To the Graduate Council

I am submitting herewith a thesis written by Ernesto Lautaro Juarez-Valdes entitled "Complex Scene Modeling and Segmentation with Deformable Simplex Meshes" I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering

M A Abidi, Major Professor

We have read this thesis
and recommend its acceptance

Accepted for the Council

Interim Vice Provost and
Dean of The Graduate School

Complex Scene Modeling and Segmentation with Deformable Simplex Meshes

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Ernesto Lautaro Juarez Valdes

December 2000

Para Bris, Lola, Manda y Miguel

Acknowledgments

This work was supported by DOE's University Research Program in Robotics (Universities of Florida, Michigan, New Mexico, Tennessee, and Texas) under grant DOE-DE-FG02-86NE37968.

The author was also supported by a scholarship granted by the National Council for Science and Technology (CONACYT) of Mexico.

Abstract

In this thesis we present a system for 3D reconstruction and segmentation of complex real world scenes. The input to the system is an unstructured cloud of 3D points. The output is a 3D model for each object in the scene. The system starts with a model that encloses the input point cloud. A deformation process is applied to the initial model so it gets close to the point cloud in terms of distance, geometry and topology. Once the deformation stops the model is analyzed to check if more than one object is present in the point cloud. If necessary a segmentation process splits the model into several parts that correspond to each object in the scene. Using this segmented model the point cloud is also segmented. Each resulting sub-cloud is treated as a new input to the system. If, after the deformation process, the model is not segmented a refinement process improves the objective and subjective quality of the model by concentrating vertices around high curvature areas.

The simplex mesh reconstruction algorithm was modified and extended to suit our application. A novel segmentation algorithm was designed to be applied on the simplex mesh.

We test the system with synthetic and real data obtained from single objects, simple, and complex scenes. In the case of the synthetic data different levels of noise are added to examine the performance of the system. The results show that the systems performs well for either of the three cases and also in the presence of low levels of noise.

Contents

1 Introduction 1

1.1 Problem Statement 1

1.2 Strategy. 5

1.3 Thesis Outline. 7

2 Review of 3D Reconstruction and Segmentation 8

2.1 3D Reconstruction 8

2.1.1 Interpolation Techniques 9

2.1.2 Approximation Techniques. 11

2.2 Segmentation 15

3 Deformable Simplex Meshes 18

3.1 Definition and Characteristics 18

3.2 Geometry 20

3.3 Deformation 25

3 3.1 Internal Force 26

3 3 2 External Force 27

3 4 Mesh Refinement 28

4 Complex Scene Modeling and Segmentation 30

4 1 Deformation 31

4 1 1 Initialization 31

4 1 2 Shape Constraint 33

4 1 3 Data Constraint 39

4 1 4 Stopping Criteria 41

4 2 Segmentation 44

5 Results 53

5 1 Single Object 3D Reconstruction 54

5 2 Scene Reconstruction-Segmentation 68

6 Conclusions and Future Work 81

Bibliography 85

Vita 94

List of Tables

4.1 Values for the external force weight, β 41

5.1 Deformation parameters for single object models 57

5.2 Deformation and segmentation parameters for scene models 68

List of Figures

1.1 Overall strategy for 3D reconstruction. 6

3.1 Examples of k-simplexes. 19

3.2 Simplex mesh and triangulation are dual of each other. 19

3.3 The simplex angle in a planar curve. 21

3.4 Definition of simplex angle 21

3.5 Cross section of figure 3.4 through a plane defined by O_i, C_i and P_i. The simplex angle can be seen as a planar angle. 23

3.6 An inversion of center changes the definition of simplex angle to make it more suitable for computation. 24

3.7 When calculating the external force the function $G(\cdot)$ limits the influence of the closest point if it is too far away. This function appeared originally in [2]. 28

4.1 The input scenes to the system include information from one or multiple objects and also background information. (a)Scene of a couple of boxes including the floor and wall information. (b)Scene of multiple objects on a flat surface. 32

4.2 Tessellation of a sphere (a)A sphere is triangulated. (b)The dual simplex mesh has all faces hexagons but six. 32

4.3 Simplex mesh deforming using mean curvature continuity constraint. There are 35000 points in the dataset and the simplex mesh has only 3000 vertices. (a) Initial mesh enclosing the data set. (b) Mesh after 513 iterations (c) Mesh after 730 iterations. (d) Final model obtained after 1214 iterations. Note the lack of smoothness in comparision with 4.4 (d). 35

4.4 Simplex mesh deforming using surface orientation continuity constraint. There are 35000 points in the dataset and the simplex mesh has only 3000 vertices.(a) Initial mesh enclosing the data set. (b) Mesh after 513 iterations (c) Mesh after 730 iterations. (d) Final model obtained after 1214 iterations. Note the smoothness in comparision with 4.3 (d). . . . 36

4.5 A simplex mesh deforming with constant α_i loses information. To solve this problem a variable α_i should be used. The simplex mesh after 150 iterations with (a) constant (= 0.5) and (b) variable α_i. 37

4 6 Examples of calculation of ζ. (a) In a clean data set, no missing data or outliers $\zeta = 0.000035$ was calculated using Equation 4.3. (b) A data set with outliers and missing data, has $\zeta = 0.01$ calculated using Equation 4.4 39

4.7 A region of the final model for the fandisk is shown. (a) An abrupt change in β, the external force weight, does not give a very smooth model. (b) A smooth output is the result of a gradual increase in β. (c) The triangle mesh of the original model. 42

4.8 There are two stopping criteria for stops to update β.(a) The change in distance to the point set. (b) The change in position of the meshe's vertices. The vertical lines indicate a change in the value of β. 45

4.9 There are two stopping criteria for the final model. (a) The average distance of the mesh to the point set. (b) The values of β. If the mesh is still far from the point set but β reach its bounding value, then the deforrmation cannot go any further. The vertical lines indicate a change in the value of β. In the case presented here, the deformation stopped when the distance became zero, the value of β at the stopping point was still smaller than 0.5. 46

4.10 Sign of the curvature on a simplex mesh. (a) The curvature will be positive if the vertex is above the plane defined by its three neighbors. (b) The curvature will be negative if the vertex is below the plane. Note that "above" and "below" are relative to the global coordinate system. . 48

4.11 Outliers create very small areas of vertices that are close to the data set. These areas are shown in red. 49

4.12 Incomplete data can create a region of vertices that are far from the point set. This region cannot be a boundary between objects since it is totally surrounded by vertices that are close to the data set. 50

4.13 A boundary between two separated objects can create a region of vertices that are far from the point set. 50

4.14 The vertex P_i does not get closer to its closest point due two opposite internal and external forces. 51

5.1 Original 3D models of single objects. Four different objects were used to test the reconstruction of single objects: (a)t-pipe, (b) valve, (c) fandisk, (d) oil pump. 55

5.2 Clouds of points obtained from the original 3D models: (a) "t-pipe" 47286 points, (b) "valve" 64321 points, (c) "fandisk" 6475 points, (d) "oil pump" 22741 points. The cloud of points for the "t-pipe" and the "valve" were obtained using a range scanner simulator, the ones for the "fandisk" and the "oil pump" are the vertices of the model's triangle meshes. 56

5.3 Reconstruction of a t-pipe model. (a)-(i) The simplex is shown at different iteration of the deformation process. (j)-(l) The simplex mesh is shown at different iterations of the refinement process. (m)-(n) The simplex mesh for the refinement outputs. Note the vertex redistribution achieved by the refinement algorithm. In this figure and all the following ones related with deformation the blue veritces are those far from the data set whereas the red ones are the ones already on the data set. 58

5.4 Valve reconstruction. (a)-(c) The deformation process at different iterations. (d) Output of the refinemet process. (e) Simplex mesh for (c). (f) Simplex mesh for (d). Note the vertex redistribution achieved by the refinement algorithm. 59

5.5 Oil pump reconstruction. (a)-(c) The deformation process at different iterations. (d) Output of the refinemet process. (e) Simplex mesh for (c). (f) Simplex mesh for (d). Note the vertex redistribution achieved by the refinement algorithm. 60

5.6 Output of the deformation process for the fandisk with different number of vertices in the simplex mesh. (a)512 vertices, (b)1568 vertices, (c)2048 vertices, (d)3200 vertices, (e)3528 vertices, (f)4608 vertices 62

5.7 Output of the refinement process for the fandisk with different number of vertices in the simplex mesh. (a)512 vertices, (b)1568 vertices, (c)2048 vertices, (d)3200 vertices, (e)3528 vertices, (f)4608 vertices 63

5.8 Average error of deformation and refinement outputs for the "fandisk" using different number of vertices. 64

5.9 "Fandisk" point cloud with different levels of noise added. Recalling Equation (5.1) (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$. . . . 64

5.10 Deformation output for the fandisk with different levels of noise added. Recalling Equation 5.1 (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$. The simplex mesh has 3528 vertices and 1766 faces. 65

5.11 Refinement output for the fandisk with different levels of noise added. Recalling Equation (5.1) (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$. The simplex mesh has 3528 vertices and 1766 faces. 66

5.12 Average error of deformation and refinement outputs for the "fandisk" adding different levels of noise. 66

5.13 Model of the fandisk using noisy points as vertices of the mesh. Different levels of noise are added to the original vertices of the model. Recalling Equation (5.1) (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$. Compare to Figure 5.10. 67

5.14 Clouds of points that represent synthetic and real scenes used to test the reconstruction-segmentation algorithm. (a)Data set that represents a box on the floor, the cloud was created using a simulator range scanner. (b)Data set created using a Coleman laser range scanner. The data given by this scanner is almost free of noise, however, outliers can occur.(c) Close up on the Coleman data to a region with outliers. (d) Data set created using a line scan camera. (e) Data set created using a Percertron laser range scanner. The cloud is composed of data from teo different points of view registered with a manual registration technique. 69

5.15 Outputs of the deformation and segmentation processes applied on the data set that represents a box. (a)Deformation output. There are some NON-OBJECT vertices(in blue) but not a clear boundary between the box and the floor.(b)Segmentation output. The segmentations creates a clear boudary between the box and the floor. 70

5.16 Box: Original data set partitioned into (a) background and (b) object. . 70

5.17 Model of the segmented box. 71

5.18 Output of the segmentation algorithm for the data set of the box with different levels of noise added. Recalling Equation 5.1 (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$. 72

5.19 Cloud of points from a box on the floor with different levels of noise added. Recalling Equation 5.1 (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$. 73

5.20 Output of the deformation and segmentation for the Coleman scanner data. (a)Simplex mesh after deformation. (b)Close up on the deformed simplex mesh in the area of outliers.(c)Simplex mesh after segmentation. (d)Close up on the segmented simplex mesh in the area of outliers. The outliers make the segmentation create an additional region (in green) in the area of the wall. 74

5.21 Partition of Coleman cloud of points. (b) The cloud that belongs to the bricks has the outliers data points. 75

5.22 Line scan camera reconstruction and segmentation 76

5.23 Partition of line scan data 78

5.24 Perceptron reconstruction and segmentation 79

5.25 Partition of perceptron data 80

Chapter 1

Introduction

1.1 Problem Statement

The process of inferring a digital description from a physical object has been a central problem in recent computer vision research [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]. Three-dimensional (3D) reconstruction, surface reconstruction, and 3D scanning are a few terms that have been widely used to describe this process. The interest of the research community in this topic is due to the number of applications in the modern world, such as the examples below.

- Task space scene analysis. In this application, a 3D model of a robot's operating environment is constructed. This is crucial in teleoperation tasks such as the decontamination and decommissioning (D&D) of nuclear weapon facilities [20].

- Virtual reality. 3D realistic models of different objects are required to simulate

the physical world in virtual environments. 3D reconstruction is used to provide such models [21].

- Industrial design. The design industry nowadays relies heavily on CAD models. Oftentimes CAD designers cannot accurately create models of very complex shapes. 3D reconstruction can assist the designers in such tasks [22, 23].

- Reverse engineering. Similar to the industrial design application, an accurate model of a physical part can be obtained, facilitating its reproduction or replication [22, 23, 24].

- Repair, improvement, and analysis. A 3D model can be used in a simulation for analysis. Based on the simulation results, the original object can be repaired or improved[22, 23].

- 3D fax. 3D printers take as input a digital 3D model which can be transmitted from a far away location.

The input to a 3D reconstruction system is primarily shape information. Such shape information can be obtained in several ways. Some examples include:

- Touch probe. Touch probes have been widely used in the manufacturing industries. As the name indicates, these probes touch an object and record the position in space of the contact point.

- Laser range scanner. The most common types of laser range scanners are time-of-flight and triangulation systems. Time-of-flight scanners measure the range by

measuring the time between laser emission and return after reflecting off an object. Triangulation scanners use a laser to illuminate a point or a set of points so that they are easily identifiable in an image taken with a traditional camera. Once a point is identified in the image, its position in space is found at the crossing of the lines of sight from the point's location and the laser.

- Ultrasound scanners. These scanners measure shape information using the time-of-flight principle with ultrasound waves. Due to the nature of the ultrasound waves, these type of scanners are not very accurate.

- Stereo. In stereo [25], a triangulation is performed using a pair of images. Finding corresponding points in the two images, however, is quite difficult.

These input systems provide, directly or indirectly, a 3D cloud of points. A cloud of points, however, is not ideal for rendering, as it gives only a rough approximation of the shape. A polygon mesh is usually created to make a 3D model that accurately describes the geometry and topology of an object, while also providing for realistic rendering. Triangular meshes are perhaps the simplest and most widely used. When a triangle mesh is rendered, shading, illumination models, and textures can be applied to make the 3D model more realistic.

The creation of a mesh from a cloud of points is one of the most important steps in 3D reconstruction. This task is also a non-trivial problem, excepting the case where only a single point-of-view is considered. Presently, there are two main strategies for mesh creation. One is to create a single mesh from each point of view and then register

[1] and stitch them together. This is known as registration–integration strategy [7, 8]. Another approach is to register the different point clouds to create a single cloud, and then create a mesh by surface interpolation or approximation [2, 5, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18].

When considering 3D reconstruction, we consider two different scenarios: single object reconstruction and scene reconstruction. In single object reconstruction, the input point cloud represents data from only one object. Scene reconstruction deals with a more realistic situation where there may be several objects and background information. By background information, we mean the additional information captured when obtaining a range scan of an object, such as walls or floors where the object lies. The reconstruction of a scene can be carried out by creating a mesh that represents the whole scene. It is often desirable, however, to perform a reconstruction that creates more than one mesh. Ideally, we would like to create a mesh for each object –or perhaps each object part– present in the scene so that the geometry and topology of the scene are represented more accurately. Intuitively, there are two ways to achieve this goal. The first is to apply segmentation as a preprocessing step and then perform single object reconstruction. The second approach is to create a mesh and then segment it into different parts. The first method is easily applied if there exists a priori structure information [26]. In the case of an unstructured cloud of points, however, segmentaion

[1] When a range image is taken, the coordinates of the points will be in reference to the position of the camera If another image is taken from a different point of view, it will be in a different coordinate system The process of finding a transformation between these two different coordinate systems is known as *registration*

by preprocessing is very difficult to apply. The second method gets structure information of the cloud of points by performing reconstruction, making the segmentation easier.

The focus of this research is complex, real-world scene reconstruction incorporating object or part segmentation. Note that we do not explicitly differentiate between objects and object parts, which is a higher level task. This research is performed in the broader context (beyond the scope of this thesis) of task space analysis, where the final goal is a system to automatically construct a 3D model of a robot's workspace for the purposes of D&D, where the workspaces involved are often very complex. In this thesis, we describe a system for 3D reconstruction of real, indoor scenes using deformable simplex meshes [2, 3, 4]. The input to the system is an unstructured cloud of points which can represent, partially or fully, a single object or a complete scene. The output is a set of triangle meshes, one for each object or part present in the scene.

1.2 Strategy.

As we discuss in the next chapter, there are several methods to obtain a 3D reconstruction from a cloud of points. We employ *deformable simplex meshes* [2, 3, 4], which are non-parametric deformable models. We illustrate the overall strategy of our approach in Figure 1.1.

The first step is data acquisition and is not an issue in our research. We begin with an unstructured cloud of points from any source (range scanner, stereo, simulated data, etc.). We enclose the data with a simplex mesh in the shape of a sphere. Next, we

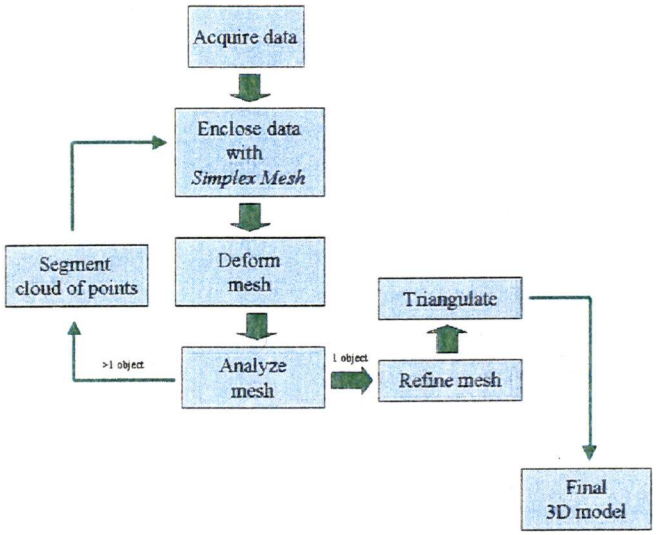


Figure 1.1: Overall strategy for 3D reconstruction.

deform the initial mesh following internal and external forces until it reaches stability. We then begin an analysis process to determine if the mesh should be split in order to separate objects from each other and/or the background. If the mesh is split, the original cloud of points is segmented using the distance to the closest mesh. We then repeat the initialization and deformation process on each cloud that results from the segmentation. Once a mesh does not split, a refinement process is applied to concentrate vertices around high curvature areas. Finally, each simplex mesh is transformed into a triangle mesh for the purposes of rendering.

Altough our simplex mesh method is similar in spirit to that of [2, 3, 4], we introduce several modifications, as described in Chapter 4, to make it better suited for complex scene modeling. Additionally, we also introduce a novel segmentation strategy to separate objects and object parts. To examine the behavior of our system, we experiment

with several types of point clouds: (1) Single object data sets with no background, (2) simple multiple object scenes where the objects are well separated, and (3) complex scenes with several objects that are close to each other. The input point clouds are provided from various sources, including simulated data and range scans of real world scenes. In the case of simulated data, different levels of artificial noise are added.

1.3 Thesis Outline.

In Chapter 2, we review relevant literature on 3D reconstruction and segmentation methods. We then review deformable simplex meshes, first developed by Delingette [2, 3, 4], in Chapter 3. In Chapter 4, we describe our enhacement of the deformable simplex meshes for the purposes of complex scene modeling and segmentation. We test our method on several data sets and report the results in Chapter 5. Finally, in Chapter 6, we present conclusions and suggestions for further research.

Chapter 2

Review of 3D Reconstruction and Segmentation

In this chapter, we review the state-of-the-art approaches to 3D reconstruction and segmentation. In section 2.1, we discuss algorithms for 3D object reconstruction and in section 2.2 we present several segmentation algorithms primarily those applied to 3D meshes.

2.1 3D Reconstruction

3D reconstruction is one of the central problems in computer vision research. Numerous researchers throughout the world are investigating this problem, resulting in many different approaches. These many approaches can be grouped into two categories: interpolation and approximation.

Interpolation methods create polygon meshes by using each point of the original data set as a vertex of the mesh. The main problem to be solved in this approach is to connect the vertices appropriately so that the output mesh represents, in some optimal sense, the geometry and topology of the original physical object. In contrast with interpolation, the approximation methods construct a mesh so that the data points lie close to the surface, but are not necessarily vertices. We describe different algorithms for these approaches in more detail in the following sections.

2.1.1 Interpolation Techniques

Here we present several interpolation type algorithms for 3D reconstruction. As noted previously, these techniques all use the original point cloud as vertices. Because of this, all interpolation approaches suffer similar drawbacks, including susceptibility to noise and failure to account for holes caused by incomplete data. These methods as described below, include integration, computational geometry, and surfaces from volumes.

Integration

Integration algorithms [7, 8] use the underlying structure of a range image to create a mesh from a single point of view. This structure, knowledge of the neighbors for each point, comes from the fact that the range image is a sampling over a regular grid. A triangulation process is repeated to create a mesh for each view point. A registration algorithm is applied to reference all the meshes to the same coordinate system. The overlapping regions between the meshes are then removed and the meshes are connected

to form a single mesh. This connection process is referred to as zippering or stitching.

Computational Geometry

Computational geometry algorithms generally have two steps. In the first step, every single point from the point cloud is connected to form a set of tetrahedra. In the second step, unnecessary faces are removed from the tetrahedra. The process of face removal is the challenging problem of this approach, and there are several proposed algorithms. O'Rourke [11] uses a criterion based on minimal area of the new triangulation, while Veltkamp [12] uses the maximal interior angle of the current triangulation. The main limitation of these methods is that they are applicable only for a fixed topology, although there are other algorithms that overcome this limitation. Edelsbrunner's and Mücke's α-*shapes* [13], for example, eliminate not only faces but also tetrahedra and edges whose corresponding minimum surrounding sphere does not fit inside a sphere of radius α. Similar results are obtained by Veltkamp's γ-*indicator*[15] and the α-*solids* of Bajaj et. al. [14].

Surfaces from Volumes

Another method for 3D reconstruction subdivides the space into volume elements (voxels). The voxels are filled using information from a sensor (e.g., a range image) so that a volume is created and from this volume, an isosurface is extracted. The filling of the voxels can be done by different methods. Wong [27] and Gray [28], for example, project the points from different views into the voxel representation at the origin of a

given coordinate system. There also exist several algorithms for the extraction of the isosurface. Perhaps the most widely known is Lorensen's et. al. *marching cubes* [9] although there are other similar algorithms such as Algorri's et. al. [16] and Wyvill's [10]

We note that these techniques do not use the original points as vertices for the constructed mesh. We consider them as interpolation methods since a vertex is created only if a voxel is occupied by a data point. The vertex created will not be in exactly the same position as the original point, although it will be very close since it will lie in the same voxel.

2.1.2 Approximation Techniques.

Approximation techniques create a discrete surface that is closest, in some optimal sense, to the point set. Since the vertices of such surfaces are not generally the data points, approximation techniques perform better than the interpolation approaches in the presence of noise, outliers, and missing data or holes.

Parametric models

Parametric models reconstruct surfaces as embedding functions. They approximate the data points by changing the parameters until a minimum in some distance meassure is attained. In [29], various algorithms are mentioned, including 3D curves, Beizer curves, B-splines, nonuniform rational B-splines (NURBS), and β-splines. In addition to curves, these methods can also model surface patches. For complete surface modeling

some examples include *superquadrics* [30] and *generalized cylinders* [31].

There are two main drawbacks to using parametric models. One is a lack of generality; the number of shapes that a parametric model can take is limited by its parameterization. Some algorithms overcome this limitation by modeling with patches. This approach, however, presents the problem of continuity across patch boundaries. Another potential drawback is that some optimization must be performed, which is often computationally expensive.

We note that parametric models can be slightly modified and used as deformable models to improve their capabilities.

Deformable Models

Deformable models begin with an initial 3D model that is deformed to fit the data. This idea was first introduced by Kass et. al. in [32] with their *Active Contour Models* or *Snakes*, which were applied to fit 2D contours. The innovative idea in these models is that they are physics-based models, as opposed to the previous purely geometric models. According to [33], there are three basic features that make deformable models suitable for the reconstruction of rigid and non-rigid objects:

1. They respond to applied (simulated) forces analogous to the way objects respond to forces in the physical world.

2. The data influence the deformable models through external forces. Likewise, shape constraints act as internal forces. The influence of both forces evolves the model

into structures of interest.

3. Deformable models are fundamentally dynamic. This fact makes them suitable to model-changing objects.

As noted in [34], deformable models can be classified into two categories according to their representation: continuous and discrete. In a discrete representation,the geometry of the model is known only at the finite set of points defined by its vertices. Continuous models are, of course, discretized for computational needs, but it is possible to compute their geometry almost everywhere on the model. Each of these categories (continuous and discrete) can be further subdivided in two subcategories. Continuous models can be parametric or non-parametric, whereas discrete models are generally represented as discrete meshes or particle systems.

The parametric models described in the previous section can be made more powerful by combining them with the basic features of deformable models [35, 36]. Such deformable, parametric models are described by a function that depends on a parameter vector. Again the main setback is that the shapes they can take are limited the number of parameters.

Contrary to parametric models, non-parametric models do not have a limitation in their domain of possible shapes. Terzopoulos and Metaxas [18] improved superquadrics to develop deformable superquadrics. Local deformations were added to allow the model to easily change topologies (i.e., the number of genus, object handles, and holes), overcoming the problem of limited shapes that superquadrics present. This characteristic

makes deformable superquadrics very desirable for modeling objects of any topology.

The level sets approaches [37, 38, 17] are other models that are also topologically flexible. In these approches, a 3D model that corresponding to the zero level set of a 4D function is deformed. The primary advantages of the level set approaches are robustness to noise and elegant mathematical formulation.

Both superquadrics and level sets require global minimizations to reach equilibrium, making them computationally expensive. In applications where high accuracy is more important than speed, such as simulation, these are perhaps a good option.

Turning our attention now to the discrete models, one of the first to appear was the spring-mass model [39, 40]. In this model, mesh is created where every vertex simulates a mass and every link a spring. The defromation is based on Newton's second law of motion, where the internal forces are spring forces and can be linear or nonlinear. Unfortunately, the spring constants that govern the internal forces are not easy to tune. These models work best with non-rigid objects; a rigid object requires large spring constants that often lead to stability problems.

Simplex meshes are discrete deformable models developed by Dellingette [2, 3, 4]. A simplex mesh has constant vertex connectivity; each vertex has three, and only three, neighbors. Due to this, the geometry of a simplex mesh is relatively simple. Simplex meshes have several advantages over the previous methods.

- Their deformation is not computationally expensive. It is based on a local energy minimization.

- They can change topology. Topological operators are easily defined on simplex meshes.

- Connected component analysis is easily applicable due to constant vertex connectivity.

- Simplex meshes should be well suited to higher level tasks, such as object recognition, since the shape of the mesh is encoded using local geometric quantities.

Surveys on deformable models can be found in [29], [33], and [34]. In the following chapter, we present a more complete review of the simplex mesh, one such deformable model that suits our application.

2.2 Segmentation

Segmentation is a partition of a data set into subsets of elements that share a common property, i.e., a partition into homogeneous regions. In 2D images, for example, a common property that can be used as the basis for segmentation is the uniformity of light density. In the case of 3D reconstruction, segmentation can be beneficial in several ways. For example, segmentation can help improve the quality of the 3D representation by permitting mesh optimization to be performed over regions or parts. Additionally, segmentation is necessary for higher level vision tasks such as object recognition.

Region segmentation partitions a 3D data set or model into regions with common properties. Mangan and Whitaker [41] segment a 3D mesh into regions of consistent

curvature bounded by regions of high curvature for the purpose of mesh reduction and texture mapping. Alrashdan et. al. [24] segment range images using curvature as the common property for regions by approximating surfaces with quadric polynomials. McIvor et. al. [42] segment range images by finding regions that correspond to surface patches from simple geometric shapes (i.e. planes, spheres, cylinders) and then use them as features for object recognition. Similarly, in [43, 44, 45] low-order polynomials are fit to extract surface patches.

Object part segmentation, combining regions to represent object parts, is a higher level task than region segmentation. This kind of segmentation is preferable, for example, in object recognition. If we know the decomposition of an object into its parts, we can recognize it even if it is partially occluded. One approach to object part segmentation is to fit basic shapes from a database of known objects. Different basic shapes have been used for this approach: polyhedra [46], generalized cones and cylinders [47, 48, 49], geons [50], and superquadrics [51, 52]. The main drawback of these methods is that they are limited in scope, since not all the objects contain the basic shapes they use. Hoffman and Richards [53] proposed a method that looks for boundaries between object parts at negative minima of curvature. These boundaries exist regardless of the shape of the object part so the approach is not limited in scope. Using the same concept of negative minima in curvature, Wu and Levine [54] segment 3D meshes by simulating distribution of electric charge; in a perfect conductor, electric charge tends to accumulate in sharp convexities and disappear on sharp concavities(negative minima of curvature).

In addition to finding object parts or regions, segmentation algorithms deal with the issue of how to break the data set or 3D model. In the case of a data set, clusters of points are formed. For 3D objects, an additional algorithm for mesh splitting should be used. Yoshino [55] and Bro-Nielsen [40] proposed splitting algorithms for active nets applied on 2D images. In their algorithms they model the mesh links as springs. When deforming to fit regions corresponding to different objects, a tension acts on the links in the boundaries between objects. When this tension exceeds a threshold the links are broken.

Chapter 3

Deformable Simplex Meshes

In this chapter, we review the fundamental properties of deformable simplex meshes. Simplex meshes are deformable models developed by Dellingette [2, 3, 4]. We have selected them for our system due to their flexibility, computationally inexpensive deformation, and ease of definition for geometric entities (which are useful in higher level vision tasks). The presentation here is modeled closely on that of [3].

3.1 Definition and Characteristics

In this document we use the term simplex mesh to refer to a 2-simplex mesh of R^3. A k-simplex mesh contains a k-simplex at each vertex. In Figure 3.1, we show k-simplexes for $k \in [0, 3]$.

We use the simplex mesh as a discrete mesh to represent a surface in R^3. It has constant connectivity; each vertex has exactly three neighbors. It can be easily visual-

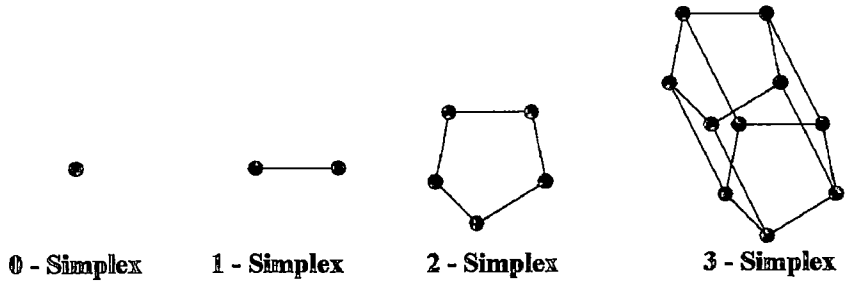


Figure 3.1. Examples of k-simplexes.

ized as a dual of a regular triangulation as shown in Figure 3 2 Although topologically equivalent, simplex meshes and triangulations are geometrically different. The simplex mesh has several properties that makes it suitable for general 3D reconstruction

- Just as a triangulation, the simplex mesh can represent surfaces of virtually any topology. There are topological operators defined on simplex meshes that allow changes in the number of genus (number of handles) and holes.
- Deformation is computationally inexpensive. The displacement of vertices is based on the computation of internal and external forces. The internal force has a

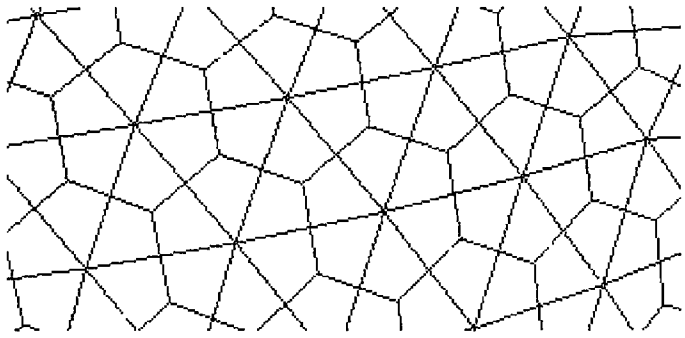


Figure 3.2 Simplex mesh and triangulation are dual of each other

nonlinear expression, but is derived from a simple finite difference method. The external force is mainly a search that depends on the number of vertices in the simplex mesh, which is always chosen smaller than the number of points in the original data set.

- It is possible to define at each vertex discrete geometric quantities such as the simplex angle and the metric parameters that define the shape of the mesh.
- Quality can be improved by adding, removing, or concentrating vertices in different areas. For example, vertices can be concentrated in high curvature areas whereas the number of vertices can be set to a minimum in areas of low curvture.

3.2 Geometry

The position of each vertex in a simplex mesh, and therefore the mesh geometry, can be determined from only its simplex angle and metric parameters, both of which we will define shortly. To ilustrate the concept of the simplex angle, we show the simplex angle in a planar, 1-simplex mesh (a discrete planar curve) in Figure 3.3. The simplex angle is the complement of the angle between two adjacent line segments. This angle, together with the length of the segments, defines the discrete curve.

The simplex angle for the 2-simplex mesh is defined using local geometric entities. The first is the circumscribed sphere of the tetrahedron formed by a vertex and its neighbors. The circle circumscribing the three neighbors and the plane defined by the three neighbors, are also used. These are all illustrated in Figure 3.4.

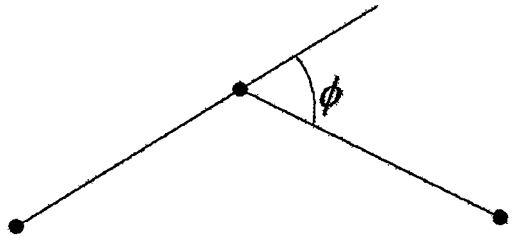


Figure 3 3: The simplex angle in a planar curve.

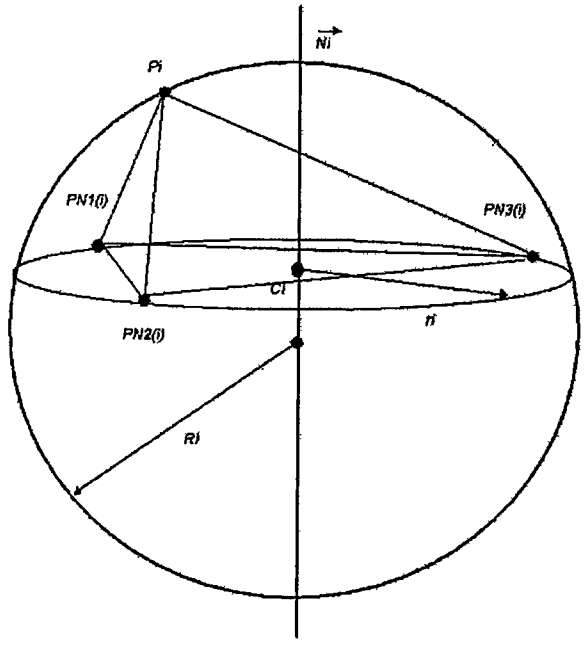


Figure 3.4: The simplex angle depends on the circumscribed sphere of the vertex and its neighbors as well as the plane and circumscribed circle defined by the three neighbors. P_i is the i-th vertex; PN1(i), PN2(i) and PN3(i) are the neighbors of the i-th vertex; N_i is the normal vector at vertex i; r_i and C_i are the radius and the center respectively of the circumscribed circle; and R_i and O_i are the radius and the center respectively of the circumbscribed sphere.

The normal of the plane formed by the neighbors of the *ith* vertex is calculated as:

$$\overline{N_i} = \frac{(P_{N_1(i)} \times P_{N_2(i)}) + (P_{N_2(i)} \times P_{N_3(i)}) + (P_{N_3(i)} \times P_{N_1(i)})}{\|(P_{N_1(i)} \times P_{N_2(i)}) + (P_{N_2(i)} \times P_{N_3(i)}) + (P_{N_3(i)} \times P_{N_1(i)})\|} \tag{3.1}$$

where P_{N_l} is the l-th neighbor. The simplex angle is defined by:

$$\begin{aligned} \varphi_i &\in [-\pi, \pi], \\ \sin(\varphi_i) &= \frac{r_i}{R_i} * sign(\overline{P_i P_{N_1(i)}} \cdot \overline{N_i}), \\ \cos(\varphi_i) &= \frac{\|\overline{O_i C_i}\|}{R_i} * sign(\overline{O_i C_i} \cdot \overline{N_i}), \end{aligned} \tag{3.2}$$

where r_i and C_i are the radius and the center of the circumscribed circle of the three neighbors, and R_i and O_i are the radius and the center of the circumbscribed sphere of the tetrahedron formed by a vertex and its neighbors.

The simplex angle can be seen as a measure of the height of the vertex with respect to the plane of its neighbors; if its value is zero, then the vertex is coplanar with its neighbors (and the sphere will have infinite radius). This is can be seen by examining a cross section of the circumscribed sphere as shown in Figure 3.5. In this cross section, the simplex angle can be interpreted as a planar angle, making explicit the relationship between the simplex angle and the height of P_i with respect to its three neighbors.

Recalling Equation (3.2), we note that large values of R_i, which occur when the vertex is on the plane defined by its neighbors, can make the computation of φ_i troublesome. To overcome this problem, we perform an inversion of center. First, we find

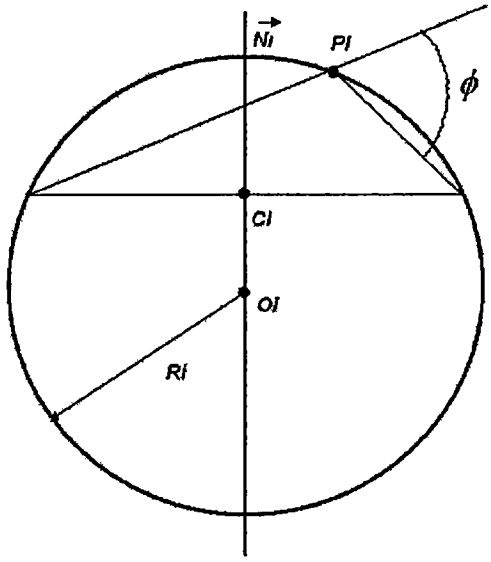


Figure 3.5: Cross section of figure 3.4 through a plane defined by O_i, C_i and P_i. The simplex angle can be seen as a planar angle.

the inverse of each neighbor $Inv(P_{N_j}(i))$, such that:

$$\overline{P_i P_{N_j}(i)} \cdot \overline{P_i Inv(P_{N_j}(i))} = 1. \tag{3.3}$$

We then find the perpendicular distance D_i from P_i to the plane with normal $\overline{u_i}$ formed by the inverse of the neighbors, as illustrated in Figure 3.6.

$$D_i = \frac{1}{2R_i} \qquad O_i = P_i + \frac{\overline{u_i}}{2D_i}. \tag{3.4}$$

Now Equation (3.2) can be written as:

$$\sin(\varphi_i) \;=\; 2r_i D_i \text{sign}(\overline{\mathrm{P_i P_{N_1(i)}}} \cdot \overline{\mathrm{N_i}}), \text{ and}$$

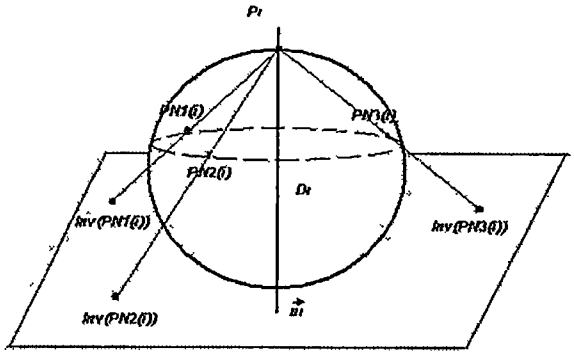


Figure 3.6: An inversion of center changes the definition of simplex angle to make it more suitable for computation.

$$\cos(\varphi_i) \quad = \quad \|2D_i\overrightarrow{P_iC_i} + \overrightarrow{u_i}\|\text{sign}(\overrightarrow{\text{O}_\text{i}\text{C}_\text{i}} \cdot \overrightarrow{\text{N}_\text{i}}). \tag{3.5}$$

We now turn our attention to the metric parameters. The metric parameters describe how a vertex is located with respect to its neighbors. If F_i is the projection of a vertex on the triangle formed by its neighbors, we have:

$$F_i = \epsilon_{1i} \cdot P_{N_1(i)} + \epsilon_{2i} \cdot P_{N_2(i)} + \epsilon_{3i} \cdot P_{N_3(i)}, \tag{3.6}$$

where $\epsilon_{1i} + \epsilon_{2i} + \epsilon_{3i} = 1$. Using the simplex angle and the metric parameters, we can find the position of each vertex as:

$$P_i = \underbrace{\epsilon_{1i} \cdot P_{N_1(i)} + \epsilon_{2i} \cdot P_{N_2(i)} + \epsilon_{3i} \cdot P_{N_3(i)}}_{F_i} + L(r_i, d_i, \varphi_i)\overrightarrow{N_i} \tag{3.7}$$

where $L(r_i, d_i, \varphi_i)$ is given by:

$$
\begin{aligned}
L(r_i, d_i, \varphi_i) &= \frac{(r_i^2 - d_i^2)\tan(\varphi_i)}{\epsilon\sqrt{r_i^2 + (r_i^2 - d_i^2)\tan^2(\varphi_i)} + r_i} \\
\text{with} \quad \epsilon &= \text{sign}(\frac{\pi}{2} - |\varphi_1|),
\end{aligned}
\tag{3.8}
$$

where d_i is the distance from F_i to C_i. as before.

3.3 Deformation

The evolution of a simplex mesh to fit a data set is based on the Newtonian law of motion:

$$
m\frac{d^2P_i}{dt^2} = -\gamma\frac{dP_i}{dt} + \overline{F_{int}} + \overline{F_{ext}}. \tag{3.9}
$$

If we integrate and discretize (3.9) we get:

$$
P_i^{t+1} = (1-\gamma)(P_i^t - P_i^{t-1}) + \overline{F_{int}} + \overline{F_{ext}}. \tag{3.10}
$$

This equation is the evolution applied to calculate the position of each vertex at every iteration. In the following sections, we elaborate on the internal and external forces.

3.3.1 Internal Force

The internal force permits shape constraints to be imposed on the mesh, assuring a regular shape. Contrary to other physics based deformable models, the internal force is calculated using a local, as opposed to a global, energy function. This local function is the energy of the tetahedron formed by each vertex and its three neighboors, and is defined by:

$$S_i = \frac{\alpha_i}{2} P_i P_i^{*2}, \tag{3.11}$$

where α_i is a weight factor. The internal force should minimize this energy, resulting in

$$\overline{F_{int}} = \alpha_i P_i P_i^{*2}. \tag{3.12}$$

Using Equation (3.7) to express P_i^*, we get

$$\overline{F_{int}} = \alpha_i[(\epsilon_{1i}^* \cdot P_{N_1(i)} + \epsilon_{2i}^* \cdot P_{N_2(i)} + \epsilon_{3i}^* \cdot P_{N_3(i)} + L(r_i, d_i, \varphi^*)\overline{N_i})] \tag{3.13}$$

where the superscript "$*$" indicates the desired result. By defining φ^*, we impose the different shape constraints.

There are four different shape constraints:

- Normal discontinuity. Set $\varphi^* = \varphi_i$ and the surface can freely deform.
- Surface orientation continuity. Set $\varphi^* = 0$ and the vertex tends to go to the plane

formed by its neighbors.

- Mean curvature continuity. Calculate φ^* as the mean of the neighborhood of the vertex. The size of the neighborhood is directly related to the notion of rigidity. The larger the neighborhood, the more rigid the mesh will be.

- Shape constraint. Set $\varphi^* = \varphi^0$, where φ^0 indicates the shape that the mesh should take.

3.3.2 External Force

The role of the external force is to drive the mesh vertices so they are close to the data. Without the internal force, the external force would force the vertices as close as possible to the data, but it would make the mesh tangle itself. To calculate this force, we look for the closest point in the data set to the vertex. The force is directly proportional to the vector that connects these two points. To avoid disorganized structure, the external force is projected in the direction of the normal N_i. The expression for the external force is then

$$\overline{F_{ext}} = \beta G\Big(\frac{\|\overline{P_i M_{CL(i)}}\|}{D_{ref}}\Big)(\overline{P_i M_{CL(i)}} \cdot \overline{N_i})\overline{N_i}, \tag{3.14}$$

where β is a weight factor and $M_{CL(i)}$ is the closest point to the vertex. The function $G(\cdot)$ determines the sphere of influence and it is illustrated in Figure 3.7. If the closest data point to the vertex is beyond D_{ref}, it will not contribute to the external force.

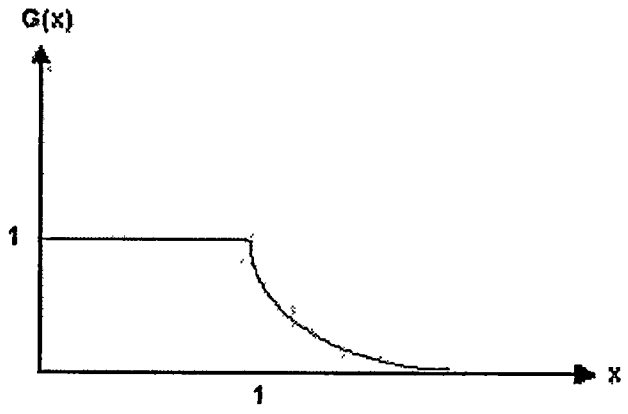


Figure 3.7: When calculating the external force the function $G(\cdot)$ limits the influence of the closest point if it is too far away. This function appeared originally in [2].

3.4 Mesh Refinement

After the deformation, the distribution of vertices on the simplex mesh is more or less homogeneous. It is proposed in [4], however, that it would be more desirable to have a high concentration of vertices in high curvature areas, since these areas have a high level of detail. Conversely a low concentration of vertices in low curvature areas should not significally impact quality. To accomplish this, we allow the metric parameters to affect the deformation. As indicated earlier, the metric parameters determine the position of a vertex's projection onto the triangle formed by its neighbors. If t denotes the iteration in the deformation, we recalculate the value of the metric parameters each $t + p$ iterations by:

$$\epsilon^{t+p} = \epsilon_i + \frac{1}{2}(\epsilon_i^* - \epsilon^{t+p}), \tag{3.15}$$

$$\epsilon_i^* = \frac{1}{3} + \delta|H|_i, \tag{3.16}$$

$$\delta|\mathbf{H}|_\mathbf{i} = \begin{pmatrix} \frac{|H_{N_1 i}| - |\overline{H_i}|}{|\overline{H_i}|} \\ \frac{|H_{N_2 i}| - |\overline{H_i}|}{|\overline{H_i}|} \\ \frac{|H_{N_3 i}| - |\overline{H_i}|}{|\overline{H_i}|} \end{pmatrix}, \tag{3.17}$$

$$|\overline{H_i}| = \frac{|H_{N_1 i}| + |H_{N_1 i}| + |H_{N_1 i}|}{3}. \tag{3.18}$$

Where H_i is the mean curvature.

As we will see in Chapter 5, vertex redistribution generally serves to improve subjective quality, but usually provides only marginal objective improvements.

Chapter 4

Complex Scene Modeling and Segmentation

In their basic form deformable simplex meshes are not well suited for the modeling of complex scenes. As indicated in [2], simplex meshes require a priori knowledge about the topology of the object for good initialization. It is also required that the cloud of points representing an object be separated from the background or other objects.

In this chapter, we present our implementation of simplex meshes. We modify and extend the algorithm of Chapter 3 so that it can be used for the reconstruction of complex scenes. Additionally, we develop and implement an algorithm to perform object segmentation based on the simplex mesh.

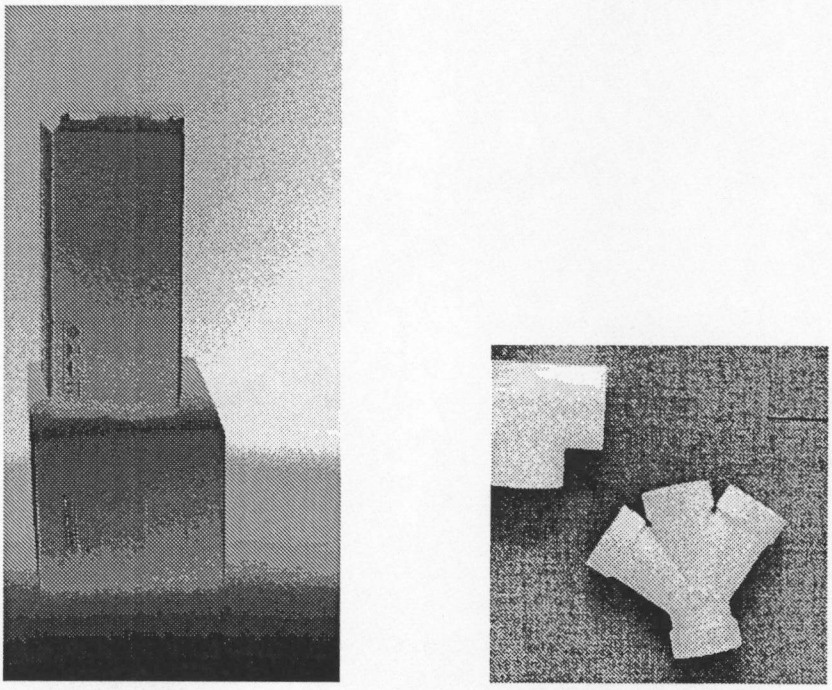
4.1 Deformation

4.1.1 Initialization

The first task in the deformation process is, of course, the initialization. Initialization is a very important step in most deformable models, and the final output generally depends strongly on how close the initialization is to the data set, in terms of distance, geometry, and topology. Since we use unstructured clouds of points as input, we have no a priori information that could help us initialize the simplex mesh close to the data.

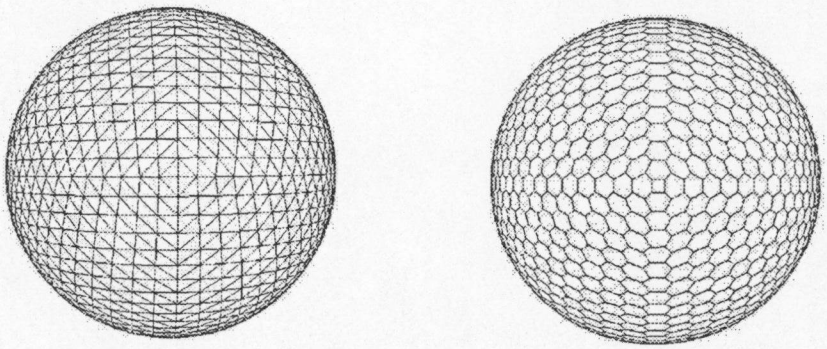
For the initial mesh, we construct a sphere that encloses the data completely. The idea behind this initial mesh is to emulate a balloon that contains all the objects inside. When deflating, the balloon will take the shape of the contained objects. This resulting mesh will be the initial shape estimation, which will subsequently be segmented and refined. We note that this initialization assures that the surfaces of interest are the outer ones. To examine the performance of our algorithm, we use scenes with multiple objects and background information, such as those in Figure 4.1. As we can see the visible surface is the outer one. If there is interest in modeling inner surfaces, then a volumetric approach [9] would be required for initialization.

It is desirable to have a regular mesh, where all the faces are the same polygon, to avoid artifacts during the deformation. We can get an almost regular mesh by triangulating a sphere as shown in Figure 4.2. When transformed into a simplex mesh, all the faces are hexagons except six, which are quadrilaterals. The number of triangles, thus vertices in the simplex mesh, can be approximated by:



(a) (b)

Figure 4.1: The input scenes to the system include information from one or multiple objects and also background information. (a)Scene of a couple of boxes including the floor and wall information. (b)Scene of multiple objects on a flat surface.



(a) (b)

Figure 4.2: Tessellation of a sphere (a)A sphere is triangulated. (b)The dual simplex mesh has all faces hexagons but six.

$$T = 8 \sum_{n=0}^{D_v - 1} (2n - 1) \tag{4.1}$$

where D_v is the number of divisions of the half circle that goes from pole to pole.

The number of vertices in the initial mesh is an important quantity as it relates to the quality of the final mesh. It should depend on the radius of the enclosing sphere; the larger the radius, the more vertices there should be. This is due to the fact that faces with small area are preferred as they are the more likely to be approximately planar. The number of vertices in the initial mesh also depends on the level of detail required. If the data set is planar, then only few vertices are required (even large faces will be planar). If the object to be modeled has many details then a large number of vertices should be used. Measuring the level of detail without any a priori information, however, is very difficult and requires the topology to be known. Since we do not have this information, we allow the number of vertices to be a user defined parameter (not automatic).

4.1.2 Shape Constraint

As we mention in Section 3.3, the internal force drives the mesh to regular shapes (i.e., shapes that are close to a manifold, which is a surface that does not intersect itself). To achieve a regular mesh, a shape constraint must be applied through the calculation of the "desired" simplex angle. The constraint is applied by calculating the internal force so it favors a desired simplex angle. In Section 3.3.1, four constraints were mentioned,

but two — normal discontinuity and fixed shape — are not applicable to our goals. The *normal discontinuity contraint* does not yield regular shapes, while the *fixed shape constraint* is not applicable since we have no a priori shape information. The remaining two constraints — mean curvature and surface orientation — are applicable. We now turn our attention to finding which of these two best suits our application.

In Figure 4.3, we see the evolution of a simplex mesh to fit the "bunny" data set using the *mean curvature constraint* taken from a bunny. The mean curvature constraint sets the desired simplex angle to the mean of the simplex angles of the neighborhood. The size of the neighborhood determines the rigidity of the mesh. As we generally initialize far from the dataset, a large rigidity is required; if it is not large enough, then the deformation outputs a mesh that is not smooth. High rigidity, however, requires much more calculation as the neighborhood for each vertex must first be found and then, at every iteration, the simplex angle for all the neighborhood vertices must be computed.

If, on the other hand, we use the *surface orientation constraint* the deformation is more uniform and there is no need for the calculation of neighborhoods or simplex angles, making it more suitable for our system. Figure 4.4 shows the surface orientation constraint applied to the "bunny" data set (note 4.3(d) and 4.4(d)).

Recalling Equation 3.13, we must also determine the weighting factor α_i to completely specify the internal force. Using the mean curvature constraint, Delingette [3, 2] suggests a value for α_i in the range $[0, 0.5]$, and leaves this value fixed during the entire deformation process. We found that this scheme performs poorly when using the surface

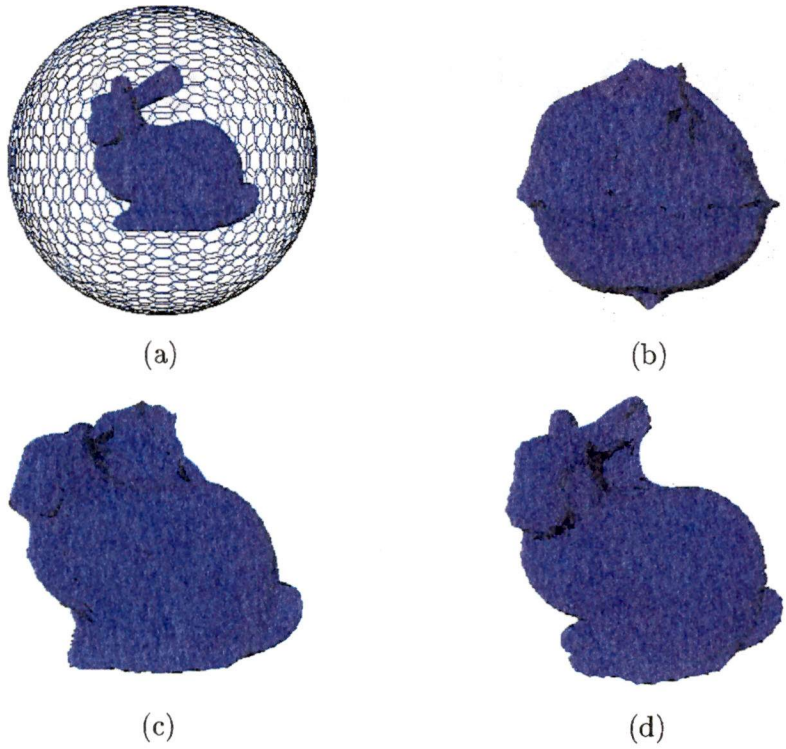


Figure 4.3: Simplex mesh deforming using mean curvature continuity constraint. There are 35000 points in the dataset and the simplex mesh has only 3000 vertices. (a) Initial mesh enclosing the data set. (b) Mesh after 513 iterations (c) Mesh after 730 iterations. (d) Final model obtained after 1214 iterations. Note the lack of smoothness in comparision with 4.4 (d).

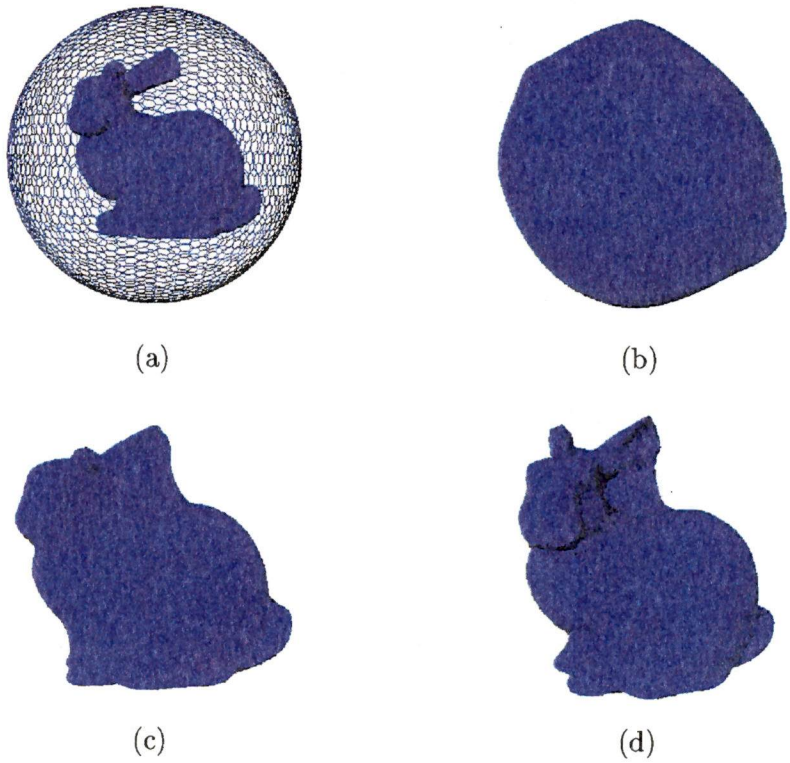
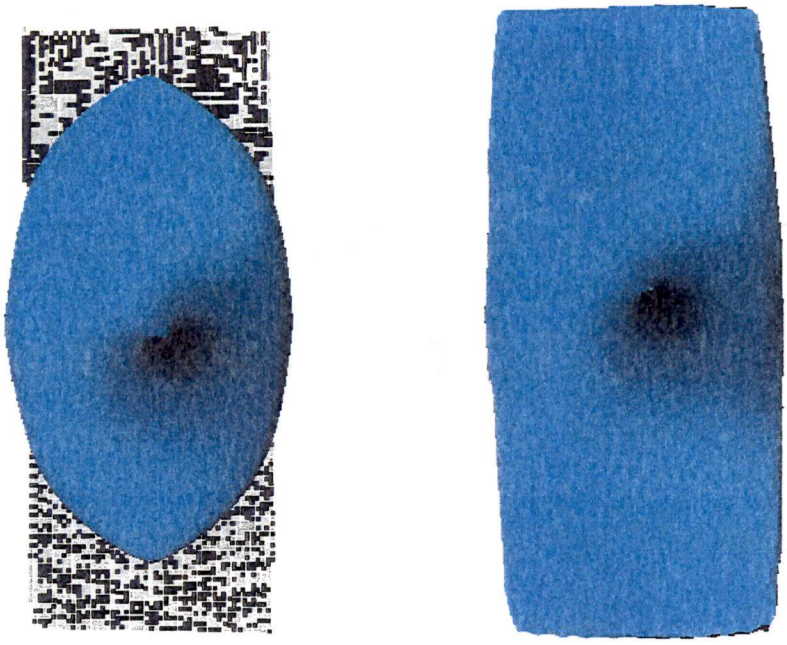


Figure 4.4: Simplex mesh deforming using surface orientation continuity constraint. There are 35000 points in the dataset and the simplex mesh has only 3000 vertices.(a) Initial mesh enclosing the data set. (b) Mesh after 513 iterations (c) Mesh after 730 iterations. (d) Final model obtained after 1214 iterations. Note the smoothness in comparision with 4.3 (d).



(a) (b)

Figure 4.5: A simplex mesh deforming with constant α_i loses information. To solve this problem a variable α_i should be used. The simplex mesh after 150 iterations with (a) constant ($= 0.5$) and (b) variable α_i.

orientation constraint. If we leave α_i large (≈ 0.5) during the whole process, we can lose some information due to the "shrinking" effect of the internal force. For example, as illustrated in Figure 4.5, a simplex mesh fitting a plane, can continue to shrink despite the closeness of the data points. To avoid this problem, a change in the weights should be performed as the vertices of the mesh get closer to the data.

To implement the change in α_i, we add a new parameter, ζ. The value of α_i is then given by

$$\alpha_i \approx \begin{cases} 0.5 & \text{if } d_{min} > \zeta, \\ 0.001 & \text{if } d_{min} < \zeta, \end{cases} \tag{4.2}$$

where d_{min} is the distance to the closest point. In other words, we neglect the internal force when a vertex is very close to the data. The parameter ζ is given by

$$\zeta = \frac{1}{N} \sum^{N} d_p, \tag{4.3}$$

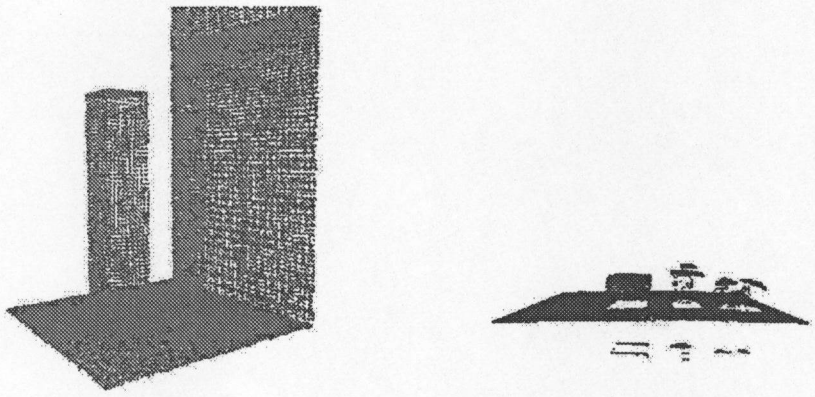
where N is the total number of points in the data set and d_p is the Euclidean distance of each point from the data set to its nearest neighbor. This value of ζ works well except in the presence of outliers. In this case, the value of ζ should be smaller than that given by Equation (4.3). Through experimentation, we have found that

$$\zeta = \mathrm{f}(\frac{1}{\mathrm{N}} \sum^{\mathrm{N}} \mathrm{d_p}, \vartheta) \tag{4.4}$$

works well, where ϑ is the variance of the average separation between nearest neighbors and the function $f(\cdot)$ was developed by an ad hoc method: the larger ϑ is, the smaller percentage of $\frac{1}{N} \sum^{N} d_p$, $f(\cdot)$ will be, that is

$$f(\frac{1}{N} \sum^{N} d_p, \vartheta) = \nu \cdot (\frac{1}{N} \sum^{N} d_p), \tag{4.5}$$

where ν is a percentage value given by a look up table. This value becomes smaller as ϑ increases.



(a) (b)

Figure 4.6: Examples of calculation of ζ. (a) In a clean data set, no missing data or outliers $\zeta = 0.000035$ was calculated using Equation 4.3. (b) A data set with outliers and missing data, has $\zeta = 0.01$ calculated using Equation 4.4

In Figure 4.6 (a) we see a data set for which ζ was calculated with Equation 4.3 (ζ=0.000035). In Figure 4.6 (b) the variance value was large, indicating the presence of outliers, so ζ's value was calculated using Equation 4.4 ($\zeta = 0.01$).

4.1.3 Data Constraint

The external force moves the vertices of the mesh close to the points of the data set and can be seen as a data constraint. In calculating the external force, given by Equation (3.14), we must select D_{ref} and β and choose a method to find the nearest data point.

The parameter D_{ref} serves to limit the influence of a vertex too far from the data set, and thereby improves the speed of the deformation. If a vertex is far from data, there is no need to calculate the external force. In [2, 4], D_{ref} is set to be 20% of the radius of the enclosing sphere at the beginning of the deformation, and then is reduced to 8%. We found out that this scheme does not work very well for our system. Instead

we set D_{ref} initially to the radius of the enclosing sphere, an then calculate the average of the distance from each vertex to the closest point in the data set as the mesh evolves. We then reduce D_{ref} in the same proportion.

Note that to calculate the external force on a vertex, we must find the closest point to it. The most straightforward way of finding the closest point is an exhaustive, brute force search of the point set. An exhaustive search will compute the distance between a vertex and every single point of the data set, and select the point with minimum distance. This method, while simple, is computationally expensive and inefficient. The computation time for one iteration is $O(N_1 N_2)$, where N_1 and N_2 represent the number of vertices in the simplex mesh and the number of points in the data set, respectively. To improve the speed, we employ the kD-tree algorithm [56]. The kD-tree extends the concept of a binary tree to multiple dimensions. The idea is to store the data points in a binary tree structure in a manner where, upon searching the tree, it is possible to eliminate entire subtrees of points that could not possibly contain the closest point sought. The performance of the kD-tree search is proportional to the total number of nodes of the tree visited during the search. A d-dimensional search in a set with N members can be performed with $O(dN^{1-1/d})$ operations. Implementing the kD-tree improves speed using an efficient search and also allows the search to easily be restricted to a sphere of radius D_{ref}.

We now turn our attention to setting the weight of the external force, β. In [4], a range of $[0, 0.5]$ is suggested, although only two values are used in that implementation.

Table 4.1: Values for the external force weight, β

0 001	0.005	0.01	0.05	0.1	0.15	0.2	0 25	0 3	0.35	0.4	0.45	0.5

In [4] a small value ($\beta \approx 0.1$) coupled with a high rigidity (Section 4.1.2) is used initially and then β is increased ($\beta \approx 0.5$) and the rigidity reduced. To get a smooth deformation and to keep the mesh as close as possible to a manifold, we implement a gradual increase in the value of β. We begin with a very small value and let the deformation proceed until it stalls (as determined by a stopping criteria described in the following section). After stalling, we increase β and repeat the process until; the deformation is completed. The values that β can take are listed in Table 4.1. The upper limit is set at 0.5; higher values do not yield regular shapes. We illustrate the differences of abruptly changing β compared to gradually changing β in Figure 4.7. In 4.7 (a) we can see that the final mesh is not very smooth, while in 4.7 (b) the mesh is much more smooth. This smooth mesh is visually closer to the original triangle mesh shown in 4.7 (c).

4.1.4 Stopping Criteria

We now turn our attention to the issue of finding appropiate criteria to determine when mesh deformation should be stopped, either to complete the modeling or to adjust β. The first logical sttoping criterion is the distance of the mesh to the point set; once the mesh is on the point set, deformation should stop. This criterion, however, is not very reliable, especially in the presence of outliers. If this is the case, a vertex that is trapped by an outlier can keep the neighboring vertices far from the point set. In this situation,

GILBERT
100% COTTON

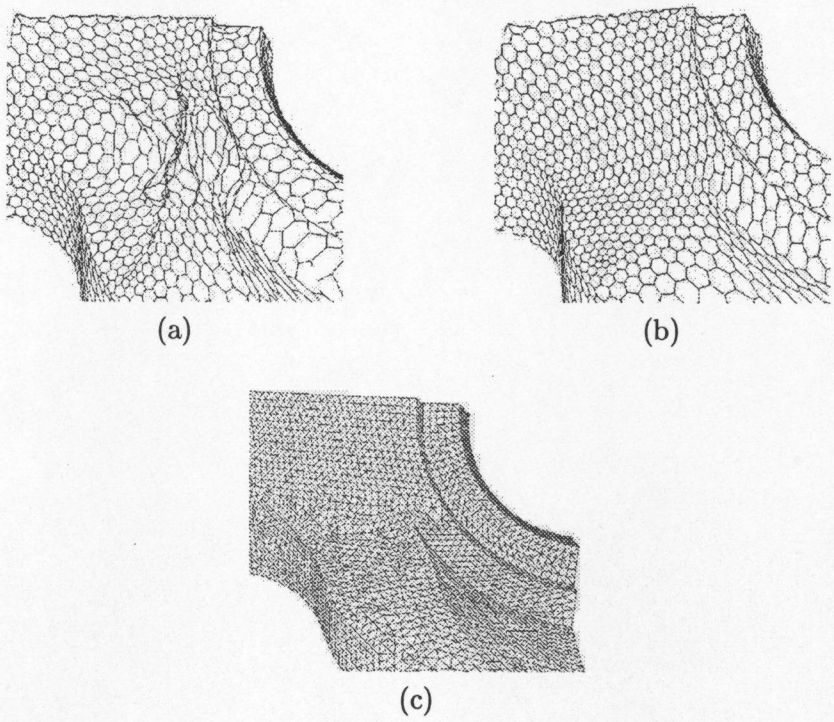


Figure 4.7: A region of the final model for the fandisk is shown. (a) An abrupt change in β, the external force weight, does not give a very smooth model. (b) A smooth output is the result of a gradual increase in β. (c) The triangle mesh of the original model.

the deformation will stall and the distance to the point set will always be large. Rather than using only the distance to the point set, we employ four different criteria. Two are used in the case when the deformation is stopped to change the value of β and two to stop the deformation when the modeling is complete. Note that we say modeling is complete when the distance to the data set is zero or the mesh cannot get any closer to the data.

The two criteria used to indicate that β should be changed are the change in the distance to the point set and the change of the mesh vertices' position.

$$\Delta_D(k) = \frac{D_{avr}(k-1) - D_{avr}(k)}{D_{avr}(k-1)}, \tag{4.6}$$

where k is the iteration number and D_{avr} is given by

$$D_{avr} = \frac{1}{n} \cdot \sum_{i=1}^{n} \|\overline{M_{CL_i} P_i}\|, \tag{4.7}$$

where P_i is the vertex position and M_{CL_i} is the position of the closest point. The change in vertex position is given by

$$\Delta_{pos} = \frac{1}{n} \cdot \sum_{i=1}^{n} \|\overline{P_i^{t-1} P_i^t}\|. \tag{4.8}$$

The criteria used to stop deformation when the modeling has finished are the average distance to the point set or β out of range. If there are no outliers, the average distance

of the mesh to the point set will be approximately zero when the modeling is finished. If there are, modeling will stop when β reaches its bounding value of 0.5.

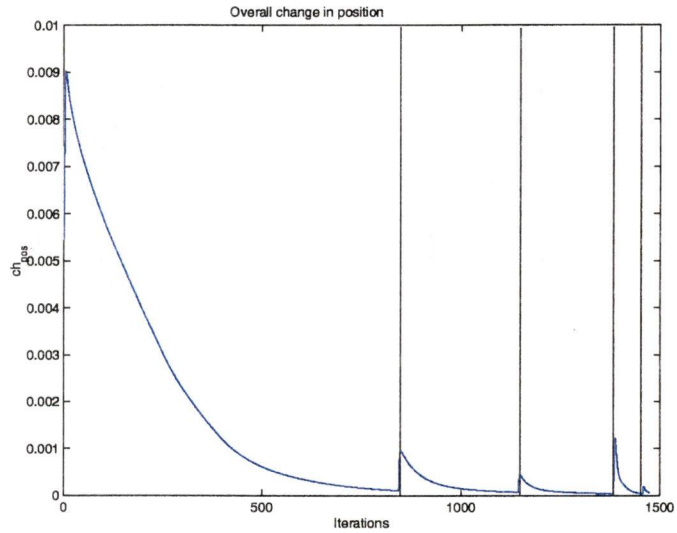
In Figure 4.8, we illustrate the evolution of the stopping criteria, for the case of a change in the value of β. In Figure 4.9, we show the criteria used for the final stop.

4.2 Segmentation

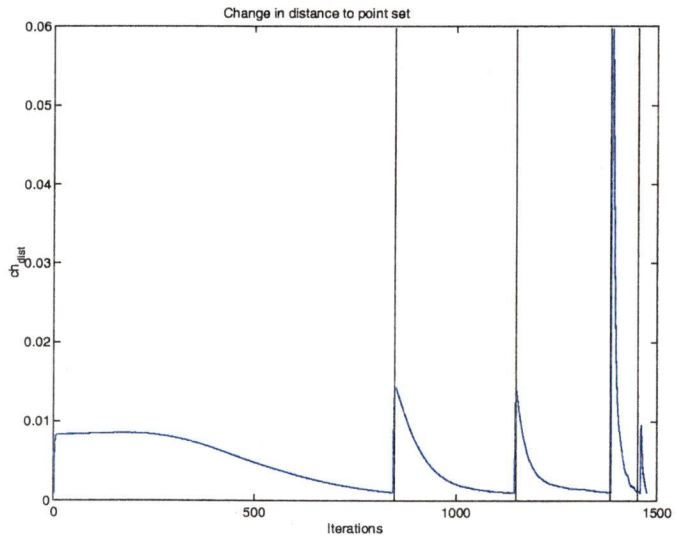
Once mesh deformation is complete, we examine the result to determine if the mesh should be segmented. The motivation for our segmentation rule, which comes from the algorithms for object part segmentation described in Section 2.2, can be stated as follows: *If in the mesh there exists a closed contour formed by vertices where the curvature has a large negative value then that contour indicates the presence of more than one object or object part, if and only if that contour cuts the mesh into two connected patches*

The key areas that indicate boundaries between objects or object parts are those that have large negative curvature. For a general discrete mesh, there are numerous methods, some quite complex, to compute curvature [57, 58]. In the case of the simplex mesh, however, curvature calculations are straightforward. From the definition of simplex angle in Equation (3.2), we recall the sine of the simplex angle:

$$\sin(\varphi_i) = \frac{r_i}{R_i} * sign(\overline{P_i P_{N_1(i)}} \cdot \overline{N_i}). \tag{4.9}$$

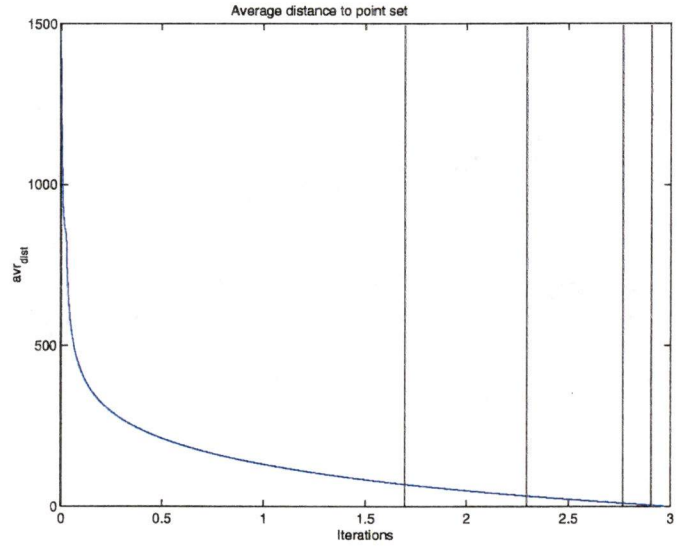


(a)

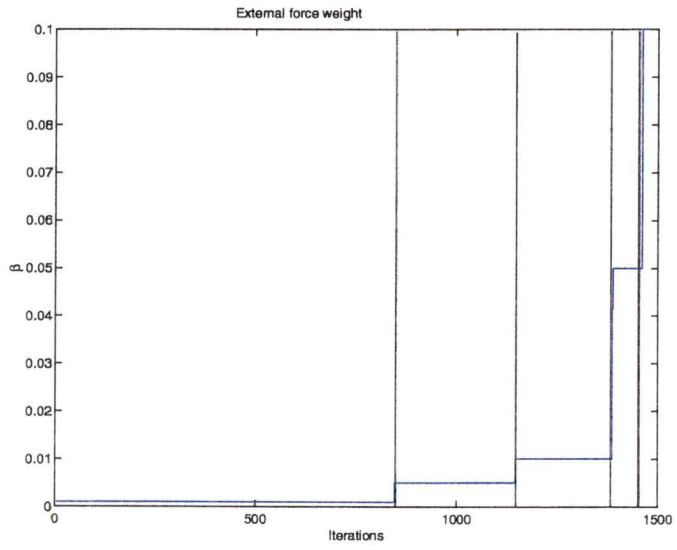


(b)

Figure 4.8: There are two stopping criteria for stops to update β.(a) The change in distance to the point set. (b) The change in position of the meshe's vertices. The vertical lines indicate a change in the value of β.



(a)



(b)

Figure 4.9: There are two stopping criteria for the final model. (a) The average distance of the mesh to the point set. (b) The values of β. If the mesh is still far from the point set but β reach its bounding value, then the deforrmation cannot go any further. The vertical lines indicate a change in the value of β. In the case presented here, the deformation stopped when the distance became zero, the value of β at the stopping point was still smaller than 0.5.

We can then calculate the mean curvature as:

$$H_i = \frac{\sin(\varphi_i)}{r_i}, \tag{4.10}$$

which is basically the inverse of the radius of the circumscribed sphere. This coincides with the notion of curvature on a continuous surface as the curvature at a point on a surface is equal to the inverse of the radius of the sphere to which the point is tangent [59].

The sign of the curvature, which is crucial for our system, is calculated by:

$$H_{sign} = \text{sign}(\overline{P_1 P_{N_1(1)}} \cdot \overline{N_1}). \tag{4.11}$$

This sign is given by the sign of the cosine of the angle between the normal vector and the vector formed by the vertex and its first neighbor. Noting Figure 4.10, the curvature will be negative if the vertex is "below" the plane formed by its three neighbors and positive if it is "above".

We note that the correct sign of the curvature depends on the consistent computation of the normal. Since the triangles where the normals are calculated are not connected through any side, as in a triangle mesh, consistent calculation is difficult. To solve this problem, we rely on the fact that a sphere must have the same curvature sign everywhere. Using the initial mesh, which is a discretized sphere, we arrange the order of the neighbors so that the curvature is always positive.

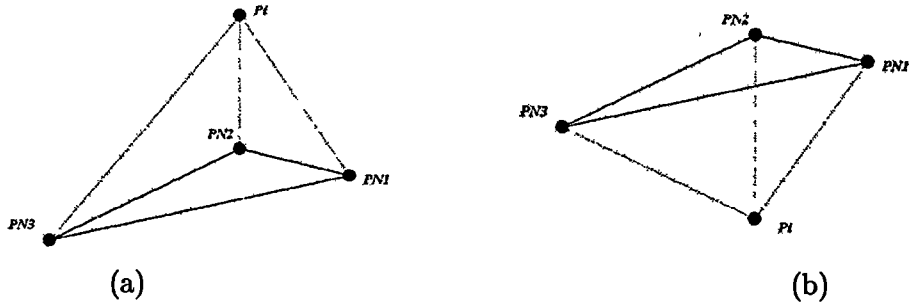


Figure 4.10: Sign of the curvature on a simplex mesh. (a) The curvature will be positive if the vertex is above the plane defined by its three neighbors. (b) The curvature will be negative if the vertex is below the plane. Note that "above" and "below" are relative to the global coordinate system.

From experimental observations, we have noted other areas of the mesh that sometimes can indicate a boundary between objects or object parts. These areas are formed by vertices that are far from the data set (further than ζ from the closest data point), and can occur due to two factors: lack of data points and closeness to a sharp concavity.

There are four cases when vertices can be far from the data due to a lack of data points. These cases are outliers, incomplete data, sparse data, and boundary between two separated objects. For purposes of segmentation we are interested in identifying the last case. The other three cases are easily identified and ignored. In the case of outliers, as shown in Figure 4.11, very small patches of vertices that are close to the data set (vertices whose distance to the closest data point is smaller than ζ, shown in red) will be created. For the case of sparse or incomplete data, the areas of far away vertices will be totally surrounded by vertices that are close to the data set. In Figure 4.12 we see an area of the "bunny" model where there are holes in the data set (incomplete data), these holes create areas of far away vertices that are enclosed by vertices that are close

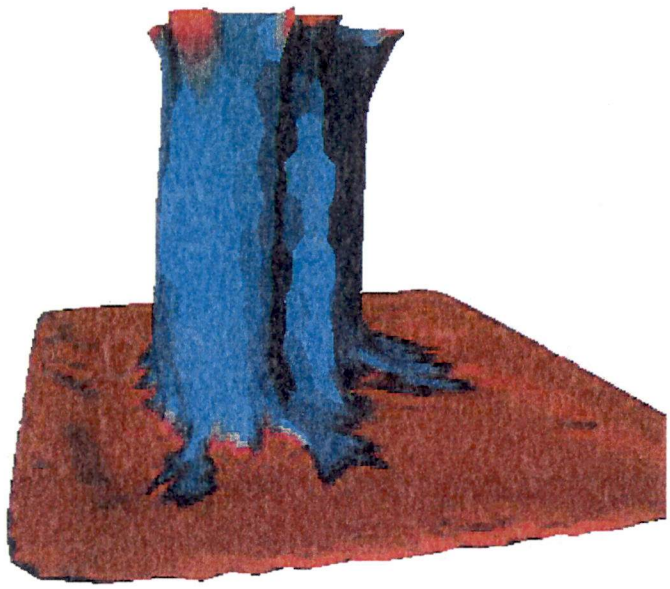


Figure 4.11: Outliers create very small areas of vertices that are close to the data set. These areas are shown in red.

to the data set. The last case, a true boundary, is illustrated in Figure 4.13. In this figure we can see an area of vertices that are far from the data set(shown in blue) in the region between two separated objects.

The other factor that generate vertices far from the data is that the vertices may be near a sharp concavity, as illustrated in Figure 4.14. The new position of the vertex P_i depends on the internal and external force. The external force F_{ext} tends to move the vertex towards the corner point. The internal force F_{int}, however, tends to move P_i towards the plane formed by its neighbors.

With the above observations, we propose the following segmentation algorithm.

1. Label each vertex in the simplex mesh as OBJECT or NON-OBJECT. The vertices

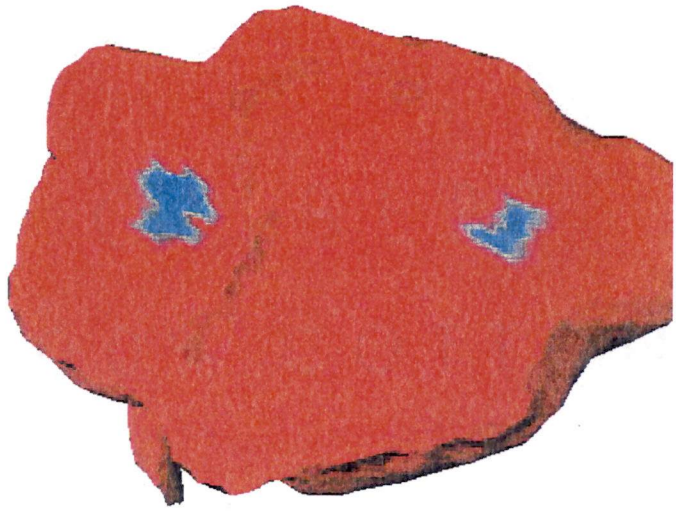


Figure 4.12: Incomplete data can create a region of vertices that are far from the point set. This region cannot be a boundary between objects since it is totally surrounded by vertices that are close to the data set.

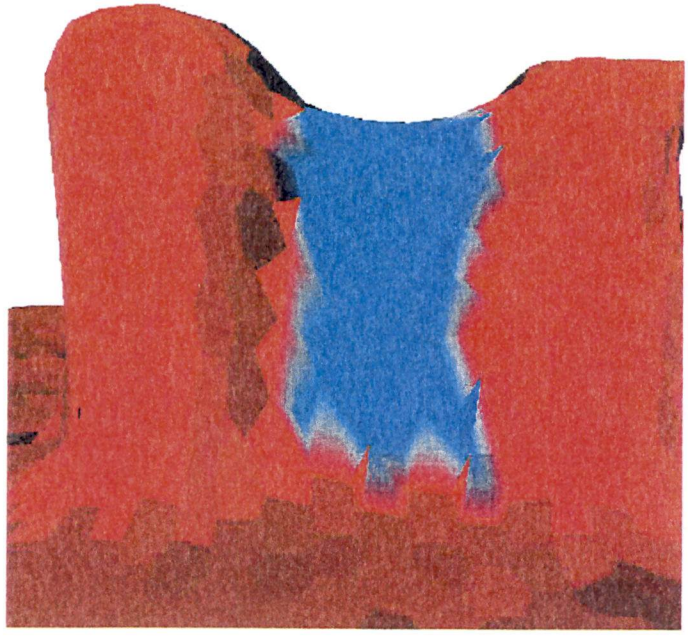


Figure 4.13: A boundary between two separated objects can create a region of vertices that are far from the point set.

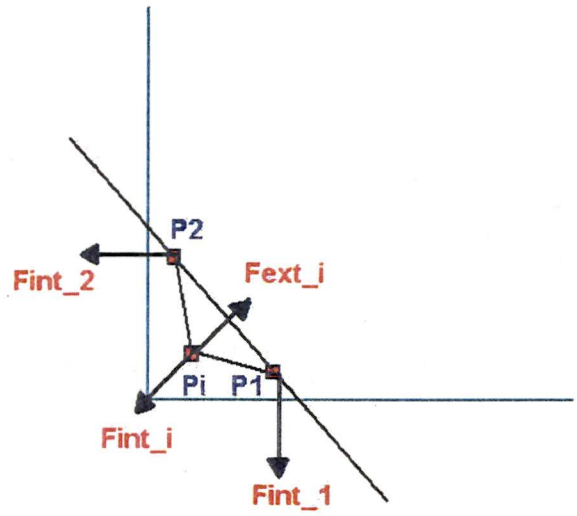


Figure 4.14: The vertex P_i does not get closer to its closest point due two opposite internal and external forces.

whose distances to the closest point are less than ζ are OBJECT, otherwise they are NON-OBJECT.

2. Check the curvature in each of the OBJECT vertices. If it is highly negative, then label the vertex as NON-OBJECT.

3. Clean isolated islands by checking every vertex. If a vertex is OBJECT and is totally surrounded by NON-OBJECT vertices, make it NON-OBJECT and vice versa.

4. Apply connected component analysis to label each of the patches formed by OBJECT vertices.

5. Segment the cloud of points using closeness of the data points to the segmented

mesh patches.

The end result is a cloud of points for the background information, and one point cloud for each object or object part in the input scene. We then begin a new fitting process for each individual cloud, followed by refinement as described in Section 3.4.

Chapter 5

Results

In this chapter, we examine the performance of our 3D reconstruction-segmentation system with several experiments. For those experiments, we use simulated (artificial) data as well as real-world data obtained by several laser range scanners. To examine the performance with noisy data in a controlled scenario, we add various levels of noise to the simulated data. To emulate the performance of real range scanners, the (x,y,z) coordinates of the simulated point clouds are transformed into (r,θ,ϕ) spherical coordinates and noise is added along only the range coordinate. The noise corrupted range value, $\hat{r}$, is given by

$$\hat{r} = r + \kappa \text{rand}()(\text{r}_{\text{max}} - \text{r}_{\text{min}}), \tag{5.1}$$

where r_{max} and r_{min} are, respectively, the minimum and maximum ranges in the data set, κ is a percentage value, and rand() is a random number generator with a range

(-1,1).

The simulated data was obtained using a software range scanning simulator that uses 3D models as input. These models are rendered in a window where the point of view can be modified and a range scan can be taken from any point of view. To get information about a complete scene, scans are taken from several points of view.

To obtain an objective measure of the quality of the mesh we use a software tool known as *Metro* [60]. *Metro* compares two triangle meshes representing the same surface and evaluates the difference using an approximation error measure. The error is calculated as the integral of the distance between the two meshes. To get this distance, each face of one of the meshes is sampled and the the distance between each sample and the other mesh is computed by

$$\epsilon(p, S) = \min_{p' \in S} d(p, p'), \tag{5.2}$$

where $d(\cdot)$ is the Euclidean distance between two points.

5.1 Single Object 3D Reconstruction

In this section, we examine results obtained using the four different object models shown in Figure 5.1. The "t-pipe" and the "valve" are synthetic models constructed by primitives. We obtain clouds of points for these two objects using the range scanner simulator. Using the simulator, we take range scans from many different points of view so we obtain clouds of points that represent the complete objects. The different range

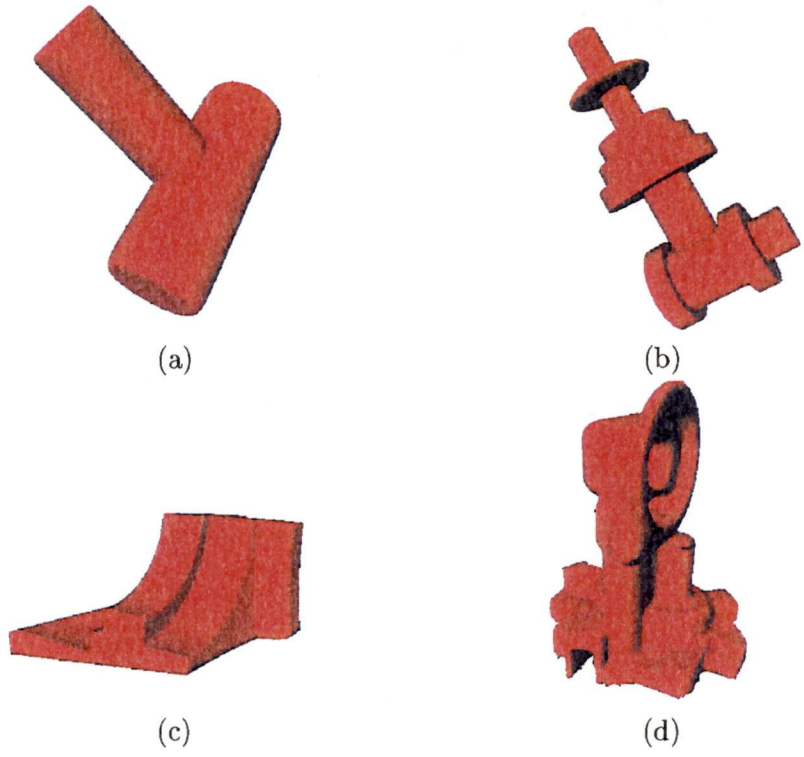


Figure 5.1: Original 3D models of single objects. Four different objects were used to test the reconstruction of single objects: (a)t-pipe, (b) valve, (c) fandisk, (d) oil pump.

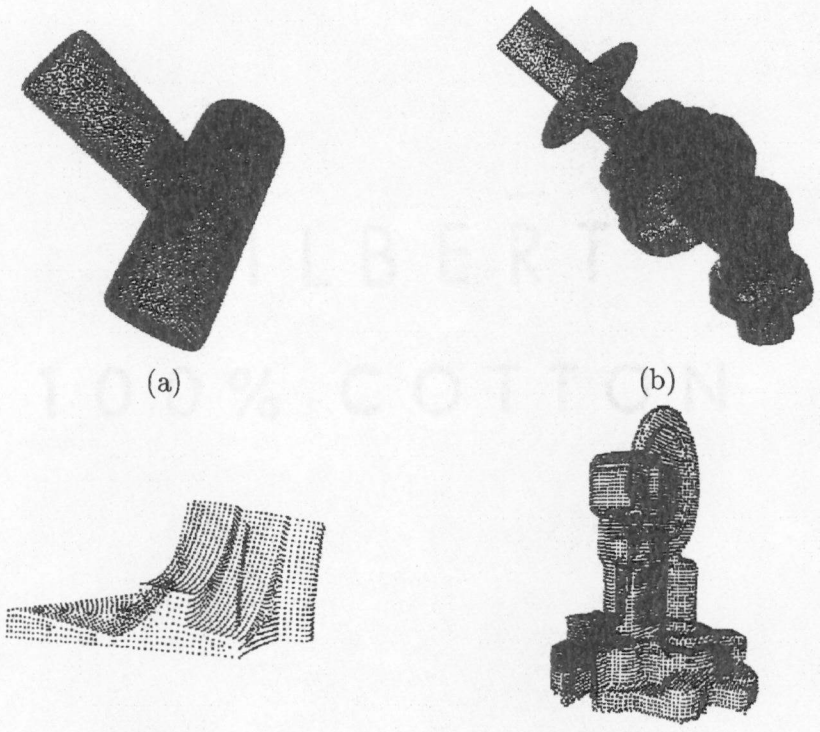


Figure 5.2: Clouds of points obtained from the original 3D models: (a) "t-pipe" 47286 points, (b) "valve" 64321 points, (c) "fandisk" 6475 points, (d) "oil pump" 22741 points. The cloud of points for the "t-pipe" and the "valve" were obtained using a range scanner simulator, the ones for the "fandisk" and the "oil pump" are the vertices of the model's triangle meshes.

scans are transformed so they lie on the same coordinate system and therefore can be combined in one cloud of points. The "fandisk" and the "oil pump" are detailed triangle meshes often used in the literature. With these two models, we use the vertices of the triangle meshes as the point clouds. The clouds of points corresponding to each of the models are shown in Figure 5.2. Table 5.1 lists the different parameters for the deformation of the simplex mesh (i.e. number of points, number of faces, total iterations, etc.) for all of the different experiments.

Table 5.1: Deformation parameters for single object models

Model	Points in cloud	Vertices in mesh	Deformation iterations	Refinement iterations	ζ
T-pipe	47286	5000	2068	500	0.01
Valve	64321	8192	2047	300	0.0003
Oil pump	22741	5000	1408	200	0.0002
Fandisk 1	6475	512	1869	700	0.008
Fandisk 2	6475	1568	2932	700	0.008
Fandisk 3	6475	2048	2928	700	0.008
Fandisk 4	6475	3200	2902	700	0.008
Fandisk 5	6475	3528	2896	700	0.008
Fandisk 6	6475	4608	2875	700	0.008
Fandisk noise 1	6475	3200	2907	700	0.00691
Fandisk noise 2	6475	3200	2842	700	0.00603
Fandisk noise 3	6475	3200	2765	700	0.00565
Fandisk noise 4	6475	3200	2724	700	0.00569

An illustration of the 3D reconstruction for "tpipe" model is shown in Figure 5.3. Figures 5.3 (a)-(i) correspond to different stages of the deformation, while Figures 5.3 (j)-(l) show different iterations of the refinement process. Deformation creates regions on the mesh that are not smooth but refinement corrects those areas and makes the edges sharper as well. The reconstructions for the "valve" and the "oil pump" are shown in Figures 5.4 and 5.5, respectively. For the "valve" example the refinement process again fixes regions where the mesh is not smooth. In the case of the "oil pump" the final result is not as visually pleasing as the others, indicating that not enough vertices are used.

Using the "fandisk", a valid representative of CAD models (sharp edges and sophisticated surface curvature), we examine both the quantitative and qualitative accuracy

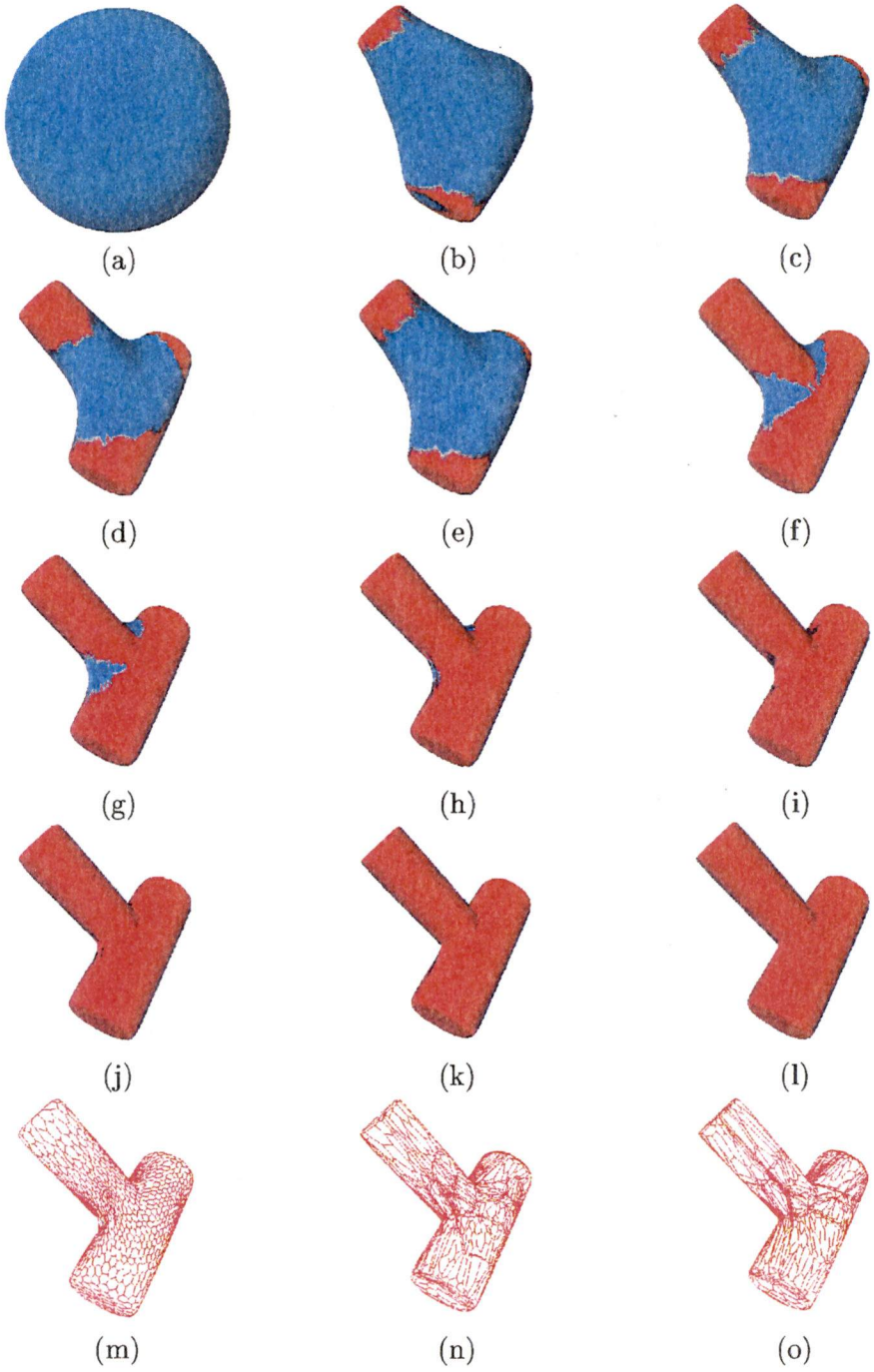


Figure 5.3: Reconstruction of a t-pipe model. (a)-(i) The simplex is shown at different iteration of the deformation process. (j)-(l) The simplex mesh is shown at different iterations of the refinement process. (m)-(n) The simplex mesh for the refinement outputs. Note the vertex redistribution achieved by the refinement algorithm. In this figure and all the following ones related with deformation the blue veritces are those far from the data set whereas the red ones are the ones already on the data set.

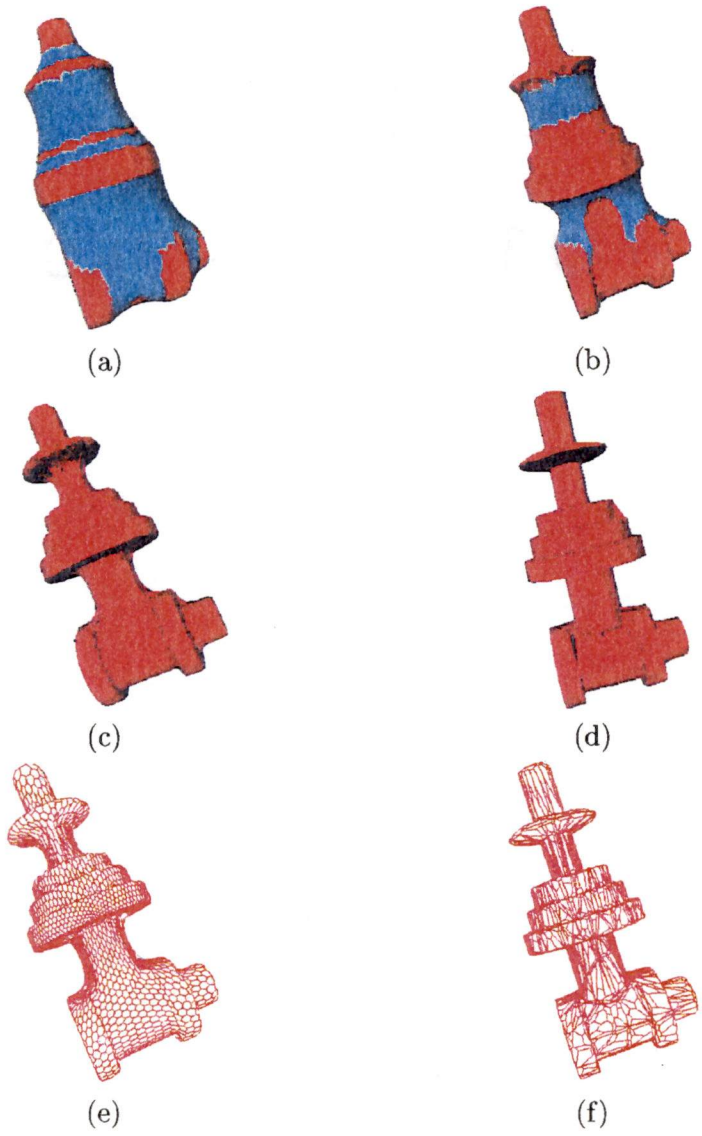


Figure 5.4: Valve reconstruction. (a)-(c) The deformation process at different iterations. (d) Output of the refinemet process. (e) Simplex mesh for (c). (f) Simplex mesh for (d). Note the vertex redistribution achieved by the refinement algorithm.

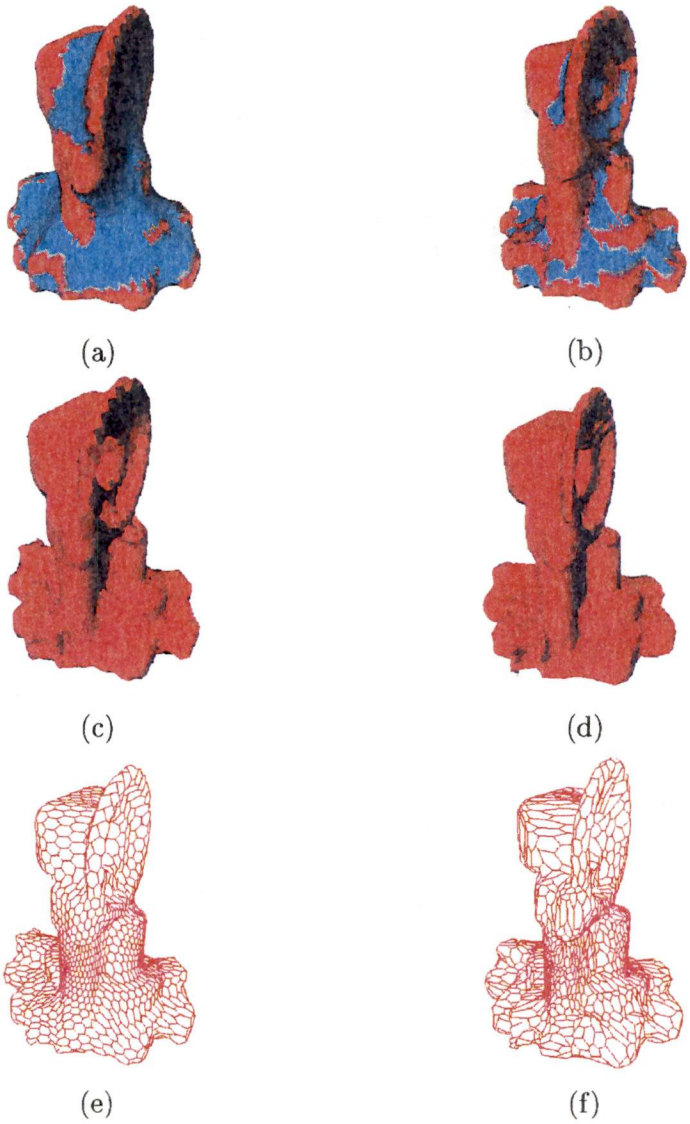


Figure 5.5: Oil pump reconstruction. (a)-(c) The deformation process at different iterations. (d) Output of the refinemet process. (e) Simplex mesh for (c). (f) Simplex mesh for (d). Note the vertex redistribution achieved by the refinement algorithm.

of our output. The quantitative measurements are obtained using *Metro*.

First we examine the influence of the number of vertices in the simplex mesh. In Figure 5.6 the output of the deformation for simplex meshes with an increasing number of points is shown. As the number of points increases, the model looks better as expected, if the refinement process is applied (Figure 5.7), the result looks even better.

The error between the reconstructed models and the original model, for the reconstruction and the refinement outputs, is graphed in Figure 5.8. We note that the quantitative error does not necessarily correspond to the the subjective quality. Although refinement either marginally improves or leaves unchanged the quantitative error, we note that subjective quality is improved. The reason for this is that the refined models have sharper edges, an appealing feature for humans, but some of their faces have very small area, which is penalized by *Metro*.

We also check the performance of our system against noise. Different levels of noise are added to the original cloud of points (Figure 5.9). The outputs of the deformation and the refinement processes are shown in Figures 5.10 and 5.11, respectively. Obviously, increased noise decreases the subjective quality. The refinement process, however, still improves the final model. As shown in Figure 5.12 the quantitative error also increases with the addition of noise. Our output does a good job for noise smoothing as evident if we compare to a model where the noisy points are used as the vertices of the mesh as in Figure 5.13.

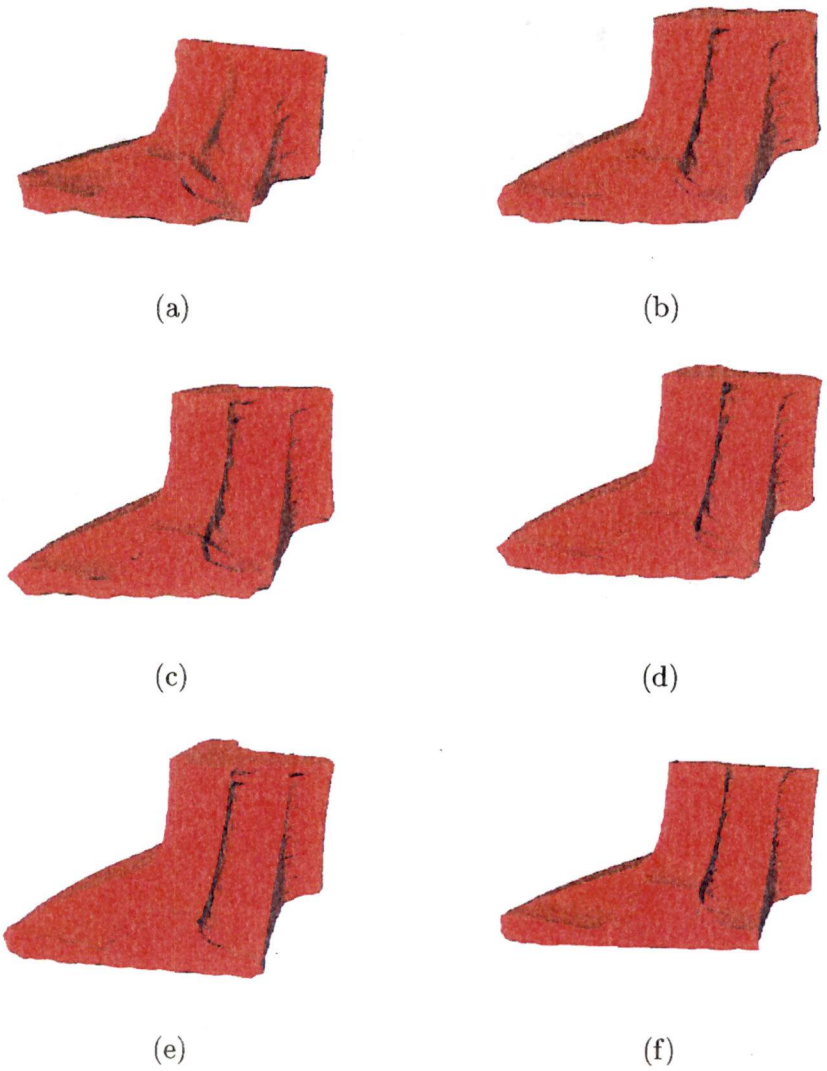


Figure 5.6: Output of the deformation process for the fandisk with different number of vertices in the simplex mesh. (a)512 vertices, (b)1568 vertices, (c)2048 vertices, (d)3200 vertices, (e)3528 vertices, (f)4608 vertices

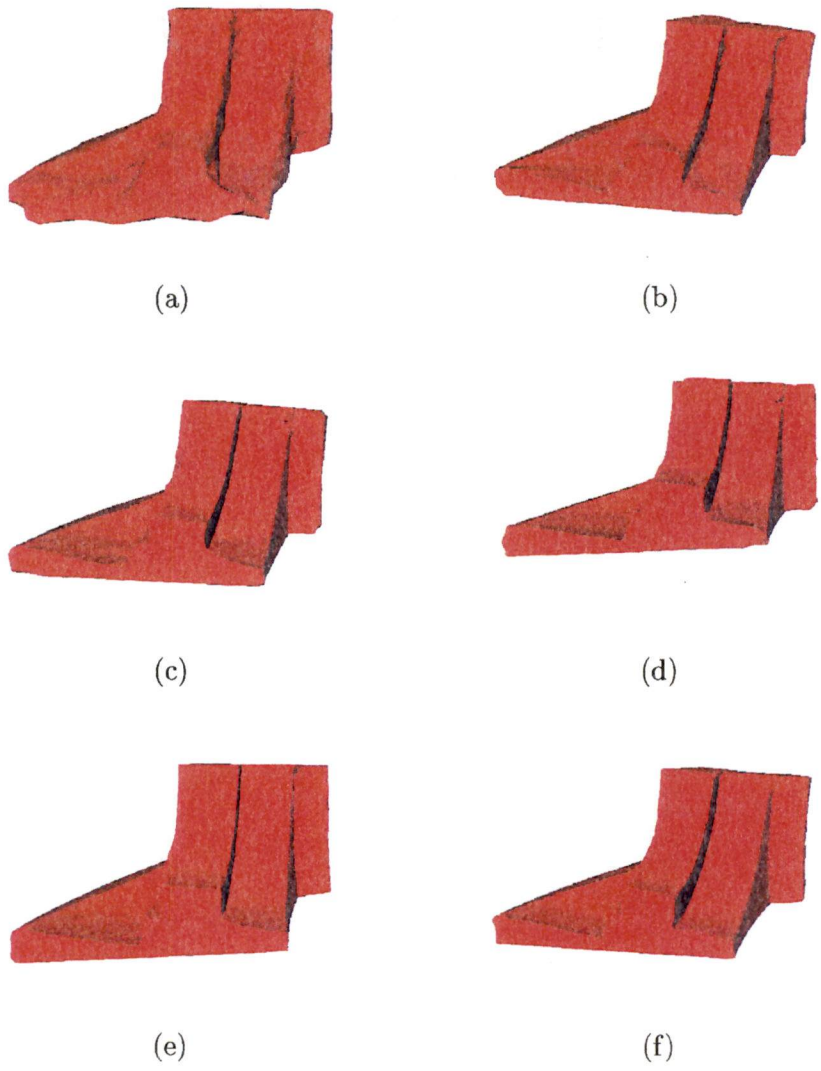


Figure 5.7: Output of the refinement process for the fandisk with different number of vertices in the simplex mesh. (a)512 vertices, (b)1568 vertices, (c)2048 vertices, (d)3200 vertices, (e)3528 vertices, (f)4608 vertices

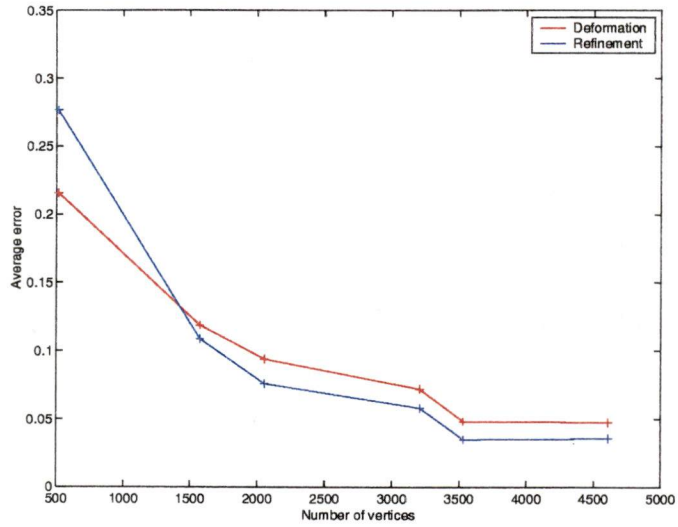


Figure 5.8: Average error of deformation and refinement outputs for the "fandisk" using different number of vertices.

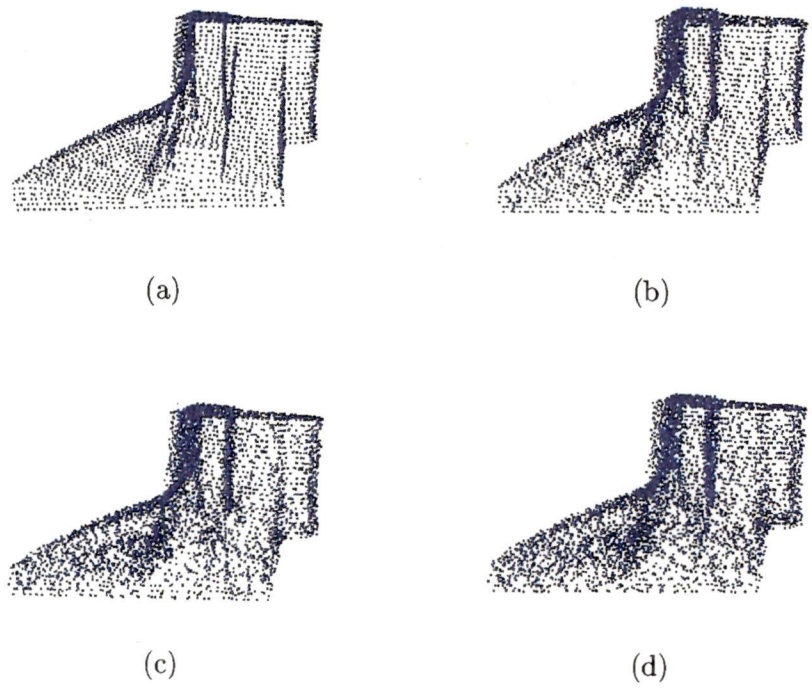


Figure 5.9: "Fandisk" point cloud with different levels of noise added. Recalling Equation (5.1) (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$.

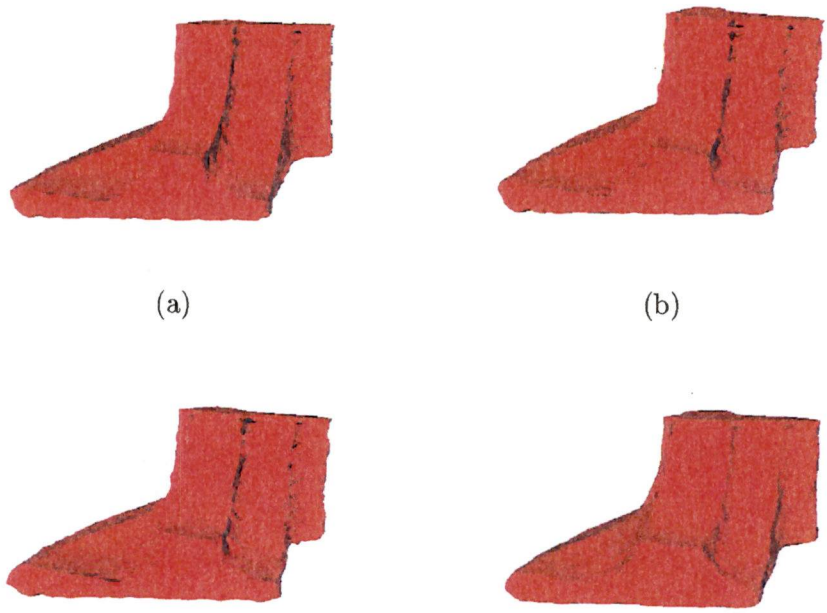
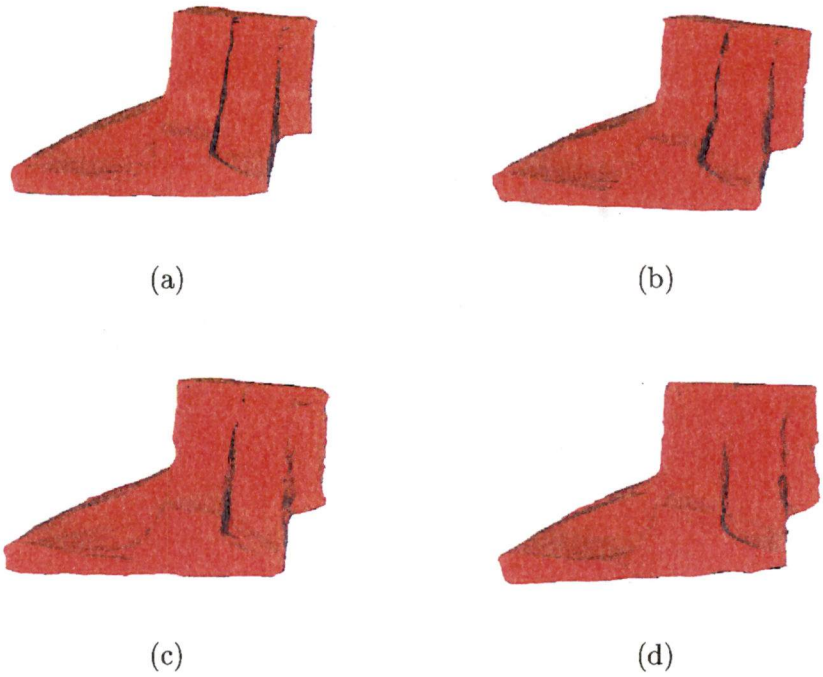


Figure 5.10: Deformation output for the fandisk with different levels of noise added. Recalling Equation 5.1 (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$. The simplex mesh has 3528 vertices and 1766 faces.



(a) (b)

(c) (d)

Figure 5.11: Refinement output for the fandisk with different levels of noise added. Recalling Equation (5.1) (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$. The simplex mesh has 3528 vertices and 1766 faces.

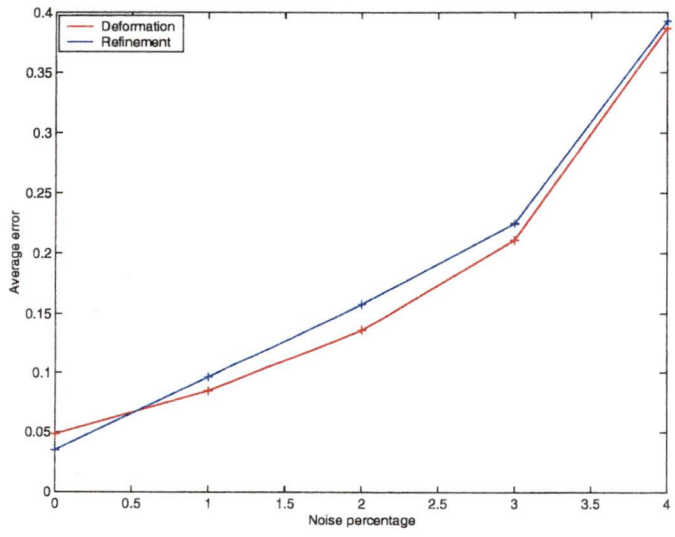
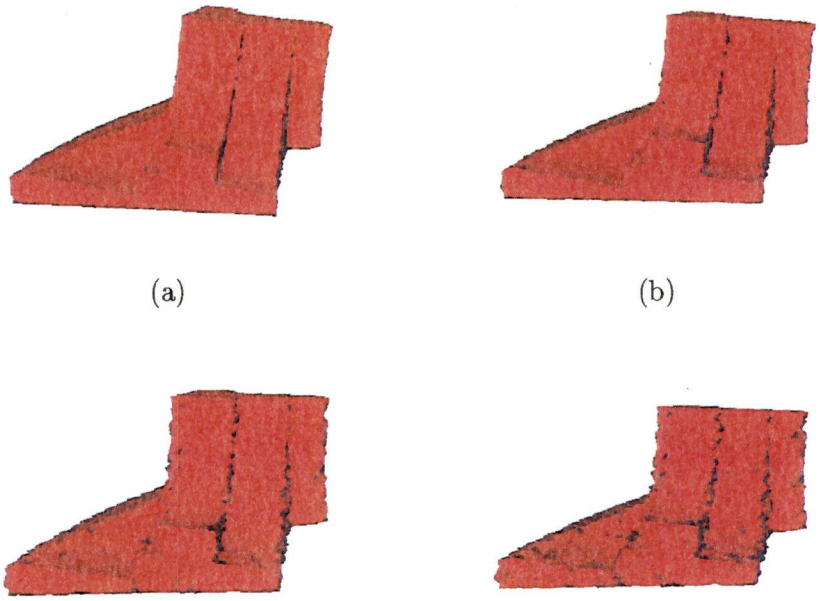


Figure 5.12: Average error of deformation and refinement outputs for the "fandisk" adding different levels of noise.



(a) (b)

(c) (d)

Figure 5.13: Model of the fandisk using noisy points as vertices of the mesh. Different levels of noise are added to the original vertices of the model. Recalling Equation (5.1) (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$. Compare to Figure 5.10.

Table 5.2: Deformation and segmentation parameters for scene models

Model	Points in cloud	Vertices in mesh	Deformation iterations	ζ	Objects found
Box	18517	2048	1953	0.001	2
Box part 1	1087	512	643	0.008	N/A
Box noise 1	18517	2048	2029	0.002	2
Box noise 2	18517	2048	1929	0.011	2
Box noise 3	18517	2048	1789	0 014	2
Box noise 4	18517	2048	1762	0.016	2
Box noise 5	18517	2048	1659	0.019	1
Box noise 6	18517	2048	1084	0.05	1
Coleman data	68967	5000	1144	0.0403	5
Line scan data	36774	4232	1515	0.127	6
Perceptron data	261039	6272	2886	0.288	5

5.2 Scene Reconstruction-Segmentation

To examine the performance of the reconstruction-segmentation algorithm, we use four different datasets. One was constructed using the range scanner simulator and a model of a box on a floor. The other three are actual clouds of points obtained from real indoor scenes using three different laser range scanners. These clouds are shown in Figure 5.14. All the deformation and segmentation parameters are listed in table 5.2.

In the first experiment, we use the cloud of points from the simulated box (Figure 5.14 (a)). The deformation process is applied until it stops, giving, as output, the model shown in Figure 5.15 (a). The mesh of this model has some vertices that are still not close enough to the data set which are highlighted. These vertices are in the area of concavity that is the boundary between the box and the floor, but do not form a clear connected patch that separates the objects. The segmentation step examines all the

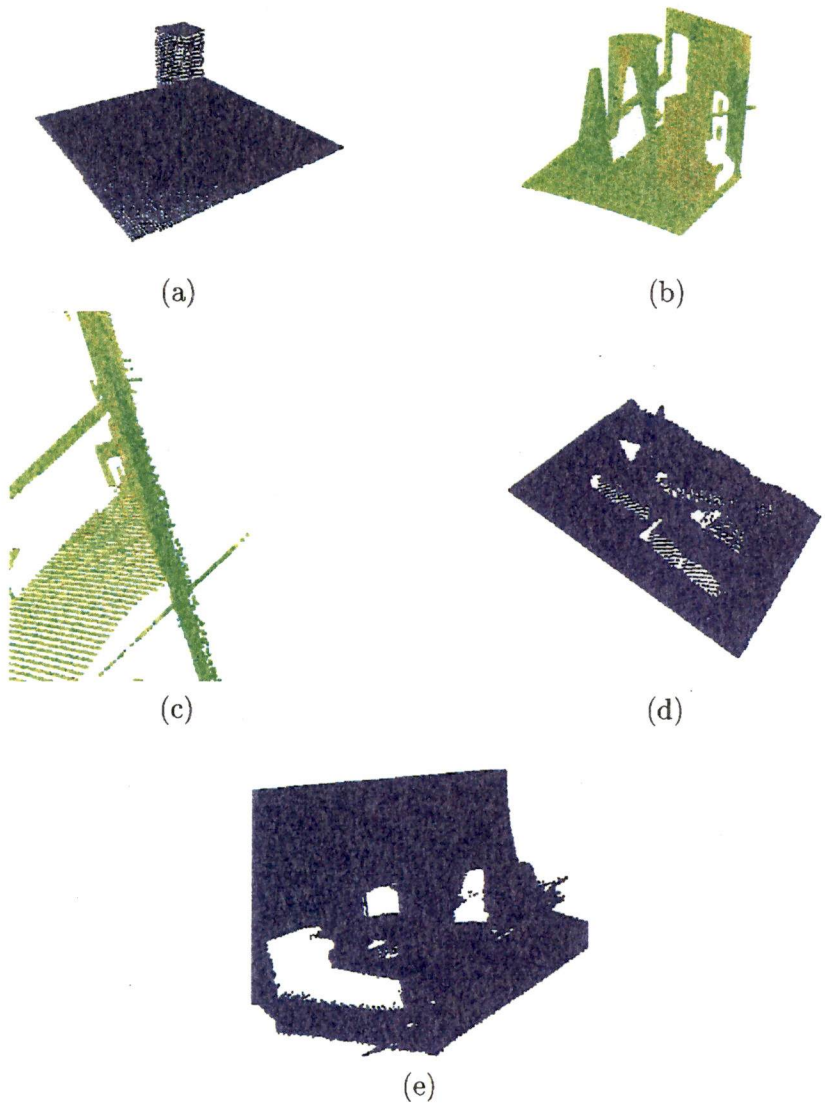


Figure 5.14: Clouds of points that represent synthetic and real scenes used to test the reconstruction-segmentation algorithm. (a)Data set that represents a box on the floor, the cloud was created using a simulator range scanner. (b)Data set created using a Coleman laser range scanner. The data given by this scanner is almost free of noise, however, outliers can occur.(c) Close up on the Coleman data to a region with outliers. (d) Data set created using a line scan camera. (e) Data set created using a Percertron laser range scanner. The cloud is composed of data from teo different points of view registered with a manual registration technique.

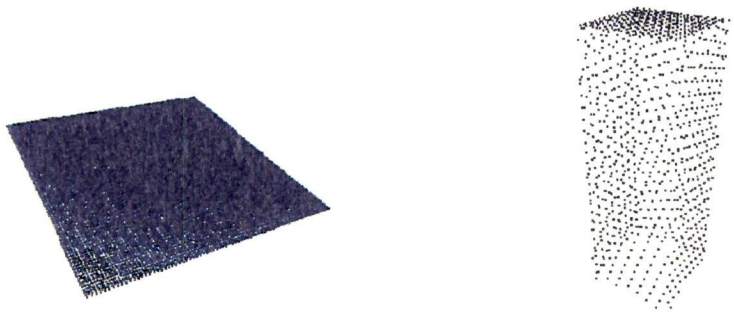


(a) (b)

Figure 5.15: Outputs of the deformation and segmentation processes applied on the data set that represents a box. (a)Deformation output. There are some NON-OBJECT vertices(in blue) but not a clear boundary between the box and the floor.(b)Segmentation output. The segmentations creates a clear boudary between the box and the floor.

vertices and identifies a clear boundary between the two objects. In Figure 5.15 (b), we can see the regions of the mesh that correspond to each object. In Figure 5.16 we show the point clouds corresponding to the box and the floor. Using these new clouds, we can apply the deformation algorithm and get a model for each. In Figure 5.17, the model for the cloud that represents the box is shown.

Noise has an undesirable effect in the segmentation algorithm. The algorithm relies



(a) (b)

Figure 5.16: Box: Original data set partitioned into (a) background and (b) object.



Figure 5.17: Model of the segmented box.

on the calculation of curvature to find the boundaries between different mesh regions. With the addition of noise, the segmentation algorithm labels several vertices as being part of a boundary between objects, when they are really part of the object. This is due to the creation of concavities in regions where the object should be flat. In Figure 5.18, we show the output of the segmentation algorithm using clouds of points with different levels of noise added (Figure 5.19). As we can see in Figure 5.18, increasing the noise increases number of false boundaries.

In Figure 5.14 (b) we show the cloud of points of a real scene obtained using a Coleman laser range scanner. The data that this scanner gives is very clean; it reduces noise by taking several measurements at each point and averaging them, outliers, however, still occur. In Figure 5.14 (c) a close up to a region of the cloud is shown; here we can see an outlying line of points. When the deformation process stops as shown in Figure 5.20 (a), the vertices that are still far from the data create boundaries between regions corresponding to the different objects in the scene. Still these boundaries are not ready for segmentation as there some points where they do not completely separate the object

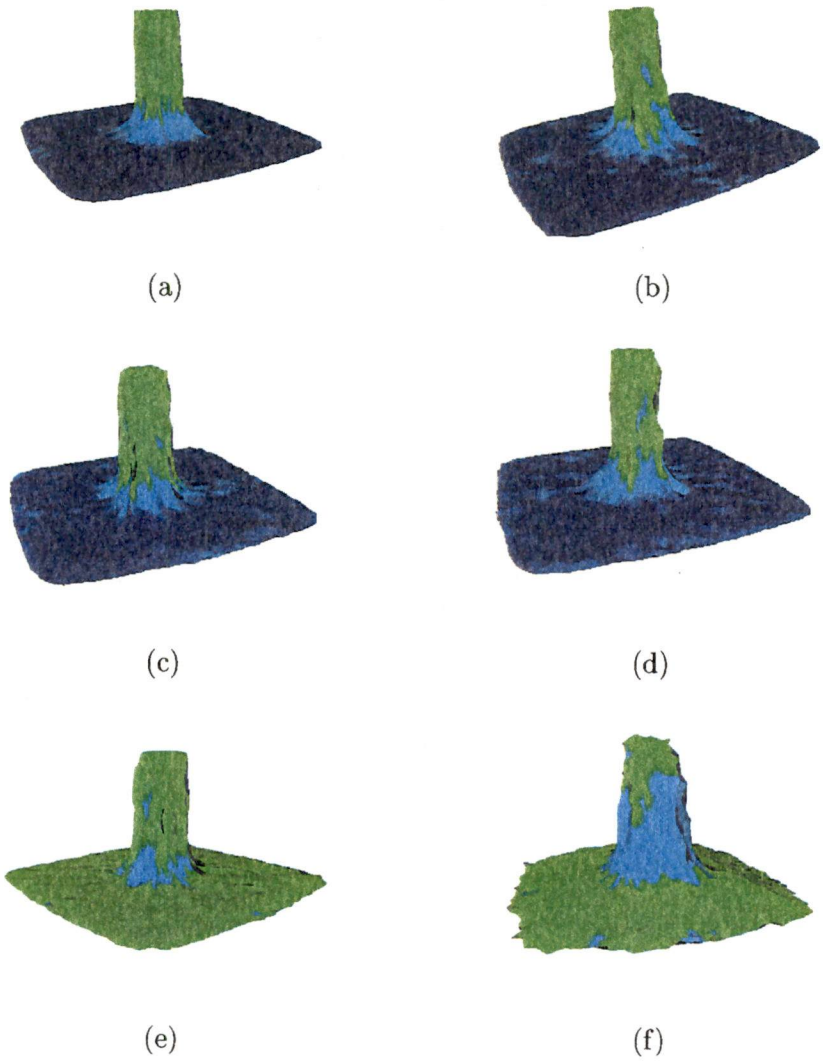


Figure 5.18: Output of the segmentation algorithm for the data set of the box with different levels of noise added. Recalling Equation 5.1 (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$.

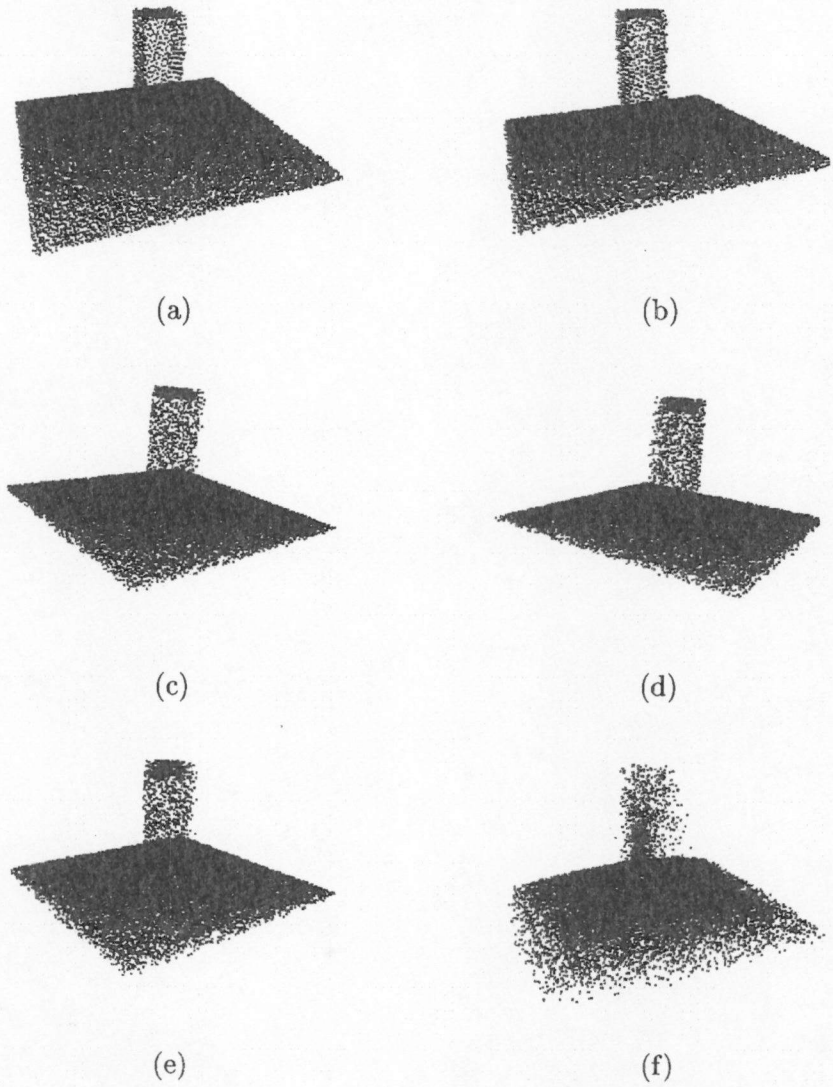


Figure 5.19: Cloud of points from a box on the floor with different levels of noise added. Recalling Equation 5.1 (a)$\kappa = 1\%$, (b)$\kappa = 2\%$, (c)$\kappa = 3\%$, and (d)$\kappa = 4\%$.

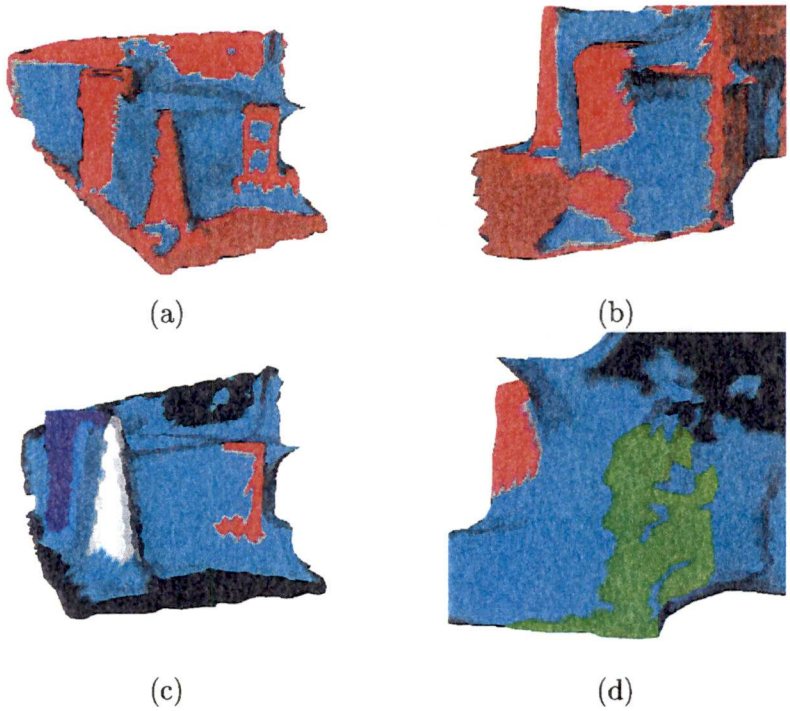


Figure 5.20: Output of the deformation and segmentation for the Coleman scanner data. (a)Simplex mesh after deformation. (b)Close up on the deformed simplex mesh in the area of outliers.(c)Simplex mesh after segmentation. (d)Close up on the segmented simplex mesh in the area of outliers. The outliers make the segmentation create an additional region (in green) in the area of the wall.

regions. As shown in Figure 5.20 (c), after the segmentation is applied, the boundaries completely separate these regions. Due to the outliers in the region that corresponds to the wall, an extra region is created as shown in green in Figure 5.20 (d). The final segmented point clouds are shown in Figure 5.21.

The second real data set used was obtained using a line scan camera. The cloud used (Figure 5.14(c)) corresponds to a scene that contains five objects laying on the floor. In Figure 5.22, we show the output of the deformation and the segmentation processes. We can see how the segmentation algorithm correctly finds each object in

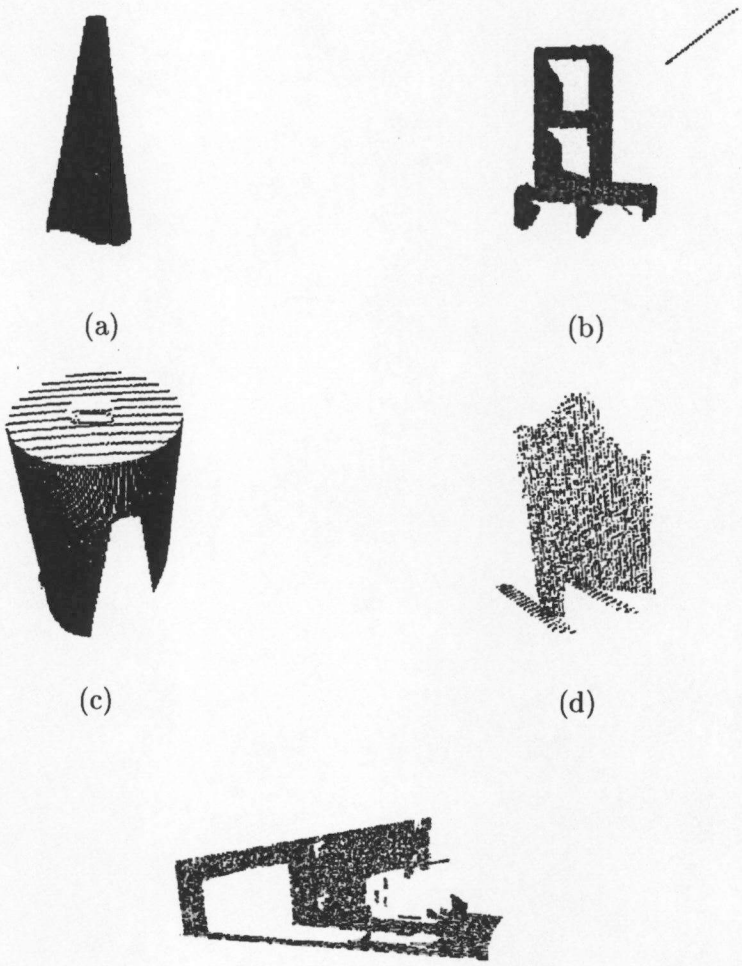
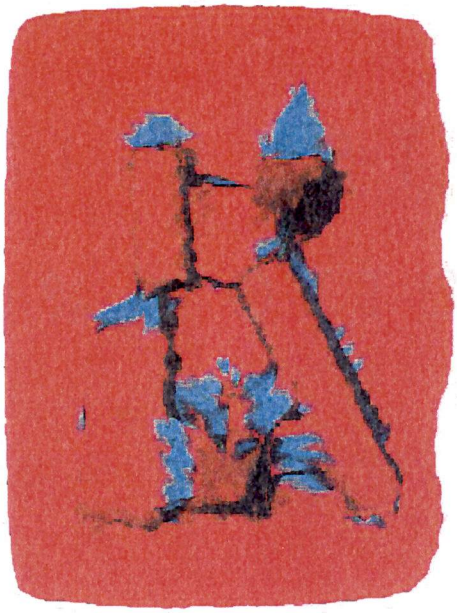


Figure 5.21: Partition of Coleman cloud of points. (b) The cloud that belongs to the bricks has the outliers data points.



(a)



(b)

Figure 5.22: Line scan camera reconstruction and segmentation

the scene. In Figure 5.23, we show the partition of the original cloud into six clouds, one each corresponding to the different objects and the floor.

Finally, we use a cloud of points created using a Perceptron laser range scanner. The cloud corresponds to four different objects in a room. The data set employed comes from two different points of view, that have been registered using Horn's [61] manual registration technique. The manual registration is far from perfect, so it additional noise is added to the data set. The mesh after deformation, as shown in Figure 5.24 (a), is messy in several regions due to the noise. The segmentation, however, still performs well as shown in Figure 5.24 (b); the four objects and the background are identified correctly. In the case of the boxes that are one on top of the other, the segmentation algorithm separates them, although there is some error (the cloud of the lower box takes a part of top one). The segmentation of the data set into separate point clouds can be seen in Figure 5.25.

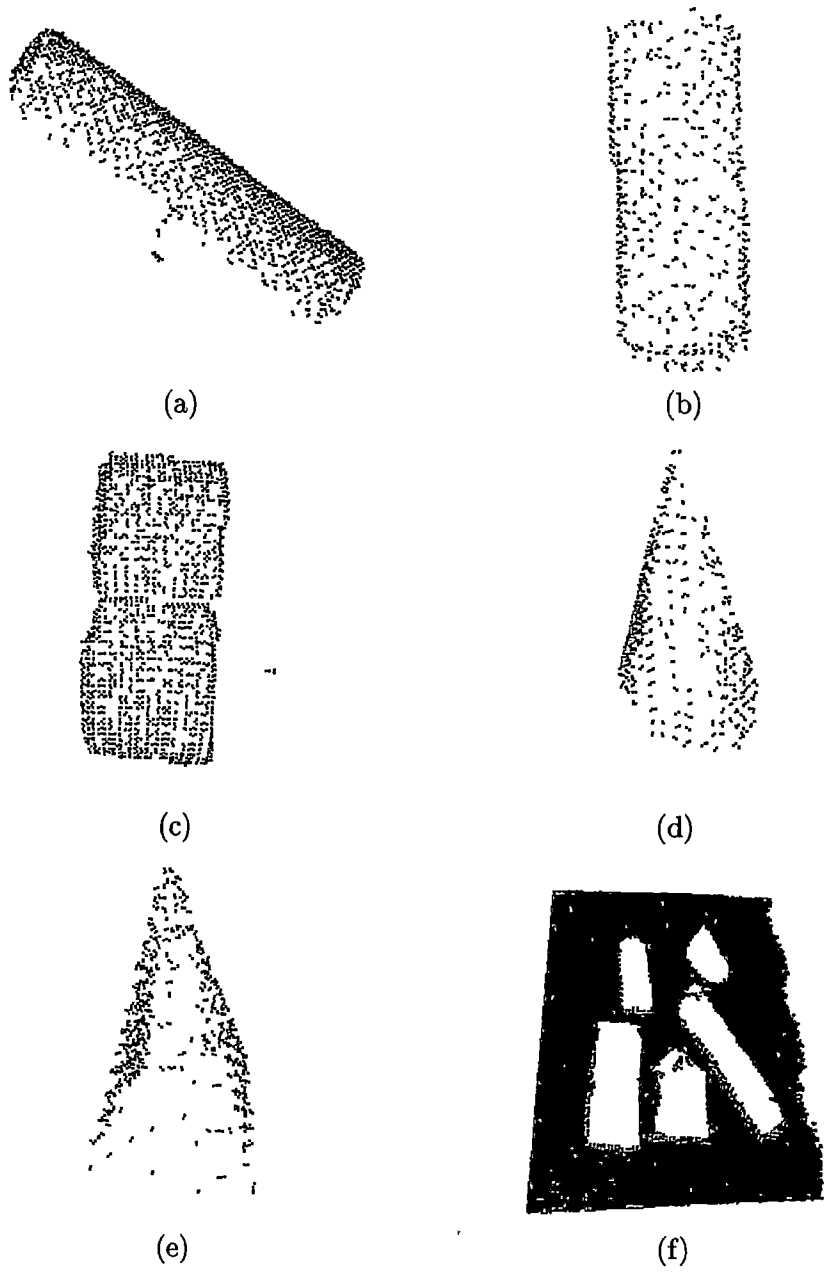
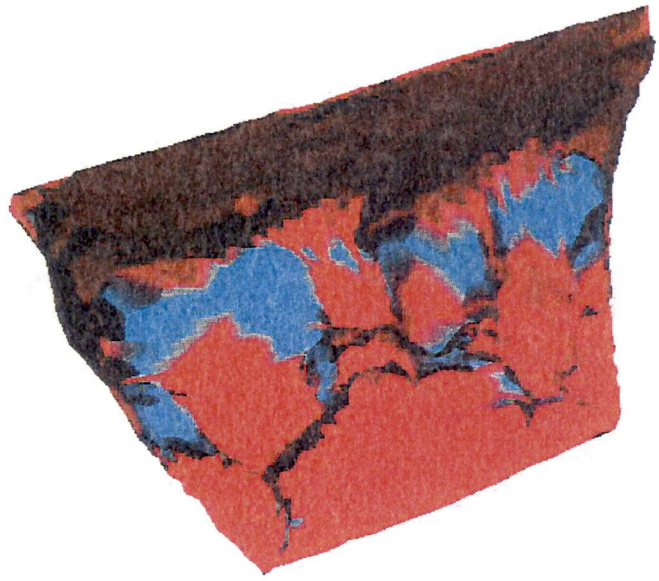
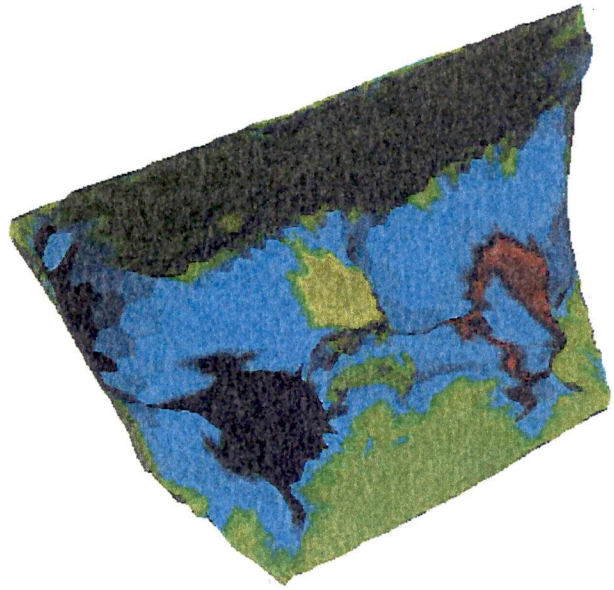


Figure 5.23· Partition of line scan data

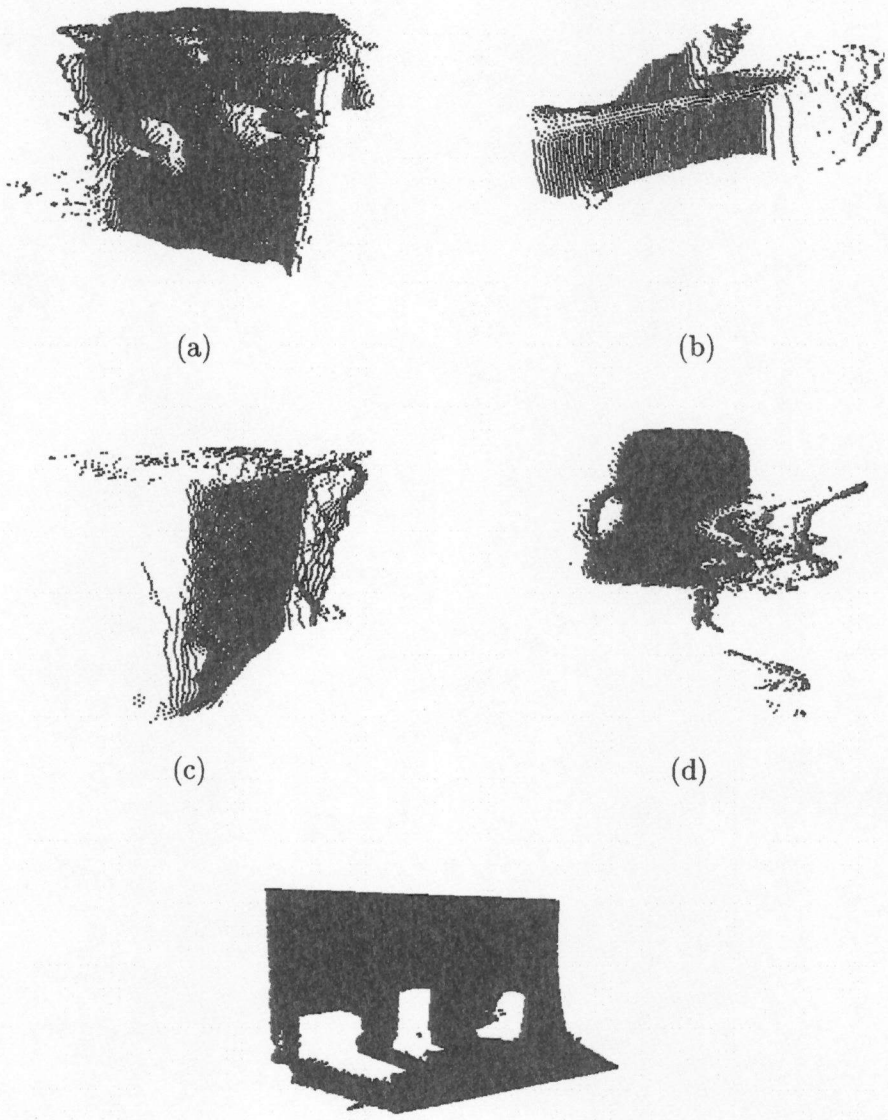


(a)



(b)

Figure 5.24: Perceptron reconstruction and segmentation



(a) (b)

(c) (d)

(e)

Figure 5.25: Partition of perceptron data

Chapter 6

Conclusions and Future Work

We have presented a system for 3D reconstruction and segmentation of multiple object, real-world scenes. We have extended the deformable simplex mesh reconstruction algorithm to use not only single objects as input, but also complete scenes. Using the basic idea that boundaries between objects are made by concave regions, we took advantage of the way the deformation of the simplex mesh evolves and constructed a segmentation algorithm.

Our implementation of simplex meshes works very well for the case of single objects. Using the refinement algorithm increases the visual quality of such models. To construct such models we used the surface orientation constraint and developed adaptive schemes to tune the required parameters. This approach improves the speed of the deformation when compared to the mean curvature continuity constraint.

Based on experimental results with real world data, the segmentation algorithm

works well in low noise scenarios. With higher levels of noise, some regions that do not correspond to complete objects are segmented. Although the scenes we have used are not overly cluttered, they do represent many scenes that can be found in practical situations. We additionally note that our system makes no assumptions about the input data; the data is totally unstructured and still the algorithm performs well.

The contributions of this thesis are:

- Modification of the simplex mesh algorithm so it can be applied to complex scenes.
  - Initialization without any a priori information. We make no assumptions at all about the input data.
  - Addition of parameter ζ. With this parameter we get rid of the shrinking problem of the surface continuity constraint. It is also used to identified regions of possible boundaries between objects.
  - Gradual increase of the external weight parameter, β. By using this scheme we obtained a smoother model.
  - Stopping criteria. These criteria allow the gradual increase of β and avoid infinite iterations in the prescence of outliers.
- The segmentation algorithm.
  - Application of a mesh segmentation. Only few researchers have studied this problem. Besides helping in the segmentation of the point clouds it also gives

an output that can be used in tasks where higher level knowledge of the scene is required.

  - Use of deformation to find boundaries between object parts. We use the evolution of the deformation to help the segmentation of the mesh. Segmentation is being performed parallel to the deformation.

- Measurement of the subjective and objective improvements of the refinement. With the experimental results we quantify the improvements given by concentrating vertices in high curvature areas.

Future work:

- Improve initialization. For the algorithm to better handle more complex scenes, a different initialization could be used. A good initialization, to continue assuming no structure at all, could be to build a binary volumetric representation and extract the isosurface with an algorithm similar to the marching cubes.

- A more robust curvature calculation. This improved curvature could take into account a bigger scale, not only the three adjacent neighbors, to make a better decision about boundaries between objects.

- Improve the refinement algorithm. A way to concentrate faces, and not only vertices, in high detail areas could be found so as to improve the quantitative performance.

- If available, include structure information. A way to include structure information to make the reconstruction more robust could be included.

Bibliography

Bibliography

[1] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. Technical report, Istituto di Elaborazione dell'Informazione-Consiglio Nazionale delle Ricerche, September 1997.

[2] H. Delingette. General object reconstruction based on simplex meshes. *International Journal of Computer Vision*, 32(2):111–146, September 1999.

[3] H. Delingette. Simplex meshes: A general representation for 3d shape reconstruction. Technical Report 2214, INRIA, March 1994.

[4] H. Delingette. General object reconstruction based on simplex meshes. Technical Report 3111, INRIA, February 1997.

[5] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Computer Graphics*, pages 71–78. SIGGRAPH, 1992.

[6] Hugues Hoppe. *Surface Reconstruction from Unorganized Points.* PhD thesis, Department of Computer Science, University of Washington, June 1994.

[7] G. Turk and M Levoy. Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH*, volume 2, pages 311–318, Orlando, FL, 1994.

[8] Y. Sun, C. Dumont, and M. A. Abidi. Mesh–based integration of range and color images. In *Proceedings of SPIE'S 14th Annual International Symposium on AeroSense/Defense Sensing, Simulation and Controls*, Orlando, FL, April 2000.

[9] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[10] G. Wyvill, C. McPheeters, and B. Wyvill. Data structures for soft objects. *The Visual Computer*, 2(4):227–234, August 1986.

[11] Joseph O'Rourke. Polyhedra of minimal area as 3d object models. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 664–666, 1981.

[12] Remco Veltkamp. *Closed Object Boundaries from Scattered Points.* PhD thesis, Erasmus Universiteit, Rotterdam, The Netherlands, 1992.

[13] Herbert Edelsbrunner and Ernst P. Mucke. Three–dimensional alpha shapes. In In A. Kaufman and W. E. Lorensen, editors, *Proceedings of 1992 Workshop on Volume Visualization*, pages 75–82, October 1992.

[14] Chandrajit Bajaj, Fausto Bernardini, and Guoliang Xu. Reconstructing surfaces and functions on surfaces from unorganized 3d data. *Algorithmica*, (19):243–261, 1997.

[15] Remco Veltkamp. *In Lecture Notes in Computer Science 885.* Srpinger Verlag, 1994.

[16] Maria-Elena Algorri and Francis Schmitt. Surface reconstruction from unstructured 3d data. *Computer Graphics forum*, 15(1):47–60, 1996.

[17] R. T. Whitaker. A level-set approach to 3d reconstruction from range data. *International Journal of Computer Vision*, 29(3), October 1998.

[18] D. Terzopoulos and D. Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. *Trans. Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991.

[19] A. Johnson, R. Hoffman, J. Osborn, and M. Herbert. A system for semi-automatic modeling of complex environments. In *International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, May 1997.

[20] Enviromental restoration and waste management five year plan fiscal years 1994-1995. Technical Report DOE/5-00097P, US Department of Energy, 1993.

[21] L. Rosenblum. Virtual and augmented reality 2000. *IEEE Computer Graphics and Applications*, 20(1):38–39, January/February 2000.

[22] Fang, Lian, and D.C. Gossard. Fitting 3d curves to unorganized data points using deformable curves. In *CG International 92, Tokio, Japan*, June 1992.

[23] W. Cho, T. Maekawa, N.M.Patrikalakis, and J. Peraire. Topologically reliable approximation of trimmed polynomial surface patches. *Graphical Models and Image Processing*, 61(2):84–109, March 1999.

[24] A. Alrashdan, S. Motavalli, and B. Fallahi. Automatic segmentation of digitized data for reverse engineering applications. *IIE Transactions*, 32:59–69, 2000.

[25] O Faugeras. *Three-Dimensional Computer Vision—A Geometric Viewpoint.* MIT Press, 1993.

[26] Y. Zhang, Y. Sun, H. Sari-Sarraf, J.R. Price, and M.A. Abidi. Impact of intensity edge map on segmentation of noisy range images. In *SPIE's Electronic Imaging 2000*, San Jose, California, January 2000.

[27] L. Wong. An algorithm and system for finding the next best view in a 3d object modeling task. Master's thesis, Department of Electrical Engineering, The University of Tennessee Knoxville, April 99.

[28] H. Gray. Simulation of a tire inspection system. *Vision*, 16(1), First quarter 2000.

[29] S. Gibson B. Mirtich. A survey of deformable modeling in computer graphics. Technical Report TR-97-19, Mitsubichi Electric Research Laboratory, November 1997.

[30] M. Gardiner. The superellipse: A curve between the ellipse and the rectangle. *Scientific American*, (213):222–234, 1965.

[31] T. O'Donnell, A. Gupta, X. Fang, and T. Boult. The extruded generalized cylinder: A deformable model for object recovery. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 174–181, 1993.

[32] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331, 1988.

[33] A. Singh, D. Goldgof, and D. Terzopoulos. *Deformable Models in Medical Image Analysis.* IEEE Computer Society, 1998.

[34] Montagnat, Delingette, and Ayache. Representation, shape, topology and evolution of deformable surfaces. application to 3d medical image segmentation. Technical Report 3954, INRIA, May 2000.

[35] S Menet, P. Saint-Marc, and G. Medioni. *Artificial Intelligence and Computer Vision*, chapter B-Snakes: Implementation and Application to Stereo. Elsevier Science, 1991.

[36] S. Leitner, M.I., S. Lavallee, and P. Cinquin. *Curves and Surfaces*, chapter Dynamic Segmentation: Finding the Edge with Spline Snakes, pages 279–284. Academic Press, 1991.

[37] J. A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid mechanics, Computer Vision, and Materials Science.* Cambridge University Press, 1996.

[38] R. Malladi, J. Sethian, and B. Vemuri. Evolutionary fronts for topology-independent shape modeling and recovery. In J.-O. Eklundh, editor, *Third European Conference on Computer Vision*, pages 3–13. Springer-Verlag, 1994.

[39] M. Valiescu and D. Terzopoulos. Adaptive meshes and shells. In *International Conference on Computer Vision*, pages 829–832, Champaign, Illinois, June 1992.

[40] M. Bro-Nielsen. Active nets and cubes. Technical Report 94-13, Institute of Mathematical Modeling, Technical University of Denmark, 1994.

[41] A. Mangan and R. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, October-December 1999.

[42] A. McIvor, D. Penman, and P. Waltenberg. Simple surface segmentation. Technical report, Industrial REsearch Limited, 1998.

[43] K.L. Bowyer, M.J. Mirza, and G. Ganguly. The robust sequential estimator: A general approach and its application to surface organization in range data. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16:987–1001, October 1994.

[44] N.S. Raja and A. K. Jain. Obtaining generic parts from range images using multiview representation. *CVGIP: Image Understanding*, 1994.

[45] B. Parvin and G. Medioni. B-rep object description from multiple range views. *Internationla Journal of Computer Vision*, 1996.

[46] L.G. Roberts. *Optical and Electro-optical Information Processing*, chapter Machine perception of three-dimensional solids, pages 211–277. MIT Press, 1965.

[47] T.O. Binford. Visual perception by computer. In *IEEE Systems Science and Cybernetics Conference*, Miami, FL., 1971.

[48] R. A. Brooks. Symbolic reasoning among 3d models and 2d images. *Artificial Intelligence*, 17:205–244, 1981.

[49] D. Marr and H.K. Nishihara. Representation and recognition of three-dimensional shapes. In *Proceedings of the Royal Society of London, Series B*, volume 200, pages 269–294, 1978.

[50] I. Biederman. Recognition-by-components: A theory of human image undestanding. *Psychological Review*, 94:115–147, 1987.

[51] A.P. Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28:293–331, 1986.

[52] Ales Jaklic. *Contruction of CAD Models from Range Images.* Ph.d's thesis, University of Ljubljana, 1997.

[53] D.D. Hoffman and W.A. Richards. Parts of recognition. *Cognition*, 18:65–96, 1984.

[54] K. Wu and M.D. Levine. 3d part segmentation using simulated electrical charge distributions. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(11):1223–1235, November 1997.

[55] K. Yoshino, T. Kawashima, and Y. Aoki. Dynamic reconfiguration of active net structure. In *Proceedings of Asian Conference on computer Vision*, pages 159–162, 1993.

[56] F Preparata and M Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1986.

[57] C. Gourley. *Pattern Vector Based Reduction of Large Multimodel Data Sets for Fixed Rate Interactivity During Visualization of Multiresolution Models.* PhD thesis, The University of Tennessee Knoxville, November 1998.

[58] A. Mangan. Photo-realistic surface reconstruction. Master's thesis, The University of Tennessee Knoxville, May 1999.

[59] M.E. Mortenson. *Geometric Modeling.* Wiley Computer Publishing, 1997.

[60] P. Cignoni, C. Montani, and R. Scopigno. Metro: Measuring error on simplified surfaces. Technical report, Istituto di Elaborazione dell'Informazione-Consiglio Nazionale delle Ricerche, 1997.

[61] B. K. P. Horn. *Robot Vision.* McGraw-Hill, 1986.

Vita

Ernesto Lautaro Juarez Valdes was born in Mexico city, Mexico, in 1976. In 1978 he moved to the city of Zacatecas, Mexico. From 1992 to 1997 he attended the Autonomous University of Zacatecas obtaining the degree of Communications and Electronics Engineer. Ernesto enrolled in the MSEE program at the University of Tennessee Knoxville in 1998 where he is currently working as a Graduate Research Assistant in the Imaging, Robotics and Intelligent Systems Laboratory under the supervision of Dr. Mongi A. Abidi. He expects to graduate in August 2000 with a specialization in computer vision and image processing.