Peanut shaped object (b) Arbitrary shaped object

Figure 4.5: Different wireframe examples of 3D objects constructed with concave polygon inputs

In Figure 4.5, we see several wireframe models of different objects. All these objects are constructed using 2D concave polygonal inputs. Our system's limitation can be seen here, all these concave objects are monotone. Thus, for example, an input that resembles the outline of a "c" letter can not be processed by our system.

Our system also supports self intersecting inputs as long as they are monotone. Several objects that are constructed from complex polygon inputs are presented in Figure 4.6.



(a) Butterfly shaped object (b) Candy wrapper shaped object

Figure 4.6: Different wireframe examples of 3D objects constructed with complex polygon inputs

4.2 Parameters

4.2.1 Resampling

The resampling is achieved with the employment of Input Point Resampling Constant (IRC). The effect of this parameter is described formally in Section 3.2.1. In this section, we discuss resulting polygons that are created using different values of IRC. In Figure 4.7 different polygons created using different IRC values are shown. Notice that with a large IRC the resulting polygon does not resemble the input. We selected IRC to be 10 in all our experiments, which generally produces good results.

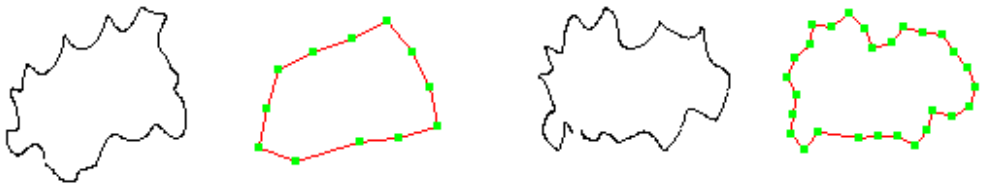
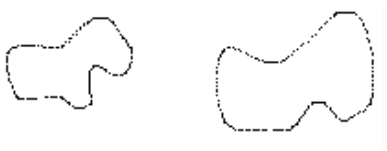


Figure 4.7: Resampling examples for different values of Input Point Resampling Constant (IRC). IRC is selected 20 for the input on the left and IRC is 10 for the input on the right.

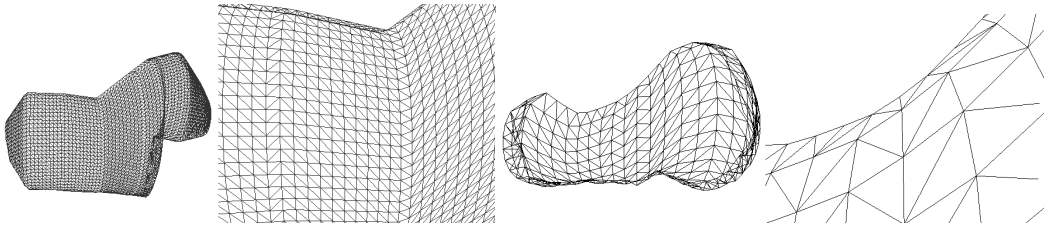
4.2.2 Sweep Line Spacing

The Sweep Line Spacing (SLS) is the distance between adjacent sweep lines. This constant is effective both in the resulting 3D mesh and also affects the final, height field applied mesh. Since the SLS directly affects the vertex density of the mesh and the discrete values in the height field are applied to the nearest vertices, unwanted results such as jagged perturbations on the mesh surface may occur if SLS is too large after the application of the height map. In both examples, Inflation Constant (IC) is 0.2 and Inflation Resampling Constant (IRC) is 10.



(a) (b)

Figure 4.8: Sketches of 3D objects constructed for SLS Testing



(a) SLS = 1 (b) SLS = 5

Figure 4.9: Wireframe examples of 3D objects with different SLS values

4.3 Shading Effects

In this section, we provide several objects created with our system. In these examples, we used complete input from the user, which involves both silhouette and shading and discuss several aspects of these results, such as the overall shape, the effects of the shading and the final image of the object. We show wireframe

models and textured images of the resulting objects. All of the meshes here are constructed with our method on a regular PC in 1 or 2 minutes after the user finishes sketching. The bottleneck is the creation of triangles. In order to provide satisfying results, our system requires a dense field of triangles representing the object; this lowers the performance. All the objects created here is resampled with IRC equal to 10. Other constants that affect the resulting mesh is mentioned for each of the meshes.

Figure 4.10: Dress sketch

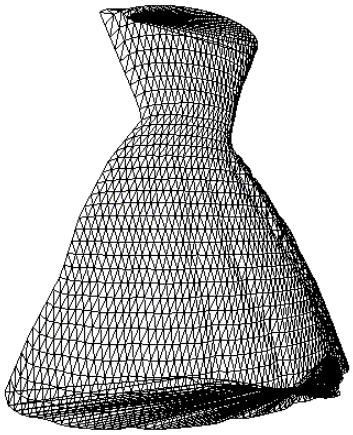


Figure 4.11: Wireframe model of a dress created with our system

In the first example, we present a dress modeled with our system. The sketch of the dress is shown in Figure 4.10 and the resulting object is shown in Figure 4.12. The sketch is very simple and it takes 1 or 2 minutes at most to draw such a sketch. The silhouette of the sketch is shaped like a sand glass. The straight lines inside the bottom part of the sketch makes the curves at the skirt part of the final mesh. The SLS is 1 and IC is 0.7 in this example which causes the object

Figure 4.12: Textured model of a dress created with our system

to have elliptic cross sections, therefore creating a 3D shape of a sand glass with elliptic cross sections which is intuitive since a silhouette of a sand glass was the input. The curves are a result of the height field calculated from the shading input combined with the conical shape of the skirt which makes the curves more obvious. Note the cylindrical shapes of the curves at the bottom end of the skirt. The curves formed on the surface of the skirt are not perfectly symmetrical, this is caused by the input error caused by the user. However non symmetrical shapes provide a more realistic final shape, as a cloth rarely has symmetric folds.

Figure 4.13: Almond sketch

The next example is an almond. The sketch of the almond is shown in Figure 4.13 and the resulting object is shown in Figure 4.15. In the sketch, silhouette of an almond is drawn from its side. The shading input is arbitrary shaped, noisy lines drawn inside the almond silhouette. These lines form the detailed surface

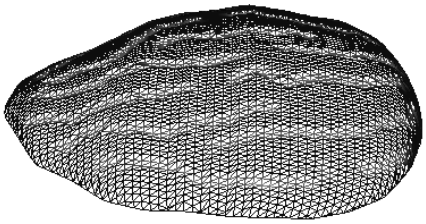


Figure 4.14: Wireframe model of an almond created with our system

Figure 4.15: Textured model of an almond created with our system

of the final mesh. The overall shape of the object is a result of its silhouette being swept through x axis with elliptic curves. SLS is 1 and IC is 0.6 in this example, causing the object to be relatively more flat than the dress mesh. The surface geometry is perturbed in the shape of noisy lines, which provides realistic results since the surface of an almond is as such. The texture applied to this example, makes the object more realistic, but the deformations can be seen in the wireframe model are similar to the ones in a real almond's surface. In the final shape, we can see the perturbations make the final mesh look more realistic and deformations visible.

One final comment on the almond example is about the triangle formation between the slabs. The triangles on the borders of the mesh are not very uniform as can be seen in Figure 4.14. This is caused by our triangulation algorithm but since these triangles cover a small area and are almost never perturbed, they do not cause any problems as can be seen in this example.

Figure 4.16: Fish sketch

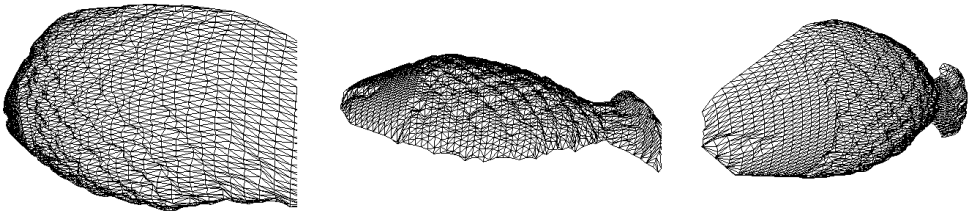


Figure 4.17: Wireframe model of a fish created with our system

We present a fish object as our next example. The sketch of the fish is shown in Figure 4.16 and the resulting object is shown in Figure 4.18. The sketch is

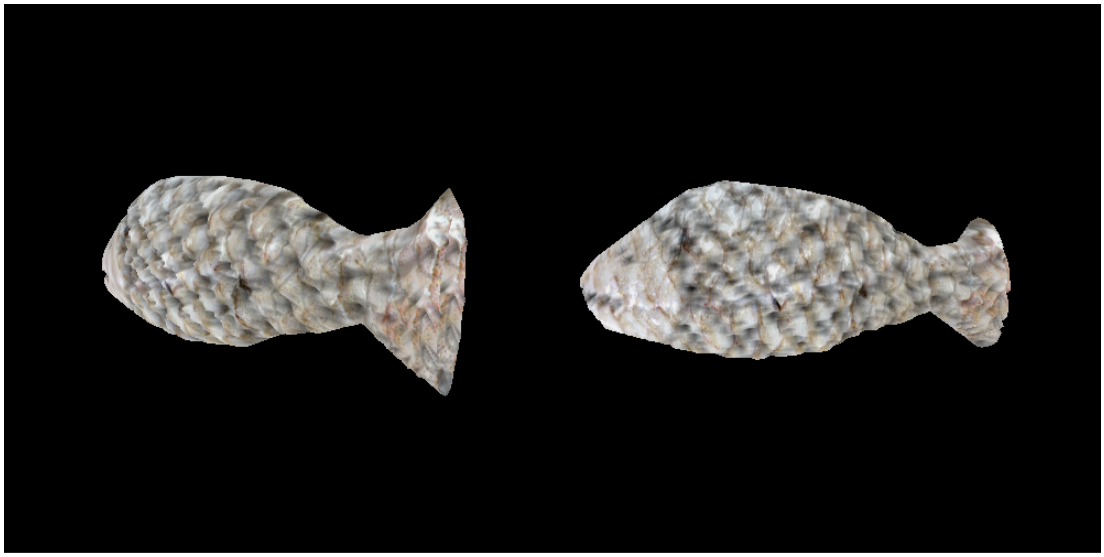


Figure 4.18: Textured model of a fish created with our system

the silhouette of a fish with the scale of the fish drawn inside. The scale drawing is an example for the effects of the patterned shading input on our system. The overall shape of the object is the result of the silhouette being swept on the x axis. The body and the tail is shaped as a fish with elliptic cross sections, as the IC is 0.6. We selected SLS to be 1 in this example. The surface perturbations are noisy bumps that are formed by straight line shaped dents with a diamond shaped pattern around them. The noisy effect is not very desirable in this case but still the final image is convincing. The diamond shaped deformation on the final object's geometry provides a more visible depth to the viewer but it might be it is a little too noisy.

The next two examples are $2\frac{1}{2}$D objects. They are constructed with our system, but the mesh vertices are not elevated when creating the mesh, therefore the results are planes in the shape of the silhouette. This is achieved by simply assigning $IC = 0$. SLS is 1 in these examples. The first objects is a leaf and the second one is a wall. The sketch of the leaf is shown in Figure 4.19 and the resulting object is shown in Figure 4.21. The sketch of the wall is shown in Figure 4.22 and the resulting object is shown in Figure 4.24.

The leaf sketch is composed of a leaf silhouette and the shading of its veins.

Figure 4.19: Leaf sketch

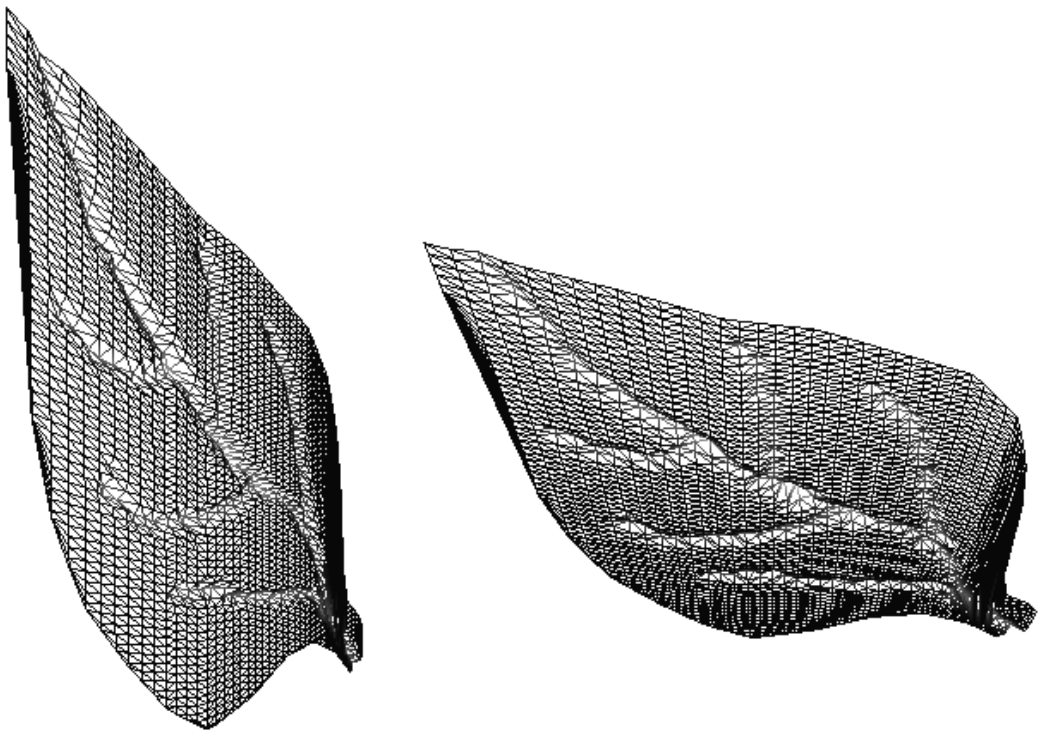


Figure 4.20: Wireframe model of a leaf created with our system

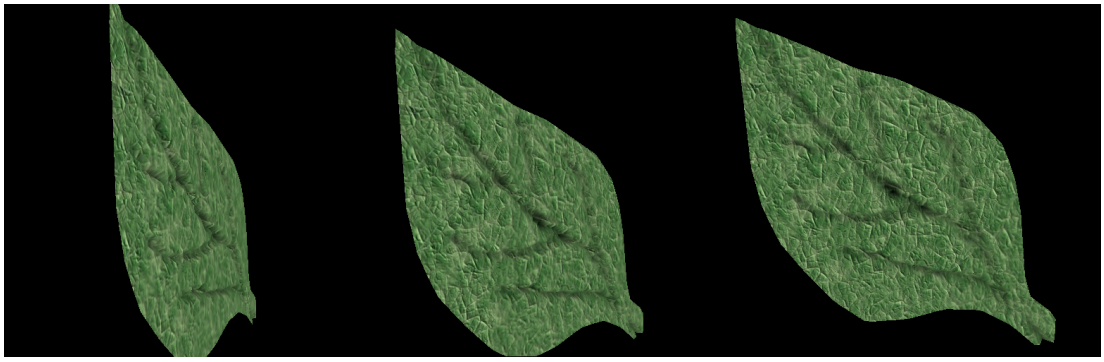


Figure 4.21: Textured model of a leaf created with our system

The curves in the sketch are applied exactly to the final mesh. As it can be seen from Figure 4.20, the nature of our image based technique causes some noise in the curvy perturbations, but they are only visible when camera is very close to the object.

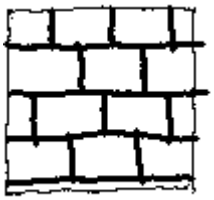


Figure 4.22: Wall sketch

The wall object is shaped as the sketch exactly, which is problematic since noise caused by the user prevents the object to be shaped as a perfect rectangle. As it can be seen from Figure 4.24, using our system, a brick illusion is created with only a sketch with simple, straight lines. The user inflicted noise of the straight lines makes the final shape look imperfect, which makes the final shape more realistic. However, we are aware that user might want bricks or any other 3D shape with a perfectly geometrical shape. Our system is unable to produce such objects.

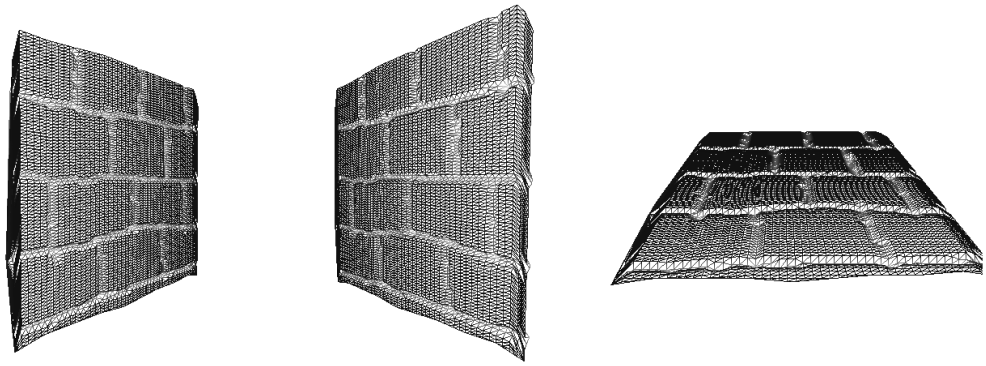


Figure 4.23: Wireframe model of a wall created with our system

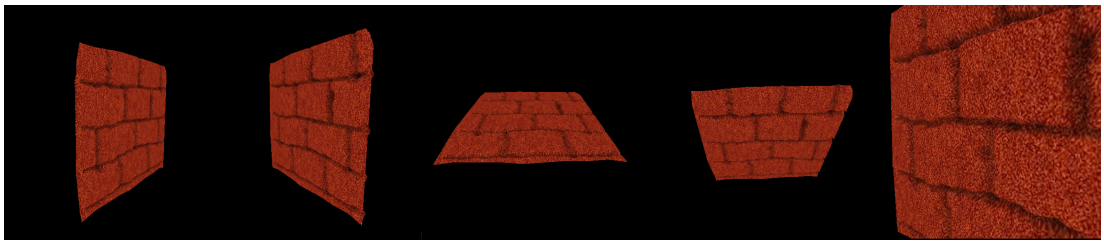


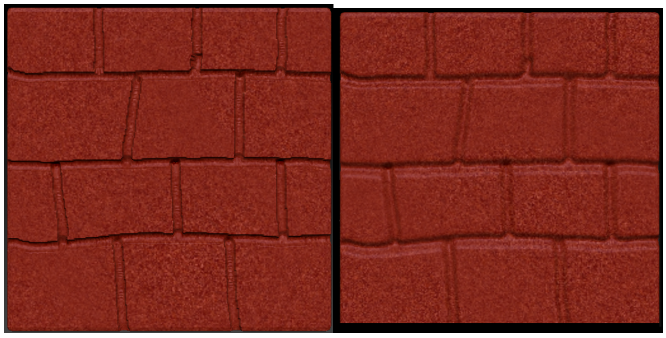
Figure 4.24: Textured model of a wall created with our system

4.4 System Comparison

We compare our results with the state-of-the-art. Contemporary modeling techniques do not provide construction of 3D objects with perturbed free-form surfaces that is based solely on sketch input. There are systems that allow $2\frac{1}{2}$D object creation from sketches [9] or only surface geometry modification techniques [8]. There is, however, the renowned ZBrush [51], a commercial modeling tool that allows depth painting as well as modeling operations. As our system is not a match to the wide range of tools that is provided by ZBrush and most of these tools are out of scope for our system, we only compare results that are obtained with our system and with ZBrush's depth painting operations. We used objects that can be modeled with only basic sculpting operations using strokes, rather than complex modeling operations that are out of our scope.

We compare two of our models that we created in the last section. To compare the depth modification features, we compare the wall and leaf models with ours.

The wall model that is created with ZBrush is presented in Figure 4.25a and a smoothed version is seen in Figure 4.25b. The time and number of strokes to create the mesh is nearly same with our system. In order to create this mesh, the user draws straight lines that form the outer shapes of the bricks.



(a) (b)

Figure 4.25: Textured model of a wall created with ZBrush. (b) is the smoothed version of (a)

The other example is the leaf model as shown in Figure 4.26. In this model, a thick plane is carved with a depth painting brush as in the wall example. The veins of the leaf are easy to create, since they only require a few strokes. The drawing time of the curves and the number of strokes are the same as our system, whereas creation of the object takes a little more time in our system.

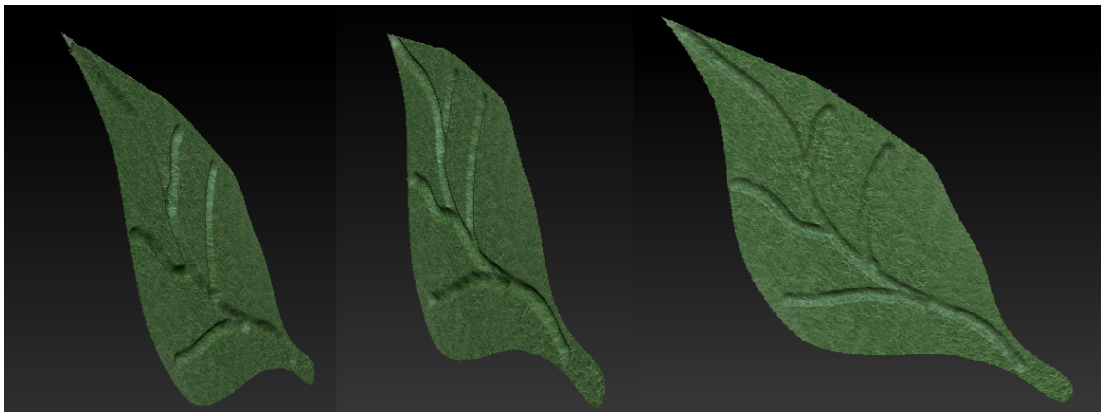


Figure 4.26: Textured model of a leaf created with ZBrush

The fundamental difference between our system and ZBrush is the creation method in the sense that we allow the user to draw a sketch on a blank canvas,

which is used to create the 3D mesh of the final object after the sketch is over. In ZBrush, user draws depth modifying strokes on a given object interactively. This may be desirable in some cases, although our system has its benefits. It is easier to learn and very easy to use. Compared to the methods used in ZBrush, from the user's perspective, our system uses techniques similar to traditional sketching whereas ZBrush uses sculpturing. The results of our system are in similar quality with the results that are obtained with ZBrush's sculpturing techniques. We suggest that our method of free-form object creation can be used as a part of a bigger modeling tool, where it can be used for easy, intuitive interaction and can be incorporated with other modeling techniques to create more complex meshes.

Chapter 5

Conclusion

We have proposed a sketch based modeling system that is used to construct 3D free-form objects with perturbed surfaces by utilizing only sketch input. Our system requires the user to provide a sketch of an object. The sketch should depict the silhouette of the object to be created, which is used in the creation of a 3D mesh. When the user is done with the silhouette input, to apply perturbations on the surface of the object, the user should draw shading strokes inside the silhouette. Our system is a hybrid of two methods. We employ a technique that creates 3D objects from sketches and one that extracts height fields from shading information. We combine the two methods in order to create a system which constructs 3D objects with perturbed surfaces using sketch input.

In order to create the 3D mesh, we first get the silhouette input as points from the user. Then we resample the input points so that we have points that form a 2D polygon that represents the silhouette's shape. Then we sweep the 2D polygon with lines that are parallel to either x or y axis. The axis selection is made according to the largest distance value among the x and y values of the points of the polygon. We sweep the polygon with sweep lines that are orthogonal to the axis with the largest distance. During sweeping, we find two intersection points with each sweep line and the 2D polygon edges. By connecting these two intersection points, we form slab edges. We define slabs, which are composed of any two adjacent slab edges. Then, each slab is triangulated. During the

triangulation, we elevate the vertices so that each slab edge forms a semi elliptic curve. We form a 3D mesh by forming triangular mesh patches for each of the slabs.

The free-form surface of the mesh is obtained by utilizing the shading input the user provides. We receive the shading input as an image based input. The shading input strokes are stored as pixels in an image. When the user finishes shading, the image based shading input is used to calculate a height field by a shape from shading algorithm. The height field is then applied to the 3D object mesh to finalize the free-form 3D object mesh.

Our results show that our system is capable of creating 3D objects with free-form surfaces using only sketches. Our system is easy to use and intuitive. However, we can only process silhouette input that forms a monotone polygon when resampled. This makes objects with a branching skeletal structure out of our scope. Also, the image based height field modification technique brings a minor fidelity problem. This causes noise on the perturbations, which is problematic when using curves. On the other hand, the noise is desired in some cases, such as the almond or brick wall example in which our system provides better results.

This work can be improved by incorporating a 3D object creation method that allows creation of more complex object with branching skeletal structures. We have observed that modification of the surface geometry is possible with a height field generated from shading input. However, creating an object with much precision is very hard so using this kind of an approach for engineering design systems is not feasible. This method can be used as a technique to create simple objects with free-form surfaces or can be incorporated with other systems in order to construct more complicated objects.

Bibliography

[1] Igarashi T, Matsuoka S, Tanaka H. Teddy: a sketching interface for 3D freeform design. In: Proceedings of the SIGGRAPH'99, pp. 173-174, 1999.

[2] Henzen, Alex, Ailenei, Neculai, Fiore, Fabian Di, Reeth, Frank Van, Patterson, John, 2005. Sketching with a low-latency electronic ink drawing tablet. In: Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia.

[3] Cherlin JJ, Samavati F, Sousa MC, Jorge JA. Sketch-based modeling with few strokes. In: Proceedings of Spring Conference on Computer Graphics (SCCG'05), 2005.

[4] Eggli L, Ching-Yao H, Bruderlin B, Elber G. Inferring 3D models from freehand sketches and constraints. Computer-Aided Design, Vol. 29, pp. 101-112, 1997.

[5] Pusch R, Samavati F, Nasri A, Wyvill B. Improving the sketch-based interface: forming curves from many small strokes. In:Proceedings of Computer Graphics International (CGI'07), Vol. 23, pp. 955-962, 2007.

[6] Fleisch T, Rechel F, Santos P, Stork A. Constraint stroke-based oversketching for 3D curves. In: Proceedings of Eurographics Workshop on Sketch-based Interfaces and Modeling (SBIM'04), pp. 161-165, 2004.

[7] Kara L, D'Eramo C, Shimada K. Pen-based styling design of 3D geometry using concept sketches and template models. In:Proceedings of ACM Solid and Physical Modeling Conference (SPM'06), Vol. 27, pp. 60-71, 2006.

[8] Rushmeier, Holly, Gomes, Jose, Balmelli, Laurent, Bernardini, Fausto, Taubin, Gabriel. Image-based object editing. In: Proceedings of Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM'03), pp. 20-27, 2003.

[9] Kerautret, Bertrand, Granier, Xavier, Braquelaire, Achille, 2005. Intuitive shape modeling by shading design. In: Butz, Andreas, Fisher, Brian, Krger, Antonio (Eds.), International Symposium on Smart Graphics Lecture Notes in Computer Science, Vol. 3638, pp. 163-174. Springer-Verlag GmbH.

[10] Sezgin TM, Stahovich T, Davis R. Sketch based interfaces: early processing for sketch understanding. In:Proceedings of Workshop on Perceptive User Interfaces (PUI'01), pp. 1-8, 2001.

[11] Kurozumi Y, Davis W, Polygonal approximation by the minimax method. Computer Graphics and Image Processing, 1982.

[12] Saykol, Gudukbay U, Gulesir G, Ulusoy O. KiMPA: a kinematics-based method for polygon approximation. Lecture Notes in Computer Science, Vol. 2457, pp. 186-194. Berlin/Heidelberg: Springer; 2002.

[13] Olsen L, Samavati F, Sousa M, Jorge J. Sketch-based modeling: A survey. Computers and Graphics, Vol. 33, pp. 85-103 (2008).

[14] Cook M. T., Agah A. 2009. A survey of sketch-based 3-D modeling techniques. Interacting with Computers, Vol. 21, pp. 201-211.

[15] Zeleznik, Robert C, Herndon, Kenneth P, Hughes, John F, 1996. SKETCH: an interface for sketching 3D scenes. In: Rushmeier, Holly (Ed.), SIGGRAPH96. Addison Wesley, pp. 163-170.

[16] Pereira, Joo P., Jorge, Joaquim A., Branco, Vasco, Ferreira, F., Nunes, 2000. Towards Calligraphic Interfaces: Sketching 3D Scenes with Gestures and Context Icons. In: Proceedings of WSCG'00, 2000.

[17] Pereira, Joo P, Branco, Vasco A, Jorge, Joaquim A, Silva, Nelson F, Cardoso, Tiago D., Ferreira, F., Nunes. 2004. Cascading recognizers for ambiguous

calligraphic interaction. In: Hughes, John F., Jorge, Joaquim A. (Eds.), EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling.

[18] Huffman, David A, 1971. Impossible objects as nonsense sentences. Machine Intelligence. Vol 8, pp. 475-492.

[19] Clowes, Maxwell B, 1971. On seeing things. Artificial Intelligence Vol. 2, pp. 79-116.

[20] Grimstead, Ian J, Martin, Ralph R, 1995. Creating solid models from single 2D sketches. In: SMA95: Proceedings of the Third Symposium on Solid Modeling and Applications. ACM Press.

[21] Turner, Alasdair, Chapman, David, Penn, Alan, 1999. Sketching a virtual environment: modeling using line-drawing interpretation. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp. 155161.

[22] Lipson, Hod, Shpitalni, Moshe, 2002. Correlation-based reconstruction of a 3D object from a single freehand sketch. In: Davis, Randall, Landay, James, Stahovich, Tom (Eds.), American Association for Artificial Intelligence Spring Symposium: Sketch Understanding. Stanford University in Palo Alto, AAAI, California.

[23] Williams, Lance. 3D paint. In: Proceedings of the 1990 Symposium on Interactive 3D graphics, Vol. 24, pp. 225-233.

[24] L. Prasad. Morphological analysis of shapes. CNLS Newsletter, Vol. 139, pp. 1-18.

[25] Karpenko, Olga, Hughes, John F, Raskar, Ramesh. Free-form sketching with variational implicit surfaces. In: Proceedings of Computer Graphics Forum, Vol. 21, pp. 585-594, 2002.

[26] Turk G, O'Brien, J. Shape Transformation using variational implicit functions. In Proceedings of SIGGRAPH'99 (August 1999), pp. 13.

[27] Tai, Chiew-Lan, Zhang, Hongxin, Fong, Jacky Chun-Kin, 2004. Prototype Modeling from sketched silhouettes based on convolution surfaces. Computer Graphics Forum 23 (1), 71-83.

[28] Schmidt, Ryan, Wyvill, Brian, Sousa, Mario Costa, Jorge, Joaquim A, 2005. ShapeShop: sketch-based solid modeling with the BlobTree. In: 2nd Eurographics Workshop on Sketch-based Interfaces and Modeling.

[29] Wesche, Gerold, Seidel, Hans-Peter, 2001. FreeDrawer: a free-form sketching system on the responsive workbench. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology. ACM Press.

[30] Michalik, Paul, Kim, Dae Hyun, Bruderlin, Beat, D., 2002. Sketch- and constraintbased design of B-spline surfaces. In: ACM Symposium on Solid and Physical Modeling Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications. ACM Press, Saarbrcken, Germany.

[31] Levet, Florian, Granier, Xavier, Schlick, Christophe, 2006. 3D sketching with profile curves. In: Proceedings of International Symposium on Smart Graphics, Vol. 4073, pp. 114-125.

[32] P.S. Tsai and M. Shah, Shape from Shading Using Linear Approximation, Image and Vision Computing J., vol. 12, no. 8, pp. 487-498, 1994.

[33] Wyvill, Brian, Guy, Andrew, 1998. The BlobTree warping, blending and boolean operations in an implicit surface modeling system. Computer Science Technical Reports.

[34] Frisken, Sarah F, Perry, Ronald N, Rockwood, Alyn P., Jones, Thouis R., 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In: Akeley, Kurt (Ed.), Siggraph 2000, Computer Graphics Proceedings. ACM Press/ACM SIGGRAPH/ Addison Wesley Longman.

[35] Bærentzen, J. Andreas, Christensen, Niels Jrgen. 2002. Volume sculpting using the levelset method. In: Proceedings of the Shape Modeling International 2002 (SMI02). IEEE Computer Society.

[36] Lawrence, Jason, Funkhouser, Thomas. A painting interface for interactive surface deformations. In: Pacific Graphics, Vol. 66, pp. 418-438, 2003.

[37] Singh, Karan, Fiume, Eugene, 1998. Wires: a geometric deformation technique, In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, pp. 405-414.

[38] Cohen, Jonathan M, Markosian, Lee, Zeleznik, Robert C, Hughes, John F. An interface for sketching 3D curves. In: Proceedings of the ACM Symposium on Interactive 3D Graphics. ACM Press, Atlanta, Georgia. 1999

[39] Grossman, Tovi, Balakrishnan, Ravin, Kurtenbach, Gordon, Fitzmaurice, George, Khan, Azam, Buxton, Bill. Interaction techniques for 3D modeling on large displays. In: SI3D01: Proceedings of the Symposium on Interactive 3D Graphics. ACM Press. 2001

[40] Tsang, Steve, Balakrishnan, Ravin, Singh, Karan, Ranjan, Abhishek. A suggestive interface for image guided 3D sketching. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, pp. 591-598, 2004

[41] Hayward V, Oliver R, Cruz-Hernandez M, Grant D, Robles-De-La-Torre G, 2004. Haptic interfaces and devices, Sensor Review, Vol. 24, pp. 16-29.

[42] Burdea G, Coiffet P, Virtual Reality Technology, John Wiley&Sons, New York, 1994.

[43] Contero M, Naya F, Jorge J, Conesa J. CIGRO: a minimal instruction set calligraphic interface for sketch-based modeling. Lecture Notes in Computer Science, Vol. 2669, pp. 549-558.

[44] Masry M, Lipson H. A sketch-based interface for iterative design and analysis of 3D objects. In: Proceedings of the Eurographics Workshop on Sketch-based Interfaces and Modeling (SBIM'05), 2005.

[45] Piquer A, Martin R R, Company P. Using skewed mirror symmetry for optimisation-based 3D line-drawing recognition.In: Proceedings of IAPR International Workshop on Graphics Recognition, 2003.

[46] Draper G, Egbert P. A gestural interface to free-form deformation. In: Proceedings of Graphics Interface 2003.

[47] Kho Y, Garland M. Sketching mesh deformations. In: ACMSI3DG: symposium on interactive 3D graphics and games 2005, 2005.

[48] Rivers A, Durand F, Igarashi T. 2010. 3D Modeling with Silhouettes. ACM Trans. Graph. 29, 4, Article 109 (July 2010), 8 pages.

[49] Stiver M, Baker A, Runions A, Samavati F. 2010. Sketch based volumetric clouds. In Proceedings of the 10th International Conference on Smart Graphics (SG'10). Springer-Verlag, Berlin, Heidelberg.

[50] Yang R., Wunsche B. 2010. Life-sketch: a framework for sketch-based modelling and animation of 3D objects. In Proceedings of the Eleventh Australasian Conference on User Interface - Volume 106 (AUIC '10), Christof Lutteroth and Paul Calder (Eds.), pp. 61-70.

[51] Pixologic, 2007. ZBrush www.pixologic.com/zbrush.

[52] AutodeskInc.,Maya www.autodesk.com/mayai.

[53] AutodeskInc.,3D Studio Max usa.autodesk.com/3ds-max

[54] AutodeskInc.,Mudbox hwww.mudbox3d.comi.

[55] Franco P. Preparata and Michael Ian Shamos (1985). Computational Geometry - An Introduction. Springer-Verlag.