

Deformable Solids and Displacement Maps:  
A Multi-scale Technique  
for Model Recovery and Recognition

by

Stanley Edward Sclaroff

B.S., COMPUTER SCIENCE AND ENGLISH  
TUFTS UNIVERSITY, 1984

SUBMITTED TO THE MEDIA ARTS AND SCIENCES  
SECTION, SCHOOL OF ARCHITECTURE AND PLANNING IN PARTIAL  
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1991

© Massachusetts Institute of Technology 1991  
All Rights Reserved

Author .....

.....  
Media Arts and Sciences Section  
May 10, 1991

Certified by .....

.....  
Alex P. Pentland  
Associate Professor  
Computer Information and Design Technology  
Thesis Supervisor

Accepted by .....

.....  
Stephen Benton  
Chairperson

MASSACHUSETTS INSTITUTE  
OF TECHNOLOGY

Departmental Committee on Graduate Studies

JUL 23 1991

LIBRARIES

Rotch

**Deformable Solids and Displacement Maps:  
A Multi-scale Technique  
for Model Recovery and Recognition**

by

**Stanley Edward Sclaroff**

Submitted to the Media Arts and Sciences  
Section, School of Architecture and Planning  
on May 10, 1991, in partial fulfillment of the  
requirements for the degree of  
Master of Science

**Abstract**

In this thesis, we formulate a model recovery, recognition, and simulation framework which uses a combination of deformable implicit functions and displacement maps to describe solid objects. Modal deformations, also known as free vibration modes, are used to describe the overall shape of a solid, while displacement maps provide local and fine surface detail by offsetting the surface of the solid along its surface normals. Displacement maps are similar to gray-scale images and can therefore be encoded via standard image compression techniques.

We present an efficient, physically-based solution for recovering these deformable solids from collections of three-dimensional range data, surface normals, surface measurements, and silhouettes, or from two-dimensional contour data. Given a sufficient number of independent measurements, the fitting solution is overconstrained and unique except for rotational symmetries. We then present a physically-based object recognition method that allows simple, closed-form comparisons of recovered models. The performance of these methods is evaluated using both synthetic and real laser rangefinder data and two-dimensional image contours. Finally, we outline how these recovered models can be used efficiently in virtual experiments and simulations — the primary advantage of this approach over the standard polygon-based representations prevalent in computer graphics is that collision detection and dynamic simulation become simple and inexpensive even for complex shapes.

Thesis Supervisor: Alex P. Pentland

Title: Associate Professor

Computer Information and Design Technology

# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Approach . . . . .	7
1.2	Overview . . . . .	10
<b>2</b>	<b>Model Recovery using Deformation Modes</b>	<b>13</b>
2.1	The Representation . . . . .	15
2.2	Recovering 3-D Part Models From 3-D Sensor Measurements . . . . .	21
2.3	Determining Spring Attachment Points . . . . .	25
2.4	Examples of Recovering 3-D Models . . . . .	30
<b>3</b>	<b>Object Recognition</b>	<b>36</b>
3.1	Using Recovered Mode Values for Recognition . . . . .	37
3.2	Recovering and Recognizing Models . . . . .	39
<b>4</b>	<b>Recovery, Tracking, and Recognition from Contour</b>	<b>47</b>
4.1	Head Recognition from Contours . . . . .	47
4.2	3-D shape from 2-D contours . . . . .	48
4.3	Dynamic Estimation of Rigid and Non-Rigid Motion . . . . .	53
<b>5</b>	<b>Displacement Maps and Physical Simulation</b>	<b>59</b>
5.1	The Geometry: Generalized Implicit Functions . . . . .	60
5.2	Computing a Displacement Map . . . . .	64
5.3	Storage and Compression . . . . .	66
<b>6</b>	<b>Conclusion</b>	<b>71</b>
<b>A</b>	<b>Programming Shortcuts</b>	<b>79</b>

# List of Figures

2-1	Several of the lowest-frequency vibrations modes of a cylinder. . . . .	19
2-2	Some shapes produced by deforming a sphere using 30 modes. . . . .	22
2-3	Using Moment Method for Initial Model Placement . . . . .	28
2-4	Fitting geometric objects using sparse 3-D point data . . . . .	30
2-5	Fitting laser rangefinder data of a human face. . . . .	32
2-6	Segmenting range data of a human figure into parts . . . . .	33
2-7	Incremental fitting to range data of a human figure . . . . .	35
3-1	Eight heads used in our recognition experiment . . . . .	40
3-2	Recognizing faces from five different points of view. . . . .	41
3-3	How recognition degrades as a function of sensor noise . . . . .	43
3-4	Using $L_2$ distance to measure object similarity . . . . .	45
3-5	Using normalized dot product to measure object similarity . . . . .	46
4-1	Recognizing heads recovered from contours. . . . .	49
4-2	Using a 2-D image contour to recover a 3-D heart model . . . . .	50
4-3	Recovering pills from contours and center axes . . . . .	52
4-4	A heart's nonrigid motion recovered from contours . . . . .	57
5-1	Three frames from a physically-based animation. . . . .	64
5-2	Simulation using a Recovered Head . . . . .	68
5-3	Displacement map pyramid. . . . .	69
5-4	Using Epic to compress a recovered head's displacement map. . . . .	70

## **Acknowledgments**

I gratefully acknowledge the support and advice of my collaborators, Trevor Darrell, Bradley Horowitz, Jun Inagawa, and Sandy Pentland (also my thesis advisor) and the rest of the ThingWorld programming posse: Irfan Essa, Thad Starner, and Martin Friedmann. Special thanks to my thesis readers: Ted Adelson, Frank Ferrie, and Demetri Terzopoulos.

This research was made possible in part by the Rome Air Development Center (RADC) and the Defense Advanced Research Projects Agency (DARPA) under contract No. F30602-89-C-0022. We would like to thank the Harvard Face Archive, Cyberware Labs, and The Visual Computing Group for providing us with range data of human heads.

# Chapter 1

## Introduction

Recovering higher-level descriptions of images is critical in any machine vision or virtual world scenario. For robots to independently understand, explore, or manipulate their environments, they must be able to extract compact and meaningful representations of the images they gather via a camera. For virtual simulation, we would like to point a camera at a scene and automatically create a computer model — thus gathering the real world, allowing it to coexist with imaginary, computer generated objects. And finally, for image compression to be successful, it will be useful to build up powerful abstractions (i.e., 3-D models) of image data to get the bit rates down to desired levels. The success of each of these depends on how expressive and stable our representation is.

To date, nearly all successful high-level computer vision systems have been unsuccessful at developing a representation which is both general enough and stable enough to be useful in coding a variety of images. Useful computer vision systems have been achieved, but they restrict themselves to a very small variety of shapes — for instance, some parts moving on a factory's conveyor belt — or they assume that they will only encounter certain objects, for which a precise computer model is already known. Such systems solve difficult machine vision problems and are useful within their restricted domains, but they are not useful when it comes to general image coding for high definition video transmission, or when it comes to AI-style image understanding for autonomous creatures who can learn and adapt to new experiences.

There are two contradictory problems which need to be solved before a successful and general machine vision system can be achieved: representation for machine vision, and representation for model recovery and physical simulation. If a computer is to recognize and reason about objects, then it needs a high-level representation which somehow reduces the confounding volume of surface information down to a manageable abstraction. If, however, a computer is to interact with the world, predict behavior, simulate reality, or encode images, then it needs a representation which is general and expressive enough to accurately represent localized surface details and physical behavior. Up until recently, there has been a clear division between representations which provide the necessary abstraction — parametric models — and those which provide the necessary expressiveness — finite element-based, deformable models. Clearly, we would better off with a hybrid method, one that offers the best of both representations.

## 1.1 Approach

Such a hybrid is not easy to achieve. This is because lurking under all of this is the problem of choosing an appropriate scale. To obtain an abstraction, we reduce shape information down to its fundamental core, but in so doing, we sacrifice local surface detail. A multiscale method is needed, one which allows for choosing the level of detail needed for comparison, simulation, or compression. A standard image compression technique for this is pyramid image encoding. This thesis will explore a similar multiresolution frequency coding method for representing three-dimensional shape. The resulting hybrid representation will be a physically-based, parametric model — based in part on Terzopoulos's work with deformable models and on Pentland's work with modal analysis — and supplemented with a *displacement map* which will be used to store local surface detail.

The notion of employing physical constraints in shape modeling has been suggested by many authors, however, the seminal paper by Terzopoulos, Witkin, and Kass [47], who obtained 3-D models by fitting simulated rubbery sheets and tubes, has focused attention on

modeling methods that draw on the mathematics used for simulating the dynamics of real objects. One motivation for using such physically-based representations is that vision is often concerned with estimating changes in position, orientation, and shape, quantities that models such as the finite element method (FEM) accurately describe. Another motivation is that by allowing the user to specify forces that are a function of sensor measurements, the intrinsic dynamic behavior of techniques such as the FEM can be used to solve fitting, interpolation, or correspondence problems.

In this implementation, computer simulated rubber sheets are “shrink-wrapped” over the range data. This rubber sheet is actually a spline, finite element mesh that simulates the reaction of such a sheet to simulated forces — the data points exert forces which essentially attract the sheet. The spline sheets are very good at capturing local detail while staying somewhat stable to noise. And since they are force-based, the resulting deformable models can participate in physical simulations.

Unfortunately, the representation has the drawback that it is not easy to compare two objects. For any mesh-based representation, the only general method for determining if two surfaces are equivalent is to generate a number of sample points at corresponding positions on the two surfaces, and observe the distances between the two sets of sample points. Not only is this a clumsy and costly way to determine if two surfaces are equivalent, but when the two surfaces have very different parameterizations it can also be quite difficult to generate sample points at “corresponding locations” on the two surfaces.

Pentland addressed this problem by adopting an approach based on *modal analysis* [31]. In this method, the standard FEM computations are simplified by posing the dynamic equations in terms of the equations’ eigenvectors. These eigenvectors are known as the object’s *deformation modes*, and together form a frequency-ordered orthonormal basis set for shape. In this representation, objects are described using the force/process metaphor of modeling clay: shape is thought of as the result of pushing, pinching, and pulling on a lump of elastic material.

The modal representation, assuming that all modes are employed, decouples the degrees of freedom within the non-rigid dynamic system, but it does not by itself reduce the total



number of degrees of freedom. Thus a complete modal representation suffers from the same non-uniqueness problems as all of the other representations.

The obvious solution to the problem of non-uniqueness is to discard a enough of the high-frequency modes that we can obtain an overconstrained estimate of shape. Use of a reduced-basis modal representation results in a *unique* representation of shape because the modes (eigenvectors) form a frequency-ordered orthonormal basis set similar to a 3-D Fourier decomposition. Just as with the Fourier decomposition, reducing the number of sample points does not change the low-frequency components, assuming regular subsampling. Similarly, local sampling and measurement noise primarily affect the high-frequency modes rather than the low-frequency modes.

By discarding the higher-order modes, we get a parametric deformable model representation. The parameters are physically meaningful, and correspond to how much bending, tapering, pinching, etc., is needed to represent an object. This is not enough to achieve the goal of a hybrid representation: there is still a trade-off between abstraction and expressiveness, since local shape information (attributed to high frequency modes) is discarded.

To get around this problem, we propose a model recovery framework which incorporates *displacement maps* to add fine detail to a solid model described by deformable analytic implicit functions. Such maps store surface displacements as a coded relief map, where brighter shades signify positive offsets (peaks), and darker shades signify negative offsets (troughs). Displacement maps are similar to the offsets used for B-splines in that they allow for local and fine-detail control over the shape of the surface. In the context of solid models, displacement maps function by offsetting the analytic surface of the solid along its surface normals. Displacement maps will provide a two-tiered method for describing an object: first the system will compute a low-order modal fit, and then it will do a second pass in which the residual error is stored as a displacement map.

This detail map methodology is especially appealing because the map can be stored like an intensity image. The most useful side-effects of this are that displacement maps can be compressed via standard image compression techniques, and that the representation is

especially well-suited for physical simulation. Since the resulting part representation is still an implicit function (displacement maps merely offset the underlying deformed function) intersection and inside/outside measurements are easy to compute.

Thus, as part of this thesis, we will explore some of these aspects of the modal deformation and displacement map combination. In addition, we will outline a method for making a possible “physical connection” between the underlying deformable model and the displacement map.

## 1.2 Overview

This thesis is built around a core representation: deformable implicit functions supplemented by displacement maps. The purpose will be not only to review and introduce methods for use in model recovery, recognition, and physical simulation, but also to present useful and interesting results of experiments and examples for applications in computer vision and computer graphics. The emphasis (and therefore structure) of the thesis will be split equally between theory and application. Besides this introductory chapter and a short concluding chapter, there are 4 other chapters in this thesis. The chapters are:

**Chapter 2:** In Chapter 2, we first review the previous work and motivation behind physically-based models in computer vision. In particular, we review the ideas behind *modal analysis*, what deformation modes look like, and why they are useful in model recovery. As part of this background, the chapter provides a brief review of the Finite Element Method (FEM) and modal analysis. We then present an efficient, physically-based solution for recovering these deformable solids from collections of three-dimensional surface measurements. Given a sufficient number of independent measurements, the fitting solution is overconstrained and unique except for rotational symmetries. The concept of displacement maps is introduced in this chapter, and will be described in more depth in Chapter 5. Examples of model recovery are given using artificial (no noise) range data, laser range data, and integrating data taken from many views of an object.

**Chapter 3:** The model recovery method described in Chapters 2 produces a vector of *mode amplitudes*; these are parameters which describe how much of each modal deformation was used in recovering the model. In Chapter 3, we describe a simple, closed-form technique for comparing recovered models. The technique's accuracy, graceful degradation when noise is added to sensor measurements, and plausibility (i.e., If two objects are similar to human observers, are they similar in our recognition system?) are evaluated and discussed using 3-D range-data of human heads, and feature points from geometric objects.

**Chapter 4:** In Chapter 4, we show how to use the techniques described in Chapters 2 and 3 for recovering, recognizing, and tracking deformable models from two and three-dimensional contours. By using a variant of *weighted least squares*, we can enforce volume or symmetry constraints; or by changing weights, we can reflect the certainty of our measurements. We also describe a simple method to include contour normals in model recovery. The examples include head contours taken from range data, and contours taken from an X-ray motion sequence of a human heart beating. In the heart example, we describe a simple recursive filter method which is used to improve the stability of the non-rigid motion recovery.

**Chapter 5:** This chapter describes our representation and its applications in computer graphics terms. Our method can be thought of as generalizing implicit functions by use of polynomial deformations and displacement maps. We then outline a physical simulation technique based on *modal dynamics*, and include example simulations. Also as part of this chapter, we describe how to compute, store and display displacement-mapped solids. Displacement map interpolation methods are outlined and then demonstrated. Since displacement maps are stored as large floating point image arrays (usually 100x100 or 256x256), there is some concern about the amount of space they occupy in memory or on disk. To get around this problem, we propose using a pyramid-based compression and interpolation scheme which significantly reduces the amount of space needed to store a displacement map.

This thesis presents some work which also appears in journals and conference proceedings. Chapters 2 and 3, were taken in part from [40, 38, 14]. Parts of Chapter 4 appear in an article

---

submitted to the IEEE Workshop on Visual Motion, to be held in Princeton later this year [37].

And finally, parts of Chapters 5 will appear in a SIGGRAPH paper, [41].

## Chapter 2

# Model Recovery using Deformation

## Modes

Vision research has a long tradition of trying to go from collections of low-level measurements to higher-level “part” descriptions such as generalized cylinders [8, 28, 29], deformed superquadrics [33, 34, 42, 10], or geons [7], and then of attempting to perform object recognition. The general idea is to use part-level modeling primitives to bridge the gap between image features (points, edges, or corners) and the symbolic, parts-and-connections descriptions useful for recognition and reasoning [51].

Recently, several researchers have successfully addressed the first of these problems — that of recovering part descriptions — using deformable models. There have been two classes of such deformable models: those based on parametric solid modeling primitives, such as our own work using superquadrics [33], and those based on mesh-like surface models, such as employed by Terzopoulos, Witkin, and Kass [47]. In the case of parametric modeling, fitting has been performed using the modeling primitive’s “inside-outside” function [34, 42, 10, 31], while in the mesh surface models a physical-motivated energy function has been employed [47].

The description of shape by use of orthogonal parameters has the advantage that it can produce a unique, compact description that is well-suited for recognition and database search,

but has the disadvantage that it does not have enough degrees of freedom to account for fine surface details. The deformable mesh approach, in contrast, is very good for describing shape details, but produces descriptions that are neither unique nor compact, and consequently cannot be used for recognition or database search without additional layers of processing. Both of these approaches share the disadvantage that they are relatively slow, requiring dozens of iterations in the case of the parametric formulation, and up to hundreds of iterations in the case of the physically-based mesh formulation.

We have addressed these problems by adopting an approach based on the finite element method (FEM) and solid modeling. This approach provides both the expressiveness and convenience of the physically-based mesh formulation, but in addition can provide much greater accuracy at physical simulation. Thus it is possible to use the models we recover from range data to accurately simulate particular materials and situations for purposes of prediction, visualization, planning, and so forth [39, 31].

More importantly, however, we have been able to obtain develop a formulation whose degrees of freedom are orthogonal, and thus *decoupled*, by posing the dynamic equations in terms of the FEM equations' eigenvectors. These eigenvectors are known as the object's *free vibration* or *deformation modes*, and together form a frequency-ordered orthonormal basis set analogous to the Fourier transform.

By decoupling the degrees of freedom we achieve substantial advantages:

- The fitting problem has a simple, efficient, closed-form solution.
- The model's intrinsic complexity can be adjusted to match the number of degrees of freedom in the data measurements, so that the solution can always be made overconstrained.
- When overconstrained, the solution is unique, except for rotational symmetries and degenerate conditions. Thus the solution is well-suited for recognition and database tasks.

The plan of this chapter is to first review our representation. We will then demonstrate its use in obtaining closed-form solutions to the shape recovery problem. This will prepare us for

Chapter 3, where we will demonstrate how closed-form comparison of different object models can be used to obtain accurate object recognition from range data.

## 2.1 The Representation

Our representation describes objects using the force-and-process metaphor of modeling clay: shape is thought of as the result of pushing, pinching, and pulling on a lump of elastic material such as clay [33, 39, 31]. The mathematical formulation is based on the finite element method (FEM), which is the standard engineering technique for simulating the dynamic behavior of an object.

The notion of employing physical constraints in shape modeling has been suggested by many authors [27, 21, 33]; however, the seminal paper by Terzopoulos, Witkin, and Kass [47], who obtained 3-D models by fitting simulated rubbery sheets and tubes, has focused attention on modeling methods that draw on the mathematics used for simulating the dynamics of real objects. One motivation for using such physically-based representations is that vision is often concerned with estimating changes in position, orientation, and shape, quantities that models such as the FEM accurately describe. Another motivation is that by allowing the user to specify forces that are a function of sensor measurements, the intrinsic dynamic behavior of techniques such as the FEM can be used to solve fitting, interpolation, or correspondence problems.

In the FEM interpolation functions are developed that allow continuous material properties, such as mass and stiffness, to be integrated across the region of interest. Note that this is quite different from the finite difference schemes commonly used in computer vision — as is explained in [23] and [38] — although the resulting equations are quite similar. One major difference between the FEM and the finite difference schemes is that the FEM provides an analytic characterization of the surface between nodes or pixels, whereas finite difference methods do not. All of the results presented in this paper will be applicable to both the finite difference and finite element formulations.

Having formulated the appropriate FEM integrals, they are then combined into a description

in terms of discrete *nodal points*. Energy functionals are then formulated in terms of nodal displacements  $\mathbf{U}$ , and the resulting set of simultaneous equations is iterated to solve for the nodal displacements as a function of impinging loads  $\mathbf{R}$ :

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{C}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{R} \quad (2.1)$$

where  $\mathbf{U}$  is a  $3n \times 1$  vector of the  $(\Delta x, \Delta y, \Delta z)$  displacements of the  $n$  nodal points relative to the object's center of mass,  $\mathbf{M}$ ,  $\mathbf{C}$  and  $\mathbf{K}$  are  $3n$  by  $3n$  matrices describing the mass, damping, and material stiffness between each point within the body, and  $\mathbf{R}$  is a  $3n \times 1$  vector describing the  $x$ ,  $y$ , and  $z$  components of the forces acting on the nodes.

Equation 2.1 is known as the *governing equation* in the finite element method, and may be interpreted as assigning a certain mass to each nodal point and a certain material stiffness between nodal points, with damping being accounted for by dashpots attached between the nodal points. Inertial and centrifugal effects are accounted for by adding appropriate off-diagonal terms to the mass matrix.

When a constant load is applied to a body it will, over time, come to an equilibrium condition described by

$$\mathbf{K}\mathbf{U} = \mathbf{R} \quad (2.2)$$

This equation is known as the *equilibrium governing equation*. The solution of the equilibrium equation for the nodal displacements  $\mathbf{U}$  (and thus of the analytic surface interpolation functions) is the most common objective of finite element analyses. In shape modeling, sensor measurements are used to define virtual forces which deform the object to fit the data points. The equilibrium displacements  $\mathbf{U}$  constitute the recovered shape.

The most obvious drawback of the all physically-based methods are their large computational expense. These methods require roughly  $3nm_k$  operations per time step, where  $3n$  is the order of the stiffness matrix and  $m_k$  is its half bandwidth.<sup>1</sup> Normally  $3n$  time steps are required to obtain an equilibrium solution. For a full 3-D model, where typically  $m_k \approx 3n/2$ ,

---

<sup>1</sup>See Bathe[23] Appendix A.2.2 for complete discussion on bandwidth of a stiffness matrix.



the computational cost scales as  $O(n^3)$ . Because of this poor scaling behavior sometimes equations are discarded in order to obtain sparse banded matrices (for example, in [47] those equations corresponding to internal nodes were discarded). In this case the computational expense is reduced to “only”  $O(n^2 m_k)$ .

A related drawback in vision applications is that the number of description parameters is often greater than the number of sensor measurements, necessitating the use of heuristics such as symmetry and smoothness. This also results in non-unique and unstable descriptions, with the consequence that it is normally difficult to determine whether or not two models are equivalent.

Perhaps the most important problem with using physically-based models for vision, however, is that all of the degrees of freedom are coupled together. Thus closed-form solutions are impossible, and solutions to the inverse problems encountered in vision are very difficult.

Thus there is a need for a method which transforms Equation 2.1 into a form which is not only less costly, but also allows closed-form solution. Since the number of operations is proportional to the half bandwidth  $m_k$  of the stiffness matrix, a reduction in  $m_k$  will greatly reduce the cost of step-by-step solution. Moreover, if we can actually diagonalize the system of equations, then the degrees of freedom will become uncoupled and we will be able to find closed-form solutions. To accomplish this goal, we utilize a FEM technique known as *Modal Analysis*; in the remainder of this section we develop this method along the lines of Bathe[23].

To diagonalize the system of equations, a linear transformation of the nodal point displacements  $\mathbf{U}$  can be used:

$$\mathbf{U} = \mathbf{P}\tilde{\mathbf{U}} \quad (2.3)$$

where  $\mathbf{P}$  is a square orthogonal transformation matrix and  $\tilde{\mathbf{U}}$  is a vector of generalized displacements. Substituting Equation 2.3 into Equation 2.1 and premultiplying by  $\mathbf{P}^T$  yields:

$$\tilde{\mathbf{M}}\ddot{\tilde{\mathbf{U}}} + \tilde{\mathbf{C}}\dot{\tilde{\mathbf{U}}} + \tilde{\mathbf{K}}\tilde{\mathbf{U}} = \tilde{\mathbf{R}} \quad (2.4)$$

where

$$\tilde{\mathbf{M}} = \mathbf{P}^T \mathbf{M} \mathbf{P}; \quad \tilde{\mathbf{C}} = \mathbf{P}^T \mathbf{C} \mathbf{P}; \quad \tilde{\mathbf{K}} = \mathbf{P}^T \mathbf{K} \mathbf{P}; \quad \tilde{\mathbf{R}} = \mathbf{P}^T \mathbf{R} \quad (2.5)$$

With this transformation of basis set a new system of stiffness, mass and damping matrices can be obtained which has a smaller bandwidth than the original system.

The optimal transformation matrix  $\mathbf{P}$  is derived from the free vibration modes of the equilibrium equation. Beginning with the governing equation, an eigenvalue problem can be derived

$$\mathbf{K} \phi_i = \omega_i^2 \mathbf{M} \phi_i \quad (2.6)$$

which will determine an optimal transformation basis set.

The eigenvalue problem in Equation 2.6 yields  $3n$  eigensolutions

$$(\omega_1^2, \phi_1), (\omega_2^2, \phi_2), \dots, (\omega_n^2, \phi_{3n})$$

where all the eigenvectors are  $\mathbf{M}$ -orthonormalized. Hence

$$\phi_i^T \mathbf{M} \phi_j \begin{cases} = 1; & i = j \\ = 0; & i \neq j \end{cases} \quad (2.7)$$

and

$$0 \leq \omega_1^2 \leq \omega_2^2 \leq \omega_3^2 \leq \dots \leq \omega_{3n}^2 \quad (2.8)$$

The eigenvector  $\phi_i$  is called the  $i^{\text{th}}$  mode's *shape vector* and  $\omega_i$  is the corresponding frequency of vibration. Each eigenvector  $\phi_i$  consists of the  $(x, y, z)$  displacements for each node, that is, the  $3j - 2$ ,  $3j - 1$ , and  $3j$  elements are the  $x$ ,  $y$ , and  $z$  displacements for node  $j$ ,  $1 \leq j \leq n$ .

The lowest frequency modes are always the rigid-body modes of translation and rotation. The eigenvector corresponding to x-axis translation, for instance, has ones for each node's x-axis displacement element, with all other elements being zero. Rotational motion is linearized, so that nodes on the opposite sides of the body have opposite directions of displacement.

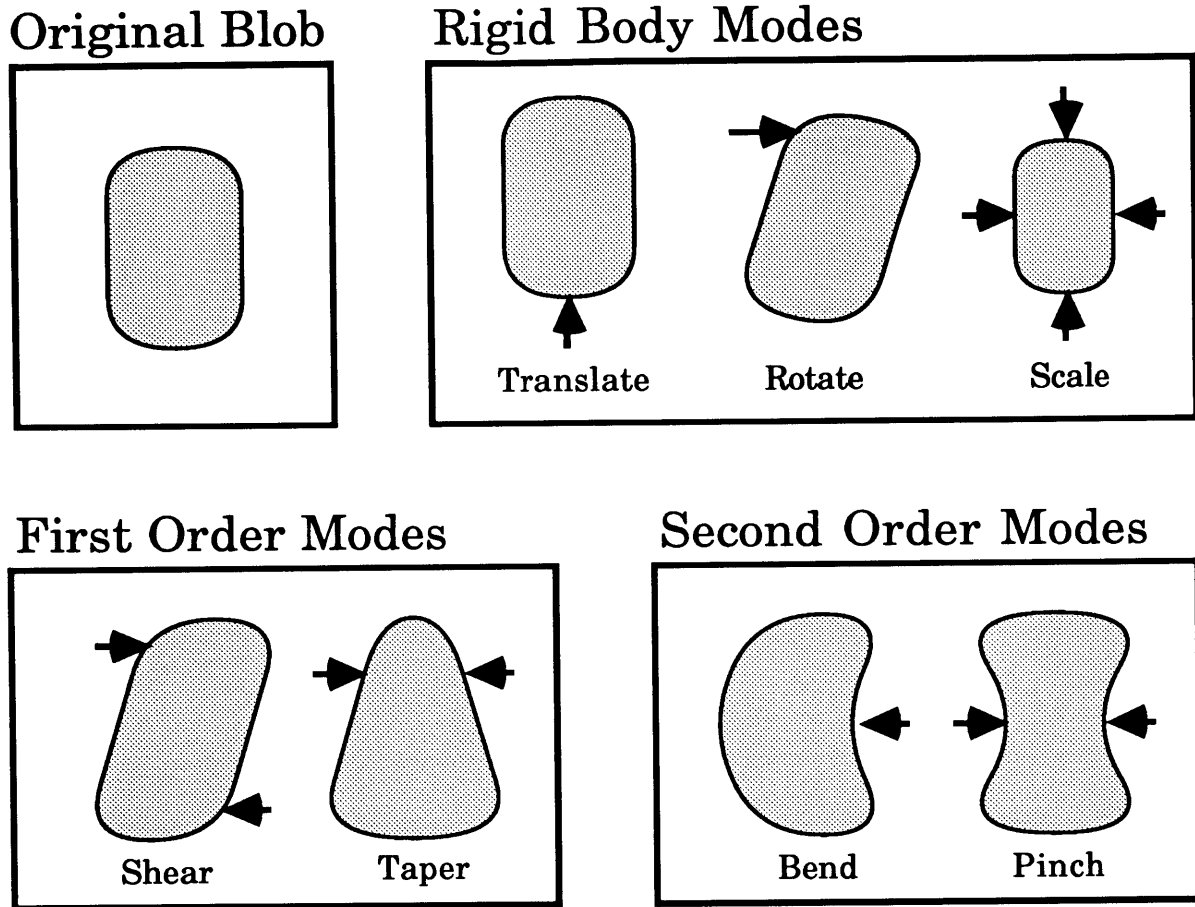


Figure 2-1: Several of the lowest-frequency vibrations modes of a cylinder.

The next-lowest frequency modes are smooth, whole-body deformations that leave the center of mass and rotation fixed. That is, the  $(x, y, z)$  displacements of the nodes are a low-order function of the node's position, and the nodal displacements balance out to zero net translational and rotational motion. Compact bodies (simple solids whose dimensions are within the same order of magnitude) normally have low-order modes that are similar to those shown in Figure 2-1. Bodies with very dissimilar dimensions, or which have holes, etc., can have very complex low-frequency modes.

Using these modes we can define a transformation matrix  $\Phi$ , which has for its columns the

eigenvectors  $\phi_i$ , and a diagonal matrix  $\Omega^2$ , with the eigenvalues  $\omega_i^2$  on its diagonal:

$$\Phi = [\phi_1, \phi_2, \phi_3, \dots, \phi_{3n}] \quad (2.9)$$

$$\Omega^2 = \begin{bmatrix} \omega_1^2 & & & & & \\ & \omega_2^2 & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & & \omega_{3n}^2 \end{bmatrix} \quad (2.10)$$

Using Equation 2.10, Equation 2.6 can now be written as:

$$\mathbf{K}\Phi = \mathbf{M}\Phi\Omega^2 \quad (2.11)$$

and since the eigenvectors are  $\mathbf{M}$ -orthonormal:

$$\begin{aligned} \Phi^T \mathbf{K} \Phi &= \Omega^2 \\ \Phi^T \mathbf{M} \Phi &= \mathbf{I} \end{aligned} \quad (2.12)$$

From the above formulations it becomes apparent that matrix  $\Phi$  is the optimal transformation matrix  $\mathbf{P}$  for systems in which damping effects are negligible.

To also diagonalize the damping matrix  $\mathbf{C}$  with the same transform,  $\mathbf{C}$  must be restricted to a special form. The normal assumption is that the damping matrix is constructed by using the *Caughey series* [23],

$$\mathbf{C} = \mathbf{M} \sum_{k=0}^{p-1} a_k [\mathbf{M}^{-1} \mathbf{K}]^k \quad (2.13)$$

Restriction to this form is equivalent to the assumption that damping, which describes the overall energy dissipation during the system response, is proportional to system response. For  $p \leq 2$ , Equation 2.13 reduces to *Rayleigh damping* ( $\mathbf{C} = a_0 \mathbf{M} + a_1 \mathbf{K}$ ), and  $\mathbf{C}$  is diagonalized by  $\Phi$ . Rayleigh damping is the most common type of damping used in finite element analysis

[23].

Under the assumption of Rayleigh damping the general governing equation, given by Equation 2.4, is reduced to

$$\ddot{\tilde{\mathbf{U}}} + \tilde{\mathbf{C}}\dot{\tilde{\mathbf{U}}} + \tilde{\mathbf{U}} = {}^T\mathbf{R}(t) \quad (2.14)$$

where  $\tilde{\mathbf{C}} = a_0\mathbf{I} + a_1\boldsymbol{\Omega}^2$ , or equivalently to  $3n$  independent and individual equations of the form

$$\ddot{u}_i(t) + \tilde{c}_i\dot{u}_i(t) + \omega_i^2 u_i(t) = \tilde{r}_i(t) \quad (2.15)$$

where  $\tilde{c}_i$  are modal damping parameters and  $\tilde{r}_i(t) = \phi_i^T \mathbf{R}(t)$  for  $i = 1, 2, 3, \dots, 3n$ . Therefore the matrix  $\Phi$  is the optimal transformation for both damped and undamped systems, given that Rayleigh damping is assumed.

In summary, then, it has been shown that the general finite element governing equation is decoupled when using a transformation matrix  $\mathbf{P}$  whose columns are the free vibration mode shapes of the FEM system [23, 39, 31]. These decoupled equations may then be integrated numerically (see [39]) or solved in closed-form for any future time  $t$  by use of a *Duhamel integral* (see [31]).

## 2.2 Recovering 3-D Part Models From 3-D Sensor Measurements

Let us assume that we are given  $m$  three-dimensional sensor measurements (in the global coordinate system) that originate from the surface of a single object

$$\mathbf{X}^w = [x_1^w, y_1^w, z_1^w, \dots, x_m^w, y_m^w, z_m^w]^T \quad (2.16)$$

Following Terzopoulos *et al.*, we then attach virtual springs between these sensor measurement points and particular nodes on our deformable model. This defines an equilibrium equation whose solution  $\mathbf{U}$  is the desired fit to the sensor data. Consequently, for  $m$  nodes with

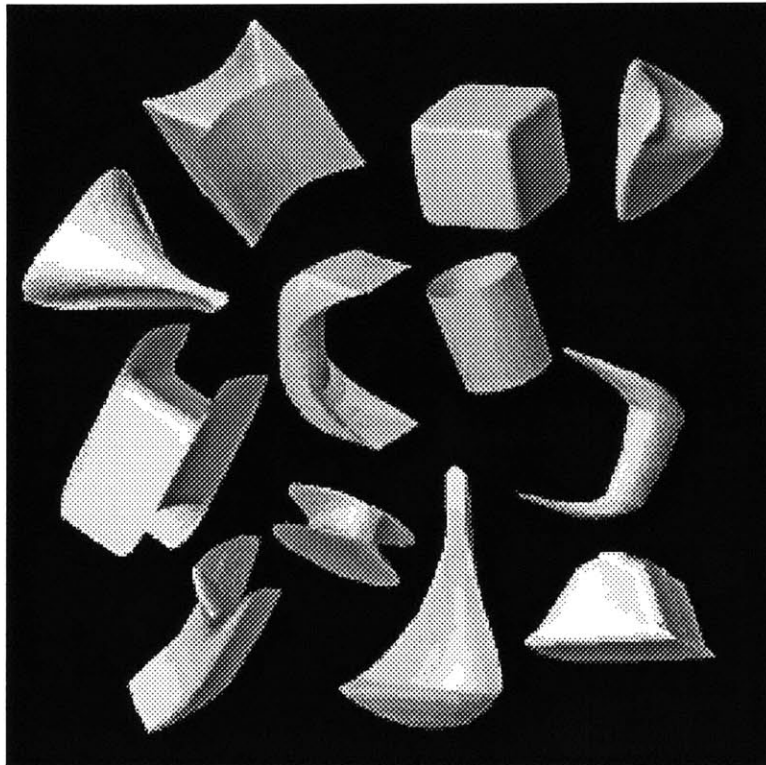


Figure 2-2: A sampling of shapes produced by elastically deforming an initial spherical shape using its 30 lowest-frequency deformation modes. As can be seen, a wide range non-rigid motions and their resulting shapes can be described by relatively few deformation modes.

---

corresponding sensor measurements, we can calculate the virtual loads  $\mathbf{R}$  exerted on the undeformed object while fitting it to the sensor measurements. For node  $k$  these loads are simply

$$[r_{3k}, r_{3k+1}, r_{3k+2}]^T = [x_k^w, y_k^w, z_k^w]^T - [x_k, y_k, z_k]^T \quad (2.17)$$

where

$$\mathbf{X} = [x_1, y_1, z_1, \dots, x_n, y_n, z_n]^T \quad (2.18)$$

are the nodal coordinates of the undeformed object in the object's coordinate frame. When sensor measurements do not correspond exactly with existing nodes, the loads can be distributed to surrounding nodes using the interpolation functions  $\mathbf{H}$  used to define the finite element model, as described in [23].

Thus to fit a deformable solid to the measured data we solve the following equilibrium equation:

$$\mathbf{K}\mathbf{U} = \mathbf{R} \quad (2.19)$$

where the loads  $\mathbf{R}$  are as above, the material stiffness matrix  $\mathbf{K}$  is as described above and in [15], and the equilibrium displacements  $\mathbf{U}$  are to be solved for. The solution to the fitting problem is simply

$$\mathbf{U} = \mathbf{K}^{-1}\mathbf{R} \quad (2.20)$$

The difficulty in calculating this solution is the large dimensionality of  $\mathbf{K}$ , so that iterative solution techniques are normally employed.

However a closed-form solution is available simply by converting this equation to the modal coordinate system. This is accomplished by substituting  $\mathbf{U} = \Phi\tilde{\mathbf{U}}$  and premultiplying by  $\Phi^T$ , so that the equilibrium equation becomes

$$\Phi^T\mathbf{K}\Phi\tilde{\mathbf{U}} = \Phi^T\mathbf{R} \quad (2.21)$$

or equivalently

$$\tilde{\mathbf{K}}\tilde{\mathbf{U}} = \tilde{\mathbf{R}} \quad (2.22)$$

where  $\tilde{\mathbf{R}} = \Phi^T \mathbf{R}$  and  $\tilde{\mathbf{K}} = \Phi^T \mathbf{K} \Phi$  is a *diagonal* matrix (see Equation 2.12). Note that the calculation of  $\Phi$  needs to be performed only once as a precomputation, and then stored for all future applications. Further, it is normally not desirable to use all of the eigenvectors (as explained in the next section), so that the  $\Phi$  matrix remains of manageable size even when using large numbers of nodes. In our implementation  $\Phi$  is normally a  $30 \times 3n$  matrix, where  $n$  is the number of nodes.

The solution to the fitting problem, therefore, is obtained by inverting the diagonal matrix  $\tilde{\mathbf{K}}$ :

$$\tilde{\mathbf{U}} = \tilde{\mathbf{K}}^{-1} \tilde{\mathbf{R}} \quad (2.23)$$

Note, however, that as this formulation is posed in the object's coordinate system the rigid body modes have zero eigenvalues, and must therefore be solved for separately by setting  $\tilde{u}_i = \tilde{r}_i$ ,  $1 \leq i \leq 6$ . The complete solution may be written in the original nodal coordinate system, as follows

$$\mathbf{U} = \Phi(\tilde{\mathbf{K}} + \mathbf{I}_6)^{-1} \Phi^T \mathbf{R} \quad (2.24)$$

where  $\mathbf{I}_6$  is a matrix whose first six diagonal elements are ones, and remaining elements are zero.<sup>2</sup>

The major difficulty in calculating this solution occurs when there are fewer degrees of freedom in sensor measurements than in the nodal positions — as is normally the case in computer vision applications. Previous researchers have suggested adopting heuristics such as smoothness and symmetry to obtain a well-behaved solution; however in many cases the observed objects are neither smooth nor symmetric, and so an alternative method is desirable.

We believe that a better, and certainly simpler, method is to discard some of the high-frequency modes, so that the number of degrees of freedom in  $\tilde{\mathbf{U}}$  is equal to or less than the number of degrees of freedom in the sensor measurements. To accomplish this, one simply row and column reduces  $\tilde{\mathbf{K}}$ , and column reduces  $\Phi$  so that their rank is less than or equal to the number of available sensor measurement degrees of freedom. The motivation for this strategy

---

<sup>2</sup>Inclusion of the matrix  $\mathbf{I}_6$  into Equation 2.24 may also be interpreted as adding an external force that constrains the solution to have no residual translational or rotational stresses.



is that:

- When there are fewer degrees of freedom in the sensor measurements than in the model, the high-frequency modes cannot in any sense be accurate, as there is insufficient data to constrain them. Their value primarily reflects the smoothness heuristic employed.
- While the high-frequency modes will not contain information, they are the *dominant* factor determining the cost of the solution, as they are both numerous and require the use of very small time steps [39].
- In any case, high-frequency modes typically have very little energy, and even less effect on the overall object shape. This is because (for a given excitatory energy) the displacement amplitude for each mode is *inversely* proportional to the *square* of the mode's resonance frequency, and because damping is proportional to a mode's frequency. The consequence of these combined effects is that high-frequency modes generally have very little amplitude. Indeed, in structural analysis of airframes and buildings it is standard practice to discard such high-frequency modes.

Perhaps the most interesting consequence of discarding some of the high-frequency modes, however, is that it allows Equation 2.24 to provide a generically *overconstrained* estimate of object shape. Note that discarding high-frequency modes is **not** equivalent to a smoothness assumption, as sharp corners, creases, etc., can still be obtained. What we cannot do with a reduced-basis modal representation is place many creases or spikes close together.

## 2.3 Determining Spring Attachment Points

Attaching a virtual spring between a data measurement and a deformable object implicitly specifies a correspondence between some of the model's nodes and the sensor measurements. In most situations this correspondence is not given, and so must be determined in parallel with the equilibrium solution. In our experience this attachment problem is the most problematic aspect of the physically-based modeling paradigm. This is not surprising, however, as the

attachment problem is similar to the correspondence problems found in many other vision applications.

Our approach to the spring attachment problem is similar to that adopted by other researchers [42, 47]. Given a segmentation of the data into objects or “blobs,” the first step is to define an ellipsoidal coordinate system by examination of the data’s moments of inertia. Data points are then projected onto this ellipsoid in order to determine the spring attachment points. We will describe only the simplest form of this method, one that does not adjust for uneven sampling of the data, as the general approach is well known.

### 2.3.1 Central Moments of Inertia

The center of the object is taken to be its center of gravity, except that because we see only the front-facing surface of the object we must employ a symmetry assumption to determine the  $z$  coordinate. After the center of gravity is found, we then determine the object’s orientation, which is assumed to be the same as its axes of inertia. The inertia matrix for sample range data consists of the moments of inertia and the cross products of inertia:

$$\mathbf{I} = \begin{pmatrix} \int (y^2 + z^2) dm & -\int xy dm & -\int xz dm \\ -\int xy dm & \int (x^2 + z^2) dm & -\int yz dm \\ -\int xz dm & -\int yz dm & \int (x^2 + y^2) dm \end{pmatrix} \quad (2.25)$$

Since we are integrating over sampled data for the solid, the integrals become summations. In order to make this simplification, we assume that the data points are more or less uniformly sampled, and that each point has the same mass. With this simplification, the matrix becomes:

$$I = \begin{pmatrix} \sum_{i=1}^n (y_i^2 + z_i^2) & \sum_{i=1}^n -x_i y_i & \sum_{i=1}^n -x_i z_i \\ \sum_{i=1}^n -x_i y_i & \sum_{i=1}^n (x_i^2 + z_i^2) & \sum_{i=1}^n -y_i z_i \\ \sum_{i=1}^n -x_i z_i & \sum_{i=1}^n -y_i z_i & \sum_{i=1}^n (x_i^2 + y_i^2) \end{pmatrix} \quad (2.26)$$

The eigenvectors of this matrix are the axes of inertia for the data points, and thus

correspond roughly with the axes of inertia for the object. The eigenvector with the smallest eigenvalue corresponds to the most elongated axis of the body, and the eigenvector with the largest eigenvalue corresponds to the shortest axis. Finally, to obtain estimates of the x, y, and z dimensions, we inverse transform the sample points (using the rotation matrix built from the eigenvectors obtained in the previous step), measure the x, y, and z range, and finally double the size estimate along the view direction.

Figure 2-3 shows the results of applying this technique to points sparsely sampled from a box. Notice how the moment method can produce a very good initial estimate of position, orientation and size if sensor data is sampled from all sides of the object, and how — if the data is only take from the front face, or is nonuniformly sampled — partial data can produce an approximation which can vary in accuracy. The top row shows the front view (a) and orthogonal side view (b) of a box and its actual coordinate axes. The middle row (c,d) shows the initial position, size, and orientation calculated via the moment method (shown as solid lines) for a small collection of box surface points. The bottom row (e,f) shows the same orientation information calculated by using only the surface points visible in (a) as input to the moment calculations.

One way to improve the initial estimate would be to push the center of gravity back along the viewing axis. Another is to allow variable point masses when uniform sampling is not available (by uniform sampling we mean uniform sampling in 3-D, uniform sampling in the image plane is not usually uniform sampling 3-D). Generally speaking, the moment method performs well and produces a guess that is correct within  $\pm 15$  degrees. Therefore, it is sufficient to use what may sometimes be a slightly flawed initial guess as input to spring attachment anyway, and then iteratively refine it (if need be) in tandem with the fitting solution.

### 2.3.2 Point Projection

This initial estimate of position, orientation, and size defines an elliptical coordinate system. Our attention now turns to attaching springs to the surface of our initial estimate. To establish a correspondence between sensor measurements and the nodes of the undeformed model, each

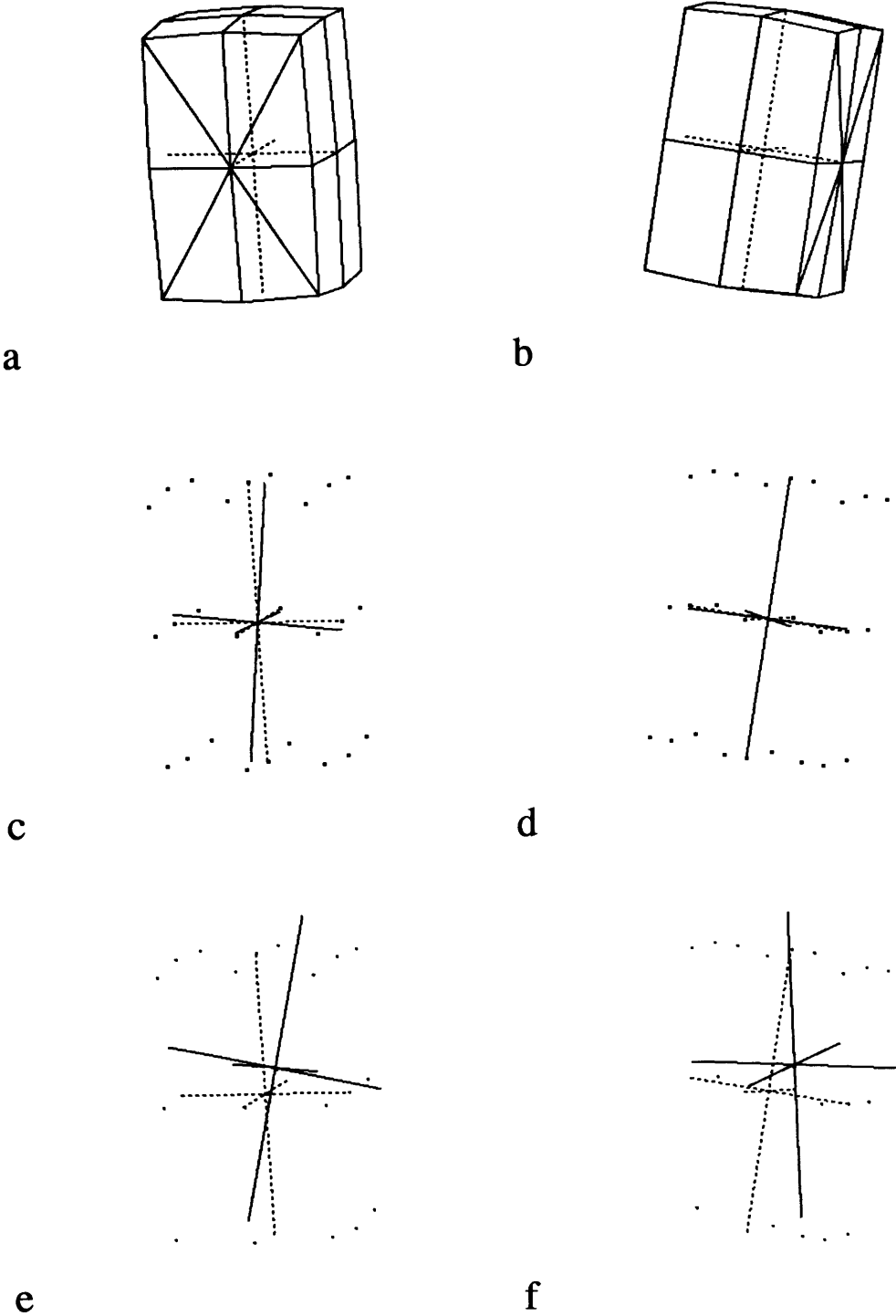


Figure 2-3: Using the moment method for initial model placement

sensor measurement,  $\mathbf{x}$  is translated and rotated into the object's coordinate frame to produce  $\tilde{\mathbf{x}}$ . The point  $\tilde{\mathbf{x}}$  is then projected onto the ellipsoid's isoparametric space by use of a normal projection function  $P(\tilde{\mathbf{x}})$ . This function returns latitude  $\omega$ , and longitude  $\eta$ .

The projection function  $P(\tilde{\mathbf{x}}) = (\eta, \omega)$  for ellipsoids is computed as follows. We first find  $\omega$  by observing:

$$\frac{a_0 \tilde{y}}{a_1 \tilde{x}} = \frac{\cos \eta \sin \omega}{\cos \eta \cos \omega} = \tan \omega \quad (2.27)$$

where  $a_0$ ,  $a_1$ , and  $a_2$  are the  $x$ ,  $y$ , and  $z$  axis sizes for the ellipsoid, and  $(\tilde{x}, \tilde{y}, \tilde{z})^T = \tilde{\mathbf{x}}$  is the sensor data point in the ellipsoid's coordinate system. From Equation 2.27, we see that:

$$\omega = \text{atan}\left(\frac{a_0 \tilde{y}}{a_1 \tilde{x}}\right). \quad (2.28)$$

The remaining parameter,  $\eta$ , is determined by either

$$\eta = \text{atan}\left(\frac{a_0 \tilde{z} \cos \omega}{a_2 \tilde{x}}\right) \quad \text{or} \quad \eta = \text{atan}\left(\frac{a_1 \tilde{z} \sin \omega}{a_2 \tilde{y}}\right) \quad (2.29)$$

depending on whether  $\tilde{x}$  or  $\tilde{y}$  is larger.

This projection implicitly defines a correspondence between the sample points and the nodal coordinates. By using the FEM interpolation functions  $\mathbf{H}$ , the virtual load generated by each data point is then distributed to the nodes whose positions surround the data point's latitude and longitude. We have found, however, that when there are enough nodes, most data points project near a node; consequently, it is sufficiently accurate to distribute these virtual loads to the surrounding nodal points using a simple Gaussian weighting or by bilinear interpolation. This is discussed further in the appendix.

For simple objects (e.g., cubes, bananas, cylinders) our experimental results show that this method of establishing spring attachment points yields accurate, unique object descriptions. It should be noted, however, that because  $\Phi$  linearizes object rotation it is important that the elliptical coordinate system established by the moment method be sufficiently close to the object's true coordinate system. We have found that as long as our initial estimate of orientation is within  $\pm 15$  degrees, we can still achieve a stable, accurate estimate of shape.

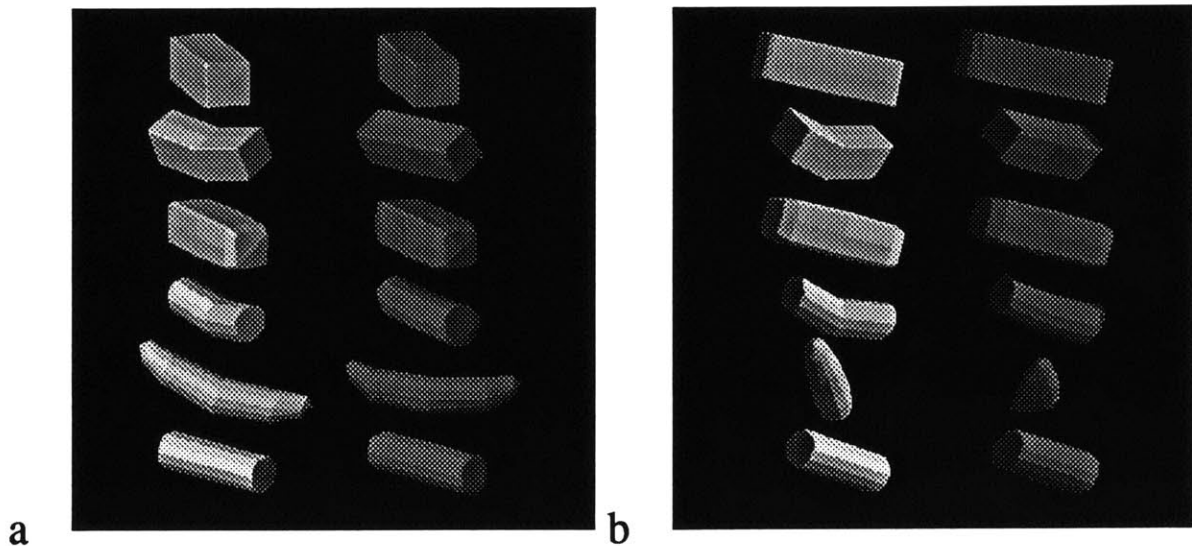


Figure 2-4: Front (a) and side (b) views of fitting a rectangular box, cylinder, banana, etc., using sparse 3-D point data with 5% noise. The original models are shown in white, and the recovered models are shown in darker grey. Given the positions of some of the object's visible surface points (as shown in (a)), we can recover the full 3-D model as is shown in the side view (b).

For complex objects, however, the spring attachment problem is sufficiently nonlinear that we have found that it necessary to establish attachment positions iteratively. We accomplish this by first fitting a solid model as above, and then use that derived model to more accurately determine the spring attachment locations. In our experiments we have found that two or three iterations of this procedure are normally sufficient to produce a good solution to both the spring attachment and the fitting problems.

## 2.4 Examples of Recovering 3-D Models

### 2.4.1 A Synthetic Example

Figure 2-4 shows an example using very sparse synthetic range information. White objects are the original shapes that the range data was drawn from, and the grey objects are the recovered

3-D models. For each white object in Figure 2-4(a) the position of visible corners, edges, and face centers was measured, resulting in between 11 and 18 data points. These data points were then corrupted by adding 5 mm of uniformly distributed noise to each data point's  $x$ ,  $y$ , and  $z$  coordinates; the maximum dimension of each object is approximately 100 mm. Note that only the *front-facing* 3-D points, e.g., those visible in Figure 2-4(a) at the left, were used in this example. Total execution time on a standard Sun 4/330 is approximately 0.1 seconds per model recovery.

Despite the rather large amount of noise and a complete lack of information about the back side of the object, it can be seen that Equation 2.24 does a good job of recovering the entire 3-D model. This is especially apparent in the side view, shown in Figure 2-4(b), where we can see that even the back side of the recovered models (grey) are very similar to the originals (white). This accuracy despite lack of 360° data reflects the fact that Equation 2.24 provides the shape estimate with the least internal strain energy, so that symmetric and mirror symmetric solutions are preferred.

## 2.4.2 An Example Using Laser Rangefinder Data

A second example uses 360° laser rangefinder data of a human head, as shown in the left-hand column of Figure 2-5. There are about 2500 data points. Equation 2.24 was then used to estimate the shape, using only the low-frequency 30 modes. The low-order recovered model is shown in the middle column; because of the large number of data points execution time on a Sun 4/330 was approximately 5 seconds. It can be seen that the low-order modes provide a sort of qualitative description of the overall head shape.

A full-dimensionality recovered model is shown in the right-hand column of 2-5. In the ThingWorld system [39, 35], rather than describing high-frequency surface details using a finite element model with as many degrees of freedom as there are data points, we normally augment a low-order finite element model with a spline description of the surface details. In our representation, this spline description is known as a *displacement map*. A displacement map can, of course, be similar to splines used by Terzopoulos *et al.* This provides us with a

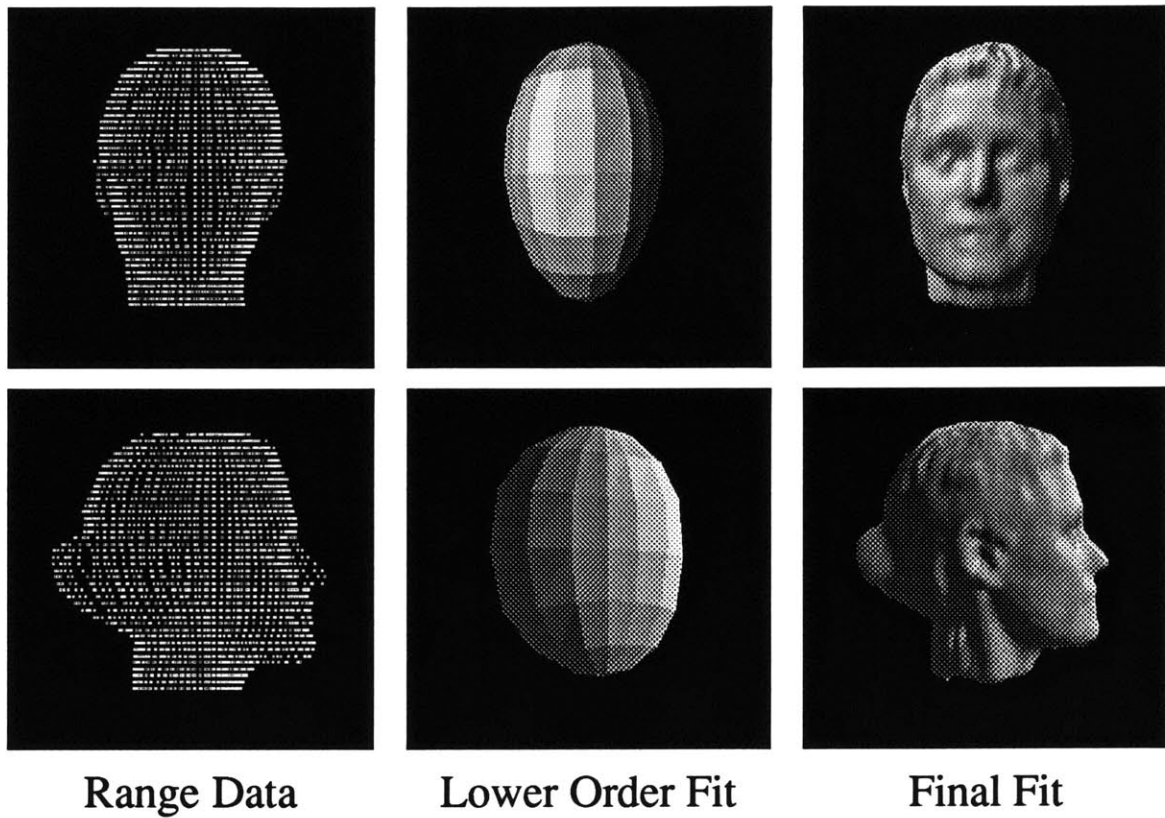


Figure 2-5: Fitting laser rangefinder data of a human face. Left column: original range data, Middle column: recovered 3-D model using only low-order modes, Left Column: full recovered model.



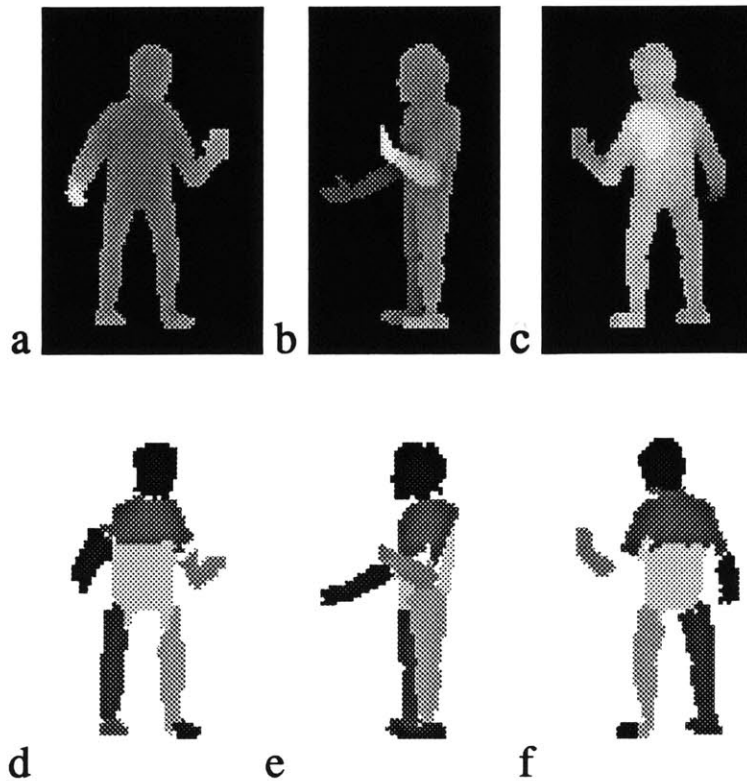


Figure 2-6: Segmenting range data of a human figure into parts. (a,b,c): range data of a human figure from three different viewpoints. (d,e,f): the  $2\frac{1}{2}$ -D segmentations for each view obtained using simple polynomial shape models. Taken from [14].

two-layered representation (low-order finite element model + surface detail spline description = final model) that we find to be both more efficient to recover and more useful in recognition, simulation, and visualization tasks than a fully-detailed finite element model. Displacement maps are used to store the residual error between the recovered deformable model and the original data measurements; we will discuss how displacement maps are actually computed in Chapter 5.

### 2.4.3 An Example of Fusing Multiple Viewpoints

Figure 2-6 shows an example of segmenting laser rangefinder data of a human figure using second order polynomial basis functions, and then integrating the various views into a single 3-D model. Figures 2-6(a,b,c) show three frames of range data of a human figure taken from

different views. Each image is 128 x 64 pixels in size. For each frame a segmentation was produced using the minimum-description-length algorithm described by Darrell, Sclaroff, and Pentland [14], yielding the segmentations shown in Figure 2-6(d,e,f). Despite use of simple polynomial shape models, a good segmentation was obtained for each view. More importantly, the segmentation is stable across widely different views — despite occlusions — so that data from the same “parts” can be related across the different views. There will be some “smaller” parts of the image (fingers for instance), which, due to the scale of the segmentation, will not be consistent across view. To get around this problem, a size threshold was used [14].

As each of the  $2\frac{1}{2}$ -D segmentations was produced it was used as input to Equation 2.24, so that a 3-D model of the scene was progressively built up over time. The segmented data was related to the previously-recovered 3-D models by rotating the data using the known changes in camera parameters. The shape of the 3-D model was then re-estimated for all of the available data using Equation 2.24.

Figure 2-7 shows two views of the 3-D models being incrementally fit to the data provided by the  $2\frac{1}{2}$ -D segmentations. It can be seen that as each  $2\frac{1}{2}$ -D segmentation is obtained, it is fused together with the previous segmentations and the 3-D shape model is thus progressively improved. Note that because the  $2\frac{1}{2}$ -D segmentations tend to miss points near the occluding boundaries, the 3-D shapes estimated from a single view (e.g., (a) and (d)) are more flat than they should be.

The final 3-D model, shown in Figure 2-7(c) and (d), encodes the data from all three views in only 276 parameters, yielding an image compression ratio of approximately 300:1 (background pixels excluded) or 900:1 if the three images are considered as independent. The signal-to-noise ratio, a measure of the accuracy of the 3-D description, is 32 dB, so that 99.94% of the data’s variance is accounted for by the recovered model.

The dB statistic is computed by comparing the 3-D variance of the original data to the 3-D variance not accounted for by the recovered 3-D models. To compute these variances we measure the 3-D distance between each original data point and the recovered surface along a line from the model’s center of mass, and compare this to the 3-D distance between the original

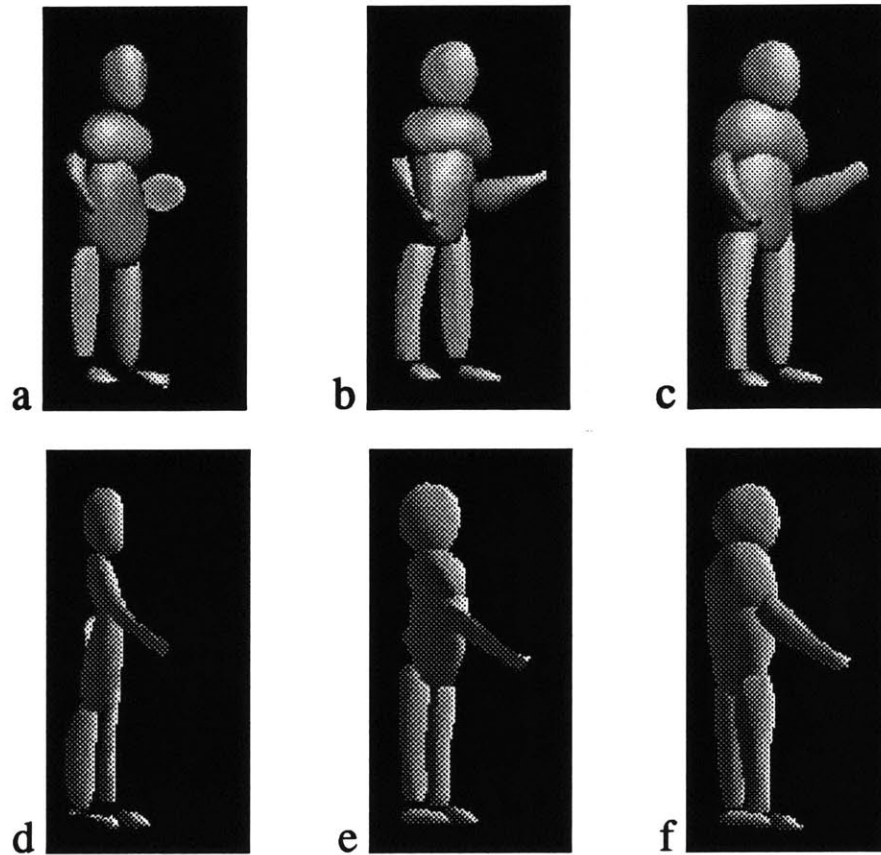


Figure 2-7: Incremental fitting to range data of a human figure. Segmentations from previous figure were used as input to a 3-D shape estimation process. Top and bottom rows show two views of recovered 3-D model. (a,d) shows fit based on points from one  $2\frac{1}{2}$ -D segmentation, (b,e) fit after two views have been used, (c,f) after all three views. Because the  $2\frac{1}{2}$ -D segmentations are stable, each additional view improves the 3-D model.

data point and the model's center of mass. The segmentation determines which model is used for each point's measurements.

# Chapter 3

## Object Recognition

Perhaps the major drawback of physically-based models has been that they are not very useful for recognition, comparison, or other database tasks. This is primarily because they normally have more degrees of freedom than there are sensor measurements, so that the recovery process is underconstrained. Therefore, although heuristics such as smoothness or symmetry can be used to obtain a solution, they do not produce a stable, unique solution.

The major problem is that when the model has more degrees of freedom than the data, the model's nodes can slip about on the surface. The result is that there are an infinite number of valid combinations of nodal positions for any particular surface. This difficulty is common to all spline and piecewise polynomial representations, and is known as the *knot problem*.

For all such representations, the only general method for determining if two surfaces are equivalent is to generate a number of sample points at corresponding positions on the two surfaces, and observe the distances between the two sets of sample points. Not only is this a clumsy and costly way to determine if two surfaces are equivalent, but when the two surfaces have very different parameterizations it can also be quite difficult to generate sample points at "corresponding locations" on the two surfaces.

The modal representation, assuming that all modes are employed, decouples the degrees of freedom within the non-rigid dynamic system of Equation 2.1, but it does not by itself reduce the total number of degrees of freedom. Thus a complete modal representation suffers from

the same problems as all of the other representations.

### 3.1 Using Recovered Mode Values for Recognition

The obvious solution to the problem of non-uniqueness is to discard enough of the high-frequency modes that we can obtain an overconstrained estimate of shape, as was done for the shape recovery problem above. Use of a reduced-basis modal representation results in a *unique* representation of shape because the modes (eigenvectors) form an orthonormal basis set. Therefore, there is only one way to represent an object, and that is in its canonical position. Further, because the modal representation is frequency-ordered, it has stability properties that are similar to those of a Fourier decomposition. Just as with the Fourier decomposition, an exact subsampling of the data points does not change the low-frequency modes. Similarly, irregularities in local sampling and measurement noise tend to primarily affect the high-frequency modes, leaving the low-frequency modes relatively unchanged.<sup>1</sup>

The primary limitation of this uniqueness property stems from the finite element method's linearization of rotation. Because the rotations are linearized, it is impossible to uniquely determine an object's rotation state. Thus object symmetries can lead to multiple descriptions, and errors in measuring object orientation will cause commensurate errors in shape description.

Thus by employing a reduced-basis modal representation we can obtain overconstrained shape estimates that are also unique except for rotational symmetries. To compare two objects with known mode values  $\tilde{\mathbf{U}}^1$  and  $\tilde{\mathbf{U}}^2$ , one simply compares the two vectors of mode values using a simple distance metric:

$$\epsilon = \sqrt{\sum_{i=1}^m (\tilde{u}_i^1 - \tilde{u}_i^2)^2} \quad (3.1)$$

---

<sup>1</sup>Note that when there are many more data points than degrees of freedom in the finite element model, the interpolation functions  $\mathbf{H}$  act as filters to bandlimit the sensor data, thus reducing aliasing phenomena. If, however, the number of modes used is much smaller than the number of degrees of freedom in the finite element model, then it is possible to have significant aliasing.

or, as we prefer, normalized dot product:

$$\varepsilon = \frac{\tilde{\mathbf{U}}^1 \cdot \tilde{\mathbf{U}}^2}{\|\tilde{\mathbf{U}}^1\| \|\tilde{\mathbf{U}}^2\|} \quad (3.2)$$

Vector norms other than the dot product can be employed; in our experience all give roughly the same recognition accuracy. Examples of comparing recovered models using both metrics are show in Figures 3-5 and 3-4 at the end of this chapter.

To recognize a recovered model with estimated mode values  $\tilde{\mathbf{U}}$ , one compares the recovered mode values to the mode values of all of the  $p$  known models:

$$\varepsilon_k = \frac{\tilde{\mathbf{U}} \cdot \tilde{\mathbf{U}}^k}{\|\tilde{\mathbf{U}}\| \|\tilde{\mathbf{U}}^k\|} \quad k = 1, 2, \dots, p \quad (3.3)$$

The known model  $k$  with the maximum dot product  $\varepsilon_k$  is the model best matching the recovered model, and thus declared to be the model recognized. Note that for each known model  $k$ , only the vector of mode values  $\tilde{\mathbf{U}}^k$  needs to be stored.

The first six entries of  $\tilde{\Phi}$  are the rigid-body modes (translation and rotation), which are normally irrelevant for object recognition. Similarly, the seventh mode (overall volume) is sometimes irrelevant for object recognition, as many machine vision techniques recover shape only up to an overall scale factor. Thus rather than computing the dot product with all of the modes  $\tilde{\mathbf{U}}$ , we typically use only modes number eight and higher, e.g.,

$$\varepsilon_k = \frac{\sum_{i=8}^{i=m} \tilde{\mathbf{u}}_i \tilde{\mathbf{u}}_i^k}{\sqrt{\sum_{i=8}^{i=m} \tilde{\mathbf{u}}_i^2} \sqrt{\sum_{i=8}^{i=m} (\tilde{\mathbf{u}}_i^k)^2}} \quad k = 1, 2, \dots, p \quad (3.4)$$

where  $m$  is the total number of modes employed. By use of this formula we obtain translation, rotation, and scale-invariant matching.

The ability to compare the shapes of even complex objects by a simple dot product makes the modal representation well suited to recognition, comparison, and other database tasks. In the following section we will evaluate the reliability of the combined shape recovery/recognition process using both synthetic and laser rangefinder data.

## 3.2 Recovering and Recognizing Models

There are many ways by which to judge the process of comparing and recognizing 3-D objects. For instance:

- **Accuracy:** Is it possible to obtain good accuracy from the system?
- **Graceful Degradation:** As noise is added to the sensor measurements, does recognition accuracy degrade slowly?
- **Generalization:** If two objects are similar to human observers, are they similar to the recognition/comparison system? This is especially useful for database tasks.

In the following sections we will address each of these issues in turn.

### 3.2.1 Accuracy

To assess accuracy, we conducted an experiment to recover and recognize face models from range data generated by a laser range finder. In this experiment we obtained laser rangefinder data of eight people's heads from a five different viewing directions: the right side ( $-90^\circ$ ), halfway between right and front ( $-45^\circ$ ), front ( $0^\circ$ ), halfway between front and left ( $45^\circ$ ), and the left side ( $90^\circ$ ). We have found that people's heads are only approximately symmetric, so that the  $\pm 45^\circ$  and  $\pm 90^\circ$  degree views of each head have quite different detailed shape. In each case the range data was from the forward-facing, visible surface only.

Data from a  $360^\circ$  scan around each head was then used to form the stored model of each head that was later used for recognition. Full-detail versions of these eight reference models are shown in Figure 3-1; note that in some cases a significant amount of the data is missing. As previously, only the low order 30 deformation modes were used in the shape extraction and recognition procedure. Because the low order modes provide a coarse, qualitative summary of the object shape (see the middle column of Figure 2-5) they can be expected to be the most stable with respect to noise and viewpoint change. Total execution time on a standard Sun 4/330 averaged approximately 5 seconds per fitting and recognition trial.

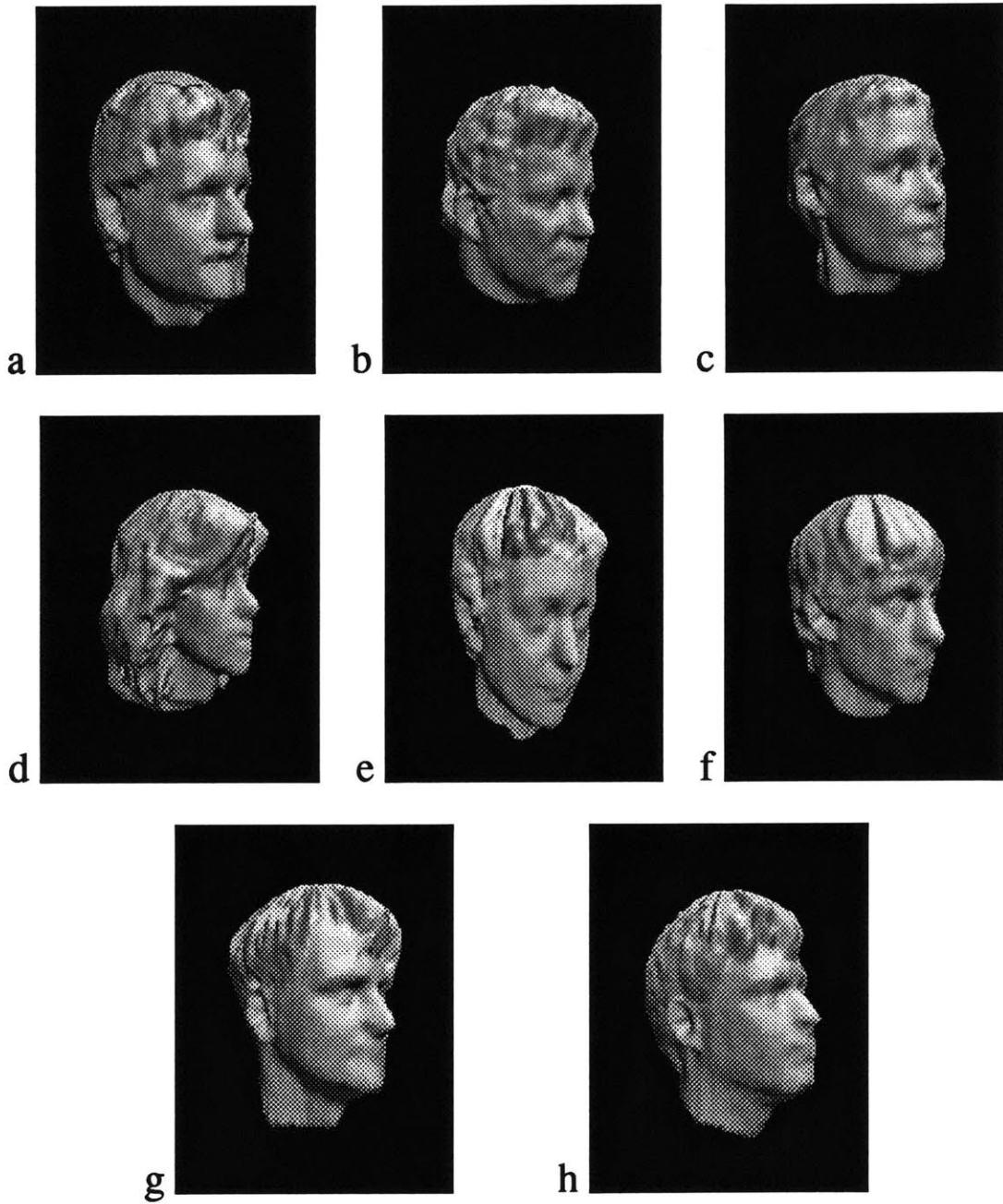


Figure 3-1: Eight heads used in our recognition experiment. Note that in some cases there is significant missing data.





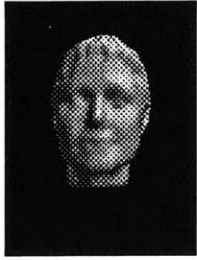


	$-90^\circ$	$-45^\circ$	$0^\circ$	$45^\circ$	$90^\circ$
					
a	0.16	-0.19	-0.24	-0.19	-0.12
b	0.26	0.28	0.35	0.37	0.30
c	<b>0.88</b>	<b>0.99</b>	<b>0.99</b>	<b>0.98</b>	<b>0.99</b>
d	0.03	0.15	0.21	0.21	0.11
e	0.06	-0.13	-0.10	-0.04	-0.06
f	0.58	0.44	0.46	0.46	0.50
g	0.53	0.53	0.52	0.50	0.58
h	0.42	0.47	0.53	0.49	0.50

Figure 3-2: Recognizing faces from five different points of view.

Recognition was accomplished by first recovering a 3-D model from the visible-surface range data, and then comparing the recovered mode values to the mode values stored for each of the three known head models using Equation 3.4. The known model producing the largest dot product was declared to be the recognized object. The first seven modes were not employed, so that the recognition process was translation, rotation, and scale invariant.

Figure 3-2 illustrates typical results from this experiment. The top row of Figure 3-2 illustrates the five models recovered from range data from the front, visible surface using viewpoints of  $-90^\circ$ ,  $-45^\circ$ ,  $0^\circ$ ,  $45^\circ$ , and  $90^\circ$ . Each of these recovered head models look similar, and more importantly have approximately the same deformation mode values  $\tilde{U}$ , despite the

wide variations in input data. Modes 8 through 30 of these recovered models were then compared to each of the stored head models. The dot products obtained are shown below each recovered head model.

In Figure 3-2 all of the input data was views of Kim (depicted as **head c** in the tables). As can be seen, the dot products between recovered 3-D model and known model are quite large for Kim's head model. In fact, in this example the smallest correct dot product is almost three times the magnitude of any of the incorrect dot products; the same was also true for range data of the other subjects.

In this experiment 92.5% accurate recognition was obtained. That is, we successfully recovered 3-D models and recognized each of the eight test subjects from each of the five different views with only three errors. Analysis of the recognition results showed that, while the average dot product between different reference models was 0.31 ( $72^\circ$ ), the average dot product between models recovered from different views of the same person was 0.95 ( $18^\circ$ ). Thus recognition was typically extremely certain. All three errors were from front-facing views, where relatively few discriminating features are visible; remember that only overall head shape, and not details of surface shape, were available to the recognition procedure as only 30 modes were employed.

### 3.2.2 Graceful Degradation

Model recognition should not only be accurate, it should also be robust with respect to noise and sparsely-sampled data. The next example, therefore, demonstrates robustness of recognition when using noisy data.

In this experiment the collection of relatively similar objects shown in the Figure 2-4 were used to generate synthetic data with varying object orientation and varying degrees of noise. From this synthetic data 3-D models were recovered, and then recognized by comparing the mode values of the recovered objects to those of the original objects. Again, translation, rotation, and scale invariant matching was employed.

Within each trial the 3-D orientation of each object was chosen from the entire viewing

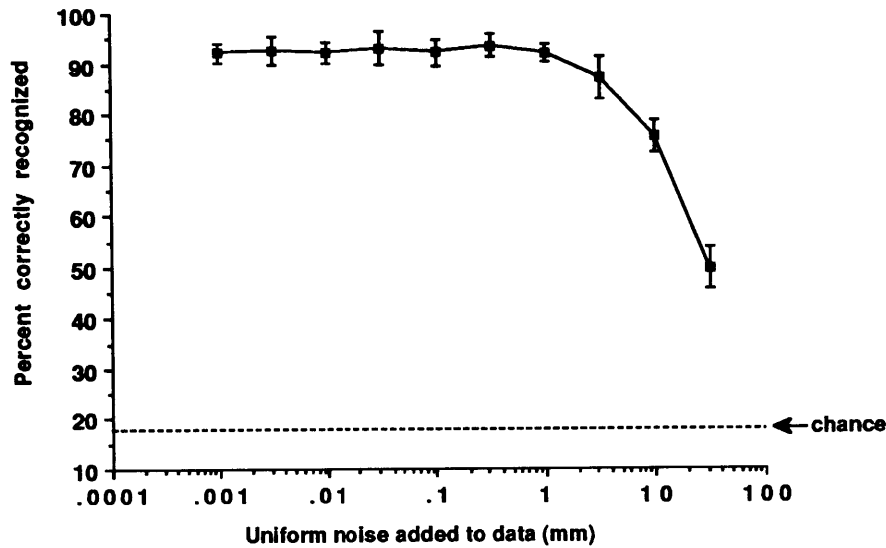


Figure 3-3: Graph showing how recognition degrades as a function of sensor noise.

sphere at random, and the position of visible corners, edges, and face centers was measured, resulting in between 11 and 18 data points. Data points were then corrupted by adding uniformly distributed noise to the data point's  $x$ ,  $y$ , and  $z$  coordinates. Note that only the *front-facing* 3-D points were used. The projection of data points onto the model was assumed to be known, so that the performance of the fitting and recognition algorithms could be evaluated independently of the moment method for estimating initial orientation. Total execution time on a standard Sun 4/330 is approximately 0.1 seconds per fitting and recognition trial.

Figure 3-3 shows results of this experiment. The horizontal axis is the amount of noise, in millimeters, added to the synthetic range data. For comparison, the maximum dimension of each object is approximately 100 mm. The vertical axis is the mean percent accuracy of recognition over 100 trials. Error bars are shown for each level of noise.

As can be seen, a high level of accuracy is maintained up to about 8 mm (approximately 8 percent) noise, and thereafter declines smoothly. In this experiment almost all errors in recognition resulted from either (1) special views in which discriminating features (such as the slight bend that differentiates the first and second objects in Figure 2-4) were not visible, or (2) cases in which noise corrupted the discriminating features sufficiently that objects were

confused with each other. Up to approximately 8 mm of noise almost all errors were due to special views, while for greater levels of noise the corruption of discriminating features dominates the error rate. We interpret the results of this experiment to indicate that our fitting and recognition algorithms are quite robust, and degrade gradually in the presence of noise.

### 3.2.3 Generalization

Given the mode values for a 3-D model, perhaps obtained from range data, we may want to perform shape comparisons and other database tasks. Using a reduced-basis modal representation such comparisons are extremely cheap, requiring only that a dot product be computed. Position, rotation and size invariant matching is even cheaper, as the first seven modes from the comparison.

However to be generally useful, the comparison by dot product must induce a well-behaved and perceptually natural similarity metric over the space of objects. That is, objects that people think are “similar” must produce large dot products, and objects that people think are dissimilar must produce small dot products. Unfortunately, to prove that the mode comparisons induce a “nice” similarity metric would require a massive psychophysical experiment.

We can, however, demonstrate that in simple cases mode comparison does indeed induce a well-behaved and perceptually natural similarity metric. To accomplish this we created a series of five objects that had various degrees of similarity with a rectangular solid. These objects were then compared using the modal dot product rule, and the inter-object similarities observed. Figure 3-5 shows the similarity measurements obtained for these objects; as can be seen, they measured similarities correspond at least roughly with our perceptual judgements of similarity. Such rough correspondence suggests that dot products of  $\tilde{U}$  provide a reasonable metric for shape comparison and will be well-behaved when used for database search or for recognition.













						
	0.0	6.825	12.903	22.399	32.267	43.709
	6.825	0.0	14.597	16.443	28.4573	40.468
	12.903	14.597	0.0	25.850	24.663	38.218
	22.399	16.443	25.850	0.0	23.226	33.452
	32.267	28.457	24.663	23.226	0.0	23.431
	43.709	40.468	38.218	33.452	23.431	0.0

Figure 3-4: Using  $L_2$  distance metric to measure object similarity. Comparisons were made translation, scale, and rotation invariant by use in a manner similar to that used in Equation 3.4













						
	1.000	0.993	0.974	0.924	0.820	0.623
	0.993	1.000	0.968	0.957	0.850	0.647
	0.974	0.968	1.000	0.900	0.903	0.774
	0.924	0.957	0.900	1.000	0.888	0.668
	0.820	0.850	0.903	0.888	1.000	0.934
	0.623	0.647	0.774	0.668	0.934	1.000

Figure 3-5: Using normalized dot product to measure object similarity. Comparisons were made translation, scale, and rotation invariant by use of Equation 3.4

# Chapter 4

## Recovery, Tracking, and Recognition from Contour

Up until now, the fitting algorithm has been used for recovering and recognizing models from static, three-dimensional point data. The purpose of this chapter will be to expand the application of our technique to contour data and to account for variable certainty of sensor data — using a variant of weighted least squares. Furthermore, we will expand the algorithm to incorporate tracking moving contours over time — using Kalman filtering.

### 4.1 Head Recognition from Contours

To fit a deformable solid to three-dimensional contours is simple enough. All we need to do is attach springs between each contour and the underlying deformable model. Typically, each contour is sampled to get a collection of data points, and then these points are plugged into the model recovery algorithm describe in Chapter 2. Results of this are demonstrated in the following example.

In a head recovery and recognition experiment similar to that shown in Figure 2-5, we used a few 3-D head contours instead of full range data to see how well our techniques performed in the case of sparse silhouette data. In this experiment, we recovered heads from 2, 3, 4,

and finally 5 contours, in order to approximate the information available in an active vision scenario. In each trial, the contours were spaced evenly in rotation. An example of the contours used in this experiment is shown in Figure 4-1; these contours were taken from the same head depicted in Figure 3-1c.

As in the previous experiment, the recovered heads were compared against the full-detail versions of the reference heads shown in Figure 3-1, and the model producing the largest dot product was declared to be the recognized object. Heads were compared using the scale, rotation, and translation invariant version of Equation 3.3.

In our experiments, recognition with two contours averaged 93.75% accurate recognition. Accuracy improved as more contours were added until 96.875 percent of the heads were correctly identified when 5 contours were used. The results were not as good if the contours did not include the traditional side “silhouette,” and performed best when the data’s spring attachment was smoothed out more across the surface. Total execution times were slightly greater than those for the full data experiment, averaging 5 seconds per fitting and recognition trial on a Sun 4/330. The greater execution time is attributable to the more careful distribution and smoothing of spring attachment between contours and the underlying deformable model.

## 4.2 3-D shape from 2-D contours

In the case where we are given only 2-D contour information, we can still sample the curves and then employ the same equations to estimate shape; however, we must generalize Equation 2.24 to reflect the uncertainty we have about the  $z$  coordinate of each contour point. In general, we can express uncertainty by using a variant of weighted least squares.

Recall from Chapter 2 that the solution to the fitting problem is obtained by inverting the diagonal matrix  $\tilde{\mathbf{K}}$ :

$$\tilde{\mathbf{U}} = \tilde{\mathbf{K}}_6^{-1} \Phi^T \mathbf{R} \quad (4.1)$$

To fit two-dimensional contour data, we alter Equation 4.1 to reflect the fact that some sensor measurements are more certain than others. We accomplish this by introducing a  $3n \times 3n$



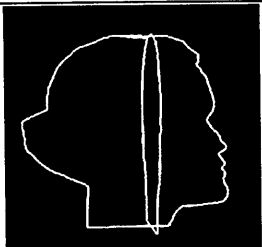
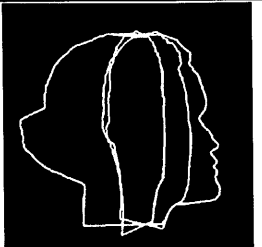
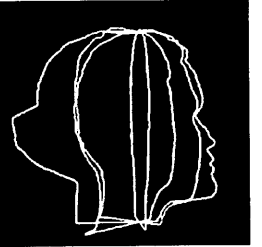
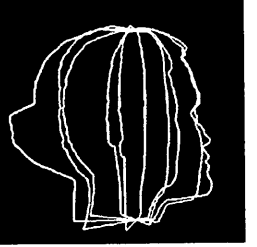
	2 Contours	3 Contours	4 Contours	5 Contours
				
a	0.53	0.56	0.60	0.65
b	0.24	0.22	0.14	0.09
c	<b>0.90</b>	<b>0.92</b>	<b>0.94</b>	<b>0.97</b>
d	0.14	0.01	-0.04	-0.02
e	-0.08	-0.06	-0.11	-0.06
f	0.36	0.38	0.42	0.54
g	0.27	0.30	0.35	0.48
h	-0.21	-0.18	-0.20	-0.07

Figure 4-1: Recognizing heads recovered from contours.

diagonal weighting matrix  $\mathbf{W}$ :

$$\tilde{\mathbf{U}} = \tilde{\mathbf{K}}_6^{-1}(\mathbf{W}\Phi)^T \mathbf{R} \quad (4.2)$$

The diagonal entries of  $\mathbf{W}$  are inversely proportional to the uncertainty (variance) of the data associated with each of the nodal coordinates. The effect of  $\mathbf{W}$  is to make the strength of the virtual springs associated with each data point reflect the uncertainty of the measurement. We will show how this technique is used for recovering models from 2-D image contours, where the  $z$  coordinate for points is unknown — though the weighting method can be used in any case where the certainty of data measurements can vary.

### Recovering a 3-D Heart Model from a 2-D X-ray Image

Figure 4-2 shows an example using contour data from a an X-ray image of human heart to obtain a 3-D modal description. Figure 4-2(a) shows the one frame from a sequence of X-ray images, with the edge contours overlaid. The contours were extracted by first thresholding

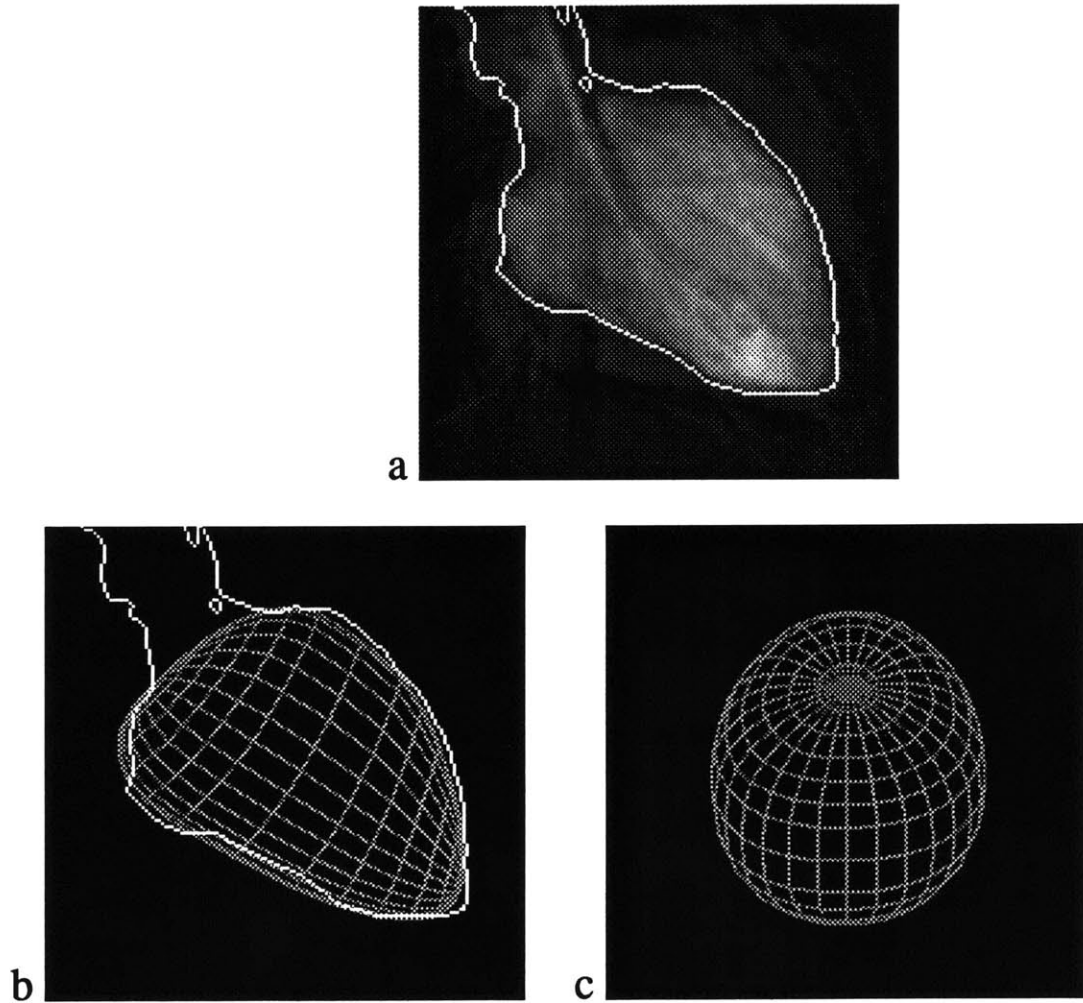


Figure 4-2: An example showing the use of a 2-D image contour to recover a 3-D deformable solid model. The original image and contour are shown in (a). The model is recovered from the contour as shown in (b). An orthogonal side view is shown in (c).

the X-ray imagery, and then finding the Laplacian zero-crossings. From these contours 3-D shape was estimated by use of Equation 4.2. The resulting estimate is shown in Figure 4-2(b) as a 3-D wireframe overlaid on the original X-ray data. Figure 4-2(c) shows the recovered model from the side. Execution time was approximately one second on a standard Sun 4/330.

It can be seen that a reasonable estimate of 3-D shape was obtained. Because only bounding contour information was available, the shape estimated along the  $z$  axis (shown in Figure 4-2(c)) is determined by finding the minimum stress state that still fits the bounding contour. The  $z$ -axis shape cannot, therefore, be regarded as accurate but only as plausible and consistent. Note that use of a minimum stress criterion for solution means that symmetric and mirror symmetric shapes are preferred.

### **Recovering Symmetric Objects from 2-D contours**

Researchers from Nippon Telephone and Telegraph have been investigating applying the methods and software developed as part of this thesis to machine vision problems. This example was taken from collaborative work we have done with Jun Inagawa, a research affiliate from NTT Data. He used modes to recover pills from the edge data extracted from greyscale images, and then intends to use this data to recognize the pills from a database of approximately 3000 pills. Since the geometry of pills is known to be axis-symmetric, they decided to find and then take advantage of the pills' axis of rotational symmetry.

By using a subset of modes which act along the axis of symmetry, we have gotten satisfactory results, as shown in Figure 4-3. The original image 4-3(a,b) was used to obtain edges and axis of symmetry 4-3(c,d). These edges and axis were then fed into the fitting algorithm, where a subset of modal amplitudes was recovered — results show in front view 4-3(e,f) and side view 4-3(g,h). The subset included only those modes which described deformations which acted along the axis of symmetry (assumed to be the  $z$  axis).

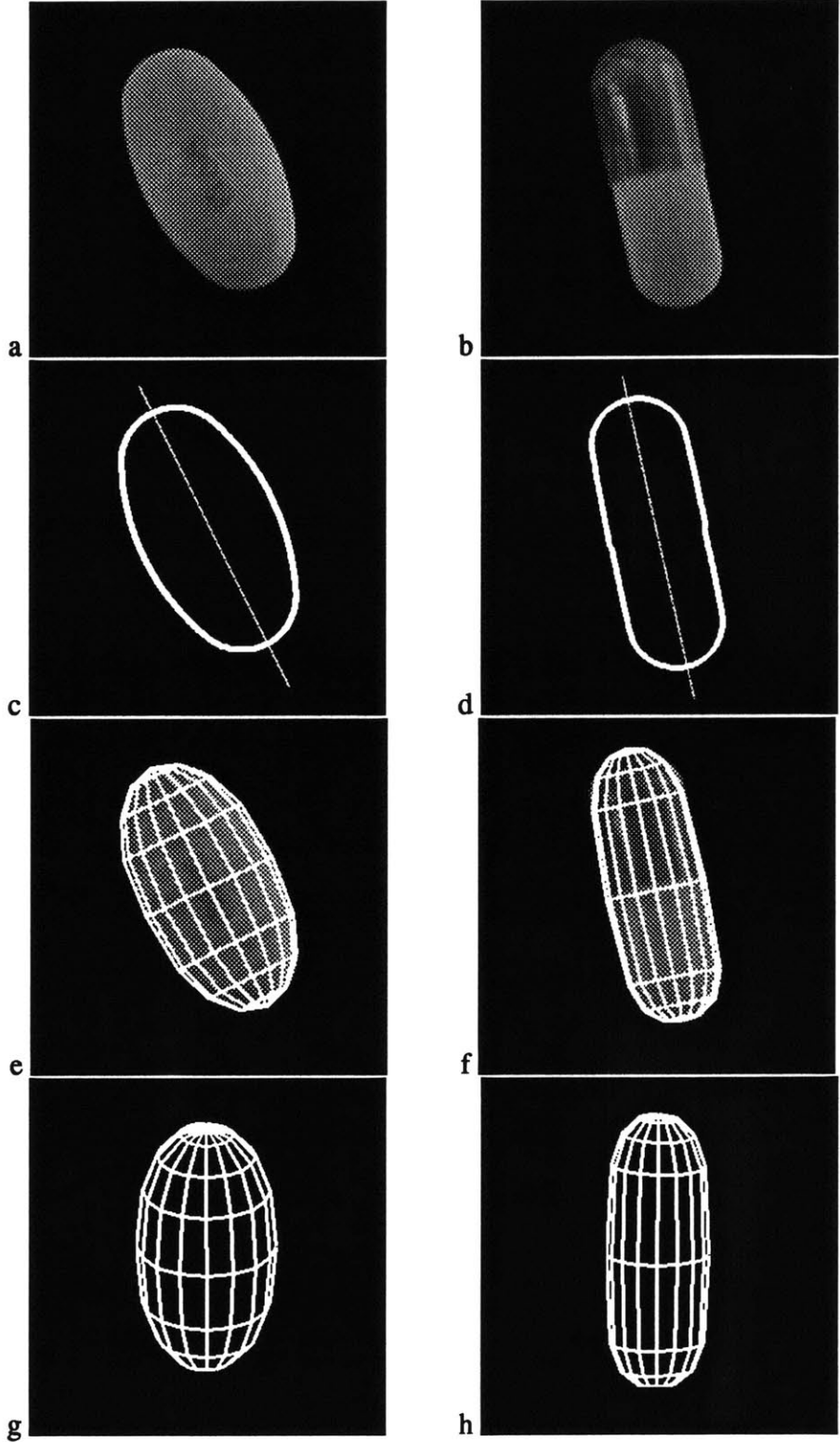


Figure 4-3: Recovering and recognizing pills from contour and center axis.

### 4.3 Dynamic Estimation of Rigid and Non-Rigid Motion

In the previous sections we have addressed static shape estimation, where data at only one instant is considered. For time sequences, however, it is necessary to also consider the *dynamic* properties of the body and of the data measurements. The Kalman filter [19] is the standard technique for obtaining estimates of the state vectors of dynamic models, and for predicting the state vectors at some later time. Outputs from the Kalman filter are the optimal (weighted) least-squares estimate for non-Gaussian noises.

In the case of motion tracking, however, the initial guess is given by the tracking process itself. That is, given a Kalman filter estimate of shape and both rigid and non-rigid motion we can extrapolate the current shape estimate to the next time instant, and then project the new data points onto the extrapolated shape in order to determine the spring attachments points. We have found that this extrapolation process produces a very good estimate of the true spring attachment points; indeed good results can be obtained as long as the extrapolated orientation and shape is within roughly  $\pm 10\%$  of the correct value.

The first use of Kalman filtering for motion estimation was by Brodia and Chellappa [11], who presented a careful evaluation of the approach. Work by Faugeras, Ayache and their colleagues, and more recently many others, has thoroughly developed the subject [16, 2]. In this section we will develop a Kalman filter that estimates position and velocity for the finite element modal parameters along the lines of Horowitz and Pentland [15]. We will then show that this particular type of Kalman filter is mathematically equivalent to time integration of the FEM governing equation for appropriate choices of mass  $M$  and stiffness  $K$ . That is, the Kalman filter may be viewed as a simulation of the model's behavior, with the observed point positions or contour data acting as guiding "forces."

### 4.3.1 The Kalman Filter

Let us define prototypical discrete-time state-space equations for a dynamic process with noise:

$$\dot{\mathbf{X}}_t = \mathbf{F}_t \mathbf{X}_t + \mathbf{G}_t \mathbf{w}_t \quad (4.3)$$

and observations

$$\mathbf{Y}_t = \mathbf{H}_t \mathbf{X}_t + \mathbf{v}_t \quad (4.4)$$

where  $\dot{\mathbf{X}}_t$  is the rate of change for the system state at time  $t$ ,  $\mathbf{Y}_t$  is the measured output of the system, and  $\mathbf{w}_t$  and  $\mathbf{v}_t$  are white noise processes having known spectral density matrices. Then the optimal estimate  $\hat{\mathbf{X}}_t$  of  $\mathbf{X}_t$  is given by the following *Kalman Filter*:

$$\hat{\mathbf{X}}_t = \mathbf{F} \mathbf{X}_t^* + \mathbf{K}_f (\mathbf{Y}_t - \mathbf{H}_t \mathbf{X}_t^*) \quad (4.5)$$

with correctly chosen Kalman gain matrix  $\mathbf{K}_f$ . At each time step,  $t$ , the filter algorithm uses a state prediction  $\mathbf{X}_t^*$ , and a sensor measurement  $\mathbf{Y}_t$  to determine an optimal linear state estimate  $\hat{\mathbf{X}}_t$ .

The prediction of the state vector  $\mathbf{X}_{t+\Delta t}^*$  for the next time step is obtained by combining the estimated  $\hat{\mathbf{X}}_t$  and Equation 4.3:

$$\mathbf{X}_{t+\Delta t}^* = \hat{\mathbf{X}}_t + \mathbf{F}_t \hat{\mathbf{X}}_t \Delta t \quad (4.6)$$

#### The Kalman Gain Factor

The gain matrix  $\mathbf{K}_f$  in Equation 4.5 minimizes the covariance matrix  $\mathbf{P}_t$  of the error  $\mathbf{e}_t = \mathbf{X}_t - \hat{\mathbf{X}}_t$ . Assuming that the cross-variance between the system excitation noise  $\mathbf{w}_t$  and the observation noise  $\mathbf{v}_t$  is zero, then  $\mathbf{K}_f = \mathbf{P}_t \mathbf{H}_t^T \mathbf{V}_t^{-1}$  where the  $n \times n$  observation noise spectral density matrix  $\mathbf{V}_t$  must be nonsingular [19]. Assuming that the noise characteristics are constant, then the optimizing covariance matrix  $\mathbf{P}_t$  is obtained by solving the *Riccati*

equation

$$\mathbf{0} = \dot{\mathbf{P}}_t = \mathbf{F}_t \mathbf{P}_t + \mathbf{P}_t \mathbf{F}_t^T - \mathbf{P}_t \mathbf{H}_t^T \mathbf{V}_t^{-1} \mathbf{H}_t \mathbf{P}_t + \mathbf{G}_t \mathbf{W}_t \mathbf{G}_t^T \quad (4.7)$$

where  $\mathbf{W}_t$  is the  $n \times n$  spectral density matrix of the acceleration noise.

### Estimation of Modal Displacement and Modal Velocity

In the current application we are primarily interested in estimation of the modal amplitudes  $\tilde{\mathbf{U}}_t$  and their velocities  $\dot{\tilde{\mathbf{U}}}_t$  at time  $t$ . In state-space notation our system of equations is

$$\begin{bmatrix} \dot{\tilde{\mathbf{U}}}_t \\ \ddot{\tilde{\mathbf{U}}}_t \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{U}}_t \\ \dot{\tilde{\mathbf{U}}}_t \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{w}_t \end{bmatrix} \quad (4.8)$$

where  $\mathbf{w}_t$  is a noise vector due to nodal accelerations. The observed variable will be the 3-D modal amplitudes  $\bar{\mathbf{U}}_t$ :

$$\bar{\mathbf{U}}_t = \tilde{\mathbf{U}}_t + \mathbf{v}_t \quad (4.9)$$

where  $\mathbf{v}_t$  is a vector of the observation noise. The Kalman filter is therefore

$$\begin{bmatrix} \hat{\dot{\mathbf{U}}}_t \\ \hat{\ddot{\mathbf{U}}}_t \end{bmatrix} = \begin{bmatrix} \dot{\tilde{\mathbf{U}}}_t^* \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{f,1} \\ \mathbf{K}_{f,2} \end{bmatrix} \left( \bar{\mathbf{U}}_t - \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{U}}_t^* \\ \dot{\tilde{\mathbf{U}}}_t^* \end{bmatrix} \right) \quad (4.10)$$

where  $\tilde{\mathbf{U}}_t^*$  is the predicted displacement, and  $\mathbf{K}_{f,1}$  and  $\mathbf{K}_{f,2}$  are the Kalman gain matrices for velocity and acceleration, respectively.

We will assume that  $\mathbf{w}_t$  and  $\mathbf{v}_t$  originate from independent noise with standard deviations  $w$  and  $v$  respectively, so that  $\mathbf{W} = w^2 \mathbf{I}$ ,  $\mathbf{V} = v^2 \mathbf{I}$ . Given  $\mathbf{W}$  and  $\mathbf{V}$  we may then determine the Kalman gain matrices, which for this simple noise model are  $\mathbf{K}_{f,1} = (\frac{2w}{v})^{1/2} \mathbf{I}$  and  $\mathbf{K}_{f,2} = (\frac{w}{v}) \mathbf{I}$ . Substituting this result into Equation 4.10 we obtain

$$\begin{bmatrix} \hat{\dot{\mathbf{U}}}_t \\ \hat{\ddot{\mathbf{U}}}_t \end{bmatrix} = \begin{bmatrix} \dot{\tilde{\mathbf{U}}}_t^* + (\frac{2w}{v})^{1/2} (\bar{\mathbf{U}}_t - \tilde{\mathbf{U}}_t^*) \\ (\frac{w}{v}) (\bar{\mathbf{U}}_t - \tilde{\mathbf{U}}_t^*) \end{bmatrix} \quad (4.11)$$

Each mode is independent within this system of equations, and so we may write the Kalman filter separately for each of the modes. We can now formulate the displacement prediction for time  $t + \Delta t$ :

$$\tilde{u}_{i,t+\Delta t}^* = \hat{u}_{i,t} + d_1 \hat{\dot{u}}_{i,t} + d_2 (\bar{u}_{i,t} - \hat{u}_{i,t}) \quad (4.12)$$

where  $d_1 = \Delta t$ , and  $d_2 = 2\Delta t^2/m_i = (\mathbf{w}/\mathbf{v})\Delta t^2 + (2\mathbf{w}/\mathbf{v})^{1/2}\Delta t$  ( $m_i$  is an estimate of the  $i^{\text{th}}$  mode's generalized mass). Equation 4.12 is exactly the central-difference update rule for direct time integration of the finite element governing equations, with “loads”  $\bar{u}_{i,t}$ ; these  $\bar{u}_{i,t}$  are in fact the “observed” mode values recovered by use of either Equation 2.24 or 4.2.

The equivalence between these Kalman filter equations and time-integration of a finite element governing equation provides an intuitive interpretation of the Kalman filter. In essence, it is integrating the shapes estimates from contour data over time by use of a mass matrix that gives “inertia” to the estimates. When the observation noise is large relative to the acceleration or excitation noise, the solution becomes similar to simple time averaging. When the acceleration noise is large relative to the observation noise, the solution becomes similar to the single-measurement case. For more a more complete discussion of filtering a tracking non-rigid motion using modal analysis and Kalman filtering, see [36].

### 4.3.2 An Example Using X-Ray Data

Figure 4-4 shows an example of recovering non-rigid motion from contour information. The 3-D shape and motion of the heart ventricle was tracked over time using the contour information shown at the top within each box of Figure 4-4. The contours were extracted by first thresholding the X-ray imagery, and then finding the zero-crossings. For each frame Equation 4.2 was used to obtain an estimate of 3-D shape from the contour information (see Figure 4-2).

The Kalman filter of Equation 4.11 was then used to estimate the 3-D rigid and non-rigid motion of the ventricle. Equation 4.12 was then used to predict the ventricle's shape, rotation, and position at the next time step, and this prediction used initialize the fitting process of



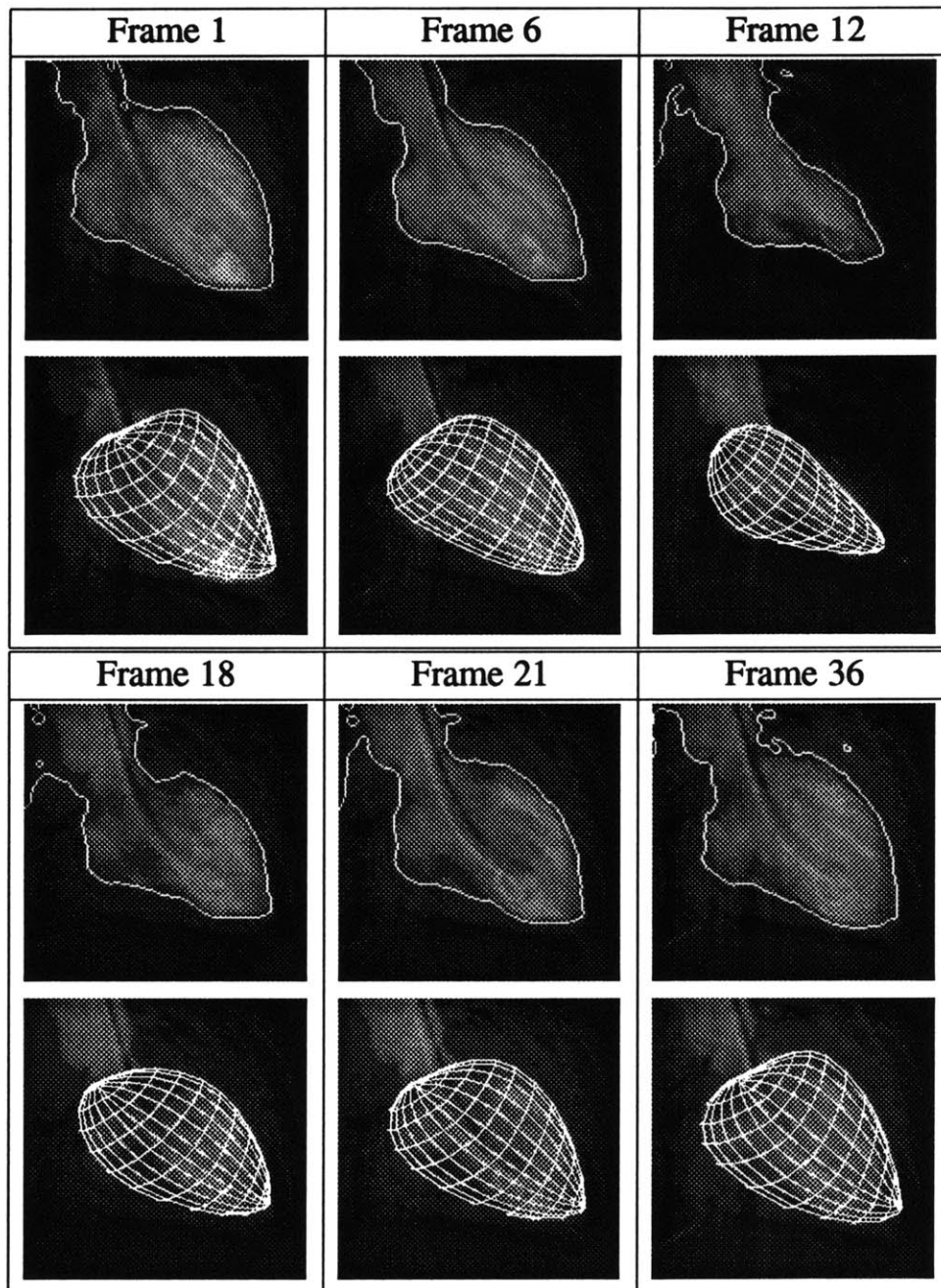


Figure 4-4: A heart's nonrigid motion as it was recovered from contours taken from a motion sequence. The contours, extracted via a simple threshold and zero crossing scheme, are shown along the top. The deformable model recovered from these contours is shown in wireframe.

Equation 4.2. The resulting rigid and non-rigid motions are shown by the 3-D wireframe model overlaid on the original X-ray imagery at the bottom of each box. Execution time was approximately one second per frame on a standard Sun 4/330.

As can be seen, a good estimate of shape was obtained despite the presence of both rigid-body rotations and non-rigid deformations. Note that the 3-D model returns almost exactly to its original position and shape, an indication of the stability of the tracking process. Because only bounding contour information was available, deformations along the  $z$  axis are determined by finding the minimum stress state that still fits the bounding contour. The  $z$ -axis deformations cannot, therefore, be regarded as accurate but only as plausible.

# Chapter 5

## Displacement Maps and Physical Simulation

In many graphics and robotics applications, and especially in physical simulations, the ability to efficiently detect and characterize collisions and intersections is essential. Unfortunately, the polygon and spline representations normally employed in computer graphics are ill suited to this task. When using a polygon representation, for instance, the computational complexity of collision detection is  $O(nm)$  operations, where  $n$  is the number of polygons and  $m$  is the number of points to be considered after pruning via bounding box considerations [30]. As a consequence, collision detection is one of the most costly operations in many graphics applications [20], despite significant efforts to optimize algorithms for collision and intersection detection [4, 30].

In contrast, one can perform collision detection relatively efficiently when employing an implicit function representation (*e.g.*, spheres, swept solids, deformable superquadrics [6]) by making use of their inside-outside function. In each case, the computational complexity of this type of collision checking is only  $O(m)$  rather than  $O(nm)$  [32]. A more subtle but perhaps equally important advantage of this approach is that the collision surface may often be characterized analytically [24, 49], allowing more accurate simulation of multibody collisions.

Unfortunately, implicit function representations have not been sufficiently expressive for

general use. And, as was demonstrated in Chapter 2, we need to somehow supplement the overall description deformation modes describe with *displacement maps*, which can describe local details. The purpose of this chapter is to describe the overall geometric representation (i.e., deformed 3-D implicit functions + displacement maps = full detail model) in computer graphics language. We will describe the underlying geometry, methods for computing displacement maps, and then give an example of how physical simulations can be carried out easily within the framework of *modal analysis* — a method described in detail by Pentland and Williams in [39].

## 5.1 The Geometry: Generalized Implicit Functions

An implicit function representation defines a surface as a level set of a function  $f$ , most commonly the set of points for which  $f(\mathbf{x}) = 0$ . For instance, the inside-outside function we use for superquadric ellipsoids, before rotation, translation or deformation, is:

$$f(\mathbf{x}) = \left[ (x^{2/\epsilon_2} + y^{2/\epsilon_2})^{\epsilon_2/\epsilon_1} + z^{2/\epsilon_1} \right]^{\epsilon_1/2} - 1. \quad (5.1)$$

In practice we have found this better behaved than the standard superquadric inside-outside function, as it is more similar to the a normal  $L_2$  distance metric.

A solid defined in this way can be easily positioned and oriented, by transforming the implicit function:

$$\hat{\mathbf{x}} = \mathbf{M}\mathbf{x} + \mathbf{b} \quad (5.2)$$

where  $\mathbf{M}$  is a rotation matrix, and  $\mathbf{b}$  is a translation vector. Similarly, the implicit function's positioned and oriented inside-outside function becomes:

$$f(\mathbf{x}) = f(\mathbf{M}^{-1}(\hat{\mathbf{x}} - \mathbf{b})). \quad (5.3)$$

To detect a collision between a point  $\mathbf{x} = (x, y, z)$  and the volume bounded by this surface, one simply substitutes the coordinates of  $\mathbf{x}$  into the function  $f$ . If the result is negative, then the

point is inside the surface and a collision has occurred. Generalizations of this basic operation may be used to find line-surface intersections or surface-surface intersections [49]. Even degenerate contact geometries, such as vertex-edge intersections, can be handled by techniques such as described in [6, 24].

### 5.1.1 Deformations

As in Barr [5, 6], this basic set of functions can be generalized further by defining an appropriate set of global deformations  $\mathcal{D}$  with parameters  $\mathbf{u}$ . For particular values of  $\mathbf{u}$  the new deformed surface is defined using a deformation matrix  $\mathcal{D}_{\mathbf{u}}$ :

$$\hat{\mathbf{x}} = \mathbf{M}\mathcal{D}_{\mathbf{u}}\mathbf{x} + \mathbf{b} \quad (5.4)$$

where  $\hat{\mathbf{x}}$  is the position vector after rotation, deformation, and translation. Similarly, the inside-outside function becomes

$$f(\mathbf{x}) = f(\mathcal{D}_{\mathbf{u}}^{-1}\mathbf{M}^{-1}(\hat{\mathbf{x}} - \mathbf{b})). \quad (5.5)$$

This inside-outside function is valid as long as the inverse deformation  $\mathcal{D}_{\mathbf{u}}^{-1}$  exists. Thus by selecting a set of deformations that can be easily inverted, we can greatly expand the class of shapes that can be described using an implicit function representation.

In the ThingWorld modeling system [39] deformations are described by a 3 x 3 deformation matrix  $\mathcal{D}_{\mathbf{u}}$ , referred to as the *modal deformation matrix*, whose entries are polynomials which mimic the free vibration modes found in real objects. As a consequence, the linear superposition of these deformation polynomials allows accurate description of the dynamic, non-rigid behavior of real objects. For an example  $\mathcal{D}_{\mathbf{u}}$ , see the appendix.

### 5.1.2 Displacement Maps

The class of implicit functions can be generalized still further by defining the surface as the set of points for which  $f(\mathbf{x}) = d$  for some displacement function  $d(\eta, \omega)$ , where  $\eta, \omega$  are the point's coordinates in the surface's two-dimensional parametric space. We define  $\tilde{\mathbf{x}}(\eta, \omega)$  to be the displaced surface point before rotation, deformation or translation:

$$\tilde{\mathbf{x}} = \mathbf{x} + d\mathbf{n} \quad (5.6)$$

This will have the effect of displacing the surface along the surface normal in the original undeformed space before applying the deformation  $\mathcal{D}_{\mathbf{u}}$ .

In the ThingWorld system displacement maps are stored as two-dimensional floating point image arrays. In our examples, the displacement map function  $d(\eta, \omega)$  is defined by bilinear interpolation between point samples. If a smoother interpolation is desired, standard spline or pyramid-based multiresolution methods [12] can be employed.

Given a point  $\mathbf{x}(\eta, \omega)$  and normal  $\mathbf{n}(\eta, \omega)$  on the undeformed implicit surface, deformation transform  $\mathcal{D}_{\mathbf{u}}$ , and scalar displacement map function  $d(\eta, \omega)$ , we define  $\bar{\mathbf{x}}$  to be the position vector including deformations and displacement map:

$$\bar{\mathbf{x}} = \mathbf{M}\mathcal{D}_{\mathbf{u}}\tilde{\mathbf{x}} + \mathbf{b} = \mathbf{M}\mathcal{D}_{\mathbf{u}}(\mathbf{x} + d\mathbf{n}) + \mathbf{b} \quad (5.7)$$

We also need to define  $\bar{\mathbf{n}}$  to be the normal vector including deformations and displacement. To find  $\bar{\mathbf{n}}$ , it is first necessary to find  $\tilde{\mathbf{n}}$ , the surface normal for the displaced surface before deformation, and then apply deformations to get  $\bar{\mathbf{n}}$ . The normal  $\tilde{\mathbf{n}}(\eta, \omega)$  is derived by taking the cross product of the partial derivatives for the undeformed displaced surface function,  $\tilde{\mathbf{x}}(\eta, \omega)$ :

$$\frac{\partial \tilde{\mathbf{x}}}{\partial \eta} = \frac{\partial \mathbf{x}}{\partial \eta} + d \frac{\partial \mathbf{n}}{\partial \eta} + \mathbf{n} \frac{\partial d}{\partial \eta}$$

and

$$\frac{\partial \tilde{\mathbf{x}}}{\partial \omega} = \frac{\partial \mathbf{x}}{\partial \omega} + d \frac{\partial \mathbf{n}}{\partial \omega} + \mathbf{n} \frac{\partial d}{\partial \omega}. \quad (5.8)$$

In our current implementation, the partials are calculated by finite differences.

The inside-outside function associated with Equation 5.7 is then:

$$f(\mathbf{x}) = f(\mathcal{D}_{\mathbf{u}}^{-1} \mathbf{M}^{-1}(\tilde{\mathbf{x}} - \mathbf{b}) - d\mathbf{n}). \quad (5.9)$$

It is difficult to evaluate a displacement mapped and deformed inside-outside function because we cannot know  $d$  or  $\mathbf{n}$  in Equation 5.9 before we know  $\mathbf{x}$ , and *vice versa*. We therefore need a parametric projection function  $P(\tilde{\mathbf{x}}) = (\eta, \omega)$ , which can project an undeformed, displacement-mapped  $\tilde{\mathbf{x}}$  point back onto the original, undisplaced implicit surface. These parameters can then be used to determine  $\mathbf{n}$ ,  $\mathbf{x}$ , and  $d$ . The projection should be *normal projection* since the displacement map displaces surface points along the surface normal.

For example, the projection function  $P(\tilde{\mathbf{x}}) = (\eta, \omega)$  for superquadric ellipsoids is computed as follows. We first find  $\omega$  by observing:

$$\frac{\tilde{y}}{\tilde{x}} = \frac{\cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega}{\cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega} = \tan^{\epsilon_2} \omega \quad (5.10)$$

where  $(\tilde{x}, \tilde{y}, \tilde{z})^T = \tilde{\mathbf{x}}$  is the undeformed, displaced surface point. From Equation 5.10, we see that  $\omega = \text{atan}^{1/\epsilon_2}(\tilde{y}/\tilde{x})$ . The remaining parameter,  $\eta$ , is determined by either  $\eta = \text{atan}^{1/\epsilon_1}((\tilde{z} \cos^{\epsilon_2} \omega)/\tilde{x})$  or  $\eta = \text{atan}^{1/\epsilon_1}((\tilde{z} \sin^{\epsilon_2} \omega)/\tilde{y})$  depending on whether  $\tilde{x}$  or  $\tilde{y}$  is larger.

### 5.1.3 An Example

Figure 5-1 shows three frames from a physically-based animation in which three seashell-like shapes drop through a viscous medium (*e.g.*, seawater), hit the sea bottom, bump into each other, and then come to rest. The simulations were conducted using the technique of modal dynamics as implemented in the ThingWorld system [39].

The seashell shapes were modeled as superquadric ellipsoids with displacement maps. Each displacement map consisted of a 100 x 100 uniformly spaced grid generated by combinations of sines and cosines. The shells were polygonalized for display and simulation purposes —

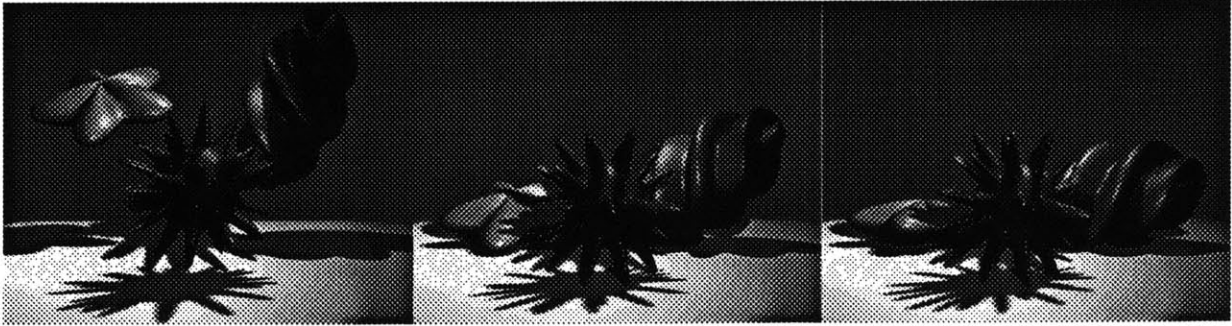


Figure 5-1: Three frames from a physically-based animation in which seashell-like shapes drop through water and come to rest.

approximately 2300 polygons for the spike seashell, and 576 polygons for the other seashells. Bounding boxes were also computed for the objects. During the simulation, if these bounding boxes crossed, then polygon vertices were plugged into the offending objects' inside/outside functions to test for collisions.

Execution time for the three active objects was 0.05 seconds per time step during the initial frames of the animation (before any contact), and 0.081 seconds per time step during the final few frames, when the three seashells were colliding with the seabed (which is not planar), and the shells were colliding with each other. Subtracting the pre-contact time from the execution time during contact, we find it took approximately 0.031 seconds per time step for contact detection and calculation of the non-rigid dynamics. Contact detection, physical simulation, and geometric updates were computed on a Sun 4/330, with a TAAC board performing rendering.

## 5.2 Computing a Displacement Map

If there are more degrees of freedom in the data points than in the deformation parameters, the deformed model will not generally pass through the data points exactly (*i.e.*,  $f(\mathcal{D}_{\mathbf{u}}^{-1}\mathbf{M}^{-1}(\bar{\mathbf{x}} - \mathbf{b})) \neq 0$ ). A more accurate approximation to the data points can be obtained by incorporating these residual differences into a displacement map. This is done by solving for the displacement



map value  $d(\eta, \omega)$  that yields  $\bar{\mathbf{x}} = M\mathcal{D}_{\mathbf{u}}(\mathbf{x} + d\mathbf{n}) + \mathbf{b}$  for some point  $\mathbf{x}$  on the undeformed implicit surface. The final result is a generalized implicit function representation that normally provides an exact fit to the set of initial data points, and provides a smoothly interpolated surface between those points — except in certain degenerate cases, such as when Nyquist criteria are not satisfied.

To compute the displacement map each data point  $\bar{\mathbf{x}}$  is subjected to the inverse deformation  $\mathcal{D}_{\mathbf{u}}^{-1}$  to obtain  $\tilde{\mathbf{x}}$ , the point in the undeformed space. Next, we project  $\tilde{\mathbf{x}}$  along the surface normal using  $P(\tilde{\mathbf{x}})$  to obtain its two-dimensional parametric space coordinate  $(\eta, \omega)$ . Finally, we compute the undeformed point's normal distance to the undeformed implicit surface by substituting its coordinates into the surface's inside-outside function:

$$d(\eta, \omega) = f(\tilde{\mathbf{x}}). \quad (5.11)$$

In our software system, ThingWorld, displacement maps are represented by a regularly spaced grid in the surface's parametric space. Thus as each point is projected and its displacement determined, the result is spread to nearest grid points by Gaussian weighted averaging. This interpolation method works well when data points are fairly dense, however when there are only a few data points more sophisticated interpolation methods must be used [18].

### 5.2.1 Example: Modeling a Head

Figure 5-2 shows four frames from a physically-based animation in which a jello-like head is struck with a wooden mallet. The head shape was recovered by converting 360° laser range data of a human head that was scanned by Cyberware at SIGGRAPH '90.

The model for the head was recovered with the method described above. It took about 5 seconds on a Sun 4/330 to recover the deformations and displacement map from 2500 data points. A color map had to be computed at higher resolution — about 10,000 data points were used. The color map was computed by projecting the color points onto the surface using  $P(\mathbf{x})$  and Gaussian interpolation for smoothing.

For the simulation, the head was sampled with 7200 polygons. The other objects had about 600 polygons each. Execution time for the three active objects averaged 0.13 seconds per time step during the collision, with approximately 0.05 seconds per time step attributable to contact detection and calculation of the non-rigid dynamics. The greater execution times for this example are primarily due to the large amount of detail in the head model.

### 5.3 Storage and Compression

Since displacement maps are stored as large floating point image arrays (usually 100x100 or 256x256), there is some concern about the amount of space they occupy in memory or on disk. To get around this problem, we propose using a pyramid-based compression and interpolation scheme which significantly reduces the amount of space needed to store a displacement map.

Figure 5-3 shows an example Gaussian pyramid which was computed for a recovered head's displacement map. The pyramid shown has three levels. At each level, the corresponding head and its rectangular displacement map are shown (whites signify peaks, blacks signify valleys). The smallest head, and its corresponding level in the pyramid, represent the lowest frequency information about bumpiness; at each higher level, the map includes higher and higher frequency (resolution) information about shape.

Among other things, the Gaussian pyramid's hierarchy of scale is particularly useful for image interpolation between scales. For image compression, we need a band-pass pyramid, lest we get duplication of information across pyramid levels — Laplacian and QMF pyramids provide this. We use a QMF-based (Quadrature Mirror Filter) compression algorithm, since they offer more effective compression rates. Since storing a displacement map can be quite memory exhausting (up to a megabyte of memory, uncompressed), it would behoove us to at least provide an example of how to store them economically. For our example, we use EPIC, a QMF-based image compression package developed by Eero Simoncelli and Ted Adelson at the Media Lab.

Figure 5-3 shows an experiment in which we compressed a displacement map (256 x 256)

with EPIC. The figure shows the original head, and then its compressed versions at various compression rates. The best we get is compressing the image to 2613 bytes; this represents a compression rate of more than 25:1. It was possible to compress further, but with greater loss. Up to a 50:1 compression rate (34 dB) is visually acceptable. Due to a bug in the package, it was not possible to get a *lossless* compression of the image, though lossless compression would be possible within the QMF pyramid framework. When the bug is fixed, the compression rate would be about 25:1 for the experiment shown.

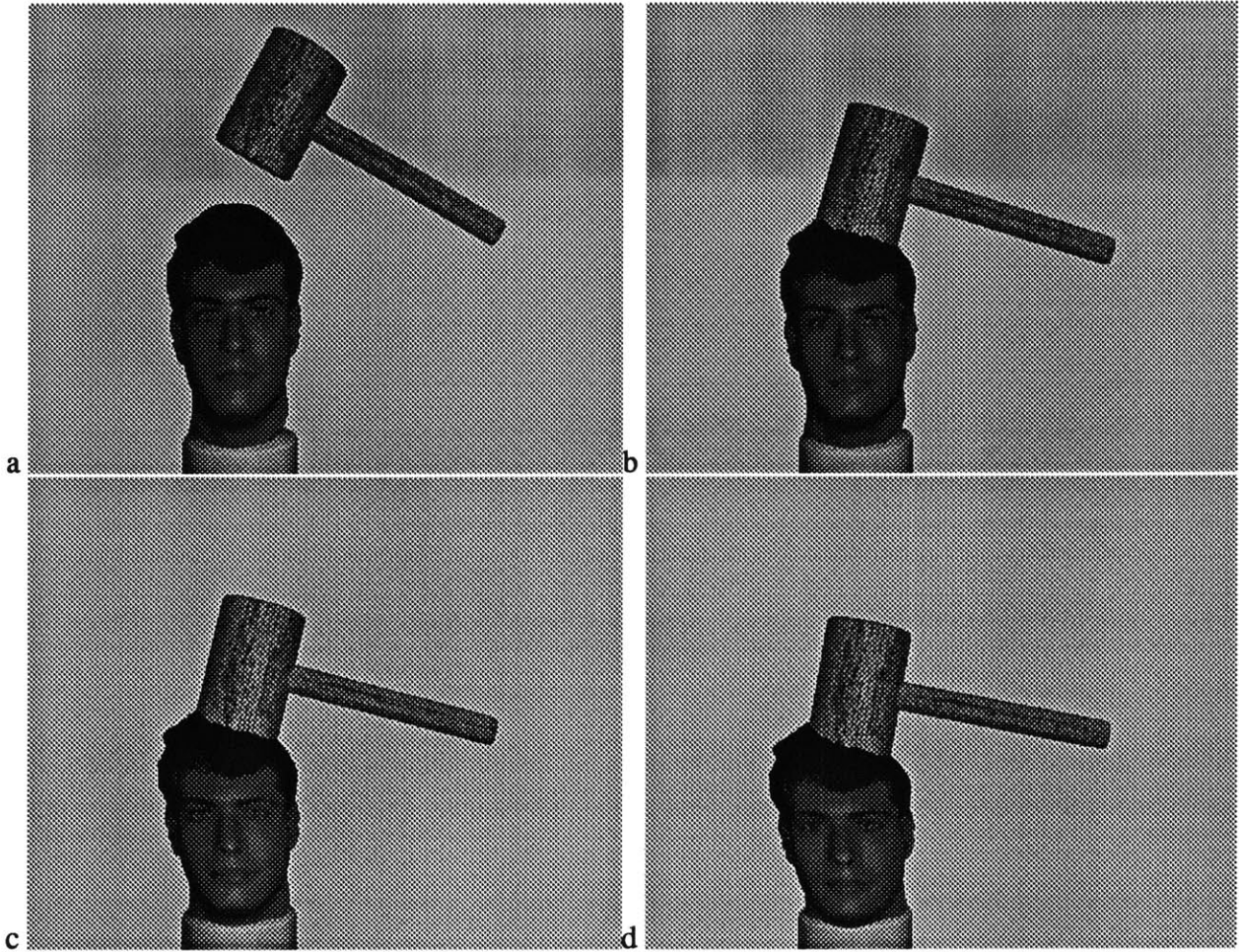


Figure 5-2: Four frames from a physically-based animation. Sun 4 execution times averaged 0.13 seconds per time step for each object, of which approximately 0.05 seconds is attributable to contact detection and non-rigid dynamics.

---

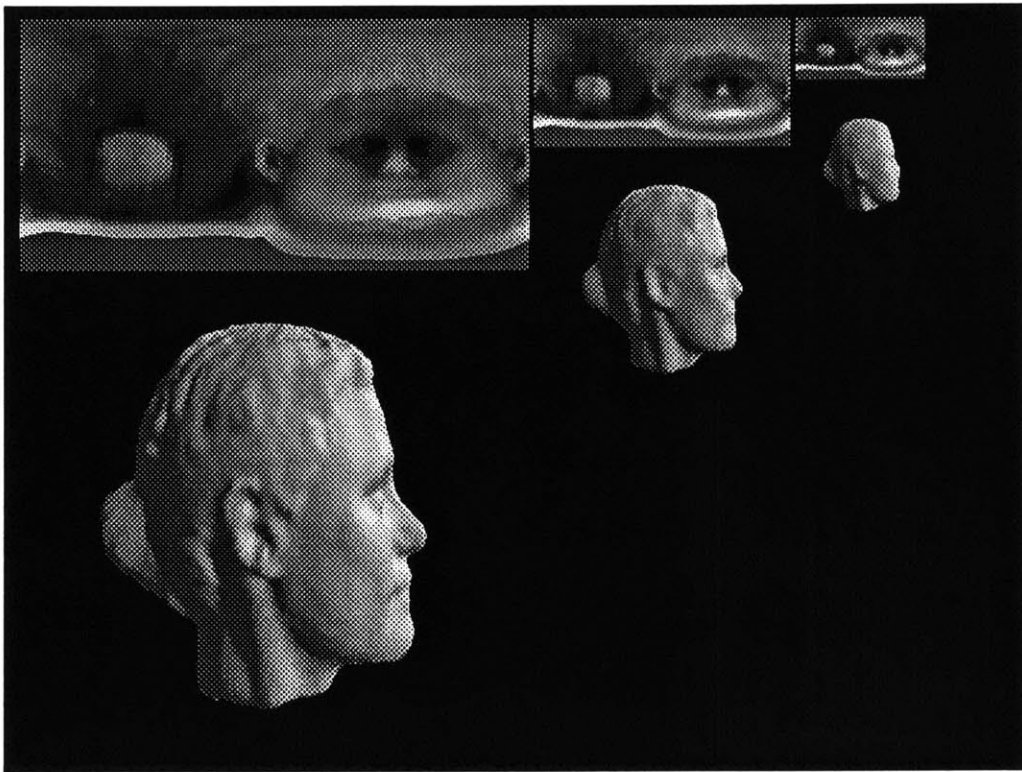
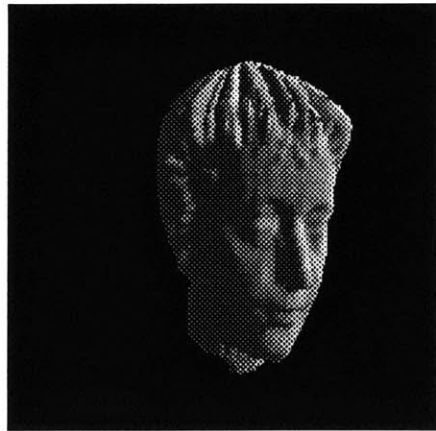


Figure 5-3: Displacement map pyramid, shown at various scales.



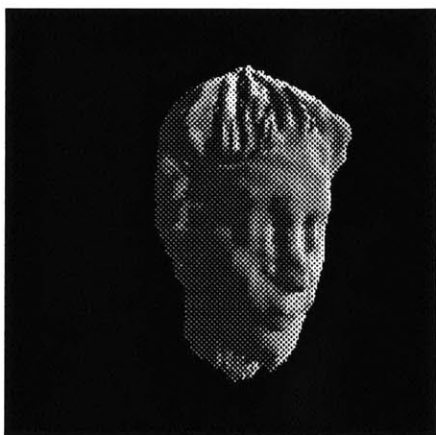
Original: 65537 bytes



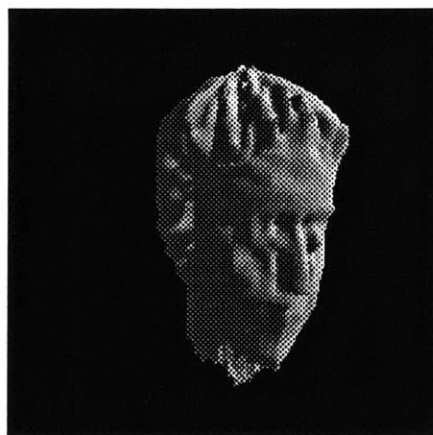
2613 bytes (36 dB)



1422 bytes (34 dB)



1073 bytes (33 dB)



901 bytes (32 dB)

Figure 5-4: Using Epic, a pyramid-based image coding package to compress a recovered head's displacement map. Signal to noise ratios shown were computed for the actual displacement maps in 2D.

# Chapter 6

## Conclusion

We have described a closed-form solution for recovering a physically-based 3-D solid model from 3-D sensor measurements. In our current implementation we typically use 30 deformation modes, so that given as few as 11 independent 3-D sensor measurements the solution is overconstrained, and therefore unique except for rotational symmetries and degenerate conditions.

We have introduced the concept of *displacement maps* for storing error residuals, so that the resulting model is a more complete representation of the original sensor data. The displacement map methodology is especially appealing because the map can be stored like an intensity image. Some useful side-effects of this are:

- Displacement maps can be compressed via standard image compression techniques.
- Displacement maps can be easily pyramid encoded for multiscaling (Laplacian pyramids), or frequency coded (Wavelet or QMF pyramids). As a result, the level of information reduction can be chosen on the fly, and not decided at model recovery time.
- The representation is especially well-suited for physical simulation. Since the resulting part representation is still an implicit function (displacement maps merely offset the underlying deformed function) intersection and inside/outside measurements are easy to compute.

There are some additional benefits of this multiscale representation that could be explored in future work. For instance, for object recognition and data base search, we could first compare the lower-order modal fit. If a low-order match is found, then we go on to compare low-frequency displacement maps, and then move up the pyramid as required. Low-order displacement maps could also be used for 3-D segmentation. If the volume under a “bump” in a displacement map is above a certain threshold, then the bump can be chopped off, and a new deformable part recovered. This threshold could be slowly decreased, revealing a hierarchy of parts. The Laplacian pyramid could be useful in this application.

Displacement maps are layered over a parametric description for overall shape: deformation modes. This underlying modal description for the recovered 3-D shape is unique except for rotational symmetries; thus we may efficiently measure the similarity of different shapes by simply calculating normalized dot products between the mode values  $\bar{U}$  of various objects. Such comparisons may be made position, orientation and/or size independent by simply excluding the first seven mode amplitudes. Thus the modal representation seems likely to be useful for applications such as object recognition and spatial database search.

We have also shown how to extend this technique to effectively recover and recognize models from contour data using a variant of weighted least squares. These deformation mode estimates can be integrated over time by use of an extended Kalman filter, allowing us to recover a stable estimate of both 3-D shape and 3-D velocity. Examples using X-ray data were presented. For a more in-depth discussion of tracking non-rigid motion using the modal method, see [22].

The two major weaknesses in our current method are initial estimation of object orientation and the need for segmenting data into parts. Currently we use a standard method-of-moments for estimation of object orientation, and a minimum description length technique to for segmentation [14]. For simple examples these techniques produce accurate, stable descriptions of shape; however for more complex scenes more sophisticated techniques will be required.

And finally, Pentland and Williams [39] presented a method for combining implicit function representations with modal dynamics to obtain near-real-time non-rigid dynamic simulations;



---

this thesis extends this method by developing a generalized implicit function representation that is sufficiently powerful to describe many of the objects commonly used in computer graphics. The representation can therefore be used to reduce the cost of contact detection and physical simulation in robotics, path planning, and computer graphics applications.

# Bibliography

- [1] R. Arnheim. *Art and Visual Perception*. University of California Press, 1974.
- [2] N. Ayache and O. D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Trans. Robotics and Automation*, 5(6):804–819, 1989.
- [3] R. Bajcsy and F. Solina. Three Dimensional Object Representation Revisited. In *Proc. First International Conference on Computer Vision*, pages 231–240, London, England, 1987.
- [4] D. Baraff. Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation. *Computer Graphics*, 24(4):19–28, 1990.
- [5] A. Barr. Global and Local Deformations of Solid Primitives. *Computer Graphics*, 18(3):21–30, 1984.
- [6] A. Barr. Superquadrics and Angle-Preserving Transforms. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.
- [7] I. Biederman. Recognition-by-Components: A Theory of Human Image Understanding. *Psychological Review*, 94(2):115–147, 1987.
- [8] T. Binford. Visual Perception by Computer. *Presented at The IEEE Conference on Systems and Control*, December 1971.
- [9] A. Blake and A. Zisserman. *Visual Reconstruction*. M.I.T. Press, 1987.

- [10] T. Boult and A. Gross. Recovery of Superquadrics from Depth Information. *Proc. of the AAAI Workshop on Spatial Reasoning and Multi-Sensor Fusion*, 128–137, 1987.
- [11] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):90–99, January 1986.
- [12] P. J. Burt and E. H. Adelson. A Multiresolution Spline With Application to Image Mosaics. *ACM Transactions on Graphics*, 2(4):217–236, 1983.
- [13] T. Darrell. *Integrated Descriptions for Vision*. Master's thesis, MIT Media Lab, 1990.
- [14] T. Darrell, S. Sclaroff, and A. Pentland. Segmentation by Minimal Description. In *Proc. Third International Conference on Computer Vision*, December 1990.
- [15] I. Essa. *Contact Detection, Collision Forces and Friction for Physically-Based Virtual World Modeling*. Master's thesis, Dept. of Civil Engineering, M.I.T., 1990.
- [16] O. D. Faugeras, N. Ayache, and B. Faverjon. Building visual maps by combining noisy stereo measurements. In *IEEE Trans. Robotics and Automation*, San Francisco, CA, April 1986.
- [17] F. Ferrie, J. Lagarde, and P. Whaite. *Darboux Frames, Snakes, and Super-Quadrics: Geometry from the Bottom-Up*. Technical Report, McGill University Computer Vision and Robotics Lab, 1989.
- [18] T. Foley, D. Lane, and G. Nielson. Interpolation of Scattered Data on Closed Surfaces. *Computer Aided Geometric Design*, 7:303–312, 1990.
- [19] Bernard Friedland. *Control System Design*. McGraw-Hill, 1986.
- [20] J. K. Hahn. Realistic Animation of Rigid Bodies. *Computer Graphics*, 22(4):299–308, 1988.
- [21] D. Hoffman and W. Richards. Parts of Recognition. In *From Pixels to Predicates*, A. Pentland (ed.), 1985.

- [22] B. Horowitz. *Syntactic and Semantic Image Representations for Computer Vision*. Master's thesis, M.I.T. Media Laboratory, 1991.
- [23] K. Bathe. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall, 1982.
- [24] D. Kalra and A. H. Barr. Guaranteed Ray Intersections with Implicit Surfaces. *Computer Graphics*, 23(3):297–306, 1989.
- [25] J. Koenderink. *Solid Shape*. M.I.T. Press, 1990.
- [26] M. Leyton. A Processs-Grammar for Shape. *Artificial Intelligence*, 34:213–247, 1988.
- [27] M. Leyton. Perceptual Organization as Nested Control. *Biological Cybernetics*, 51:141–153, 1984.
- [28] D. Marr and K. Nishihara. Representation and Recognition of the Spatial Organization of Three-dimensional Shapes. In *Proc. of the Royal Society - London B*, 1978.
- [29] R. Mohan and R. Nevatia. Using Perceptual Organization to Extract 3D Structures. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(11):1121–1139, November 1989.
- [30] M. Moore and J. Wilhelms. Collision Detection and Response for Computer Animation. *Computer Graphics*, 22(4):289–298, 1988.
- [31] A. Pentland. Automatic Extraction of Deformable Part Models. *International Journal of Computer Vision*, 107–126, 1990.
- [32] A. Pentland. Computational Complexity Versus Virtual Worlds. *Computer Graphics*, 24(2):185–192, 1990.
- [33] A. Pentland. Perceptual Organization and Representation of Natural Form. *Artificial Intelligence*, 28(3):293–331, 1986.
- [34] A. Pentland. Recognition by Parts. In *Proc. First International Conference on Computer Vision*, pages 612–620, London, England, 1987.

- [35] A. Pentland, I. Essa, M. Friedmann, B. Horowitz, and S. Sclaroff. The Thingworld Modeling System: Virtual Sculpting by Modal Forces. *Computer Graphics*, 24(2):143–144, 1990.
- [36] A. Pentland and B. Horowitz. Recovery of Non-Rigid Motion and Structure. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13, to appear in July 1991. Special Issue on Physically-Based Modeling.
- [37] A. Pentland, B. Horowitz, and S. Sclaroff. Non-Rigid Motion and Structure from Contour. In *submitted to Proc. IEEE Workshop on Visual Motion*, October 1991.
- [38] A. Pentland and S. Sclaroff. Closed-Form Solutions for Physically-Based Shape Modeling and Recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13, to appear in July 1991. Special Issue on Physically-Based Modeling.
- [39] A. Pentland and J. Williams. Good Vibrations : Modal Dynamics for Graphics and Animation. *Computer Graphics*, 23(4):215–222, 1989.
- [40] S. Sclaroff and A. Pentland. Closed-Form Solutions for Physically-Based Shape Modeling and Recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, to appear in June 1991.
- [41] S. Sclaroff and A. Pentland. Generalized Implicit Functions for Computer Graphics. In *Computer Graphics*, to appear in July 1991.
- [42] F. Solina and R. Bajcsy. Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(2):131–147, 1990.
- [43] Franc Solina. *Shape Recovery and Segmentation with Deformable Part Models*. PhD thesis, University of Pennsylvania, 1987.

- 
- [44] D. Terzopoulos and D. Metaxas. Deformable Superquadrics: Dynamic Models with Local and Global Deformations. In *Proc. Third International Conference on Computer Vision*, December 1990.
- [45] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically Deformable Models. *Computer Graphics*, 21(4):205–214, 1987.
- [46] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion. *Artificial Intelligence*, 36:91–123, 1988.
- [47] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-Seeking Models for 3-D Object Reconstruction. In *Proc. First Conference on Computer Vision*, pages 269–276, London, England, December 1987.
- [48] D’Arcy Thompson. *On Growth and Form*. Cambridge University Press, 1988.
- [49] B. von Herzen, A. Barr, and H. Zatz. Geometric Collisions for Time-Dependent Parametric Surfaces. *Computer Graphics*, 24(4):39–48, 1990.
- [50] P. Whaite and F. Ferrie. From Uncertainty to Visual Exploration. In *Proc. Third International Conference on Computer Vision*, December 1990.
- [51] P. Winston, T. Binford, B. Katz, and M. Lowry. Learning Physical Description from Functional Descriptions, Examples, and Precedents. In *Proc. of AAAI*, pages 433–439, 1983.

# Appendix A

## Programming Shortcuts

When presented with this approach, some researchers have expressed concern over the expense of calculating the mass and stiffness matrices  $\mathbf{M}$  and  $\mathbf{K}$ , as well as the cost of calculating their low-order eigenvectors  $\Phi$ . However, efficient FEM codes for producing  $\mathbf{K}$  have been commercially available for over twenty years, and many are in the public domain. Similarly, given a stiffness matrix  $\mathbf{K}$  the low-frequency eigenvectors can then be efficiently obtained (in ascending order) via the power method. Moreover, all of these calculations will typically be performed off-line as a precomputation step.

However, for many vision (and some graphics) applications even this level of effort is sometimes unnecessary. This is because (1) it is often unnecessary to have detailed estimates of the material properties, as these are either unknown or unimportant, and (2) the low-frequency eigenvectors are determined by the low-order moments of inertia, so that compact bodies of similar size have similar eigenvectors. Thus for vision applications with relatively noise-free data it appears that it is sufficient to use a single, extremely elastic FEM model of the base shape (in our case an ellipsoid) that is to be fit to sensor data.

The following sections explain how these two facts can be utilized to efficiently obtain rough-and-ready approximations to the  $\Phi$ ,  $\tilde{\mathbf{M}}$ , and  $\tilde{\mathbf{K}}$  matrices for arbitrary samplings of an object's surface.

## Calculating a Polynomial Characterization of the Modes

The modes of an object can be characterized by 3-D polynomial functions whose coefficients are determined by a linear regression of polynomials with  $m$  terms in appropriate powers of  $x$ ,  $y$ , and  $z$ , against the  $n$  triples of  $x$ ,  $y$  and  $z$  that compose  $\phi_{i*}$ , a  $3n \times 1$  vector containing the elements of the  $i^{\text{th}}$  column of  $\Phi$ ,

$$\alpha = (\beta^T \beta)^{-1} \beta^T \phi_{i*} \quad , \quad (\text{A.1})$$

where  $\alpha$  is an  $m \times 1$  matrix of the coefficients of the desired deformation polynomial,  $\beta$  is an  $3n \times m$  matrix whose first column contains the object-centered coordinates of the nodes  $(x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_n, y_n, z_n)^T$ , and whose remaining columns consist of the modified versions of the nodal coordinates where the  $x$ ,  $y$ , and/or  $z$  components have been raised to various powers. See reference [39] for more details.

By linearly superimposing the various deformation mappings, one can obtain an accurate accounting of the object's non-rigid deformation. In the ThingWorld modeling system [39, 35] the set of polynomial deformations is combined into a 3 by 3 matrix of polynomials,  $\mathcal{D}_u$ , that is referred to as the *modal deformation matrix*. This matrix transforms pre-deformation point positions  $\mathbf{X} = (x, y, z)^T$  into the deformed positions

$$\mathbf{X}^* = \begin{pmatrix} d_{00} & d_{01} & d_{02} \\ d_{10} & d_{11} & d_{12} \\ d_{20} & d_{21} & d_{22} \end{pmatrix} \mathbf{X} \quad . \quad (\text{A.2})$$



A simple version of  $\mathcal{D}_{\mathbf{u}}$  that is derived from a 20 node element is the following:

$$\begin{aligned}
 d_{00} &= u_6 + yu_{12} + zu_{15} - (u_{13} + u_{16})\text{sgn}(x) - u_{14} - u_{17}, \\
 d_{01} &= u_{11} + 2y(u_{13} + \text{sgn}(x)u_{14}), \\
 d_{02} &= u_{10} + 2z(u_{16} + \text{sgn}(x)u_{17}), \\
 d_{10} &= u_{11} + 2x(u_{19} + \text{sgn}(y)u_{20}), \\
 d_{11} &= u_7 + xu_{18} + zu_{21} - (u_{19} + u_{22})\text{sgn}(y) - u_{20} - u_{23}, \\
 d_{12} &= u_9 + 2z(u_{22} + \text{sgn}(y)u_{23}), \\
 d_{20} &= u_{10} + 2x(u_{25} + \text{sgn}(z)u_{26}), \\
 d_{21} &= u_9 + 2y(u_{28} + \text{sgn}(z)u_{29}), \\
 d_{22} &= u_8 + xu_{24} + yu_{27} - (u_{25} + u_{28})\text{sgn}(z) - u_{26} - u_{29}.
 \end{aligned}$$

The parameters  $u_i$  are the amplitudes of the various free vibration modes, and have simple, intuitive meanings. The lowest frequency modes are the rigid-body modes of translation ( $u_0$  through  $u_2$ ) and rotation ( $u_3$  through  $u_5$ ) which specify the values of  $\mathbf{b}$  and  $\mathbf{M}$ . The next-lowest frequency modes are smooth, whole-body deformations that leave the center of mass and rotation fixed:  $u_6$  through  $u_8$  control the x, y, and z radii,  $u_9$  through  $u_{11}$  are shears about the x, y, and z axes, and  $u_{12+3j}$  through  $u_{12+3j+2}$  are tapering, bending, and pinching around the  $j^{\text{th}}$  pairwise combination of the x, y, and z axes. Note that because the rigid body modes are calculated in the object's coordinate system, they must be rotated to global coordinates before being integrated with the remainder of any dynamic simulation.

### Approximate Calculation of $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{K}}$

We can make use of the above mode characterization to numerically approximate the matrix  $\Phi$  for any point sampling of the surface. In essence, we are precalculating how energy in each of the various modes displaces of a set of points on the surface. We then store this mode-displacement relationship into the  $\Phi$  matrix, thus approximating how these points distribute load to the original nodal points and the various vibration modes.

Once point correspondences have been established, we can proceed with fitting. The task

will be to deform the original undeformed points,  $\mathbf{X}$ , to their desired positions,  $\bar{\mathbf{X}}$ . At the end of this process, we will have recovered the deformed implicit function which best fits the data points.

To begin with, the effect of each of the  $m$  deformation parameters  $u_i$  in  $\mathcal{D}_{\mathbf{u}}$  on the position of the undeformed points,  $\mathbf{X}$  is calculated, to obtain a  $m \times 3n$  matrix  $\Phi$  whose  $i^{\text{th}}$  column  $\phi_{i*}$  is:

$$\phi_{i*} = \left( \frac{\partial x_1}{\partial u_i}, \frac{\partial y_1}{\partial u_i}, \frac{\partial z_1}{\partial u_i}, \dots, \frac{\partial x_n}{\partial u_i}, \frac{\partial y_n}{\partial u_i}, \frac{\partial z_n}{\partial u_i} \right)^T . \quad (\text{A.3})$$

The matrix  $\Phi$  can be computed by finite differences — *i.e.*, analytically, or by applying a small amount of each deformation  $u_i$  and measuring the resulting change in the coordinates of each point. In the ThingWorld modeling system the deformations used are the object's free vibration modes, so that the columns of  $\Phi$  define a coordinate transformation that diagonalizes the object's finite element equations. This allows the object's rigid and non-rigid dynamics to be simulated very inexpensively, as described in reference [39].

The matrix  $\Phi$  is the Jacobian of  $\mathcal{D}_{\mathbf{u}}$  at each point in  $\mathbf{X}$ , and so may be used in a modified Newton-Raphson iteration to obtain the minimum RMS error estimate of deformation parameters  $\mathbf{u}$  as follows:

$$\mathbf{u}^{k+1} = \Phi^{-1} (\bar{\mathbf{X}} - \mathbf{X}^k) + \mathbf{u}^k \quad (\text{A.4})$$

where  $\mathbf{X}^k$  is the projection of the data points on the surface defined by the deformations  $\mathbf{u}^k$  at iteration  $k$ , and  $\mathbf{u}^0 = \mathbf{0}$ ,  $\mathbf{X}^0 = \mathbf{X}$ . We have found that a single iteration is often sufficient to obtain a satisfactory estimate of the deformation parameters  $\mathbf{u}$ . Because  $\Phi$  is usually not square, use of the pseudoinverse  $\Phi^{-1} = (\Phi\Phi^T)^{-1}\Phi^T$  is required; as  $(\Phi\Phi^T)$  is only an  $m \times m$  matrix, this calculation is inexpensive.

We can use this method to efficiently obtain a sampling of the surface that matches the projection of the sensor data onto the surface, thus considerably simplifying the fitting process. Even when iterating the solution to find better virtual spring attachment points, the distances between nodes and projected data positions will be small, and so simple bilinear interpolation can be used to distribute the data forces to the surrounding three surface points without

incurring significant error.

Given a  $\Phi$  with nodes evenly sampled throughout the body, we can use the common assumption that the object's mass is distributed equally among the sampled points to obtain

$$\tilde{\mathbf{M}} = \Phi^T \mathbf{M} \Phi = \Phi^T m \mathbf{I} \Phi \quad (\text{A.5})$$

where  $m$  is the mass assigned to each sample point. The modal stiffnesses  $\tilde{k}_i$  are more difficult to estimate; however, given that detailed material properties are unimportant, one may simply assign a stiffness proportional to  $\tilde{m}_i$  for the non-rigid modes  $\tilde{u}_i$ , and a stiffness of 0.0 for the rigid-body modes  $\tilde{k}_i$ ,  $1 \leq i \leq 6$ .