
PHYSICS-BASED DEFORMABLE MODELS

Applications to Computer Vision,
Graphics and Medical Imaging

**THE KLUWER INTERNATIONAL SERIES
IN ENGINEERING AND COMPUTER SCIENCE**

PHYSICS-BASED DEFORMABLE MODELS

Applications to Computer Vision, Graphics and Medical Imaging

Dimitris N. Metaxas

*Department of Computer and Information Science
University of Pennsylvania
Philadelphia, Pennsylvania, USA*

SPRINGER SCIENCE+BUSINESS MEDIA, LLC

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

ISBN 978-1-4613-7909-6 ISBN 978-1-4615-6335-8 (eBook)
DOI 10.1007/978-1-4615-6335-8

Copyright © 1997 Springer Science+Business Media New York
Originally published by Kluwer Academic Publishers, New York in 1997
Softcover reprint of the hardcover 1st edition 1997

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher, Springer Science+Business Media, LLC.

Printed on acid-free paper.

**Dedicated To My Parents
Ada and Nikolas**

CONTENTS

PREFACE	xi
0.1 Summary of Contents	xii
1 INTRODUCTION	1
1.1 Illustrative Examples of Modeling and Estimation	4
1.2 Chapter Outline	13
2 GEOMETRY OF DEFORMABLE MODELS	17
2.1 Related Work	17
2.2 Hybrid Deformable Models	20
3 KINEMATICS AND DYNAMICS	27
3.1 Kinematics	28
3.2 Dynamics	30
3.3 Choosing the Order of the Motion Equations	36
4 FINITE ELEMENT IMPLEMENTATION	39
4.1 Choosing the Appropriate Elements	40
4.2 Various Model Tessellations	40
4.3 C^0 Elements	42
4.4 C^1 Triangular Elements	47
4.5 Approximation of the Lagrange Equations	48
5 APPLIED FORCES	53
5.1 Computer Vision and Medical Imaging Applications	53
5.2 Computer Graphics Applications	60
5.3 Force-Based Estimation	63
6 MODEL IMPLEMENTATION	65
6.1 Integrating the Motion Equations	65
6.2 Model Initialization	66
6.3 Computer Vision Experiments	67
6.4 Computer Graphics Experiments	71

7	CONSTRAINED NONRIGID MOTION	77
7.1	Holonomic Constraints and Lagrange Multipliers	77
7.2	Stabilized Constraints	79
7.3	Fast Constraint Force Computation for Multibody Objects	80
7.4	Experiments with Constraints	84
8	SHAPE AND NONRIGID MOTION ESTIMATION	87
8.1	Recursive Estimation	88
8.2	Kalman Filter Implementation	89
8.3	Recursive Estimation of Shape and Nonrigid Motion	90
9	MULTI-LEVEL SHAPE REPRESENTATION	95
9.1	Related Work	96
9.2	Deformable Models: Geometry and Dynamics	97
9.3	Locally Adaptive Finite Elements	97
9.4	Summary of Model Fitting to Range Data	103
9.5	Experiments	104
10	TOPOLOGICALLY ADAPTIVE MODELS BASED ON BLENDING	111
10.1	Related Work	112
10.2	Blended shapes	113
10.3	Reconstruction and evolution	118
10.4	Experiments	120
11	INTEGRATION OF QUALITATIVE SEGMENTATION AND PBM METHODS	125
11.1	Related Work	126
11.2	Qualitative Model Extraction	127
11.3	Quantitative Shape and Motion Recovery	131
12	MOTION-BASED PART SEGMENTATION AND TRACKING	149
12.1	Related Prior Work	151
12.2	Deformable Models: Extensions	151
12.3	Inferring structure in 2D	158
12.4	Two-dimensional human body model acquisition	169
12.5	Three-dimensional Human Body model acquisition	171
12.6	Human Body Tracking	175
12.7	Experimental Results	178

13 VOLUMETRIC ANALYSIS OF THE LEFT VENTRICULAR WALL MOTION FROM MRI-SPAMM	191
13.1 Related Work	192
13.2 Data Extraction Techniques for Cardiac Motion Studies	194
13.3 Volumetric Deformable Models with Parameter Functions	198
13.4 Model Force Computation	200
13.5 Model Parameters	204
13.6 Implementation of Model Fitting Procedure	208
13.7 Experimental Results	211
14 VISUALIZING RESPIRATORY MECHANICS BASED ON ANATOMICAL AND PHYSIOLOGICAL MODELING	219
14.1 Related Work	220
14.2 Basic Anatomy and Physiology	220
14.3 Methods	222
14.4 Results	229
15 Recursive Dynamics and Adaptive Control for Animating Articulated Figures	233
15.1 System Description	236
15.2 Efficient Forward Dynamics	238
15.3 Collision handling	239
15.4 Dynamic Control	243
15.5 Results	248
16 Animating Liquids	253
16.1 Navier-Stokes Equations	255
16.2 Solving the Navier-Stokes equations	256
16.3 Tracking fluid position	261
16.4 Buoyancy	265
16.5 Summary of the Navier-Stokes Algorithm	266
16.6 Control	267
16.7 Examples	268
17 CONCLUSIONS	273
A	277
A.1 Derivation of B matrix	277
A.2 Integration Rules	278

REFERENCES	281
INDEX	305

PREFACE

The development and use of Physics-Based Modeling (PBM) methods and techniques by many researchers [16, 17, 18, 25, 36, 42, 44, 51, 61, 76, 88, 116, 131, 136, 139, 191, 175, 192, 211, 189, 250, 273, 215, 222, 291] has made it possible to address successfully, difficult problems in computer vision (e.g., estimation of human body motion), computer graphics (e.g., modeling of visco-elastic materials and fluid phenomena), and medical imaging (e.g., visualization and analysis of heart motion), that were not possible with purely geometric and kinematic techniques. PBM methods utilize geometry, kinematics, dynamics and material properties in order to model physical objects and their interactions with the physical world. Therefore, as opposed to purely geometric models, physics-based models incorporate additional constraints (e.g., material properties) that are very useful in both modeling and estimation applications.

A unique feature of PBM is that it provides a unified methodology for the modeling and estimation of rigid, articulated, and deformable models, and their motions. When a PBM formulation is used, it is often the case that the major difference between a modeling and an estimation application, is in the computation of the forces that are exerted on the deformable model(s) used.

In this book we will demonstrate how physics-based deformable models offer the power and generality needed in many computer vision, computer graphics and medical imaging applications. We will demonstrate through a series of examples and illustrations, how PBM methods can be applied to address difficult, real-world problems.

We will present a systematic physics-based framework for modeling rigid, articulated, and deformable objects, their interactions with the physical world, and the estimation of their shape and motion from visual data. We present a large variety of methods and associated experiments in computer vision, graphics and medical imaging that help the reader better understand the presented material. In addition special emphasis has been given on the development of techniques with interactive or close to real-time performance.

The first eight chapters present the beginnings of the PBM theory pertaining to the mathematical formulation of deformable models, and their use in estimation and modeling problems. This theory stems from the 1992 Ph.D thesis of the author in the Department of Computer Science at the University of Toronto, Ontario, Canada, under the supervision of Professor Demetri Terzopoulos. It was the first thesis in PBM to address problems in both computer vision and computer graphics. The rest of the chapters present significant advances to the initial theory, since the author joined as an Assistant Professor the Department of Computer and Information Science of the University of Pennsylvania, in

September of 1992. They demonstrate results from the Ph.D. work of Michael Chan, Douglas DeCarlo, Nick Foster, Ioannis Kakadiaris, Jonathan Kaye, Eunyoung Koh, Evangelos Kokkevis, and Jinah Park, who have been supervised by the author since 1992. The methodology presented on the integration of qualitative and quantitative techniques was done in joint work with Sven Dickinson, Assistant Professor, Department of Computer Science, Rutgers University. The author has collaborated on the development of the heart analysis method with Leon Axel, M.D., Ph.D., Professor in the Department of Radiology, University of Pennsylvania, and inventor of the MRI-SPAMM¹ technique, and Dr. Alistair Young, Lecturer in Biomedical Imaging, Department of Anatomy and Radiology, School of Medicine, University of Auckland, New Zealand. The author would especially like to thank his colleagues Norman I. Badler and Ruzena Bajcsy for stimulating discussions, support and encouragement, as well as Ahmed A. Shabana, Professor in the Department of Mechanical Engineering, University of Illinois at Chicago, for discussions on dynamics.

Since joining the University of Pennsylvania, the author has conducted most of his research in PBM at the Center for Human Modeling and Simulation (HMS) which is directed by Professor Norman I. Badler. HMS is one of the leading centers in the development of computer modeling and animation techniques with applications to computer graphics, computer vision and medical imaging. The author's work on human part segmentation and motion estimation has been conducted at the General Robotics and Active Sensory Perception (GRASP) Laboratory which is directed by Professor Ruzena Bajcsy. GRASP is a leading research center in robotics, computer and human vision.

The research presented in this book has been funded by the American Heart Association (AHA), the Army Research Office (ARO), the Defense Army Research Projects Agency (DARPA), the National Science Foundation (NSF), and the National Library of Medicine (NLM). Research papers, animations, and results related to the research presented in this book can be found at the author's World Wide Web page: <http://www.cis.upenn.edu/~dnm>.

0.1 SUMMARY OF CONTENTS

The various chapters present systematically the theory of the PBM framework that the author has been developing over the past eight years and its applications to computer vision, computer graphics and medical imaging. In particular this book presents:

1. a systematic approach for geometrically defining deformable primitives whose descriptive power varies depending on the application. The geometric definition is based on the use of parameterized primitives, global

¹SPAMM stands for SPAtial Modulation of Magnetization.

- parameterized deformations (e.g., tapering, bending) and local deformations based on the use of finite elements,
2. a systematic approach based on Lagrangian dynamics and the finite element method to convert the geometric parameters of the primitives to dynamic degrees of freedom and to add visco-elastic material properties,
 3. the use of Lagrange multiplier-based methods for the development of efficient physics-based holonomic constraints (e.g., point-to-point constraints) between deformable primitives that may be used to synthesize and recover complex articulated models,
 4. a recursive technique for estimating shape and nonrigid motion from noise corrupted data based on applying Kalman filtering theory to the dynamic models,
 5. the development of adaptive finite element techniques that together with global deformations can be used for the multi-level shape estimation and representation,
 6. the development of topologically adaptive deformable models based on the theory of blending that allow complex shape estimation and abstraction,
 7. the integration of qualitative aspect-based segmentation and quantitative PBM techniques for segmentation, shape and motion estimation of scenes with multiple occluded static and moving objects,
 8. the development of motion-based simultaneous part segmentation, shape and motion estimation techniques that allow the recovery of complex articulated shapes and their motions from single and multiple cameras. Particular emphasis is given to the human shape and motion estimation,
 9. the development of a new class of deformable models whose global parameters are functions instead of constants, that are suitable for biomedical applications and in particular the motion analysis of the heart's left-ventricle,
 10. a framework for the integration of anatomical and physiological techniques for modeling and visualizing the mechanics of the respiratory system and its pathologies,
 11. the development of very efficient recursive dynamics methods and adaptive control theory for modeling complex articulated objects and their interactions with the physical world,
 12. computationally efficient techniques based on the use of the Navier-Stokes equations to model liquid phenomena,
 13. a large number of examples and experiments that demonstrate the power and effectiveness of the presented methodology.

Finally, in chapters where a new concept or technique is introduced, we include a brief description of related work by other researchers to clarify the differences between our method and their research.

PHYSICS-BASED DEFORMABLE MODELS

Applications to Computer Vision,
Graphics and Medical Imaging

INTRODUCTION

This book presents a Physics-Based Modeling (PBM) framework for 3D shape and motion modeling, animation and estimation, with applications to computer vision, computer graphics and medical imaging. The framework addresses a variety of difficult modeling problems common to the above fields, through the development of deformable models and techniques for animating and estimating their shape, motion and interactions.

Despite the large body of research on object modeling for shape and motion estimation in computer vision, most existing techniques are limited to rigid objects with simple shapes. The shapes of many natural objects, however, cannot be represented accurately¹ in terms of simple shape primitives. They also undergo motions that are nonrigid and may be subject to various constraints. Animal bodies, for instance, produce surprisingly complex motions, not only as a consequence of their articulated skeletons, but also because of soft tissue deformations due to muscle action and gravitational effects. In such cases, identifying an animal's constituent parts and their shape is definitely a very challenging problem.

The computer graphics literature, on the other hand, is replete with mathematical representations of solid objects. In particular, the field of solid modeling has developed geometric methods for representing object shape, but geometric techniques are often inconvenient for modeling object motion. The insufficiency of pure geometry becomes particularly evident when one faces the problem of realistically animating deformable objects and their complex physical interactions.

Finally, in medical imaging applications it is often desirable to animate and estimate the complex shapes and motions of internal organs such as the lungs and the heart. In such applications, deformable models provide the appropriate mechanism for modeling tissue properties, estimating and visualizing the

¹The required degree of accuracy for object shape representation in computer vision depends on the application. For example, accurate shape representation may be essential in order to grasp an object by a vision-guided robot arm; the same level of accuracy is not necessarily required for object recognition or identification.

nonlinear motions of these organs, and for extracting model parameters which can be useful for illness diagnosis.

Initially, we present a class of deformable object models whose behaviors are determined not only by their geometric structures, but also by Lagrangian mechanics principles involving mass and damping distributions, and internal strain energies. The geometric structure of the models supports both global deformation parameters which efficiently represent the gross shape features of the salient parts of natural objects and local deformation parameters which capture shape details. In the context of graphics modeling, these models represent the physics-based unification of the parameterized and free-form paradigms. An important benefit of this global/local descriptive power in the context of computer vision is that it can potentially satisfy the often conflicting requirements of shape reconstruction and shape recognition.

The partial differential equations of motion that govern the dynamics of the modeling primitives make them responsive to externally applied forces. External forces may be derived from input data, may arise from interactions and other force fields in simulated physical environments, or may be applied interactively by the user. We incorporate physical constraints among primitives in order to compose articulated models with globally and locally deformable parts and provide force-based techniques for fitting such models to sparse, noise-corrupted 2D and 3D visual data. The motion equations are discretized using finite element methods and integrated over time using standard numerical analysis techniques.

Our physics-based modeling framework leads to shape and nonrigid motion estimators that apply dynamically deformable models to time-varying observations in order to track nonrigidly moving 3D objects. We employ the equations of motion of the models to formulate a shape and nonrigid motion estimator. The estimator is a continuous extended Kalman filter that recursively transforms the discrepancy between the sensory data and the estimated model state into generalized forces. These forces adjust the translational, rotational, and deformational degrees of freedom of the model, such that it evolves in a consistent fashion with the given noisy data.

In subsequent chapters we generalize the above theory to develop: 1) deformable models whose grid resolution is automatically adapted depending on the error of fit, 2) models which can automatically change topology and provide shape abstraction based on the theory of shape blending, 3) models whose global parameters are functions instead of constants, for improved shape estimation, 4) formulations that allow the application of the methodology to data originating from images, under the assumption of perspective projection, 5) methods for the motion-based simultaneous segmentation, shape and motion estimation of deformable and articulated objects, 6) methods for tracking complex objects in three-dimensions by integrating multiple views, 7) recursive dynamic formulations for efficient animations involving articulated objects, and 8) methods for modeling liquid phenomena.

In addition, we demonstrate how to further augment the capabilities of our PBM framework by coupling it with other theories and methodologies. In computer vision, we cope with the difficult problem of segmentation and shape estimation in both static and dynamic scenes, by presenting a methodology that is based on the use of an aspect-based qualitative shape segmentation technique to provide strong fitting constraints to our quantitative physics-based estimation approach. The combination of quantitative PBM and qualitative techniques allows the automatic shape estimation (from monocular and stereo images) of complex shapes whose parts belong to a certain vocabulary of objects, including objects with internal surface discontinuities. In computer graphics, we couple very efficient recursive dynamics formulations with adaptive control theory techniques to develop methods for the animation of articulated figures and their interactions with the physical world. In addition, we develop methods for liquid simulations based on the Navier-Stokes equations, and we couple them with the above PBM framework to create interactions between liquids and rigid objects. Finally, in medical imaging we present a framework for the integration of physiology and anatomy that has allowed the development of an interactive simulator for modeling the mechanics of the pulmonary system.

Depending on the application requirements, we may employ two types of simplifications to our framework. The first is at the model level where we idealize model behavior by appropriately ignoring certain higher-order physical effects, such as Coriolis interactions among the degrees of freedom. The second is at the level of the numerical techniques used to solve the discretized Lagrange equations of motion. These simplifications allow the real time or interactive time simulation of our models including continuous display on commonly available graphics workstations.²

In computer vision and medical imaging we address problems in shape reconstruction, quantitative model extraction from biomedical data for analysis and visualization, shape estimation and abstraction, shape segmentation and motion tracking. The graphics applications exploit the generative power of our framework to synthesize constrained shapes, nonrigid object motions including fluids, and object interactions for the purposes of computer animation. Furthermore, we have included a large number of experiments, figures and diagrams that will help the reader better understand the material presented.

²A simulation is real time if it can keep up with real world events, particularly the data input rates from real sensors. Interactive time refers to a simulation rate which is fast enough so that the user can visualize and interact with it without frustration.

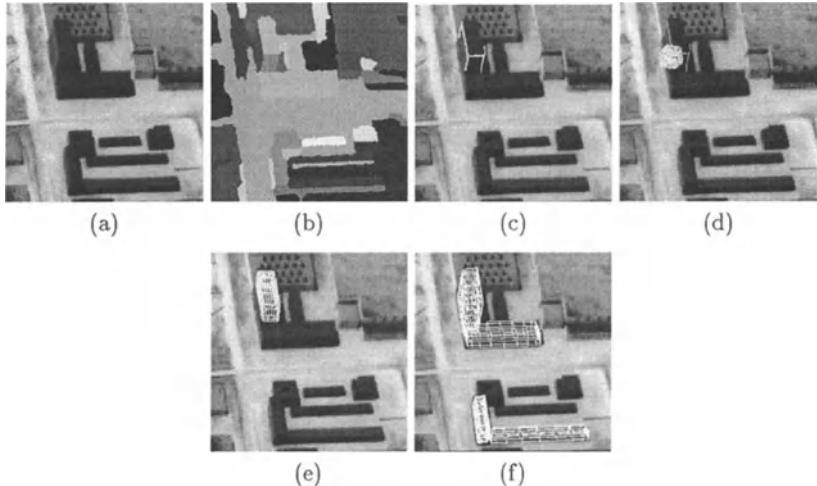


Figure 1.1 (a) Original image, (b) region extraction, (c) example of extracted primitive, (d-e) intermediate steps in fitting a deformable model to the building, (f) models fitted to other buildings present in the image.

1.1 ILLUSTRATIVE EXAMPLES OF MODELING AND ESTIMATION

We will now illustrate the spectrum of modeling and estimation problems that fall within the scope of this book through a series of examples from computer vision, computer graphics and medical imaging applications.

1.1.1 Computer Vision

A fundamental problem in computer vision is the segmentation and shape estimation of multiple objects from their gray-level images. It is particularly difficult to reconstruct the shapes of 3D objects from a single monocular image because the problem is seriously underconstrained. Our method for the integration of qualitative aspect-based segmentation and PBM shape estimation presented in Chapter 11 allows the segmentation and object shape estimation in scenes with multiple objects and occlusion. The following two experiments illustrate the approach in case of single and stereo images.

In the first experiment, we demonstrate the model extraction and fitting operations with an image of a site model with buildings in a cluttered background, as shown in Fig. 1.1(a)³. Fig. 1.1(b) shows the regions extracted from the qualitative analysis of Fig. 1.1(a), while Fig. 1.1(c) shows the extracted bounding contours of the taller building in the scene. Due to the inadequate intensity

³Courtesy of the RADIUS project.

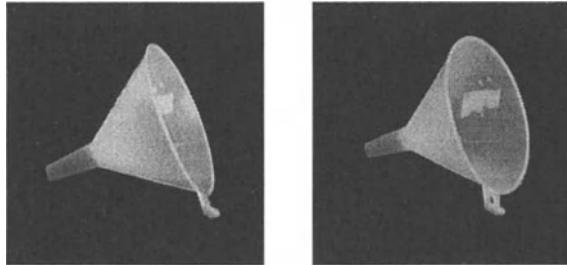


Figure 1.2 Initial pair of stereo images of a funnel.

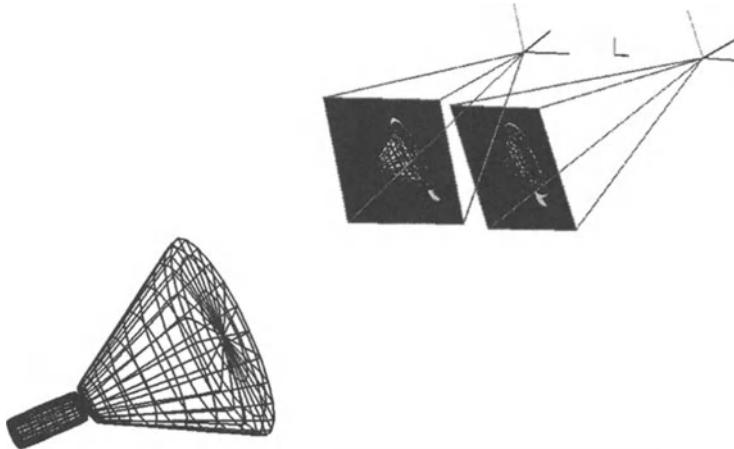


Figure 1.3 The 3D models fitted to the images of the funnel.

contrast in the images, the bounding contours of the building are not closed. However, three orthogonal sides have been recovered allowing the identification of its associated qualitative aspect. Subsequently, we can estimate the shape and pose of the building using the quantitative module. The fitting process and the associated deformation of the parallelepiped is shown in Figs. 1.1(d-e). Similarly, Fig. 1.1(f) shows the final fit to the building together with three more fitted buildings. The fitting algorithm works by first recovering the orientation of the building and then simultaneously refines its dimensions and the orientation.

In the second experiment, two stereo images of a funnel were taken from two viewpoints, where the relative displacements and orientations of the two cameras were known. Fig. 1.2 shows the original images of the funnel. The quali-

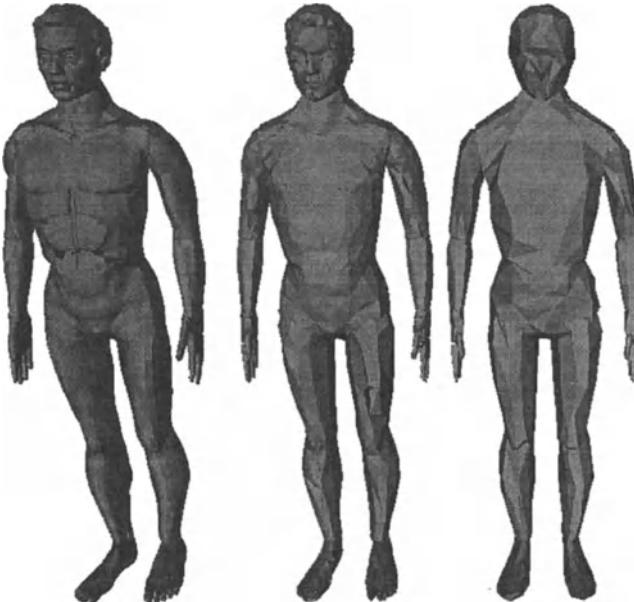


Figure 1.4 A front view of a human body displayed at three different levels of detail.

tative shape recovery process performed in both images decomposed the funnel into a tapered cylinder and a short cylinder. Two 3D deformable models are then fitted simultaneously to both images and Fig. 1.3 shows the fitted models in 3D.

Techniques for estimating and abstracting the shape of objects from dense or sparse 3D range data are very often used in computer vision, computer graphics and CAD applications. In Chapter 9 we present a method for the multi-level shape representation based on the use of global deformations and adaptive finite element methods. Fig. 1.4 shows a front view of a human body figure displayed at three different levels of detail. The human body figure consists of 15 parts: head, torso, lower torso, 3 parts for each arm, 3 parts for each leg. The numbers of polygons used at each level of representation were 18155, 7292, and 2260, respectively, and the numbers of nodes were 18005, 3696, and 1180, respectively.

In Chapter 10, we have developed a method for the hierarchical object shape estimation from range data based on blending. Based on this method, the initial model is a sphere that automatically blends with other deformable primitives, based on the error of fit. No apriori knowledge of the object being estimated is assumed. Using this method, the fitting of a mug is shown in Fig. 1.5. The initial fitting of the deformable sphere to the incomplete and sparse data shown

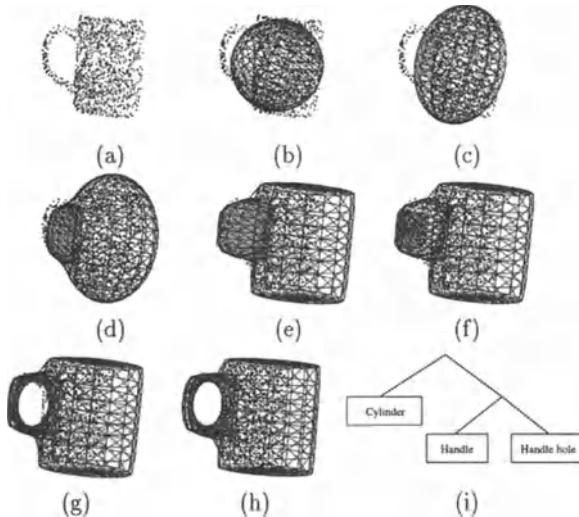


Figure 1.5 Fitting of a mug.

in (a) is shown in (b) and (c). In (d), (e) and (f) we show the fitting of the handle. After further fitting of the handle in (f), a hole blend is added. After rough fitting and hole opening (g), the final fit is obtained (h). Finally, in (i) we show the mug shape abstraction (with manually generated names).

With the advent of low cost, real time visual sensors and image processing hardware, dynamic object representation, estimation and tracking tasks are gaining a significant presence in the vision literature. An interesting problem is to infer the shapes and motions of complex single- or multi-part deformable/articulated objects from noisy 2D image sequences. Such estimation problems are particularly challenging when they must be solved on-line, and Kalman filtering has become a popular approach for accomplishing this. In Chapter 12 we have developed a motion-based method for the segmentation, shape and motion estimation of articulated objects from single and multiple images.

In Fig. 1.6 we have acquired images of a subject moving his arms in 3D. To simplify the acquisition of the occluding contours, we designed the background to be a simple one. The process of tracking the motion of the subject's arms consists of two phases. The first phase is the model building phase. The subject was asked to perform a set of motions to acquire the anthropometric dimensions of its (left and right) forearm and of its (left and right) upper arm. The subject placed himself in the viewing volume of the cameras and assumed a position where the body is erect and the arms are placed straight against the sides of the body facing medially. After standing still for a few seconds in order to acquire the initial apparent contour, the subject lifted his arms to the front until the arms reached the horizontal position. Continuing from this position,

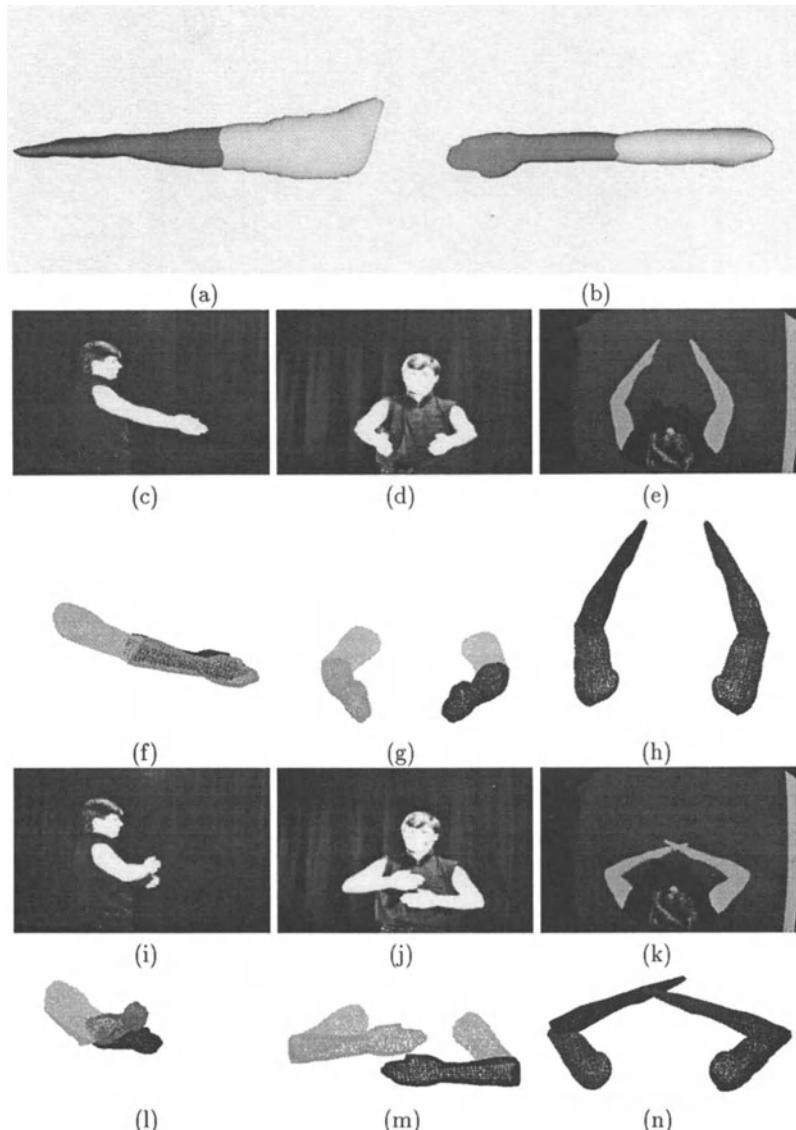


Figure 1.6 (a) Side view and (b) top view of the estimated three-dimensional models of the parts of subject's arm, (c-e,i-k) sample input images from the three cameras for frames 1 and 16 in the motion sequence, and (f-h,l-n) tracking results for frames 1 and 16 in the motion sequence.



Figure 1.7 The estimated motion parameters for the motion of the subject's arms have been applied to a graphics model of the subject (three frames from four different views).

the subject rotated his hands so that the palms are facing upwards and bent the elbows, bringing the forearms to the vertical position. By employing our part segmentation algorithm, we extracted the three-dimensional shape of the forearms and of the upper arms of the subject. Notice that since the subject did not flex his wrists, the estimated models (Fig. 1.6(a,b)) for the forearms, includes the wrists also. In the second phase, we asked the subject to move his arms in 3D. Figs. 1.6(c-e,i-k) show sample input images from the three cameras. Figs. 1.6(f-h,l-n) show the estimated 3D pose of the subject's parts of the arm. Fig. 1.7 shows four views for three frames of an animation. The animation was created by applying the estimated motion parameters to a graphics model of the subject.

1.1.2 Medical Imaging

We have recently developed a new class of deformable models (see Chapter 13) whose global parameters are functions instead of constants. Using these new volumetric models we have been able to analyze the three-dimensional shape and nonlinear motion of the LV based on data acquired from a new magnetic resonance imaging (MRI) technique based on *magnetic tagging* (MRI-SPAMM) which has been developed at the University of Pennsylvania for imaging of the regional heart wall motion (Axel and Dougherty [9]). This fast, non-invasive technique promises to be very useful in the analysis of heart wall motion because it provides temporal correspondence of material points within the heart wall.

Fig. 1.8 shows model fitting results for a normal Left Ventricle (LV) over 5 time frames (only 4 are shown) from end-diastole (ED) to end-systole (ES). The top row shows a view from the base of the LV of the fitted model. The twisting of the inner wall (shown in white) is obvious. The middle row shows a side view of the model, while the last row is similar to the first row and shows a view of the model from the apex. We can easily observe the longitudinal contraction as well as the radial contraction. The combination of computed forces from the SPAMM datapoints from two orthogonal planes allows us to recover the deformation of the model in 3D.

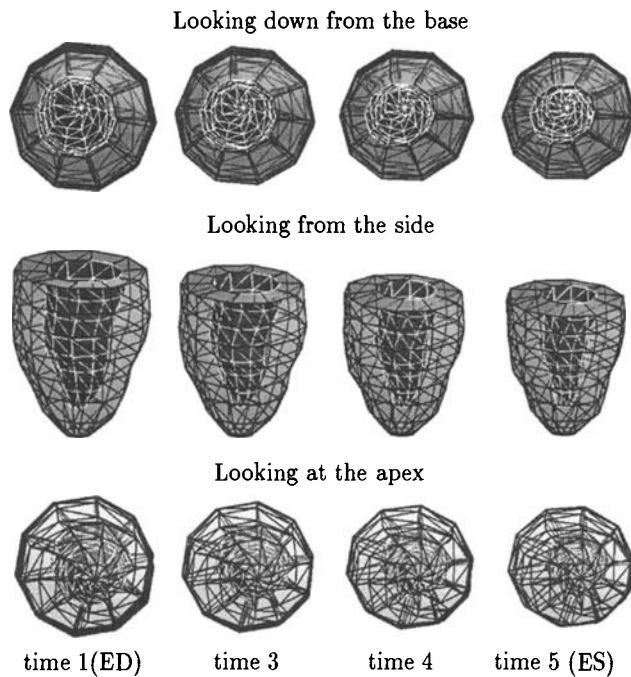


Figure 1.8 Fitted models during systole.

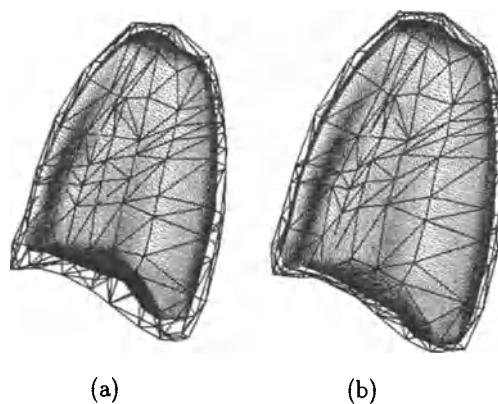


Figure 1.9 3D models showing inhalation, (a) is starting shape, (b) is during inhalation.

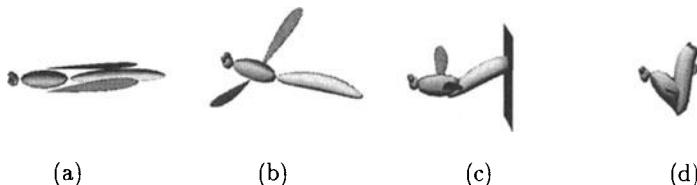


Figure 1.10 Dragonfly. Self-assembly (a). Flight (b). Swatting (c). Swatted fly (d).

In Chapter 14 we present a new direction in medical applications which stems from our recent efforts to couple PBM methods and physiology modeling to visualize respiratory mechanics. This effort is hoped to result in future sophisticated virtual environments that can be used in both clinical applications and in medical training. Based on this new approach we have been able to model both quiet normal breathing as well as various types of pneumothoraces. In Fig. 1.9 we show the lung and the chest wall during inhalation. Notice the increased chest wall size in (b) and the corresponding increase in lung size.

1.1.3 Computer Graphics

The deformable modeling primitives and the constraint methods that we present in this book, are useful for a variety of computer graphics applications. Because they are dynamic, our models are particularly well suited to physics-based animation tasks. Such an application is illustrated in Fig. 1.10 which shows the automatic construction of a minimalist dragonfly from its constituent deformable model parts. Fig. 1.10(a) shows the disjoint parts in their initial configurations. After activating the stabilized Lagrange multiplier-based constraint algorithm, the model self-assembles to form the articulated dragonfly. Four point-to-point constraints hold the deformable body parts together. The dragonfly “works” inasmuch as forces can induce opening and flapping of the wings, as is illustrated in Fig. 1.10(b). In Fig. 1.10(c) we swat the dragonfly in the rear with an impenetrable plane. The body parts deform in response to the blow, but the point-to-point constraints continue to hold them together. The mangled dragonfly is shown in Fig. 1.10(d).

Fig. 1.11 illustrates the self-assembly and subsequent animation of a snowman. Fig. 1.11(a) shows a view of the disjoint deformable body parts of a snowman. The snowman self-assembles when the constraints are activated (Fig. 1.11(b)). There are 12 point-to-point constraints holding the snowball parts together. Gravity is activated and the snowman drops to the impenetrable floor and locomotes along a prespecified path by repeatedly bouncing in a controlled fashion (Fig. 1.11(c-d)).

Although physics-based simulation is guaranteed to generate natural looking motion, it provides the user with very little control over the results produced.

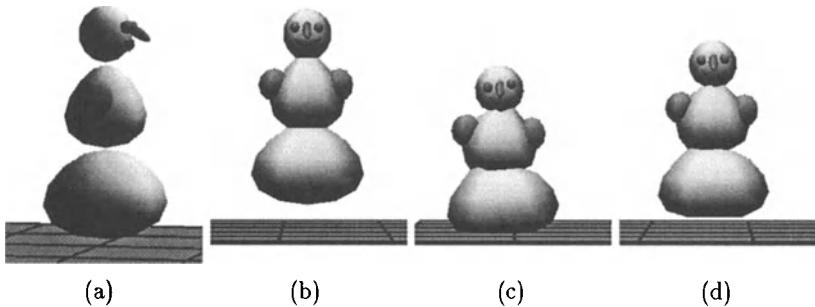


Figure 1.11 Yellow snowman. Disjoint parts (a). Self-assembly (b). Hopping (c-d).

In kinematically controlled animation, the user is responsible for every aspect of the generated motion; however, no guarantees can be made about the result's physical correctness. In the method we present in Chapter 15 we provide the direct control of kinematic animation to dynamically generated motion, hence taking advantage of the strengths of both techniques. In this approach kinematically predefined or interactively defined joint and limb trajectories of articulated objects are used to dynamically control a simulation. The dynamic control is based on the integration of a recursive dynamics formulation (for improved performance compared to a Lagrange multiplier method) with adaptive control theory.

Fig. 1.12 demonstrates an attempt to follow a physically incorrect trajectory which involves penetration between the arms and a table. Although kinematically the motion is valid, when played back dynamically the contacts are detected and penetration is prevented. A motion trajectory is originally provided for the shoulders and the elbow. Once the contact forces prevent any further motion, the output of the joint controllers reaches its maximum value and stays there, stubbornly trying to make the arms follow the desired motion.

So far all the PBM examples presented have dealt with modeling phenomena involving rigid and viscoelastic single/multiple objects. In Chapter 16 we present a PBM method for modeling a different class of physical phenomena and in particular liquids. In addition, we also demonstrate how to model simple interactions between buoyant rigid objects and a liquid.

The frames in Fig. 1.13 show screen shots from an animation involving buoyant objects. Water flows into a closed container carrying soda cans along with it. When the flow is turned off, the cans gather at the far corner of the container because the walls in this example were set as non-slip so the tangential fluid velocity is zero. This simulates the effect that objects tend to gather in stagnant parts of a flow. The water motion was precomputed in thirty minutes over a $30 \times 10 \times 20$ grid. The soda cans were added later using an interactive editor which takes a precomputed velocity and pressure field, and calculates the forces on an

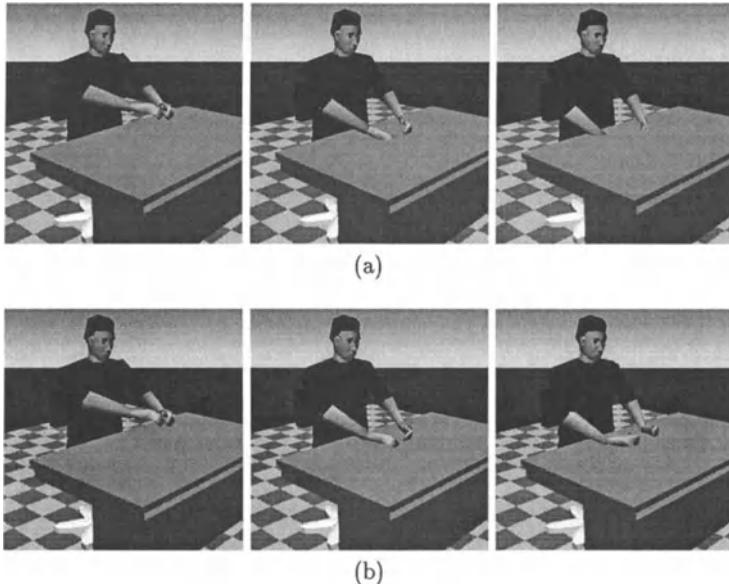


Figure 1.12 Following a physically incorrect motion trajectory: (a) Pure kinematics allows the motion, whereas (b) Dynamic simulation prevents inter-penetration between the arms and the desk.

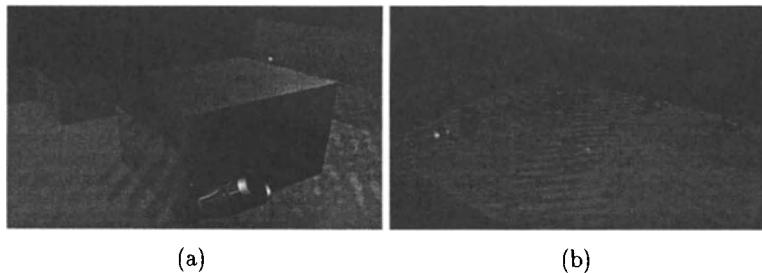


Figure 1.13 Dynamic objects. Soda cans are carried along with the incoming water, colliding with obstacles (a), and getting caught in local eddies (b).

object within the mesh. In this way, many different shapes and sizes of object can be experimented with, without having to re-do the fluid calculation.

1.2 CHAPTER OUTLINE

The book is organized as follows. The first eight chapters introduce the basic PBM formulation that allows the mathematical formulation of deformable models. The rest of the chapters present significant advancements to the ini-

tial theory that allow the successful solution of many difficult and important problems in computer vision, computer graphics and medical imaging. In particular chapters nine to twelve present more advanced computer vision methods, chapters thirteen and fourteen present medical imaging methods, and chapters fifteen and sixteen present advanced computer graphics methods. Finally, in chapter seventeen we give our conclusions and our future research goals.

Chapter 1 gives the introduction to the approach and examples from the application of deformable models to various domains.

Chapter 2 presents the geometric formulation of deformable models. It describes a general technique for defining global and local geometric degrees of freedom and applies it to create a class of deformable models that we dub *hybrid models*.

Chapter 3 develops the kinematic and dynamic formulation of the models. We describe a systematic method for converting the geometric degrees of freedom of a deformable model into generalized coordinates, or dynamic degrees of freedom, by using the Lagrange equations of motion. Furthermore, we present a general technique for deriving the stiffness matrix associated with the elastic properties of the model from a given deformation energy expression.

Chapter 4 presents the finite element method that we employ to discretize the Lagrange equations of motion. In particular, we give various finite element tessellations of the parametric space of a deformable model, examples of suitable finite elements, the discretization of the Lagrange equations of motion through finite elements, and finally the criteria for choosing the appropriate elements for specific vision, graphics and medical imaging applications.

Chapter 5 presents techniques for converting visual data into forces that can be applied to deformable models in data fitting scenarios. In case of computer animation purposes, we describe two algorithms which use the elastic properties of our models to calculate forces stemming from collisions and friction against impenetrable solid or deformable surfaces.

Chapter 6 first describes the integration schemes we use to approximate the solution of the differential equations of motion and the algorithm for determining the initial condition of our models. Second we present computer vision and computer graphics experiments which test the modeling methods developed to this point.

Chapter 7 presents a technique to model holonomic constraints within a deformable model formulation. We demonstrate the method through a series of experiments involving point-to-point constraints between deformable part models which should not be violated, regardless of the magnitude of the forces exerted on the parts. The chapter considers a stabilized Lagrange multiplier method and describes the various integration schemes we use to simulate the constrained differential equations of motion.

Chapter 8 applies continuous nonlinear Kalman filtering theory to construct a recursive estimator which employs the Lagrange equations of 3D nonrigid motion that we have presented in the previous chapters as a system model. This estimator allows the recovery of shape and nonrigid motion in the presence of noise. We also describe a computationally efficient implementation of the Kalman filter equations. Finally, we present computer vision experiments involving shape and nonrigid motion estimation from 3D data.

Chapter 9 presents a method for improving shape estimation and for multi-level shape representation based on the use of global deformations and adaptive finite elements.

Chapter 10 presents a generalization of the previously presented class of deformable primitives based on the theory of shape blending. Through this method, we can blend parameterized dynamic primitives to estimate and abstract complex shapes, based on the error of fit. The method always uses a sphere as an initial shape. Based on this theory, the sphere can automatically change topology which is necessary to recover shapes of higher genus (e.g., a torus or other objects with holes).

Chapter 11 presents an approach to both static and dynamic segmentation, shape and motion estimation that is based on the integration of qualitative aspect-based shape segmentation and quantitative PBM shape and motion estimation.

Chapter 12 presents a method for the motion-based simultaneous segmentation, shape and motion estimation of articulated objects from image sequences, without *a priori* knowledge of the object being tracked. We demonstrate the application of this method to various domains, including that of human motion shape and motion estimation from multiple cameras.

Chapter 13 presents a new class of deformable models that are useful for the motion analysis of the heart's left ventricle. These new models have global parameters that are functions, instead of constants, and can capture the variability in the shape and motion of the heart during systole. We present motion analysis results for the heart's left ventricle. The data were obtained from several patients using tagged MRI data.

Chapter 14 presents a new direction in medical applications which stems from our recent efforts to couple PBM methods and physiology modeling to visualize respiratory mechanics. This effort is hoped to result in future sophisticated virtual environments that can be used in both clinical applications and in medical training.

Chapter 15 presents an animation system which integrates a recursive dynamics formulation and a robust adaptive control theory scheme to allow for fully automated dynamic simulations of articulated objects based on user specified kinematic motion requirements. The recursive dynamics formulation is used

for improved efficiency in case of articulated objects, as opposed to the use Lagrange multipliers presented in Chapter 7.

Chapter 16 presents a PBM method for modeling a different class of phenomena and in particular liquids. As opposed to using the Lagrange equations of motion that were useful for modeling visco-elastic objects, we employ the Navier-Stokes equations, that allow the modeling of liquids. In addition, we also demonstrate how to model simple interactions between buoyant rigid objects and a liquid.

Chapter 17 draws conclusions from our work and gives a perspective on future research directions.

GEOMETRY OF DEFORMABLE MODELS

This chapter presents a technique for creating classes of *hybrid deformable models* whose underlying geometric structure allows the combination of parametric models (e.g., spheres, cylinders, superquadrics), parameterized global deformations (e.g., tapers, bends, shears, twists) and local spline free-form deformations (e.g., membranes, thin-plates). The local degrees of freedom will allow the representation of fine scale structure and the modeling of irregular real world objects, while the global deformations capture salient features of shape that are innate to natural parts in a computationally efficient way. The technique is applicable to any parametric model, parameterized global deformation and local deformation, as long as the resulting equations are differentiable with respect to the underlying parameters. We formulate 3D solids and illustrate the approach for the case of 3D surface models.

2.1 RELATED WORK

After more than a decade of research, the notion of early visual reconstruction as a data fitting problem using generalized spline models is now in a highly evolved state of development, most evidently so in the context of the surface reconstruction problem [35, 267, 274]. Generalized spline techniques underlie the notion of regularization and its application to a variety of reconstruction problems in early vision [225, 272]. The many degrees of freedom and local deformation properties of generalized splines allow them to conform to low-level visual data with ease.

On another front, much effort has gone into the search for suitable models for the purposes of object recognition. Biederman [30] reports the results of psychophysical experiments suggesting that the recovery of arrangements of two or three major primitive components or parts results in fast recognition of objects, even when the objects are observed from different viewpoints, are occluded, or are unfamiliar. Parameterized part models capture the structure of the world by describing meaningful chunks of data in terms of a few parameters. Such models are beneficial for object representation, since dealing with a manage-

able number of parameters simplifies the problem of indexing into a database of stored models and verifying match hypotheses.

Throughout the 70's, the research of Binford and his coworkers on generalized cylinders focussed on the problem of recovering parameterized models of objects and led to vision systems such as ACRONYM which use reasoning to recover parameterized parts [40]. Marr and Nishihara [171] were among the first to propose a hierarchical representation of objects in terms of parts. Their work uses generalized cylinders to describe each part, thereby limiting the scope of the representation to objects adequately describable as collections of generalized cylinders.

Motivated by the generalized cylinder idea and the need to go beyond geometry to exploit computational physics in the modeling process, Terzopoulos, Witkin and Kass [275] propose a deformable cylinder constructed from generalized splines. They develop force field techniques for fitting their model to monocular, binocular, and dynamic image data. The distributed nature of this deformable model enhances its descriptive power and allows the representation of natural objects with asymmetries and fine detail. However, the generalized spline components of the model do not explicitly provide an abstract representation of object shape in terms of a few parameters.

The generalized cylinder representation requires the specification of an axis, generally a space curve, and the cross-section function. Pentland [214] proposes the use of a simpler part model with scalar parameters, the superquadric ellipsoid with parameterized deformations [23] (the notion of a superquadric was introduced by Hein [94]). Pentland's proposal has spawned a flurry of efforts to reconstruct superquadric ellipsoids with global geometric deformations from 3D data, and these have met with some success [105, 107, 263].

Inspired by the PBM approach, Pentland [215, 217] further proposes an alternative method for fitting deformable part models. Based on modal analysis, a technique for analyzing the vibrations of linear mechanical systems under periodic forcing conditions described by Bathe and Wilson [27], he applies to superquadrics a polynomial approximation to the deformation "modes" of a 21 node element. The deformation modes are efficient and useful for the recovery of smooth, symmetrically deformed parts. On the down side, global deformation modes lack an obvious physical meaning and it has not been demonstrated so far how to select the right number of them given an application. Finally, in case of complex shapes many modes are required, rendering this approach no more efficient than a nodal finite element solution [27].

In the computer graphics literature, mathematical representations of solid objects has been the commonplace. The field of solid modeling [118] has developed geometric methods for representing object shape, but these techniques are often inconvenient for modeling object motion. The insufficiency of purely geometric techniques (e.g., Sederberg and Parry [252] define local deformations of solid primitives as ambient space warps) becomes particularly evident when

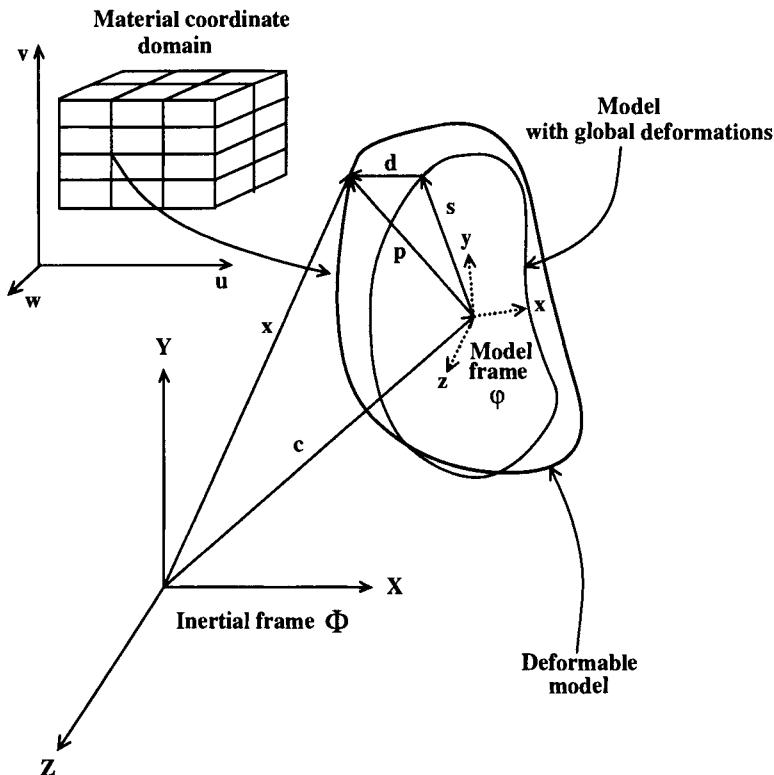


Figure 2.1 Geometry of deformable model.

one faces the problem of realistically animating deformable objects. Physically-based models have been pursued for this purpose (e.g., [273, 111, 224, 274, 192, 45, 104, 215, 138, 44]). These methods introduce realistic physical behaviors into free-form geometric models of solids or their surfaces.

In this chapter we present the basic formulation of a hybrid class of deformable models which can be used in both shape abstraction and shape estimation applications. The coupling of rigid-body and deformation dynamics is similar to that described in [275], but the formulation accommodates global deformations defined by fully nonlinear parametric equations. Hence, the models are more general than the restrictive, linearly deformable ones in [293, 19] and quadratically deformable ones in [215, 249]. Since their initial introduction in [278, 176, 182], significantly more powerful formulations have been developed by the author and they will be presented in subsequent chapters.

2.2 HYBRID DEFORMABLE MODELS

Geometrically, the models we develop are 3D solids in space whose intrinsic (material) coordinates are $\mathbf{u} = (u, v, w)$, defined on a domain Ω^1 .

The positions of points on the model relative to an inertial frame of reference Φ in space are given by a vector-valued, time varying function of \mathbf{u} :

$$\mathbf{x}(\mathbf{u}, t) = (x_1(\mathbf{u}, t), x_2(\mathbf{u}, t), x_3(\mathbf{u}, t))^T, \quad (2.1)$$

where T is the transpose operator. We set up a noninertial, model-centered reference frame ϕ and express these positions as

$$\mathbf{x} = \mathbf{c} + \mathbf{R}\mathbf{p}, \quad (2.2)$$

where $\mathbf{c}(t)$ is the origin of ϕ at the center of the model and the orientation of ϕ is given by the rotation matrix $\mathbf{R}(t)$. Thus, $\mathbf{p}(\mathbf{u}, t)$ denotes the positions of points on the model relative to the model frame. To incorporate global and local deformations, we further express \mathbf{p} as the sum of a reference shape $\mathbf{s}(\mathbf{u}, t)$ and a displacement function $\mathbf{d}(\mathbf{u}, t)$:

$$\mathbf{p} = \mathbf{s} + \mathbf{d}. \quad (2.3)$$

In the next two subsections we first formulate the reference shape \mathbf{s} to account for global deformations consisting of parameterized primitives (e.g., superquadrics) and parameterized global deformations (e.g., tapers, bends). We then describe the displacement \mathbf{d} which defines the local deformations of our models.

2.2.1 Global Deformations

We define the reference shape as

$$\mathbf{s} = \mathbf{T}(\mathbf{e}(\mathbf{u}; a_0, a_1, \dots); b_0, b_1, \dots) = \mathbf{T}(\mathbf{e}; \mathbf{b}). \quad (2.4)$$

Here, a geometric primitive \mathbf{e} , defined parametrically in \mathbf{u} and parameterized by the variables a_i , is subjected to the *global deformation* \mathbf{T} which depends on the parameters b_i . Although generally nonlinear, \mathbf{e} and \mathbf{T} are assumed to be differentiable (so that we may compute the Jacobian of \mathbf{s}) and \mathbf{T} may be a composite sequence of primitive deformation functions. We define the vector of global deformation parameters

$$\mathbf{q}_s = (a_0, a_1, \dots, b_0, b_1, \dots)^T. \quad (2.5)$$

The above formulation is general and can be carried out for an arbitrary reference shape \mathbf{s} given as a differentiable parameterized function of \mathbf{u} with respect to \mathbf{q}_s .

¹For the case of a 3D “shell,” $\mathbf{u} = (u, v, 1)$

In the next subsections we consider three examples of global deformations that are useful in computer vision, graphics and medical applications. Furthermore, we calculate the Jacobian matrix

$$\mathbf{J} = \frac{\partial \mathbf{s}}{\partial \mathbf{q}_s}, \quad (2.6)$$

whose computation is essential for the kinematic and dynamic formulation to follow in subsequent chapters.

Example 1: Superquadric ellipsoid

We consider a parametric model consisting of a superquadric ellipsoid solid which describes a useful class of part models suitable for vision and graphics applications.

The parametric equation of a superquadric ellipsoid solid $\mathbf{e} = (e_1, e_2, e_3)$ is

$$\mathbf{e} = a_0 w \begin{pmatrix} a_1 C_u^{\epsilon_1} C_v^{\epsilon_2} \\ a_2 C_u^{\epsilon_1} S_v^{\epsilon_2} \\ a_3 S_u^{\epsilon_1} \end{pmatrix}, \quad (2.7)$$

where $-\pi/2 \leq u \leq \pi/2$, $-\pi \leq v < \pi$, $0 \leq w \leq 1$, and $S_u^\epsilon = \text{sgn}(\sin u)|\sin u|^\epsilon$, $C_u^\epsilon = \text{sgn}(\cos u)|\cos u|^\epsilon$, and similarly for C_v^ϵ and S_v^ϵ . Here, $a_0 \geq 0$ is a scale parameter, $0 \leq a_1, a_2, a_3 \leq 1$, are aspect ratio parameters, and $\epsilon_1, \epsilon_2 \geq 0$ are “squareness” parameters [23, 24].

We collect the parameters in \mathbf{s} into the parameter vector

$$\mathbf{q}_s = (a_0, a_1, a_2, a_3, \epsilon_1, \epsilon_2)^T. \quad (2.8)$$

The Jacobian matrix \mathbf{J} of the superquadric ellipsoid solid is a 3×6 matrix whose non-zero entries are

$$\begin{aligned} \mathbf{J}_{11} &= wa_1 C_u^{\epsilon_1} C_v^{\epsilon_2}, \\ \mathbf{J}_{12} &= a_0 w C_u^{\epsilon_1} C_v^{\epsilon_2}, \\ \mathbf{J}_{15} &= a_0 w a_1 \ln(|\cos u|) C_u^{\epsilon_1} C_v^{\epsilon_2}, \\ \mathbf{J}_{16} &= a_0 w a_1 \ln(|\cos v|) C_u^{\epsilon_1} C_v^{\epsilon_2}, \\ \mathbf{J}_{21} &= wa_2 C_u^{\epsilon_1} S_v^{\epsilon_2}, \\ \mathbf{J}_{23} &= a_0 w C_u^{\epsilon_1} S_v^{\epsilon_2}, \\ \mathbf{J}_{25} &= a_0 w a_2 \ln(|\cos u|) C_u^{\epsilon_1} S_v^{\epsilon_2}, \\ \mathbf{J}_{26} &= a_0 w a_2 \ln(|\sin v|) C_u^{\epsilon_1} S_v^{\epsilon_2}, \\ \mathbf{J}_{31} &= wa_3 S_u^{\epsilon_1}, \\ \mathbf{J}_{34} &= a_0 w S_u^{\epsilon_1}, \\ \mathbf{J}_{35} &= a_0 w a_3 \ln(|\sin u|) S_u^{\epsilon_1}. \end{aligned} \quad (2.9)$$

Example 2: Superellipsoid with tapering and bending

We now generalize the above model to allow parameterized tapering and bending deformations to increase the geometric coverage of the part representation. Combining ideas from [24] and [263], we define these deformations so that they are continuously differentiable and commutative.

We combine linear tapering along principal axes 1 and 2 and bending along principal axis 3² of the superquadric \mathbf{e} into a single parameterized deformation \mathbf{T} , and express the reference shape as

$$\mathbf{s} = \mathbf{T}(\mathbf{e}, t_1, t_2, b_1, b_2, b_3) = \begin{pmatrix} \left(\frac{t_1 e_3}{a_0 a_3 w} + 1 \right) e_1 + b_1 \cos\left(\frac{e_3 + b_2}{a_0 a_3 w} \pi b_3\right) \\ \left(\frac{t_2 e_3}{a_0 a_3 w} + 1 \right) e_2 \\ e_3 \end{pmatrix}, \quad (2.10)$$

where $-1 \leq t_1, t_2 \leq 1$ are the tapering parameters in principal axes 1 and 2, respectively, and where b_1 defines the magnitude of the bending and can be positive or negative, $-1 \leq b_2 \leq 1$ defines the location on axis 3 where bending is applied and $0 < b_3 \leq 1$ defines the region of influence of bending. Our method for incorporating global deformations is not restricted to only tapering and bending deformations. Any other deformation that can be expressed as a continuous parameterized function can be incorporated as our global deformation in a similar way.

We collect the parameters in \mathbf{s} into the parameter vector

$$\mathbf{q}_s = (a_0, a_1, a_2, a_3, \epsilon_1, \epsilon_2, t_1, t_2, b_1, b_2, b_3)^T. \quad (2.11)$$

Defining $r = \frac{e_3 + b_2}{a_0 w a_3} \pi b_3$, the Jacobian matrix \mathbf{J} is a 3×11 matrix whose non-zero entries are

$$\begin{aligned} \mathbf{J}_{11} &= (t_1 S_u^{\epsilon_1} + 1) w a_1 C_u^{\epsilon_1} C_v^{\epsilon_2} + \frac{b_1 b_2 b_3}{a_0^2 w a_3} \pi \sin(r), \\ \mathbf{J}_{21} &= (t_2 S_u^{\epsilon_1} + 1) w a_2 C_u^{\epsilon_1} S_v^{\epsilon_2}, \\ \mathbf{J}_{31} &= w a_3 S_u^{\epsilon_1}, \\ \mathbf{J}_{12} &= (t_1 S_u^{\epsilon_1} + 1) a_0 w C_u^{\epsilon_1} C_v^{\epsilon_2}, \\ \mathbf{J}_{23} &= (t_2 S_u^{\epsilon_1} + 1) a_0 w C_u^{\epsilon_1} S_v^{\epsilon_2}, \\ \mathbf{J}_{14} &= \frac{b_1 b_2 b_3}{a_0 w a_3^2} \pi \sin(r), \\ \mathbf{J}_{34} &= a_0 w S_u^{\epsilon_1}, \\ \mathbf{J}_{15} &= t_1 \ln(|\sin u|) S_u^{\epsilon_1} a_0 w a_1 C_u^{\epsilon_1} C_v^{\epsilon_2} + (t_1 S_u^{\epsilon_1} + 1) a_0 w a_1 \\ &\quad \ln(|\cos u|) C_u^{\epsilon_1} C_v^{\epsilon_2} - b_1 b_3 \pi \ln(|\sin u|) S_u^{\epsilon_1} \sin(r), \\ \mathbf{J}_{25} &= t_2 \ln(|\sin u|) S_u^{\epsilon_1} a_0 w a_2 C_u^{\epsilon_1} S_v^{\epsilon_2} \\ &\quad + (t_2 S_u^{\epsilon_1} + 1) a_0 w a_2 \ln(|\cos u|) C_u^{\epsilon_1} S_v^{\epsilon_2}, \\ \mathbf{J}_{35} &= a_0 w a_3 \ln(|\sin u|) S_u^{\epsilon_1}, \end{aligned}$$

²The principal axes 1, 2 and 3 correspond to the x , y and z axes of the model frame ϕ

$$\begin{aligned}
\mathbf{J}_{16} &= (t_1 S_u^{\epsilon_1} + 1) a_0 w a_1 \ln(|\cos v|) C_u^{\epsilon_1} C_v^{\epsilon_2}, \\
\mathbf{J}_{26} &= (t_2 S_u^{\epsilon_1} + 1) a_0 w a_2 \ln(|\sin v|) C_u^{\epsilon_1} S_v^{\epsilon_2}, \\
\mathbf{J}_{17} &= S_u^{\epsilon_1} a_0 w a_1 C_u^{\epsilon_1} C_v^{\epsilon_2}, \\
\mathbf{J}_{28} &= S_u^{\epsilon_1} a_0 w a_2 C_u^{\epsilon_1} S_v^{\epsilon_2}, \\
\mathbf{J}_{19} &= \cos(r), \\
\mathbf{J}_{110} &= -\frac{b_1 b_3}{a_0 w a_3} \pi \sin(r), \\
\mathbf{J}_{111} &= -b_1 \pi \sin(r) r,
\end{aligned} \tag{2.12}$$

where $S_\theta^\epsilon = \text{sgn}(\sin \theta) |\sin \theta|^\epsilon$ and $C_\theta^\epsilon = \text{sgn}(\cos \theta) |\cos \theta|^\epsilon$.

Example 3: Model with tapering, bending, shearing and twisting

We now generalize the model defined in the previous example by allowing global transformations that include tapering, bending, shearing, and twisting along each of the three principal axes of the deformable model.

Without loss of generality we define a superquadric ellipsoid \mathbf{e} as in the previous examples and we define a global vector transformation $\mathbf{T}(\mathbf{e}; \mathbf{b})$ that includes tapering, bending, shearing, and twisting along each of the three principal axes 1, 2 and 3 of the solid. We define this vector transformation \mathbf{T} as a composition of three simpler transformations \mathbf{T}_3 , \mathbf{T}_1 and \mathbf{T}_2 along axes 3, 1 and 2, respectively, i.e.,

$$\mathbf{T}(\mathbf{e}; \mathbf{b}) = \mathbf{T}_2(\mathbf{T}_1(\mathbf{T}_3(\mathbf{e}; \mathbf{b}))). \tag{2.13}$$

The reference shape is therefore defined as

$$\mathbf{s} = \mathbf{T}(\mathbf{e}; \mathbf{b}). \tag{2.14}$$

We give the definition of these transformations in clock-wise order starting from axis 3.

The transformation along axis 3 is given by

$$\mathbf{r}_3 = \mathbf{T}_3(\mathbf{e}; \mathbf{b}) = \begin{pmatrix} A_3 \cos(\phi_3) - B_3 \sin(\phi_3) \\ A_3 \sin(\phi_3) + B_3 \cos(\phi_3) \\ e_3 \end{pmatrix}, \tag{2.15}$$

where

$$A_3 = [(t_3^1 e_3 / a_0 a_3 w) + 1] e_1 + b_3^{a_3^1} \cos[(e_3 + b_3^{l_3^1}) / (a_0 a_3 w) \pi b_3^{r_3^1}] + e_3 s_3^1, \tag{2.16}$$

$$B_3 = [(t_3^2 e_3 / a_0 a_3 w) + 1] e_2 + b_3^{a_3^2} \cos[(e_3 + b_3^{l_3^2}) / (a_0 a_3 w) \pi b_3^{r_3^2}] + e_3 s_3^2 \tag{2.17}$$

and

$$\phi_3 = (e_3 / a_0 a_3 w) \pi \tau_3. \tag{2.18}$$

Here τ_3 is a twisting parameter along axis 3, t_3^j are tapering parameters along axes $j \neq 3$, s_3^j are shearing parameters, and b^{aj}_3 , b^{lj}_3 , b^{rj}_3 define the amount, location, and range of bending in the plane spanned by axis 3 and axis j .

The transformation along axis 1 is given by

$$\mathbf{r}_1 = \mathbf{T}_1(\mathbf{T}_3(\mathbf{e}; \mathbf{b})) = \begin{pmatrix} \mathbf{r}_{3_3} \\ A_1 \cos(\phi_1) - B_1 \sin(\phi_1) \\ A_1 \sin(\phi_1) + B_1 \cos(\phi_1) \end{pmatrix}, \quad (2.19)$$

where

$$A_1 = [(t_1^2 \mathbf{r}_{3_1}/a_0 a_1 w) + 1] \mathbf{r}_{3_2} + b^{a1}_1 \cos[(\mathbf{r}_{3_1} + b^{l1}_1)/(\mathbf{a}_0 a_1 w) \pi b^{r1}_1] + \mathbf{r}_{3_1} s_1^2, \quad (2.20)$$

$$B_1 = [(t_1^3 \mathbf{r}_{3_1}/a_0 a_1 w) + 1] \mathbf{r}_{3_3} + b^{a3}_1 \cos[(\mathbf{r}_{3_1} + b^{l3}_1)/(\mathbf{a}_0 a_1 w) \pi b^{r3}_1] + \mathbf{r}_{3_1} s_1^3 \quad (2.21)$$

and

$$\phi_1 = (\mathbf{r}_{3_1}/a_0 a_1 w) \pi \tau_1. \quad (2.22)$$

Here τ_1 is a twisting parameter along axis 1, t_1^j are tapering parameters along axes $j \neq 1$, s_1^j are shearing parameters, and b^{aj}_1 , b^{lj}_1 , b^{rj}_1 define the amount, location, and range of bending in the plane spanned by axis 1 and axis j .

Finally, the transformation along axis 2 is given by

$$\mathbf{s} = \mathbf{T}_2(\mathbf{T}_1(\mathbf{T}_3(\mathbf{e}; \mathbf{b}))) = \begin{pmatrix} A_2 \cos(\phi_2) - B_2 \sin(\phi_2) \\ \mathbf{r}_{1_2} \\ A_2 \sin(\phi_2) + B_2 \cos(\phi_2) \end{pmatrix}, \quad (2.23)$$

where

$$A_2 = [(t_2^3 \mathbf{r}_{1_2}/a_0 a_2 w) + 1] \mathbf{r}_{1_3} + b^{a3}_2 \cos[(\mathbf{r}_{1_2} + b^{l3}_2)/(\mathbf{a}_0 a_2 w) \pi b^{r3}_2] + \mathbf{r}_{1_2} s_2^3, \quad (2.24)$$

$$B_2 = [(t_2^1 \mathbf{r}_{1_2}/a_0 a_2 w) + 1] \mathbf{r}_{1_1} + b^{a1}_2 \cos[(\mathbf{r}_{1_2} + b^{l1}_2)/(\mathbf{a}_0 a_2 w) \pi b^{r1}_2] + \mathbf{r}_{1_2} s_2^1, \quad (2.25)$$

and

$$\phi_2 = (\mathbf{r}_{1_2}/a_0 a_2 w) \pi \tau_2. \quad (2.26)$$

Here τ_2 is a twisting parameter along axis 2, t_2^j are tapering parameters along axes $j \neq 2$, s_2^j are shearing parameters, and b^{aj}_2 , b^{lj}_2 , b^{rj}_2 define the amount, location, and range of bending in the plane spanned by axis 2 and axis j .

We collect the 39 global deformation parameters associated with \mathbf{s} into the vector

$$\mathbf{q}_s = (a_0, a_1, a_2, a_3, \epsilon_1, \epsilon_2, \tau_i, t_i^j, s_i^j, b^{aj}_i, b^{lj}_i, b^{rj}_i)^T, \quad (2.27)$$

where $i, j = 1, 2, 3, j \neq i$. The Jacobian matrix \mathbf{J} is a 3×39 matrix whose entries are computed in an analogous way as the Jacobian of the previous examples.

2.2.2 Local Deformations

In general [253], we can express the displacement \mathbf{d} anywhere within a deformable model as a linear combination of an infinite number of basis functions (e.g., polynomials) $b_j(\mathbf{u})$

$$\mathbf{d} = \sum_i \mathbf{S}_i \mathbf{q}_{d,i}, \quad (2.28)$$

where the diagonal matrix \mathbf{S}_i is formed from the basis functions and where $\mathbf{q}_{d,i}$ are local degrees of freedom or local generalized coordinates which depend only on time. The basis functions must be admissible; i.e., they must satisfy the kinematic boundary conditions of the model.

When we reduce the problem to finite dimensions, classical approximation methods such as the *Rayleigh-Ritz method* and the *Galerkin method* [114] are employed. These methods express the displacement \mathbf{d} in terms of a finite number of basis functions. In this case the series of (2.28) are truncated leading to

$$\mathbf{d} = \sum_{i=1}^n \mathbf{S}_i \mathbf{q}_{d,i}. \quad (2.29)$$

We can rewrite (2.29) in the compact matrix form

$$\mathbf{d} = \mathbf{S} \mathbf{q}_d, \quad (2.30)$$

where \mathbf{S} is the basis matrix whose elements are the basis functions b_j and the vector of local degrees of freedom \mathbf{q}_d consists of the local degrees of freedom $\mathbf{q}_{d,i}$

$$\mathbf{q}_d = (\dots, \mathbf{q}_{d,i}^T, \dots)^T. \quad (2.31)$$

In this book, we will use the finite element method [306], to compute the local displacement \mathbf{d} . Through this technique the deformable model is approximated by dividing it into a finite number of small regions called elements. The finite elements are assumed to be interconnected at nodal points on their boundaries. The local degrees of freedom \mathbf{q}_d can describe displacements, slopes and curvatures at selected nodal points on the deformable model. Between these selected nodal points the displacement field within the element is approximated using a finite number of interpolating polynomials called shape functions. The details of the finite element method and its use will be given in a following chapter.

KINEMATICS AND DYNAMICS

This chapter presents the kinematic and dynamic formulation of the deformable models. The kinematic formulation leads to the computation of a Jacobian matrix \mathbf{L} which allows the transformation of 3D vectors into q -dimensional vectors, where q is the number of geometric degrees of freedom of the deformable model. These parameters are also called generalized coordinates. The dynamic formulation is based on Lagrangian dynamics and uses generalized coordinates. We propose a systematic procedure for converting geometrically defined parameters into physical degrees of freedom. The resulting motion equations govern the evolution of generalized coordinates as a result of the application of external forces on the model. The forces stem either from data fitting techniques in vision applications or from interactions with simulated physical worlds in graphics applications. We also present a force-based estimation technique for shape and nonrigid motion estimation in computer vision applications.

The method for deriving the equations of motion is general across geometric primitives and deformations. The dynamic coupling of rigid-body motions and deformations that we derive is related to that described in [275, 276], but it is much more general, in part, because we must accommodate geometric primitives. The treatment of global deformation dynamics is similar to Witkin and Welch's [293] formulation of linearly deformable primitives, but we must deal with nonlinear global deformations and with the fact that the parametric equations defining common geometric primitives can be nonlinear (e.g., the ones defined by Barr [24]). The implicitly defined models of Sclaroff and Pentland [249] may be viewed as complementary to the global parametric deformations defined in this book; however, the approach is quite different, since it is not based on a modal analysis. One of the distinguishing features of our approach is that it combines the global/parameterized and local/free-form modeling paradigms within a single physically-based model. We incorporate local deformations into our model using finite element techniques described by Kardestuncer [135].

3.1 KINEMATICS

From (2.2), the velocity of a point on the model is given by

$$\begin{aligned}\dot{\mathbf{x}} &= \dot{\mathbf{c}} + \dot{\mathbf{R}}\mathbf{p} + \mathbf{R}\dot{\mathbf{p}} \\ &= \dot{\mathbf{c}} + \mathbf{B}\dot{\boldsymbol{\theta}} + \mathbf{R}\dot{\mathbf{s}} + \mathbf{R}\mathbf{S}\dot{\mathbf{q}}_d,\end{aligned}\quad (3.1)$$

where $\boldsymbol{\theta} = (\dots, \theta_j, \dots)^T$ is the vector of rotational coordinates of the model and $\mathbf{B} = [\dots \partial(\mathbf{R}\mathbf{p})/\partial\theta_j \dots]$. Furthermore,

$$\dot{\mathbf{s}} = \left[\frac{\partial \mathbf{s}}{\partial \mathbf{q}_s} \right] \dot{\mathbf{q}}_s = \mathbf{J} \dot{\mathbf{q}}_s, \quad (3.2)$$

where \mathbf{J} is the Jacobian of the reference shape with respect to the global deformation parameter vector (see the previous chapter for examples).

We can therefore write the model geometry as

$$\mathbf{x} = \mathbf{c} + \mathbf{R}(\mathbf{s} + \mathbf{d}) = \xi(\mathbf{q}), \quad (3.3)$$

and the model kinematics compactly as

$$\dot{\mathbf{x}} = [\mathbf{I} \ \mathbf{B} \ \mathbf{R}\mathbf{J} \ \mathbf{R}\mathbf{S}] \dot{\mathbf{q}} = \mathbf{L} \dot{\mathbf{q}}, \quad (3.4)$$

where ξ is a function that nonlinearly combines the generalized coordinates \mathbf{q} to compute the position \mathbf{x} of a point on the model, while \mathbf{L} is a model Jacobian matrix that maps generalized coordinates \mathbf{q} into 3D vectors.

Here,

$$\mathbf{q} = (\mathbf{q}_c^T, \mathbf{q}_{\theta}^T, \mathbf{q}_s^T, \mathbf{q}_d^T)^T, \quad (3.5)$$

with $\mathbf{q}_c = \mathbf{c}$ and $\mathbf{q}_{\theta} = \boldsymbol{\theta}$, serves as the vector of generalized coordinates for the dynamic model.

3.1.1 Computation of \mathbf{R} and \mathbf{B} Using Quaternions

We represent \mathbf{q}_{θ} using quaternions. Updating quaternions is easier than directly updating a rotation matrix and ensuring that it remains orthogonal. Quaternions also avoid the problems with “gimbal lock” and singularities that may arise when Euler angles are used to represent rotations [294].

A quaternion $\mathbf{q}_{\theta} = [s, \mathbf{v}_q]^T$ with unit magnitude [258],

$$\|[\mathbf{s}, \mathbf{v}_q]\| = s^2 + \mathbf{v}_q^T \mathbf{v}_q = 1, \quad (3.6)$$

specifies a rotation of the model from its reference position through an angle $\theta = 2 \cos^{-1} s$ around an axis aligned with vector $\mathbf{v}_q = (v_1, v_2, v_3)^T$ as shown in Fig. 3.1.

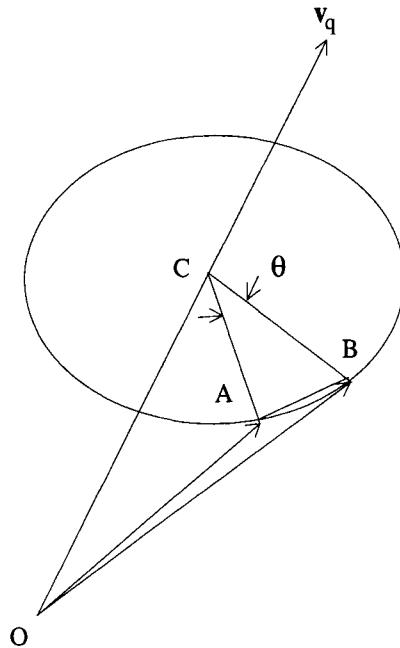


Figure 3.1 Rotation by angle θ through an axis of rotation \mathbf{v}_q .

The rotation matrix corresponding to $[s, \mathbf{v}_q]$ is

$$\mathbf{R} = \begin{bmatrix} 1 - 2(v_2^2 + v_3^2) & 2(v_1 v_2 - s v_3) & 2(v_1 v_3 + s v_2) \\ 2(v_1 v_2 + s v_3) & 1 - 2(v_1^2 + v_3^2) & 2(v_2 v_3 - s v_1) \\ 2(v_1 v_3 - s v_2) & 2(v_2 v_3 + s v_1) & 1 - 2(v_1^2 + v_2^2) \end{bmatrix}. \quad (3.7)$$

An important property of the rotation matrix \mathbf{R} is *orthogonality*

$$\mathbf{R}^T = \mathbf{R}^{-1}. \quad (3.8)$$

The matrix \mathbf{B} is given by [253] (see the Appendix for a proof)

$$\mathbf{B}(u) = -\mathbf{R} \tilde{\mathbf{p}}(\mathbf{u}) \mathbf{G}, \quad (3.9)$$

where \mathbf{R} represents the rotation matrix at time t , $\tilde{\mathbf{p}}(\mathbf{u})$ is the dual 3×3 matrix of the position vector $\mathbf{p}(\mathbf{u}) = (p_1, p_2, p_3)^T$ (see (2.3)) defined as

$$\tilde{\mathbf{p}}(\mathbf{u}) = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}, \quad (3.10)$$

and where \mathbf{G} is a 3×4 matrix whose definition is based on the value of the quaternion $\mathbf{q}_\theta = [s, \mathbf{v}_q]^T$ representing the rotation at time t

$$\mathbf{G} = 2 \begin{bmatrix} -v_1 & s & v_3 & -v_2 \\ -v_2 & -v_3 & s & v_1 \\ -v_3 & v_2 & -v_1 & s \end{bmatrix}. \quad (3.11)$$

3.2 DYNAMICS

In computer vision and in medical imaging applications (e.g., fitting of models to data, tracking of objects) our goal is to recover the model degrees of freedom \mathbf{q} , while in computer graphics (e.g., animations) we want to update \mathbf{q} . The components \mathbf{q}_c and \mathbf{q}_θ are the global rigid motion coordinates, \mathbf{q}_s are the global deformation coordinates, and \mathbf{q}_d are the local deformation coordinates of the model. Our approach carries out the generalized coordinate update procedure according to physical principles. We make our model dynamic in \mathbf{q} by introducing mass, damping, and a deformation strain energy. Through the apparatus of Lagrangian dynamics, we arrive at a set of equations of motion governing the behavior of our model under the action of externally applied forces. The following sections derive the Lagrange equations of motion for the geometrically defined models in the previous chapter.

3.2.1 Lagrange Equations of Motion

Let \mathcal{T} be the kinetic energy of the deformable model, \mathcal{F} the kinetic energy dissipation and \mathcal{E} the deformation strain energy of the model.

The Lagrange equations of motion for the model take the form

$$\frac{d}{dt} \left(\frac{\partial \mathcal{T}}{\partial \dot{\mathbf{q}}} \right)^T - \left(\frac{\partial \mathcal{T}}{\partial \mathbf{q}} \right)^T + \left(\frac{\partial \mathcal{F}}{\partial \dot{\mathbf{q}}} \right)^T + \delta_{\mathbf{q}} \mathcal{E} = \mathbf{f}_q. \quad (3.12)$$

These equations can be written in the form

$$\mathbf{M} \ddot{\mathbf{q}} + \mathbf{D} \dot{\mathbf{q}} + \mathbf{K} \mathbf{q} = \mathbf{g}_q + \mathbf{f}_q, \quad (3.13)$$

where \mathbf{M} , \mathbf{D} , and \mathbf{K} are the mass, damping, and stiffness matrices, respectively, where \mathbf{g}_q are inertial forces arising from the dynamic coupling between the local and global degrees of freedom, and where $\mathbf{f}_q(\mathbf{u}, t)$ are the generalized external forces associated with the degrees of freedom q of the model.

In the following subsections we derive from (3.12) formulas for the matrices and vectors in (3.13).

3.2.2 Kinetic Energy: Mass Matrix

The kinetic energy T of the model is given by

$$T = \frac{1}{2} \int \mu \dot{\mathbf{x}}^T \dot{\mathbf{x}} du = \frac{1}{2} \dot{\mathbf{q}}^T \left[\int \mu \mathbf{L}^T \mathbf{L} du \right] \dot{\mathbf{q}} = \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{M} \dot{\mathbf{q}}, \quad (3.14)$$

where $\mathbf{M} = \int \mu \mathbf{L}^T \mathbf{L} du$ is the symmetric mass matrix of the object and $\mu(u)$ is the mass density of the object. Using the expression for \mathbf{L} from (3.4), we can rewrite \mathbf{M} as a block symmetric matrix as follows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{cc} & \mathbf{M}_{c\theta} & \mathbf{M}_{cs} & \mathbf{M}_{cd} \\ \mathbf{M}_{c\theta} & \mathbf{M}_{\theta\theta} & \mathbf{M}_{\theta s} & \mathbf{M}_{\theta d} \\ \mathbf{M}_{cs} & \mathbf{M}_{\theta s} & \mathbf{M}_{ss} & \mathbf{M}_{sd} \\ \mathbf{M}_{cd} & \mathbf{M}_{\theta d} & \mathbf{M}_{sd} & \mathbf{M}_{dd} \end{bmatrix}, \quad (3.15)$$

where

$$\begin{aligned} \mathbf{M}_{cc} &= \int \mu \mathbf{I} du, & \mathbf{M}_{\theta s} &= \int \mu \mathbf{B}^T \mathbf{R} \mathbf{J} du, \\ \mathbf{M}_{c\theta} &= \int \mu \mathbf{B} du, & \mathbf{M}_{\theta d} &= \int \mu \mathbf{B}^T \mathbf{R} \mathbf{S} du, \\ \mathbf{M}_{cs} &= \mathbf{R} \int \mu \mathbf{J} du, & \mathbf{M}_{ss} &= \int \mu \mathbf{J}^T \mathbf{J} du, \\ \mathbf{M}_{cd} &= \mathbf{R} \int \mu \mathbf{S} du, & \mathbf{M}_{sd} &= \int \mu \mathbf{J}^T \mathbf{S} du, \\ \mathbf{M}_{\theta\theta} &= \int \mu \mathbf{B}^T \mathbf{B} du, & \mathbf{M}_{dd} &= \int \mu \mathbf{S}^T \mathbf{S} du. \end{aligned} \quad (3.16)$$

3.2.3 Calculation of Acceleration and Inertial Forces

The acceleration of a point \mathbf{x} on the deformable model is shown in [253] to satisfy

$$\ddot{\mathbf{x}} = \mathbf{L} \ddot{\mathbf{q}} + \dot{\mathbf{L}} \dot{\mathbf{q}}, \quad (3.17)$$

where

$$\dot{\mathbf{L}} \dot{\mathbf{q}} = \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{R} \mathbf{p}) + 2\boldsymbol{\omega} \times \mathbf{R} \dot{\mathbf{p}}. \quad (3.18)$$

Here, $\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{R} \mathbf{p})$ are the centrifugal and $2\boldsymbol{\omega} \times \mathbf{R} \dot{\mathbf{p}}$ are the Coriolis accelerations. We compute

$$\dot{\mathbf{p}} = \dot{\mathbf{s}} + \dot{\mathbf{d}} = \mathbf{J} \dot{\mathbf{q}}_s + \mathbf{S} \dot{\mathbf{q}}_d, \quad (3.19)$$

and the angular velocity of the deformable model with respect to the world coordinate system using [253]

$$\boldsymbol{\omega} = \mathbf{Q} \dot{\boldsymbol{\theta}}, \quad (3.20)$$

where \mathbf{Q} is a 3×4 matrix whose definition is based on the value of the quaternion $\boldsymbol{\theta} = \mathbf{q}_\theta = [s, (v_1, v_2, v_3)]^T$ representing the rotation at time t

$$\mathbf{Q} = 2 \begin{bmatrix} -v_1 & s & -v_3 & v_2 \\ -v_2 & v_3 & s & -v_1 \\ -v_3 & -v_2 & v_1 & s \end{bmatrix}. \quad (3.21)$$

The virtual work due to inertia on the deformable model is computed as follows [253]

$$\delta \mathbf{W}_I = \int \mu \delta \dot{\mathbf{x}}^T \ddot{\mathbf{x}} du = \int \delta \dot{\mathbf{q}}^T \mathbf{L}^T (\mathbf{L} \ddot{\mathbf{q}} + \dot{\mathbf{L}} \dot{\mathbf{q}}) du = \delta \dot{\mathbf{q}}^T (\mathbf{M} \ddot{\mathbf{q}} - \mathbf{g}_q), \quad (3.22)$$

where the generalized mass matrix is

$$\mathbf{M} = \int \mu \mathbf{L}^T \mathbf{L} du \quad (3.23)$$

and the generalized inertial forces are

$$\mathbf{g}_q = - \int \mu \mathbf{L}^T \dot{\mathbf{L}} \dot{\mathbf{q}} du. \quad (3.24)$$

Using (3.14) and (3.22), the first two terms of (3.12) which express inertial forces can be written as

$$\frac{d}{dt} \left(\frac{\partial \mathcal{T}}{\partial \dot{\mathbf{q}}} \right)^T - \left(\frac{\partial \mathcal{T}}{\partial \mathbf{q}} \right)^T = \mathbf{M} \ddot{\mathbf{q}} - \mathbf{g}_q, \quad (3.25)$$

where

$$\begin{aligned} \mathbf{g}_q &= -\dot{\mathbf{M}} \dot{\mathbf{q}} + \frac{1}{2} \left[\frac{\partial}{\partial \mathbf{q}} (\dot{\mathbf{q}}^T \mathbf{M} \dot{\mathbf{q}}) \right]^T \\ &= - \int \mu \mathbf{L}^T \dot{\mathbf{L}} \dot{\mathbf{q}} du \end{aligned} \quad (3.26)$$

gives the centrifugal and Coriolis forces [253].

3.2.4 Energy Dissipation: Damping Matrix

We assume velocity dependent kinetic energy dissipation, which can be expressed in terms of the (Raleigh) dissipation functional:

$$\mathcal{F} = \frac{1}{2} \int \gamma \dot{\mathbf{x}}^T \dot{\mathbf{x}} du, \quad (3.27)$$

where $\gamma(u)$ is a damping density. Since it has the same form as (3.14) we can rewrite (3.27) as follows:

$$\mathcal{F} = \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{D} \dot{\mathbf{q}}, \quad (3.28)$$

where the damping matrix \mathbf{D} has the same form as \mathbf{M} , except that γ replaces μ .

Using (3.14), we express the third term of (3.12)

$$\frac{\partial \mathcal{F}}{\partial \dot{\mathbf{q}}} = \mathbf{D} \dot{\mathbf{q}}. \quad (3.29)$$

3.2.5 Strain Energy: Stiffness Matrix

The stiffness matrix \mathbf{K} determines the global and local elastic properties of the model and has the general form

$$\mathbf{K} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ & & \mathbf{K}_{ss} & \mathbf{K}_{sd} \\ & & \mathbf{K}_{sd} & \mathbf{K}_{dd} \end{pmatrix}. \quad (3.30)$$

symmetric

The zero submatrices indicate that only the global \mathbf{q}_s and local \mathbf{q}_d deformational degrees of freedom can contribute to the stiffness matrix through their associated deformation strain energy. In particular \mathbf{K}_{ss} determines the stiffness of the model related to the global deformations, \mathbf{K}_{dd} determines the stiffness of the model related to the local deformations and \mathbf{K}_{sd} determines the stiffness as a result of the interaction between local and global deformations.

We will demonstrate a general technique of deriving \mathbf{K}_{ss} from a Hookean global deformation energy and \mathbf{K}_{dd} from a local deformation strain energy. We also assume independence of the above two defined energies which yields $\mathbf{K}_{sd} = \mathbf{0}$. Furthermore, in computer vision applications we want the global deformation parameters \mathbf{q}_s to freely account for as much of the data as possible. Consequently, we impose no deformation energy on \mathbf{q}_s , i.e., we set $\mathbf{K}_{ss} = \mathbf{K}_{sd} = \mathbf{0}$ in (3.30). The local deformation parameters \mathbf{q}_d , however, must be constrained to yield a small and continuous deformation function.

Global Strain Energy and Derivation of \mathbf{K}_{ss}

We assume that the energy \mathcal{E}_{s_i} associated with each of the global parameters \mathbf{q}_{s_i} is Hookean and given by the expression

$$\mathcal{E}_{s_i} = \frac{1}{2} k_{s_i} (\mathbf{q}_{s_i} - \mathbf{q}_{s_{i0}})^2, \quad (3.31)$$

where k_{s_i} is the stiffness associated with the global parameter \mathbf{q}_{s_i} and $\mathbf{q}_{s_{i0}}$ is the natural rest value associated with parameter \mathbf{q}_{s_i} . The corresponding global stiffness matrix \mathbf{K}_{ss} is a diagonal matrix whose nonzero entries are the k_{s_i} , i.e.,

$$\mathbf{K}_{ss} = \text{diag}(k_{s_i}). \quad (3.32)$$

Differentiating (3.31) with respect to \mathbf{q}_{s_i} , we derive the global elastic force associated with parameter \mathbf{q}_{s_i} ,

$$\mathbf{f}_{s_i} = k_{s_i} (\mathbf{q}_{s_i} - \mathbf{q}_{s_{i0}}). \quad (3.33)$$

Hence, if an external force acting on the model causes for example a bending deformation, then after that force vanishes the model deforms back to the rest value of the corresponding bending parameter.

Local Strain Energy and Derivation of \mathbf{K}_{dd}

Depending on the desired continuity of the deformable model surface, we impose on \mathbf{q}_d an appropriate deformation strain energy. We will now present two C^0 and C^1 continuous deformation strain energies. The first is that of a loaded membrane spline and the second of a thin plate under tension spline.

A loaded membrane deformation energy [276], suitable for C^0 continuous model surface, is given in continuous form by

$$\mathcal{E}_m(\mathbf{d}) = \int w_{10} \left(\frac{\partial \mathbf{d}}{\partial u} \right)^2 + w_{01} \left(\frac{\partial \mathbf{d}}{\partial v} \right)^2 + w_{00} \mathbf{d}^2 du, \quad (3.34)$$

where the function $w_{00}(u)$ controls the local magnitude and $w_{10}(u)$, $w_{01}(u)$ control the local variation of the deformation. In our implementation, we reduce these functions to scalar stiffness parameters $w_{00} = w_0$ and $w_{10} = w_{01} = w_1$.

A thin plate under tension deformation energy, suitable for C^1 continuous model surface, is given by the functional [272]

$$\begin{aligned} \mathcal{E}_p(\mathbf{d}) = & \int w_{20} \left(\frac{\partial^2 \mathbf{d}}{\partial u^2} \right)^2 + w_{11} \left(\frac{\partial^2 \mathbf{d}}{\partial u \partial v} \right)^2 + w_{02} \left(\frac{\partial^2 \mathbf{d}}{\partial v^2} \right)^2 + \\ & w_{10} \left(\frac{\partial \mathbf{d}}{\partial u} \right)^2 + w_{01} \left(\frac{\partial \mathbf{d}}{\partial v} \right)^2 + w_{00} \mathbf{d}^2 du. \end{aligned} \quad (3.35)$$

The nonnegative weighting functions w_{ij} control the elasticity of the material. The w_{10} and w_{01} functions control the tensions in the u and v directions, respectively. The w_{02} and w_{20} functions control the bending rigidities in the u and v directions, respectively. The w_{11} function controls the twisting rigidity. Increasing w_{01} and w_{10} makes the deformations have more membrane properties, while increasing the w_{20} , w_{11} and w_{02} , the deformations behave more like a thin plate. The weighting functions may be used to introduce depth and orientation discontinuities in the material. In our implementation however, we reduce these functions to scalar stiffness parameters $w_{ij}(u) = w_{ij}$.

We will now describe the general technique of deriving \mathbf{K}_{dd} from a local deformation strain energy \mathcal{E} . In accordance to the theory of elasticity, we can express a local deformation strain energy \mathcal{E} as

$$\mathcal{E} = \int \boldsymbol{\sigma}^T \boldsymbol{\epsilon} du, \quad (3.36)$$

where $\boldsymbol{\sigma}$ and $\boldsymbol{\epsilon}$ are the stress and strain vectors respectively. Furthermore we can always express the relation between the strain vector $\boldsymbol{\epsilon}$ and the local deformation \mathbf{d} as

$$\boldsymbol{\epsilon} = \mathcal{P} \mathbf{d}, \quad (3.37)$$

where \mathcal{P} is a differential operator that is derived from the local deformation strain energy (e.g., 3.34 and 3.35). In terms of the generalized local coordinates \mathbf{q}_d we can rewrite (3.37) as

$$\boldsymbol{\epsilon} = \mathcal{P} \mathbf{S} \mathbf{q}_d. \quad (3.38)$$

We can also express the relation between the stress σ and strains ϵ as

$$\sigma = \mathbf{D}\epsilon, \quad (3.39)$$

where the symmetric matrix \mathbf{D} is derived from the local deformation strain energy. Substituting (3.38) into (3.39) yields

$$\sigma = \mathbf{D}\mathcal{P}\mathbf{S}\mathbf{q}_d. \quad (3.40)$$

Substituting (3.38) and (3.40) into (3.36) we get

$$\mathcal{E} = \int \mathbf{q}_d^T (\mathcal{P}\mathbf{S})^T \mathbf{D}\mathcal{P}\mathbf{S}\mathbf{q}_d du, \quad (3.41)$$

where we utilize the symmetry of \mathbf{D} . Since \mathbf{q}_d depends only on time we can rewrite (3.41) as

$$\mathcal{E} = \mathbf{q}_d^T \left(\int (\mathcal{P}\mathbf{S})^T \mathbf{D}\mathcal{P}\mathbf{S} du \right) \mathbf{q}_d \quad (3.42)$$

or in compact form

$$\mathcal{E} = \mathbf{q}_d^T \mathbf{K}_{dd} \mathbf{q}_d, \quad (3.43)$$

where

$$\mathbf{K}_{dd} = \int (\mathcal{P}\mathbf{S})^T \mathbf{D}\mathcal{P}\mathbf{S} du \quad (3.44)$$

is the symmetric positive definite local deformation stiffness matrix.

In the next chapter we will give formulas for the elements of the stiffness matrix \mathbf{K}_{dd} using the above technique and the finite element method, for a loaded membrane deformation energy.

From (3.31) and (3.43) the fourth term of (3.12), the variation of \mathcal{E} with respect to \mathbf{q} , can be written

$$\delta_{\mathbf{q}} \mathcal{E} = \mathbf{K} \mathbf{q}. \quad (3.45)$$

3.2.6 External Forces and Virtual Work

The external forces $\mathbf{f}(u, t)$ applied to the model do virtual work which can be written as

$$\delta \mathbf{W}_F = \int \mathbf{f}^T \mathbf{L} \delta \mathbf{q} = \mathbf{f}_q^T \delta \mathbf{q}, \quad (3.46)$$

where

$$\mathbf{f}_q^T = \int \mathbf{f}^T \mathbf{L} du = (\mathbf{f}_c^T, \mathbf{f}_\theta^T, \mathbf{f}_s^T, \mathbf{f}_d^T), \quad (3.47)$$

and

$$\begin{aligned}\mathbf{f}_c^T &= \int \mathbf{f}^T d\mathbf{u}, & \mathbf{f}_s^T &= \int \mathbf{f}^T \mathbf{RJ} d\mathbf{u}, \\ \mathbf{f}_\theta^T &= \int \mathbf{f}^T \mathbf{B} d\mathbf{u}, & \mathbf{f}_d^T &= \int \mathbf{f}^T \mathbf{RS} d\mathbf{u},\end{aligned}\tag{3.48}$$

is the vector of generalized external forces associated with the degrees of freedom of the model.

3.3 CHOOSING THE ORDER OF THE MOTION EQUATIONS

For convenience, we rewrite the second-order equations (3.13) in standard dynamical system form, as the coupled set of first-order equations

$$\dot{\mathbf{u}}_q = \mathbf{F}\mathbf{u}_q + \mathbf{g},\tag{3.49}$$

with state vector \mathbf{u} , system matrix \mathbf{F} , and driving function \mathbf{g} as follows:

$$\mathbf{u}_q = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{q} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} -\mathbf{M}^{-1}\mathbf{D} & -\mathbf{M}^{-1}\mathbf{K} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \mathbf{M}^{-1}(\mathbf{g}_q + \mathbf{f}_q) \\ \mathbf{0} \end{bmatrix}.\tag{3.50}$$

A full implementation and simulation of the above dynamic equations would be appropriate for physics-based animation where it is important to achieve realistic motion synthesis [276]. Moreover, when the dynamical system is used to track moving objects and a significant part of a tracked object becomes occluded temporarily, significant portions of the generalized data force \mathbf{f}_q will suddenly vanish. The second-order system (3.13) (or, equivalently, (3.49) and (3.50)) is appropriate in such a case, since the mass term provides inertia; once the system is set in motion, the generalized coordinates will continue to evolve even when all the data forces vanish. Because of inertia, the system stands a better chance of regaining its lock on the object when it reappears, assuming the motion of the object did not undergo sudden changes during the occlusion.

However, in computer vision and geometric design applications involving the fitting of hollow “deformable shells” to static data, we may simplify the underlying motion equations for the sake of computational efficiency. In such applications we want the hollow “shell” to fit the data and achieve equilibrium as soon as the internal elastic forces balance the external data forces.¹ We usually care about the final equilibrium fit and not about the intermediate motion of the hollow “shell”. For static shape reconstruction problems (i.e., $\mathbf{z}(t) = \mathbf{z}$) it makes sense to simplify the motion equations by setting the mass density to

¹In computer vision the data forces are artificial and are computed using algorithms described in a subsequent chapter, while in computer graphics the forces are based on interactions in a simulated physical world.

zero, which nonetheless preserves useful first-order dynamics that achieve the equilibrium fit.

Setting the mass density $\mu(u)$ (see (3.14)) to zero in (3.13) results in the first-order dynamic system

$$\mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{f}_q \quad (3.51)$$

(since \mathbf{M} and \mathbf{g}_q vanish). Because these equations lack an inertial term, the system comes to rest as soon as all the forces equilibrate or vanish. We can also rewrite the first order-system (3.51) in the standard form (3.49), where

$$\mathbf{u}_q = \mathbf{q}, \quad \mathbf{F} = -\mathbf{D}^{-1}\mathbf{K}, \quad \mathbf{g} = \mathbf{D}^{-1}\mathbf{f}_q. \quad (3.52)$$

FINITE ELEMENT IMPLEMENTATION

We present the methodology for the computer implementation of the finite elements that we will employ to discretize the Lagrange equations of motion. The presentation examines the criteria for choosing the appropriate elements for a given problem, gives various finite element tessellations of the parametric space of a deformable model, shows examples of finite elements, and finally presents the finite element-based approximation to the Lagrange equations of motion.

We will be using the finite element method [27], to compute the local displacement \mathbf{d} . Through this technique the deformable model is approximated by a finite number of small regions called elements. The deformable model is partitioned by imaginary lines or surfaces into a number of finite elements that are assumed to be interconnected at nodal points on their boundaries. The local degrees of freedom \mathbf{q}_d can describe displacements, slopes and curvatures at selected nodal points on the deformable model. Between these selected nodal points the displacement field within the element \mathbf{d}^j is approximated using a finite number of interpolating polynomials called *shape functions*

$$\mathbf{d}^j = \sum_{i=1}^n N_i^j(\mathbf{u}) \mathbf{q}_{d,i}^j = \mathbf{N}^j \mathbf{q}_d^j, \quad (4.1)$$

where N_i^j are the element shape functions, matrix \mathbf{N}^j is the finite element shape matrix whose elements consist of the N_i^j , $\mathbf{q}_{d,i}^j$ is the element's nodal displacement, $\mathbf{q}_d^j = (\mathbf{q}_{d,1}^{j^T}, \dots, \mathbf{q}_{d,n}^{j^T})^T$, and n is the number of the element's nodes.

Using (4.1) we can express the displacement \mathbf{d} anywhere within the deformable model as

$$\mathbf{d} = \mathbf{S} \mathbf{q}_d, \quad (4.2)$$

where the basis matrix \mathbf{S} is computed from the finite element shape functions N_i^j .

A very important use of the finite element shape functions is the following. If we know the value of a point force $\mathbf{f}(\mathbf{u})$ within an element j , then we can

extrapolate it to the nodes of the element using the formula

$$\mathbf{f}_i = N_i(\mathbf{u})\mathbf{f}(\mathbf{u}), \quad (4.3)$$

where N_i is the shape function that corresponds to node i and \mathbf{f}_i is the extrapolated value of $\mathbf{f}(\mathbf{u})$ to node i .

4.1 CHOOSING THE APPROPRIATE ELEMENTS

Errors in the finite element method can be divided into two classes

1. Discretization errors resulting from geometric differences between the boundaries of the model and its finite element approximation.
2. Modeling errors, due to the difference between the true solution and its shape function representation.

Discretization errors can be reduced by using smaller elements; the errors tend to zero as the element size tends to zero. Shape function errors do not decrease as the element size reduces and may thus prevent convergence to the exact solution or even cause divergence. There are two main criteria required of the shape function to guarantee convergence [113]

1. *Completeness*: A complete polynomial¹ of order at least p , be used for the representation of the variable within an element, where p is the order of the highest derivative of the variable appearing in the energy functional.

In the following sections we will give descriptions of elements which guarantee C^0 or C^1 continuity of the approximated model surface.

2. *Conformity*: The elements must be conforming, that is, the representations of the variable and its derivatives up to and including order $p - 1$ must be continuous across interelement boundaries, where p is the order of the highest derivative appearing in the functional.

4.2 VARIOUS MODEL TESSELLATIONS

We will give two possible tessellations of the material coordinate system $\mathbf{u} = (u, v, 1)$ into finite element domains for the case of a deformable superquadric

¹In two dimensions a complete polynomial of order p can be written as $f(x, y) = \sum_{r=1}^l a_r x^i y^j, i + j \leq p$, where the number of terms in the polynomial is $l = (p + 1)(p + 2)/2$. In three dimensions a complete polynomial of order p can be written as $f(x, y, z) = \sum_{r=1}^l a_r x^i y^j z^k, i + j + k \leq p$, where the number of terms in the polynomial is $l = (p + 1)(p + 2)(p + 3)/6$.

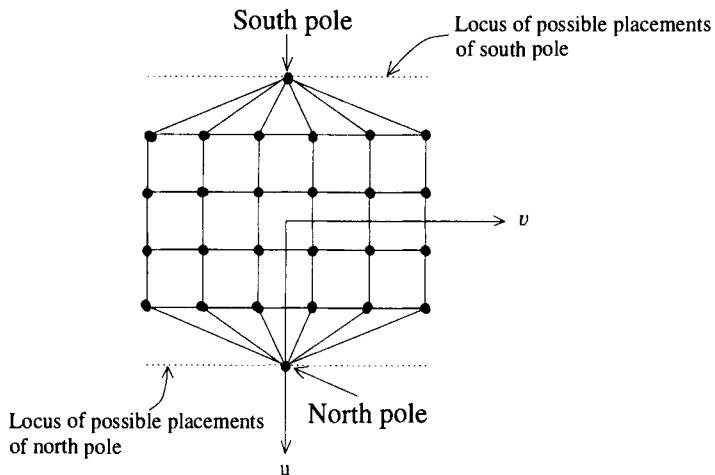


Figure 4.1 Model tessellation in material coordinates: rectangular and triangular elements.

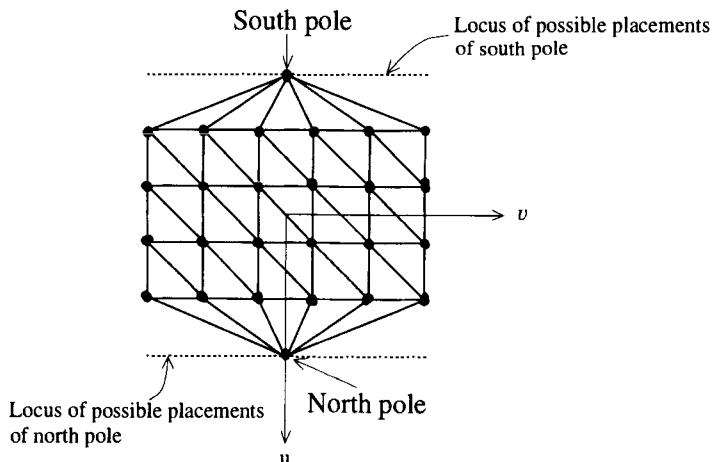


Figure 4.2 Model tessellation in material coordinates: triangular elements.

ellipsoid. The first is illustrated in Fig. 4.1. The need for quadrilateral and triangular elements is evident. Equation (2.7) implies that the v material coordinate of both north ($u = \pi/2$) and south ($u = -\pi/2$) poles may be arbitrary. This is illustrated in the figure by the dotted lines. We initially used this tessellation in vision applications. The problem though with using one rectangular instead of two triangular elements is that the shape representation is not so accurate. To achieve the latter we use the second representation shown in Fig. 4.2 where each of the rectangular elements has been replaced by two triangular elements. We use such a representation in applications requiring shape accuracy.

We will now give the geometry of the various kinds of elements and their corresponding shape functions N_i that we will be using throughout the book. We will be using elements with C^0 or C^1 continuity² across elements depending on the smoothness of the desired solution. In both cases each element and its shape functions are defined in a local reference coordinate system $\xi, \eta = (\xi, \eta)$. Furthermore, due to the model discretization in material coordinates \mathbf{u} , we give the relationship between the reference and the material coordinates. For simplicity, we will ignore the computation of the determinant of the jacobian matrix that relates the world coordinates \mathbf{x} and the material coordinates \mathbf{u} that should also be used in all subsequent integrations. This jacobian is computed from (2.2). All integrals are computed in the world coordinate system and therefore the integral of a function f is computed as follows

$$\int f(\mathbf{x}) d\mathbf{x} = \int f(\mathbf{u}) \det \mathbf{J}_{\mathbf{x}, \mathbf{u}} d\mathbf{u} = \int f(\xi, \eta) \det \mathbf{J}_{\mathbf{x}, \mathbf{u}} \det \mathbf{J}_{\mathbf{u}, \xi, \eta} d\xi d\eta, \quad (4.4)$$

where $\mathbf{J}_{\mathbf{i}, \mathbf{j}}$ represents the jacobian that relates coordinates in systems \mathbf{i} and \mathbf{j} .

4.3 C^0 ELEMENTS

We describe the bilinear quadrilateral and linear triangular elements that we employ in the above two tessellations. For the first tessellation shown in Fig. 4.1 we use bilinear quadrilateral and south and north linear triangular elements. For the second tessellation shown in Fig. 4.2 we use north, south and mid-region linear triangular elements. Instead of using different elements depending on the location in the grid, we could have used an isoparametric formulation [114] and use the same element everywhere within the grid. However, for simplicity we will follow the first approach.

²In a following section we will present the necessary criteria finite elements must satisfy to ensure a desired continuity in the solution.

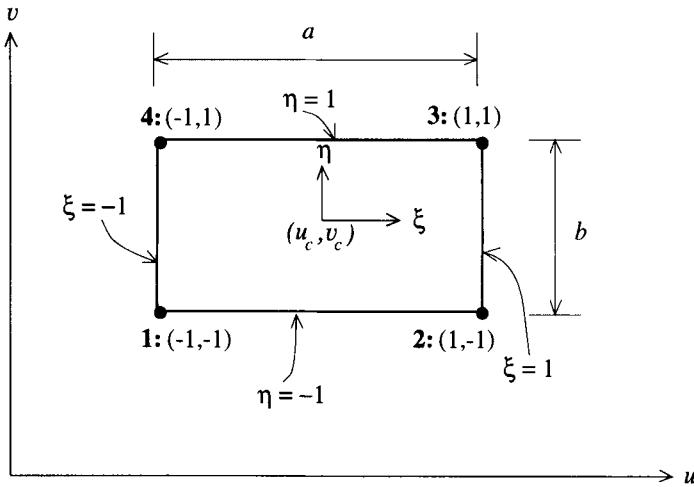


Figure 4.3 Bilinear quadrilateral element. The four nodes are numbered.

4.3.1 Bilinear Quadrilateral Elements

The nodal shape functions of the bilinear quadrilateral element (Fig. 4.3) are

$$N_i(\xi, \eta) = \frac{1}{4}(1 + \xi\xi_i)(1 + \eta\eta_i), \quad (4.5)$$

where \$(\xi_i, \eta_i)\$ are the reference coordinates of node \$i\$ shown in the figure. The relationship between the reference coordinates and material coordinates \$\mathbf{u} = (u, v)\$ is given by

$$\xi = \frac{2}{a}(u - u_c), \quad \eta = \frac{2}{b}(v - v_c), \quad (4.6)$$

where \$(u_c, v_c)\$ are the coordinates of the element center. The required derivatives of the shape functions may be computed as follows:

$$\frac{\partial N_i}{\partial u} = \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial u} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial u} = \frac{1}{2a}\xi_i(1 + \eta\eta_i), \quad (4.7)$$

$$\frac{\partial N_i}{\partial v} = \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial v} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial v} = \frac{1}{2b}(1 + \xi\xi_i)\eta_i, \quad (4.8)$$

and we may integrate a function \$f(u, v)\$ over \$E_j\$ by transforming to the reference coordinate system:

$$\iint_{E_j} f(u, v) dudv = \int_{-1}^1 \int_{-1}^1 f(\xi, \eta) \frac{ab}{4} d\xi d\eta. \quad (4.9)$$

We approximate such integrals using Gauss-Legendre quadrature rules (see the Appendix for various integration rules).

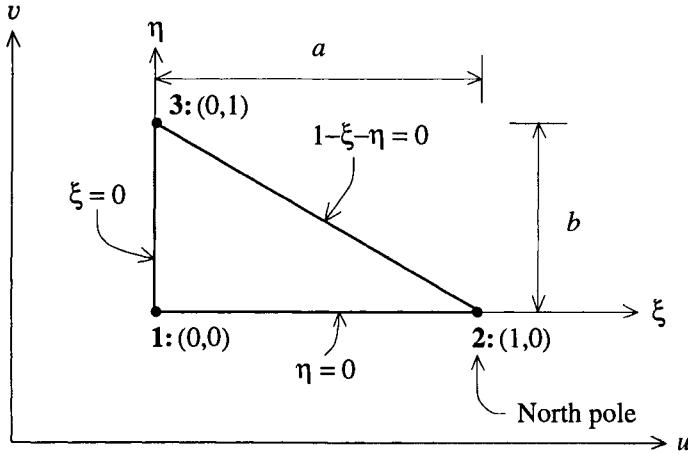


Figure 4.4 North pole linear triangular element. The three nodes are numbered.

4.3.2 North Pole Linear Triangular Elements

The nodal shape functions for the north pole linear triangular element (Fig. 4.4) are

$$N_1(\xi, \eta) = 1 - \xi - \eta, \quad (4.10)$$

$$N_2(\xi, \eta) = \xi, \quad (4.11)$$

$$N_3(\xi, \eta) = \eta. \quad (4.12)$$

The relationship between the uv and $\xi\eta$ coordinates is

$$\xi = \frac{1}{a}(u - u_1), \quad (4.13)$$

$$\eta = \frac{1}{b}(v - v_1), \quad (4.14)$$

where (u_1, v_1) are the coordinates of node 1 at which $(\xi_1, \eta_1) = (0,0)$. Computing the derivatives of the shape functions as in (4.7) and (4.8) yields

$$\frac{\partial N_1}{\partial u} = -\frac{1}{a}; \quad \frac{\partial N_2}{\partial u} = \frac{1}{a}; \quad \frac{\partial N_3}{\partial u} = 0; \quad (4.15)$$

$$\frac{\partial N_1}{\partial v} = -\frac{1}{b}; \quad \frac{\partial N_2}{\partial v} = 0; \quad \frac{\partial N_3}{\partial v} = \frac{1}{b}; \quad (4.16)$$

and we may integrate a function $f(u, v)$ over the E_j using

$$\iint_{E_j} f(u, v) \, du \, dv = \int_0^1 \int_0^{(1-\eta)} f(\xi, \eta) \, ab \, d\xi \, d\eta. \quad (4.17)$$

We approximate such integrals using Radau quadrature rules (see the Appendix).

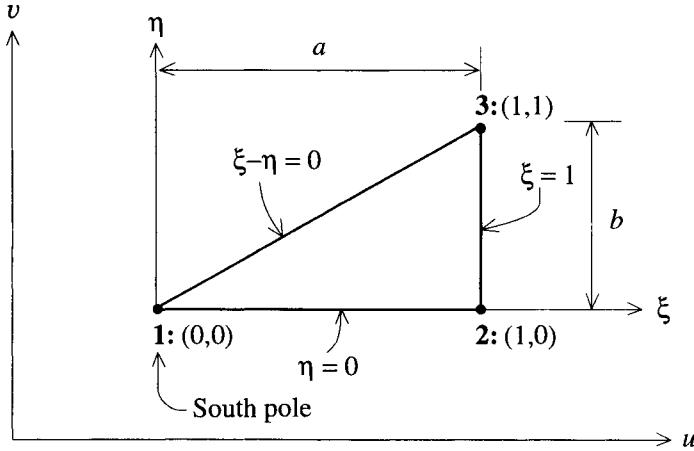


Figure 4.5 South pole linear triangular element. The three nodes are numbered.

4.3.3 South Pole Linear Triangular Elements

The nodal shape functions for the south pole linear triangular element (Fig. 4.5) are

$$N_1(\xi, \eta) = 1 - \xi, \quad (4.18)$$

$$N_2(\xi, \eta) = \xi, \quad (4.19)$$

$$N_3(\xi, \eta) = \eta. \quad (4.20)$$

The relationship between the uv and $\xi\eta$ coordinates is

$$\xi = \frac{1}{a}(u - u_1), \quad (4.21)$$

$$\eta = \frac{1}{b}(v - v_1), \quad (4.22)$$

where (u_1, v_1) are the coordinates of node 1 at which $(\xi_1, \eta_1) = (0, 0)$. Computing the derivatives of the shape functions as in (4.7) and (4.8) yields

$$\frac{\partial N_1}{\partial u} = -\frac{1}{a}; \quad \frac{\partial N_2}{\partial u} = \frac{1}{a}; \quad \frac{\partial N_3}{\partial u} = 0; \quad (4.23)$$

$$\frac{\partial N_1}{\partial v} = 0; \quad \frac{\partial N_2}{\partial v} = -\frac{1}{b}; \quad \frac{\partial N_3}{\partial v} = \frac{1}{b}; \quad (4.24)$$

and we may integrate a function $f(u, v)$ over the E_j using

$$\iint_{E_j} f(u, v) dudv = \int_0^1 \int_0^\xi f(\xi, \eta) ab d\eta d\xi. \quad (4.25)$$

We approximate such integrals using Radau quadrature rules.

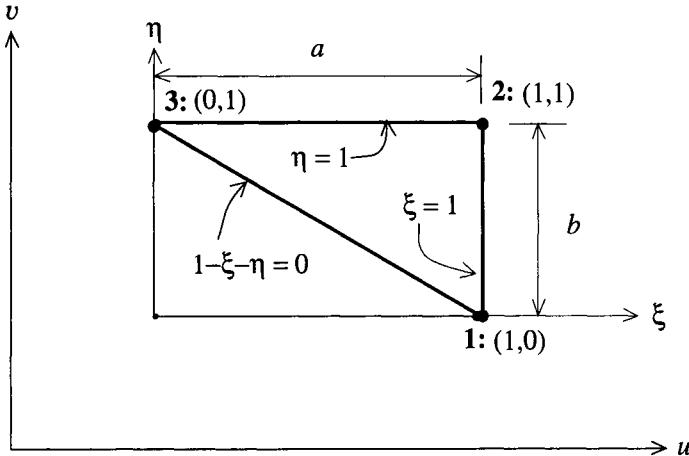


Figure 4.6 Mid-region triangular element. The three nodes are numbered.

4.3.4 Mid-Region Triangular Elements

The nodal shape functions for the mid-region linear triangular element (Fig. 4.6) are

$$N_1(\xi, \eta) = 1 - \eta, \quad (4.26)$$

$$N_2(\xi, \eta) = \xi + \eta - 1, \quad (4.27)$$

$$N_3(\xi, \eta) = 1 - \xi. \quad (4.28)$$

The relationship between the uv and $\xi\eta$ coordinates is

$$\xi = \frac{1}{a}(u - u_3), \quad (4.29)$$

$$\eta = \frac{1}{b}(v - v_1), \quad (4.30)$$

where u_3 and v_1 are the u and v coordinates of nodes 3 and 1 respectively at which $(\xi_3, \eta_3) = (0, 1)$ and $(\xi_1, \eta_1) = (1, 0)$. Computing the derivatives of the shape functions as in (4.7) and (4.8) yields

$$\frac{\partial N_1}{\partial u} = 0; \quad \frac{\partial N_2}{\partial u} = \frac{1}{a}; \quad \frac{\partial N_3}{\partial u} = -\frac{1}{a}; \quad (4.31)$$

$$\frac{\partial N_1}{\partial v} = -\frac{1}{b}; \quad \frac{\partial N_2}{\partial v} = \frac{1}{b}; \quad \frac{\partial N_3}{\partial v} = 0; \quad (4.32)$$

and we may integrate a function $f(u, v)$ over the E_j using

$$\iint_{E_j} f(u, v) dudv = \int_0^1 \int_{(1-\xi)}^1 f(\xi, \eta) ab d\eta d\xi. \quad (4.33)$$

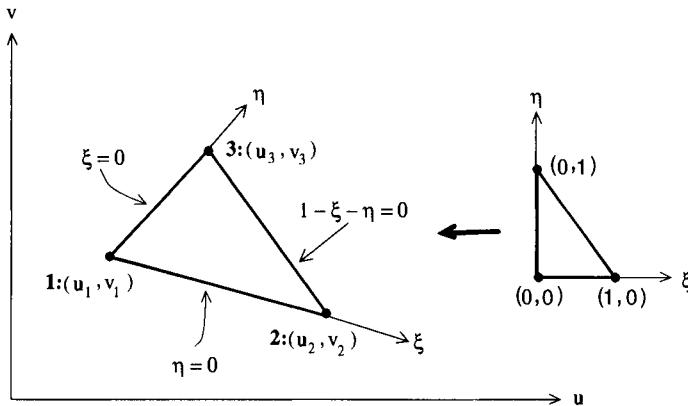


Figure 4.7 C^1 Continuous triangular element. The three nodes are numbered.

We approximate such integrals using Radau quadrature rules.

The above elements are conforming and satisfy the completeness requirements for a loaded membrane functional. For example, the shape functions of the rectangular elements are bilinear and contain the first order polynomial terms plus the xy of the higher order quadratic polynomial, therefore the elements are complete. Among element edges ($\xi = \pm 1, \eta = \pm 1$) these shape functions become linear and the displacement \mathbf{d} can be uniquely defined by the two corresponding nodal values of \mathbf{d} on that edge. If the adjacent element is also linear then \mathbf{d} will be continuous between elements since its values are uniquely defined by the shared nodal values on that edge. Therefore we have a C^0 conforming element.

4.4 C^1 TRIANGULAR ELEMENTS

Such elements can be used for the tessellation shown in Fig. 4.2 for problems requiring C^1 continuity.

The relationship between the uv and $\xi\eta$ coordinates is

$$u = (1 - \xi - \eta)u_3 + \xi u_1 + \eta u_2 \quad (4.34)$$

$$v = (1 - \xi - \eta)v_3 + \xi v_1 + \eta v_2, \quad (4.35)$$

where (u_i, v_i) are the material coordinates at the nodes of the triangular element.

The nodal variable for this element is a vector consisting of the nodal displacement \mathbf{d}_i , along with its first and second partial derivatives evaluated at each node i . The nodal variable vector for our models is therefore

$$\mathbf{q}_{d,i}(t) = \left[\mathbf{d}_i^T, \frac{\partial \mathbf{d}_i^T}{\partial u}, \frac{\partial \mathbf{d}_i^T}{\partial v}, \frac{\partial^2 \mathbf{d}_i^T}{\partial u^2}, \frac{\partial^2 \mathbf{d}_i^T}{\partial u \partial v}, \frac{\partial^2 \mathbf{d}_i^T}{\partial v^2} \right]^T. \quad (4.36)$$

Concatenating the $\mathbf{q}_{d,i}$ at each of the three nodes of element j , the 18-dimensional element nodal vector $\mathbf{q}_d^j = [\mathbf{q}_{d,1}^T, \mathbf{q}_{d,2}^T, \mathbf{q}_{d,3}^T]^T$ is obtained.

The 18 nodal shape functions $N_i(\xi, \eta)$ are given by the following formula.

For node 1

$$\begin{aligned} N_1 &= \lambda^2(10\lambda - 15\lambda^2 + 6\lambda^3 + 30\xi\eta(\xi + \eta)), & N_2 &= \xi\lambda^2(3 - 2\lambda - 3\xi^2 + 6\xi\eta), \\ N_3 &= \eta\lambda^2(3 - 2\lambda - 3\eta^2 + 6\xi\eta), & N_4 &= \frac{1}{2}\xi^2\lambda^2(1 - \xi + 2\eta), \\ N_5 &= \xi\eta\lambda^2, & N_6 &= \frac{1}{2}\eta^2\lambda^2(1 + 2\xi - \eta), \end{aligned}$$

for node 2

$$\begin{aligned} N_7 &= \xi^2(10\xi - 15\xi^2 + 6\xi^3 + 15\eta^2\lambda), & N_8 &= \frac{\xi^2}{2}(-8\xi + 14\xi^2 - 6\xi^3 - 15\eta^2\lambda), \\ N_9 &= \frac{\xi^2\eta}{2}(6 - 4\xi - 3\eta - 3\eta^2 + 3\xi\eta), & N_{10} &= \frac{\eta}{2}(2\xi(1 - \xi)^2 + 5\eta^2\lambda), \\ N_{11} &= \frac{\xi^2\eta}{2}(-2 + 2\xi + \eta + \eta^2 - \xi\eta), & N_{12} &= \frac{\xi^2\eta^2\lambda}{4} + \frac{\xi^3\eta^2}{2}, \end{aligned}$$

and for node 3

$$\begin{aligned} N_{13} &= \eta^2(10\eta - 15\eta^2 + 6\eta^3 + 15\xi^2\lambda), & N_{14} &= \frac{\xi\eta^2}{2}(6 - 3\xi - 4\eta - 3\xi^2 + 3\xi\eta), \\ N_{15} &= \frac{\eta^2}{2}(-8\eta + 14\eta^2 - 6\eta^3 - 15\xi^2\lambda), & N_{16} &= \frac{\xi^2\eta^2\lambda}{4} + \frac{\xi^3\eta^3}{2}, \\ N_{17} &= \frac{\xi\eta^2}{2}(-2 + \xi + 2\eta + \xi^2 - \xi\eta), & N_{18} &= \frac{\eta^2}{4}(2\eta(1 - \eta)^2 + 5\xi^2\lambda), \end{aligned}$$

where $\lambda = 1 - \xi - \eta$.

The computation of the derivatives of the shape functions with respect to the material coordinates is done in the same way as above, while we use the Radau quadrature rules to approximate integrals of functions $f(u, v)$ over the E_j , i.e.,

$$\iint_{E_j} f(u, v) dudv. \quad (4.37)$$

The element is complete up to fourth-order and its shape functions contain three fifth-order terms [69]. The shape functions are C^∞ continuous within elements and they ensure C^1 continuity between elements. Since (3.35) contains up to second order derivatives, the element is conforming. Therefore such an element is appropriate for a thin plate under tension deformation energy [188].

4.5 APPROXIMATION OF THE LAGRANGE EQUATIONS

We may approximate the Lagrange equations of motion (3.13) by using the finite element method described previously. Through this method, all quantities

necessary for the Lagrange equations of motion are derived from the same quantities computed independently within each finite element. The various matrices and vectors involved in the Lagrange equations of motion (see chapter 3) are assembled from matrices computed within each of the elements³, i.e.,

$$\mathbf{M} = \sum_j \mathbf{M}_{q \times q}^j; \quad \mathbf{D} = \sum_j \mathbf{D}_{q \times q}^j; \quad \mathbf{K} = \sum_j \mathbf{K}_{q \times q}^j; \quad \mathbf{f}_q = \sum_j \mathbf{f}_q^j; \quad \mathbf{g}_q = \sum_j \mathbf{g}_q^j, \quad (4.38)$$

where $\mathbf{M}_{q \times q}^j$, $\mathbf{D}_{q \times q}^j$, $\mathbf{K}_{q \times q}^j$, \mathbf{f}_q^j and \mathbf{g}_q^j are the appropriately expanded [114] mass, damping, stiffness matrices, external forces and inertial forces respectively associated with element j .

The calculation of all the above integrals is done within an element through the shape functions. Any quantity that must be integrated over an element is approximated using the shape functions and the corresponding nodal quantities. For example,

$$\mathbf{M}_{dd} = \int \mu \mathbf{N}^{j^T} \mathbf{N}^j du, \quad (4.39)$$

where \mathbf{N}^j is the shape matrix associated with element j .

4.5.1 Derivation of Stiffness Matrix \mathbf{K}_{dd}

We present the procedure of deriving the local stiffness matrix \mathbf{K}_{dd} by applying finite elements to a deformation energy first for the case of a loaded membrane energy (3.34).

We discretize the model in material coordinates \mathbf{u} using finite elements. We can derive \mathbf{K}_{dd} as an assembly of the local stiffness matrices \mathbf{K}_{dd}^j associated with each element domain $E_j \subset \mathbf{u}$. Since

$$\mathbf{d}(\mathbf{u}, t) = \mathbf{d}^j(\mathbf{u}, t) = [d_1^j(\mathbf{u}, t), d_2^j(\mathbf{u}, t), d_3^j(\mathbf{u}, t)]^T, \quad (4.40)$$

we can rewrite the membrane spline deformation energy (3.34) on E_j as the sum of component energies

$$\mathcal{E}^j(\mathbf{d}) = \mathcal{E}^j(d_1^j) + \mathcal{E}^j(d_2^j) + \mathcal{E}^j(d_3^j), \quad (4.41)$$

where for $k = 1, 2, 3$,

$$\mathcal{E}^j(d_k^j) = \int_{E_j} w_{10}^j \left(\frac{\partial d_k}{\partial u} \right)^2 + w_{01}^j \left(\frac{\partial d_k}{\partial v} \right)^2 + w_{00}^j d_k^2 du. \quad (4.42)$$

³The dimensionality of these matrices is equal to the number of degrees of freedom of the corresponding element. These matrices are then augmented to matrices of size $q \times q$, where q is the number of the model degrees of freedom.

In accordance to the theory of elasticity, (4.42) can be written in the form

$$\mathcal{E}^j(d_k^j) = \int_{E_j} \boldsymbol{\sigma}_k^{j^T} \boldsymbol{\epsilon}_k^j du, \quad (4.43)$$

where

$$\boldsymbol{\epsilon}_k^j = \left[\frac{\partial d_k^j}{\partial u}, \frac{\partial d_k^j}{\partial v}, d_k^j \right]^T \quad (4.44)$$

is the strain vector and

$$\boldsymbol{\sigma}_k^j = \mathbf{D}_k^j \boldsymbol{\epsilon}_k^j = \begin{pmatrix} w_{10}^j & 0 & 0 \\ 0 & w_{01}^j & 0 \\ 0 & 0 & w_{00}^j \end{pmatrix} \boldsymbol{\epsilon}_k^j \quad (4.45)$$

is the stress vector associated with component k of \mathbf{d} . Therefore, the element stress vector is

$$\boldsymbol{\sigma}^j = \mathbf{D}^j \boldsymbol{\epsilon}^j = \begin{pmatrix} \mathbf{D}_1^j & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2^j & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_3^j \end{pmatrix} \begin{pmatrix} \boldsymbol{\epsilon}_1^j \\ \boldsymbol{\epsilon}_2^j \\ \boldsymbol{\epsilon}_3^j \end{pmatrix}, \quad (4.46)$$

where $\mathbf{D}_1^j = \mathbf{D}_2^j = \mathbf{D}_3^j$.

We denote the finite element nodal shape functions by N_i^j , $i=1, \dots, n$, where n is the number of nodes associated with element E_j . Hence, we can write (4.44) as

$$\boldsymbol{\epsilon}_k^j = \sum_{i=1}^n \boldsymbol{\gamma}_i^j (q_{d,k}^j)_i = \boldsymbol{\Gamma}_k^j \mathbf{q}_{d,k}^j, \quad (4.47)$$

where

$$\boldsymbol{\gamma}_i^j = \left[\frac{\partial N_i^j}{\partial u}, \frac{\partial N_i^j}{\partial v}, N_i^j \right]^T, \quad (4.48)$$

$$\boldsymbol{\Gamma}_k^j = (\boldsymbol{\gamma}_1^j \boldsymbol{\gamma}_2^j \dots \boldsymbol{\gamma}_n^j)^T \quad (4.49)$$

and

$$\mathbf{q}_{d,k}^j = [(q_{d,k}^j)_1, (q_{d,k}^j)_2, \dots, (q_{d,k}^j)_n]^T. \quad (4.50)$$

We can write the element strain vector $\boldsymbol{\epsilon}^j$ as

$$\boldsymbol{\epsilon}^j = \begin{pmatrix} \boldsymbol{\epsilon}_1^j \\ \boldsymbol{\epsilon}_2^j \\ \boldsymbol{\epsilon}_3^j \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Gamma}_1^j & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Gamma}_2^j & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Gamma}_3^j \end{pmatrix} \begin{pmatrix} \mathbf{q}_{d,1}^j \\ \mathbf{q}_{d,2}^j \\ \mathbf{q}_{d,3}^j \end{pmatrix} = \boldsymbol{\Gamma}^j \mathbf{q}_d^j, \quad (4.51)$$

where $\boldsymbol{\Gamma}_1^j = \boldsymbol{\Gamma}_2^j = \boldsymbol{\Gamma}_3^j$. Thus the element stiffness matrix is

$$\mathbf{K}_{dd}^j = \int_{E_j} \boldsymbol{\Gamma}^{j^T} \mathbf{D}^j \boldsymbol{\Gamma}^j du = \text{diag} \left(\int_{E_j} \boldsymbol{\Gamma}_k^{j^T} \mathbf{D}_k^j \boldsymbol{\Gamma}_k^j du \right) = \text{diag}(\mathbf{K}_{ddk}^j). \quad (4.52)$$

For a thin plate under tension, the derivation is the same except that

$$\boldsymbol{\epsilon}_k^j = \left[d_k^j, \frac{\partial d_k^j}{\partial u}, \frac{\partial d_k^j}{\partial v}, \frac{\partial^2 d_k^j}{\partial u^2}, \frac{\partial^2 d_k^j}{\partial u \partial v}, \frac{\partial^2 d_k^j}{\partial v^2} \right]^T, \quad (4.53)$$

$$\mathbf{D}_k^j = \begin{bmatrix} w_{00}^j & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{10}^j & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{01}^j & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{20}^j & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{11}^j & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{02}^j \end{bmatrix}, \quad (4.54)$$

and $\boldsymbol{\gamma}_i^j$ has exactly the same form as $\boldsymbol{\epsilon}_d^j$ with N_i^j replacing d_k^j .

Using the above formulas for the case of a loaded membrane energy with $w_{10}(\mathbf{u}) = w_{01}(\mathbf{u}) = w_1$ and $w_{00}(\mathbf{u}) = w_0$ the elements of the symmetric stiffness matrix $\mathbf{K}_{dd_k}^j$ corresponding to a bilinear rectangular element are given by the following formulas

$$\begin{aligned} \mathbf{K}_{11} &= ab\left(\frac{w_1}{3a^2} + \frac{w_1}{3b^2} + \frac{w_0}{9}\right), \\ \mathbf{K}_{12} = \mathbf{K}_{21} &= \frac{ab}{4}\left(-\frac{4w_1}{3a^2} + \frac{2w_1}{3b^2} + \frac{2w_0}{9}\right), \\ \mathbf{K}_{13} = \mathbf{K}_{31} &= \frac{ab}{4}\left(-\frac{2w_1}{3a^2} - \frac{2w_1}{3b^2} + \frac{w_0}{9}\right), \\ \mathbf{K}_{14} = \mathbf{K}_{41} &= \frac{ab}{4}\left(\frac{2w_1}{3a^2} - \frac{4w_1}{3b^2} + \frac{2w_0}{9}\right), \\ \mathbf{K}_{22} &= ab\left(\frac{w_1}{3a^2} + \frac{w_1}{3b^2} + \frac{w_0}{9}\right), \\ \mathbf{K}_{23} = \mathbf{K}_{32} &= \frac{ab}{4}\left(\frac{2w_1}{3a^2} - \frac{4w_1}{3b^2} + \frac{2w_0}{9}\right), \\ \mathbf{K}_{24} = \mathbf{K}_{42} &= \frac{ab}{4}\left(-\frac{2w_1}{3a^2} - \frac{2w_1}{3b^2} + \frac{w_0}{9}\right), \\ \mathbf{K}_{33} &= ab\left(\frac{w_1}{3a^2} + \frac{w_1}{3b^2} + \frac{w_0}{9}\right), \\ \mathbf{K}_{34} = \mathbf{K}_{43} &= \frac{ab}{4}\left(-\frac{4w_1}{3a^2} + \frac{2w_1}{3b^2} + \frac{2w_0}{9}\right), \\ \mathbf{K}_{44} &= ab\left(\frac{w_1}{3a^2} + \frac{w_1}{3b^2} + \frac{w_0}{9}\right), \end{aligned} \quad (4.55)$$

while for a linear south pole element the elements of $\mathbf{K}_{dd_k}^j$ are

$$\begin{aligned} \mathbf{K}_{11} &= \frac{ab}{2}\left(\frac{w_1}{a^2} + \frac{w_0}{6}\right), \\ \mathbf{K}_{12} = \mathbf{K}_{21} &= \frac{ab}{2}\left(-\frac{w_1}{a^2} + \frac{w_0}{12}\right), \\ \mathbf{K}_{13} = \mathbf{K}_{31} &= \frac{ab}{24}w_0, \end{aligned}$$

$$\begin{aligned}\mathbf{K}_{22} &= \frac{ab}{2} \left(\frac{w_1}{b^2} + \frac{w_1}{a^2} + \frac{w_0}{6} \right), \\ \mathbf{K}_{23} = \mathbf{K}_{32} &= \frac{ab}{2} \left(-\frac{w_1}{b^2} + \frac{w_0}{12} \right), \\ \mathbf{K}_{33} &= \frac{ab}{2} \left(\frac{w_1}{b^2} + \frac{w_0}{6} \right).\end{aligned}\tag{4.56}$$

APPLIED FORCES

We will present various techniques for computing external forces and for applying them to deformable models. In computer vision and medical imaging applications, such forces originate from datapoints or image potentials and are assigned to points on the deformable model [275]. In computer graphics applications, we assign forces to points on the deformable model due to collisions of the deformable model with other rigid or deformable models, or due to a force field (e.g., gravity) [277]. Finally, we describe our force-based technique for shape and nonrigid motion estimation.

5.1 COMPUTER VISION AND MEDICAL IMAGING APPLICATIONS

In the dynamic model fitting process, the data are transformed into an externally applied force distribution $\mathbf{f}(\mathbf{u}, t)$. Using (3.48), we convert the external forces to generalized forces \mathbf{f}_q which act on the generalized coordinates of the model. We employ two types of forces based on the structure of the input data. For regularly sampled image or voxel data we assign short-range forces¹ obtained through gradients of potential functions. For 3D range data we assign long-range forces² based on distances between data points and the model's surface.

5.1.1 Short Range Forces

We will describe various techniques for generating suitable potential functions from monocular, binocular, and dynamic image sequences. For example, to attract a 3D model towards significant intensity gradients in a continuous image

¹The strength of such forces increases close to the image or voxel data.

²Long range forces are spring-like forces and therefore are proportional to the distances between data points and the corresponding nodes on the model's surface.

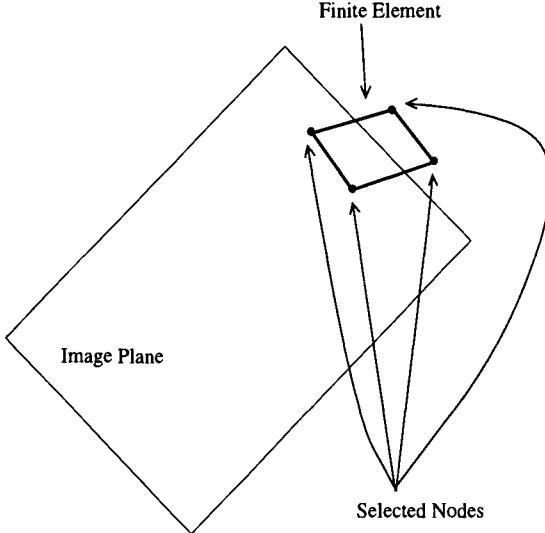


Figure 5.1 Application of image forces to nodes.

$I(x, y)$, we construct the potential function [275]

$$P(x, y) = \|\nabla(G_\sigma * I)\|, \quad (5.1)$$

where G_σ denotes a Gaussian smoothing filter of characteristic width σ , which determines the extent of the region of attraction of the intensity gradient. Typically, the attraction has a relatively short range. The potential function applies a force

$$\mathbf{f} = \beta \nabla P(\Pi \mathbf{x}) \quad (5.2)$$

to the model, where β controls the strength of the force and Π is a suitable projection (e.g., perspective, orthographic) of points on the model into the image plane. We now discuss two ways of selecting points on the model on which to apply short range forces using orthographic projection, assuming that the surface of the deformable model has been tessellated into triangular elements (see chapter 4 for more details).

Algorithm 1

We select those nodes from the finite elements whose vertical distance from the image plane shown in Fig. 5.1, is less than an experimentally defined threshold ϵ . To each of those nodes we apply the forces (5.2).

Algorithm 2

We first compute the intersection points between the edges of the finite elements and the image plane shown in Fig. 5.2. We then assign to each intersection

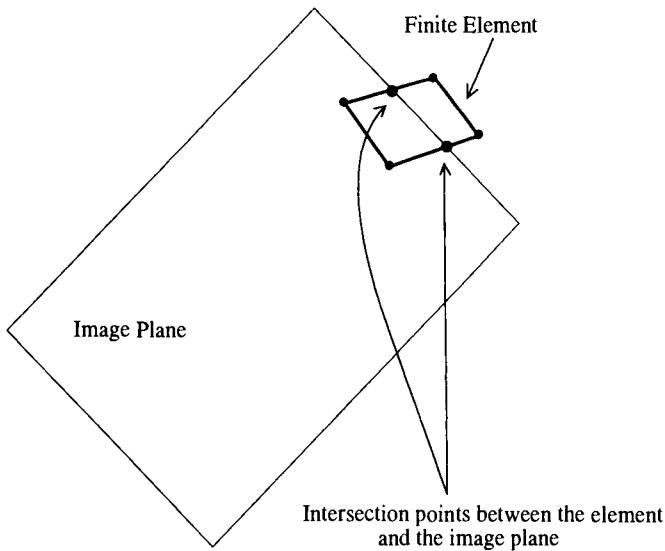


Figure 5.2 Accurate application of image forces to nodes.

point \mathbf{u}_i ; the force (5.2). We then extrapolate this force to each of the two corresponding nodes of the triangular element using the shape functions

$$\mathbf{f}_k = N_k(\mathbf{u}_i)\mathbf{f}(\mathbf{u}_i), \quad (5.3)$$

where $k \in \{1, 2, 3\}$, \mathbf{f}_k is the nodal force and N_k is the shape function corresponding to node k . This algorithm computes the corresponding forces to the element's nodes more accurately than the previous one, but is more computationally expensive.

To compute the potential function in practice, we begin with a digital image $I(i, j)$, convolve it with a discrete filter G_σ , and compute at each pixel (i, j) the magnitude of the discrete gradient operator calculated from central finite differences of neighboring pixel values. To evaluate (5.2) at the location of a projected model point $\Pi\mathbf{x} = (x, y)$, we first calculate using central finite differences the discrete gradients ∇P_k at the four pixels $k = 1, \dots, 4$ that surround (x, y) . We then consider these pixels as the nodes of the quadrilateral finite element of Fig. 4.3, with $a = b = 1$, in order to define a bilinear interpolant in the region between the pixels; i.e., using (4.5) and (4.6), the interpolant is given by

$$\nabla P(x, y) = \sum_{k=1}^4 N_k(2(x - x_c), 2(y - y_c)) \nabla P_k, \quad (5.4)$$

where (x_c, y_c) denotes the centroid of the four pixels and N_k are given by the same formulas as the shape functions of a bilinear rectangular element.

The above techniques for the calculation of short-range forces can be generalized in a straightforward manner in case of 3D potentials computed from 3D data.

5.1.2 Long Range Forces

Alternatively, we may define long range forces

$$\mathbf{f}(\mathbf{u}_z) = \beta \|\mathbf{z} - \mathbf{x}(\mathbf{u}_z)\|, \quad (5.5)$$

based on the separation between a datapoint \mathbf{z} in space and the force's point of influence \mathbf{u}_z on the model's surface. In general, $\mathbf{u}_z = (u_z, v_z)$ will fall somewhere within an element on the surface of the model. We can compute $\mathbf{x}(\mathbf{u}_z)$ in the domain of a quadrilateral element, for instance, according to its bilinear local interpolant

$$\mathbf{x}(\mathbf{u}_z) = \sum_{i=1}^4 N_i(2(u_z - u_c)/a, 2(v_z - v_c)/b) \mathbf{x}_i, \quad (5.6)$$

where the \mathbf{x}_i are the nodal positions (see (4.5) and (4.6)), and (u_c, v_c) is the location of the centroid of the quadrilateral. The equivalent forces on each of the four nodes are

$$\mathbf{f}_i = N_i(2(u_z - u_c)/a, 2(v_z - v_c)/b) \mathbf{f}(\mathbf{u}_z). \quad (5.7)$$

When we use triangular elements, the computations proceed in an analogous fashion using the corresponding shape function formulas (see chapter 5).

Minimum Distance Based Forces

In many applications, we want \mathbf{u}_z to minimize the distance d between a given datapoint \mathbf{z} and the model, where

$$d(\mathbf{u}) = \|\mathbf{z} - \mathbf{x}(\mathbf{u})\|. \quad (5.8)$$

A closed form analytic formula for $\mathbf{x}(\mathbf{u}_z)$ is unavailable for a discrete deformable model. There are various algorithms for computing $d(\mathbf{u})$ which achieve various levels of accuracy in the computation of the minimum $d(\mathbf{u}) = d(\mathbf{u}_z)$ depending on whether they use

1. finite element nodes,
2. finite elements,
3. both finite element nodes and finite elements.

We will now describe four algorithms in order of increasing accuracy in the computation of d . Furthermore, in assessing the complexity of each algorithm we will use the fact that the number of elements is proportional to the number of nodes. The constant of proportionality depends on the geometry of the mesh used to discretize the model surface.

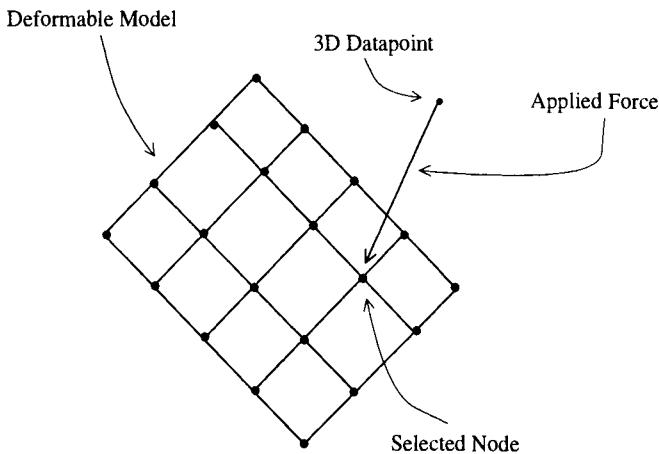


Figure 5.3 Force application to the model node with minimal distance from a 3D datapoint.

Algorithm 1

A brute-force approach that worked well in our experiments is to select from all the model nodes the one that minimizes d as shown in Fig. 5.3. The complexity of this operation is $\Theta(mn)$, where m is the number of datapoints and n is the number of nodes.

Algorithm 2

A more accurate algorithm involves a dynamic procedure for migrating points of influence over the model's surface until they arrive at locations of minimal distance from the given datapoints. Starting from initial points of influence not necessarily at minimal distance (Fig. 5.4), we project the force at each time step onto the unit tangent vectors $(\partial \mathbf{x} / \partial u) / \| \partial \mathbf{x} / \partial u \|$ and $(\partial \mathbf{x} / \partial v) / \| \partial \mathbf{x} / \partial v \|$ to the surface at the current point of influence $P_0 = \mathbf{u}_0$, and we migrate the point in the \mathbf{u} plane by taking an Euler step in u and v proportional to the magnitude of the respective projections. Thus, the point of influence migrates to a point P_r of minimal distance, where the tangential components of the force vanish. The scheme works well, unless the surface is highly convoluted, in which case it may converge to a local minimum. The complexity of the algorithm is $\Theta(mn)$.

Algorithm 3

In this algorithm we first apply Algorithm 1 to find the model node which has the smallest distance $d(\mathbf{u})$ from a given datapoint. We then use a minimization procedure³ within each of the elements that the previously selected model node belongs as shown in Fig. 5.5. We perform the minimization within each element

³The shape functions determine whether the minimization is linear or nonlinear

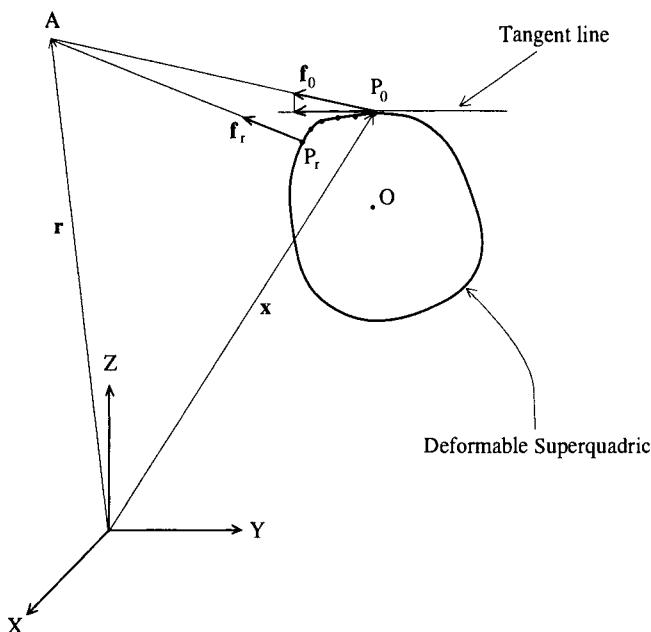


Figure 5.4 Migration of data force point of influence over model surface.

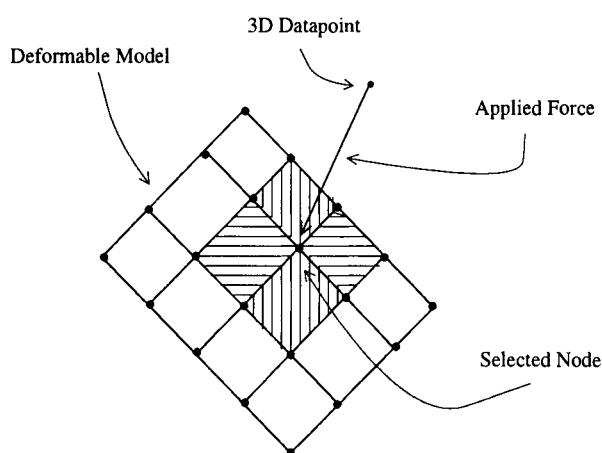


Figure 5.5 Force application to a model point with minimal distance from a 3D datapoint. The model point lies within the elements (shaded) which share the model node with minimal distance from the 3D datapoint.

j using the element's shape functions

$$d(\mathbf{u}) = \min^j \|\mathbf{z} - \mathbf{x}(\mathbf{u})\| = \min^j \left\| \mathbf{z} - \sum_{i=1}^n N_i(\mathbf{u}) \mathbf{x}_i \right\|, \quad (5.9)$$

where N_i is the shape function corresponding to node i whose nodal position is \mathbf{x}_i and j is a finite element from the set E of finite elements that comprise the model surface. The complexity of the algorithm is $\Theta(mn)$.

Algorithm 4

In this algorithm for each datapoint we compute the point in each finite element whose distance minimizes (5.9). From these computed model points we select the one whose distance from the datapoint is the minimum of all the computed distances. The complexity of the algorithm is $\Theta(mn)$.

Algorithm 5

This is a faster version of algorithm 4. In most shape estimation applications, the surface shape does not vary significantly from one step of the integration to another. In such cases, we determine initially based on algorithm 4 the attachment of all datapoints to points on the surface. At each subsequent step, for each datapoint, we compute (5.9) on finite elements which are in the neighborhood of of the finite element whose point \mathbf{u}_z has a minimum distance from the datapoint in the previous step. The complexity of the algorithm is $\Theta(m)$.

In cases where there are outliers in the datapoints, a robust statistics-based method is necessary in order to remove them. In most of our applications we use the above algorithms. However, there are many other types of long range forces whose use depends on the particular applications. Among them, a very useful type of forces are the radial forces.

Radial Forces

In some applications we want to compute radial forces in the direction of the line connecting a datapoint \mathbf{z} to the model frame shown in Fig. 5.6. The intersection of this line with the model surface defines a point $\mathbf{x}(\mathbf{u})$ on the model to which this force will be exerted. The force is then computed using (5.5). This kind of force is only suitable for convex models because otherwise the radial line might intersect with the model surface in more than one point. Furthermore, when the original shape of the model is “far” from the shape of the datapoints, such forces offer a more uniform force distribution than forces based on minimum distance. In the latter case, several datapoints can be assigned to the same node or the same point within a finite element therefore concentrating the assignment of forces around particular nodes. Once the model shape and the datapoints are “close” the radial and minimum distance based force assignment algorithms become equivalent.

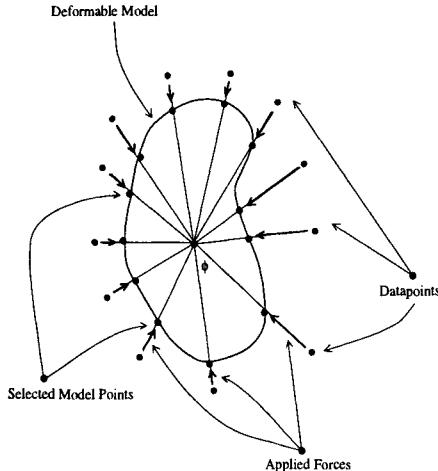


Figure 5.6 Application of radial forces to model points.

The above force assignment algorithms need not be executed at every iteration of the program. They can be used for a period of l iterations, where l depends on the amount of model deformation at every iteration. Furthermore, in applications where twisting deformations are involved, any of the above force assignment algorithms should be applied only once initially. In this way the assignment of datapoints to model points is constant over time, allowing for recovery of twisting deformations.

Depending on the application, other types of forces may be necessary to define. Balloon forces, for example, were introduced by Cohen and Cohen [53] to improve the boundary segmentation of internal organs (e.g., the heart) based on the use of active contours.

5.2 COMPUTER GRAPHICS APPLICATIONS

We compute forces stemming from collisions of rigid or deformable models using the following two algorithms. This one of the two possible approaches to deal with collisions. The other alternative is to use a momentum balance approach [254, 255] and is not going to be discussed here. The approach we present here is general in the sense that it can deal with collisions, rolling and resting contact.

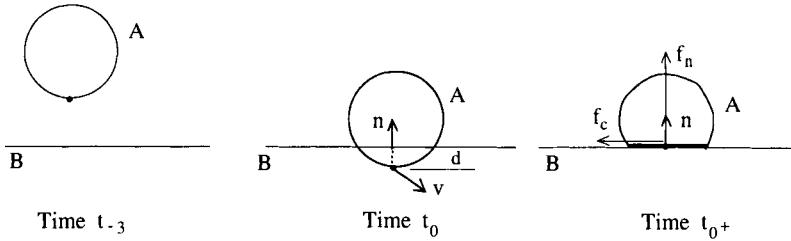


Figure 5.7 Collision of a deformable with a rigid model.

5.2.1 Collisions of Deformable with Rigid Models

Suppose that a deformable model is colliding with a rigid model as illustrated in Fig. 5.7. At every time step we check whether a node has penetrated the rigid model. In Fig. 5.7 where the deformable model collides with a rigid ground plane at time t_0 , we trivially do so by checking the sign of the inner product between the ground normal and the vector between a node on the deformable model and an arbitrary point on the ground. If that inner product is positive, then no penetration has occurred. Otherwise penetration has occurred and we take the following steps.

1. We project each of the penetrating nodes along the normal of the collision plane ⁴ back to the surface of the rigid object. This guarantees that no penetration will ever appear.
2. We calculate the new local deformation \mathbf{d} corresponding to each penetrating node and update the vector of local deformations \mathbf{q}_d . We then assign two forces to each penetrating node due to the collision.

- (a) We first compute the new elastic forces of the deformable model

$$\mathbf{f}_{q_e} = \mathbf{K}\mathbf{q}_d, \quad (5.10)$$

where \mathbf{K} is the global stiffness matrix. For each penetrating node we then select the corresponding component \mathbf{f}_i of \mathbf{f}_{q_e} and assign to that node a force \mathbf{f}_n along the unit normal \mathbf{n} of the collision plane

$$\mathbf{f}_n = a(\mathbf{f}_i \cdot \mathbf{n})\mathbf{n}, \quad (5.11)$$

where a is a constant that models the elasticity of the collision.

- (b) We compute the projection \mathbf{v}_p of the nodal velocity \mathbf{v} on the collision plane. Assuming Coulomb friction we assign to the node the following frictional force

$$\mathbf{f}_c = -\beta_f \|\mathbf{f}_n\| \frac{\mathbf{v}_p}{\|\mathbf{v}_p\|}, \quad (5.12)$$

⁴The collision plane is the tangent plane between two surfaces at the point of collision.

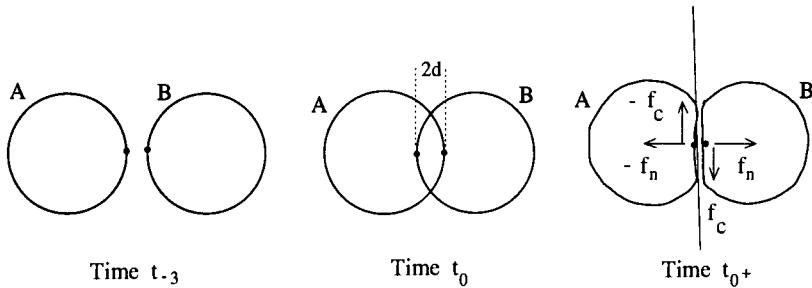


Figure 5.8 Collision of two deformable models.

where $0 \leq \beta_f \leq 1$ is the friction coefficient.

5.2.2 Collisions between Deformable Models

Let's assume that two deformable models will penetrate one another as shown in Fig. 5.8. Then at every time step we apply the following algorithm

1. We assume that the surface of each of the deformable models has been discretized using finite elements and we approximate each finite element with a polygon. We check if a polygon from the first model intersects with any polygons of the second model. If it does, then we mark the nodes of this polygon and the nodes of the intersecting polygons of the second model. For every marked node of the second deformable model we associate a marked node from the first model that has the smallest distance from that node. We thus form pairs of nodes from the selected marked nodes.
2. For each such pair of nodes we check for interpenetration using the following criterion. If the distance between the first models' model frame and the chosen node on the first model is greater than the distance between the frame and the node on the second model, then interpenetration has occurred at these two nodes.
3. We compute the plane at the midpoint which is perpendicular to the line connecting those two model nodes and project both nodes to this plane, along the normal to the plane (time t_0 in Fig. 5.8).
4. We assign two forces \mathbf{f}_n and $-\mathbf{f}_n$, which have opposite directions, but equal magnitudes and are perpendicular to the above defined plane. Such forces are computed in the same way as in the previous algorithm.
5. Assuming Coulomb friction we assign as in the previous algorithm two frictional forces \mathbf{f}_c and $-\mathbf{f}_c$, which have opposite directions, but equal magnitudes

$$\|\mathbf{f}_c\| = b_f \|\mathbf{f}_n\| \quad (5.13)$$

and lie on the collision plane.

The above two described algorithms work better when the Euler step of the simulation is reduced when a collision is detected. In such a case penetration is more gradual and the computed collision forces are smaller resulting in more realistic simulations. The Euler step is increased again after the collision to its value before the instant of collision detection. For a more detailed description of these methods see [226].

5.3 FORCE-BASED ESTIMATION

We present our force-based estimation technique for shape and nonrigid motion in computer vision and medical imaging applications. We take a physics-based approach to visual estimation. The visual data are converted into traction forces and applied to the dynamic model based on the previously presented algorithms. We fit the model by integrating the equations of motion (3.49) driven by these forces.

More specifically, let the observation vector $\mathbf{z}(t)$ denote time-varying input data. Using (3.3) we can relate $\mathbf{z}(t)$ to the model's state vector $\mathbf{u}_q(t)$ as follows:

$$\mathbf{z} = \mathbf{h}(\mathbf{u}_q) + \mathbf{v}, \quad (5.14)$$

where $\mathbf{v}(t)$ represents uncorrelated measurement errors as a zero mean white noise process with known covariance $\mathbf{V}(t)$, i.e., $\mathbf{v}(t) \sim \mathbf{N}(0, \mathbf{V}(t))$ ⁵. In force-based estimation, the rate of change of the estimated model state vector $\hat{\mathbf{u}}_q$ is given from (3.49) by

$$\dot{\hat{\mathbf{u}}}_q = \mathbf{F}\hat{\mathbf{u}}_q + \mathbf{g}. \quad (5.15)$$

Furthermore, according to (3.4), (3.3), and (5.14), the driving function \mathbf{g} in (3.50) includes the term

$$\mathbf{f}_q = \mathbf{H}^T \mathbf{V}^{-1} (\mathbf{z} - \mathbf{h}(\hat{\mathbf{u}}_q)), \quad (5.16)$$

where

$$\mathbf{H} = \left. \frac{\partial \mathbf{h}(\mathbf{u}_q)}{\partial \mathbf{u}_q} \right|_{\mathbf{u}_q=\hat{\mathbf{u}}_q}. \quad (5.17)$$

Matrix \mathbf{H} maps the 3-space observation forces $(\mathbf{z} - \mathbf{h}(\hat{\mathbf{u}}_q))$ scaled by \mathbf{V}^{-1} , to q-space generalized forces \mathbf{f}_q .⁶ If the data are very noisy, the entries of \mathbf{V} have large values, yielding small generalized forces, hence nominal changes in \mathbf{q} . If the data are accurate, \mathbf{V} will have small entries and the generalized forces will have a significant effect on the model.

⁵For example, if \mathbf{z} consists of observations of time varying positions of model points at material coordinates \mathbf{u}_k on the model's surface, the components of \mathbf{h} are computed using (3.3) evaluated at \mathbf{u}_k . If \mathbf{z} includes velocity estimates, then we also use (3.4).

⁶In particular, for the function \mathbf{h} which is associated with observations of model positions, at material coordinates \mathbf{u}_k , then \mathbf{H} is a matrix whose entries are computed using $\mathbf{L}|_{\mathbf{u}_q=\hat{\mathbf{u}}_q}$ evaluated at \mathbf{u}_k .

MODEL IMPLEMENTATION

We will first present a set of techniques to integrate the motion equations (3.13) and (3.51), and then we describe the initialization of our models. Second, we present various computer vision and computer graphics experiments. In computer vision we present experiments that demonstrate the performance of the framework in deformable model fitting to static 2D and 3D data. In the context of computer graphics, we will present dynamic simulations of our deformable models that interact with each other and their simulated physical environments through collisions, gravity and friction against impenetrable surfaces.

6.1 INTEGRATING THE MOTION EQUATIONS

Our approach partitions complicated nonrigid shapes and motions into rigid-body motions and local deformations away from globally deforming reference shapes. This partitioning improves the stability of simulation algorithms. We achieve real or interactive response¹ by employing standard numerical methods to integrate (3.13). The simplest method is the first-order Euler procedure which updates the state vector over a time step Δt from time t to $t + \Delta t$ according to the formulas

$$\begin{aligned}\ddot{\mathbf{q}}^{(t)} &= \mathbf{M}^{-1}(\mathbf{g}_q^{(t)} + \mathbf{f}_q^{(t)} - \mathbf{K}\mathbf{q}^{(t)} - \mathbf{D}\dot{\mathbf{q}}^{(t)}) \\ \dot{\mathbf{q}}^{(t+\Delta t)} &= \dot{\mathbf{q}}^{(t)} + \Delta t \ddot{\mathbf{q}}^{(t)} \\ \mathbf{q}^{(t+\Delta t)} &= \mathbf{q}^{(t)} + \Delta t \dot{\mathbf{q}}^{(t+\Delta t)}.\end{aligned}\quad (6.1)$$

For the simplified equations of motion (3.51), the Euler procedure becomes

$$\mathbf{q}^{(t+\Delta t)} = \mathbf{q}^{(t)} + \Delta t \mathbf{D}^{-1} (\mathbf{f}_q^{(t)} - \mathbf{K}\mathbf{q}^{(t)}). \quad (6.2)$$

It is worth mentioning that the Euler method is the simplest numerical integration method to implement, but the least accurate and it is often slower

¹See the Introduction for definitions.

than more sophisticated methods. A fixed step fourth order Runge-Kutta, an adaptive Euler or an adaptive Runge-Kutta method will almost always be more accurate and often faster [226], since the integration step is usually significantly bigger.

The following details about our numerical solution are noteworthy. First, we represent the rotation component of the models using quaternions (see Chapter 3). This simplifies the updating of \mathbf{q}_θ and $\dot{\mathbf{q}}_\theta$. Second, the explicit Euler method does not assemble and factorize a finite element stiffness matrix, as is common practice when applying the finite element method. Rather, we compute $\mathbf{K}\mathbf{q}$ very efficiently in an element-by-element fashion for the local deformation coordinates \mathbf{q}_d [278]. Third, for added efficiency without significant loss of accuracy, we may lump masses² to obtain a diagonal \mathbf{M}_{dd} , and we may assume mass-proportional damping, i.e., $\mathbf{D} = \alpha\mathbf{M}$ where α is the damping coefficient [135]. We can in addition parallelize on multiprocessor computers the updating of the model degrees of freedom and the computation of $\mathbf{K}\mathbf{q}$.

6.2 MODEL INITIALIZATION

We will now discuss various ways of initializing our models in computer vision applications. Our model fitting algorithm based on (3.13) and (3.51) does not require user intervention beyond the initialization phase. The initialization step involves two steps. The first is the segmentation of the given data into parts and the second is the initialization of the translation and rotation parameters of the deformable model given some data that correspond to a part.

As will become evident from the following experiments, we require a rough segmentation of the image or range data into corresponding object parts, through the application of an early-vision algorithm. The initialization can be automated using available image segmentation techniques such as the reaction-diffusion segmentation process of Kimia, Tannenbaum, and Zucker [145], the dynamic segmentation technique of Leonardis, Gupta and Bajcsy [158] and the qualitative shape recovery and segmentation technique based on primitive aspects of Dickinson, Pentland and Rosenfeld [71, 72]. In a subsequent section we will address this problem and present some recent algorithms we have developed where the method of Dickinson, Pentland and Rosenfeld [71, 72] was employed.

Given some data that belong to a part of an object, we initialize the translation of the model frame to the center of mass $\bar{\mathbf{r}}$ of the data and its rotation using the matrix of central moments [263]. In case of 3D data we first compute the

²In the lumped mass formulation the total mass of the body is distributed among the grid points. All space integrals in the Lagrange equations of motion reduce to summations over grid points, making computations much faster.

matrix of central moments

$$\mathcal{M} = \frac{1}{m} \sum_{i=1}^N \begin{bmatrix} \mu_{11} & \mu_{12} & \mu_{13} \\ & \mu_{22} & \mu_{23} \\ & & \mu_{33} \end{bmatrix}, \quad (6.3)$$

where

$$\begin{aligned} \mu_{11} &= (d_{i2} - \bar{r}_2)^2 + (d_{i3} - \bar{r}_3)^2, & \mu_{12} &= -(d_{i2} - \bar{r}_2)(d_{i1} - \bar{r}_1), \\ \mu_{13} &= -(d_{i3} - \bar{r}_3)(d_{i1} - \bar{r}_1), & \mu_{22} &= (d_{i1} - \bar{r}_1)^2 + (d_{i3} - \bar{r}_3)^2, \\ \mu_{23} &= -(d_{i3} - \bar{r}_3)(d_{i2} - \bar{r}_2), & \mu_{33} &= (d_{i1} - \bar{r}_1)^2 + (d_{i2} - \bar{r}_2)^2, \end{aligned} \quad (6.4)$$

$\mathbf{d}_i = (d_{i1}, d_{i2}, d_{i3})^T$ and $\bar{\mathbf{r}} = (\bar{r}_1, \bar{r}_2, \bar{r}_3)^T$ are the 3D coordinates and the center of mass of the m datapoints, respectively. We then use the Jacobi method to compute the eigenvectors of \mathcal{M} . These eigenvectors provide an estimate of the model-centered coordinate system. We orient this coordinate system by comparing the corresponding eigenvalues [263], so that its z axis lies along the longest side for elongated objects and along the shortest for flat objects, the underlying assumption being that tapering and bending deformations affect objects along their longest side. We consequently compute by a simple geometric transformation the quaternion corresponding to the rotation of the computed coordinate system with respect to the inertial frame.

The experiments we will present in the following sections and chapters require the correct setting of certain model parameters (e.g., w_0 and w_1). These parameters are set manually on a trial and error basis. Even though we do not include in this book, we have recently developed a new method for the automatic adaptation of the model's elastic parameters [184]. Furthermore, the Euler step we have used in the shape and nonrigid motion experiments amounted to 20–100 algorithm steps required for a model to fit each data frame.

6.3 COMPUTER VISION EXPERIMENTS

We will present examples of fitting deformable hybrid models to static 2D image data and 3D range data from the NRCC 3D image database [237]. The parametric model used is a superquadric ellipsoid and a loaded membrane model was used for the local deformations. The Euler method time step was 4.0×10^{-5} s and we used a unit damping matrix \mathbf{D} .

6.3.1 Image Data

Fig. 6.1 shows the various steps of fitting a deformable model to a 120×128 , 256-intensity monocular image under the assumption of orthographic projection. Fig. 6.1(a) of a 3D object—a pestle. The size of the image is rescaled to fit within the unit square on the x — y plane. Fig. 6.1(b) shows the potential function $P(x, y)$ generated from the image by computing the magnitude of the

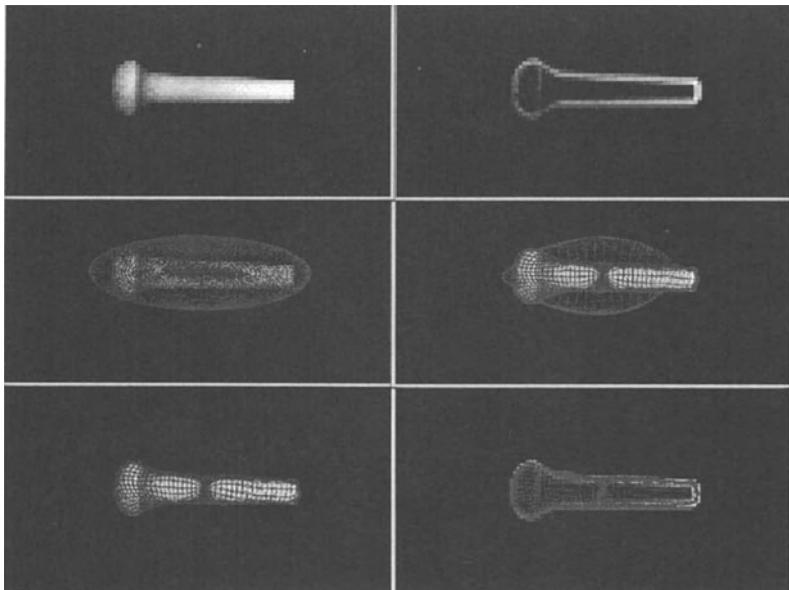


Figure 6.1 Fitting a deformable model to pestle image.

intensity gradient. Fig. 6.1(c) shows the initial state of the deformable model displayed in wireframe projected onto the image. The surface of the model is discretized into 5043 nodes. The initialization consists of specifying the center of the model \mathbf{c} , along with the major and minor axes, $a \cdot a_1$ and $a \cdot a_2$, by picking four points with the mouse. This initializes the translation \mathbf{q}_c and rotation \mathbf{q}_θ of the model. We also fix $\epsilon_1 = \epsilon_2 = 1.0$. In this and subsequent experiments, the local deformation \mathbf{q}_d is initially set to zero. Note that the initialization step produces a very crude first approximation to the pestle.

Fig. 6.1(d) shows an intermediate step in the fitting process which simulates the equations of motion using stiffness parameters $w_0 = 1.0 \times 10^{-6}$ and $w_1 = 4.0 \times 10^{-2}$. Using an orthographic projection Π , nodes of the model whose positions \mathbf{x} in space lie near the image plane ($|x_3| < 0.2$) are subject to a force directed parallel to the image plane:

$$\mathbf{f} = \beta \left(\frac{\partial P}{\partial x}, \frac{\partial P}{\partial y}, 0 \right)^T, \quad (6.5)$$

where the force strength factor is $\beta = 4.0 \times 10^{-6}$.

The forces deform the model and Figs. 6.1(e) and (f) show the final state of the model at equilibrium, superimposed on the image and the potential, respectively.

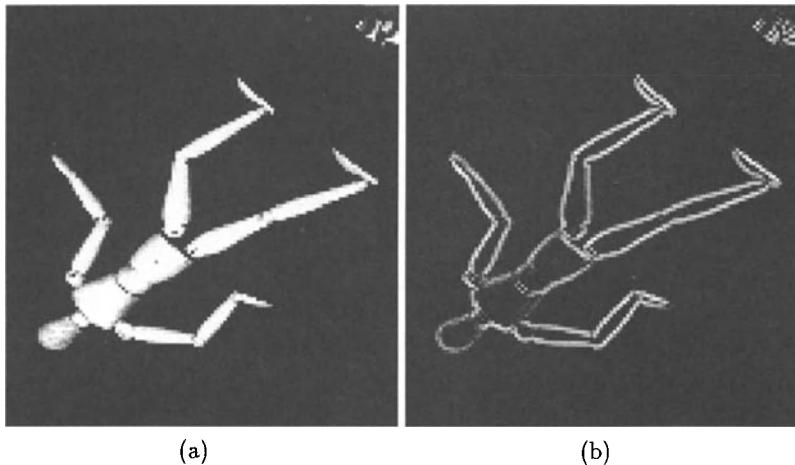


Figure 6.2 (a) Doll image. (b) Doll potential.

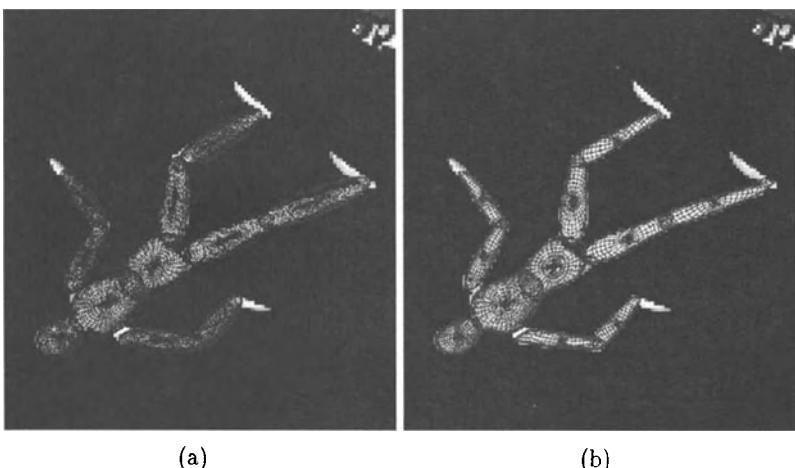


Figure 6.3 (a) Initialization of deformable models to the doll image. (b) Fitting deformable models to the doll image.

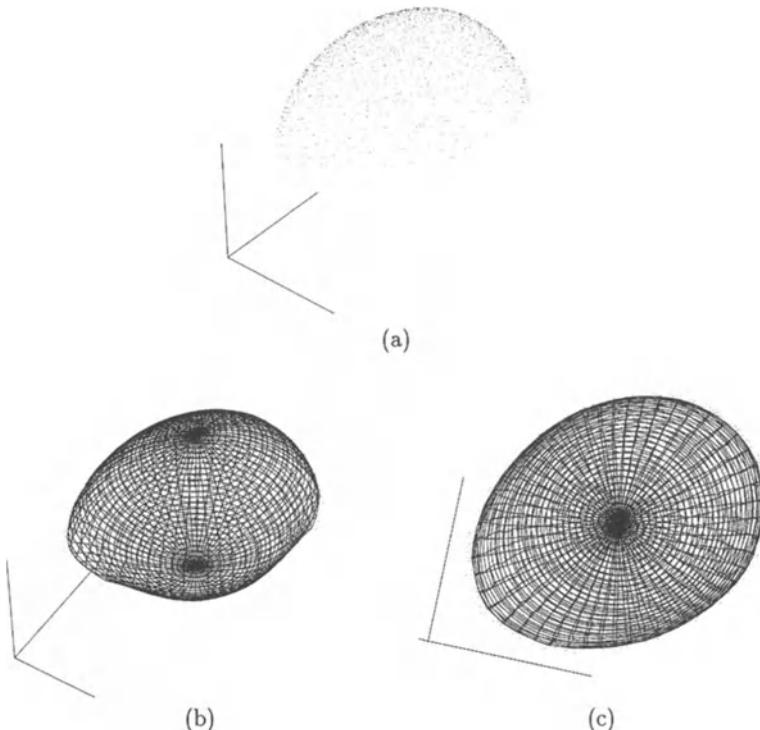


Figure 6.4 Fitting deformable model (b-c) to egg range data (a).

In the second experiment we use the image of a doll shown in Fig. 6.2(a) whose potential is shown in Fig. 6.2(b). The specifics of this experiment are identical to those of the previous one, except that the discrete models consisted of 963 nodes and their stiffness parameters were $w_0 = 0.001$ and $w_1 = 0.1$. Fig. 6.3(a) illustrates the results of the initialization phase for the doll image, which was carried out as described above, showing 11 crude approximations to the major body parts of the figure. The image forces deform the part models into the final shapes shown in Fig. 6.3(b).

6.3.2 Range Data

The following experiments utilize range data from the NRCC 3D image database [237]. We fit the model to 3D data using the force techniques described in a previous chapter. In the following simulations, we have applied forces to the model using the brute-force nearest-node search method, updating the nodes of attachment for each datapoint every 200 algorithm steps.

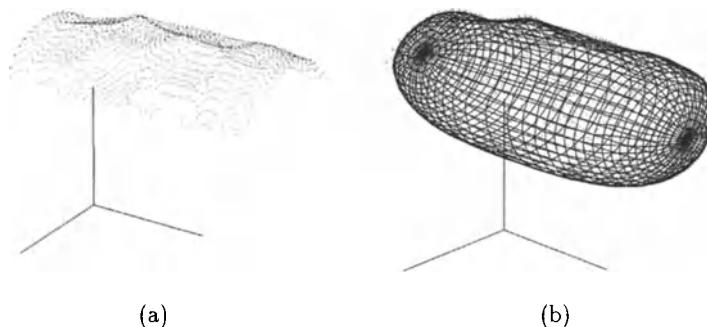


Figure 6.5 Fitting deformable model (b) to mug range data (a).

In the first experiment we fit a deformable model with 2,603 nodes to 3D data sparsely sampled from the upper “hemisphere” of an egg (from range map EGG 1 CAT # 233). Fig. 6.4(a) shows the 499 range datapoints. The stiffness parameters of the model were $w_0 = 1.0 \times 10^{-6}$ and $w_1 = 0.1$. We initialized the model to a sphere located at the center of gravity of the data ($a = 1.2$, $a_1 = a_2 = a_3 = 0.5$, $\epsilon_1 = \epsilon_2 = 1.0$). Fig. 6.4(b) shows the fitted deformable model at equilibrium and Fig. 6.4(c) shows a top view of the fitted model. Evidently, the fit is accurate over the portions of the surface covered by datapoints, but it begins to deteriorate at the boundary of the data near the “equator” because of the influence of the underside of the model which remains too spherical due to the lack of datapoints.

In the second experiment we fit a model with 1,683 nodes to 3D data sparsely sampled from the upper part of a mug with a pitted surface (from range map MUG 1 CAT # 251). Fig. 6.5(a) shows the 651 range datapoints. The stiffness parameters of the model were $w_0 = 0.01$ and $w_1 = 0.1$. We initialized the model to a “tubular” shape ($a = 1.5$, $a_1 = a_2 = 0.3$, $a_3 = 0.8$, $\epsilon_1 = 0.7$, $\epsilon_2 = 1.0$). Fig. 6.5(b) shows the fitted deformable model at equilibrium. The underside of the model is smooth due to the lack of data, but the pitted texture of the top surface has been accurately reconstructed by the local deformational degrees of freedom of the deformable model.

6.4 COMPUTER GRAPHICS EXPERIMENTS

We have created several real-time physics-based animation examples involving models made from deformable model primitives. We employed parallelization of the updating of the model degrees of freedom, after computing the collision and

other external forces, whenever more than one deformable model was involved in a simulation.

Figs. 6.6(a), (b) and (c) illustrate an example where user interaction with a deformable model is allowed. A user can pick the node on the deformable model that is closest to the position of a 3D mouse which can be moved in 3D under user control. A spring force shown as a line is then exerted from the mouse position on the deformable model which causes it to rotate, translate and deform. The elastic parameters of the deformable membrane model were $w_0 = 1.1$ and $w_1 = 5.0$, the Euler step 0.00002, the nodal mass 10^{-7} , the damping coefficient $\nu = 4000.0$ and the global superquadric parameters were $\mathbf{q}_s = (7.0, 0.4, 0.4, 1.0, 2.5, 1.0)^T$.

Fig. 6.7 illustrates a strobed motion sequence of a morphing deformable shell designed to show the possible global deformations of our models starting from upper right to a sphere. An elastic object collides under the influence of gravity with several planes, and in between collisions, changes shape by geometric modification of the relevant global deformation parameters. The elastic object initially has a spherical shape and after going through a series of global shape transformations, comes to its rest position in the shape of a sea shell. Throughout the physical simulation, the elastic parameters of the object were $w_0 = 65.0$ and $w_1 = 85.0$, the Euler step 0.0031, the nodal mass 6.8, the damping coefficient $\nu = 1.2$, the plane elasticity 5000.0 and the friction coefficient 0.3.

Fig. 6.8 shows a strobed motion sequence of a “jello-ball” dropping on a slanted plane and subsequently colliding with the ground. The elastic parameters of the “jello-ball” were $w_0 = 20.0$ and $w_1 = 15.0$, the Euler step 0.031, the nodal mass 8.0, the damping coefficient $\nu = 1.4$, the plane elasticity 670.0 and the friction coefficient 0.3. As a result of collision forces with the plane and the ground which include friction, the “jello-ball” rotates in between collisions and rolls on the ground until it comes to rest.

Fig. 6.9 shows a strobed motion sequence of an elastic “rugby-ball” dropping on a slanted plane, subsequently colliding with a box and the ground and eventually coming to rest. The elastic parameters of the “rugby-ball” were $w_0 = 65.0$ and $w_1 = 85.0$, the Euler step 0.0031, the nodal mass 6.8, the damping coefficient $\nu = 1.2$, the plane elasticity 1050.0 and the friction coefficient 0.15. The “rugby-ball” rotates in between collisions as a result of collision forces with the ground and the box which include friction.

Figs. 6.10(a-i) illustrate collisions of two locally deformable balls with planes and a spring loaded see-saw, while Fig. 6.11 shows the strobed motion sequence. The elastic parameters of each ball were $w_0 = 65.0$ and $w_1 = 85.0$, the Euler step 0.0031, the nodal mass 6.8, the damping coefficient $\nu = 1.2$, the plane elasticity 5000.0 and the friction coefficient 0.4. The see-saw is also a physical object which is rigid and can only undergo rotation around its pivoting axis. We achieve the spring loading of the see-saw by imposing a spring-like “stiffness

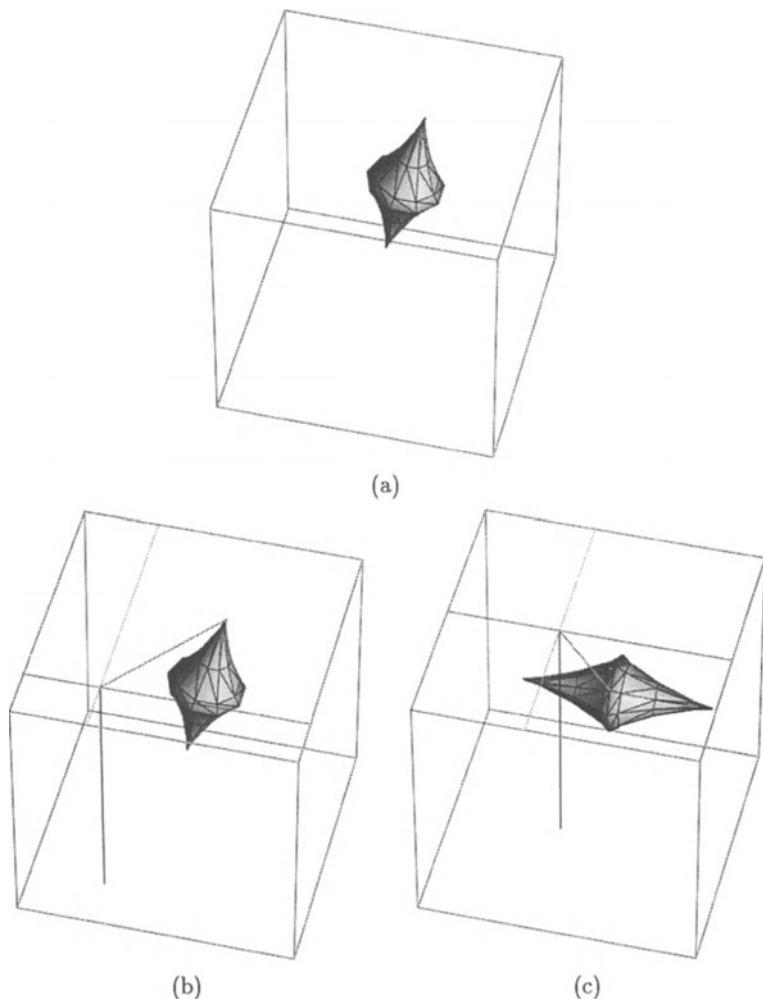


Figure 6.6 Interactive deformable model.

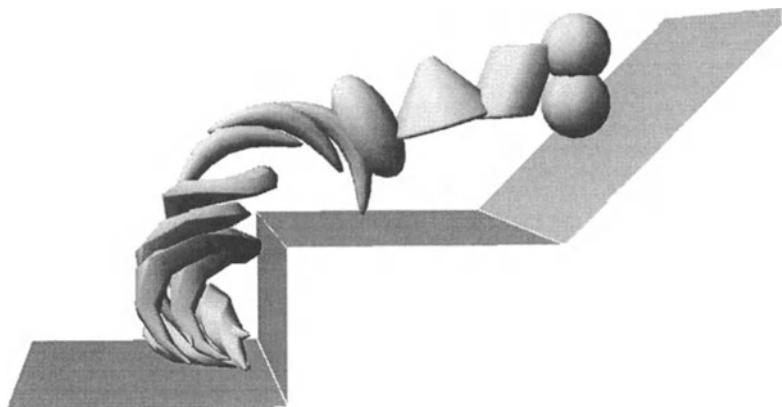


Figure 6.7 Morphing shell.

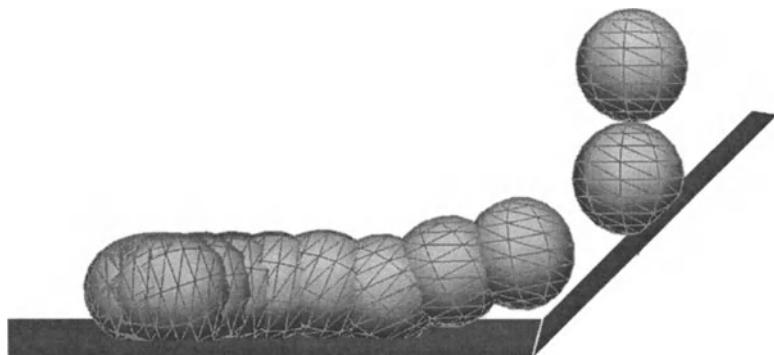


Figure 6.8 “Jello-ball” dropped on a plane.

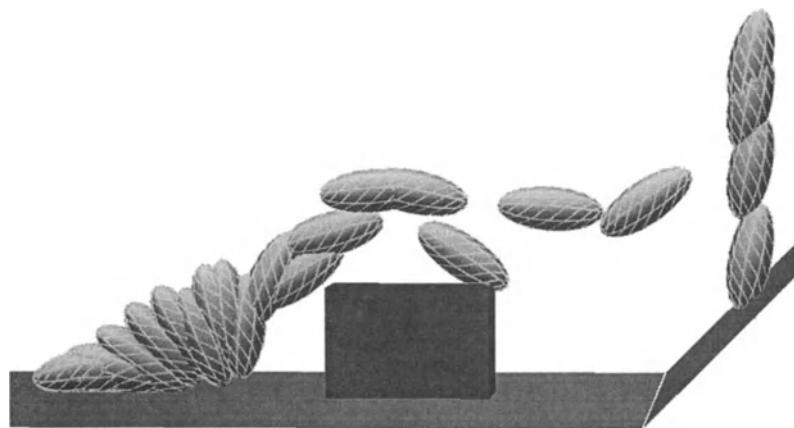


Figure 6.9 Elastic “rugby-ball” dropped on a plane.

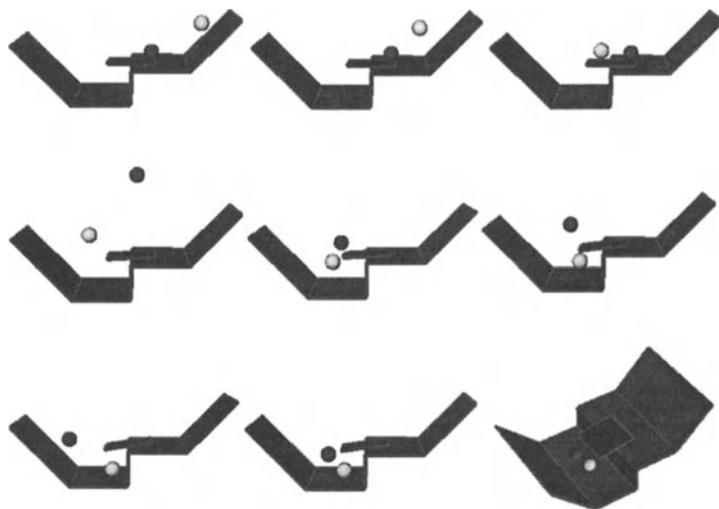


Figure 6.10 Collisions of deformable balls with planes and a spring loaded see-saw.

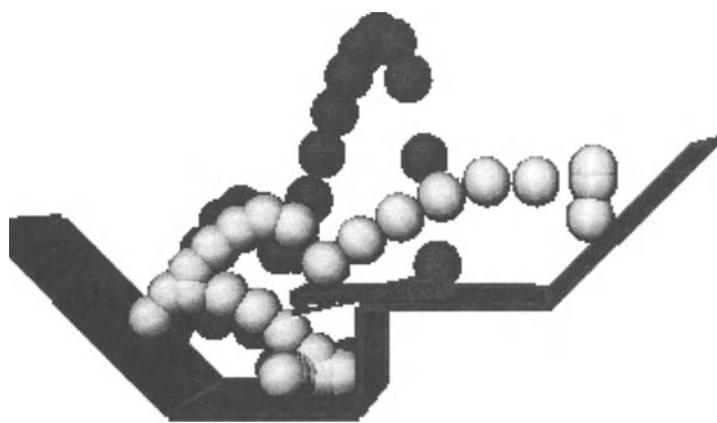


Figure 6.11 Strobbed motion sequence of previous example.

energy” associated with the rotation of the see saw. Its formulation has the same form as that of the stiffness energy associated with global deformations that we presented in Chapter 3.

CONSTRAINED NONRIGID MOTION

We will now extend the equations of motion (3.13) to account for the motions of composite models with interconnected deformable parts which are constrained not to separate.

Several researchers have proposed physics-based constraint methods for working with and controlling animations involving rigid and nonrigid primitives [25, 224, 223, 291, 293]. We describe a computationally efficient method for computing generalized constraint forces between our deformable models which is based on the constraint stabilization technique of Baumgarte [28, 294]. This is a Lagrange multiplier type of method with additional stabilization for improved numerical stability. The resulting constraint satisfaction algorithm is robust and efficient for the general class of holonomic constraints. It requires at every step the inversion of a matrix whose size is proportional to the number of constraints defined. Depending on the structure of the matrix, i.e., how sparse it is, sparse matrix techniques can be used to significantly improve its inversion [255]. The algorithm allows the construction and animation of articulated objects composed of rigid and/or deformable parameterized parts. The constraint algorithm is a generalization of work on physics-based constraints between rigid parts developed by Barzel and Barr [25] and linearly deformable parts developed by Witkin and Welsch [293]. As in [25], it can support the assembly of complex objects satisfying constraints from inappropriately shaped and mispositioned deformable parts that do not initially satisfy the constraints. All the experiments presented run in real-time.

7.1 HOLONOMIC CONSTRAINTS AND LAGRANGE MULTIPLIERS

Shabana [253] describes the well-known Lagrange multiplier method for multi-body systems. We form a composite generalized coordinate vector \mathbf{q} and force vectors \mathbf{g}_q and \mathbf{f}_q for an n -part model by concatenating the \mathbf{q}_i , \mathbf{g}_{qi} , and \mathbf{f}_{qi} associated with each part $i = 1, \dots, \bar{n}$. Similarly, the composite matrices \mathbf{M} ,

\mathbf{D} , and \mathbf{K} for the n -part model are block diagonal matrices with submatrices \mathbf{M}_i , \mathbf{D}_i , and \mathbf{K}_i , respectively, for each part i . The problem is then posed as follows.

Given a set of holonomic constraint equations (i.e., the constraints depend only on \mathbf{q} and not on any of the time derivatives of \mathbf{q})

$$\mathbf{C}(\mathbf{q}, t) = 0, \quad (7.1)$$

where

$$\mathbf{C} = [\mathbf{C}_1^T, \mathbf{C}_2^T, \dots, \mathbf{C}_{\bar{k}}^T]^T \quad (7.2)$$

expresses \bar{k} constraints among the \bar{n} parts of the model, we want to compute the generalized constraint forces \mathbf{f}_{g_c} acting among the parts.

Once we compute \mathbf{f}_{g_c} , we augment the Lagrange equations of motion and arrive to the following system of equations

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{g}_q + \mathbf{f}_q + \mathbf{f}_{g_c}, \quad (7.3)$$

while in case of the simplified model, the equations become

$$\mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{f}_q + \mathbf{f}_{g_c}. \quad (7.4)$$

We can rewrite both of the above equations as a first order system of the form

$$\dot{\mathbf{u}} = \mathbf{F}\mathbf{u} + \mathbf{g}, \quad (7.5)$$

where \mathbf{u} , \mathbf{F} have exactly the same form as in (3.50) and (3.52) for second and first order systems respectively. The vector \mathbf{g} is now given by

$$\mathbf{g} = \begin{bmatrix} \mathbf{M}^{-1}(\mathbf{g}_q + \mathbf{f}_q + \mathbf{f}_{g_c}) \\ 0 \end{bmatrix}, \quad (7.6)$$

for second-order systems and

$$\mathbf{g} = \mathbf{D}^{-1}(\mathbf{f}_q + \mathbf{f}_{g_c}), \quad (7.7)$$

for first-order systems.

In the Lagrange multiplier method the composite equations of motion take the form

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{g}_q + \mathbf{f}_q - \mathbf{C}_q^T \boldsymbol{\lambda}, \quad (7.8)$$

where the generalized constraint forces \mathbf{f}_{g_c} are computed as

$$\mathbf{f}_{g_c} = -\mathbf{C}_q^T \boldsymbol{\lambda}. \quad (7.9)$$

The term \mathbf{C}_q^T is the transpose of the constraint Jacobian matrix and

$$\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1^T, \dots, \boldsymbol{\lambda}_{\bar{n}}^T)^T \quad (7.10)$$

is the vector of Lagrange multipliers that must be determined.

Equation (7.8) comprises fewer equations than unknowns. To obtain the additional equations, we differentiate (7.1) twice with respect to time

$$\ddot{\mathbf{C}}(\mathbf{q}, t) = \mathbf{0}, \quad (7.11)$$

yielding $\mathbf{C}_q \ddot{\mathbf{q}} + \mathbf{C}_{tt} + (\mathbf{C}_q \dot{\mathbf{q}}) \dot{\mathbf{q}} + 2\mathbf{C}_{qt} \dot{\mathbf{q}} = \mathbf{0}$. Rearranging terms we get

$$\gamma = \mathbf{C}_q \ddot{\mathbf{q}} = -\mathbf{C}_{tt} - (\mathbf{C}_q \dot{\mathbf{q}}) \dot{\mathbf{q}} - 2\mathbf{C}_{qt} \dot{\mathbf{q}}. \quad (7.12)$$

Appending this equation to (7.8) and rearranging terms, we arrive at the augmented equations of motion

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_q^T \\ \mathbf{C}_q & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\mathbf{D}\dot{\mathbf{q}} - \mathbf{K}\mathbf{q} + \mathbf{g}_q + \mathbf{f}_q \\ \gamma \end{bmatrix}. \quad (7.13)$$

In principle, these equations may be integrated from initial conditions $\mathbf{q}(0)$ and $\dot{\mathbf{q}}(0)$ satisfying $\mathbf{C}(\mathbf{q}(0), 0) = \mathbf{0}$ and $\dot{\mathbf{C}}(\mathbf{q}(0), 0) = \mathbf{0}$.

There are two practical problems in applying (7.13) to model-based visual estimation and computer animation. First, the constraints must be satisfied initially. In computer vision, due to a lack of full information and errors in the data, the parameter values of the various parts may be initialized such that the parts do not satisfy the constraints (i.e., $\mathbf{C}(\mathbf{q}, 0) \neq \mathbf{0}$). In computer graphics we would also like to give the modeler the freedom to place the various parts of an object in positions that do not satisfy these constraints initially, allowing for the self-assembly of complicated objects. Second, even if the constraints may be satisfied at a given time step of the dynamic estimation process (i.e., $\mathbf{C}(\mathbf{q}, t) = \mathbf{0}$, $\dot{\mathbf{C}}(\mathbf{q}, t) = \mathbf{0}$), they may not be satisfied at the next time step (i.e., $\mathbf{C}(\mathbf{q}, t + \Delta t) \neq \mathbf{0}$) because of numerical integration errors, noise, etc.

7.2 STABILIZED CONSTRAINTS

To remedy these two problems, we apply and adapt a method proposed by Baumgarte which stabilizes the constrained equations through linear feedback control [28, 294]. The method replaces the differential equation (7.12) by other equations which have the same solutions, but which are asymptotically stable in the sense of Ljapunov; for example, the damped second-order differential equations

$$\ddot{\mathbf{C}} + 2\alpha \dot{\mathbf{C}} + \beta^2 \mathbf{C} = \mathbf{0}, \quad (7.14)$$

where α and β are stabilization factors modify (7.13) as follows

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_q^T \\ \mathbf{C}_q & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\mathbf{D}\dot{\mathbf{q}} - \mathbf{K}\mathbf{q} + \mathbf{g}_q + \mathbf{f}_q \\ \gamma - 2\alpha \dot{\mathbf{C}} - \beta^2 \mathbf{C} \end{bmatrix}. \quad (7.15)$$

Fast stabilization means choosing $\beta = \alpha$ to obtain the critically damped solution

$$\mathbf{C}(\mathbf{q}, t) = \mathbf{C}(\mathbf{q}, 0) e^{-\alpha t} \quad (7.16)$$

which, for given α , has the quickest asymptotic decay towards constraint satisfaction $C = 0$. A caveat in applying the constraint stabilization method is that it might introduce additional eigenfrequencies into the dynamical system. Increasing α in an attempt to increase the rate of constraint satisfaction will eventually result in constrained motion equations which are dominated by the stabilizing terms and possibly in numerically stiff equations. However, we did not experience this phenomenon in any of our experiments.

7.3 FAST CONSTRAINT FORCE COMPUTATION FOR MULTIBODY OBJECTS

The method proposed by Baumgarte (7.15) is potentially expensive for our models, since the matrix in (7.15) can be large, depending on the number of finite elements used in the model discretization and the number of the parts composing an object. Several researchers have proposed computationally faster techniques than the standard Lagrange multiplier method by developing methods that integrate only a minimum number of coordinates equal to the object's degrees of freedom [2, 290, 48, 255] or by developing more robust numerical analysis techniques [221, 101] for the solution of (7.13). We have devised a general solver for the unknown constraint forces f_{g_c} and we demonstrate it in the case of point-to-point constraints between deformable primitives. Our method is also applicable for the simpler case of rigid objects and for any type of holonomic constraints. This method requires the solution of linear systems of size proportional to the number of constraints¹, which is usually small. Our technique is similar to the coordinate reduction techniques in the sense of solving a reduced system of equations for the unknown constraint forces. In our technique we do not reduce the number of generalized coordinates, since the method would become difficult to implement in case of complex systems. Finally, numerical stabilization is achieved through Baumgarte's stabilization technique. The method is also similar to the dynamic constraint technique of [25]; however, it is based on a Lagrangian as opposed to a Newtonian formulation, it is more general since it is also suitable for nonrigid objects, and finally, it demonstrates how to automatically construct the relevant linear system. We derive the method for second-order dynamic systems (3.13).

In the remainder of this section, we first give a simple example of a multibody object with two constraints. We then present the general algorithm for an articulated object with an arbitrary number of parts and point-to-point constraints. It is worthwhile to mention that this technique can be used for any type of holonomic constraint.

¹Its size is $3\bar{k} \times 3\bar{k}$, where \bar{k} is the total number of constraints.

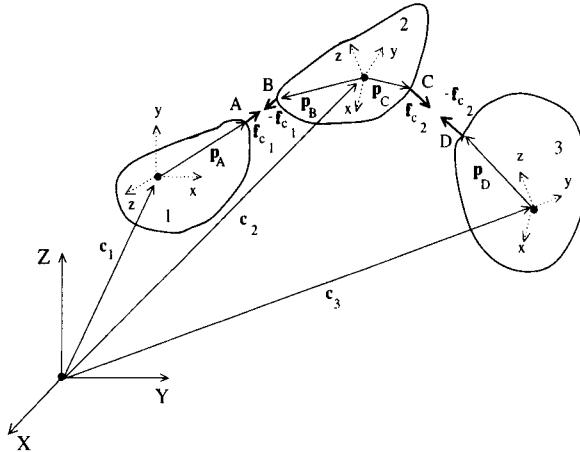


Figure 7.1 Two point-to-point constraints.

7.3.1 Multibody Object with Two Constraints

Fig. 7.1 illustrates three parts, 1, 2 and 3, of an articulated object. We constrain points A and B and points C and D to be in contact, and must compute the necessary constraint forces $\mathbf{f}_{c_1(t)}$ at point A , $-\mathbf{f}_{c_1(t)}$ at point B , $\mathbf{f}_{c_2(t)}$ at point C , and $-\mathbf{f}_{c_2(t)}$ at point D . From (3.13), the motion equations of the parts are

$$\begin{aligned}\ddot{\mathbf{q}}_1 &= \mathbf{M}_1^{-1}(\mathbf{f}_{g_{c_A}} + \mathbf{g}_{q_1} + \mathbf{f}_{q_1} - \mathbf{K}_1\mathbf{q}_1 - \mathbf{D}_1\dot{\mathbf{q}}_1) \\ &= \mathbf{M}_1^{-1}(\mathbf{f}_{g_{c_A}} + \mathbf{V}_1),\end{aligned}\quad (7.17)$$

$$\begin{aligned}\ddot{\mathbf{q}}_2 &= \mathbf{M}_2^{-1}(\mathbf{f}_{g_{c_B}} + \mathbf{f}_{g_{c_C}} + \mathbf{g}_{q_2} + \mathbf{f}_{q_2} - \mathbf{K}_2\mathbf{q}_2 - \mathbf{D}_2\dot{\mathbf{q}}_2) \\ &= \mathbf{M}_2^{-1}(\mathbf{f}_{g_{c_B}} + \mathbf{f}_{g_{c_C}} + \mathbf{V}_2),\end{aligned}\quad (7.18)$$

$$\begin{aligned}\ddot{\mathbf{q}}_3 &= \mathbf{M}_3^{-1}(\mathbf{f}_{g_{c_D}} + \mathbf{g}_{q_3} + \mathbf{f}_{q_3} - \mathbf{K}_3\mathbf{q}_3 - \mathbf{D}_3\dot{\mathbf{q}}_3) \\ &= \mathbf{M}_3^{-1}(\mathbf{f}_{g_{c_D}} + \mathbf{V}_3),\end{aligned}\quad (7.19)$$

where the generalized constraint forces at points A , B , C and D are, respectively,

$$\begin{aligned}\mathbf{f}_{g_{c_A}} &= \mathbf{L}_A^T \mathbf{f}_{c_1}, & \mathbf{f}_{g_{c_B}} &= -\mathbf{L}_B^T \mathbf{f}_{c_1}, \\ \mathbf{f}_{g_{c_C}} &= \mathbf{L}_C^T \mathbf{f}_{c_2}, & \mathbf{f}_{g_{c_D}} &= -\mathbf{L}_D^T \mathbf{f}_{c_2},\end{aligned}\quad (7.20)$$

and $\mathbf{L}_A, \mathbf{L}_B, \mathbf{L}_C, \mathbf{L}_D$, are computed using (3.4).

From (2.2) and (3.4), the two constraint equations and their time derivatives are

$$\begin{aligned}\mathbf{C}_1 &= \mathbf{x}_A - \mathbf{x}_B = (\mathbf{c}_1 + \mathbf{R}_1 \mathbf{p}_A) - (\mathbf{c}_2 + \mathbf{R}_2 \mathbf{p}_B) \\ \dot{\mathbf{C}}_1 &= \mathbf{L}_A \dot{\mathbf{q}}_1 - \mathbf{L}_B \dot{\mathbf{q}}_2 \\ \ddot{\mathbf{C}}_1 &= \mathbf{L}_A \ddot{\mathbf{q}}_1 + \dot{\mathbf{L}}_A \dot{\mathbf{q}}_1 - \mathbf{L}_B \ddot{\mathbf{q}}_2 - \dot{\mathbf{L}}_B \dot{\mathbf{q}}_2,\end{aligned}\quad (7.21)$$

and

$$\begin{aligned}\mathbf{C}_2 &= \mathbf{x}_C - \mathbf{x}_D = (\mathbf{c}_2 + \mathbf{R}_2 \mathbf{p}_C) - (\mathbf{c}_3 + \mathbf{R}_3 \mathbf{p}_D) \\ \dot{\mathbf{C}}_2 &= \mathbf{L}_C \dot{\mathbf{q}}_2 - \mathbf{L}_D \dot{\mathbf{q}}_3 \\ \ddot{\mathbf{C}}_2 &= \mathbf{L}_C \ddot{\mathbf{q}}_2 + \dot{\mathbf{L}}_C \dot{\mathbf{q}}_2 - \mathbf{L}_D \ddot{\mathbf{q}}_3 - \dot{\mathbf{L}}_D \dot{\mathbf{q}}_3.\end{aligned}\quad (7.22)$$

Replacing these expressions into Baumgarte's equation (7.14) (with $\alpha = \beta$), we obtain the linear system of equations

$$\begin{aligned}(\mathbf{L}_A \mathbf{M}_1^{-1} \mathbf{L}_A^T + \mathbf{L}_B \mathbf{M}_2^{-1} \mathbf{L}_B^T) \mathbf{f}_{c_1} - (\mathbf{L}_B \mathbf{M}_2^{-1} \mathbf{L}_C^T) \mathbf{f}_{c_2} + \mathbf{r}_{11} &= \mathbf{N}_1 \mathbf{f}_c + \mathbf{r}_1 = \mathbf{0} \\ -(\mathbf{L}_C \mathbf{M}_2^{-1} \mathbf{L}_B^T) \mathbf{f}_{c_1} + (\mathbf{L}_C \mathbf{M}_2^{-1} \mathbf{L}_C^T + \mathbf{L}_D \mathbf{M}_3^{-1} \mathbf{L}_D^T) \mathbf{f}_{c_2} + \mathbf{r}_{22} &= \mathbf{N}_2 \mathbf{f}_c + \mathbf{r}_2 = \mathbf{0},\end{aligned}\quad (7.23)$$

in the unknown constraint forces $\mathbf{f}_c = (\mathbf{f}_{c_1}^T, \mathbf{f}_{c_2}^T)^T$. The 3×1 vectors \mathbf{r}_{11} and \mathbf{r}_{22} are

$$\mathbf{r}_{11} = \dot{\mathbf{L}}_A \dot{\mathbf{q}}_1 - \dot{\mathbf{L}}_B \dot{\mathbf{q}}_2 + 2a \dot{\mathbf{C}}_1 + b^2 \mathbf{C}_1 + \mathbf{L}_A \mathbf{M}_1^{-1} \mathbf{V}_1 - \mathbf{L}_B \mathbf{M}_2^{-1} \mathbf{V}_2, \quad (7.24)$$

$$\mathbf{r}_{22} = \dot{\mathbf{L}}_C \dot{\mathbf{q}}_2 - \dot{\mathbf{L}}_D \dot{\mathbf{q}}_3 + 2a \dot{\mathbf{C}}_2 + b^2 \mathbf{C}_2 + \mathbf{L}_C \mathbf{M}_2^{-1} \mathbf{V}_2 - \mathbf{L}_D \mathbf{M}_3^{-1} \mathbf{V}_3, \quad (7.25)$$

and the matrices and vectors on the right hand side are

$$\mathbf{N}_1 = \begin{pmatrix} \mathbf{L}_A \mathbf{M}_1^{-1} \mathbf{L}_A^T + \mathbf{L}_B \mathbf{M}_2^{-1} \mathbf{L}_B^T & -(\mathbf{L}_B \mathbf{M}_2^{-1} \mathbf{L}_C^T) \\ 0 & 0 \end{pmatrix}, \quad (7.26)$$

$$\mathbf{N}_2 = \begin{pmatrix} 0 & 0 \\ -(\mathbf{L}_C \mathbf{M}_2^{-1} \mathbf{L}_B^T) & \mathbf{L}_C \mathbf{M}_2^{-1} \mathbf{L}_C^T + \mathbf{L}_D \mathbf{M}_3^{-1} \mathbf{L}_D^T \end{pmatrix}, \quad (7.27)$$

$$\mathbf{r}_1 = \begin{pmatrix} \mathbf{r}_{11} \\ 0 \end{pmatrix}, \quad (7.28)$$

$$\mathbf{r}_2 = \begin{pmatrix} 0 \\ \mathbf{r}_{22} \end{pmatrix}. \quad (7.29)$$

Combining (7.23), we arrive at the linear system

$$\mathbf{N} \mathbf{f}_c + \mathbf{r} = (\mathbf{N}_1 + \mathbf{N}_2) \mathbf{f}_c + (\mathbf{r}_1 + \mathbf{r}_2) = \mathbf{0}, \quad (7.30)$$

which we solve using a standard LU matrix factorization algorithm.

7.3.2 Multibody Object with Multiple Constraints

For multiple point-to-point constraints the computation must take into account all of the constraint forces acting on the various parts of the model. This requires the solution of a system of constraint equations whose size is $3\bar{k} \times 3\bar{k}$, where \bar{k} is the total number of constraints.

Suppose we specify \bar{k} constraints among \bar{n} model parts. Let \mathbf{f}_{c_i} be the constraint force for constraint i . We assemble the multibody object's constraint force vector $\mathbf{f}_c = (\mathbf{f}_{c_1}^T, \mathbf{f}_{c_2}^T, \dots, \mathbf{f}_{c_{\bar{k}}}^T)^T$ and express the equation for constraint forces \mathbf{f}_{c_i} as in (7.23):

$$\mathbf{N}_i \mathbf{f}_c + \mathbf{r}_i = \mathbf{0}, \quad (7.31)$$

where

$$\mathbf{r}_i = (0^T, \dots, \mathbf{r}_{ii}^T, \dots, 0^T)^T \quad (7.32)$$

and \mathbf{N}_i is a $3\bar{k} \times 3\bar{k}$ matrix. Assembling the \bar{k} systems, we arrive at a composite system in the form of

$$\mathbf{N} \mathbf{f}_c + \mathbf{r} = \mathbf{0}, \quad (7.33)$$

where

$$\mathbf{N} = \sum_{i=1}^{\bar{k}} \mathbf{N}_i \quad (7.34)$$

and

$$\mathbf{r} = \sum_{i=1}^{\bar{k}} \mathbf{r}_i. \quad (7.35)$$

The pattern of nonzero entries in matrix \mathbf{N} depends on the connectivity of the parts. It is worthwhile to mention that for special classes of multipart objects (e.g., objects with tree-like structure) the number of zero entries in \mathbf{N} is significant. In such a case it is possible to use sparse matrix techniques (see also [21] for an implementation for the simple case of a loop-free system of rigid bodies) instead of the LU factorization method to solve the resulting system, and therefore significantly speed-up its solution.

Based on the form of equations (7.31), we can devise the following algorithm to automatically compute the entries of \mathbf{N} .

Algorithm to Compute \mathbf{N}

1. Divide \mathbf{N} into submatrices as follows:

$$\mathbf{N} = \begin{bmatrix} \mathbf{N}_{11} & & & & \mathbf{N}_{1\bar{k}} \\ & \ddots & & & \\ & & \mathbf{N}_{ij} & & \\ & & & \ddots & \\ \text{symmetric} & & & & \mathbf{N}_{\bar{k}\bar{k}} \end{bmatrix}, \quad (7.36)$$

where \mathbf{N}_{ij} are 3×3 submatrices and \bar{k} is the number of constraints. Set $\mathbf{N} = \mathbf{0}$. Iterate through all the constraints and for each constraint mark as *one*, one of the two associated model parts and as *two*, the other part (the labeling is arbitrary).

2. For each constraint l , $1 \leq l \leq \bar{k}$, where constraint l requires that points i_A and j_B of the two associated deformable parts i and j , respectively, should always be in contact do the following two steps:

- (a) Compute entry \mathbf{N}_{ll} based on the following formula:

$$\mathbf{N}_{ll} = \mathbf{L}_{i_A} \mathbf{M}_i^{-1} \mathbf{L}_{i_A}^T + \mathbf{L}_{j_B} \mathbf{M}_j^{-1} \mathbf{L}_{j_B}^T. \quad (7.37)$$

- (b) Compute off-diagonal terms \mathbf{N}_{lm} , $l < m \leq \bar{k}$, as follows:

Iterate through each of the constraints m .

- If constraint m involves a point i_J which belongs to part i , then
 - If in constraint m , part i is marked as *one*:

$$\mathbf{N}_{lm} = \mathbf{L}_{i_A} \mathbf{M}_i^{-1} \mathbf{L}_{i_J}^T. \quad (7.38)$$

- else (in constraint m , part i is marked as *two*):

$$\mathbf{N}_{lm} = -\mathbf{L}_{i_A} \mathbf{M}_i^{-1} \mathbf{L}_{i_J}^T. \quad (7.39)$$

- else (constraint m involves a point j_J which belongs to part j)
 - If in constraint m , part j is marked as *one*:

$$\mathbf{N}_{lm} = -\mathbf{L}_{j_B} \mathbf{M}_j^{-1} \mathbf{L}_{j_J}^T. \quad (7.40)$$

- else (in constraint m , part j is marked as *two*):

$$\mathbf{N}_{lm} = \mathbf{L}_{j_B} \mathbf{M}_j^{-1} \mathbf{L}_{j_J}^T. \quad (7.41)$$

In applying the fast point-to-point constraint algorithm at each time step we assemble \mathbf{N} and \mathbf{r} and compute the constraint forces \mathbf{f}_c by solving (7.33) using *LU* factorization. We then augment the equations of motion of each part with the appropriate constraint forces and we integrate using the Euler method, or a more advanced integration method.

7.4 EXPERIMENTS WITH CONSTRAINTS

We present various computer graphics experiments. Computer vision experiments will be demonstrated in the next chapter after the introduction of recursive estimation. The simulations demonstrate complex models that interact with each other and their simulated physical environments through constraints, collisions, gravity, and friction against impenetrable surfaces. All the examples run in real time on an Silicon Graphics Indigo 2 workstation. The parameterized primitives used in the construction of the deformable models are superquadrics and a membrane model was used for the local deformations.

Fig. 7.2 shows several frames from an animation of two deformable model “balloons” suspended in gravity by inextensible strings attached to a ceiling using

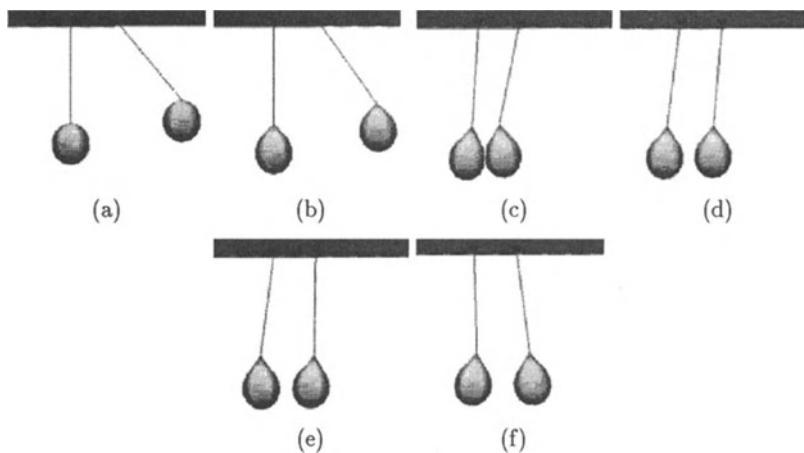


Figure 7.2 Two balloon pendulums.

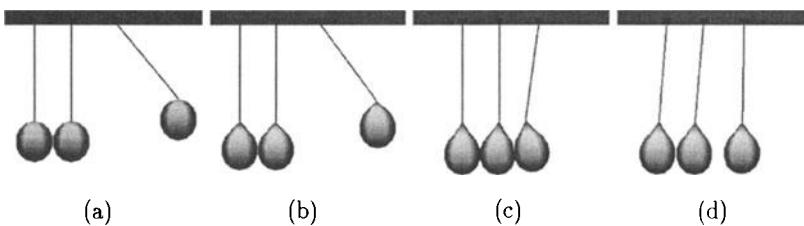


Figure 7.3 Balloon multi-pendulum. Initial state (a). Swinging and colliding (b-d).

point-to-point constraints. Point-to-point constraints also connect the balloons to the strings. Fig. 7.2(a) shows the initial configuration with the left balloon pulled to the side. Gravity is activated in Fig. 7.2(b). The balloons deform under their own weight. The left balloon swings to the right, colliding inelastically with its neighbor in Fig. 7.2(c), thereby transferring some of its kinetic energy. In Figs. 7.2(d-f) the balloons collide repeatedly until all the kinetic energy is dissipated. The collisions are implemented using the collision algorithm described above. Fig. 7.3 shows a similar scenario involving three balloons. The middle balloon cushions the left balloon from the blow of the collision by deforming. The left balloon therefore swings a shorter distance than the left balloon did in the 2-balloon animation. The “balloon” elastic parameters were $w_0 = 60.0$ and $w_1 = 90.0$, the Euler step was 0.003, the nodal mass 10.0 and the damping coefficient $\nu = 1.6$. The “string” elastic parameters were $w_0 = 60.0$ and $w_1 = 100.0$, the Euler step was 0.0001, the nodal mass 0.8 and the damping coefficient $\nu = 2.6$.

Fig. 7.4 shows the automatic construction of a minimalist dragonfly from its constituent deformable model parts. Fig. 7.4(a) shows the disjoint parts in their

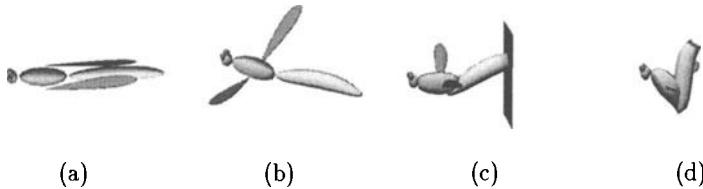


Figure 7.4 Dragonfly. Self-assembly (a). Flight (b). Swatting (c). Swatted fly (d).

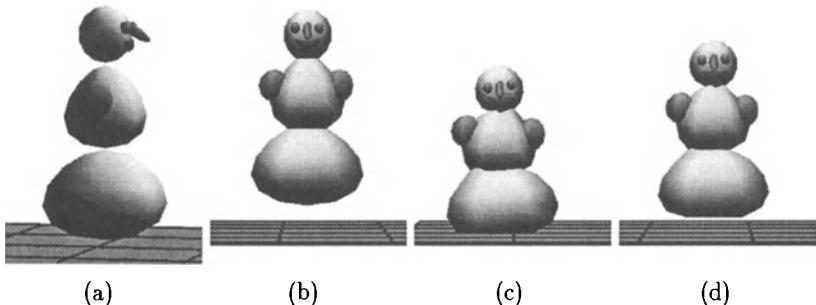


Figure 7.5 Yellow snowman. Disjoint parts (a). Self-assembly (b). Hopping (c-d).

initial configurations. After activating our constraint algorithm, the model self-assembles to form the articulated dragonfly. Four point-to-point constraints hold the deformable body parts together. The dragonfly “works” inasmuch as forces can induce opening and flapping of the wings, as is illustrated in Fig. 7.4(b). In Fig. 7.4(c) we swat the dragonfly in the rear with an impenetrable plane. The body parts deform in response to the blow, but the point-to-point constraints continue to hold them together. The mangled dragonfly is shown in Fig. 7.4(d). The Euler step was 0.0031, the nodal mass 2.0, and the elastic parameters were $w_0 = 0.1$ and $w_1 = 0.5$

Fig. 7.5 illustrates the self-assembly and subsequent animation of a snowman. Fig. 7.5(a) shows a view of the disjoint deformable model body parts of a snowman. The snowman self-assembles when the constraints are activated (Fig. 7.5(b)). There are 12 point-to-point constraints holding the snowball parts together. Gravity is activated and the snowman drops to the impenetrable floor and locomotes along a prespecified path by repeatedly bouncing in a controlled fashion (Fig. 7.5(c-d)). The elastic parameters of each part were $w_0 = 0.1$ and $w_1 = 0.5$, the Euler step 0.001, and the nodal mass 1.0.

SHAPE AND NONRIGID MOTION ESTIMATION

In our previous presentation of the shape and nonrigid motion estimation algorithm we did not incorporate formally in the estimation process the noise in the data.

In this chapter we present a recursive technique for estimating shape and nonrigid motion from incomplete, noisy observations available sequentially over time. The estimator is a nonlinear Kalman filter that employs the constraint equations of motion as a system model. The technique transforms the discrepancy between current observations and points on the model into forces, using the algorithms described in a previous chapter. The Kalman filter includes an error covariance matrix which transforms the forces in accordance with the system dynamics and the prior observation history. The transformed forces induce changes in the translational, rotational, and deformational state variables of the system model to reduce the inconsistencies. Thus the system model synthesizes nonstationary shape and motion estimates in response to the visual data.

Kalman filtering techniques described by Gelb [97] have been applied in the vision literature to the estimation of surface depth [172, 112], dynamic features (e.g., Deriche and Faugeras [68]), and rigid motion parameters [77, 38, 39] of objects from image sequences. These Kalman filters have been discrete filters with rather simple system models, usually constant velocity assumptions (e.g., Pentland and Horowitz [217]) or constant error covariance matrices [217]. These sorts of simplifications can severely limit the ability of an estimator to recover shape and motion parameters accurately from real-world data, especially when confronted with articulated or fully nonrigid motion.

By contrast, the continuous Kalman filter that we present can incorporate the rather sophisticated Lagrange equations of 3D nonrigid motion as system models (simpler dynamic models for the estimation of rigid body motion parameters [76, 77] or 2D nonrigid motion parameters developed [268, 36] are also available). To gain efficiency with minimal loss of accuracy, we design an efficient large-scale Kalman estimator through state decoupling. Our work establishes a direct connection with existing dynamic vision models derived from physical principles. We show, for example, that the force-based tracking scheme pro-

posed in [136, 275] and applied to deformable superquadrics by Terzopoulos and Metaxas [278] amounts to a degenerate “Kalman filter” with a constant, unit error covariance matrix.

8.1 RECURSIVE ESTIMATION

The force based estimation scheme described in Chapter 5 employs data forces (5.16) which take into account the current observations only. This section proposes a more sophisticated estimator which transforms the generalized forces through an error covariance matrix before applying them to the model. The covariance matrix takes into account the modeling uncertainty, along with the history of prior observations and their uncertainties.

Our estimator is a continuous extended Kalman filter. The basic idea behind Kalman filtering is to perform optimal least squares estimation recursively as new measurements arrive, making use of nonstationary models known as system models [97]. We exploit our dynamic models of nonrigid objects within the Kalman framework by employing their differential equations of motion as system models.

To do this, we assume that vectors $\mathbf{w}(t)$ and $\mathbf{v}(t)$ represent uncorrelated modeling and measurement errors, respectively, as zero mean white noise processes with known covariances, i.e., $\mathbf{w}(t) \sim N(\mathbf{0}, \mathbf{Q}(t))$ and $\mathbf{v}(t) \sim N(\mathbf{0}, \mathbf{V}(t))$.¹ In view of (3.49) and (5.14) the nonlinear Kalman filter equations for our dynamic model take the form

$$\begin{aligned}\dot{\mathbf{u}}_q &= \mathbf{F}\mathbf{u}_q + \mathbf{g}_1 + \mathbf{w} \\ \mathbf{z} &= \mathbf{h}(\mathbf{u}_q) + \mathbf{v},\end{aligned}\tag{8.1}$$

where

$$\mathbf{g}_1 = \begin{bmatrix} \mathbf{M}^{-1}(\mathbf{g}_q + \mathbf{f}_{g_c}) \\ \mathbf{0} \end{bmatrix}\tag{8.2}$$

($\mathbf{g}_1 = \mathbf{D}^{-1}\mathbf{f}_{g_c}$ for the first-order model). The vector \mathbf{g}_1 which includes generalized inertial and constraint forces is treated as a deterministic disturbance in the system model. The state estimation equation for uncorrelated system and measurement noises (i.e., $E[\mathbf{w}(t)\mathbf{v}^T(\tau)] = \mathbf{0}$) is

$$\dot{\hat{\mathbf{u}}}_q = \mathbf{F}\hat{\mathbf{u}}_q + \mathbf{g}_1 + \mathbf{P}\mathbf{H}^T\mathbf{V}^{-1}(\mathbf{z} - \mathbf{h}(\hat{\mathbf{u}}_q)),\tag{8.3}$$

where \mathbf{H} is computed using (5.17). The expression $\mathbf{G}(t) = \mathbf{P}\mathbf{H}^T\mathbf{V}^{-1}$ is known as the Kalman gain matrix. The symmetric error covariance matrix $\mathbf{P}(t)$ is the

¹Kalman filtering is optimal for linear system and measurement models, assuming the associated noise processes are Gaussian [97]. The Gaussian noise assumption is unrealistic in many applications. Often in practice, however, all we can economically measure about the characteristics of a noise process is its autocorrelation function; hence, a Gaussian model is the most convenient choice.

solution of the matrix Riccati equation

$$\dot{\mathbf{P}} = \mathbf{F}\mathbf{P} + \mathbf{P}\mathbf{F}^T + \mathbf{Q} - \mathbf{P}\mathbf{H}^T\mathbf{V}^{-1}\mathbf{H}\mathbf{P}. \quad (8.4)$$

Note that the term $\mathbf{P}\mathbf{H}^T\mathbf{V}^{-1}(\mathbf{z} - \mathbf{h}(\hat{\mathbf{u}}_q))$ in (8.3) is a generalized force. It results from the instantaneous disparity between the measurements \mathbf{z} and the estimated state of the model $\hat{\mathbf{u}}_q$. For unit covariance matrix $\mathbf{P}(t) = \mathbf{I}$, this term reduces to the generalized data forces (5.16) which are applied to dynamic models in force-based estimation [278, 275]. We interpret the Kalman filter physically: The system model continually synthesizes nonrigid motions in response to generalized forces that arise from inconsistencies between its state variables and the incoming observations. The observation forces account formally for instantaneous uncertainties in the data.

The improvement offered by the Kalman filter over force-based estimation can be explained intuitively by realizing that the covariance matrix $\mathbf{P}(t)$ comprises a time-varying measure of the uncertainty in the estimate $\hat{\mathbf{u}}_q$ [97]. The measure depends on current and prior observations, system dynamics, and modeling errors. Consequently, the Kalman gain matrix \mathbf{G} becomes “proportional” to the uncertainty in the estimate and “inversely proportional” to the measurement noise. If, on the one hand, there is significant measurement noise and the state estimate errors are small, the term in parentheses in (8.3) is due mainly to noise and only small changes in the state estimates should be made. On the other hand, small measurement noise and large uncertainty in the state estimates suggest that the same term contains significant information about errors in the estimates. Therefore, the difference between the actual and the predicted measurement will be used as a basis for making strong corrections to the estimates.

8.2 KALMAN FILTER IMPLEMENTATION

The continuous Kalman estimation equations (8.3) and (8.4) can be integrated using standard numerical techniques; the simplest being Euler’s method.

The cost of computing (8.4) can be large because the size of the error covariance matrix is proportional to the size of \mathbf{q}_d , which depends on the number of finite elements used to discretize the model. We take a common approach to implementing practical large-scale Kalman filters: suboptimal filter design [97]. There are several ways to simplify the Kalman filter equations, such as decoupling state variables, deleting state variables, simplifying filter gains, etc. Suitable simplifications are often based on knowledge about the particular system model used in the Kalman filter equations. They can lead to a significantly reduced computer burden with little reduction in estimation accuracy.

We simplify the Kalman filter equations by decoupling state variables. The decoupling is dictated by the structure of the state vector \mathbf{u} in (3.49) which is comprised of the translation ($\mathbf{q}_c, \dot{\mathbf{q}}_c$), rotation ($\mathbf{q}_\theta, \dot{\mathbf{q}}_\theta$), global deformation ($\mathbf{q}_s, \dot{\mathbf{q}}_s$), and local deformation ($\mathbf{q}_d, \dot{\mathbf{q}}_d$) state variables for the various part models. To decouple (8.4), we ignore the off-diagonal block submatrices of \mathbf{P} which specify covariances among the different sets of state variables. Each set can then be updated independently. Note that the submatrix of \mathbf{P} associated with the local deformations is structured like the stiffness matrix \mathbf{K} [278]; i.e., it is a sparse matrix which may be updated on a per-element basis. Each symmetric elemental error covariance submatrix is full, since the elemental stiffness submatrix is also a full symmetric matrix, i.e., its nodal displacements are interdependent.

For additional savings, we may assume independent modeling and measurement errors, which lead to covariance matrices \mathbf{Q} and \mathbf{V} that are scalar multiples of the identity matrix. Note that although the entries of the measurement covariance matrix \mathbf{V} are often known, it is not easy to determine appropriate values for the system covariance matrix \mathbf{Q} [97]. The latter are therefore considered as filter tuning parameters. If the entries of \mathbf{Q} are inappropriate for the particular application, the filter may diverge or converge to the wrong value [39]. This is due to the nonlinear measurement equations (8.1). The entries of \mathbf{Q} should therefore be tuned so that the filter converges to the correct solution.

8.3 RECURSIVE ESTIMATION OF SHAPE AND NONRIGID MOTION

We have carried out various shape and nonrigid motion estimation experiments utilizing 3D human motion data. The human motion data were collected using WATSMART, a commercial non-contact, 3D motion digitizing and analysis system. Using multiple optoelectric measurement cameras, WATSMART can track as many as 64 infrared light emitting diode markers attached to various body parts of a moving subject. WATSMART produces 3D coordinates of the markers at sampling rates of 19 Hz to 400 Hz. At least two cameras must see a marker before its three dimensional coordinates can be calculated using a direct linear transformation technique. Our data were collected using 4 cameras and 32 markers at a 50 Hz sampling rate.

In the experiments, we couple the models to the data points, indicated by dark dots in the forthcoming figures, by searching for the node on the model that is nearest to each datapoint. In the experiments our algorithms execute at rates of 0.5 seconds per frame of input data on a single processor of a Silicon Graphics Indigo 2 workstation, including real-time 3D graphics. The estimator advances to the next frame of data when the change in each of the estimated parameters falls below 10^{-4} . The Euler method time step was 4.0×10^{-5} s and we used a unit damping matrix \mathbf{D} .

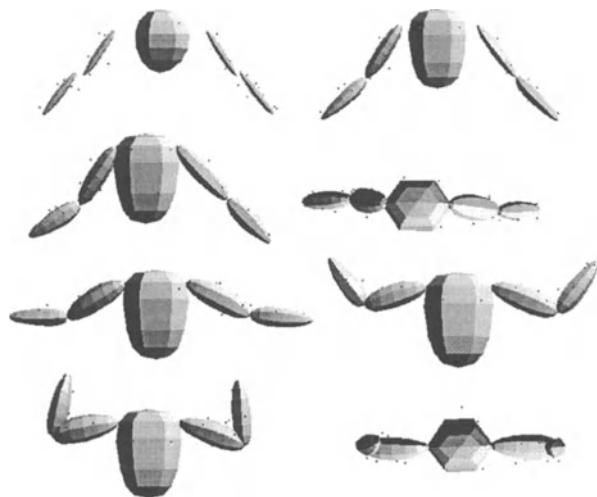


Figure 8.1 Tracking of raising and flexing human arm motion.

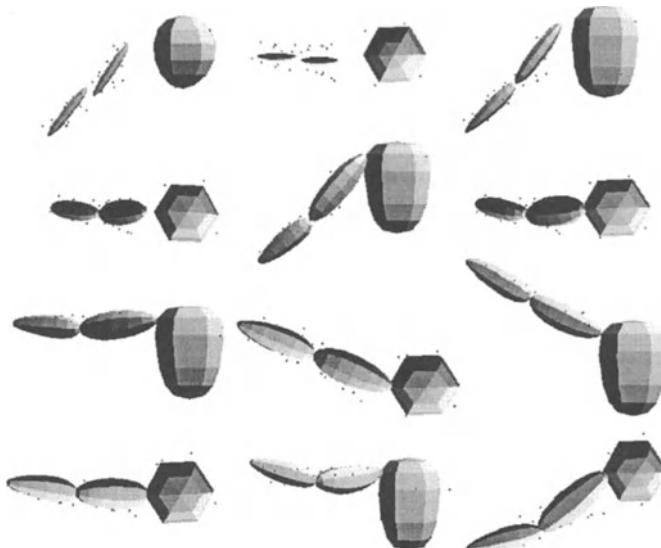


Figure 8.2 Tracking of rotating human arm motion.

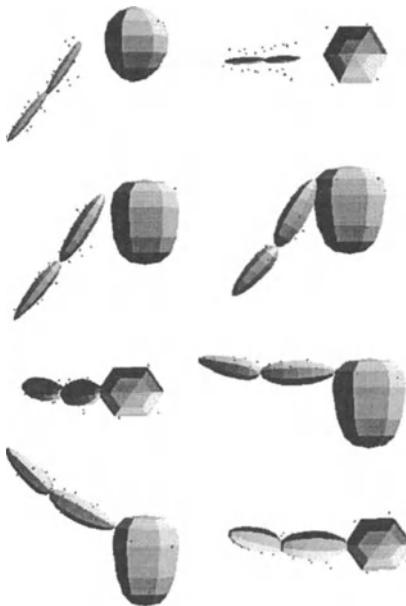


Figure 8.3 Tracking of up-down human arm motion.

We present three experiments involving 3D data from arm and torso motions of a human subject using the WATSMART system. Approximately 120 time frames were used in each experiment. The incoming data are segmented, inasmuch as individual datapoints can be associated with the correct body parts. We used 49 nodes for each deformable part, a superquadric, to model the arms and 36 nodes for the torso. The models were automatically initialized to the center of mass of the relevant data and their initial orientations were computed using the matrix of central moments of the data. The initial size along the longest axis (we initialized the model along the other two axes arbitrarily) was set by simply calculating the distance between the furthermost data points along the initial model centered coordinate system [263]. The initial Kalman filter covariance matrix was $\mathbf{P}(0) = \mathbf{I}$, and we used $\mathbf{Q} = 0.1 \mathbf{I}$ and $\mathbf{V} = 4.0 \mathbf{I}$. Local deformations were not permitted in these experiments, since the available data were very sparse, i.e., about 6 to 13 points per deformable superquadric part.

Fig. 8.1 shows shape and motion estimation results involving an articulated multipart model composed of 5 deformable models connected by 4 point-to-point constraints. The input data for the estimation were collected from the raising and flexing motion of the two arms of a human subject. Fig. 8.1(a) shows a frontal view of the data and the initial models. Fig. 8.1(b) shows an intermediate step of the fitting process driven by stabilized constraint forces (with $a = b = 6$) and data forces from the first frame of the motion sequence.

Figs. 8.1(c) and (d) show frontal and top views of the models fitted to the initial data. Figs. 8.1(e) and (f) show intermediate frames of the models tracking the nonrigid motion of the arms, while Figs. 8.1(g) and (h) show frontal and top views of the final position of the models.

In Fig. 8.2 (the numbering is top to bottom and left to right) we fit 3 deformable superquadrics to data collected from the rotating motion of the right arm of a human subject. Figs. 8.2(a) and (b) show two views of the range data and the initial models. Figs. 8.2(c) and (d) show an intermediate step of the fitting process driven by data forces from the first frame of the motion sequence. Figs. 8.2(e) and (f) show the models fitted to the initial data. Each of the remaining three frames in Figs. 8.2(g-l) shows (each shows a frontal and a top view) views of the models tracking the rotational motion of the arm.

In Fig. 8.3 we fit 3 deformable superquadrics to data collected from the up-down motion of the right arm of a human subject. Figs. 8.3(a) and (b) show two views of the range data and the initial models. Fig. 8.3(c) shows an intermediate step of the fitting process driven by data forces from the first frame of the motion sequence. Figs. 8.3(d) and (e) show the models fitted to the initial data. Fig. 8.3(f) shows an intermediate frame of the models tracking the nonrigid motion of the arm, while Figs. 8.3(g) and (h) show two views of the final position of the models.

The above experiments demonstrate that the combination of our constraint forces algorithm together with the application of the Kalman filter produce very good fits to the sparse and noise corrupted data.

Even though it is beyond the scope of this book, the characterization of the noise in the data (e.g., Gaussian or other type) is of paramount importance to achieve correct Kalman filter convergence results.

MULTI-LEVEL SHAPE REPRESENTATION

We present a method for the multi-level shape estimation and abstraction of an object's surface from range data. The surface shape is estimated based on the parameters of a superquadric that is subjected to global deformations (tapering and bending) and a variable number of levels of local deformations. Local deformations are implemented based on locally adaptive finite elements, whose shape functions guarantee C^1 continuity. The surface pose is estimated based on the model's translational and rotational degrees of freedom. The method is a generalization of the shape representation and estimation method presented in the previous chapters due to the introduction of multiple levels of local deformations. Through the application of Lagrangian mechanics, the model's translational, rotational, global and local deformation parameters are modified based on forces that originate from the range datapoints. These forces are computed from approximation errors between the model and the data.

Even though the formulation of global deformations is independent of the number of model nodes used, local deformations require a tessellation of the model surface into a grid of finite elements. The quality of the model fit to the data depends on the number of the finite elements used. In the previous chapters it was assumed that the sampling density of the finite element grid remained constant throughout the model fitting process. Clearly this poses a significant limitation in case of model fitting applications where the user assumes no prior knowledge of the complexity of the given data.

We will now generalize the previously presented shape representation approach. The algorithm first does a coarse model fit, solving for a first approximation to the translation, rotation and global deformation parameters and then does several passes of mesh refinement, by locally subdividing triangles based on the distance between the given datapoints and the model. In this way with a resulting small number of model nodes we can very efficiently and accurately represent the shape of an object at various levels of detail.

The local subdivision algorithm utilizes the properties of triangles bisected by the median that corresponds to the longest side. That is, the interior angles of the refined triangles do not go to zero as the level of subdivision goes to

infinity. Also this triangulation improves the shape regularity of the subdivided triangles as the subdivision proceeds. The local subdivision algorithm can be shown to satisfy conformity non-degeneracy and smoothness which are desirable properties for finite element meshes and ensure the accuracy of the solution.

The adaptive subdivision algorithm combined with the global deformations of the dynamic models, inherently allows the reconstruction, representation and abstraction of shape at various levels of detail. The shape hierarchy consists of using a superquadric with global deformations only, then global and one coarse level of local deformation and finally global and local deformations with various levels of local deformations extracted from our locally adaptive subdivision algorithm. This hierarchical surface detail representation is important in many computer vision and computer graphics applications, such as obstacle avoidance, object recognition and shape representation.

9.1 RELATED WORK

Most of the current shape recovery algorithms which use surface models, assume fixed-size grids [275, 217, 180, 53, 55, 64, 65]. Terzopoulos and Vasilescu [283] proposed a technique for adaptive subdivision of meshes consisting of nodal masses interconnected by adjustable springs. In case of local subdivision of the mesh, a computationally expensive constraint procedure has to be applied to ensure that the triangular structure of the mesh is maintained. McInerney and Terzopoulos [189] developed a deformable balloon model for shape estimation and tracking with uniform refinement capabilities. Huang and Goldgof [123] present a geometric adaptive subdivision algorithm for nonrigid motion analysis which uses planar triangular patches. Tanaka and Kishino [270] develop a geometric adaptive mesh generation algorithm for surface reconstruction. Even though the algorithm supports local subdivision, it does not use a 3D model, and in order to guarantee a smooth solution special algorithms need to be employed to deal with cracks often occurring during subdivision. Delingette [66] uses simplex meshes for reconstructing the geometry of complex objects from range data. The mesh is refined based on its curvature, its distance from the 3D data and its elongation. This method also allows changes in the topology of the mesh, but the process requires manual intervention. This approach is most closely related to ours (see also [147]), but it is mostly suited for smoothly varying objects due to the use of curvature. Motivated by the work of Metaxas and Terzopoulos [278] on deformable superquadrics with local and global deformations, Vemuri and Radisavljevic [284] developed a probabilistic wavelet-based hierarchical representation for the local deformations defined in [278] where a uniform non-adaptive mesh was used. As opposed to shape abstraction, their main motivation is the use of this representation scheme as prior information in a probabilistic framework for efficient surface reconstruction. The method is most useful for the estimation of classes of objects for which training data is available from objects within that particular class.

In computer graphics there have been many attempts to develop techniques for surface reconstruction [251, 120, 15]. The results are impressive, but the emphasis of the work is on accurate shape reconstruction of complete and dense data with very small amounts of noise, as opposed to a multi-level shape representation with abstraction. Schmitt et al [251] present an adaptive subdivision method for object reconstruction from range data. The method is restricted to surfaces or rectangular topology and assumes dense data. Hoppe et al [120] presented a method for the recovery of surfaces of variable topology with sharp features such as creases and corners. However, the authors have not yet shown its use in case of sparse range data with significant amount of noise and/or outliers.

Contrary to many of the above techniques, the proposed method provides a very efficient and intuitive way of representing and abstracting shape at various levels of detail and can be applied to both sparse and dense range data.

9.2 DEFORMABLE MODELS: GEOMETRY AND DYNAMICS

For the applications in this chapter, the deformable model's global deformations will be modeled based on a superquadric ellipsoid that can undergo a tapering deformation in the x and y model frame axes, and a bending deformation in the x model axis (see Chapter 2 for formulas). Local displacements \mathbf{d} are computed based on the use of C^1 triangular finite elements that were presented in Chapter 4, section 4.4.

When fitting the model to visual data our goal is to recover the vector \mathbf{q} of the model's degrees of freedom which consists of the rotation, translation, global and local degrees of freedom. The model fitting procedure is done in a physics-based way based on (3.51). The force assignment from a datapoint \mathbf{z} to the deformable model is done based on algorithm 4 in Chapter 5. The corresponding model point has material coordinates \mathbf{u}_z and the force assignment from this point to nodes of the finite element in which it lies is computed based on (4.3). This process can be visualized in Fig. 9.1.

9.3 LOCALLY ADAPTIVE FINITE ELEMENTS

The deformable models we use are topologically isomorphic to a sphere and can be mapped to a rectangular material coordinate space $\mathbf{u} = (u, v)$ (see Chapter 4). We discretize the deformable model in the rectangular material coordinate domain and define the triangular elements. Every triangular finite element has three nodes and in every node i , we store the vector of nodal variables $\mathbf{q}_{d,i}$

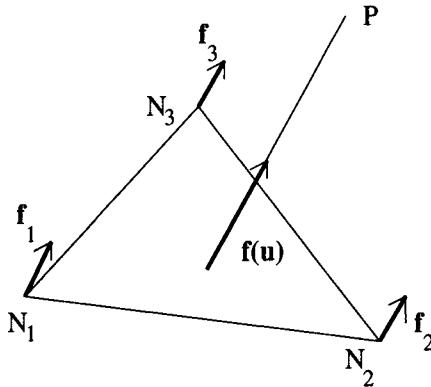


Figure 9.1 Extrapolation of force to the element nodes.

from (4.36), since we are using C^1 elements. We use a thin plate under tension strain energy (3.35) and derive the corresponding stiffness matrix based on the methodology described in Chapter 4.

The adaptive subdivision algorithm we will develop, can both accurately and efficiently represent shape. Without prior knowledge of the complexity of the given data, new model nodes are added to the initial grid in order to minimize the error of fit of the model to the data. The experiments we present in a later section clearly demonstrate the limitations of a uniform static grid in terms of the accuracy and efficiency of shape estimation. Another disadvantage of using a uniform grid is that the tessellation produces triangles whose orientation is uniform and does not necessarily match the complexity of the surface to fit. The adaptive subdivision algorithm overcomes this problem. In addition, the process of model fitting is now automated, since the user does not have to experiment with a variety of grid sizes before an acceptable fit can be achieved.

9.3.1 Criterion for Finite Element Subdivision

We use a criterion based on the distance from the datapoints to the finite elements, to decide whether an element or elements should be subdivided. We first compute using the above mentioned algorithm the point \mathbf{u}_z on the model whose distance $d(\mathbf{u}_z)$ is the minimum from the given datapoint \mathbf{z} . If

$$d(\mathbf{u}_z) > \tau_d, \quad (9.1)$$

where τ_d is a threshold, we subdivide the elements that this nearest model point is on. We distinguish the following three cases.

1. If \mathbf{u}_z lies inside an element, then the element is selected for subdivision.

2. If \mathbf{u}_z lies on an edge, then the two adjacent elements to the edge are subdivided.
3. If \mathbf{u}_z is a model node, then all the elements adjacent to the node are subdivided.

Once the above criterion is satisfied we subdivide the chosen elements and apply the following subdivision algorithm to ensure that the resulting grid has properties necessary for the application of the finite element method. It is worth mentioning that the curvature calculation is very sensitive to noise, and since we want to use our technique in case of sparse data, we did not consider using curvature as a criterion for subdivision.

9.3.2 Subdivision Algorithm

Our subdivision algorithm has the following two basic steps and is an adaptation for computer vision applications of the adaptive finite element technique developed by Rivara [238]. The first is a *bisection* operation in which a single finite element is subdivided according to a certain rule. The second is a *conforming* recursive operation which ensures that the rest of the finite elements satisfy properties necessary to apply the finite element method. We will now describe in detail each of these operations. The subdivision algorithm is employed in the material coordinate domain of the model which for shapes of genus 0 can be mapped to a rectangle [181] whose two out of the four sides are identical. However, during the application of the algorithm we treat them as being distinct and we remedy any node discrepancies on the identical sides based on the approach we will explain later.

Step 1: Bisection Operation

If the above defined distance criterion is met for a particular finite element, we perform a *bisection operation* as follows:

- Let T be a triangle with vertices A, B , and C ; We subdivide the triangle T into two triangles by bisecting it along its longest edge (the length is defined in 3D). Let AB be the longest edge of T , and D the midpoint of AB . Then T is subdivided into two triangles, ADC and BCD as shown in Fig. 9.2(a).

This subdivision has been shown to provide properties desirable for use in finite element applications. First, none of the elements will become ‘skinny’ (triangles with very acute angles) as the level of subdivision increases. Rosenberg and Stenger [241] proved that if α_i is the smallest angle of the triangulation obtained by the i -th iterative subdivision, then $\alpha_i \geq \frac{\alpha_0}{2}$ for any i , where α_0 is the smallest interior angle of the initial triangulation. Second, the subdivision improves the shape regularity of the triangles that is, the ratio between the longest and

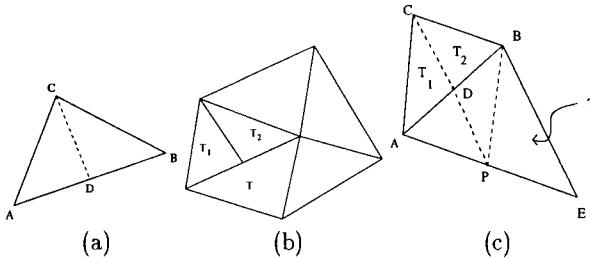


Figure 9.2 Various subdivision examples. (a) Subdivision of a triangle by the longest edge, (b) An example of non-conforming triangulation, (c) An illustration of conforming operation.

shortest sides of the triangles gets smaller as the level of subdivision increases [266], in case of triangles with large such ratios.

Step 2: Conforming Operation

The second part of the algorithm ensures that the resulting finite element grid generates properties necessary for the application of the finite element method. A triangulation is defined to be conforming if any two adjacent triangles must share either a common vertex or a common edge [306]. In Fig. 9.2(b), the triangulation is not conforming, because conformity is violated between T_1 and T , and between T_2 and T . In the finite element method, we must maintain the continuity across inter-element boundaries, i.e., it is necessary to maintain the conformity of the triangulation.

In Fig. 9.2(c) we demonstrate how to address this problem. If we introduce a new node, D , as a result of bisecting element ABC , the element, T , adjacent to the subdivided edge AB becomes non-conforming. In order to ensure conformity, further subdivision must be performed on T along the common edge with midpoint D . However, it is possible that the common edge may not be the longest edge of T . Therefore, this subdivision will cause the triangulation to lose the aforementioned properties of shape regularity. To remedy this problem, we take the following approach in subdividing element T as shown in Fig. 9.2(c). We first bisect T by its longest edge, AE , at its midpoint, P . If AE is the common edge, then we stop subdividing. Otherwise, we further subdivide T by connecting P to the midpoint D of AB . As a result of this process, conformity is preserved and the subdivision will not produce ‘skinny’ triangles. This process is called a *conforming operation*.

In our local subdivision algorithm, the conforming operation is performed whenever subdivision of an element causes non-conformity. The conforming operation, however, may create new non-conformity. In order to ensure the conformity, this conforming operation is recursively applied until the triangulation becomes entirely conforming. This recursive process is guaranteed to stop because there is only a limited number of triangles to start with. Fig. 9.3 illustrates an example of applying a series of conforming operations which were necessary

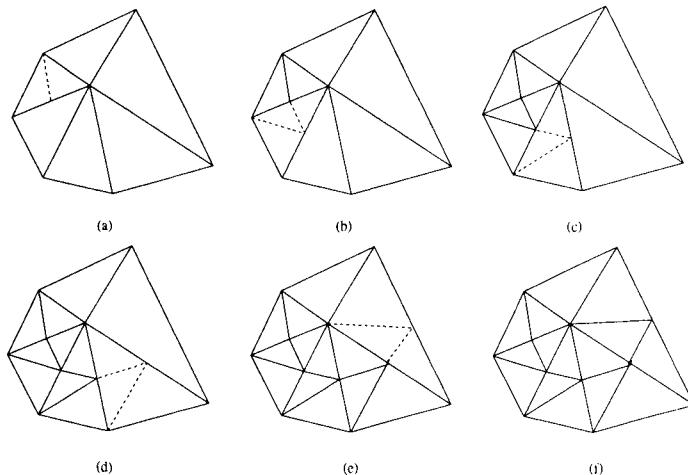


Figure 9.3 An example of recursive local subdivision in material coordinate space.

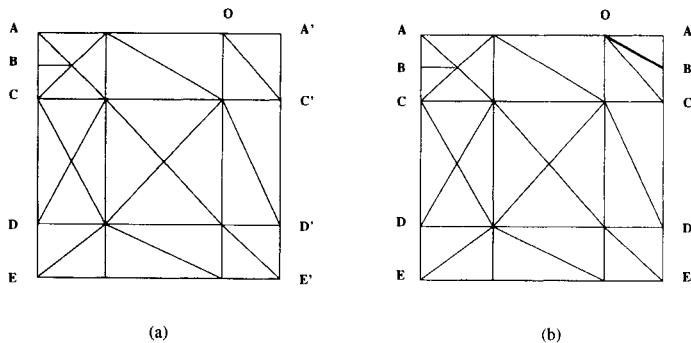


Figure 9.4 An example of how node discrepancy is accounted at the end of the recursive subdivision in case of objects that are topologically equivalent to a sphere. In this example sides AE and $A'E'$ are identical and should have the same number of nodes. (a) shows the result of recursive subdivision where node B has no corresponding node B' . (b) shows the correction to this discrepancy through the introduction of node B' and the two triangles $OA'B'$ and $OB'C'$ to replace $OA'C'$.

because of propagating non-conformity. This example captures shapes that are topologically equivalent to a sphere (these are the types of objects whose shape we estimate in this chapter) since they can be mapped in material coordinates to a polygon. The simplest type of polygon to define such objects is a rectangle whose two of the four sides are identical (edges AE and $A'E'$ in Fig. 9.4) and we have used in the past [181, 180]. As mentioned earlier, the triangulation is performed in the material coordinate space and the two corresponding sides AE and $A'E'$ of the rectangle are treated as non-corresponding when the algorithm is applied. Therefore the algorithm is guaranteed to terminate. After the termination of the recursive conforming operation (Fig. 9.4(a)), all nodes on the two identical sides of the rectangle are checked so that an equal number of them exists on each of the corresponding sides AE and $A'E'$, and are also in the same location. Wherever there is a discrepancy (e.g., node B in Fig. 9.4(a) has no corresponding node on side $A'E'$), it is automatically corrected by the introduction of new nodes (e.g., node B' on side $A'E'$ is introduced in Fig. 9.4(b)) at the correct locations on each of the corresponding sides of the rectangle. We also adjust locally the triangulation (e.g., introduction of the two triangles $OA'B'$ and $OB'C'$ to replace $OA'C'$ in Fig. 9.4(b)) to ensure that all finite elements share the same nodes.

Algorithm

Let the *subdivision set* be the set of finite elements which have been chosen for subdivision, but have not yet been subdivided. Based on the above two operations, our local subdivision algorithm can be described as follows:

```

While the subdivision set is not empty
BEGIN
    Set  $T$  as a triangle from the subdivision set.
    Remove  $T$  from the subdivision set.
    If  $T$  has not been bisected then
        BEGIN
            Set  $E_{longest}$  as the longest edge of  $T$ .
            Subdivide  $E_{longest}$ .
            Bisect  $T$  by  $E_{longest}$ .
            Set  $T'$  as the adjacent triangle of  $T$  by the edge  $E_{longest}$ .
            Conform( $T'$ ,  $E_{longest}$ ).
        END
    END

```

The conforming operation, **Conform**(T' , E'), is described in pseudocode as follows:

```

Set  $E'_{longest}$  as the longest edge of  $T'$ .
If  $E'_{longest}$  is the same as  $E'$  then

```

```

BEGIN
    Bisect  $T'$  by  $E'$ .
    Return.
END
Subdivide  $E'_{longest}$ .
Bisect  $T'$  by  $E'_{longest}$ .
Set  $\hat{T}$  as one of the sub-triangles from the previous step
    which contains the edge  $E'$ .
Bisect  $\hat{T}$  by  $E'$ .
Set  $\tilde{T}$  as the adjacent triangle of  $T'$  by the edge  $E'_{longest}$ .
Conform( $\tilde{T}$ ,  $E_{longest}$ ).

```

In summary, our local subdivision algorithm satisfies several desirable properties for finite element grid generation. They are: 1) conformity: any two adjacent elements share only either a node or an edge; 2) non-degeneracy: the triangulation maintains the shape regularity of the refined elements, i.e., the elements do not become ‘skinny’; and, 3) smoothness: there is no abrupt size difference between adjacent elements, and hence the transition between small and large elements is smooth.

9.4 SUMMARY OF MODEL FITTING TO RANGE DATA

In the previous sections we defined the translation, rotation, global and local degrees of freedom, and the adaptive finite element subdivision algorithm. Based on the above formulations our model fitting algorithm has the following steps:

1. Initialize the model to the data by placing the model’s model frame to the center of mass of the data and estimate its orientation based on the matrix of central moments (see Chapter 6).
2. Fit the model to the given data by using only the translation, rotation and global degrees of freedom.
3. Fit the model to the data using both local (do not apply any subdivision) and global deformations.
4. Apply the recursive finite element subdivision algorithm in locations where the error of fit between the model and the data is beyond a user specified threshold. At the end of every subdivision level (the end of each recursion step), fit the model to the data using both global and local deformations to further refine the fit.
5. Repeat Step 4 until the error of fit is everywhere below the user specified threshold.

It is very important to mention that the fitting of the model to the given data is done automatically. The user only specifies a threshold for terminating the fitting of the model to the data, and the strength of the forces. The threshold is defined as a percentage of the longest dimension of the data, which is computed based on the use of the matrix of central moments.

9.5 EXPERIMENTS

We have carried out various experiments to test our locally adaptive finite element algorithm. These include 3D range data taken from the Cyberware, Inc., 3D digitizer, and a human head model provided by Viewpoint, Inc. Using our new force assignment algorithm the models deform globally and locally and subdivide locally to fit the given data. Our experiments run at interactive rates on an R4000 Iris Crimson workstation with VGX graphics. In all the experiments we used a time step size equal to $\times 10^{-5}$ (iterating with the Euler method) and a unit damping matrix \mathbf{D} , while the threshold for local subdivision was roughly $\tau_d = 0.1\%$ of the longest dimension of the given data. In addition, the force strength is defined by the user. In every experiment we compute the model's error of fit with respect to the input data. The error of fit is defined in terms of the distances of the fitted deformable model's finite elements from the original input data (we normalize it with respect to the model's biggest dimension). The fitted model is obtained by fitting the deformable model to the input data until the movement of each element becomes relatively negligible. This is achieved by measuring the difference between two positions of each element obtained from two consecutive iterations during fitting, and we allow the model to continue fitting until the maximum relative difference in the position of the model elements is less than a threshold value of the order of 10^{-5} . Even though we did not experience them, there are cases where the fitting process can become unstable. In such cases an adaptive integration algorithm (e.g., adaptive Runge-Kutta) solves this problem. In addition, we initialize the model based on the center of mass of the data and the matrix of central moments, and we first fit the model to the data using only the global deformations.

In the first experiment, a deformable model with 66 nodes was fitted to 857 3D datapoints sampled from a biomedical image of the left part of a human lung (Fig. 9.5(a)). The initial fitting model was an ellipsoid (Fig. 9.5(b)). Fig. 9.5(c) shows the model after global deformations. Fig. 9.5(d) shows the model after local deformations. Fig. 9.5(e) shows the model with 243 nodes after three levels of our locally adaptive subdivision. Comparison of Fig. 9.5(d) and Fig. 9.5(e) shows that the subdivision was concentrated in the boundary area with more complex geometry. The fitting process took approximately five and a half minutes and 400 iterations.

Table 9.1 shows error statistics collected from the hierarchical fitting of a deformable model to data from a human lung. In the rows we present for each level of shape detail the number of nodes in the model, the number of elements

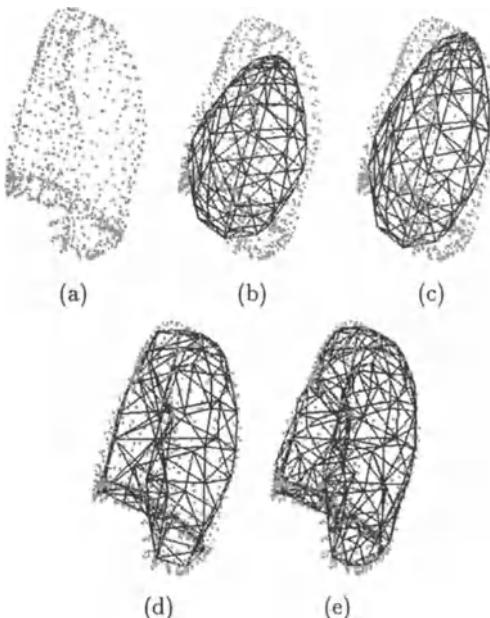


Figure 9.5 Fitting of model to data of a human lung. (a) input data of a human lung (left part), (b) model initialization, (c) model fitted to data with only global deformations, (d) model fitted to data with global and local deformations, (e) model fitted to data after three levels of local subdivision.

Model	Initial Fit	Sub-1	Sub-2	Sub-3	No Sub #1	No Sub #2
# nodes	66	129	168	243	256	627
# elements	128	254	332	482	512	1250
# iterations	114	169	212	373	209	222
CPU mins.	1	2.9	3.8	5.5	3	6
$mean_{error}$	0.081739	0.023317	0.013317	0.008100	0.019040	0.011834
max_{error}	1.351201	0.371757	0.318139	0.104073	0.281506	0.247791
σ_{error}	0.162214	0.038362	0.017991	0.010340	0.034987	0.026694
σ^2_{error}	0.026313	0.001472	0.000324	0.000107	0.001224	0.000713

Table 9.1 Error statistics from fitting the model to 857 data points of a human lung. The second column shows statistics from fitting the initial model with 66 nodes to the data. The third, fourth and fifth columns represent statistics after the first, second and third levels of grid subdivision, respectively. The sixth and seventh columns represent fitting results in case of non-adaptive grids with a constant number of nodes.

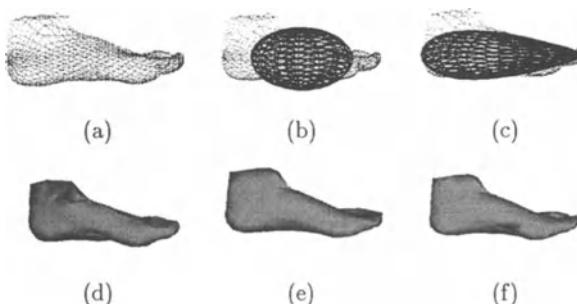


Figure 9.6 Fitting of model to foot data. (a) Foot data, (b) Model Initialization, (c) Intermediate step of model fitting to the data with apparent global deformations, (d) Model fitted to data without local subdivision, (e) Model fitted to data after one level of local subdivision, (f) Model fitted to data after four levels of local subdivision.

in the model, the number of iterations, the CPU time in minutes, the mean error value, the maximum error value, the standard deviation and the variance of the error of fit. Then we subdivided the model elements as explained in the previous sections. We started fitting the model (Initial Fit column in the table) with 66 nodes and 128 elements while the number of input data points was 857. At the first level of subdivision (Sub-1 column), 87 elements were selected for subdivision, and the model resulted in a total of 129 nodes and 254 elements. At the second level of subdivision (Sub-2 column), 50 elements were selected and the subdivision process produced a total of 168 nodes and 332 elements. At the third level (Sub-3 column), 77 elements were selected, and the subdivision process generated a total of 243 nodes and 482 elements.

We also fitted a model with a regular constant grid whose number of elements is comparable to the number of elements of the model after three levels of local subdivision and measured the error of fit. This model contained 256 nodes and 512 elements. As shown in the table, the mean and maximum errors of fit derived using this model notably exceed those of the subdivided model. In the last column of Table 9.1 we also show the error of fit obtained by fitting a model with a constant number of 627 nodes and 1250 elements. From this last example it is apparent that our locally adaptive subdivision algorithm significantly improves the quality of model fitting.

In the second experiment we use range datapoints obtained from the Cyberware, Inc., 3D digitizer. In Fig. 9.6 we fit a deformable model with initially 227 nodes, to 3825 3D range datapoints obtained from a mannequin foot. The local deformation stiffness parameters of the model were $w_{00} = 0.5$, $w_{01} = 0.5$, $w_{10} = 0.5$, $w_{02} = 0.1$, $w_{11} = 0.1$ and $w_{20} = 0.1$. The initial model was an ellipsoid ($\mathbf{q}_s = (2.3, 0.3, 0.5, 0.3, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0)^T$) and the force strength parameter was $\beta = 20.0$. Fig. 9.6(a) shows the given foot data. Fig. 9.6(b) shows a view of the range data and the initial model, while Fig. 9.6(c)

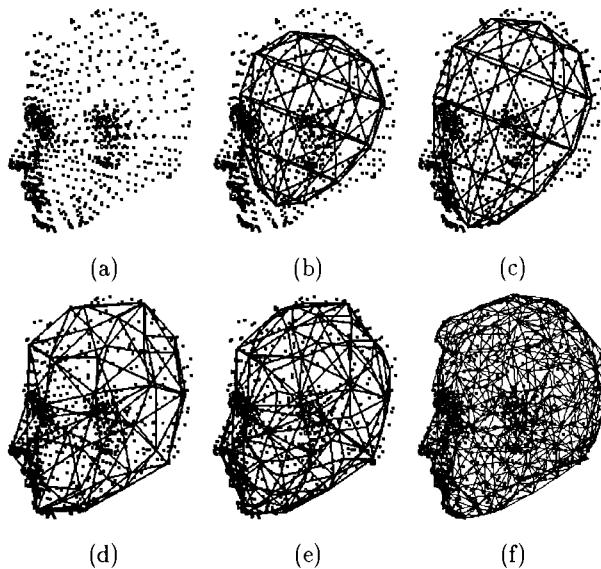


Figure 9.7 Fitting of the model to input data of a human head. (a) input data, (b) the initial model, (c) the model after global deformations, (d) the model after local deformations, (e) the model after two levels of subdivision, (f) the model after four levels of subdivision.

shows an intermediate step in the fitting of the model to the data where the global deformations are apparent. Fig. 9.6(d) shows the model fitted to the data without local subdivision, Fig. 9.6(e) shows the model fitted to the data after one level of local subdivision, while Fig. 9.6(f) shows the final model fitted to the data after four levels of local subdivision. After six and a half minutes and a total of 452 iterations, the new final number of model nodes is 640, which is significantly smaller than the number of given datapoints.

In the next experiment we fit a deformable model with 47 nodes to 1269 3D data points defining a human head. The input data were obtained from the Viewpoint, Inc. The local deformation stiffness parameters of the model were $\omega_{i,j} = 0.05$. The initial model was an ellipsoid ($\mathbf{q}_s = (7.0, 0.5, 0.5, 0.6, 1.0, 1.0, 0.0, -0.3, 0.0, 0.0, 1.0)^T$) and the force strength parameter was $\beta = 1.0$. Fig. 9.7(a) shows a view of the range data. Fig. 9.7(b) shows the initial model. Fig. 9.7(c) shows the model after global deformations. Fig. 9.7(d) shows the model after local deformations. Fig. 9.7(e) shows the intermediate model after two levels of local subdivision. Fig. 9.7(f) shows the final model after four levels of local subdivision that took five minutes and a total of 283 iterations.

Figs. 9.8 (a) and (b), respectively, show a front view and a back view of human body figures displayed at three different levels of detail. The human body figure consists of 15 parts: head, torso, lower torso, 3 parts for each arm, 3 parts for each leg. For coarser levels of detail, approximations of each body part were

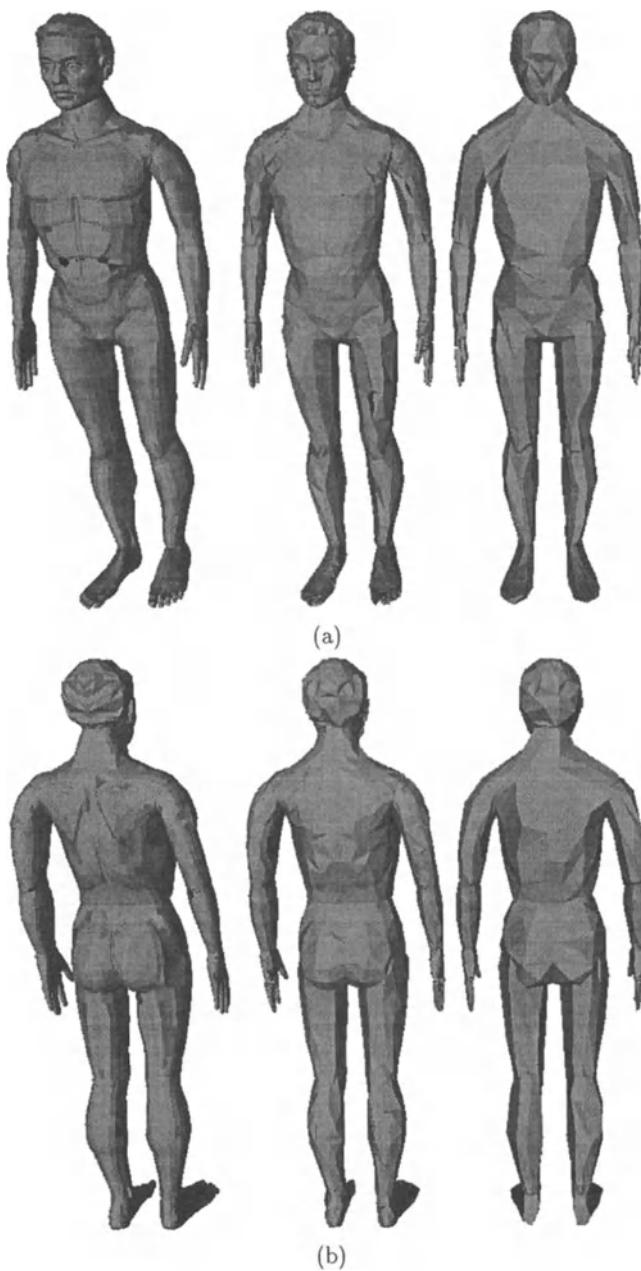


Figure 9.8 A human body displayed at three different levels of detail: (a) the front view, (b) the back view.

obtained as described in the previous experiments. The numbers of polygons used at each level of representation were 18155, 7292, and 2260, respectively, and the numbers of nodes were 18005, 3696, and 1180, respectively.

TOPOLOGICALLY ADAPTIVE MODELS BASED ON BLENDING

We will now extend and generalize the shape coverage of our deformable models in order to recover and abstract the shape of more complex shapes.

We present a method for shape representation called *blending*, which allows two different shapes to be combined into a single blended shape. Specific regions on each of the two shapes are selected and “glued” together. It is also possible to glue in a hole, permitting the creation of holes through any part of the shape. The application of blending in a hierarchical way provides the ability to perform multiple blends. This arbitrary blending of global shape is a generalization of our initial axial blending method presented in [60]. The importance of this new method is that it also provides shape abstraction and can therefore be used for recognition purposes.

Shape estimation is accomplished using the physics-based deformable model framework presented in the previous chapters. The ability to automatically *adapt* the model based on the detection of force equilibration is introduced, which allows blending to be added to particular regions of the model when needed and allows the model to better represent the data. A similar method for the automatic addition of holes using self-proximity detection is also introduced. By repeated application of blending, the shape model can *evolve* from the initial shape of a sphere to a representation that accurately fits the data.

Upon the addition of a blend, only the representation of the shape changes—the geometry remains the same. We feel that by avoiding abrupt geometric changes, the impact of the decision to apply the blend is reduced, leading to greater robustness (when using global shapes, clearly some representational change is necessary).

The evolution of the model produces a hierarchy of blended shapes. Due to the global nature of the component shapes, a symbolic representation of the model can be created directly from this hierarchy. This symbolic representation is expressed as a tree where qualitative adjacency relationships are described, and the leaves are shape primitives.

As opposed to all previous part-based shape estimation techniques, the model can be represented as a set of connected components where each component is an integral part of the model. An additional benefit of using globally parameterized models is their robustness to noise and their suitability in estimating the underlying shape of sparse and incomplete data.

We present several experiments demonstrating the shape coverage of the method and experiments to test the stability of the method to data acquired from different viewpoints.

10.1 RELATED WORK

The development of models used in applications ranging from reconstruction to recognition makes us aware of their often conflicting requirements—representational accuracy versus abstractive power.

Many researchers have addressed this problem by creating expressive models using many parameters that are appropriate for reconstruction only [66, 165, 168, 269, 274] or have developed compact models with limited coverage suitable only for recognition [31, 32, 108, 171, 263].

Models for both reconstruction and recognition have also been presented [60, 85, 159, 181, 217, 285]. Shape has been represented using a collection of parameters ordered by the level of detail [217, 285]. Techniques using superquadrics [85, 159] were presented to obtain part-level models. The models in [181, 278] represented prominent shape features using global deformations and surface detail using local deformations.

Axial blending [60] allowed the combination of shape primitives along common axes. In addition to a compact representation, the distinguishing feature of these models was the ability to represent objects with varying topology (such as a sphere and a torus) in a unified way.

In order to be applied to a wide class of objects, a shape model should have no a priori parameterization or imposition of physical properties. Examples of such restrictions can be seen in modal analysis [217], where the modal calculation depends on some predefined elastic properties, and in [60, 181], where predefined types of global and local deformations constrain the class of objects that can be represented.

Without prior knowledge, such a shape model should have an initial shape that does not favor *any* particular shape—a sphere is a reasonable choice. Furthermore, the associated shape estimation technique should allow the model to evolve based on the shape of the given data, making the representation data driven and not predefined. In the process of model evolution, abrupt changes in the model should be avoided, since such drastic decisions are rarely robust.

Finally, the object representation should satisfy the requirements of both reconstruction and recognition. An additional, useful feature of our method is that it can be used with sparse and incomplete data, as opposed to level set methods [168, 145] that require complete and dense data.

10.2 BLENDED SHAPES

In the following sections, we will describe the process of shape blending. The models used to perform the blending are 3-D surface shape models defined by a global shape s . Locations on the shape s are identified by material coordinates $\mathbf{u} = (u, v)$ which are specified over a domain Ω . For every 3-D shape (such as a superellipsoid), we have $s : \Omega \rightarrow \mathbb{R}^3$. An example (in this case, a sphere) is shown in Fig. 10.1, and shows how the material coordinates (the domain Ω) are “folded up” resulting in the closed shape s .

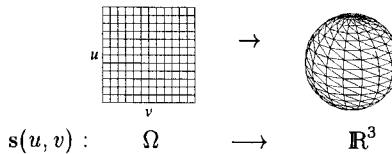


Figure 10.1 Example shape primitive $s(\mathbf{u}) = s(u, v)$.

To represent the shape, a mesh of nodes is used, where each node is assigned a unique point in Ω . The edges connecting the nodes (determined by adjacency in Ω) specify the topology. Nodes are merged together where points in Ω map to the same 3-D model location (such as for the poles of a sphere) resulting in a mesh topology that agrees with the topology of the shape primitive.

A *blended* shape is a combination of two component shapes. The desired pieces of each component shape are selected, and are “glued” together smoothly using linear interpolation. The gluing operation performed here is a generalization of the axial blending presented in [60]. We can blend together two shapes using the following formula:

$$s(\mathbf{u}) = s_1(\mathbf{u})\alpha(\mathbf{u}) + \mathcal{R}\left(s_2(\mathcal{B}(\mathbf{u}))\right)\left(1 - \alpha(\mathcal{B}(\mathbf{u}))\right), \quad (10.1)$$

where s_1 and s_2 are the component shapes, as in Fig. 10.2(a), and $\alpha : \Omega \rightarrow [0, 1]$ is the blending function which controls how each of the component shapes are expressed and how they are glued together. α is specified using the blending regions shown in Fig. 10.2(b). These blending regions are overlaid on the material coordinate spaces of the component shapes. The portion of these spaces corresponding to the retained surface is shown in white, while the portion which

is removed is shown in gray. The retained portions of the surfaces corresponding to these regions are shown in Fig. 10.2(c). Basically, the blending regions specify where to “cut” the shape (the boundary line), and which part of the shape to keep (white or gray). Ideas similar to these cutting and gluing operations have been used in studying manifold surgery for topology [115], where manifolds of arbitrary topology are constructed using a few simple components.

In order to glue arbitrary parts of shapes together, they first must be prepared—the mappings $\mathcal{B} : \Omega \rightarrow \Omega$ and $\mathcal{R} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ serve this purpose. For the moment, we will assume that both \mathcal{B} and \mathcal{R} are identity mappings.

Finally, s is the resulting blended shape, shown in Fig. 10.2(d).

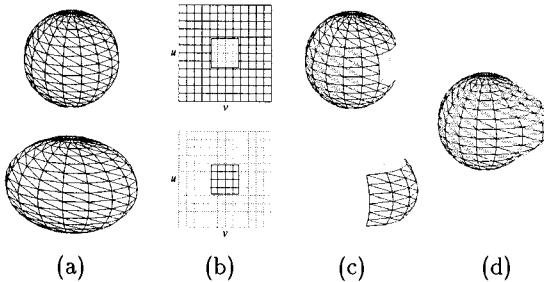


Figure 10.2 Simple blending example.

The resulting blended shape s has a smooth transition between each of its components (s_1 and s_2). For the example blending region given in Fig. 10.3(a), the corresponding blending function α is shown in (b). Notice how α is 1 where the region is white, 0 where the region is gray, and has a smooth transition from 0 to 1 connecting these two regions.

The area on the shape primitive expressed by this particular blending region is displayed in Fig. 10.3(c) as the white region on the shape. A discussion of the implementation of α , as well as the size of the transition region (the “steepness” of the plateau) is given in section 10.2.3.

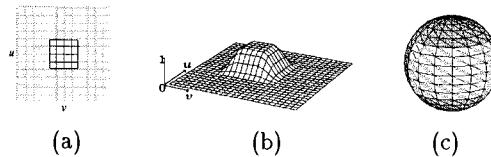


Figure 10.3 Example blending function.

The blending regions of each of the component shapes in Fig. 10.2(b) line up exactly. In general, the blending region boundary for s_1 does not have such a simple correspondence with the blending region boundary for s_2 , since arbitrary

locations of the component shapes can be selected. The invertible mapping \mathcal{B} is used to form a simple correspondence.

Figs. 10.4(b) and (c) show how \mathcal{B} maps the blending region for s_2 to correspond with s_1 , shown in (a). \mathcal{B} maps the boundary *and* its neighborhood, permitting a correspondence to be performed for those points near the boundary where linear interpolation is performed (where $0 < \alpha < 1$).

Each boundary has an associated direction (clockwise or counter-clockwise), which must be preserved by \mathcal{B} . Not doing so can produce a non-orientable self-intersecting surface (not a likely property for an estimated shape!).

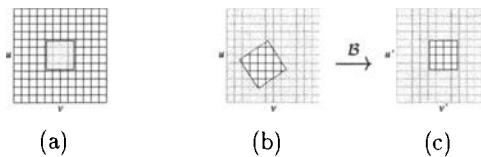


Figure 10.4 Blending region alignment using \mathcal{B} .

In addition to the correspondence in material coordinate space performed by \mathcal{B} , there must also be a correspondence in 3-D space. In other words, the two components must be spatially aligned before blending is performed. This spatial correspondence is performed by \mathcal{R} , a rigid transformation (translation and rotation) applied to s_2 . An example of this spatial alignment is shown in Fig. 10.5, where the result of applying \mathcal{R} is shown in Fig. 10.5(c). The blending regions used are given in Figs. 10.4(a) and (b).

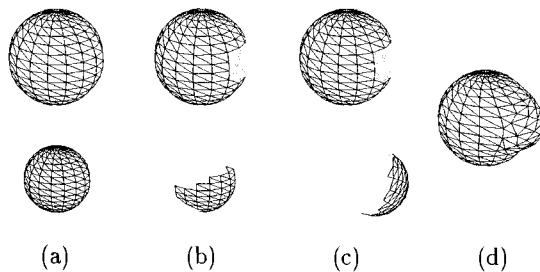


Figure 10.5 Another blending example.

The mesh of the blended result in Fig. 10.5(d) is a composite mesh formed by combining the meshes in (c). A mesh merging algorithm [244] can be used to perform this merging as a post-processing step to blending. The bottleneck in mesh merging algorithms is the nearest-node computations, which become constant time operations since \mathcal{B} provides correspondence between meshes.

10.2.1 Hole additions and indentations

The addition of a hole requires two blending regions—two parts of a shape are cut out, and a hole is glued in its place. Fig. 10.6(a) shows a sphere with two parts removed. The blended regions for this sphere are also displayed, showing how two regions in Ω are removed. Fig. 10.6(b) shows the hole of a torus. Two separate cuts are made around the torus hole, and the resulting hole is glued into the sphere, forming the torus shown in Fig. 10.6(c).

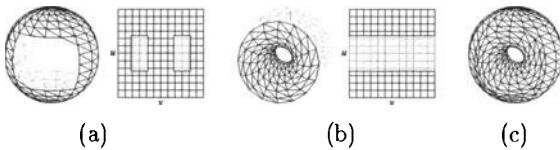


Figure 10.6 Hole addition example.

Unlike the holes in [60] which could only be added between the poles of a sphere, the holes presented here are general, and can be added between any two locations of a blended object. We will see in section 10.3.1 how the hole additions are performed automatically during estimation.

Indentations can also be produced using blending. Fig. 10.7(a) shows a cylinder with part of the top removed by the given blending region. Fig. 10.7(b) shows a shape which is turned inside-out by having a negative axis length, producing a closed surface with the surface on the inside. Removing part of the surface reveals the inside, which can be used for making a cavity. Blending together the cylinder and inverted cylinder produces the cup shown in Fig. 10.7(c).

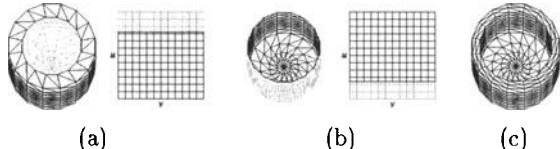


Figure 10.7 Indentation formation.

10.2.2 Hierarchy of blended shapes

In the sections above, the blending described is performed on simple shapes. Without additional machinery, we can perform blending on component shapes which are already blended. This results in a tree structure, where the leaves of the tree are shape primitives, and the internal nodes of the tree are blended shapes. Since \mathcal{B} is invertible, it can be used to find corresponding points on any two component shapes anywhere in this tree structure.

Since holes can be added anywhere, this means that a closed object of any topology can be represented. We will see an example of the fitting of a two-holed object in section 10.4. We can produce a symbolic description of a blended shape using this hierarchy. For each of the experiments performed in section 10.4, a symbolic description is provided. Such a description may be useful for recognition purposes.

10.2.3 Representation of blending regions

For the applications we present, the blending regions are defined as four-sided regions in Ω space. Each region has 6 parameters, $c_u, c_v, p_{1u}, p_{1v}, p_{2u}$ and p_{2v} (where the vector p_1 is linearly independent of p_2), as shown in Fig. 10.8(a). For blending regions which include a pole of s , a slightly different formulation will be needed.

There are also additional parameters for each blended shape—a translation and rotation to specify \mathcal{R} (6 parameters), the parameter d which controls the extent of the transition region of α (plateau “steepness”), and h which restricts the blending function range to $[h, 1]$. h is used to control the magnitude of the contribution of s_2 to the resulting shape (h is used to smoothly add a hole in section 10.3.1).

A straightforward basis transformation operation maps points in Ω to points in the basis (p_1, p_2) . We will denote the point $\mathbf{u} \in \Omega$ expressed in this basis as $\bar{\mathbf{u}}$. Points on the boundary of the blending region have $\|\bar{\mathbf{u}}\| = 1$, and points on the interior have $\|\bar{\mathbf{u}}\| < 1$. \mathcal{B} is constructed by transforming a value of \mathbf{u} to $\bar{\mathbf{u}}$ using the blending region for s_2 , and then back using the blending region for s_1 . α is computed from $\bar{\mathbf{u}}$ by setting α to 0 where $\|\bar{\mathbf{u}}\| < 1 - d$, to 1 where $\|\bar{\mathbf{u}}\| > 1 + d$, and to an intermediate value using the Hermite polynomial $H(x) = 3x^2 - 2x^3$ for $x \in [0, 1]$ to produce a C^1 smooth piecewise polynomial surface.

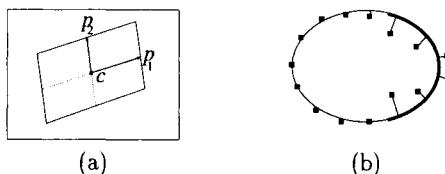


Figure 10.8 (a) Blending region representation (b) Region of force equilibrium.

10.3 RECONSTRUCTION AND EVOLUTION

Blending is incorporated into the previously developed physics-based framework (see Chapters 2 and 3) to accomplish the shape reconstruction. This requires the collection of parameters of the model into generalized coordinates \mathbf{q}_s :

$$\mathbf{q}_s = (\mathbf{q}_{s_1}^T, \mathbf{q}_{s_2}^T, \mathbf{q}_b^T, \mathbf{q}_T^T)^T. \quad (10.2)$$

\mathbf{q}_{s_1} and \mathbf{q}_{s_2} are the parameters of each of the component shapes, \mathbf{q}_b are the parameters that specify α , \mathcal{B} and \mathcal{R} , and \mathbf{q}_T are parameters of global parameterized deformations (such as bending), which can be added on top of a blended shape. For a hierarchical blended shape, each of the component shapes \mathbf{q}_{s_1} and \mathbf{q}_{s_2} can either be a primitive, or a child blended shape which will have the form given by (10.2).

When fitting the model to data, the goal of shape reconstruction is to recover \mathbf{q} . The application of forces to the surface of the model deforms it into the shape represented by the data (see Chapter 5). The dynamics framework requires the computation of $\mathbf{J}_s = \partial s / \partial \mathbf{q}_s$, the Jacobian for the shape s . \mathbf{J}_s “converts” the applied forces into generalized forces, which deform the global shape. The addition of blending alters the computation of \mathbf{J}_s . In particular, from (10.1):

$$\mathbf{J}_s = \left[\begin{array}{c|c|c} \alpha(\mathbf{u})\mathbf{J}_{s_1} & \left(1 - \alpha(\mathcal{B}(\mathbf{u}))\right)\mathcal{R}_{\text{rot}}(\mathbf{J}_{s_2}) & \mathbf{J}_b \end{array} \right], \quad (10.3)$$

where $\mathbf{J}_{s_1} = \partial s_1 / \partial \mathbf{q}_{s_1}$ is the Jacobian for s_1 , $\mathbf{J}_{s_2} = \partial s_2 / \partial \mathbf{q}_{s_2}$ is the Jacobian for s_2 , and $\mathbf{J}_b = \partial s / \partial \mathbf{q}_b$ is the Jacobian for the parameters of the blending function, and reflects how the global shape s changes with respect to the blending function parameters \mathbf{q}_b . \mathcal{R}_{rot} is the rotational component of \mathcal{R} . From (10.1), we find that

$$\mathbf{J}_b = s_1(\mathbf{u}) \frac{\partial \alpha(\mathbf{u})}{\partial \mathbf{q}_b} - \mathcal{R}\left(s_2(\mathcal{B}(\mathbf{u}))\right) \frac{\partial \alpha(\mathcal{B}(\mathbf{u}))}{\partial \mathbf{q}_b} \quad (10.4)$$

Given the implementation of α presented in section 10.2.3, we can construct a piecewise derivative of each of the blending parameters in \mathbf{q}_b .

10.3.1 Model evolution

Our initial model in our estimation system is always a sphere (represented using a single superellipsoid). During reconstruction, blending is automatically performed based on forces from the data points and the current shape. The addition of each blend splits the model, requiring the specification of blending regions. Repeated application of blending causes an *evolution* from a sphere to the final model.

For a given shape model and data, the estimation process eventually reaches a steady state—all applied forces equilibrate or vanish. Forces equilibrate due to

the inability of the model to reach a shape that represents the data. Fig. 10.8(b) shows a hypothetical situation where forces have equilibrated (the ellipsoid cannot produce the tapered shape represented by this data).

A region growing algorithm (where a region consists of nodes, and connectivity is defined by the mesh) is applied to the mesh to isolate regions on the shape with equilibrating forces, such as the bold region in Fig. 10.8(b). Using equilibrated forces is similar to finding residual data points [165]. Once such a region is located, the current model is automatically *split*, and is replaced with a blended model where s_1 and s_2 are initially equal. The boundary of both blending regions is initialized to the region of equilibration, and \mathcal{R} is initialized to the identity transformation.

This operation changes only the representation of the model, *not* the geometry of the shape. The benefit of performing this representational split is that the shape can now deform to better capture the deviation in the data. According to (10.3), the blending function has the desirable effect of localizing the effect of a force to the appropriate shape component. The forces that were previously pulling on s_1 in the blending region now cause the deformation of s_2 .

The addition of a hole to the model requires the proximity detection of different parts of the shape. Surface self-intersection detection methods such as [170] can be used. Once such a proximity is detected, a blended shape is automatically constructed which combines the previous model with a hole that is aligned to join the two areas in proximity.

Holes do not necessarily form between two locations on the same component shape. If the hole is deep, it is very common for the hole to form between two indentations formed due to the presence of the hole in the data, as seen in the experiment in Figs. 10.13(f)-(g). The blending regions for s_1 are computed from the results of the proximity detection. A superquadric toroid is used for s_2 using the blending region from Fig. 10.6(b). \mathcal{R} must also be specified with the hole aligned with the proximate regions on s_1 . This is performed by a separate dynamic fitting process where the positions of boundary regions of s_1 pull on a model containing the boundary regions of s_2 . A bending deformation can be added to the hole to allow bent holes.

Fig. 10.9 shows the transformation from a shape model shown in (a) (in this case, a sphere) to include a hole (d). Fig. 10.9(b) shows the model which is constructed when the hole blend is initially created. The torus hole is not expressed in the resulting shape since $h = 1$, and the topology of the hole mesh is that of a sphere [60]. During the fitting process, if a hole is present in the data, data forces change h to have the value 0, shown in Fig. 10.9(c). When $h = 0$, the topology of the hole mesh becomes that of a torus, and the result is a torus with a closed hole, which can now open as shown in Fig. 10.9(d). When $h > 0$, the torus hole is not permitted to open, since doing so produces an open surface.

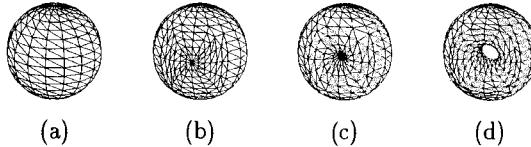


Figure 10.9 Hole evolution.

It is by these two processes—splitting and hole addition, which produce the evolution of the model from a sphere to the final model when combined with the physics-based deformation. Neither of these processes cause abrupt changes in the geometry of the model. It is interesting to note that the number of parameters required for a blended shape exceeds that of a CSG model. Blending requires the explicit specification of blending region boundaries, while CSG uses object interpenetration. The benefit of this specification is seen during estimation, where a prior segmentation of data is not necessary. Instead, the evolution methods described above provide a good initial guess at a part segmentation. The physics-based estimation technique then finds a set of parameters (including those which specify the blending regions) which better represent the data, to form the parts of the model.

10.4 EXPERIMENTS

Data	Source	Points	MSE	Parm	Iter
block/cylinder	MSU (column2)	1034	1.4%	29	190
mug	MSU (cup1)	1207	2.8%	68	372
two holed	CAD generated	893	1.0%	83	511

MSU: data from the Michigan State University PRIP database

Figure 10.10 Experiment data and statistics.

In the following three fitting experiments, we show the results of using our shape estimation system using superquadric ellipsoids and toroids. Fig. 10.10 shows information on each of the experiments including the source and size of the set, the number of parameters in the final model, the number of iterations taken by the solver, and the resulting mean squared error (MSE). Each iteration takes approximately 1/2 second on a 100 MHz SGI R4000.

In each of the three examples, the data set and initial model of a sphere is shown. The first fit shown is a rough fit—by fitting the axis parameters of the initial model. After the final result, a symbolic representation of the object is shown (with manually generated names).

Fig. 10.11(d) shows the first adaptation of the shape model, where a blending region was added (in gray). The blended region added is shown in (e). After

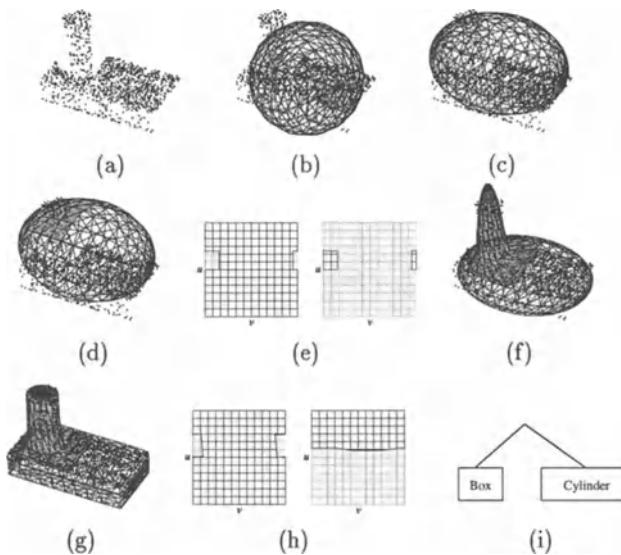


Figure 10.11 Fitting of a block and cylinder.

fitting the new shape model, the shape in (f) is formed, followed by the final fit in (g), and the final blending regions in (h).

The fitting of a mug is shown in Fig. 10.12. The blending region which corresponds to the mug handle forms in (d), and the rough fitting of the handle is shown in (e) with the corresponding blending region in (f). After further fitting of the handle in (g), a hole blend is added in (h) with blending region (i). After rough fitting and hole opening (j), the final fit is obtained in (k).

Finally, the fitting of an object with two holes is shown in Fig. 10.13. An indentation blending region for one of the holes is formed in (d), and after initial fitting it becomes indented (e). After several more blends, the shape evolves two additional indentations (f). After further fitting, a hole has formed in (g). The second hole has formed in (h), and the final fitted result is shown in (i).

10.4.1 Examination of stability

We first examine the effect of using different range-scanner views of the same object. In this case, three different views of a two-holed object (synthesized data) are examined. The data in Fig. 10.14(a) and (d) show views where the hole interiors are visible. The resulting estimated blended shapes, shown in (b) and (e), respectively, appear similar in shape, while clearly the hierarchies formed,

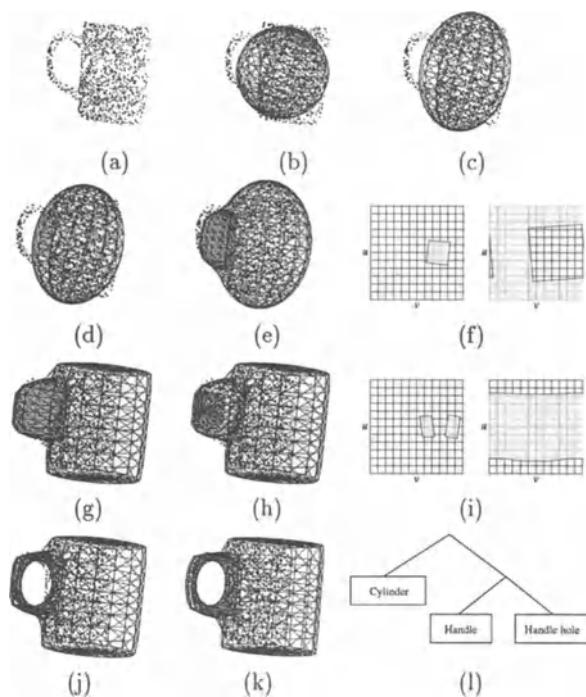


Figure 10.12 Fitting of a mug.

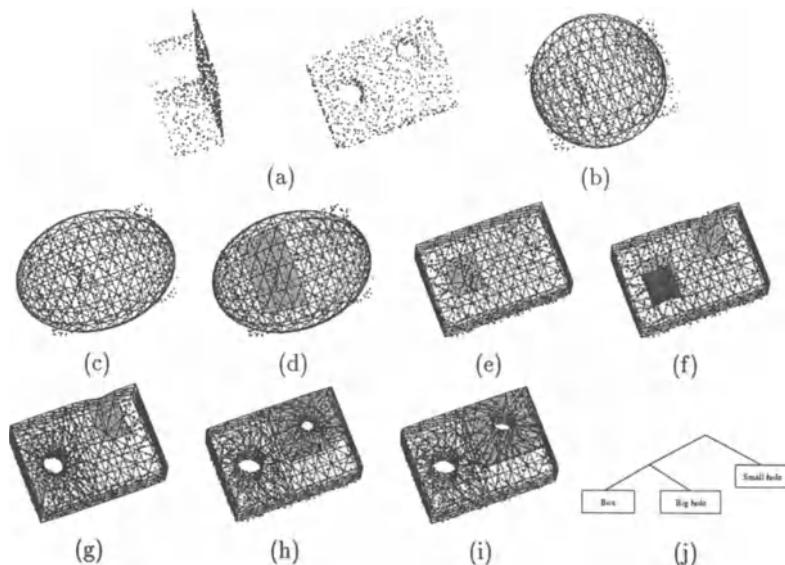


Figure 10.13 Fitting of a two holed object.

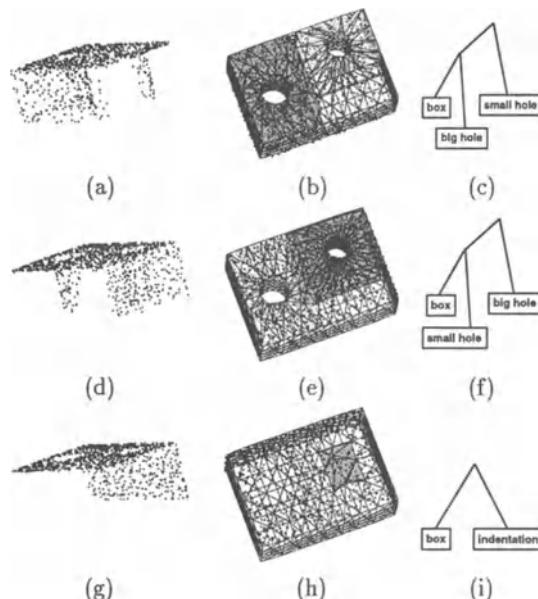


Figure 10.14 Various 3D range scanner views of two holed object.

shown in (c) and (f) are qualitatively different (the order of the big and small holes in the trees are reversed). There are algorithms to test the equivalence of the two representations should it be necessary, depending on the application, to compare the two representations. This is due to the different order of decisions made in the model evolution (see section 10.3.1) during estimation. The next example, shown in Fig. 10.14(g), is a view which contains very little interior hole data. As a result, only an indentation forms, as seen in (h) and (i).

Figs. 10.15(a),(d), and (g) show data for a box shaped object with a hole, and a varying sized cylindrical attachment. As seen by the fitting results in (b),(e) and (h), the resulting shapes fit the data well, yet their hierarchies (c),(f) and (i) can sometimes vary as in the earlier example.

When using a global description of shape, one would expect representational changes to occur (such as the tree ordering differences seen here). For our method, representational changes can occur when the size of extracted parts of the object change size or relative position.

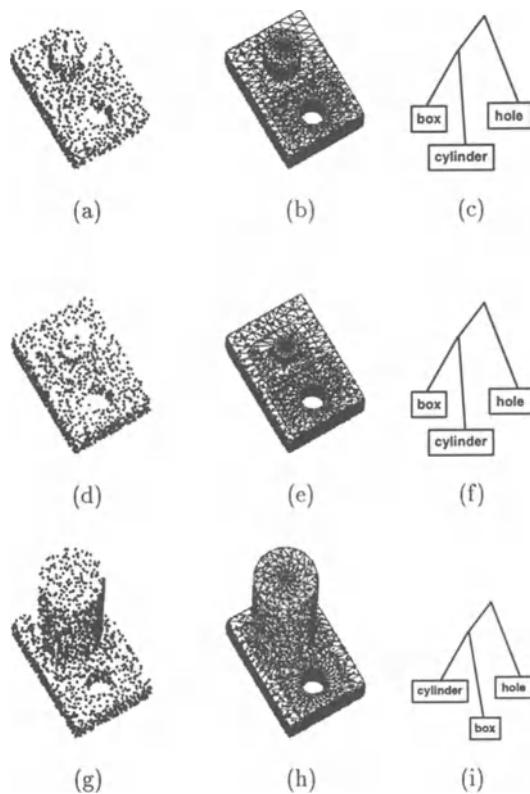


Figure 10.15 Various cylinder sizes in box/cylinder/hole object.

INTEGRATION OF QUALITATIVE SEGMENTATION AND PBM METHODS

The performance of most physics-based shape estimation techniques depends on the accuracy of the initial segmentation and the initial placement of the model given the segmented data [275, 216, 229]. In order to address the above limitations, a new approach to shape recovery from 2D images which integrates qualitative shape recovery [71] and quantitative physics-based estimation techniques [181] has been recently proposed by Dickinson and Metaxas [183]. Through qualitative shape recovery based on aspect matching, we extract the qualitative shapes of objects that are composed of primitives which belong to a fixed vocabulary of shapes. Furthermore, the qualitative shape recovery handles occlusion through a hierarchical aspect representation. We then use the qualitative correspondence of edge segments in the images and related contours on the models to provide strong fitting constraints to our physics-based estimation technique presented in the previous chapters. The “tokens” for matching features in the images are groups of edge segments (curved or straight) corresponding to an aspect of a part (as opposed to points or lines). Since the qualitative and quantitative shape recovery processes are both model-based, the approach is robust to noise and occlusion.

In this chapter, we present the integration of qualitative and quantitative shape and motion recovery from 2-D images. In particular, we use knowledge of both a primitive’s qualitative shape and its orientation (encoded by its aspect) to provide strong constraints in fitting a deformable model to the contour data. Since the qualitative primitive recovery technique supports primitive occlusion through a hierarchical aspect representation, it can selectively pass to the model fitting stage only those contours belonging to the object. In addition, the correspondence between image faces and model surfaces encoded in the recovered qualitative primitive can be exploited to provide strong constraints on the initial placement of the deformable model. Furthermore, this correspondence allows us to extend the previously presented PBM methodology which is limited to orthographic projection of occluding boundaries to the case of more general objects with internal surface discontinuities under general orthographic and perspective projections. In addition, we present algorithms for shape estimation from multiple views , where stereo is the simpler case. This estimation process can handle complex scenarios where significantly different or incomplete (but con-

sistent) sets of edge segments from the same object are extracted from multiple images taken from cameras with arbitrary relative orientations. Moreover, to be able to model and recover a larger variety of complex objects, we propose a new definition for the global bending deformation, the parameters of which can be decoupled during recovery. We also present a new efficient algorithm which approximately tessellates the model surface uniformly in the 3D Euclidean space, allowing a more robust recovery as the model deforms to fit the data. Finally, we present tracking examples based on the use of an extended Kalman filter.

11.1 RELATED WORK

Recently, several researchers have proposed various segmentation techniques to partition image or range data, in order to automate the process of fitting volumetric primitives to the data. Most of those approaches are applied to range data only [263, 107], while Pentland [216] describes a two-stage algorithm to fit superquadrics to image data. In the first stage, he segments the image using a filtering operation to produce a large set of potential object “parts”, followed by a quadratic optimization procedure that searches among these part hypotheses to produce a maximum likelihood estimate of the image’s part structure. In the second stage, he fits superquadrics to the segmented data using a least squares algorithm. This approach is applicable to the case of occluding boundary data under simple orthographic projection, as is true of earlier work of Terzopoulos et al. [275], Terzopoulos and Metaxas [278], and Pentland and Sclaroff [218], which address only the problem of model fitting. The fundamental difference between the method we present and the above approaches is that we use a qualitative segmentation of the image to provide sufficient constraints to our deformable model fitting procedure.

Various 3D model-based approaches to object tracking have been studied [77, 167, 98]. These techniques require exact geometric specification of the objects. The use of 3D deformable models has been also explored in [122, 217, 181]. (The 2D problem has received similar attention [137, 36].) However, the focus of these approaches have been mainly on the use of range data. In this chapter, we deal with the segmentation, initialization and tracking of objects in 2D intensity images within a unified framework [47, 46]. The physics-based model updating (tracking) process requires only a gradient computation in each frame which is computationally efficient. Assuming relatively small object motion between frames, local forces derived from stereo images are sufficient to update the positions, orientations, and shapes of the models in 3D. Another advantage of our technique is that we do not need to perform costly feature correspondences during 3D tracking. Finally, we use an extended Kalman filter to better predict the object’s motion and possible edge occlusion and disocclusion. The occurrence of such situations may be due to changes of an object’s aspect and/or due to motions of other independently moving objects and the associated occlusion effects. Based on our model-based approach, we can handle these situations by predicting their occurrence. We can determine which part of an object will be

occluded and suppress its contribution to the net forces applied to the model. Furthermore, in case of severe or unexpected object occlusion, which is detected through abrupt changes in the force field, we use a feedback mechanism to trigger the qualitative model extraction and quantitative shape recovery.

11.2 QUALITATIVE MODEL EXTRACTION

A framework for integrating qualitative model extraction techniques and quantitative deformable fitting techniques has been proposed [183, 74]. In the following we present briefly, the qualitative shape modeling and matching technique described in [70, 71, 72, 73].

11.2.1 Object-Centered Models

Given a database of object models representing the domain of a qualitative recognition task, we seek a set of three-dimensional volumetric primitives that, when assembled together, can be used to construct the object models. Many 3-D object recognition systems have successfully employed 3-D volumetric primitives to construct objects. Commonly used classes of volumetric primitives include polyhedra (e.g., Lowe [167]), generalized cylinders (e.g., Brooks [40]), and superquadrics (e.g., Pentland [214]). Whichever set of volumetric modeling primitives is chosen, they will be mapped to a set of viewer-centered aspects.

To demonstrate the approach of qualitative object recognition, we have selected an object representation similar to that used by Biederman [30], in which the Cartesian product of contrastive shape properties gives rise to a set of volumetric primitives called geons. For our investigation, we have chosen three properties including cross-section shape, axis shape, and cross-section size variation (Dickinson et al. [70]). The values of these properties give rise to a set of ten primitives (a subset of Biederman's geons), modeled based on superquadrics and illustrated in Fig. 11.1(a). To construct objects, the primitives are attached to one another with the restriction that any junction of two primitives involves exactly one distinct surface from each primitive.

11.2.2 Viewer-Centered Models

To recover the volumetric primitives from an image, we need some way of modeling their appearance in the image. Traditional aspect graph representations of 3-D objects model an entire object with a set of aspects, each defining a topologically distinct view of an object in terms of its visible surfaces [146]. Dickinson's approach differs in that it uses aspects to represent a (typically small) set of volumetric primitives from which each object in our database is

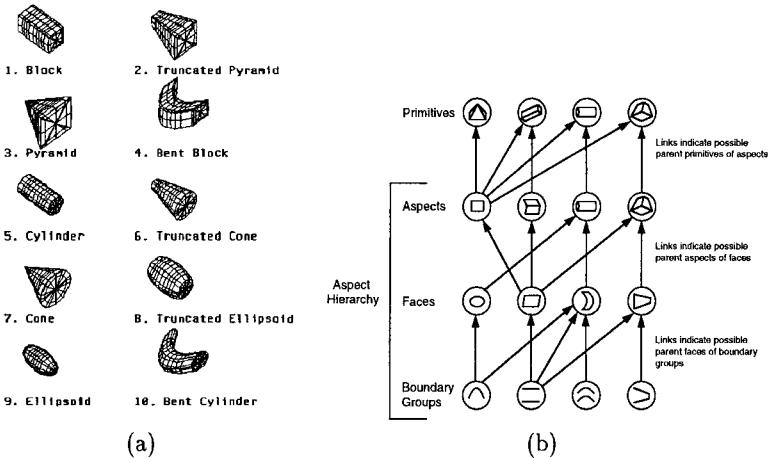


Figure 11.1 (a) The Ten Modeling Primitives, (b) The Aspect Hierarchy.

constructed, rather than representing an entire object directly. Consequently, the goal is to use aspects to recover the 3-D primitives that make up the object in order to carry out a recognition-by-parts procedure, rather than attempting to use aspects to recognize entire objects. The advantage of this approach is that since the number of qualitatively different primitives is generally small, the number of possible aspects is limited and, more important, *independent* of the number of objects in the database. The disadvantage is that if a primitive is occluded from a given 3-D viewpoint, its projected aspect in the image will also be occluded. Thus we must accommodate the matching of occluded aspects, which we accomplish by use of a hierarchical representation we call the *aspect hierarchy*.

The aspect hierarchy consists of three levels, consisting of the set of *aspects* that model the chosen primitives, the set of component *faces* of the aspects, and the set of *boundary groups* representing all subsets of contours bounding the faces. Fig. 11.1(b) illustrates a portion of the aspect hierarchy. The ambiguous mappings between the levels of the aspect hierarchy are captured in a set of conditional probabilities (Dickinson et al. [70, 71]), mapping boundary groups to faces, faces to aspects, and aspects to primitives. These conditional probabilities result from a statistical analysis of a set of images approximating the set of *all* views of *all* the primitives.

11.2.3 Qualitative Shape Recovery

Qualitative primitive recovery consists of three steps, resulting in a graph representation of the image in which nodes represent recovered 3-D primitives, and arcs represent hypothesized connections between the primitives; details of the

complete recovery process, including algorithms to handle various segmentation errors, can be found in Dickinson et al. [71]. In the following, we briefly review the three steps to recovering qualitative shape.

Step 1: Face Recovery

The first step to recovering a set of faces is a region segmentation of the input image. We begin by applying Saint-Marc, Chen, and Medioni's edge-preserving adaptive smoothing filter to the image [245], followed by a morphological gradient operator [157]. A hysteresis thresholding operation is then applied to produce a binary image from which a set of connected components is extracted. Edge regions are then burned away, resulting in a *region topology graph* in which nodes represent regions and arcs specify region adjacencies.

From a region topology graph, each region is characterized according to the qualitative shapes of its bounding contours. First, the bounding contour of each region is partitioned at curvature extrema using Saint-Marc, Chen, and Medioni's adaptive smoothing curve partitioning technique [245]. Next, each bounding contour is classified as straight, convex, or concave, by comparing the contour to a fitted line. Finally, each pair of bounding contours is checked for cotermination, parallelism, or symmetry. The result is a *region boundary graph* representation for a region in which nodes represent bounding contours, and arcs represent pairwise nonaccidental relations between the contours.

Face labeling consists of matching a region boundary graph to the graphs representing the model faces in the aspect hierarchy. Region boundary graphs that exactly match a face in the aspect hierarchy will be given a single label with probability 1.0. For region boundary graphs that do not match due to occlusion, segmentation errors, or errors in computing their graphs, we descend to an analysis at the boundary group level and match subgraphs of the region boundary graph to the graphs representing the boundary groups in the aspect hierarchy. Each subgraph that matches a boundary group generates a set of possible face interpretations (labels), each with a corresponding probability. The result is a *face topology graph* in which each node contains a set of face labels (sorted by decreasing order of probability) associated with a given region.

Step 2: Aspect Recovery

In an unexpected object recognition domain in which there is no a priori knowledge of scene content, we can formulate the problem of extracting aspects as follows: Given a face topology graph with a set of face hypotheses (labels) at each node (region), find an *aspect covering* of the face topology graph using aspects in the aspect hierarchy, such that no region is left uncovered and each region is covered by only one aspect. Or, more formally: Given an input face topology graph, FTG , partition the nodes (regions) of FTG into disjoint sets, $S_1, S_2, S_3, \dots, S_k$, such that the graph induced by each set, S_i , is isomor-

phic to the graph representing some aspect, A_j , from a fixed set of aspects, $A_1, A_2, A_3, \dots, A_n$.

There is no known polynomial time algorithm to solve this problem (see [71] for a discussion on the problem's computational complexity); however, the conditional probability matrices embedded in the aspect hierarchy provide a powerful constraint that can make the problem tractable. For each face hypothesis (for a given region), we can use the face to aspect mapping to generate the possible *aspect hypotheses* that might encompass that face¹. At each face, we collect all the aspect hypotheses (corresponding to all face hypotheses) and rank them in decreasing order of probability.²

We can now reformulate the bottom-up aspect recovery problem as a search through the space of aspect labelings of the regions (nodes) in the face topology graph. In other words, we wish to choose one aspect hypothesis from the list at each node, such that the instantiated aspects completely cover the face topology graph. For our search through the possible aspect labelings of the face topology graph, we employ Algorithm A (Nilsson [202]) with a heuristic designed to meet three objectives. First, we favor selections of aspects instantiated from higher probability aspect hypotheses. Second, we favor selections whose aspects have fewer occluded faces, since we are more confident of their labels. Finally, we favor those aspects covering more faces in the image; we seek the minimal aspect covering of the face topology graph. Since there may be many labelings which satisfy this constraint, and since we cannot guarantee that a given aspect covering represents a correct interpretation of the scene, we must be able to enumerate, in decreasing order of likelihood, all aspect coverings until the objects in the scene are recognized.

In an expected object recognition domain in which we are searching for a particular object or part, we use the aspect hierarchy as an attention mechanism to focus the search for an aspect at appropriate regions in the image. Moving down the aspect hierarchy, target objects map to target volumes which, in turn, map to target aspect predictions which, in turn, map to target face predictions. Verification of the target aspect prediction occurs at those faces in the face topology graph whose labels match the target face prediction. The scores of the matching faces are used to order the recovery process which attends first to high-quality faces. This attention mechanism has been used to drive an active recognition system which moves the cameras to obtain either a more likely or unambiguous view of an object's part [73].

Step 3: Primitive Recovery

In the expected object recognition approach described above, primitive recovery consists of mapping the recovered aspect directly to the target primitive

¹The probability of an aspect hypothesis is the product of the face to aspect mapping and the probability of the face hypothesis from which it was inferred.

²For a detailed discussion of aspect instantiation and how occluded aspects are instantiated, see [71].

prediction. Primitive recovery for the unexpected object recognition case is more complex. From an *aspect covering* of the regions in the image, a set of primitive labels and their corresponding probabilities is inferred (using the aspect hierarchy) from each aspect. Primitive recovery is formulated as a search through the space of primitive labelings of the aspects in the aspect covering, guided by a heuristic based on the probabilities of the primitive labels. Each solution, or *primitive covering*, found by the search is a valid primitive interpretation of the input image. Encoded in each recovered primitive is the aspect in which it is viewed; the aspect, in turn, encodes the faces that were used in instantiating the aspect, while each face specifies those contours in the image used to instantiate the face.

Each recovered qualitative part defines: (i) the relevant non-occluded contour data belonging to the part, (ii) a mapping between the image faces in their projected aspects and the 3D surfaces on the quantitative models, and (iii) a qualitative orientation (that the aspect encodes) which is exploited during model fitting (see [183] for details.). The mapping of certain edge segments corresponding to the *occluding contour* on the model surfaces in (ii) needs to be updated continuously during the fitting process [183].

For model extraction from stereo images, the qualitative shape recovery process is independently applied to the left and right images. The correspondence problem then consists of matching qualitative primitive descriptions in the two images. A pair of primitives in two different images are considered a match if: (i) the primitives have the same label (shape constraint), (ii) their aspects have the same label (orientation constraint), and (iii) for each pair of corresponding faces in their aspects, there exists an epipolar line (not restricted only to parallel geometry) such that both faces intersect this line. For model extraction from multiple views, the process is similar except that correspondences have to be established for each pair of images.

11.3 QUANTITATIVE SHAPE AND MOTION RECOVERY

Based on the constraints provided by the qualitative shapes, we extend the previously presented PBM framework to allow object pose and shape estimation from single/multiple cameras in case of orthographic and perspective projections. In the case of stereo, the two cameras have arbitrary relative orientations. To recover complex objects, we generalize the definition for the global bending deformation. We also present an algorithm for model discretization which evenly tessellates the model surface and significantly improves the numerical behavior of our algorithm.

To illustrate our approach, we use a deformable superquadric ellipsoid with global tapering and bending deformation as a reference shape. In the following

section, we define a bending deformation that ensures constant curvature along the major axis of bending. This is a generalization of the previous definition in Chapter 2.

11.3.1 Constant Curvature Bending

We define the bending transformation $s = \mathbf{T}_b(\mathbf{e}; b_0)$ about the y -axis of a primitive \mathbf{e} as

$$\begin{aligned} s_x &= \cos(b_0(e_z - e_{z_0})) \cdot (e_x - \frac{1}{b_0}) + \frac{1}{b_0}, \\ s_y &= e_y, \\ s_z &= -\sin(b_0(e_z - e_{z_0})) \cdot (e_x - \frac{1}{b_0}) + e_{z_0}, \end{aligned} \quad (11.1)$$

where b_0 is the radius of curvature, and e_{z_0} denotes the offset of the center of bending along the z -axis.

We generalize this deformation to deal with bending about an arbitrary axis in the xy -plane by introducing the following additional transformation

$$s = \mathbf{T}'_b(\mathbf{e}; b_0, b_1) = \mathbf{J}_b^{-1}(\mathbf{T}_b(\mathbf{J}_b \cdot \mathbf{e})), \quad (11.2)$$

where \mathbf{T}_b is defined above and the matrix \mathbf{J}_b is defined as

$$\mathbf{J}_b = \begin{bmatrix} \cos(b_1) & -\sin(b_1) & 0 \\ \sin(b_1) & \cos(b_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (11.3)$$

Here, b_1 is the angle between the y -axis and the axis about which bending takes place. The introduction of \mathbf{J}_b decouples the recovery of the rotation and bending parameters $\mathbf{q}_s = (b_0, b_1)^T$ during model fitting (which will be demonstrated later).

11.3.2 Model Fitting to Multiple Images

In the approach described here, each deformable model deforms based on forces exerted from the qualitatively segmented corresponding edges from single/multiple images. During the fitting process with stereo/multiple images, the cameras can have arbitrary relative orientation.

Jacobian Computation for Generalized Perspective Projection

To allow shape and pose estimation in a world coordinate frame from images taken from a camera with a different frame of reference, the Jacobian matrix \mathbf{L} defined in Chapter 2 needs to be modified appropriately.

Let $\mathbf{x} = (x, y, z)^T$ denote the location of a point j w.r.t the world coordinate frame. Then we can write

$$\mathbf{x} = \mathbf{c}_c + \mathbf{R}_c \mathbf{x}_c, \quad (11.4)$$

where \mathbf{c}_c and \mathbf{R}_c are respectively the translation and rotation of the camera frame w.r.t. the world coordinate frame, and $\mathbf{x}_c = (x_c, y_c, z_c)^T$ is the position of the point j w.r.t to the camera coordinate frame.

Under perspective projection, the point \mathbf{x}_c projects into an image point $\mathbf{x}_p = (x_p, y_p)^T$ according to

$$x_p = \frac{x_c}{z_c} f, \quad y_p = \frac{y_c}{z_c} f, \quad (11.5)$$

where f is the focal length of the camera. By taking the time derivative of (11.5) we get $\dot{\mathbf{x}}_p = \mathbf{H} \dot{\mathbf{x}}_c$, where

$$\mathbf{H} = \begin{bmatrix} f/z_c & 0 & -x_c/z_c^2 f \\ 0 & f/z_c & -y_c/z_c^2 f \end{bmatrix}. \quad (11.6)$$

Based on (3.4), (11.4) and (11.6), we obtain

$$\dot{\mathbf{x}}_p = \mathbf{H}(\mathbf{R}_c^{-1} \dot{\mathbf{x}}) = \mathbf{H}\mathbf{R}_c^{-1}(\mathbf{L}\dot{\mathbf{q}}) = \mathbf{L}_p \dot{\mathbf{q}}. \quad (11.7)$$

By replacing the Jacobian matrix \mathbf{L} in (3.47) by $\mathbf{L}_p = \mathbf{H}\mathbf{R}_c^{-1}\mathbf{L}$, two dimensional image forces \mathbf{f} can be appropriately converted into generalized forces \mathbf{f}_q measured in the world coordinate frame.

Model Discretization

To ensure that the forces are evenly applied to the 3D model surface during model fitting, we need to have a uniform discretization of the model surface in the 3D Euclidean space. Here we present a very fast technique to discretize a superquadric surface dynamically that maintains approximate uniform tessellation. Our aim is to counteract the effect of the squareness parameters ϵ_1 and ϵ_2 which causes the nonuniform tessellation of the model surface when their values approach zero. Although the material coordinates $\mathbf{u} = (u, v)$ of the superquadric surface are defined for $-\frac{\pi}{2} < u < \frac{\pi}{2}$ and $0 < v < 2\pi$, it suffices due to symmetry to illustrate our method for the uniform tessellation of the first quadrant of the xy plane, with corresponding material coordinate $v \in [0, \frac{1}{2}\pi]$.

The points generated by uniform sampling in the material coordinate space are quite uniformly distributed on the model surface when $\epsilon_2 = 1$. By projecting these model points along the x and y axes onto a model surface where $\epsilon_2 \ll 1$, their positions remain relatively “spread out” for $0 < v < v_{cx}$ and $v_{cy} < v < \frac{\pi}{2}$, respectively. The values of v_{cx} and v_{cy} are obtained by projecting along the axes the point on the model surface in the xy plane where the normal vector to the surface is at 45° from each axis. Mathematically, these are given by $v_{cx} = \sin^{-1}(\sin^{\epsilon_2} v_c)$, $v_{cy} = \cos^{-1}(\cos^{\epsilon_2} v_c)$, where $v_c = \tan^{-1}[(\frac{a_2}{a_1})^{\frac{1}{\epsilon_2-2}}]$. These are illustrated in Fig. 11.2.

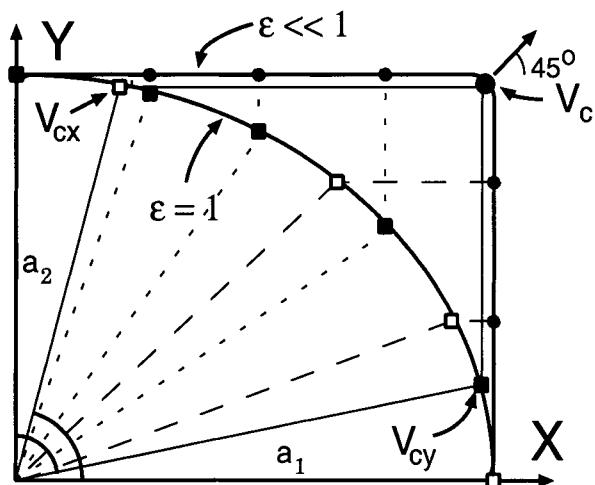


Figure 11.2 Illustration of the resampling scheme in the v -coordinate of the surface mesh.

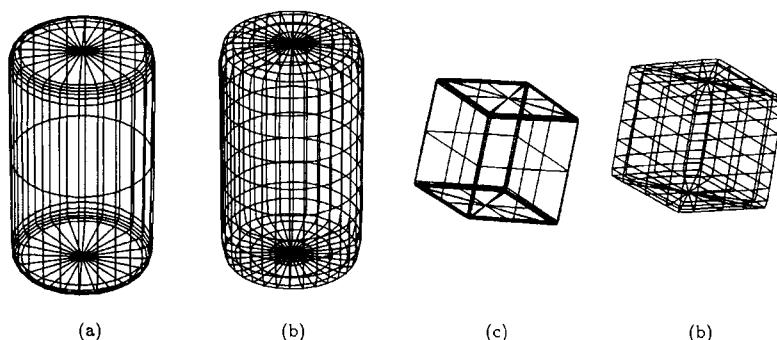


Figure 11.3 Retessellation of a cylinder (a) before (b) after; and a block (c) before (d) after.

Based on the above, instead of uniformly sampling v in its entire range $[0, \frac{1}{2}\pi]$, we sample v uniformly in each of the above two intervals and we transform the chosen values of v so that the x or y values of the superquadrics with $\epsilon_2 = 1$ and $\epsilon_2 \ll 1$ are equal. If v_k is the material coordinate chosen, the transformed value is given by:

$$v'_k = \begin{cases} \sin^{-1}(\sin^{\frac{1}{\epsilon_2}}(v_k)), & 0 < v_k < v_{cx} \\ \cos^{-1}(\cos^{\frac{1}{\epsilon_2}}(v_k)), & v_{cy} < v_k < \frac{\pi}{2} \end{cases}, \quad (11.8)$$

where these formulas result in approximately uniform tessellation in the first quadrant of the xy plane. Fig. 11.3 illustrates the improved tessellation algorithm.

Quantitative Multi-View Integration

Based on the above qualitative shape recovery, correspondences are established between edge segments in the images and the corresponding subsets of nodes on the model surface. If $e_{j,i}$ is the j th edge point in the i th image ($i = L$ or R for stereo case), and $\mathcal{M}_{j,i}$ is the subset of model nodes in correspondence with the edge segment that $e_{j,i}$ belongs to, then the 2D force exerted by that edge point on the projected model (based on a shortest distance criterion) is given by

$$\mathbf{f}_{j,i} = \beta \min_{k \in \mathcal{M}_{j,i}} (\mathbf{P}_i(\mathbf{R}_{c_i}^{-1}(\mathbf{x}_k - \mathbf{c}_{c_i})) - \mathbf{e}_{j,i}), \quad (11.9)$$

where \mathbf{x}_k is the position of k th model node, β controls the magnitude of the force and \mathbf{P}_i is the perspective projection operator w.r.t. the i th image.

The generalized force exerted on the model can be computed by replacing the integral in (3.47) by the following summation:

$$\mathbf{f}_q = \sum_j \mathbf{L}_{m_{j,1}}^T f_{j,1} + \dots + \sum_j \mathbf{L}_{m_{j,n}}^T f_{j,n}, \quad (11.10)$$

where $m_{j,i}$ is the model node at which forces are exerted by the $e_{j,i}$ and $\mathbf{L}_{m_{j,i}}$ is the Jacobian matrix evaluated at $m_{j,i}$. ³ This net force will appropriately deform, position and orient our model so as to recover the shape and pose of the underlying object.

Based on the above generalizations, we can now integrate possibly incomplete measurements from more than two cameras which are not necessary parallel. Note that we do no need to perform point correspondences between the left and right images, but only line and curve correspondences to the 3D model, which is more robust.

For static shape recovery, the fitting of the model to the data given the computed image forces is based on the use of (3.51).

³In case of occlusion, resulting in both occluded aspects and occluded faces, only data points belonging to the unoccluded portions (boundary groups) of the faces exert external forces on the models.

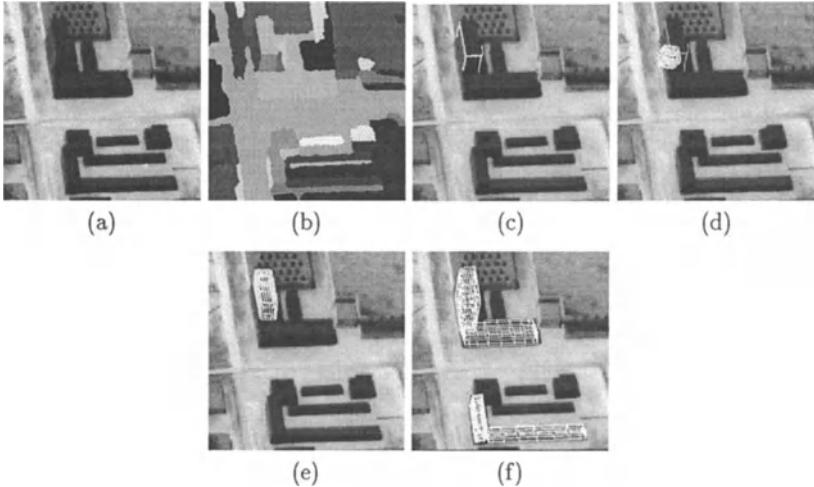


Figure 11.4 (a) Original image, (b) region extraction, (c) example of extracted primitive, (d-e) intermediate steps in fitting a deformable model to the building, (f) models fitted to other buildings present in the image.

11.3.3 Static Shape Estimation Examples

We show the results of our technique described above in a series of experiments on model recovery from single and stereo images as well as from multiple views. This is performed using Euler’s numerical integration. The Euler step used was 0.01 and our algorithm runs at interactive rates on a SGI Indigo 2 XZ. In all the experiments, for the recovery of qualitative shape we used the system OPTICA [71].

In the first experiment, we demonstrate the model extraction and fitting operations with an image of a site model with buildings in a cluttered background, as shown in Fig. 11.4(a)⁴. Fig. 11.4(b) shows the regions extracted from the qualitative analysis of Fig. 11.4(a), while Fig. 11.4(c) shows the extracted bounding contours of the taller building in the scene. Due to the inadequate intensity contrast in the images, the bounding contours of the building are not closed. However, three orthogonal sides have been recovered allowing the identification of its associated qualitative aspect. Subsequently, we can estimate the shape and pose of the building using the quantitative module. The fitting process and the associated deformation of the parallelepiped is shown in Figs. 11.4(d-e). Similarly, Fig. 11.4(f) shows the final fit to the building together with three more fitted buildings. The fitting algorithm works by first recovering the orientation of the building and then simultaneously refines its dimensions and the orientation.

⁴Courtesy of the RADIUS project.



Figure 11.5 An image of a lock.

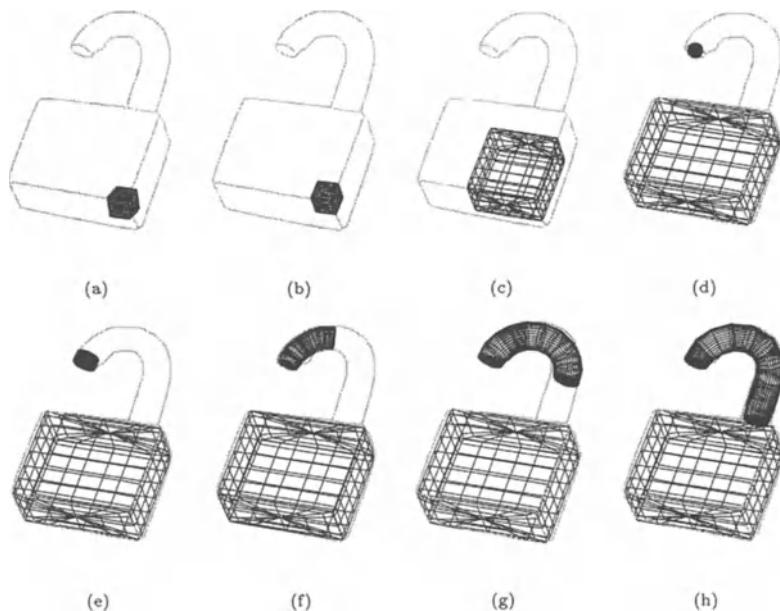


Figure 11.6 Two models fitted to the image of a lock.

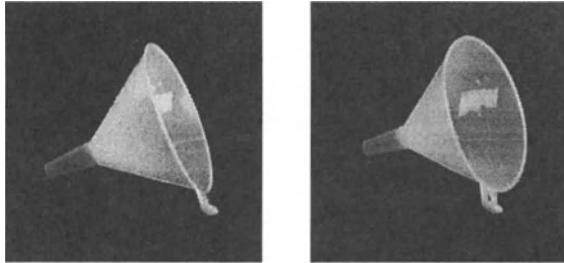


Figure 11.7 Initial pair of stereo images of a funnel.

The experiment shown in Fig. 11.6 demonstrates the use of our framework to estimate object shapes under perspective projection from the single image of a lock, shown in Fig. 11.5. Using OPTICA [71], a box-like primitive and a bent cylinder primitive are recovered. Figs. 11.6(a-d) and (e-h) show the quantitative fitting stages of the 2 models to the lower and upper parts of the lock, respectively. Note that the employment of the new tessellation method is especially important here, since the bending deformation is used in the recovery process. Moreover, with the proposed bending deformation, we can decouple the recovery of the corresponding bending parameters (as shown in Fig. 11.6(f-h)) from that of the rotational parameters (as shown in Fig. 11.6(d-e)).

In the third experiment, two stereo images of a funnel were taken from two viewpoints, where the relative displacements and orientations of the two cameras were known. Fig. 11.7 shows the original images of the funnel. The qualitative shape recovery process performed in both images decomposed the funnel into a tapered cylinder and a short cylinder. Fig. 11.8(a-h) shows the intermediate stages of the fitting process. Note that one of the edge segments has not been recovered because of low contrast in the left image. Each 3D model is fitted simultaneously to both images and Fig. 11.9 shows the fitted models in 3D.

In the last experiment, images of a scene with three objects were taken from three different viewpoints (see Fig. 11.10(a)). Three object parts were recovered with the qualitative shape recovery process as shown in Fig. 11.10(b). Some edge segments have not been recovered because of occlusion, poor contrast in the original images, or because they were not used to deduce the corresponding shape primitives. Fig. 11.10(c) shows the projections of the fitted models overlaid on the respective images. Finally, Fig. 11.11 shows the shape and pose of the recovered 3D models. This experiment demonstrates how information from incomplete and partially occluded image data from multiple views can be integrated in our model-based approach.

11.3.4 Tracking

After the part models have been initially extracted and fitted, any subsequent motions of the objects are tracked based on only local forces derived from the

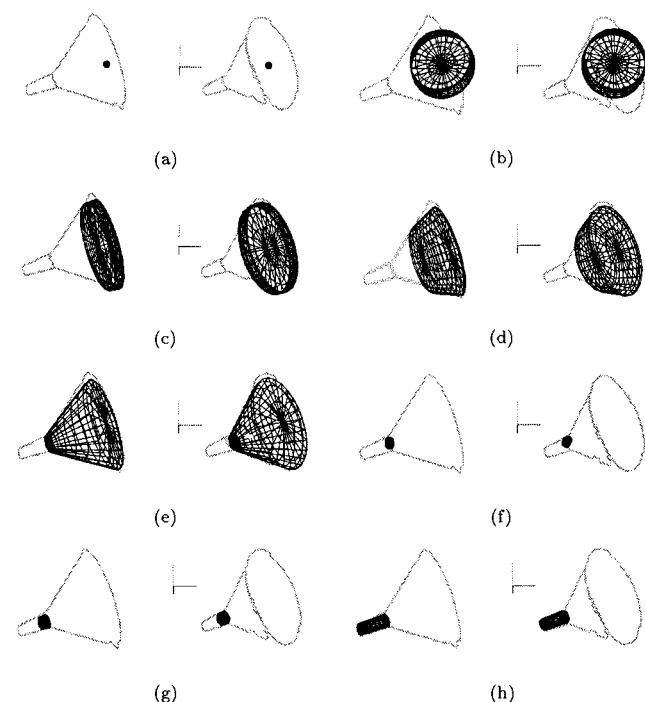


Figure 11.8 Fitting deformable part models to stereo images of a funnel (the wire frame models are the corresponding projections of the 3D model).

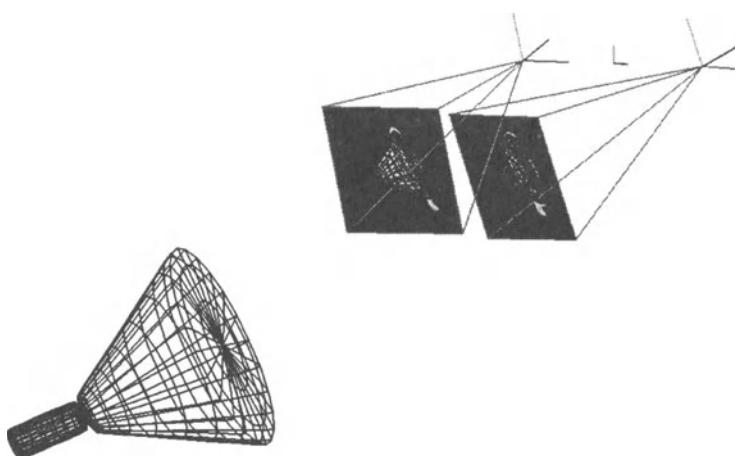


Figure 11.9 The 3D models fitted to the images of the funnel.

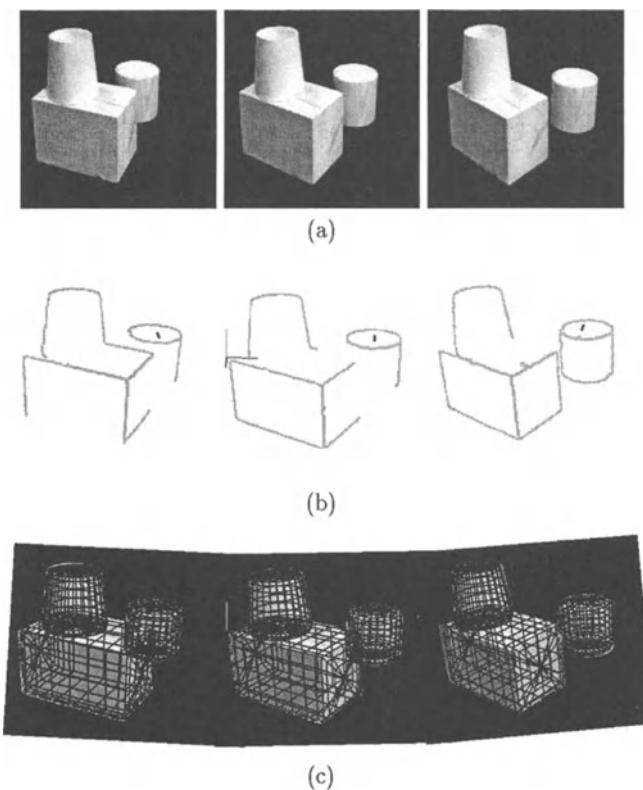


Figure 11.10 (a) original images of a scene with 3 objects from 3 different viewpoints, (b) relevant bounding contours of regions used to both qualitatively identify the shapes and constrain the fitting module, (c) projection of fitted 3D model overlaid on the original images.

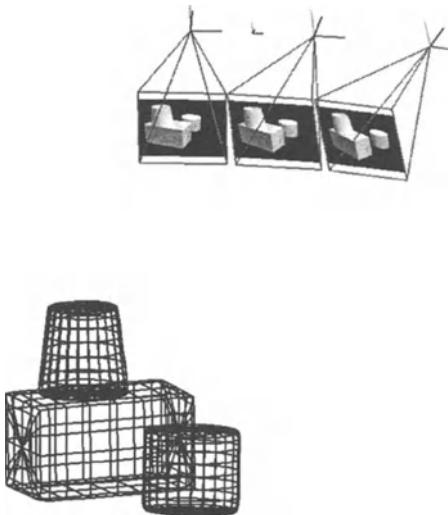


Figure 11.11 Recovered 3D shape and pose of objects in the scene from 3 different views.

images. The tracking process is fast because correspondences between model features and those in the images are not required. We describe in this section the details of this process.

Local Forces from Image Potentials

For each successive frame in the image sequence, we create an image potential such that the “valleys” of this potential correspond to the locations in the image where there are sharp changes in intensity or edge features. If we denote the intensity image by $I(x, y)$, the image potential can be computed as follows [278]:

$$\Pi(x, y) = -\beta |\nabla(G_\sigma * I)(x, y)|, \quad (11.11)$$

where σ determines the width of the Gaussian function G_σ , $*$ denotes the convolution operation, and β determines the “steepness” of the potential surface. This potential induces a 2D force field given by:

$$\mathbf{f}(x, y) = -\nabla\Pi(x, y). \quad (11.12)$$

The model’s degrees of freedom respond to the 2D force field through a process which first projects the model’s nodes into the image. As the projected nodes are attracted to the valleys of the potential surface, the model’s degrees of freedom are updated to reflect this motion. The mapping of 2D image forces to generalized forces acting on the model requires the derivation of a Jacobian matrix.

Forces from Stereo Images

By computing generalized forces in the world coordinate frame, the 2D image forces in a pair of stereo images can be simultaneously transformed into generalized forces \mathbf{f}_q measured in a common world coordinate frame. Measurements from two different views are sufficient to determine the scale and depth parameters of the model. If we denote the position of the j th active node on the model surface by \mathbf{x}_j , then the generalized force on the model can be computed by replacing the integral in (3.47) by the summation

$$\begin{aligned}\mathbf{f}_q = & \sum_{j \in \mathcal{A}_L} \mathbf{L}_{p_L}^T(\mathbf{f}_L(\mathbf{P}(\mathbf{R}_{c_L}^{-1}(\mathbf{x}_j - \mathbf{c}_{c_L})))) \\ & + \sum_{j \in \mathcal{A}_R} \mathbf{L}_{p_R}^T(\mathbf{f}_R(\mathbf{P}(\mathbf{R}_{c_R}^{-1}(\mathbf{x}_j - \mathbf{c}_{c_R})))),\end{aligned}\quad (11.13)$$

where \mathcal{A} is the set of indices of *active* nodes, the subset of model nodes on which image forces are to be exerted (see below for a detailed definition). Here the subscripts L and R denote dependence on the left and right images respectively, and \mathbf{P} describes the perspective projection equation.

Determining Active Model Nodes

When our measurements are 2D images, as opposed to 3D range data, only a subset of the nodes on the model surface are selected to respond to forces. From a given viewpoint, we can compute this active subset of model nodes based on the model's shape and orientation. In particular, a model node is made active if at least one of the following conditions is true:

1. it lies on the occluding contour of the model from that viewpoint (see also [278]),
2. the local surface curvature at the node is sufficiently large and the node is visible.

Instead of calculating analytically the positions of the active nodes on the model surface, we "loop" over all the nodes on the discretized model surface and check if one of the above two conditions is true. Condition 1 is true if

$$|\mathbf{i}_j \cdot \mathbf{n}_j| < \tau, \quad (11.14)$$

where \mathbf{n}_j is the unit normal at the j th model node, \mathbf{i}_j is the unit vector from the focal point to that node on the model, and τ is a small threshold. Condition 2 is true if

$$\exists k \in K_j \text{ s.t. } |\mathbf{n}_k \cdot \mathbf{n}_j| < \kappa \text{ & } \exists k \in K_j \text{ s.t. } \mathbf{n}_k \cdot \mathbf{i}_k < 0, \quad (11.15)$$

where K_j is a set of indices of the nodes adjacent to the j th nodes on the model surface. κ in (11.15) is a threshold ($\kappa = \cos(\pi/6)$ in our experiments) to determine if the angle between adjacent normal vectors is sufficiently large.

Tracking and Prediction

We incorporate a Kalman filter into our dynamic deformable model formulation by treating the differential equations of motion (3.49) as the system model. Based on the corresponding extended Kalman filter expressed by (8.1), we perform tracking by updating the model's generalized coordinates \mathbf{q} according to (8.3). Since we are measuring local forces directly from the computed image potential, the term $\mathbf{z} - \mathbf{h}(\hat{\mathbf{u}}_q)$ in (8.3) represents the 2D image forces. Using the above Kalman filter, we can predict at every step the expected location of the data in the next image frame, based on the magnitude of the estimated parameter derivatives $\dot{\mathbf{u}}_q$.

Self Occlusion and Disocclusion

As an object rotates in space, or as the viewpoint of the observer changes substantially, certain faces of the object will become occluded or disoccluded (*a visual event*). Hence, the bounding contours of such faces in the image will appear or disappear over time. By using the Kalman filter to predict the position and orientation of the model in the next time frame, we can quantitatively predict the occurrence of a visual event. In other words, we can determine by using our active node determination approach, which subset of the model nodes will be active in the next image frame, and suppress their contributions to the net forces applied to the model. For stereo images, this prediction can be performed independently to the left and right images. In this case, two sets of active model nodes are maintained at any particular moment.

Tracking Multiple Objects with Feedback

Our framework for object tracking can be extended to deal with multiple independently moving objects and multi-part objects. The complication here is that object parts may occlude one another in different ways. By tracking objects using stereo images, we can predict the 3D positions of the nodes on each model based on the current estimates of their respective model parameters and their rate of change. Active nodes on each model will be made "inactive" if they are predicted to be occluded by surfaces of other models. This visibility checking is performed for each model node against every surface of the other models in the scene. In practice, much of this checking can be avoided based on approximate estimates of each object's size and 3D location. We demonstrate in the next section that our approach can handle partial object occlusion.

There are two more cases of object occlusion for the case of multiple independently moving objects. The first case occurs when another moving object that was not previously present in the scene occludes the object being tracked. The second is due to an error from the qualitative segmentation system which did not detect an object during the model initialization step. By monitoring the exerted forces on the model's active nodes, i.e., checking to see if their magnitude exceeds a threshold or suddenly vanishes, a feedback mechanism is

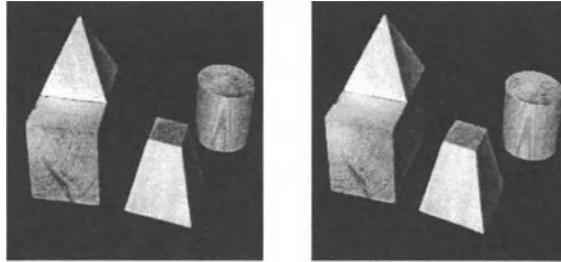


Figure 11.12 Initial pair of stereo images of a multi-object scene.

invoked which triggers the application of the qualitative segmentation system to resolve the ambiguity. After proper re-initialization of our models, we continue the quantitative tracking using local image forces based on our physics-based technique.

Furthermore, in order to track an object which maybe composed of a few parts of varying complexity, we do not necessarily need to track all the parts of the object. We demonstrate in one of our experiments that tracking can be achieved by focusing on only a portion of an object which can be modeled by our primitives.

11.3.5 Tracking Experiments

We demonstrate our approach with two tracking experiments involving stereo image sequences.

In the first experiment, we consider a sequence of stereo images of a scene containing multiple objects, including a two-part object. Fig. 11.12 shows the initial stereo images of the scene. The cameras are rotated around the scene at a constant rate. Fig. 11.13(a) shows the initialized models recovered using the same technique as before. Fig. 11.13(b) shows image potentials at an instant which the aspects of some parts have changed and some parts have become partially occluded. Each object is successfully tracked under these circumstances. Figs. 11.13(c-f) show the individual part models overlaid on the potentials in Fig. 11.13(b).

In the second experiment, we consider a sequence of stereo images of 3 independently moving toy cars. Fig. 11.14 shows three snapshots of the image sequence. Fig. 11.15 shows the corresponding image potentials. Fig. 11.16, Fig. 11.17 and Fig. 11.18 demonstrate the ability of the system to track the motion of each toy car. The tracking of each car is achieved by recovering only a portion of the car which is modeled by a single part model (the rest cannot be recovered by our primitives and is not necessary since it is a rigid object). Fig. 11.17 illustrates the ability of the system to handle partial occlusion as before.

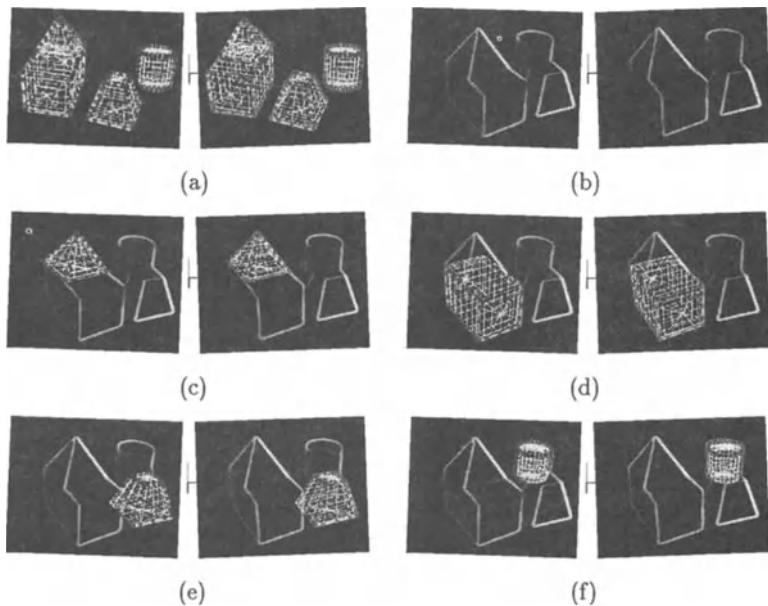


Figure 11.13 Tracking multiple objects in a sequence of stereo images (a) initialized models, (b) image potentials of an intermediate frame (both occlusion and visual events have occurred), (c-f) each object part correctly tracked with part models overlaid on the image potentials in (b). Note that the active model nodes are highlighted.

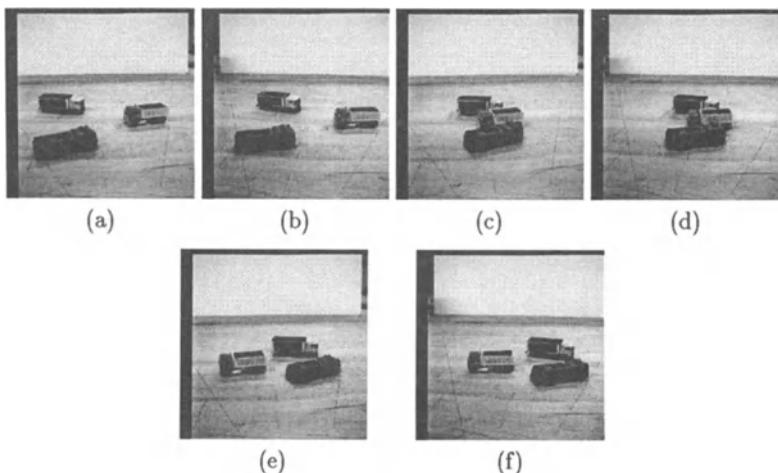


Figure 11.14 Cars sequence: three different snapshots.

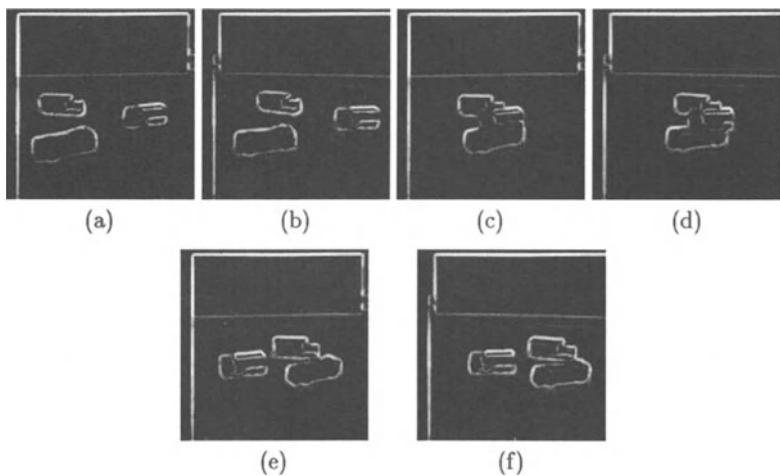


Figure 11.15 Image potential of the cars sequence.

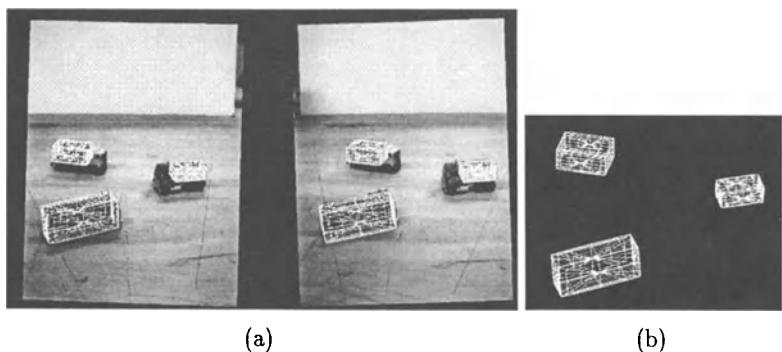


Figure 11.16 (a) Projection of the recovered 3D part models onto the stereo images in the first snapshot shown in Fig. 11.14(a-b), (b) the corresponding recovered 3D part models.

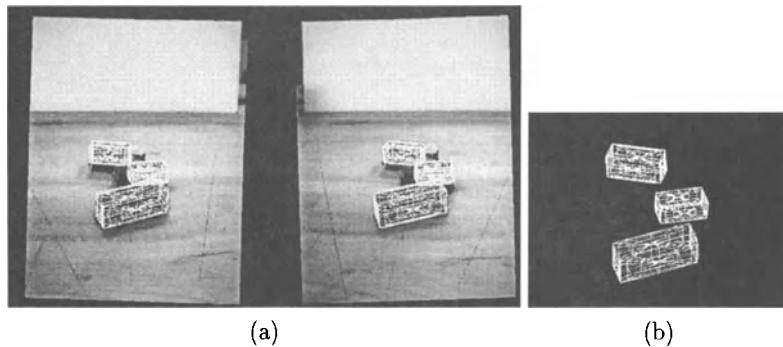


Figure 11.17 (a) Projection of the recovered 3D part models onto the stereo images in the second snapshot shown in Fig. 11.14(c-d), (b) the corresponding recovered 3D part models. Partial occlusion can be handled here.

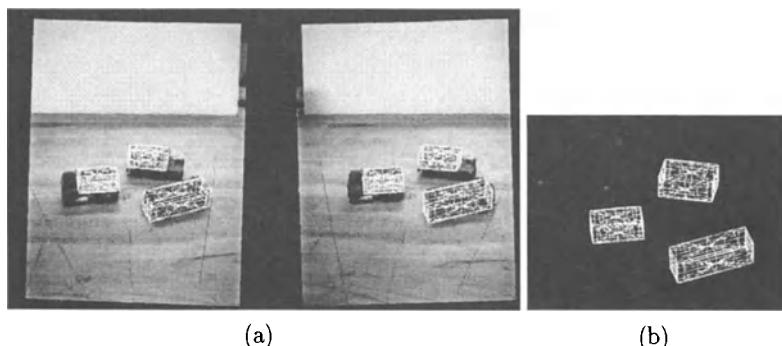


Figure 11.18 (a) Projection of the recovered 3D part models onto the stereo images in the third snapshot shown in Fig. 11.14(e-f), (b) the corresponding recovered 3D part models.

MOTION-BASED PART SEGMENTATION AND TRACKING

This chapter extends the previously developed PBM framework to allow for simultaneous part segmentation, shape and motion estimation of complex multi-part objects, including humans. This extension addresses a variety of problems in computer vision and computer graphics.

There are many applications such as anthropometry, tracking, teleconferencing, and performance measurement of athletes and patients with psycho-motor disabilities that require the determination of an object's parts, joints and their relative motion. In some tracking applications, such as determining if a person is moving towards or away from the camera, information about the movement of the centroid of the silhouette is adequate. However, in applications where the goal is the understanding of interacting objects and reasoning about scene dynamics [169], information about the movement of the parts of an articulated object is essential.

In addition, there are applications that benefit from the synergy between computer vision and computer graphics techniques such as virtual reality, teleconferencing and ergonomics. For example, in certain virtual reality applications, the movement of a virtual human is slaved to the movement of the participant. These applications require the identification of the parts of a multi-part object (e.g., the human body) and the estimation of their shape and motion parameters.

However, at present, no technique exists that segments the moving parts of a multi-part object. In addition, no technique exists that segments the apparent contour (in a monocular image sequence) of a moving human and estimates the instant rotation center of the body parts. To address this problem, most researchers have either used a standardized human model [13], based on a given statistical population (e.g., based on the Army General Forces [198]), or they have manually measured the shape parameters of the body parts that are to be tracked [102, 235]. Moreover, most of the existing approaches that employ a human body model [207, 119, 3, 160, 161, 240, 234, 163] use non-deformable models that can only approximate the human body and cannot adapt to different body sizes. Since tracking is sensitive to the shape parameters used,

considerable amount of time is spent to tune these parameters [102]. Thus, since there is no such thing as an “average human” (arithmetic mean) it is necessary to develop techniques for the automatic acquisition of human body models based on computer vision techniques.

The three-dimensional tracking of humans in motion is an important and still open research problem. The main challenges in developing algorithms for motion analysis of body parts are the complex three-dimensional non-rigid motions of humans and the occlusion among these parts. To overcome these problems and to recover the degrees of freedom associated with the motion of a moving human body, most of the existing approaches either use a prior model of the human body [207, 119, 3, 160, 161, 296, 50, 33, 162, 240, 95, 102, 150, 234, 235] or employ assumptions on the kinds of motions they can analyze [33, 204, 240]. Other techniques, such as magnetic tracking and optical capture, require that a set of markers are strapped onto various parts of the subject’s body. If the tracking of humans could be done automatically in a non-invasive, fast and practical way, applications such as vision-based computer interfaces, virtual reality, teleconferencing, surveillance and numerous others would become feasible.

In this chapter, we present a methodology for

1. detecting the moving parts of a multi-part object from a monocular image sequence. This technique allows for: (a) shape estimation of the object in each image, (b) determination of the parts that the object is composed of and their spatial relationships, (c) estimation of the parts’ shape and motion with respect to the environment [129].
2. obtaining the three-dimensional shape model of a subject’s body parts from multiple monocular image sequences [131], and
3. tracking a subject’s body parts in a dynamic environment using the three-dimensional shape models obtained with the method above [132].

The techniques presented in this chapter are based on the observation that the geometry of the apparent contour of a moving multi-part object and its deformation are related to its structure. We assume that the object under observation is either nominally rigid or it is composed of multiple parts which are nominally rigid and can move relative to each other. Assuming a non-accidental view, if the object is composed of multiple parts, then, as these parts move relative to one another, the geometry of the apparent contour of the object dynamically changes. Therefore, deviation from the assumption of nominal rigidity provides clues to assume an underlying part structure. In particular, spatiotemporal analysis of the deforming silhouettes of an object whose parts move relative to one another enables the simultaneous segmentation and shape and motion estimation of the parts. The novelty of our technique is that no prior model of the object’s structure is employed.

In the following sections, we will present the related prior work, the new theoretical advancements in PBM, and the above mentioned three techniques for motion-based two-dimensional part segmentation and tracking in 3D. Special emphasis will be given to human shape and motion estimation.

12.1 RELATED PRIOR WORK

In most previous shape estimation and tracking methods involving articulated objects, it is assumed that a segmentation into parts is given [289, 217, 296, 181]. When human shape and motion estimation is the goal, most of the existing approaches [207, 119, 3, 160, 161, 240, 234, 163] use a non-deformable human model that can only approximate the human body and cannot adapt to different body sizes. To overcome this limitation other researchers assume prior segmentation of the given data into parts [152, 230] and then fit deformable models [216, 181]. Therefore, the process of part segmentation and the process of shape and motion estimation are decoupled leading to possible inaccuracies and lack of robustness.

To acquire a three-dimensional model of the shape of a human body, laser-based methods have been developed. In particular, to capture the shape and color of a subject's body, Cyberware® has presented the Cyberware WB4 Whole Body Scanner. This scanner produces a high resolution mesh model of a human's body, and has a very high acquisition speed (the whole scan takes about 12 seconds). However, the model delivered is not articulated (therefore it can not be used for tracking of a subject's body parts in 3D), and the cost of the equipment is very high (\$410,000). vspace*-.4cm

12.2 DEFORMABLE MODELS: EXTENSIONS

To extend the number of shapes that we can model using parameterized primitives, we generalize the definition of symmetric global deformations presented in Chapter 2. We define piecewise continuous asymmetric global deformations and a parameterized *piecewise* bending deformation [133]. An example of an object whose shape can be captured more accurately using piecewise continuous asymmetric tapering is that of an upper human arm. On the other hand, to represent large protrusions or concavities and their shape evolution in a compact and intuitive way, we introduce a new shape representation based on the *parametric composition* of primitives.

12.2.1 Constant Curvature Bending

We extend the class of allowable global deformations to include a parameterized *piecewise* bending deformation that allows areas of the model not to bend and

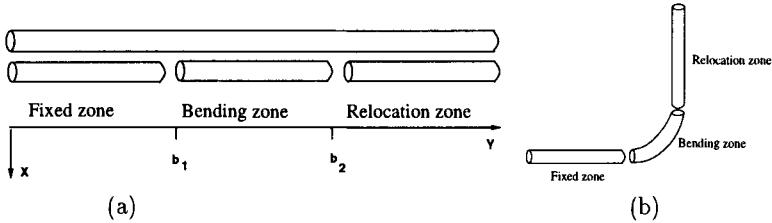


Figure 12.1 (a) The primitive prior to bending, and the three non-intersecting zones of the domain of bending, (b) the zones after bending the primitive.

ensures constant curvature along the major axis of bending. This definition is inspired from [24] and is useful for many natural and man-made objects. The new definition though, allows us to decouple the recovery of the rotation and bending parameters during model fitting.

A full treatment of the elastic bending of an object requires detailed knowledge of the physical properties of the materials used as well as the geometry of the object. The bending deformation presented here is a mathematical transformation and is based entirely on the geometric properties of the object.

The domain of the bending function is a bounded subspace of the Euclidean space \mathbf{R}^3 . This domain is partitioned into three non-intersecting zones: the *fixed zone*, the *bending zone* and the *relocation zone* (Fig. 12.1). The fixed zone remains unchanged during bending. The bending zone is where the bending occurs and the parameter b_0 denotes the radius of curvature. The range of the bending zone is controlled by parameters b_1 and b_2 and the center of the bend occurs at $(b_1 + b_2)/2$. The relocation zone is translated and rotated rigidly. The isotropic bending deformation $s = T_b(\mathbf{e}; b_0, b_1, b_2)$, along a centerline parallel to the y -axis of a primitive $\mathbf{e} = (e_x, e_y, e_z)^T$ is given by:

$$\begin{aligned} s_x &= \cos\theta(e_x - \frac{1}{b_0}) + \frac{1}{b_0} + \sin\theta(e_y - \hat{e}_y) \\ s_y &= -\sin\theta(e_x - \frac{1}{b_0}) + \frac{b_1 + b_2}{2} + \cos\theta(e_y - \hat{e}_y) \\ s_z &= e_z, \end{aligned} \quad (12.1)$$

where the bending angle θ is constant at the extremities and changes linearly in the bending zone. Specifically:

$$\theta = b_0 \left(\hat{e}_y - \frac{(b_1 + b_2)}{2} \right), \quad (12.2)$$

$$\theta^b = b_0 \left(\frac{b_1 - b_2}{2} \right), \quad (12.3)$$

$$\hat{e}_y = \begin{cases} b_1, & \text{if } e_y \leq b_1 \\ e_y, & \text{if } b_1 < e_y < b_2 \\ b_2, & \text{if } e_y \geq b_2 \end{cases}. \quad (12.4)$$

12.2.2 Piecewise Continuous Asymmetric Tapering

We generalize the definition of symmetric global deformations to allow piecewise continuous asymmetric global deformations. Although, we demonstrate the formulation of piecewise continuous asymmetric tapering only, the same extension can be applied to any global deformation. An example of an object whose shape can be captured more accurately using piecewise continuous asymmetric tapering is that of an upper human arm.

To define a linear tapering along the y -axis of the underlying primitive e , we identify two non-intersecting zones. The first is defined for the areas of the model where $e_y \geq b_3$ and the second for the areas of the model where $e_y \leq b_3$. For these two areas we can prescribe a different rate of tapering. The piecewise continuous asymmetric tapering deformation $s = \mathbf{T}_t(e; b_3, b_4, b_5)$ about the y -axis of a primitive e is given by the formula:

$$\begin{aligned} s_x &= e_x \\ s_y &= \begin{cases} \left(\frac{b_4(e_y - b_3)}{a_3 + b_3} + 1\right)e_y, & \text{if } e_y < b_3 \\ \left(\frac{b_5(e_y - b_3)}{a_3 - b_3} + 1\right)e_y, & \text{if } e_y \geq b_3 \end{cases} \\ s_z &= e_z \end{aligned}$$

12.2.3 Parametric Composition of Geometric Primitives

Initially, we will use a single deformable model to fit the apparent contour of a human or an articulated object. As a multi-part object moves and attains new postures, its apparent contour changes dynamically and large protrusions may emerge as the result of the motion of its parts (Fig. 12.18(c)). To represent large protrusions or concavities and their shape evolution in a compact and intuitive way, we introduce a new shape representation based on the *parametric composition* of primitives. Intuitively, using the parametric composition of primitives we employ a new geometric primitive to represent the protrusions and the overall shape can be described as the shape of the union of these primitives. Fig. 12.2 depicts an example of composition (union) of two superellipsoids. The composition is performed along the material v of the geometric primitive and the composition function was chosen to illustrate how different parts of the defining primitives are expressed at the composed deformable model. Referring to Fig. 12.2, the shape of the composed deformable model has the shape of the root primitive \mathbf{x}_0 for values of the material coordinate v that belong to the interval $[-\pi, 0.65\pi] \cup (0.77\pi, \pi]$. For values of v that belong to the interval $[0.65\pi, 0.77\pi]$, however, the composed deformable model has the shape of the intersecting primitive. For simplicity, we formulate the theory of composition in 2D to represent the shape of the boundary of the union of primitives.

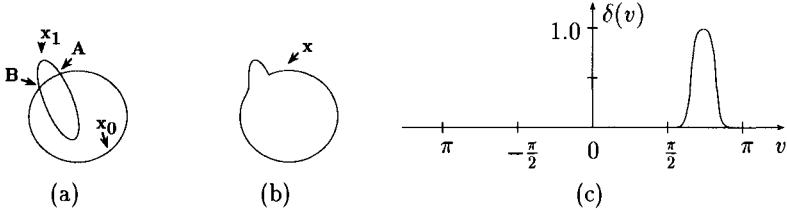


Figure 12.2 An example of composition of two superellipsoids: (a) depicts $x_1(v_1)$ which intersects $x_0(v_0)$ at points A and B , (b) depicts their composition $x(v)$, and (c) depicts the composition function $\delta(v; 0.65\pi, 0.77\pi, 10)$.

Let \mathbf{x}_0 and \mathbf{x}_1 be two 2D parametric primitives (defined by the mappings $C_0:[v_{0_b}, v_{0_e}) \rightarrow \mathbb{R}^2$ and $C_1:[v_{1_b}, v_{1_e}) \rightarrow \mathbb{R}^2$, respectively - where the subscripts b and e denote the beginning and end of the domain) positioned in space so that \mathbf{x}_1 , the *intersecting primitive*, intersects \mathbf{x}_0 , the *root primitive*, at points A and B . The material coordinates of these two points can be expressed in terms of either the material coordinate v_0 of \mathbf{x}_0 or the material coordinate v_1 of \mathbf{x}_1 . Let v_0^A and v_0^B be the values of v_0 , and v_1^A and v_1^B be the values of v_1 at points A and B , respectively. Based on the above, the shape \mathbf{x} of the composed primitive ($C:[v_b, v_e) \rightarrow \mathbb{R}^2$) can be defined in terms of the parameters of the defining primitives \mathbf{x}_0 and \mathbf{x}_1 as follows:

$$\mathbf{x}(v; v_0^A, v_0^B, c) = [1 - \delta(v; h_0^{-1}(v_0^A), h_0^{-1}(v_0^B), c)] \mathbf{x}_0(h_0(v)) + \delta(v; h_0^{-1}(v_0^A), h_0^{-1}(v_0^B), c) \mathbf{x}_1(h_1(h_0(v))), \quad (12.5)$$

where $\delta:[v_b, v_e) \rightarrow [0, 1]$ is the *composition function* for the union of two primitives. Specifically,

$$\delta(v; v_{\min}, v_{\max}, c) = \zeta(c(v - v_{\min})) - \zeta(c(v - v_{\max})), \quad (12.6)$$

$$\zeta(x) = \frac{1 + \tanh(x)}{2}, \quad (12.7)$$

where the function $\zeta:\mathbb{R} \rightarrow [0, 1]$ approximates the step function and c is a constant that controls the shape of the function δ at the neighborhoods of v_{\min} and v_{\max} . The piecewise linear function $h_0:[v_b, v_e) \rightarrow [v_{0_b}, v_{0_e})$ maps the material coordinate v to v_0 and the piecewise linear function $h_1(x):[v_{0_b}, v_{0_e}) \rightarrow [v_{1_b}, v_{1_e})$ maps the material coordinate v_0 to v_1 (a detailed presentation of the function definitions is presented in [131, 133]).

12.2.4 Kinematics of a composed deformable model

As we have explained in Chapter 3, in the physics-based framework, the geometric degrees of freedom of a shape (translation, rotation, global and local parameters) form the generalized coordinates \mathbf{q} of a model. If \mathbf{q}_0 and \mathbf{q}_1 are the generalized coordinates of the root and intersecting primitives of a composed deformable model, respectively, then the generalized coordinates of the

composed model are given by the formula:

$$\mathbf{q} = (\mathbf{q}_0^T, \mathbf{q}_1^T, \mathbf{q}_{\text{comp}}^T)^T, \quad (12.8)$$

where $\mathbf{q}_{\text{comp}} = (v_{\min}, v_{\max}, c)^T$ are the generalized coordinates of the composition function. The mapping from the \mathbf{q} -space to 3D space is achieved through the jacobian matrix. For the case of a composed deformable model, the jacobian \mathbf{L}_{comp} can be derived as follows:

$$\begin{aligned} \mathbf{x}(v) &= (1 - \delta(v)) \mathbf{x}_0(h_0(v)) + \delta(v) \mathbf{x}_1(h_1(h_0(v))) \Rightarrow \\ \dot{\mathbf{x}}(v) &= (1 - \delta(v)) \dot{\mathbf{x}}_0(h_0(v)) + \delta(v) \dot{\mathbf{x}}_1(h_1(h_0(v))) + (\mathbf{x}_1(v) - \mathbf{x}_0(v)) \dot{\delta}(v) \Rightarrow \\ \ddot{\mathbf{x}}(v) &= (1 - \delta(v)) \mathbf{L}_0 \dot{\mathbf{q}}_0 + \delta(v) \mathbf{L}_1 \dot{\mathbf{q}}_1 + (\mathbf{x}_1(v) - \mathbf{x}_0(v)) \frac{\partial \delta(v)}{\partial \mathbf{q}} \dot{\mathbf{q}} \Rightarrow \\ \ddot{\mathbf{x}}(v) &= \mathbf{L}_{\text{comp}} \dot{\mathbf{q}}. \end{aligned} \quad (12.9)$$

Since the function δ is dependent only on the values of the parameters \mathbf{q}_{comp} then

$$\frac{\partial \delta(v)}{\partial \mathbf{q}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \frac{\partial \delta(v)}{\partial \mathbf{q}_{\text{comp}}} \end{bmatrix}. \quad (12.10)$$

In summary,

$$\mathbf{L}_{\text{comp}}(v) = \begin{bmatrix} (1 - \delta(v)) \mathbf{L}_0 & \delta(v) \mathbf{L}_1 & \mathbf{0}_{3 \times q_0} & \mathbf{0}_{3 \times q_1} & [\mathbf{x}_1(v) - \mathbf{x}_0(v)] \frac{\partial \delta(v)}{\partial \mathbf{q}_{\text{comp}}} \end{bmatrix}.$$

Therefore, the jacobian $\mathbf{L}_{\text{comp}}(v)$ at a point of the composed model with material coordinate v depends on the Jacobians $\mathbf{L}_0(v)$ and $\mathbf{L}_1(v)$ of the defining models. The degree of dependence is regulated by the values of the composition function.

12.2.5 Dynamics of a composed deformable model

Our goal, when fitting the model to visual data, is to recover the vector of generalized coordinates \mathbf{q} . This is achieved based on forces that the data exert to the surface of the model (see Chapter 5). From the two-dimensional or three-dimensional forces that the data apply to the two-dimensional or three-dimensional deformable model respectively, we can compute the generalized forces that appear at the simplified equations of motion 3.51. In particular, the generalized forces \mathbf{f}_q^T are computed using the formula:

$$\mathbf{f}_q^T = \int \mathbf{f}_{\text{applied}}^T \mathbf{L} du, \quad (12.11)$$

where $\mathbf{f}_{\text{applied}}$ are the external two-dimensional or three-dimensional forces that are exerted on the model and \mathbf{L} is the jacobian matrix. For the case of a composed deformable model, the following formula applies:

$$\mathbf{f}_q^T = \int \mathbf{f}_{\text{applied}}^T \mathbf{L}_{\text{comp}} du. \quad (12.12)$$

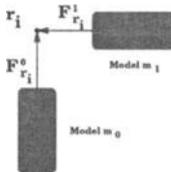


Figure 12.3 Force assignment.

In addition, these generalized forces can be broken down into the following components:

$$\begin{aligned} \mathbf{f}_q &= (\mathbf{f}_{q_0}^T, \mathbf{f}_{q_1}^T, \mathbf{f}_{q_{\text{comp}}}^T)^T, \text{ where} \\ \mathbf{f}_{q_i} &= (\mathbf{f}_{q_{c_i}}^T, \mathbf{f}_{q_{\theta_i}}^T, \mathbf{f}_{q_{s_i}}^T, \mathbf{f}_{q_{d_i}}^T)^T, \quad i \in \{0, 1\}. \end{aligned} \quad (12.13)$$

In the next section, we explain how the external applied forces are computed.

12.2.6 All neighbors force assignment based on fuzzy clustering

In the PBM data fitting method, image datapoints apply external forces to the deformable models. In the previous work [181], the *nearest neighbor force assignment* method was employed. In that approach, it is assumed that a datapoint can be associated with only one model. However, there are two main problems with the nearest neighbor method; first, with some probability, the nearest datapoint is not always the most appropriate one to use and second, in the case where the data from more than one object are fitted, the method is order dependent. Therefore, this method is most effective for fitting a single object or multiple non-interfering objects in low density clutter.

However, in the case of multiple objects or parts, force assignment becomes more complicated. The reason being that a decision has to be made for the determination of the association of the observed datapoint with a model. To overcome this problem, we describe next an *all neighbors force assignment* algorithm inspired by the theory of fuzzy clustering [242]. In this *all neighbors force assignment* algorithm a datapoint can be associated with more than one models. By introducing a datapoint's degree of membership in a particular model (that is, which deformable model describes this datapoint) a datapoint exerts a force to each of the deformable models used in fitting (Fig. 12.3) instead of only to one. In particular, each datapoint \mathbf{r}_i applies forces to each model proportional to a weight $p_{r_i,j}$ which is the probability that this point is correctly associated with the model j . Hence, the force that a datapoint \mathbf{r}_i applies to the nearest point \mathbf{x}_i^j of a deformable model j is given by the formula $\mathbf{F}_{r_i}^j = p_{r_i,j}(\mathbf{r}_i - \mathbf{x}_i^j)$, where $p_{r_i,j}$ is the *certainty weight* of a point \mathbf{r}_i to a model j . One can think the process of deciding to which deformable model an orphan datapoint should apply forces to, as deciding to which cluster (for our problem

to which deformable model) the vector of the position of the datapoint belongs to. This classification is based on the distance of this vector from the prototype vectors. The prototype vectors are the positions of the points on the deformable models which lie at the nearest distance from the datapoint. Since the certainty weights represent the degree of membership of a datapoint to a given model, our algorithm can be viewed as gradually decreasing the fuzziness of the associations. We compute these weights, whose sum is always equal to one, by minimizing an appropriately selected energy expression (see equation 12.14).

Let $\{\mathbf{r}_i\}, i = \{1, \dots, n\}$, be the set of datapoints for which certainty weights will be computed, where n is the cardinality of the set. Also, let \mathbf{x}_i^j be the point of model j that is located in the nearest distance from datapoint \mathbf{r}_i , and let $p_{r_i,j}$ be the certainty that the datapoint \mathbf{r}_i is best fitted by the model j , $j = \{1, \dots, m\}$, where m is the number of models employed in fitting. Furthermore, let $\mu_{x_i^j,k}$ be the certainty measure that a model point \mathbf{x}_i^j belongs to model \mathbf{m}_k where $k = \{1, \dots, m\}$ (its value is 1 if $j = k$ and 0 otherwise).

One can think the process of deciding to which deformable model an orphan datapoint should apply forces to, as the fuzzy classification of a finite number of unlabeled vectors based on their proximity to a set of labeled prototype vectors with known membership. Let $\{\{x_1^1, \dots, x_1^m\}, \{x_2^1, \dots, x_2^m\}, \dots, \{x_n^1, \dots, x_n^m\}\}$ be the set of the labeled prototype vectors. The set of datapoints is the set of unlabeled vectors. Therefore, the certainty weights $p_{r_i,j}$ of a point \mathbf{r}_i to a model j are designed to minimize the following energy term:

$$\mathcal{E}_r = \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^m \frac{(p_{r_i,j} - \mu_{x_i^j,k})^2}{(\mathbf{r}_i - \mathbf{x}_i^j)^T (\mathbf{r}_i - \mathbf{x}_i^j)}, \quad (12.14)$$

subject to the constraint $\sum_{j=1}^m p_{r_i,j} = 1$, $i = \{1, \dots, n\}$. This energy is designed in a way that the membership $p_{r_i,j}$ assigned to a point \mathbf{r}_i is similar to the membership $\mu_{x_i^j,k}$ of x_i^j when \mathbf{r}_i and \mathbf{x}_i^j are close. Therefore, the corresponding energy for each datapoint \mathbf{r}_i is

$$\mathcal{E}_{r,i} = \sum_{j=1}^m \sum_{k=1}^m \frac{(p_{r_i,j} - \mu_{x_i^j,k})^2}{(\mathbf{r}_i - \mathbf{x}_i^j)^T (\mathbf{r}_i - \mathbf{x}_i^j)}. \quad (12.15)$$

We perform the minimization for each datapoint \mathbf{r}_i using the method of Lagrange multipliers and minimize the following function:

$$\mathcal{E}_{r,i} = \sum_{j=1}^m \sum_{k=1}^m \frac{(p_{r_i,j} - \mu_{x_i^j,k})^2}{(\mathbf{r}_i - \mathbf{x}_i^j)^T (\mathbf{r}_i - \mathbf{x}_i^j)} - \lambda \left(\sum_{j=1}^m p_{r_i,j} - 1 \right) \quad (12.16)$$

with respect to $p_{r_i,j}$. The minimization of $\mathcal{E}_{r,i}$ is performed by setting $\frac{\partial \mathcal{E}_{r,i}}{\partial p_{r_i,j}} = 0$, $i = \{1, \dots, n\}$. After some algebraic manipulation we obtain the following expression:

$$p_{r_i,j} = \frac{1}{m} + \frac{\sum_{k=1}^m \frac{1}{(\mathbf{r}_i - \mathbf{x}_i^j)^T (\mathbf{r}_i - \mathbf{x}_i^j)} (\mu_{x_i^j,k} - \frac{1}{m})}{\sum_{k=1}^m \frac{1}{(\mathbf{r}_i - \mathbf{x}_i^j)^T (\mathbf{r}_i - \mathbf{x}_i^j)}}. \quad (12.17)$$

For the case of two deformable models \mathbf{m}_0 and \mathbf{m}_1 the weights are the following:

$$\begin{aligned} p_{r_i,0} &= \frac{(\mathbf{r}_i - \mathbf{x}_i^1)^T (\mathbf{r}_i - \mathbf{x}_i^1)}{(\mathbf{r}_i - \mathbf{x}_i^0)^T (\mathbf{r}_i - \mathbf{x}_i^0) + (\mathbf{r}_i - \mathbf{x}_i^1)^T (\mathbf{r}_i - \mathbf{x}_i^1)}, \\ p_{r_i,1} &= \frac{(\mathbf{r}_i - \mathbf{x}_i^0)^T (\mathbf{r}_i - \mathbf{x}_i^0)}{(\mathbf{r}_i - \mathbf{x}_i^0)^T (\mathbf{r}_i - \mathbf{x}_i^0) + (\mathbf{r}_i - \mathbf{x}_i^1)^T (\mathbf{r}_i - \mathbf{x}_i^1)}. \end{aligned} \quad (12.18)$$

One consequence of the fact that a datapoint attracts multiple models is that the new force assignment algorithm allows partial overlap between the two models at a joint.

12.3 INFERRING STRUCTURE IN 2D

This section presents a technique for building a segmented two-dimensional model of a multi-part object whose parts are moving relative to one another. The goal is to automatically segment the apparent contour of the multi-part object and estimate the shape of its parts without assuming a prior model of the object or of its parts. The structure of a multi-part object is denoted by a graph in which there is a node for each part of the object that specifies the geometric model of the part. Arcs between nodes denote that a joint exists between parts of an object and each one allows a range of positions between the two parts.

First, we present the *Part Segmentation Algorithm* and then we elaborate on the criteria and their associated algorithms. The identification of the parts of a multi-part object is driven by visual events that link changes in the apparent contour of the multi-part object to its structure. In addition, we model the process of decomposition using the theory of Supervisory Control Theory of Discrete Event Systems.

To accomplish the goal of segmenting the apparent contour of a multi-part object and to estimate the shape of its parts in 2D, we use a sequence of images in which movement between the parts occurs and we employ the Part Segmentation Algorithm (PSA) (Fig. 12.4). The technique is based on the observation that the geometry of the model of the apparent contour of a multi-part object with nominally rigid parts and its deformation under motion are related to the structure of the object under observation. Therefore, deviation from the assumption of nominal rigidity provides clues to assume an underlying part structure.

The apparent contour of a multi-part object is the projection of the locus of points on the object which separates the visible from the occluded surfaces of the parts. An initial assumption is made that the object under observation consists of a single part. A single deformable model is fitted to time-varying image data using the physics-based framework [180]. If the object is composed from multiple parts, when these parts move relative to one another, the geometry

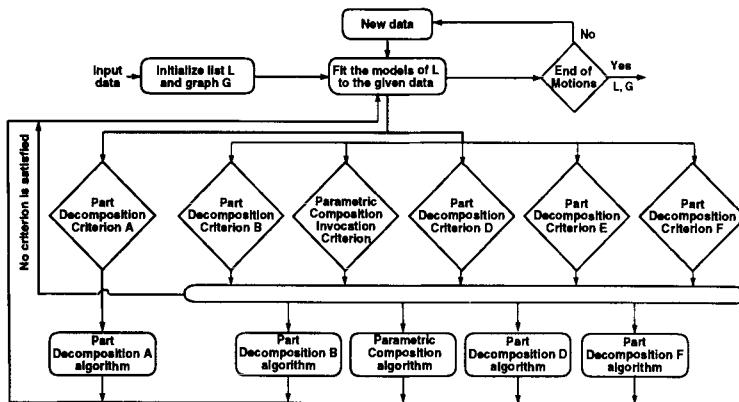


Figure 12.4 Flow diagram of the Part Identification Algorithm.

of the apparent contour of the object dynamically changes. Specifically, there is a set of viewpoints from which, when one observes the relative movement of the parts the geometry of the apparent contour changes. Assuming that one observes the movement of one or multiple objects which are either nominally rigid or are composed of multiple parts which are nominally rigid and can move relative to each other then the changes to the apparent contour can be attributed to the following reasons (assuming that the observer is stationary and a non-accidental point of observation): (1) multiple independent objects occlude each other partially or fully initially and then during their movement they become unoccluded, and (2) parts of a multi-part object which initially occlude each other become unoccluded as they move. As the parts of a multi-part object move relative to each other (or as multiple parts which are initially occluded move relative to each other) the apparent contour changes dynamically (protrusions or intrusions may appear within the outline) and it might change topology as well. In addition, a hole¹ might be evolving within the apparent contour. Based on observing these visual events six criteria have been developed that lead to the hypothesis of multiple parts or multiple objects.

- The first criterion (*Part Decomposition Criterion A*) refers to the case where objects which are partially occluding each other (and they appear as a single object initially) they become completely visible. An example where this criterion will apply is the case of removing the cap from a pen as in Figs. 12.16(a-f). The topology change of the model of the apparent contour indicates clearly the existence of multiple parts.
- The second criterion (*Part Decomposition Criterion B*) refers to the case where the parts of an articulated object which are connected at their endpoints and do not fully occlude each other initially rotate in a plane parallel

¹By the term “hole”, we will denote the existence of a closed contour within the apparent contour. The significance of a hole within the apparent contour in section will be discussed in section 12.3.4.

to the viewing plane. An example where this criterion will apply is the case of a robot arm moving as in Fig. (12.7).

- The third, fourth and sixth criteria (*Parametric Composition Invocation Criterion* and *Part Decomposition Criteria D* and *F*) refer to the case of articulated objects whose parts are joined at their endpoints, but initially one of them is fully occluded. As the occluded parts move, significant protrusions may be observed within the initial apparent contour. An example of this case occurs when we observe the movement of a human arm from the vertical to the horizontal position (assuming the observer observes the sagittal plane of the human) (Fig. 12.18(a-e)). Our treatment is based on the existence of significant protrusions and on whether there is a hole within the apparent contour.
- The fifth criterion (*Part Decomposition Criterion E*) refers to the case of articulated objects whose parts are joined (but not at their endpoints) and they move relative to each other. As the occluded parts move, intrusions may be observed within the initial apparent contour. An example of this case is the movement of opening the handles of a pair of pliers as in Fig. (12.10).

If any of the above criteria is satisfied then the corresponding algorithm is employed to hypothesize multiple parts or multiple objects. It should be noted that this algorithm is general because instead of using templates, which are domain specific, our effort is concentrated in specifying general criteria sufficient to analyze and to segment a dynamic apparent contour into parts. Below, the structure of the PSA algorithm is presented.

Algorithm: Part Segmentation (PSA)

Step 1: Initially, assume that the multi-part object consists of a single part.

Create a list of deformable models L (with one entry initially) that will be used to model the objects present in the scene and their parts. In addition, create a graph G with one node. The nodes of the graph G denote the parts recovered by the algorithm. The arcs of the graph denote which parts are connected by joints.

Step 2: If not all the frames of the motion sequence have been processed, fit the models of the list L to the image data using the physics-based shape and motion estimation framework [180] and execute steps 3 and 4. Otherwise, output L and G .

Step 3: For each model in L , determine:

- a: if the *Part Decomposition Criterion A* is satisfied,
- b: else if the *Part Decomposition Criterion B* is satisfied,
- c: or if the *Parametric Composition Invocation Criterion* is satisfied.
- d: or if the *Part Decomposition Criterion D* is satisfied.
- e: or if the *Part Decomposition Criterion E* is satisfied.

For each composed model in L , determine:

- f: if the *Part Decomposition Criterion F* (explained in section 12.3.6) is satisfied (e.g., this criterion is satisfied during the later stages of the movement of the arm with respect to the torso).



Figure 12.5 Change of topology of the apparent contour indicates the existence of multiple objects.

Step 4: For each model in L , at most one of the criteria specified in step 3 will be satisfied. Depending on the outcome, determine which of the following algorithms to invoke.

For each model in L

- If 3a is satisfied, invoke the *Part Decomposition A* algorithm.
- If 3b is satisfied, invoke the *Part Decomposition B* algorithm.
- If 3c is satisfied, invoke the *Parametric Composition* algorithm.
- If 3d or 3e is satisfied, invoke the *Part Decomposition D* algorithm.
- If 3f is satisfied, invoke the *Part Decomposition F* algorithm.

For step 2 of the PSA, the models included in list L are fitted to the image data, using the *all neighbors* force assignment algorithm (section 12.2.6) and the physics-based framework [181]. The *all neighbors* force assignment algorithm allows the overlap between models and therefore, the instant rotation center of a part can be determined (see section 12.3.7).

In the following, the Parametric Composition Invocation Criterion and Part Decomposition Criteria (and the related algorithms), through which one is able to fully segment the apparent contours of a multi-part object, are presented.

12.3.1 PSA - Step 3a: Topologically adaptive deformable models.

Observing the evolution of the apparent contours in the image sequence of Fig. 12.5, one can offer two possible interpretations. On the one hand, if one assumes that the object observed is elastic then large deformations may be due to growth. On the other hand, if one assumes that the object is nominally rigid then large spatiotemporal changes in the geometry of the apparent contours are due to the existence of multiple objects or multiple parts that are occluded initially. One of the criteria that signal the existence of multiple objects is changes in the topology of the apparent contour. Before we present the criterion, we refer to the definitions of a *part line* and of a *neck line* as given in [259]. A *part-line* is a curve which is entirely embedded in the two-dimensional geometric model of the shape and whose end points rest on its boundary. In addition, a part line divides a shape into two connected components. A *neck line* is a part-line which is also a local minimum of the diameter of an inscribed circle.

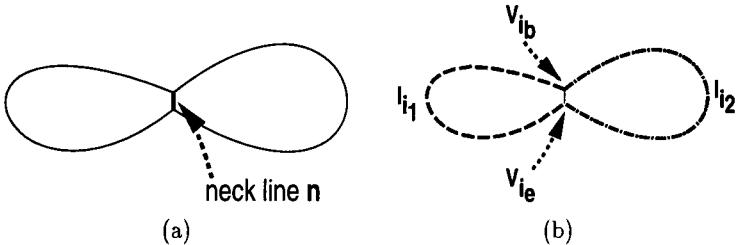


Figure 12.6 Notation pertaining to Part Decomposition A algorithm. (a) The neck line n within the model of an apparent contour divides the model into two element domains (b).

Part Decomposition Criterion A

If the shape model of the apparent contour dynamically changes and a “short” neck line is formed, then invoke the Part Decomposition A algorithm. More specifically, if $\|\mathbf{p}_i(v, t) - \mathbf{p}_i(v, t_{init})\| > K_{A_1}$, where K_{A_1} is an *a priori* defined constant, and $\mathbf{p}_i(v, t)$, $\mathbf{p}_i(v, t_{init})$ represent the current and the initial shapes (w.r.t. the model-centered reference frame) of the model \mathbf{m}_i of the apparent contour and there is a neck line n within the model for which $\|n\| < K_{A_2}$, where K_{A_2} is an *a priori* defined constant, then invoke the Part Decomposition A algorithm.

Algorithm: Part Decomposition A

- Step 1:** Determine the material coordinates v_{ib} and v_{ie} of the endpoints of the neck line and identify the two element domains I_{i_1} and I_{i_2} ($I_{i_1} \subset \Omega$ and $I_{i_2} \subset \Omega$) of the material coordinate v in which the model will be partitioned (Fig. 12.6).
- Step 2:** Perform an eigenvector analysis on the datapoints that correspond to the two domains I_{i_j} , ($j = 1, 2$) to approximate the parameters of two deformable models \mathbf{m}_{i_1} and \mathbf{m}_{i_2} which can fit these datapoints.
- Step 3:** Update the list L by replacing the model \mathbf{m}_i with the new models \mathbf{m}_{i_1} and \mathbf{m}_{i_2} .

12.3.2 PSA - Step 3b

The time-varying image contour data from an elastic object which bends or from the rotating parts of an articulated object (assuming that the parts are connected at their endpoints and do not fully occlude each other initially) can be fitted using a deformable model which undergoes bending deformation. An example of such a movement is shown in Fig. 12.7. To determine the existence of multiple parts, the values of the model’s bending parameters are monitored. The first part of the criterion ensures that the object under observation undergoes a bending deformation (undergoing a transition from an unbend to a bend state or vice versa). If the error of fit $\psi(t)$ does not change during the bending process, then we observe an elastic object that bends. However, if the error of fit does change, then this is an indication that we observe multiple objects



Figure 12.7 Significant bending of the model of the apparent contour of a multi-part object along with an increase of the error of fit indicates the existence of multiple parts.

rotating relative to each other or multiple parts of the same object rotating relative to each other. As stated previously, the list L contains the deformable models $\mathbf{m}_i, i = \{1, \dots, n\}$ that model the objects and their parts in a scene. Let $\theta_i^b(t)$ be the b^b bending parameter of a deformable model \mathbf{m}_i at time t (see equation 12.3) and $\theta_i^b(t_{init})$ be the bending parameter of the model fitted to the data at time t_{init} , when the model was initialized. In addition, let $\psi(t)$ and $\psi(t_{init})$ be the error of fit of the model \mathbf{m}_i fitted to the data at times t and t_{init} , respectively.

Part Decomposition Criterion B: If the bending parameters of a model \mathbf{m}_i satisfy the relation $\theta_i^b(t) - \theta_i^b(t_{init}) > K_{B_1}$ and the error of fit (using only global deformations) of the model satisfies the relation $(\psi(t) - \psi(t_{init})) > K_{B_2}$, where K_{B_1} and K_{B_2} are a priori defined constants, then the underlying object is not an elastic object, but it is composed from multiple parts. Therefore, invoke the Part Decomposition B algorithm to replace the model \mathbf{m}_i with two new models \mathbf{m}_{i_1} and \mathbf{m}_{i_2} .

Algorithm: Part Decomposition B

Step 1: Based on the bending parameters b_{i_1} and b_{i_2} , (see equation (12.1)) of the deformable model \mathbf{m}_i , identify the datapoints that correspond to the fixed and relocation zones of the bent model \mathbf{m}_i and mark them as to be modeled by the two new models \mathbf{m}_{i_1} and \mathbf{m}_{i_2} , respectively. However, the datapoints that correspond to the bending zone of the model \mathbf{m}_i are marked as *orphan datapoints* since it is uncertain as to which of the two new models they should be assigned. This is necessary since we do not know in advance the shape of the underlying parts.

Step 2: Perform an eigenvector analysis on the datapoints that correspond to the fixed and relocation zones of the model \mathbf{m}_i , to approximate the parameters of two deformable models \mathbf{m}_{i_1} and \mathbf{m}_{i_2} which can describe these data.

Step 3: Update list L by replacing the model \mathbf{m}_i with the two new models \mathbf{m}_{i_1} and \mathbf{m}_{i_2} .

12.3.3 PSA - Step 3c

The third criterion refers to the case of articulated objects whose parts are joined at their endpoints, but initially one of them is fully occluded. As the

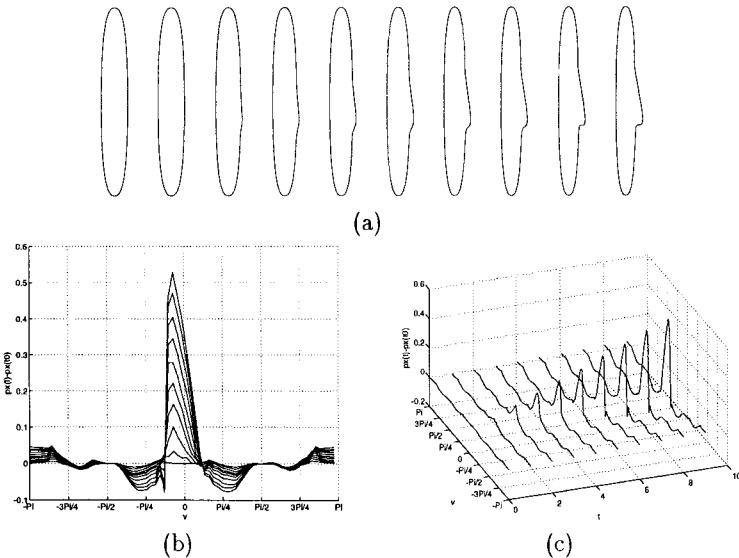


Figure 12.8 As the protrusions to a shape are evolving over time (a), the deformations of the shape are monitored (shown superimposed in (b) and over time in (c)).

parts of the object move and attain new postures the apparent contour changes dynamically and large protrusions may emerge (Fig. 12.18(c)). If there is no hole present within the apparent contour and there is a significant deformation of the apparent contour (Fig. 12.8), we represent the protrusions compactly based on the composition of two primitives.

Parametric Composition Invocation Criterion

Signal the need for parametric composition of primitives if no hole is evolving within the apparent contour and $\|\mathbf{p}_i(v, t) - \mathbf{p}_i(v, t_{init})\| > K_{A_1}$, where K_{A_1} is the same a priori defined constant as in Part Decomposition Criterion A, and $\mathbf{p}_i(v, t)$, $\mathbf{p}_i(v, t_{init})$ represent the current and the initial shapes (w.r.t. the model-centered reference frame) of the model (of the apparent contour) \mathbf{m}_i .

Algorithm: Parametric Composition

Step 1: Determine the interval I_i of the material coordinate v of the model \mathbf{m}_i , in which maximum variation of the shape over time is detected.

Step 2: Perform an eigenvector analysis on the datapoints that correspond to the interval I_i , to approximate the parameters of a new deformable model \mathbf{m}_{i_1} which can fit these datapoints.

Step 3: Construct (as explained in section 12.2.3) a composed parametric primitive \mathbf{n} with \mathbf{m}_i and \mathbf{m}_{i_1} as defining primitives.

Step 4: Update list L by replacing the model \mathbf{m}_i with the composed model \mathbf{n} .

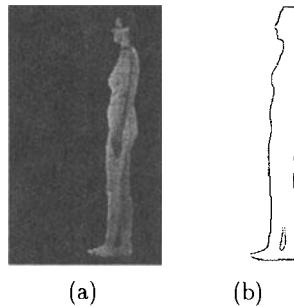


Figure 12.9 When the occluded leg gradually becomes visible, a new contour evolves within the apparent contour.

12.3.4 PSA - Step 3d

The visual event of a hole evolving within the tracked apparent contour indicates that parts which were initially occluded are gradually becoming visible. During this process though, the still occluded regions of the part that is becoming visible are not contiguous, thus, the appearance of the hole (Fig. 12.9). Therefore, if there is a hole present within an apparent contour whose shape has changed significantly, the parametric composition algorithm is not invoked. Instead, the evolution of the hole is monitored and the Part Decomposition D algorithm is invoked only when the hole ceases to exist (provided that the apparent contour is still deformed w.r.t. the initial shape).

Part Decomposition Criterion D

If an evolving contour within the apparent contour ceases to exist and for a model \mathbf{m}_i the relation $\|\mathbf{p}_i(v, t) - \mathbf{p}_i(v, t_{init})\| > K_{A_1}$ (where K_{A_1} is the same a priori defined constant as in Part Decomposition Criterion A, and $\mathbf{p}_i(v, t)$, $\mathbf{p}_i(v, t_{init})$ represent the current and the initial shapes w.r.t. the model-centered reference frame) holds, then invoke the Part Decomposition D algorithm.

Algorithm: Part Decomposition D

Step 1: Determine the model's domain \mathcal{I}_{i_1} ($\mathcal{I}_{i_1} \subset \Omega$) of the material coordinate v , in which maximum variation of the shape over time is detected.

Step 2: Monitor the evolution of the shape of the deformable model \mathbf{m}_i and cluster its finite elements based on their change of orientation. Determine the model's domain \mathcal{I}_{i_2} , $\mathcal{I}_{i_2} \subseteq \mathcal{I}_{i_1}$, for which consistent change of the orientation of the finite elements within the element domain \mathcal{I}_{i_1} of the deformable model is detected.

Step 3: If n ($n > 1$) clusters are recovered, perform an eigenvector analysis on the datapoints that correspond (e.g., are applying forces) to these finite elements to approximate the parameters of the deformable models \mathbf{m}_{i_j} , ($j = 1, \dots, n$) which can fit these datapoints.

Step 4: Update the list L by replacing the model $\mathbf{m}_i(t)$ with the models $\mathbf{m}_i(t_{init})$ and \mathbf{m}_{i_j} , ($j = 1, \dots, n$).

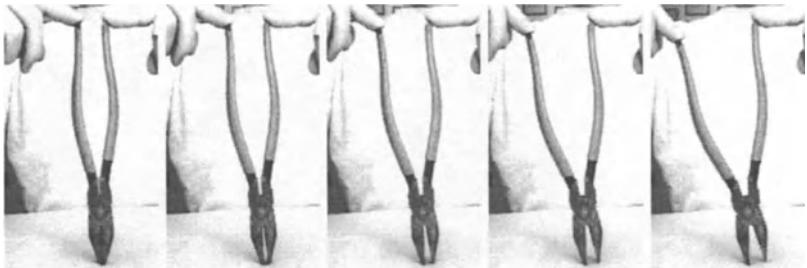


Figure 12.10 Large intrusions within the apparent contour indicate the existence of multiple parts.

12.3.5 PSA - Step 3e

The fifth criterion refers to the case of articulated objects whose parts are joined (but not at their endpoints) and they move relative to each other. As the parts of the object move and attain new postures the apparent contour changes dynamically and large intrusions are emerging. The evolution of the shape is monitored and if there is no neck line formed with the apparent contour then the following criterion applies (Fig. 12.10).

Part Decomposition Criterion E

If for a model \mathbf{m}_i the relation $\|\mathbf{p}_i(v, t) - \mathbf{p}_i(v, t_{init})\| > K_{A_1}$ (where K_{A_1} is the same a priori defined constant as in Part Decomposition Criterion A, and $\mathbf{p}_i(v, t)$, $\mathbf{p}_i(v, t_{init})$ represent the current and the initial shapes w.r.t. the model-centered reference frame) holds, then invoke the Part Decomposition D algorithm.

12.3.6 PSA - Step 3f

For step 2 of the PSA, the models of the updated list L are continuously fitted to the time-varying image data, using our *all neighbors* force assignment algorithm (described in section 12.2.6) and the physics-based framework introduced in [181]. If the parameters of a composed deformable model indicate that its defining primitives are moving with respect to one another, then this signals the presence of two distinct parts. Therefore, it verifies the hypothesis that the deformation of the apparent contour is the result of two parts moving relative to each other.

Part Decomposition Criterion F

If the generalized coordinates of a composed model \mathbf{m}_i (whose defining primitives are the models \mathbf{m}_{i_0} and \mathbf{m}_{i_1}) satisfy the relation $\|\Delta\mathbf{q}_t(t) - \Delta\mathbf{q}_t(t_{init})\| > K_{F_1} \vee \|\Delta\mathbf{q}_\theta(t) - \Delta\mathbf{q}_\theta(t_{init})\| > K_{F_2}$, where K_{F_1} and K_{F_2} are two *a priori* defined constants, t_{init} is the time that the com-

posed deformable model \mathbf{m}_i was initialized, $\Delta \mathbf{q}_t(t) = \mathbf{q}_{t_{i_1}}(t) - \mathbf{q}_{t_{i_0}}(t)$ and $\Delta \mathbf{q}_\theta(t) = \mathbf{q}_{t_{i_1}}(t) - \mathbf{q}_{t_{i_0}}(t)$ represent the relative translation and orientation of the model-centered coordinate systems of \mathbf{m}_{i_0} and \mathbf{m}_{i_1} , then decompose \mathbf{m}_i into its constituent parts.

Algorithm: Part Decomposition F

Step 1: Construct two new deformable models \mathbf{n}_{i_0} and \mathbf{n}_{i_1} using the parameters of the defining models \mathbf{m}_{i_0} and \mathbf{m}_{i_1} of the composed model \mathbf{m}_i .

Step 2: Update G and L by replacing \mathbf{m}_i with \mathbf{n}_{i_0} and \mathbf{n}_{i_1} .

12.3.7 Computation of the instant rotation center

This section presents the equations related to determining the instant rotation center around which a model rotates for the case of two-dimensional deformable models. Let \mathbf{m}_i be a deformable model which lies on the plane xy , let ω denote the rotation of the object coordinate system around the z -axis and let ${}^\Phi \mathbf{x}^i = [x_1^i, x_2^i]^T$ be the position of node i of the deformable model. Assuming that the new position is the result of rotating around a point (which we will call the projection of the instant rotation center) with coordinates ${}^\Phi \mathbf{r} = [r_1, r_2]^T$ with respect to Φ , then

$$\mathbf{Q}(t)[{}^\Phi \mathbf{x}^i(t) - {}^\Phi \mathbf{r}(t)] + {}^\Phi \mathbf{r}(t) = {}^\Phi \mathbf{x}^i(t + \delta t), \quad (12.19)$$

where $\mathbf{Q} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$. Due to noise, the vector

$$\mathbf{k} = \mathbf{Q}(t)[{}^\Phi \mathbf{x}^i(t) - {}^\Phi \mathbf{r}(t)] + {}^\Phi \mathbf{r}(t) - {}^\Phi \mathbf{x}^i(t + \delta t) \quad (12.20)$$

will not be equal to $\mathbf{0}$. Our goal is to determine the vector ${}^\Phi \mathbf{r}$ such that the quantity $\mathbf{k}^T \mathbf{k}$ is minimized. To find the minimum of the quantity $\mathbf{k}^T \mathbf{k}$, one has to solve the following system of non-linear equations

$$\frac{\partial(\mathbf{k}^T \mathbf{k})}{\partial r_1} = \frac{\partial(\mathbf{k}^T \mathbf{k})}{\partial r_2} = \frac{\partial(\mathbf{k}^T \mathbf{k})}{\partial \omega} = 0, \quad (12.21)$$

where

$$\begin{aligned} \frac{\partial(\mathbf{k}^T \mathbf{k})}{\partial r_1} &= 4(1 - \cos\omega)r_1 + 2(x_1^i(t + \delta t) + x_1^i(t))\cos\omega + \\ &\quad 2(x_2^i(t + \delta t) + x_2^i(t))\sin\omega - 2(x_1^i(t) + x_1^i(t + \delta t)), \\ \frac{\partial(\mathbf{k}^T \mathbf{k})}{\partial r_2} &= 4(1 - \cos\omega)r_2 + 2(x_2^i(t) + x_2^i(t + \delta t))\cos\omega + \\ &\quad 2(x_1^i(t) - x_1^i(t + \delta t))\sin\omega - 2(x_2^i(t) + x_2^i(t + \delta t)), \\ \frac{\partial(\mathbf{k}^T \mathbf{k})}{\partial \omega} &= 2(r_1^2 + r_2^2 + x_1^i(t)x_1^i(t + \delta t) + x_2^i(t)x_2^i(t + \delta t))\sin\omega + \\ &\quad 2(x_2^i(t)x_1^i(t + \delta t) - x_1^i(t)x_2^i(t + \delta t))\cos\omega + \\ &\quad (2(x_2^i(t + \delta t) - x_2^i(t))\cos\omega - 2(x_1^i(t) + x_1^i(t + \delta t))\sin\omega)r_1 + \\ &\quad (2(x_1^i(t) - x_1^i(t + \delta t))\cos\omega - 2(x_2^i(t) + x_2^i(t + \delta t))\sin\omega)r_2. \end{aligned}$$

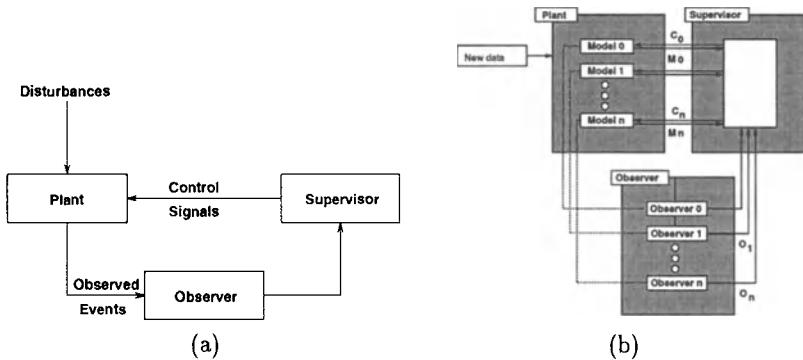


Figure 12.11 (a) The components of a dynamic system according to the Supervisory Control Theory of Discrete Event Dynamic systems. (b) Communication channels between the plant, the observers and the supervisor.

The system of equations (12.21) is solved using the Newton-Raphson method for non-linear system of equations [226]. For the method to converge in a few iterations, a good initial estimate is provided. The description of the computation of the initial estimate is provided below. The deformable model which represents the shape of the occluding contour is discretized to n nodes. Two of these nodes, for which the assumption of rigidity stands, are chosen. Specifically, for the assumption of rigidity to hold for a node, its position with respect to the model-centered coordinate system \mathbf{p}^i should remain the same during the motion. Let's assume that nodes i and j are chosen. Then, these are the nodes for which the quantity $\|\mathbf{p}^i(t + \delta t) - \mathbf{p}^i(t)\|$ takes the least and second least value over the set of nodes. Then, an initial estimate of the rotation center is the point where the normals to the midpoint of the vectors $\Phi \mathbf{x}^i(t + \delta t) - \Phi \mathbf{x}^i(t)$, and $\Phi \mathbf{x}^j(t + \delta t) - \Phi \mathbf{x}^j(t)$, intersect.

12.3.8 Implementation of the PSA

In this section, we describe the modeling and the control of the processes involved in the system implementing the PSA, using the **Discrete Event Systems** theory [232].

As we explained in section 12.3, for each frame of the image sequence the PSA fits the models to the data and when one of the criteria is satisfied, the corresponding algorithm is employed to replace one model with two or more new models. Therefore, at each time there is a number of deformable models that are fitted to the image data and clearly there is a need for controlling the interaction between segmentation and fitting. Since the number of the models and data are changing over time we have a dynamic system. A discrete event system is a dynamical system in which changes in the state occur at discrete instances

of time. The Supervisory Control Theory of Discrete Event Systems developed by Ramadge and Wonham [232] allows us to encapsulate both the discrete and the continuous aspects of the fitting and segmentation processes. In addition, it allows us to formulate in a compact and elegant way the basic concepts and algorithms for our problem. Therefore, the addition of a new deformable model in the system can be done in a modular and hierarchical fashion. Using the DES framework allows us to decompose the overall design into components modeled by Finite State Machines (FSM) and provides efficient methods to investigate the behavior of the participating processes which operate concurrently.

We first describe the components of the system that implements PSA in terms of the DES theory. There are three components to the system: the *plant*, the *observer(s)* and the *supervisor* (Fig. 12.11(a)). The plant is the dynamic system that we want to control, the supervisor controls the plant based on information that receives from the observers and the observers are processes that monitor the behavior of the dynamic system. For our system (Fig. 12.11(b)), the set of all deformable models that are being fitted to the image data at each time instant constitutes the *plant*. The observers are processes that monitor the quantitative changes to the shape of the modeling primitives. Each observer, depending on which of the criteria that are associated with the PSA is satisfied, sends the corresponding message to the *supervisor*. The supervisor is the process that controls the behavior of the system, handles the complex interactions between the fitting processes and invokes the appropriate part composition or decomposition algorithms depending on the input from the observers. Therefore, the observers provide feedback to the supervisor regarding the state of the plant. The three components of the system (Fig. 12.11(b)) are communicating through a set of channels $CC = \{C_i, M_i, O_i\}$, where $i = \{1, \dots, n\}$, n is the number of deformable models used to describe the image data at each time instant, C_i is the set of channels through which the supervisor sends control commands to each fitting process (each of the deformable models), M_i is the set of channels through which each fitting process sends messages to the supervisor, and O_i is the set of channels through which the observers are sending messages to the supervisor. A detailed treatment of the subject is offered in [133].

12.4 TWO-DIMENSIONAL HUMAN BODY MODEL ACQUISITION

In this section, we present a technique for building a two-dimensional model of the body of a subject who performs a set of controlled movements [131]. Our goal is to automatically segment body apparent contours of moving humans and estimate the shape of a subject's body parts without assuming a prior model of the human body or a body part segmentation. Therefore, we use a sequence of images in which a subject performs a set of movements according to a generic, subject invariant protocol that reveals the structure of the human

body and ensures that all the major parts of the human body become visible and we employ the PSA.

Assuming that we observe the motion of the subject using a camera whose image plane is parallel to the sagittal plane (Fig. 12.13(a-d)), one possible set of movements that will lead to the identification of the shape and connectivity of the subject's body parts is described below. The protocol of movements differs depending on the viewing position with respect to the subject. Therefore, if the initial orientation of the subject with respect to the camera is different from the one described above, a different protocol is needed. All the movements start and end in the position where the body is erect and the arms are placed straight against the sides of the body facing medially (*first reference position*).

Protocol of Movements: MovA

- 1. Stand still:** The subject is requested to stand still for few moments to acquire the apparent contour.
- 2. Head movement:** The subject tilts her head backwards as far as possible.
- 3. Left upper body extremities' movements:** In the first phase of the movement, the subject lifts her left arm to the front until the arm reaches the horizontal position. Continuing from this position to the second phase, the subject rotates her hand so that the palm is facing upwards and she flexes her wrist. Then, the subject bends her elbow, bringing the forearm to the vertical position.
- 4. Right upper body extremities' movements:** The subject lifts her left arm backwards to a comfortable position, in which the arm is not fully occluded by the torso. Then, the subject performs the left upper body extremities' movements using her right arm.
- 5. Lower body extremities' movements:** This movement consists of two phases. In the first phase, the subject extends her left leg to the front. When the leg reaches the maximum comfortable position, the subject flexes her foot and then she bends the knee. In the second phase, from the position where her left leg is extended, the subject steps forward and raises her right leg. Again, when that leg reaches a comfortable position, the subject flexes her foot and then she bends the knee.

We assume initially that the human body under observation consists of a single part. A single deformable model is fitted to time-varying image data. As the subject moves his/her body parts, the geometry of the apparent contour of the body dynamically changes. Since the human body is comprised from several parts which are nominally rigid and move relative to each other then, assuming that we observe only one human, the changes in the apparent contour can be attributed to the event that parts of the body which initially occlude each other become unoccluded as they move. The result of processing the image data from the MovA set of movements using the PSA Algorithm will be the two-dimensional shape models of the subject's major body parts and their connectivity. We denote the structure of a subject's body by a graph in which there is a node for each body part that specifies the geometric model of the part. Arcs between nodes denote that a joint exists between body parts and each one allows a range of positions between the two parts. In particular, the

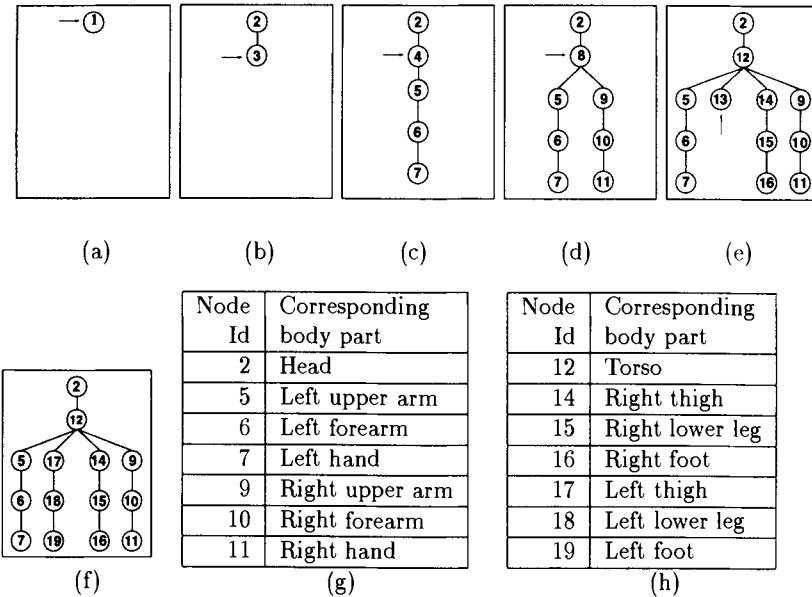


Figure 12.12 (a) Initially the graph of parts consists of one node. The arrow denotes the node that has been refined in the next iteration. (b) At the completion of the head movement the graph consists of one node for the head and one for the rest of the body. (c) At the end of the movement of the left arm, the node for the rest of the body will be refined to consist of one node for the left wrist, one for the left forearm, one for the left upper arm and one for the rest of the body. (d) Similarly, at the end of the movement of the right hand, the graph G will contain nodes for the right upper body extremities. (e) At the end of the movement of the left leg, G will contain nodes for the lower body extremities (left). (f) Depicts the nodes of the graph at the end of all pre-specified movements, and (g-h) list the body parts that the nodes represent.

algorithm detects the parts that correspond to the head, torso, left and right upper arm, left and right forearm, left and right hand, left and right thigh, left and right lower leg, and left and right foot (Fig. 12.12). In section 12.7 (in the sixth experiment in particular), we provide the details of applying the PSA to the image data from the MovA set of movements.

It should be noted that this algorithm applies to humans of any anthropometric dimension because instead of using templates, which are domain specific, we concentrate our effort in specifying general criteria sufficient to analyze and to segment the model of the deforming apparent contours from multi-part objects.

12.5 THREE-DIMENSIONAL HUMAN BODY MODEL ACQUISITION

In this section, we demonstrate the method for combining two-dimensional information from multiple views in order to estimate the three-dimensional shape

of a subject's body parts [131]. In particular, we present a new algorithm which selectively integrates the (segmented by the PSA) two-dimensional apparent contours from three mutually orthogonal views (Fig. 12.13(a)). The initial pose of the subject with respect to the cameras is such that the image plane of the first camera is parallel to the sagittal plane of the subject (Fig. 12.13(b)), the image plane of the second camera is parallel to his/her coronal plane (Fig. 12.13(c)) and the third camera overlooks the scene (Fig. 12.13(d)) and it is parallel to the transverse plane of the subject. Due to the complexity of the human's body shape, and due to occlusion among the body parts, the subject is requested to perform a set of movements according to the protocol MovB that incrementally reveals the structure of the body and allows the integration of information from multiple views.

The movements are performed in sequence. Each movement starts at the end of the previous one. At the end of each movement the subject remains still to signal the end of movement.

Protocol of Movements: **MovB**

1. **Stand still:** The subject just stands still for few moments, in order for the system to acquire the apparent contour.
2. **Tilt Head Back:** The subject tilts her head backward as far as possible and then she returns to the first reference position.
3. **Lift arms:** The subject lifts her arms to the side until the arms reach the horizontal position (that is the *second reference position*).
4. **Flex wrists:** The subject flexes her wrists downwards as far as possible and then she returns to the second reference position.
5. **Bend elbows:** The subject bends her elbows, bringing the forearms to the vertical position and then she returns to the second reference position.
6. **Extend legs side:** The subject extends her left leg to the side and then she returns to the second reference position. Next, she extends her right leg to the side and then she returns to the second reference position.
7. **Extent legs front:** The subject extends her left leg to the front and then she returns to the second reference position. Next, she extends her right leg to the front and then she returns to the second reference position.
8. **Flex left leg:** The subject flexes her left foot, she bends her left knee and then she returns to the second reference position.
9. **Flex right leg:** The subject flexes her right foot, she bends her right knee and then she returns to the second reference position.

Our approach is to first build a single three-dimensional model of the human standing, and then to incrementally refine it by extracting the three-dimensional models of the different parts as they become visible to the different views. At each stage of the algorithm, the image data from a specific movement are processed. The stages of the algorithm are summarized in table 12.1.

For each movement, the apparent contours from each active view are fitted using the techniques described in the previous section. Therefore, for each active view there is a set of two-dimensional deformable models that fit the corresponding image data. Due to occlusion, the number of deformable models

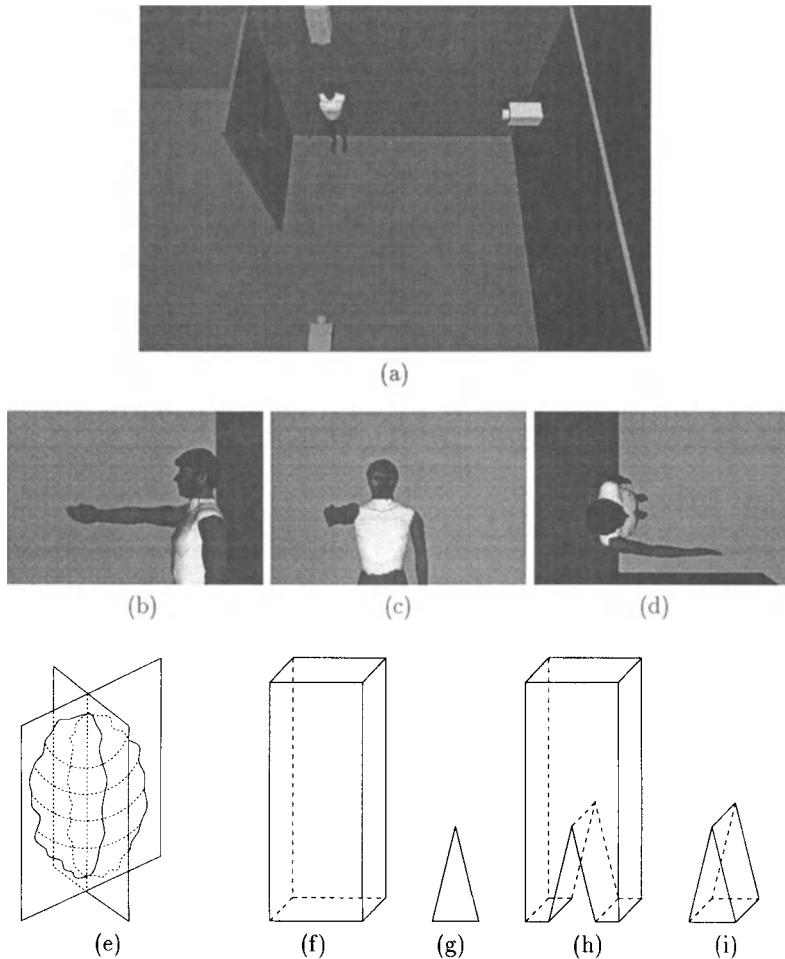


Figure 12.13 (a) The movements of the subject are observed by three cameras placed orthogonally to each other, (b) view from the side camera, (c) view from the front camera, (d) view from the top camera, (e) results of the Integration algorithm, and (c-f) results of the Intersection algorithm for a simple example. In particular, assuming that (f) is the initial three-dimensional shape of a part, (g) is the two-dimensional model of the apparent contour of one of its sub-parts, then (h,i) are the three-dimensional shapes of the part and the sub-part, respectively.

Step	Movement Sub - Sequence	Active Views			Shape Estimation	3D Parts Acquired
		Side	Front	Top		
1	Stand still	✓	✓		IG	
2	Tilt head back	✓			IS	Head
3	Lift arms	✓		✓	IG	Arms
4	Flex wrists	✓			IS	Hands
5	Bend elbows	✓			IS	Upper and lower arms
6	Extend legs side		✓			
7	Extent legs front	✓			IG	Legs and torso
8	Flex left leg	✓			IS	Left thigh, left lower leg and left foot
9	Flex light leg	✓			IS	Right thigh, right lower leg and right foot

Table 12.1 Specification of the active views during each movement, the method employed (IG stands for the Integration algorithm and IS for the Intersection algorithm) for the estimation of the three-dimensional shape of the parts, and which are the parts whose three-dimensional shape is obtained.

in each of the views may not be the same. Consequently, depending on the type of movement a new part or parts may be detected in some of the two-dimensional apparent contours. The new part is either a previously unseen body part or a subpart of a part whose three-dimensional model has already been recovered. In the first case, the two-dimensional models of the corresponding apparent contours of a part are integrated using the algorithm described below (see also Fig. 12.13(e)). In the second case, the model of the apparent contour of the subpart is intersected with the model for the three-dimensional shape of the part to obtain two new three-dimensional shapes (see also Figs. 12.13(f-i)), the three-dimensional shape of the sub-part and the three-dimensional shape of the rest of the part.

Specifications: Intersection Algorithm (IS)	
Input:	<i>A 3D model of the shape of a part, the 2D model of the apparent contour of one of its sub-parts, the spatial relation of the model-centered reference frame of the sub-part with respect to the reference frame of the part.</i>
Output:	<i>A 3D model of the shape of the sub-part and a 3D model of the shape of the rest of the part.</i>

In the following, the algorithm for the integration of the two-dimensional models of two apparent contours is described. The input to the algorithm is the two-dimensional models of the apparent contours of the part as observed from two mutually orthogonal views, and the spatial relation between the views. A three-dimensional deformable model is initialized and the nodes that lie on its

meridians whose plane is parallel to the planes of the apparent contours are fitted to the nodes of the two-dimensional models. Local deformations are employed to capture the exact shape of the two-dimensional models. The rest of the three-dimensional shape is interpolated. Due to the fact that the local deformations may be large in magnitude and to avoid shape discontinuities between the fitted nodes and the rest of the shape, a thin-plate deformation energy is imposed during the fitting process. Therefore, as shown in Fig. 12.13(e), the deformable model fits accurately the two-dimensional models, while its shape in between the apparent contours (in the absence of any image data) varies smoothly.

	Specifications: Integration Algorithm (IG)
Input:	<i>The 2D models of the apparent contours of a part as observed from two orthogonal views, and the spatial relation between the views.</i>
Output:	<i>A 3D model of the shape of the part.</i>

It should be noted that the three-dimensional shape of the new part is obtained at the end of the appropriate set of movements. This is a desirable solution, since the shape fitting is done in 2D continuously and the three-dimensional shape estimation is done only once, making it computationally efficient.

12.6 HUMAN BODY TRACKING

In this section, we present a formal framework for tracking the motion of a subject's body parts based on information extracted from multiple cameras [132].

To alleviate the problem of degenerate views and severe occlusion from a specific view, we employ three calibrated cameras placed in a mutually orthogonal configuration. Since we are using multiple cameras one of the questions we are called upon to answer is how many views should we employ. Should we process the occluding contours from all the cameras or from only one? Using portions from all three of them simultaneously to compute forces on a particular part can result into an incorrect solution due to incorrect association of contour points to model points. Therefore, at every image frame and for each part, we derive a subset of the cameras that provide the most informative views for tracking. In particular, we decide how many of the three cameras to use (one, two or all three) based on two criteria. The first criterion checks the visibility of a subject's body part from the particular camera and the second checks the observability of its motion w.r.t the particular camera (these criteria are presented in detail in [132]). For a part i and a camera C_k , $k \in \{1, 2, 3\}$, we define as visibility index $VI_{i,k}$ the ratio of the area of the visible projection of the part i to the image plane of camera C_k to its projected area when we do not take into account the occlusions. According to our definition, a part i is considered visible from camera k if $VI_{i,k} \geq 0.2$. Once a part is considered visible from

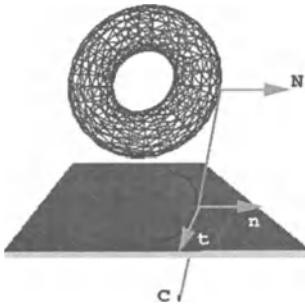


Figure 12.14 Relation of a point on the occluding contour of an object to its corresponding point on the object’s surface.

a particular camera, we further check if the camera’s viewpoint is degenerate given the predicted motion of the part (based on a Kalman Filter [132]). Let \mathcal{P}_i be the set of nodes that belong to the occluding contour of a part i and let p_i be the cardinality of the set \mathcal{P}_i . Also, let ${}^I\dot{\mathbf{x}}_{i,j}$ be the change to image coordinates of these nodes due to predicted motion. Then, we rank the cameras based on the quantity $\|\sum_{k=1}^{p_i} {}^I\dot{\mathbf{x}}_{i,k}\|$.

At each step, based on the above two criteria, we select a set of cameras and the associated occluding contours to provide constraints to the estimation of the rotational and translational motion of each of the parts.

In the PBM framework, image datapoints apply external forces to the deformable model. However, the algorithm has to decide for each image datapoint the model node on which it should apply forces to. To accomplish this goal, we invoke a theorem from projective geometry that relates points on the occluding contour of an object to points on the surface of the object itself (see also [84, 128, 156]). More specifically, if ${}^\Phi\mathbf{x}_{i,j}$ is a point on the surface of a part whose projection point ${}^I\mathbf{x}_{i,j}$ on the image plane lies on the occluding contour of the part, then the part’s normal \mathbf{N} at point ${}^\Phi\mathbf{x}_{i,j}$ is parallel to the vector \mathbf{n} . This vector, \mathbf{n} , is normal to the plane defined by the origin of the camera coordinate system, the point ${}^I\mathbf{x}_{i,j}$ and the tangent of the occluding contour at the point ${}^I\mathbf{x}_{i,j}$, that is $\mathbf{N} \cdot \mathbf{n} = 1$. Based on the above theorem, we can establish correspondences between the occluding contour of a part and the projection of its model (see Fig. 12.14).

Since we know the shape model of the object prior to initializing the tracking, we compute the normal $\mathbf{n}_{i,j}$ (for simplicity \mathbf{n}_j) at each node j of the model’s part i . Also, to characterize the variation of the normal over the model, we

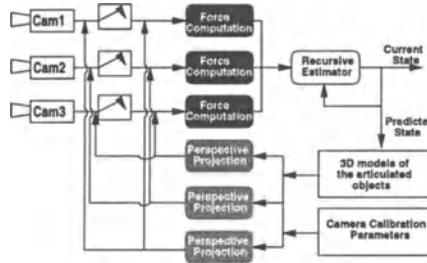


Figure 12.15 Flow diagram of the 3D tracking algorithm.

compute for each node m of the model, the quantity

$$\epsilon_m = \min_{k \in \mathcal{N}_i} (\mathbf{n}_m \cdot \mathbf{n}_k), \quad (12.22)$$

where \mathcal{N}_i is the set of nodes neighboring node m . Therefore, the variation of the normal of the model i is $\epsilon_i = \sum_{m=1}^n \frac{1}{n} \epsilon_m$, where n is the number of the nodes of the model of part i .

Specifically, assuming that the initial pose of the parts is known, for every occluding contour we employ the algorithm below:

Algorithm: Force Assignment

Step 1: Fit a two-dimensional deformable model to the occluding contour to obtain a smooth differentiable model of the curve. At every point \mathbf{z} of the occluding contour, we compute the tangent vector ${}^I\mathbf{t}_z$ on the image plane, and the vector ${}^C\mathbf{k}_z$ which is normal to the plane defined by the origin of the camera coordinate system C , point \mathbf{z} and ${}^I\mathbf{t}_z$.

Step 2: Determine the set of nodes \mathbf{S}_i of the tessellated shape model for the part i whose normal ${}^{\phi_i}\mathbf{n}_{i,j}$ (as expressed in the model frame ϕ_i), satisfies the relationship

$${}^{\phi_i}\mathbf{n}_{i,j} \cdot {}^{\phi_i}\mathbf{k}_z \geq \epsilon_i. \quad (12.23)$$

Step 3: Based on the current position of the shape model (for the first frame) or the predicted position of the model (for the rest of the frames) compute the projection of the nodes \mathbf{S}_i of the model to the image plane. The point \mathbf{z} of the occluding contour will apply forces to the node j of the model which is a member of the set \mathbf{S}_i and whose Euclidean distance from the node is the smallest.

Step 3: The force that the point \mathbf{z} applies to the node j has two components:

$$\mathbf{f}_{im} = (\mathbf{z} - {}^I\mathbf{x}_{i,j}), \quad (12.24)$$

$$\mathbf{f}_{3D} = {}^C\mathbf{n}_{i,j} - {}^C\mathbf{k}_z. \quad (12.25)$$

These two forces are then converted into a generalized force \mathbf{f}_q as follows:

$$\mathbf{f}_q = \mathbf{L}_p^T \mathbf{f}_{im} + \mathbf{L}^T \mathbf{f}_{3D}, \quad (12.26)$$

where \mathbf{L}_p is the Jacobian matrix for the case of perspective projection. After we compute the force assignments for all the points at the occluding contour, we estimate the new pose parameters of the part (see Fig. 12.15) based on the extended Kalman filter [132].

12.7 EXPERIMENTAL RESULTS

This section presents selected experiments from the ones that were carried out on real image sequences in order to evaluate the effectiveness of the proposed approach in general and of the specific algorithms in particular.

At the GRASP Lab² we have constructed a testbed (*3D studio*) for an integrated approach to human body analysis and for the design and prototyping of rehabilitation aids. The goal of the 3D studio is to capture in a non-intrusive, quick, and robust fashion the static and dynamic parameters of human appendages for analysis and design. Applications include, among other, prosthetic design and manufacture, and interactions with virtual environment. The imaging hardware of the 3D-studio consists of three calibrated XC-77RR gray-scale CCD SONY cameras and a network of DSP modules based on the Texas Instruments TMS320C40 digital signal processor. The four DSP modules form a MIMD architecture that is capable of digitizing images at interactive rates.

In all the experiments, the region of interest has been obtained by subtracting the current image from the known background and the outlines have been obtained by applying a variation of the Canny edge detector [43] to the input image sequence. Edge following is employed to detect the presence of a closed contour within the apparent contour, which we referred to as a hole within the apparent contour. The values of the a priori constants for the PSA are the following: $K_{A_1} = 0.5 \min(a_1, a_2)$, $K_{A_2} = 2$ pixels, $K_{B_1} = 15\text{deg}$, $K_{B_2} = 0.2\psi(t_{\text{init}})$, $K_{F_1} = 10$ pixels and $K_{F_2} = 15\text{deg}$. These values were experimentally determined and remained the same for all the experiments. The local deformation stiffness parameters of the initial model were set to $w_0 = 0.1$ and $w_1 = 0.2$, and the time step for the Euler method was 10^{-6}s .

The first experiment was designed to evaluate the performance of the Part Decomposition Algorithm A. In this experiment, we observe the motion of removing the cap from a pen. Figs. 12.16(a-f) show six frames from the image sequence. Figs. 12.16(ra-rf) depict the fitted deformable model to the corresponding image data during the movement. In Fig. 12.16(rf) the Part Decomposition Criterion A is satisfied and the Part Decomposition A algorithm is invoked. Fig. 12.16(rg) shows the two new models that are employed to fit the data.

²General Robotics and Active Sensory Perception Laboratory, University of Pennsylvania.

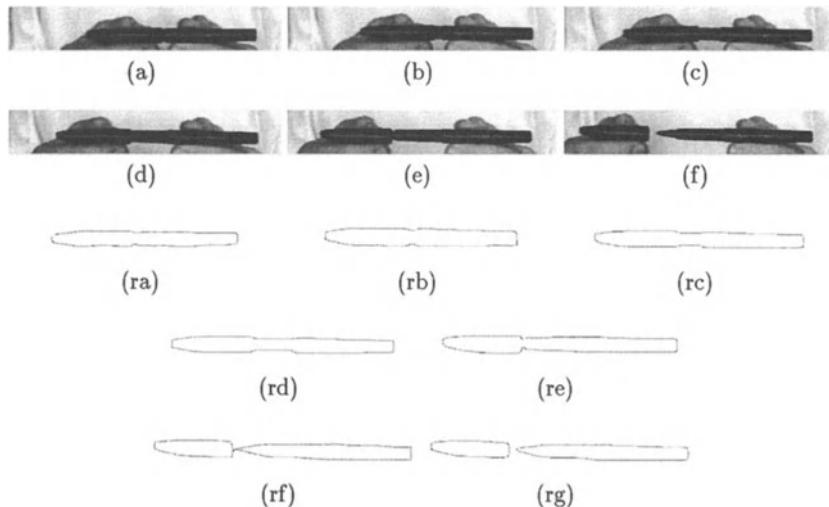
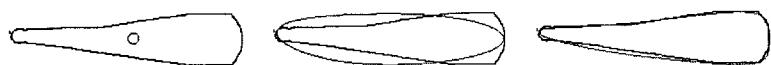


Figure 12.16 Part segmentation, shape and motion estimation of a pen and its cap. (a-e) A sample of the image sequence. (ra-rg) Evolution of fitting. (rg) The models of the two parts.

The second experiment was designed to evaluate the performance of the Part Decomposition Algorithm B. In this experiment, we observe the planar motion of a robot arm, which is composed of two parts. Fig. 12.7 shows five frames from the image sequence. Figs. 12.17(a-l) depict both the image data and the deformable model or models at each stage of the fitting. Figs. 12.17(a-c) show the initialization of a deformable model at the center of gravity of the data, an intermediate stage in the fitting and the finally fitted model to the first frame using only global deformations, including tapering and bending. Fig. 12.17(d) shows the model fitted to a subsequent image frame where bending becomes apparent. In Fig. 12.17(e) the model fitted to a subsequent image frame, satisfies the Part Decomposition Criterion B and a hypothesis is generated that the object is comprised from two parts. Figs. 12.17(f-h) demonstrate the fitting of the two new models to the image data. Fig. 12.17(f) shows the two new models initialized to the fixed and relocation zone of the initial model. The orphan datapoints, exert forces to the new models based on our algorithm for all neighbors force assignment. Fig. 12.17(g) shows an intermediate step in the fitting process, while Fig. 12.17(h) shows the finally fitted models. The overlap between the two models allows to compute the location of the joint over several frames. Fig. 12.17(i) shows the models prior to fitting the data from a subsequent frame at the image sequence. Figs. 12.17(j-k) demonstrate the models fitted to data from another two frames from the forward motion of the robot arm. Finally, Fig. 12.17(l) shows the two models fitted to the final frame, when the robot arm moved backwards, to the original position.



(a)

(b)

(c)



(d)

(e)

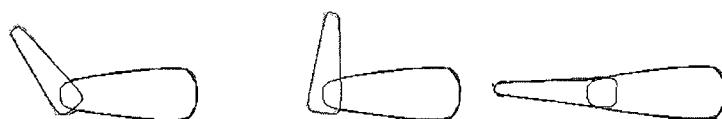
(f)



(g)

(h)

(i)



(j)

(k)

(l)

Figure 12.17 Two-dimensional part segmentation, shape and motion estimation of a robot arm.

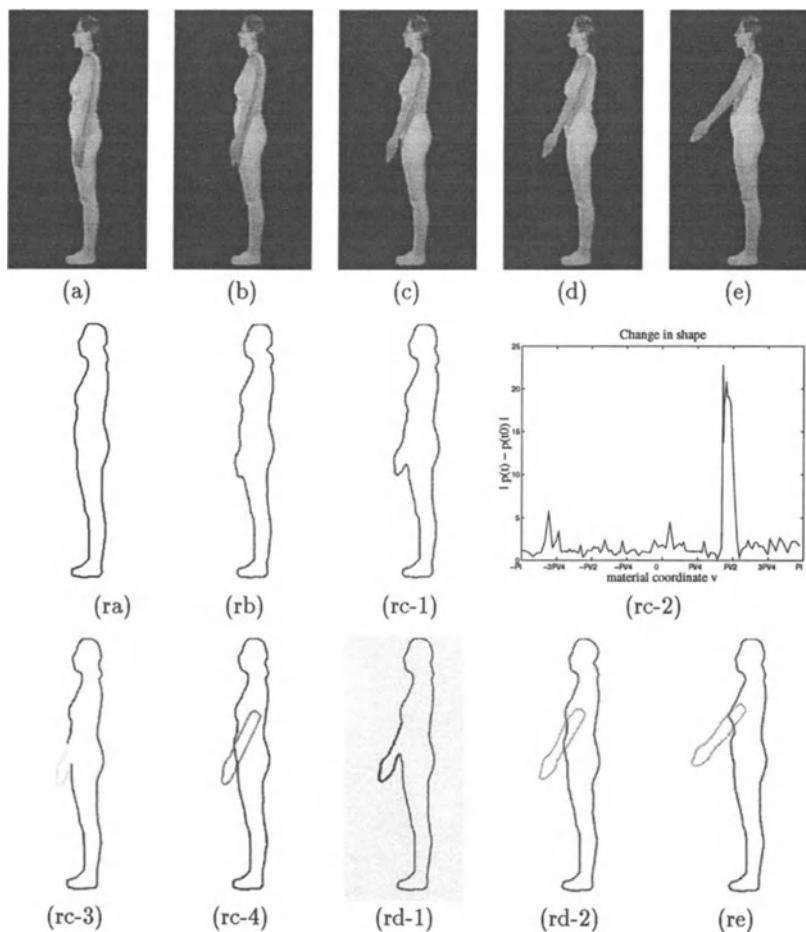


Figure 12.18 Two-dimensional part segmentation, shape and motion estimation of a human arm.

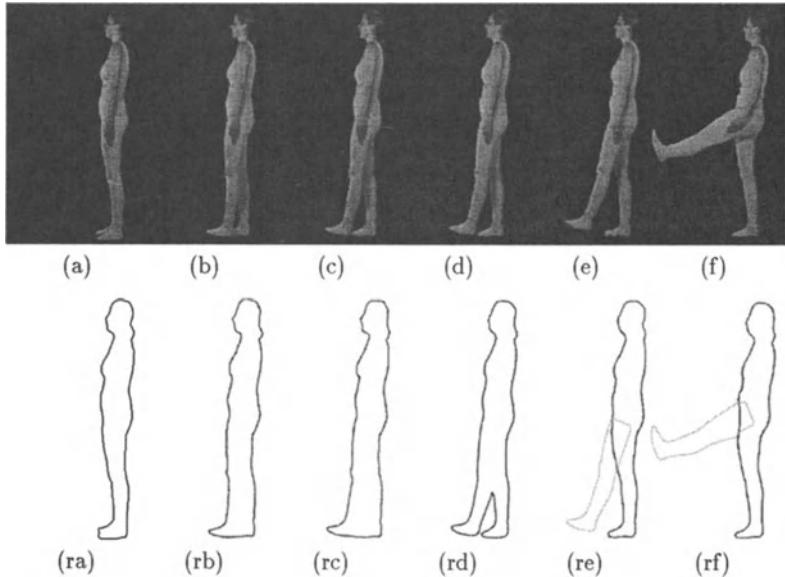


Figure 12.19 Two-dimensional part segmentation, shape and motion estimation of a human leg.

The third experiment was designed to evaluate the performance of the Part Composition Invocation Criterion and the performance of the Part Decomposition F algorithm. In this experiment, the subject, starting from the first reference position, lifted her left arm until the arm reached the horizontal position. Figs. 12.18(a-e) show five frames from the image sequence. Fig. 12.18(ra) shows the fitted model to the first frame using global and local deformations. Figs. 12.18(rb,rc-1) show the fitting of the apparent contours in Figs. 12.18(b,c), respectively. In Fig. 12.18(rc-1) notice the deformation from the initial shape of the model which is quantified in Fig. 12.18(rc-2). At this frame, the Parametric Composition Invocation Criterion was satisfied and therefore the Parametric Composition algorithm was invoked. The composed model, depicted in Fig. 12.18(rc-3) is fitted to the data in Fig. 12.18(c), and its defining primitives are shown in Fig. 12.18(rc-4). Fig. 12.18(rd-1) shows the fitting of the composed model to the data in Fig. 12.18(d). In this frame, the Part Decomposition Criterion F was satisfied and the Part Decomposition F algorithm was invoked in order to recover the underlying parts. Fig. 12.18(rd-2) shows the recovered models for the parts, while Fig. 12.18(re) shows the fitting results of these recovered models to the data in Fig. 12.18(e).

The fourth experiment was designed to demonstrate the Part Decomposition Criterion D and the corresponding algorithm. In the case that parts of an articulated assembly are totally occluded at the initial position, as the parts move relative to each other and become gradually unoccluded, a hole might evolve within their apparent contour. In this experiment, the subject starting from

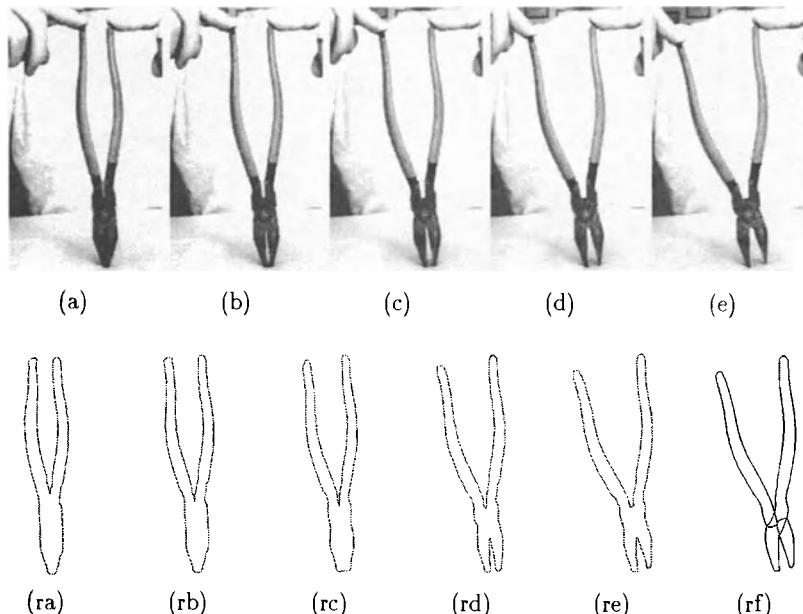


Figure 12.20 Two-dimensional part segmentation, shape and motion estimation of pliers. (a-e) A sample of the image sequence, (ra-re) fitting results.

the first reference position, extended her left leg to the front until she reached a comfortable position. Figs. 12.19(a-f) show six frames from the image sequence. Figs. 12.19(ra-rd) depict the models fitted to the data in Figs. 12.19(a-d). While fitting the data in Fig. 12.19(e) the Part Decomposition Criterion D was satisfied and therefore the Part Decomposition D algorithm was invoked. Fig. 12.19(re) depicts the two models recovered, while Fig. 12.19(rf) depicts the results of fitting these models to the data in Fig. 12.19(f).

The fifth experiment was designed to evaluate the performance of the Part Decomposition Algorithm E. In this experiment, we observe the motion of one of the handles of a pair of pliers. Figs. 12.20(a-e) show five frames from the image sequence. Figs. 12.20(ra-re) depict the fitted deformable model to the corresponding image data during the movement. The Part Decomposition Criterion E is satisfied for the deformable model depicted in Fig. 12.20(re). Then, the Part Decomposition E algorithm was employed and it recovered two areas that correspond to moving parts. These are the areas in which dynamic deformation of the apparent contour was observed. Fig. 12.20(rf) shows the three models that are employed to fit the data. The first deformable model, models the data from the left handle of the pair of pliers, the second models the data from the right jaw and the third models the data from both the right handle and left jaw. The data from the right handle and left jaw are modeled together, since they remained unmoved and therefore were considered to be one part.

The purpose of the sixth experiment was to demonstrate the results of the PSA when applied to image data from a subject which moves according to the protocol MovA (section 12.4 Figs. 12.21(a,b) show two frames from the movement where the subject tilted her head back. Initially, when the subject was standing still at the first reference position, only one deformable model was used to fit the data (Fig. 12.21(c)). During the head movement the Part Decomposition Criterion B is satisfied therefore two models (one for the head and one for the rest of the body) are recovered as depicted in Figs. 12.21(d-e). As the subject lifted her left arm, the shape of the left upper body extremities was estimated (Fig. 12.21(f)). First, the Part Composition Invocation Criterion was satisfied and then the Part Decomposition Criterion F. Since the arm is kept straight during the movement, the algorithm cannot identify the hand, forearm and upper arm as distinct body parts. However, when the subject flexed the wrist and bent the elbow, the models for these parts were estimated as shown in Figs. 12.21(g-h) (since the Part Decomposition Criterion B was satisfied during the movement of these parts). If instead of lifting her arm the subject lifts her leg, then the model for the left lower body extremities is recovered as shown in Fig. 12.21(m) (since the Part Decomposition Criterion D is satisfied). When the subject flexed the foot (Figs. 12.21(i-j)) and bent the knee (Figs. 12.21(k-l)), this model was replaced by three models; one for the left foot, one for the left lower leg and one for the left thigh as shown in Figs. 12.21(n-q) (the Part Decomposition Criterion B is satisfied for all these cases). Fig. 12.21(r) depicts these models when the subject returned to the first reference position.

The seventh experiment was designed to demonstrate the feasibility of estimating the three-dimensional shape of body parts by selectively integrating the (segmented by PSA) two-dimensional apparent contours from multiple views. First, the two-dimensional models for the apparent contours from the side and front views were integrated to obtain a single three-dimensional model of a subject's body. During the *tilt-the-head-back* movement a new part (which corresponds to the head) was identified and its two-dimensional shape was estimated. Since the new part belongs to an articulated assembly whose three-dimensional shape has already been estimated (the single three-dimensional model of the subject standing) the intersection algorithm was invoked to estimate the shape of the new part. Then the subject lifted her left arm. When the arm reached the horizontal position, the integration algorithm was invoked to estimate the shape of the left upper body extremities. The inputs to this algorithm were the two-dimensional models of the segmented apparent contours from the side and top view. Figs. 12.22(a-d) show several views of the recovered left arm of the subject, while Figs. 12.22(e-g) show the models for the hand, forearm and upper arm, respectively. Fig. 12.22(h) shows the spatial layout of these parts. Figs. 12.22(i-k) show three views of the recovered model for the torso. Finally, Figs. 12.22(l-q) show several views and the corresponding parts of the left leg of the subject *Julie*.

Finally, in the last experiment we acquired images of a subject moving his arms in 3D. To simplify the acquisition of the occluding contours, we designed the background to be a simple one. The process of tracking the motion of the sub-

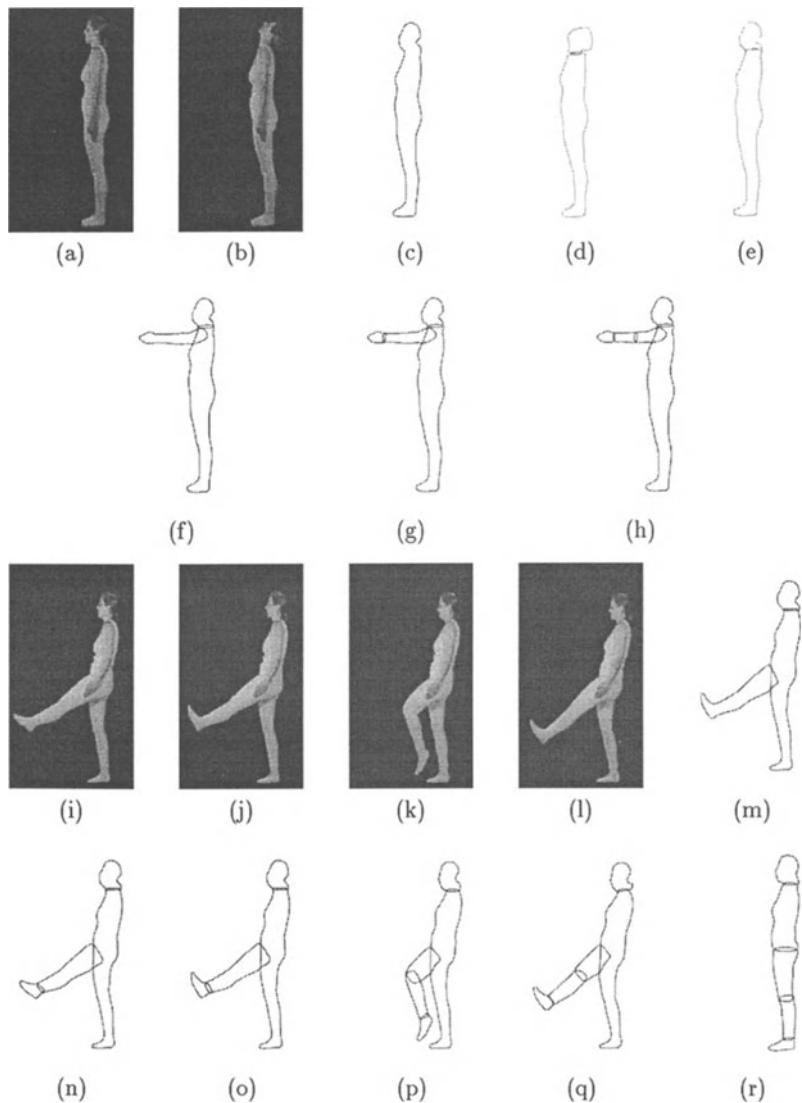


Figure 12.21 Two-dimensional human body model acquisition (subject *Julie*).

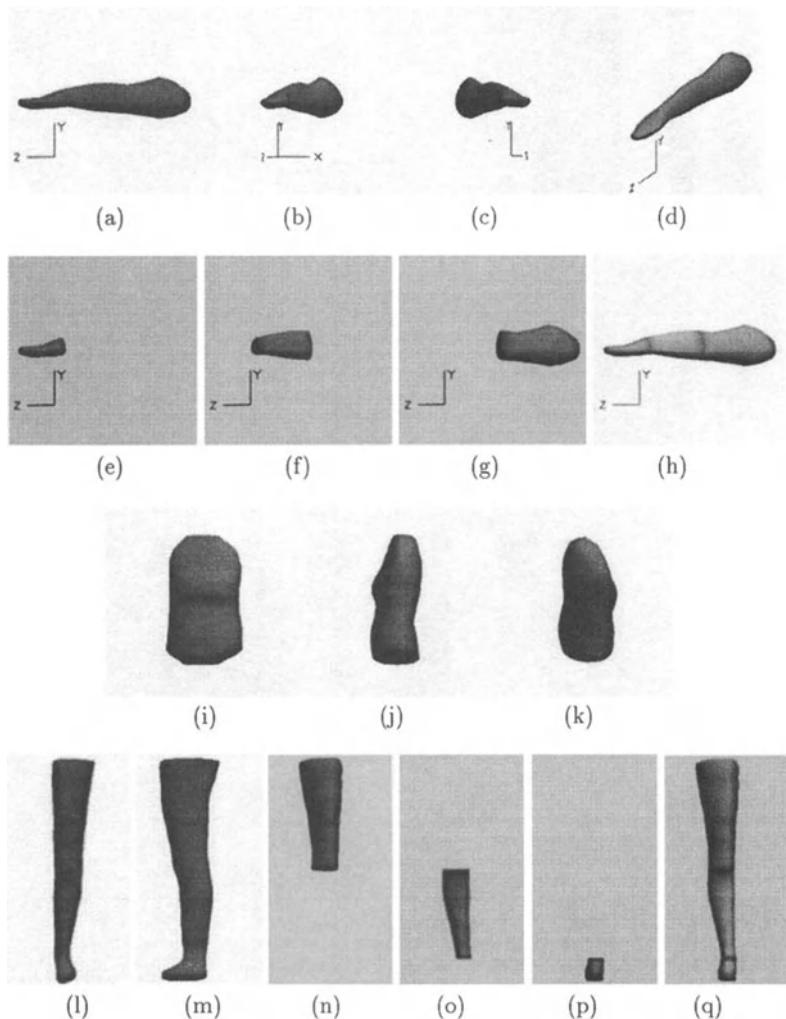


Figure 12.22 Three-dimensional human body model acquisition (subject Julie).

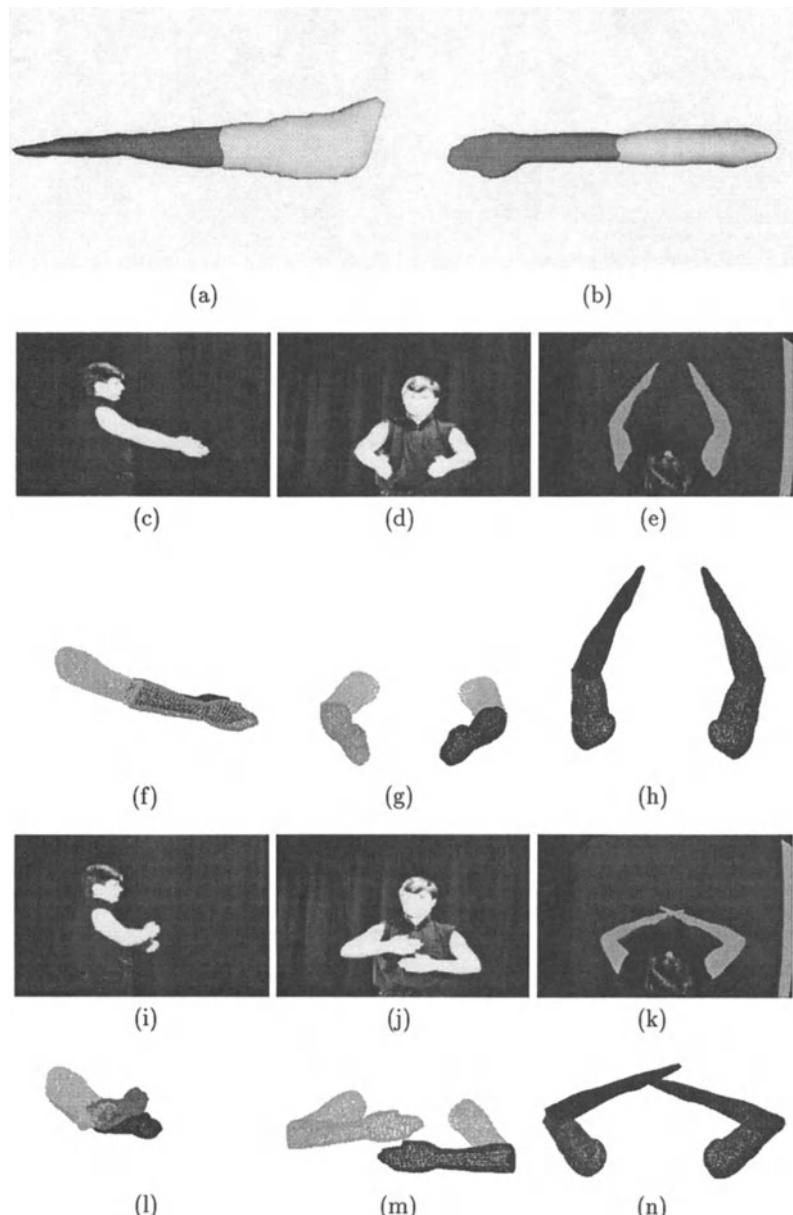


Figure 12.23 (a) Side view and (b) top view of the estimated three-dimensional models of the parts of subject's arm, (c-e,i-k) sample input images from the three cameras for frames 1 and 16 in the motion sequence, and (f-h,l-n) tracking results for frames 1 and 16 in the motion sequence.

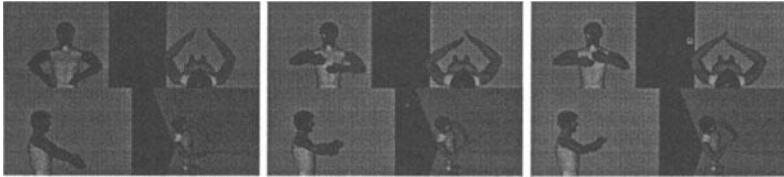


Figure 12.24 The estimated motion parameters for the motion of the subject's arms have been applied to a graphics model of the subject (three frames from four different views).

ject's arms consists of two phases. The first phase is the model building phase. The subject was asked to perform a set of motions to acquire the anthropometric dimensions of its (left and right) forearm and of its (left and right) upper arm. The subject placed himself in the viewing volume of the cameras and assumed a position where the body is erect and the arms are placed straight against the sides of the body facing medially. After standing still for a few seconds in order to acquire the initial apparent contour, the subject lifted his arms to the front until the arms reached the horizontal position. Continuing from this position, the subject rotated his hands so that the palms are facing upwards and bent the elbows, bringing the forearms to the vertical position. By employing the PSA, we extracted the three-dimensional shape of the forearms and of the upper arms of the subject. Notice that since the subject did not flex his wrists, the estimated models for the forearms (Fig. 12.23(a,b)), include the wrists. In the second phase, we asked the subject to move his arms in 3D. Figs. 12.23(c-e,i-k) show sample input images from the three cameras. Figs. 12.23(f-h,l-n) show the estimated 3D pose of the subject's parts of the arm. Fig. 12.24 shows four views for three frames of an animation. The animation was created by applying the estimated motion parameters to a graphics model of the subject.

We have performed several experiments with subjects of different age and sex. In all cases, we were able to recover a shape model of the body parts requested. A detailed report on the accuracy of the shape acquisition and tracking is provided in [133]. The criteria presented in this chapter cover visual events that signal the existence of multiple parts when the observation of the movement is performed from a non-accidental view. For example, if we observe the opening of a subject's fist from a viewpoint in which the fingers occlude each other and there is no relative motion between the fingers, then the algorithm will not signal the existence of multiple parts. Concerning the figure background segmentation, one alternative will be to employ the background subtraction algorithm proposed by Russel, Starner and Pentland [243]. The system is initialized by acquiring 40 images of the static background scene and the mean and variance of the red, green, blue and luminance values of each pixel location are computed. After initialization, each pixel is classified as part of the background based on four conditions and two thresholds. In addition, we assume that changes in the lighting do not affect the topology of the extracted figure

from the background. Therefore, any changes in the topology of the observed object are due to the existence of multiple parts.

The experiments that were carried out have proven that it is possible to determine a subject's body parts from an image sequence of the subject performing a set of controlled movements. Moreover, it is possible to estimate their shape in 3D. The three-dimensional shape estimation is accurate up to the integration of information from the three views. Therefore, if one wishes to obtain accurate anthropometric data, the subject should wear tight clothes.

VOLUMETRIC ANALYSIS OF THE LEFT VENTRICULAR WALL MOTION FROM MRI-SPAMM

Estimating the volumetric shape, motion and deformations of the left ventricle (LV) accurately, and in a clinically useful way, is a very important yet open research problem. Conventional cardiac imaging methods (e.g., MRI) still have many limitations, such as no explicit data correspondence between frames, and insufficient resolution of the extracted data. In addition, most of the existing models for the analysis of the LV shape and motion are based on the use of parameters that are either too complex or too few to be used by a physician.

Recently, a new magnetic resonance imaging (MRI) technique based on *magnetic tagging* (MRI-SPAMM) has been developed at the University of Pennsylvania for imaging of regional heart wall motion (Axel and Dougherty [9]). This fast, non-invasive technique promises to be very useful in the analysis of heart wall motion because it provides temporal correspondence of material points within the heart wall. This correspondence, in conjunction with the use of the three-dimensional (3D) location of each tagged datum, can subsequently be used as input to a motion analysis technique to extract the three dimensional left ventricular motion parameters. The motion parameters can then be statistically analyzed to explore their correlation with the various types of LV disease.

While there is some initial experience in the use of tagged MRI and related techniques to study the 3D motion of the heart, there is still no generally accepted method for analysis and display of the 3D heart motion. For example, Young and Axel [299] previously developed a finite element based method for reconstructing the 3D motion and deformation of the LV from MRI-SPAMM data. Although their method has provided a good local strain analysis, their model results in a large number of model parameters which must be non-trivially post-processed to provide meaningful geometric information on the nature of the deformation.

In an effort to overcome the limited clinical usefulness of most existing models for analyzing the LV, we have previously developed a new class of deformable 3D *surface* models whose deformations can be described with a small number of intuitive parameters that are functions (Park *et al.* [208, 209, 211]) instead

of constants. These parameter functions comprise an *intelligent* grouping into a small number of sets, of the many local parameters that are necessary to analyze the LV mid-wall. An example of a parameter function is longitudinal contraction, which is the set of parameters describing the contraction of the LV from the apex to the base. Based on this new model, we can analyze both locally and globally the shape and motion of the LV in a way that is readily understood by a physician.

We will first present a new class of **volumetric** deformable models that can analyze in a clinically useful way the volumetric motion of the LV. These volumetric models are a generalization of our previously defined **surface** models. Second, we present a new quantitative algorithm for the calculation of forces exerted from MRI-SPAMM data to our deformable model. The force calculation algorithm is based on the constrained physical motion of SPAMM datapoints in three dimensions. We combine forces from SPAMM datapoints originating at two sets of orthogonal planes, since only the in-plane motion of a SPAMM datapoint from a given set of planes can be estimated. In addition to forces from the SPAMM datapoints, we use forces from datapoints sampled from the inner and outer walls of the LV to estimate its shape. The combination of forces from boundary and SPAMM datapoints allows the extraction of the volumetric shape and motion of the LV. We also develop a new set of visualization tools which include parameter graphs, a 3D representation of the LV shape and motion, and the color-based mapping of the model's parameters on the 3D volumetric model. These tools allow the quantitative analysis of the LV shape and motion, and the visual representation of the analysis results in a clinically useful manner. Finally, based on the above new model we can quantitatively verify and visualize in 3D the knowledge about the LV that was qualitatively known to physicians.

13.1 RELATED WORK

There have been many model-based approaches to the analysis of the complicated motion of the LV, and the quantification of its measured deformation. Simple analytical models like spheres, ellipsoids or cylinders are often used. For example, Beyar and Sideman [29] constructed a mechanical model where the LV is modeled as a nested shell spheroidal to explain the effect of twisting motion; Arts *et al.* [6] used a thick-walled cylinder composed of eight concentric cylinder shells to describe the LV deformation; Kim *et al.* [144] used a thick-walled ellipsoidal model for the computation of the LV wall stress, and a truncated ellipsoidal model for simulation of regional stress; Azhari *et al.* [12] used a hollow conical shell (with a constant shear modulus) to characterize transmural twist motion; and Arts *et al.* [7, 8] developed a kinematic model by combining the deformation modes from a spherical, a prolate ellipsoidal and a cylindrical model.

However, the shape of the LV is neither spherical nor cylindrical. Even a prolate ellipsoid is a gross simplification of the shape of an LV. Therefore, as Guccione and McCulloch [106] pointed out, the analyses made by these models make simplifying assumptions about the material behavior of the heart muscle and the governing equations of motion.

Recently, techniques based on the use of deformable models for reconstructing the 3D *surface* shape and motion of the LV from CT or MRI data have been developed (e.g., Amini and Duncan [5]; Cohen and Cohen [52]; Huang and Goldgof [124]; McInerney and Terzopoulos [189]; Nastar and Ayache [199]; Shi *et al.* [256]). They use finite elements, spring-mass systems, deformation modes, bending and stretching thin-plate models, and other physics-based or geometric techniques. The main limitation of these techniques is that they do not provide intuitive motion parameters to model the rigid and non-rigid motion of the LV. Most of the techniques represent the LV motion as a set of local displacement vectors which either requires non-trivial post-processing to be useful to a physician or is only good for qualitative visualization (e.g., Amini and Duncan [5]; Chen and Huang [49]; Cohen and Cohen [52]; Huang and Goldgof [124]; McInerney and Terzopoulos [189]; Mishra and Goldgof [194]; Shi *et al.* [256]). On the other hand, models developed by Bardinet *et al.* [22] or Friboulet *et al.* [91] are formulated in terms of very few parameters that can offer only a gross approximation to the motion of the LV. And attempts to characterize the LV motion based on deformation modes do not provide a localization of the LV deformations and motion in a clinically useful manner, due to the global definition of each modal deformation (Nastar and Ayache, [199]; Pentland *et al.* [217]). Moreover, most techniques ignore the twisting or wringing motion of the LV known to occur during systole. To overcome the above limitations, we have previously developed a new class of surface deformable primitives whose global parameters are functions, and have demonstrated that the LV surface shape and motion can be accurately represented with these parameters that can be easily interpreted by physicians (Park *et al.* [208, 209, 211]). The input to these models were datapoints sampled from the mid-wall surface of the 3D finite element model of Young and Axel [299].

However, the LV motion cannot be captured entirely with surface models. To capture the LV shape and motion throughout its volume, we need a *volumetric* deformable model. Recently, techniques for analyzing the volumetric LV shape and motion from tagged MR image sequences have also been developed based on the use of 3D finite elements (Denney and Prince [67]; Moore *et al.* [195]; O'Dell *et al.* [205]; Young and Axel [299]). Such model-based approaches to the recovery of the volumetric shape and motion of the LV are necessary in order to overcome the limitation of current medical imaging modalities. Such modalities cannot provide explicitly, the time-varying 3D motion of material datapoints from the LV. The translation of the deforming heart through the fixed image planes puts a different cross section of myocardium into the image plane at each time point. Therefore the imaging techniques can provide only the two-dimensional motion of material points on the imaging plane at best.

Finite element modeling is a typical choice for volumetric motion analysis, since it provides strain analysis throughout the ventricular wall. However, the finite element representation does not directly lend itself to an understanding of the underlying kinematics in a clinically useful way. The parameters of the model are nodal displacements, resulting in a relatively large number of parameters, the physical interpretation of which can be difficult. The 3D strain tensor, for example, has three normal components and three shear components, each of which may vary with position in the wall. In order to understand the complex relationship between these components and other motion parameters, it is desirable to characterize the motion in terms of meaningful physical parameters that offer sufficient accuracy. In the remaining sections we will demonstrate how this can be achieved with our technique.

13.2 DATA EXTRACTION TECHNIQUES FOR CARDIAC MOTION STUDIES

Characterization of heart wall motion on a regional level is required to understand cardiac mechanics and the processes underlying a disease. In order to accurately measure heart wall motion, a number of material points must be located and tracked. Methods for providing intra-myocardial markers in the past have included the implantation of radiopaque markers (Arts *et al.* [8]; Brower *et al.* [41]; Harrison *et al.* [110]; Meier *et al.* [174]), lead beads (Waldman *et al.* [287]), or ultrasonic crystals (Rankin *et al.* [233]; Villarreal *et al.* [286]), use of naturally occurring landmarks (Chen and Huang [49]; Kim *et al.* [144]; Young [297]; Young *et al.* [298]), and magnetic resonance (MR) tagging (Axel and Dougherty [9]; Azhari *et al.* [12]; Denney and Prince [67]; Fischer *et al.* [86]; Moore *et al.* [195]; Zerhouni *et al.* [305]).

Although the implantation methods have been used for the LV motion analysis, and provide accurate localization, the invasive nature of the procedures does not allow a sufficient number of markers to be implanted for describing the LV geometry. Moreover, it poses the potential problem of local myocardium property alteration due to the introduction of foreign objects. On the other hand, the methods which utilize naturally occurring landmarks, like bifurcations of coronary arteries, do not require a surgery and can provide potentially many markers. However, intra-coronary injection of contrast medium is usually required to highlight the blood vessels in acquired images. When the blood supply is severely obstructed due to arterial occlusion, the tracing of the feature points around the region can be very difficult to achieve (Kim *et al.* [144]).

MR tagging has its advantages over the aforementioned approaches because a large number of material points may be marked and tracked during systole in a non-invasive manner. By locally perturbing the magnetization in tissue, one can create spatially encoded patterns such as star-bursts (e.g., Azhari *et al.* [12]; Denney and Prince [67]; Moore *et al.* [195]; Zerhouni *et al.* [305]) or grids

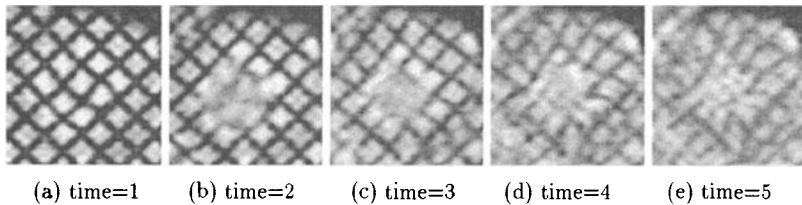


Figure 13.1 SPAMM images of a left ventricle during systole.

(e.g., Axel and Dougherty [9]; Fischer *et al.* [86]; Young and Axel [299]). Those patterns or magnetization tags are seen as dark regions in subsequent images (within a certain relaxation time T_1). As magnetization moves with tissue, the magnetization tags will move in the corresponding images, directly reflecting the motion of the underlying tissue and allowing us to follow the motion patterns within otherwise featureless structures such as the heart wall. One drawback of current MR tagging techniques, is that the tracking is possible only during systole or diastole at one time (i.e., not for a complete heart cycle), due to the decay of the magnetization signal over time.¹

13.2.1 Data Extraction Techniques based on the Derivation of Feature Points

Recently, curvature-based point correspondence recovery techniques have been proposed by researchers as an alternative to the above methods. The method by Goldgof *et al.* is based on the computation of the Gaussian curvature of a model that deforms based on the assumption of conformal motion (Kambhamettu and Goldgof [134]; Mishra and Goldgof [194]). The method of Amini and Duncan utilizes the potential energy of their bending and stretching model to estimate the curvature (Amini and Duncan [5]; Shi *et al.* [256]). Shi *et al.* [257] combined curvature extraction with Phase Velocity MRI, in an attempt to assess the transmural myocardial deformation in 2D. Friboulet *et al.* [92] demonstrated the stability of the Gaussian curvature computation in an experiment where the Gaussian curvature was computed through an iterative relaxation scheme from voxel-based surface rendering of CT left-ventricle volumetric data over a cardiac cycle. The derivation of point correspondences based on curvature has a potential in the sense that it can be widely used to datasets from many different medical imaging modalities and it can provide point correspondence over an entire heart cycle. But the technique is still under development.

13.2.2 Data Based on MRI-SPAMM

Even though our LV shape and motion estimation technique can be used with any type of LV data, we will apply it to data from MRI-SPAMM (Axel and

¹The weakening of the magnetization signal over time can be observed in Fig. 13.1 as well.

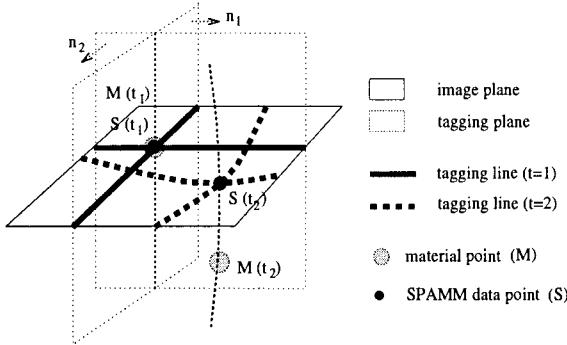


Figure 13.2 Tagging planes and image planes.

Dougherty [9]), an MR tagging method. The advantage of the SPAMM (Spatial Modulation of Magnetization) technique over the other data extraction techniques, is that a number of material points can be marked in a very short time with a simple procedure, and they can be tracked during systole in a non-invasive setting which provides temporal correspondence of the material points. The SPAMM technique has been previously used to estimate regional motion patterns during systole (Axel and Dougherty [9]; Rogers *et al.* [239]; Young *et al.* [301, 302]), and the methods for estimating material deformation have been validated using deformable phantoms (Young *et al.* [300]).

The SPAMM data collection technique is based on the application prior to imaging of a saturation pulse sequence where the amplitude of the magnetization varies spatially, in a sinusoidal-like fashion. This saturation pulse sequence forms the *tagging planes* (see Fig. 13.2). At the minima of this sinusoidal-like variation of the magnetization, dark lines appear in the *image plane* which intersects the tagging planes. If we continue to image the tissue after the saturation pulse sequence is applied, we can see those dark lines move, allowing us to track the motion of the underlying tissue. In order to track points instead of lines, another set of saturation pulse sequences is applied to form a set of tagging planes orthogonal to the previous set of tagging planes. As a result, the grids appear in the image plane. Figs. 13.1(a-e) show short-axis views of an LV from end-diastole (time=1) to end-systole (time=5), where the SPAMM datapoints are defined by the intersections of the respective dark lines.

Given that every image plane is spatially fixed, while a heart being imaged moves, the through-plane motion (Rogers *et al.* [239]; Fischer *et al.* [86]) cannot be captured by the SPAMM datapoints on the image plane. Fig. 13.2 shows the location of a SPAMM datapoint S at two different times t_1 and t_2 . Initially, $S(t_1)$ coincides with a material point $M(t_1)$. However, the motion of the SPAMM datapoint between these two time instances corresponds approximately to the components, on the image plane, of the motion of the material point M which lies somewhere along the line where the tagging planes inter-

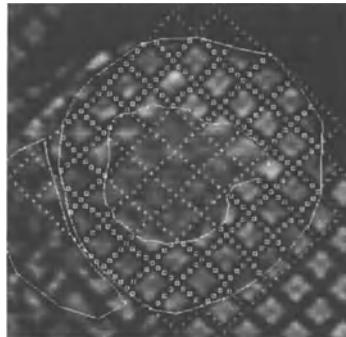


Figure 13.3 SPAMM datapoint extraction.

sect at time t_2 (a possible position of the material point \mathbf{M} at t_2 is marked in Fig. 13.2). Therefore, we can obtain only the in-plane motion from MRI-SPAMM images acquired over time. In order to assess the 3D motion of material points of the LV from the set of such 2D time-varying data, we need to combine through a model-based approach, two sets of data obtained from mutually orthogonal image planes.

As described above, each SPAMM datapoint is defined as the intersection of two tagging planes. Young *et al.* [301] developed a method for extracting the intersection points of tagged lines. The method can be summarized as follows. For each image sequence, the tag stripes within the heart muscle are tracked semi-automatically using a 2D active contour model (Kass *et al.* [137]). In this procedure, each stripe is discretized into equally spaced points and is modeled as a thin, flexible beam with a small inherent resistance to stretching and bending. Stripes are connected at the intersection points and the entire mesh deforms to track the corresponding intensity values in the image. Fig. 13.3 shows an instance of tracking the tagging grid and the inner and outer LV walls where active contours are overlaid onto the image. Bilinear interpolation of image intensity between pixels enables sub-pixel resolution in stripe tracking. Only those points between the inner and outer boundaries were influenced by the image, and the remaining inactive points of the grid were maintained to provide a weak continuity for the structure and allow points to move onto/off the image plane.²

During the SPAMM acquisition process, the spatial location of each image plane is evaluated based on the acquired spatial locations of the corners of each image plane so that we can express the coordinates of each datapoint with respect to the centroid of the LV.

²This semi-automatic tracking algorithm is incorporated into a software package SPAMMVU (Axel *et al.* [11]; Young *et al.* [301]) which is developed at the Radiology Department of the University of Pennsylvania.

It is important to mention that the SPAMM datapoints in the two orthogonal sets of image planes do not correspond to the same material points, but to different material points. These observations will be used in the calculation of the forces exerted from the SPAMM datapoints to the volumetric deformable model.

In addition to the SPAMM datapoints, boundary datapoints from the inner and outer LV walls are also extracted using active contour models and manual initialization. The fully automatic extraction of both the SPAMM and the boundary data is beyond the scope of this work.

13.3 VOLUMETRIC DEFORMABLE MODELS WITH PARAMETER FUNCTIONS

The new class of deformable models we develop allows the use of global parameters that can characterize a volumetric shape in terms of a small number of parameters that are *functions*. The fact that these parameters are functions and not scalars allows the localization and the detailed estimation of the LV shape and motion. Our technique for creating a primitive with parameter functions to model the LV is based initially on the replacement of the constant parameters of a superellipsoid primitive with differentiable parameter functions (e.g., axial deformation). We then add additional global deformations (e.g., axial twisting) with continuous parameter functions to capture the LV deformation better. We assume that those parameter functions are piecewise linear in order not to impose any shape continuity constraints on the LV shape. In other words, the model deforms based on the motion dictated by the SPAMM data and not on the imposition of constraints such as artificial elastic properties (it is impossible with the current non-invasive techniques to measure the viscoelastic properties of the LV for a particular patient in close to real time). Note that it is the same technique used for surface models defined in Park *et al.* [208, 209, 211], except that the space of material coordinates \mathbf{u} is three dimensional, namely $\mathbf{u} = (u, v, w)$, instead of two dimensional $\mathbf{u} = (u, v)$.

13.3.1 Geometric Definition of the Model

To create a volumetric model for the LV with material coordinates $\mathbf{u} = (u, v, w)$, we first define an inertial frame of reference Φ and express the model positions by a vector-valued, time-varying function $\mathbf{x}(\mathbf{u}, t) = (x(\mathbf{u}, t), y(\mathbf{u}, t), z(\mathbf{u}, t))^T$, where T denotes transposition. We also set up a non-inertial, model-centered reference frame ϕ and express the position of a point on the model as

$$\mathbf{x} = \mathbf{c} + \mathbf{R}\mathbf{s}, \quad (13.1)$$

where the center of the model $\mathbf{c}(t)$ is the origin of ϕ and the rotation matrix $\mathbf{R}(t)$ gives the orientation of ϕ relative to Φ with a reference shape \mathbf{s} . Thus, $\mathbf{s}(\mathbf{u}, t)$ gives the positions of points on the model relative to the model frame. To model the shape of the LV through \mathbf{s} , we first define a generalized volumetric primitive \mathbf{e} as follows³:

$$\mathbf{e} = a_0 w \begin{pmatrix} a_1(\mathbf{u}) & \cos u & \cos v \\ a_2(\mathbf{u}) & \cos u & \sin v \\ a_3(\mathbf{u}) & \sin u & \end{pmatrix}, \quad (13.2)$$

where $-\frac{\pi}{2} \leq u \leq \frac{\pi}{4}$, $-\pi \leq v < \pi$, $w > 0$, $a_0 > 0$, and $0 \leq a_1(\mathbf{u}), a_2(\mathbf{u}), a_3(\mathbf{u}) \leq 1$. a_0 is a scale parameter and a_1 , a_2 and a_3 are the aspect ratio parameter functions along the x -, y - and z -axes, respectively. Given the above defined primitive $\mathbf{e} = (e_1, e_2, e_3)^T$, we define the parameterized twisting along the z -axis, and offset deformations which allow the axis to be non-straight in the x and y directions. The resulting global shape \mathbf{s} is then expressed as follows:

$$\mathbf{s} = \begin{pmatrix} e_1 \cos(\tau(\mathbf{u})) - e_2 \sin(\tau(\mathbf{u})) + e_{1_o}(\mathbf{u}) \\ e_1 \sin(\tau(\mathbf{u})) + e_2 \cos(\tau(\mathbf{u})) + e_{2_o}(\mathbf{u}) \\ e_3 \end{pmatrix}, \quad (13.3)$$

where $\tau(\mathbf{u})$ is the twisting parameter function along the z -axis, and $e_{1_o}(\mathbf{u})$ and $e_{2_o}(\mathbf{u})$ are axis-offset parameter functions in the x and y directions, respectively. Therefore the deformation parameter vector \mathbf{q}_s is defined as

$$\mathbf{q}_s = (a_1(\mathbf{u}), a_2(\mathbf{u}), a_3(\mathbf{u}), \tau(\mathbf{u}), e_{1_o}(\mathbf{u}), e_{2_o}(\mathbf{u}))^T. \quad (13.4)$$

The parameters, \mathbf{q} , of the model that we want to recover are

$$\mathbf{q} = (\mathbf{q}_c^T, \mathbf{q}_\theta^T, \mathbf{q}_s^T)^T, \quad (13.5)$$

where $\mathbf{q}_c = \mathbf{c}$, is the global translation, and \mathbf{q}_θ , is the quaternion that corresponds to the global rotation matrix \mathbf{R} .

13.3.2 Model Dynamics

By incorporating the geometric definition of the models into our physics-based framework, we create dynamic models that deform due to forces exerted from datapoints and conform to the given dataset [181]. In our dynamic formulation, we create a model that has no inertia and comes to rest as soon as all the applied forces equilibrate or vanish. Given that the localization and tracking of SPAMM datapoints is relatively accurate, and to avoid undesired smoothness in the solution caused by the incorporation of incorrect elasticity in the model, we assume a zero stiffness matrix⁴ so that there is no resistance to deformation.

³it is a generalization of an ellipsoid primitive

⁴While we use the above defined deformations, finite elements are often used as an alternative, to model deformations. Possible errors from using finite elements to model the heart

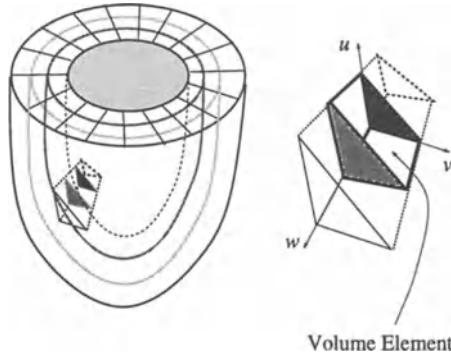


Figure 13.4 Discretization of the LV model based on Volume elements.

The resulting equation of motion is (3.51), and \mathbf{q} is the vector of the model's degrees of freedom. For fast interactive response, a first-order Euler method (Press *et al.* [226]) is employed to integrate equation (3.51).

In computing the correct forces from the data, the algorithm exploits the geometry of the motion of the SPAMM datapoints over time. Once these forces are computed, we use (3.51) to estimate the model parameters.

13.4 MODEL FORCE COMPUTATION

The model is discretized based on prismatic volume elements as shown in Fig. 13.4. The resolution of discretization of the volume of the myocardium depends on how dense the SPAMM datapoints are distributed throughout the volume. In our experiments, we have rather sparse SPAMM datapoints across the myocardium and we therefore tessellated the model so that each volume element has its triangular faces at the inner and outer walls of the LV (see Fig. 13.5).

We use two types of datapoints to estimate the LV shape and motion, as depicted in Fig. 13.5. From these data, we compute the corresponding forces on the model depending on their type. Boundary data provide forces for the estimation of the LV shape, while SPAMM datapoints provide forces for the estimation of the volumetric motion of the LV. The following subsections describe the algorithms for extracting and distributing forces from the given dataset.

wall elasticity arise due to the incorrect computation of the material stiffness matrix when the elasticity of the myocardium which varies spatially is not known. Simplifying assumptions for the elasticity of the myocardium have been attempted. For examples, in the work of Janz and Waldron [127], the myocardium was approximated with a homogeneous, isotropic and incompressible material, and in the work of Ghista and Hamid [99], the isoparametric elements are utilized.

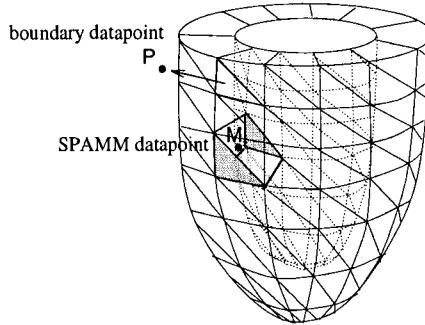


Figure 13.5 Types of datapoints acting on the model.

13.4.1 Force Computation from Boundary Data

Boundary datapoints simply constrain the shape of the inner and outer walls of the LV and provide no correspondence of points over time. Therefore, the forces from each boundary datapoint \mathbf{P} to the corresponding model wall (inner or outer) are computed as follows. We approximate each boundary triangular element with a plane. We then determine the element whose distance from \mathbf{P} is minimum, and the associated intersection point \mathbf{Q} on the element (see Fig. 13.6(a)). The force that \mathbf{P} exerts on the model is

$$\mathbf{f}_P = \gamma_1(\mathbf{P} - \mathbf{Q}), \quad (13.6)$$

where γ_1 is the strength of the force. Then \mathbf{f}_P is distributed to the nodes $\mathbf{x}_1, \mathbf{x}_2$ and \mathbf{x}_3 of the associated boundary triangular element based on the formula

$$\mathbf{f}_{\mathbf{x}_i} = m_i \mathbf{f}_P, \quad i = 1, 2, 3, \quad (13.7)$$

where the m_i 's are computed from the solution of the following linear system

$$\sum_i m_i \mathbf{x}_i = \mathbf{Q}, \quad (13.8)$$

subject to the following constraint

$$\sum_i m_i = 1.0. \quad (13.9)$$

This constraint is necessary so that the following equation is also true

$$\sum_i \mathbf{f}_{\mathbf{x}_i} = \sum_i m_i \mathbf{f}_P = \mathbf{f}_P. \quad (13.10)$$

Intuitively, each of the m_i 's is a weight given to each element node and the vector \mathbf{Q} is the location of the center of mass of the element's nodes.

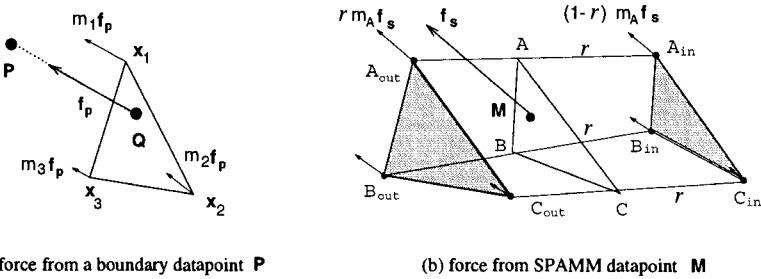
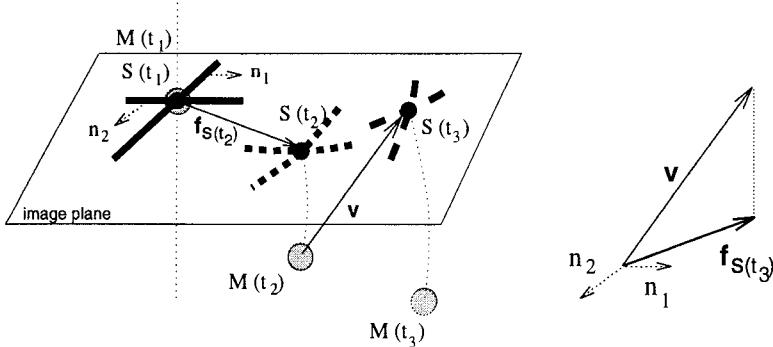


Figure 13.6 Distributing forces on the model from datapoints.

13.4.2 Force Computation from SPAMM Data



Forces $\mathbf{f}_s(t_2)$, $\mathbf{f}_s(t_3)$ from SPAMM points $\mathbf{S}(t_2)$ and $\mathbf{S}(t_3)$, respectively.

Figure 13.7 Force calculation from SPAMM datapoints.

As opposed to the boundary data, SPAMM data provide correspondence over time of the associated SPAMM datapoints. Due to the way that SPAMM datapoints are defined (see section 13.2.2), their correspondence over time leads to the computation of forces on the model that are parallel to their corresponding image plane. These forces are computed as follows.

Initially, we assume that the SPAMM datapoints and the model's material points coincide. Let $\mathbf{M}(t_1)$ be the material point which coincides initially with a SPAMM datapoint $\mathbf{S}(t_1)$ at time t_1 (see Fig. 13.7). Let also $\mathbf{S}(t_2)$ and $\mathbf{S}(t_3)$ be the corresponding SPAMM datapoints to the point $\mathbf{S}(t_1)$ at the next two time frames. Then the force on $\mathbf{M}(t_1)$ from $\mathbf{S}(t_2)$ is computed as

$$\mathbf{f}_{s(t_2)} = \gamma_2 [([\mathbf{S}(t_2) - \mathbf{M}(t_1)] \cdot \mathbf{n}_1) \mathbf{n}_1 + ([\mathbf{S}(t_2) - \mathbf{M}(t_1)] \cdot \mathbf{n}_2) \mathbf{n}_2], \quad (13.11)$$

where γ_2 is the strength of the force and $\mathbf{n}_1, \mathbf{n}_2$, are the unit normals of the corresponding initial (i.e., at time t_1) tagging planes as shown in Fig. 13.2.

$\mathbf{f}_S(t_2)$ will cause $\mathbf{M}(t_1)$ to move to a new position $\mathbf{M}(t_2)$. Subsequently, the force $\mathbf{f}_S(t_3)$ on $\mathbf{M}(t_2)$ from $\mathbf{S}(t_3)$ will be computed in a similar fashion and it is shown in Fig. 13.7. The forces $\mathbf{f}_S(t)$ are always parallel to the corresponding image plane and orthogonal to the initial tagging plane of the SPAMM datapoint. These forces are spring-like forces and are not computed as a result of imposing hard-constraints on the projected motion of a material point. This decision is made because when the through-plane motion of a material point is large and the straight-line projection assumption of the material point's location on the image plane is not valid. Therefore, its projection should not coincide exactly with the location of the corresponding SPAMM datapoint. In such cases, based on these spring-like forces, a correct non-straight line correspondence is computed as shown in Fig. 13.16.

Once we compute these forces, we distribute them to the nodes of the deformable model. These nodal forces will cause the model to deform. We distribute at any time frame t_i , the computed force \mathbf{f}_S to the nodes of the prism $A_{out}B_{out}C_{out}A_{in}B_{in}C_{in}$ within which \mathbf{M} lies, as follows.

Based on the finite element theory, we compute the nodal positions, $\mathbf{A}, \mathbf{B}, \mathbf{C}$, of the triangle ABC in which \mathbf{M} lies such that

$$r = \frac{\mathbf{A} - \mathbf{A}_{in}}{\mathbf{A}_{out} - \mathbf{A}_{in}} = \frac{\mathbf{B} - \mathbf{B}_{in}}{\mathbf{B}_{out} - \mathbf{B}_{in}} = \frac{\mathbf{C} - \mathbf{C}_{in}}{\mathbf{C}_{out} - \mathbf{C}_{in}}, \quad (13.12)$$

where r is a scalar. To compute r , we solve the following cubic scalar equation using the Newton-Raphson method

$$(\mathbf{M} - \mathbf{A}) \cdot ((\mathbf{C} - \mathbf{A}) \times (\mathbf{B} - \mathbf{A})) = 0, \quad (13.13)$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are computed with respect to r from equation (13.12).

The force \mathbf{f}_S is then extrapolated to the nodes of triangle ABC based on the same algorithm we used for the force computation from boundary data. The scalars m_A, m_B, m_C , correspond to the triangle's nodes A, B, C , respectively. Then the forces on the nodes of the prism which lie on the inner and outer walls of the LV are computed as follows (see Fig. 13.6(b))

$$\begin{aligned} \mathbf{f}_{N_{out}} &= r m_N \mathbf{f}_S \\ \mathbf{f}_{N_{in}} &= (1 - r) m_N \mathbf{f}_S, \end{aligned} \quad (13.14)$$

where $N = \{A, B, C\}$.

The computation of r which determines the model's material point which corresponds to a SPAMM datapoint is only done once at the beginning of the LV motion estimation. It is the correspondence of SPAMM datapoints over time that allows us to estimate the twisting motion of the LV. In addition, by combining forces from two orthogonal sets of SPAMM datapoints we estimate the full 3D shape and motion of the LV.

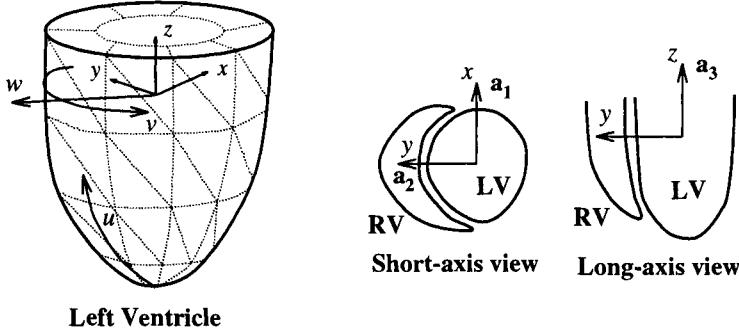


Figure 13.8 Definition of model's coordinate system.

13.5 MODEL PARAMETERS

The parameters of the LV model that are estimated during the fitting process reflect the shape changes during systole. The deformable model has six parameter functions q_s , to model deformations, as defined in (13.4), which can be interpreted intuitively without complex post-processing. In addition, the model has global translation q_t and global rotation q_θ parameters (see (13.5)). The initial orientation of the model is important in order to understand the role of each parameter. As depicted in Fig. 13.8, the short and long axis views initially coincide with the xy - and yz - planes in the model frame of reference. The center of the model is chosen at the centroid of the LV with the y -axis pointing towards the right ventricle (RV). The material coordinates are $\mathbf{u} = (u, v, w)$, where u runs from the apex to the base of the LV, v starts and ends at the point where the septum is located, and w is used for the definition of model points between the inner and outer walls of the deformable model. The parameter functions we use are functions of \mathbf{u} , so that we can model accurately the local variation throughout the LV. As we demonstrate in the following sections, the variation of those parameters with respect to u , v , and w is chosen so that they are independent.

We now present in detail the interpretation and use of each of the model's parameters. Table 13.1 summarizes what each parameter function captures during the 3D shape and wall motion estimation of the LV.

Parameters	Representation
a_1, a_2	Radial contractions
a_3	Longitudinal contraction
τ	Twisting about the long axis
e_{1o}, e_{2o}	Long axis deformation

Table 13.1 Model parameters.

13.5.1 Radial Contraction

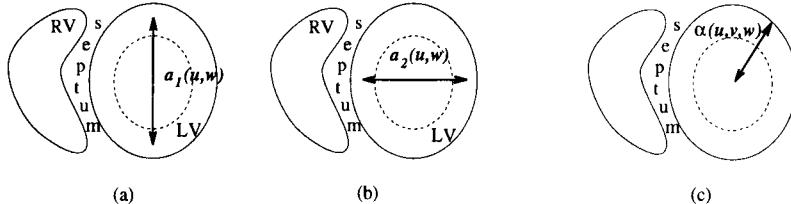


Figure 13.9 Parameters for radial contraction: $a_1(u, w)$, $a_2(u, w)$.

Since the short axis views coincide initially with the xy -plane⁵, the parameter functions a_1 and a_2 (which are the aspect ratios along the x - and y -axes of the LV model, respectively) will capture the radial contraction motion. Since the y -axis points towards the septum of the heart, the a_2 parameter captures the motion of the constrained wall, while the a_1 parameter captures the motion of the free wall as shown in Figs. 13.9(a-b). For each time frame t , we estimate the values of $a_1(u, w)$ and $a_2(u, w)$, and compute the percentage of the change with respect to their values at end-diastole (ED) which corresponds to the initial time frame (or t_{ED}). For example, the graphs shown in Fig. 13.17(a) are the plot of a_1 computed as follows:

$$\frac{a_1(t) - a_1(t_{ED})}{a_1(t_{ED})} \times 100.$$

Although we do not show here, we can combine parameters $a_1(u, w)$ and $a_2(u, w)$, as one parameter $\alpha(u, v, w)$ to model the radial contraction of the LV (see Fig. 13.9(c)). In this case, the geometric definition of the model given in (13.2) and (13.3) will become

$$\mathbf{s} = \begin{pmatrix} e_1 \cos(\tau(\mathbf{u})) - e_2 \sin(\tau(\mathbf{u})) \\ e_1 \sin(\tau(\mathbf{u})) + e_2 \cos(\tau(\mathbf{u})) \\ e_3 \end{pmatrix}, \quad (13.15)$$

where

$$\begin{aligned} e_1 &= a_0 w \alpha(\mathbf{u}) \cos u \cos v, \\ e_2 &= a_0 w \alpha(\mathbf{u}) \cos u \sin v, \\ \text{and } e_3 &= a_0 w a_3(\mathbf{u}) \sin u. \end{aligned} \quad (13.16)$$

Note that in this case the axis deformation parameters ($e_{1,0}$ and $e_{2,0}$) are not needed, and parameter independence is maintained.

13.5.2 Longitudinal Contraction and Global Translation

Since the long axis views coincide with the yz -plane, the parameter function a_3 (which is the aspect ratio along z -axis) will capture the longitudinal contraction

⁵The xy -plane does deviate over time from the short axis view due to the global rotation of the LV, but not significantly.

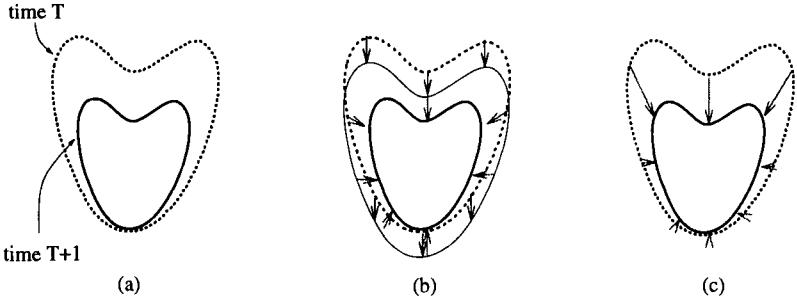


Figure 13.10 Parameters for longitudinal contraction and global translation.

motion. However, since we do not have enough time frames over the heart cycle, the estimation of the global translation in the z direction of the model frame can be arbitrary (the two parameters can't be independently estimated) and may result in the false estimation of especially the longitudinal deformation parameters (the global translation in the x and the y directions of the model frame is negligible). Consider the following case shown in Fig. 13.10(a). It is a typical motion of the LV observed at two subsequent time instances (T and $T + 1$), which shows that the motion at the apex of the LV is relatively small. Suppose that the length of the LV is 10cm and 8cm, at times T and $T + 1$, respectively. If the origin of the model frame at time T is at the half way along the LV, the origin would globally translate 1cm at time $T + 1$. Yet this amount of translation is arbitrary because it depends on where the origin is. Moreover, in this case we would capture a relatively uniform longitudinal contraction from apex to base as shown in Fig. 13.10(b). This has the result that, even though the combination of translation and longitudinal contraction would give the correct 3D motion of the LV, the longitudinal contraction would lose its intuitive meaning. However, as mentioned above this is an artifact of the sparse sampling over time.

Due to the above sparse sampling, and the fact that we could not observe a significant overall translation of the model in all our datasets, we kept the global translation constant during the fitting process for the subsequent time frames. We therefore capture the LV longitudinal motion as a deformation based on a_3 (see Fig. 13.10(c)). In our experiments, we compute the actual displacement based on the a_3 parameter as follows:

$$a_{3_d} = a_3(t) \sin(u) - a_3(t_{ED}) \sin(u).$$

We plot the percentage of the changes with respect to the initial length of the LV in order to compare the amount among different LVs. For example, the graphs shown in Fig. 13.17(c) are the plot of a_3 computed as follows:

$$\frac{a_{3_d}}{\text{length of the LV}} \times 100.$$

In case there is a pathological case that the LV translates (globally) significantly along the z axis in addition to contracting, then we can very easily estimate this global translation by simply subtracting from a_{3d} the common least amount of deformation from apex to base. However, it is not clear that knowledge of the global translation is clinically useful.

13.5.3 Twisting and Global Rotation

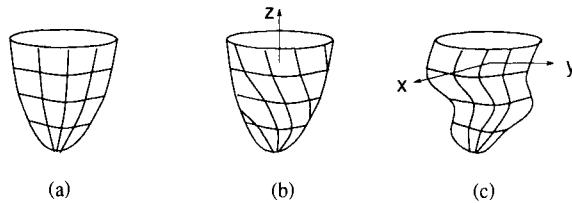


Figure 13.11 Twisting and long axis deformations.

The twisting parameter function $\tau(u, v, w)$ captures the twisting about the long axis of the LV as shown in Fig. 13.11(b). For each location w , and for each location u , we allow the parameter to vary along v in order to obtain the variation of twisting as a function of circumferential position as well. The average of the twisting values at all the circumferential positions for each location u is plotted in Fig. 13.17 and Fig. 13.18. In this way we quantify the twisting motion from the apex to the base of each LV.

At each subsequent time frame, we first estimate the global rotation of the model before estimating the deformation parameters. In this way, we can subtract the global rotation from the twisting deformation, and therefore estimate the overall tilting of the LV.

The global rotation of our model is expressed as a quaternion. This results in numerically more robust estimation of global rotation during model fitting to the data [181]. In addition, from the quaternion we can compute very efficiently the corresponding rotation matrix \mathbf{R} that we use to compute the nodal positions \mathbf{x} based on (13.1).

13.5.4 Long Axis Deformation

By having the axis-offset parameters $e_{1o}(u, w)$ and $e_{2o}(u, w)$, we allow the centroid of each cross section at a different location along the long-axis of the LV to displace globally in the xy -plane (see Fig. 13.11(c)). In this way we can capture the shape of the LV accurately, and at the same time observe the bending of the LV, if any, without using any special function for the bending.

13.5.5 Ejection-fraction and Other Parameters

It is also very important to mention that our technique for constructing these deformable models is general and we can therefore add other global deformations such as bending and shearing. However, for the experiments we present the above described parameters were adequate.

If needed, it is also possible to do strain analysis as follows. Based on the extracted model parameters, we first compute the positions, \mathbf{x}_i , of the element nodes from (13.1) and then we apply the standard strain computation procedure as in Young *et al.* [301, 302].

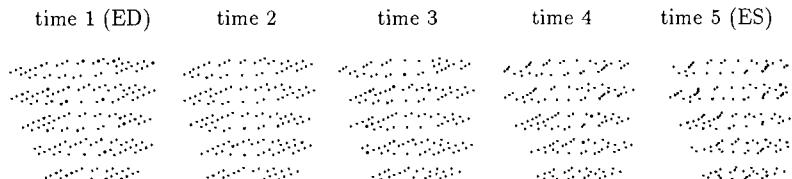
Finally, by adapting Gauss' theorem (O'Neil [206]) we can accurately calculate over time the volume of the blood pool, and therefore compute ejection-fraction.

13.6 IMPLEMENTATION OF MODEL FITTING PROCEDURE

We have applied our approach to MRI-SPAMM data of LVs obtained at the Department of Radiology of the University of Pennsylvania. As discussed in Section 13.2.2, in order to estimate the 3D left ventricular wall motion, we used two sets of SPAMM datapoints obtained from mutually orthogonal image planes (short-axis and long-axis views). We used data from ten image planes, five from short-axis view planes, and another five from long-axis view planes. These image planes span the spatial extent of the LV. Furthermore, for each image plane, we have datasets over five time sequences during systole (from end-diastole to end-systole). Therefore, in total, $10 \times 5 = 50$ datasets of 2D images, containing time-varying datapoints of the LV, were utilized. Fig. 13.12 shows a set of SPAMM datapoints extracted from the MR images of a normal LV. Since some of the SPAMM datapoints on the image plane disappear and/or reappear at subsequent times, we used at every time frame only those points which have a corresponding point at the previous time frame. Therefore, the number of *active* points decreases towards end-systole.

From each image, we also extracted the boundary datapoints representing inner and outer walls. Through the following experiment we give the details of the model fitting procedure to the boundary and the SPAMM datapoints.

Starting from a generalized ellipsoid, we first estimate the shape of the LV at end-diastole from the boundary dataset. Then we fit the LV model to the data from subsequent time frames gradually up to end-systole and we estimate its motion over time. Note that the fitting process for each time frame takes approximately 45 seconds on a Silicon Graphics R4400 Indigo workstation.



(a) SPAMM datapoints from short-axis view image planes.



(b) SPAMM datapoints from long-axis view image planes.

Figure 13.12 SPAMM datapoint sets.

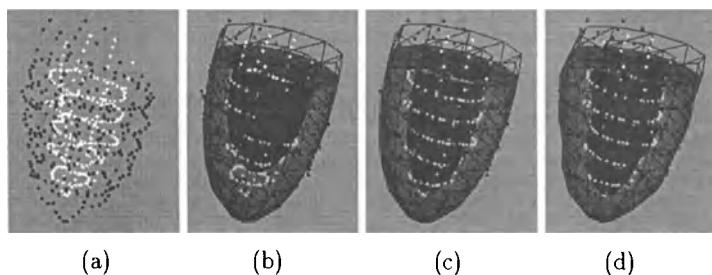


Figure 13.13 Initial model shape recovery from boundary data.

13.6.1 Model Fitting to Boundary Datapoints at End-Diastole

Based on the boundary datapoints from the inner and outer walls, we first recover the shape of the LV at time 1 (i.e., end-diastole). This is done by first overlaying a simple volumetric model, which resembles a volumetric ellipsoid, onto the data (Fig. 13.13(a) shows all boundary datapoints at time 1, where black dots and white dots are on the outer and inner walls, respectively.). Initially, the model frame is placed at the center of mass of the boundary datapoints. The forces acting on the model will cause it to translate and to rotate to find a suitable position and orientation.

Then, based on the computation of boundary forces, the nodes on the inner and outer walls of the model are pulled towards the inner and outer boundary datapoints, respectively. As a result of these forces, the model parameter functions change so that the model conforms to the dataset. When all applied forces

equilibrate or vanish, or the error of fit (the distance between a datapoint and the model surface) falls within an acceptable tolerance specified by the user, the model comes to rest. For efficient and effective model fitting, we come up with the following fitting schedule:

1. The initial model has constant parameter functions; in particular, $a_1(\mathbf{u}) = c_1$, $a_2(\mathbf{u}) = c_2$, $a_3(\mathbf{u}) = c_3$ and $\tau(\mathbf{u}) = e_{1_o}(\mathbf{u}) = e_{2_o}(\mathbf{u}) = 0$, where $0 < c_1, c_2, c_3 < 1$, for all $\mathbf{u} = (u, v, w)$. The model rotates and translates to find its optimal location for the model center in the reference frame, while estimating c_1 , c_2 and c_3 . We also estimate the value of the material coordinate w for the inner and outer walls at this stage (Fig. 13.13(b) : $w_{in} = 0.896$ and $w_{out} = 1.368$).
2. Once the fitting of the model in the first step is completed and we have estimated w for the inner and outer walls, the a_1 , a_2 and a_3 parameters are allowed to vary with respect to w only, so that we can recover the wall thickness more accurately (Fig. 13.13(c) : $a_1(w_{out}) = 0.360$, $a_1(w_{in}) = 0.282$, $a_2(w_{out}) = 0.341$, $a_2(w_{in}) = 0.276$, $a_3(w_{out}) = 0.807$, and $a_3(w_{in}) = 0.924$).
3. Finally, the parameters are allowed to vary also in u to estimate the non-symmetrical shape of the LV. We first estimate the parameter functions $a_1(u, w)$, $a_2(u, w)$ and $a_3(u, w)$ and then $e_{1_o}(u, w)$ and $e_{2_o}(u, w)$ (Fig. 13.13(d)). Note that the twist parameter τ is activated when the datapoints from next time frame (time 2) are loaded.

13.6.2 Registration of SPAMM Datapoints to the Model

After the model fits the initial boundary data at end-diastole (ED), we use it to fit data from subsequent time frames till end-systole (ES). We first read in the SPAMM datapoints at ED in order to register the locations of SPAMM datapoints which coincide with material points. For each SPAMM datapoint, we find the volume element which encloses the datapoint, and compute r and m_N as described in Section 13.4.2. The values of r and m_N give the relative location of each material point marked on the model with respect to the enclosed volume element. Therefore we can always locate the material point regardless of how the volume element is deformed.

13.6.3 Model Fitting to Datapoints Over Time

The SPAMM datapoints at time 2 are loaded onto the previously fitted model (time 1). The model deforms due to the SPAMM data forces, as described

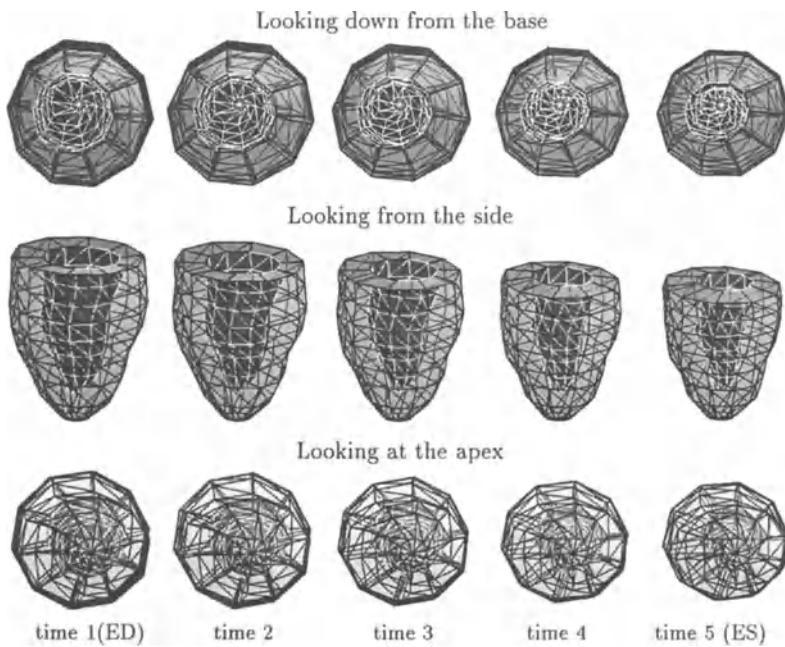


Figure 13.14 Fitted models during systole.

in Section 13.4.2. The combination of forces from the SPAMM datapoints in two orthogonal sets allow us to recover the deformation of the model in all three components, effectively recovering the missing through-plane motion (see Fig. 13.15). Once the model comes to rest, the SPAMM datapoints at time 3 are loaded onto the fitted model of time 2. And we repeat the process for up to end-systole. The deformation from ED to ES is captured in the parameter functions. By plotting the parameter functions over time, we can characterize and quantify the motion of the LV.

Note that we also utilize boundary datapoints during the subsequent time fittings. We first estimate rigid motion using both boundary and SPAMM datapoints, and then the deformation using only SPAMM datapoints.

13.7 EXPERIMENTAL RESULTS

In the following experiments we present our LV shape and motion analysis results for normal and abnormal LVs with hypertrophic cardiomyopathy (**htcm**).

Fig. 13.14 shows model fitting results for a normal LV over 5 time frames from end-diastole (ED) to end-systole (ES). The top row shows a view from the base

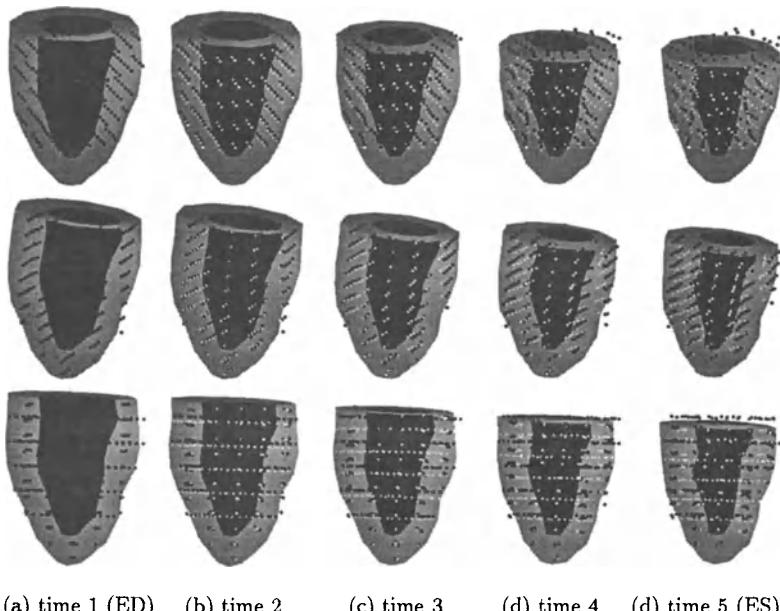


Figure 13.15 Fitted models during systole with SPAMM datapoints and material points.

of the LV of the fitted model. The twisting of the inner wall (shown in white) is obvious. The middle row shows a side view of the model, while the last row is similar to the first row and shows a view of the model from the apex. We can easily observe the longitudinal contraction as well as the radial contraction. The combination of computed forces from the SPAMM datapoints from two orthogonal planes allows us to recover the deformation of the model in 3D.

Fig. 13.15 shows the fitted model superimposed on the SPAMM datapoints over 5 time frames. SPAMM datapoints are denoted with black dots, while the corresponding model material points are denoted with white dots. Initially, the SPAMM datapoints coincide with the material points on the model (therefore, only black dots are shown in the Fig. 13.15(a)). The first two rows show the model with superimposed short-axis SPAMM datapoints (Fig. 13.12(a)), and long-axis SPAMM datapoints (Fig. 13.12(b)), respectively. The last row shows both sets of SPAMM datapoints. We can observe that the material points move in 3D space, thus recovering the through-plane motion, while the image planes (where black dots are located in Fig. 13.15) are stationary.

Fig. 13.16 shows a top portion of the model at the end-systole and the relationship between SPAMM datapoints and material points. The various arrows indicate: 1) SPAMM datapoints⁶ at end-diastole, which coincide with the cor-

⁶It is a subset of SPAMM datapoints from a short-axis view image plane.

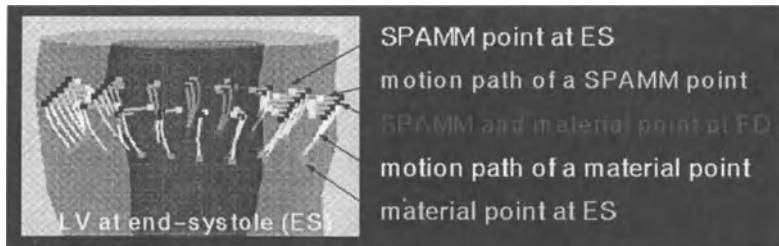


Figure 13.16 Material and SPAMM datapoint paths during systole.

responding material points, 2) the SPAMM datapoints at the end-systole and their motion paths over time, 3) the corresponding material points at the end-systole, and 4) the 3D motion paths of the material points during systole. Notice that at every instant the projection of the material point location on the image plane approximately coincides with the location of the SPAMM datapoint. Notice that in agreement with the SPAMM datapoint generation procedure (see previous sections), the projection is not along a straight line, but along a curved line.

13.7.1 Parameter Graphs

Fig. 13.17 shows graphs of the extracted model parameters as functions of u (i.e., varying along the long axis of the LV) at the inner and outer walls. In general, the contraction and the twisting deformation are more significant on the inner LV wall (Figs. 13.17(g-j)) compared to the outer wall (Figs. 13.17(a-d)). The difference in the corresponding parameter values is obvious. The graphs allow us to quantify the motion and shape changes of the LV during systole. For example, by studying the graphs in Figs. 13.17(a,b), we can conclude that the overall percent changes of the magnitude of radial contraction of the outer wall during systole is approximately 15 – 20%. However, towards the base of the LV the contraction along the y axis (it is approximately 10% from Fig. 13.17(b)) is less than the contraction along the x axis (it is approximately 17% from Fig. 13.17(a)) making the base look more elliptical. From Fig. 13.17(d) we can quantify the twisting motion of the outer wall during systole to approximately 14 degrees total from base to apex. The graph shows a small amount of global rotation before the twisting occurs.⁷ It is not easy to see this kind of subtle motion when one watches the model contract and twist on the monitor. Figs. 13.17(e, f, k, l) show the estimation results for the long axis deformation, which represent small changes and are not significant. By having the graphs of the parameter functions plotted in conjunction with the animation, we can quantify and easily characterize the detailed motion of the deforming model over time.

⁷ Compare the changes in the parameter values in Figs. 13.17(d,j) from time 1 to time 2 with the changes from time 2 to time 3.

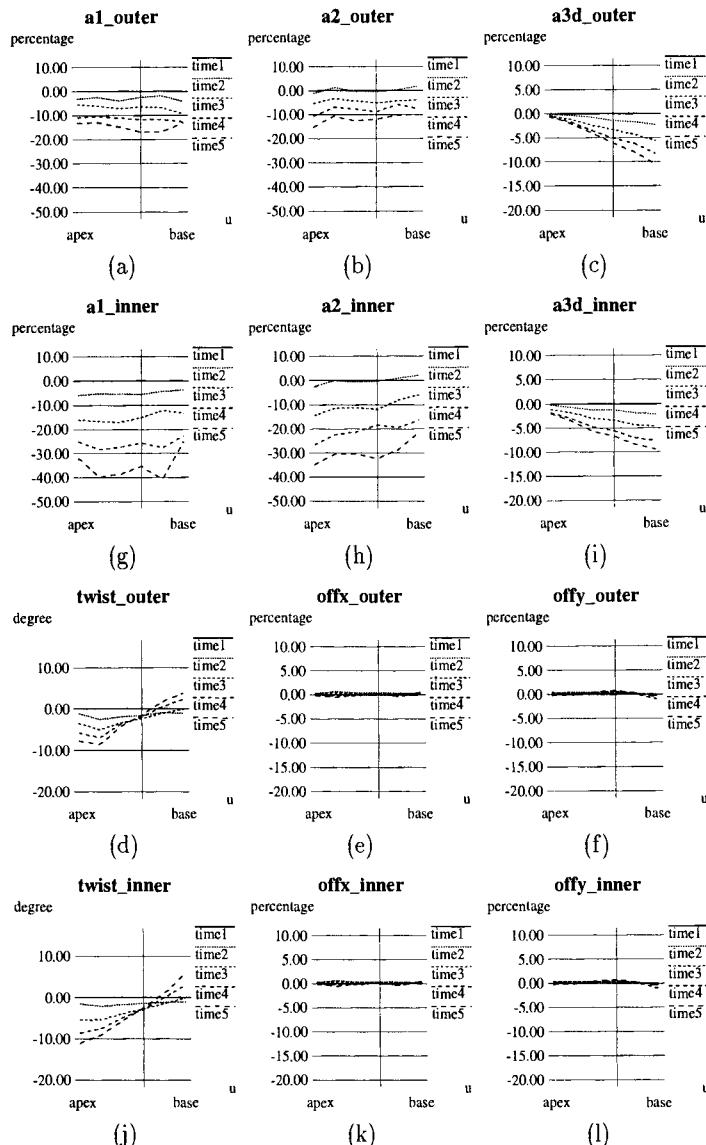


Figure 13.17 Extracted model parameters as functions of u (Normal LV).

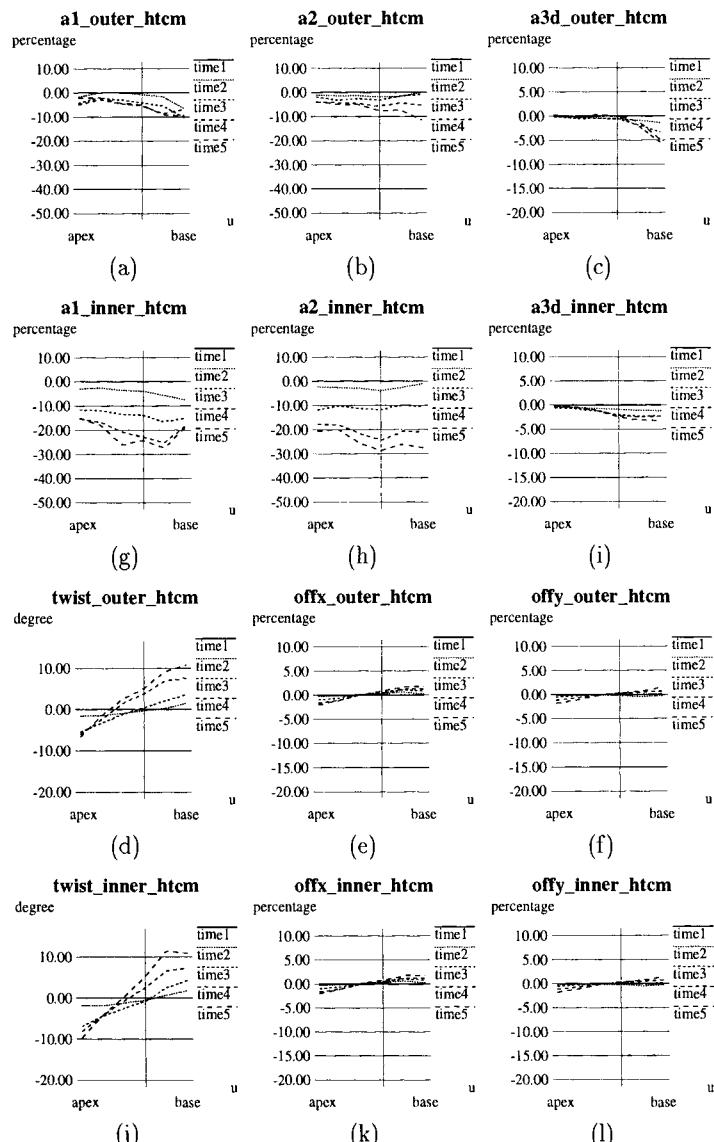


Figure 13.18 Extracted model parameters as functions of u (Abnormal LV).

In Fig. 13.18 we show fitting results to SPAMM data from an abnormal LV with hypertrophic cardiomyopathy (htcm). By comparing the corresponding graphs we can observe that the abnormal heart does not contract as much as the normal heart (especially towards the apex), but twists more, and has a bigger long axis deformation. This latter deformation corresponds to a spatial bending of the LV.

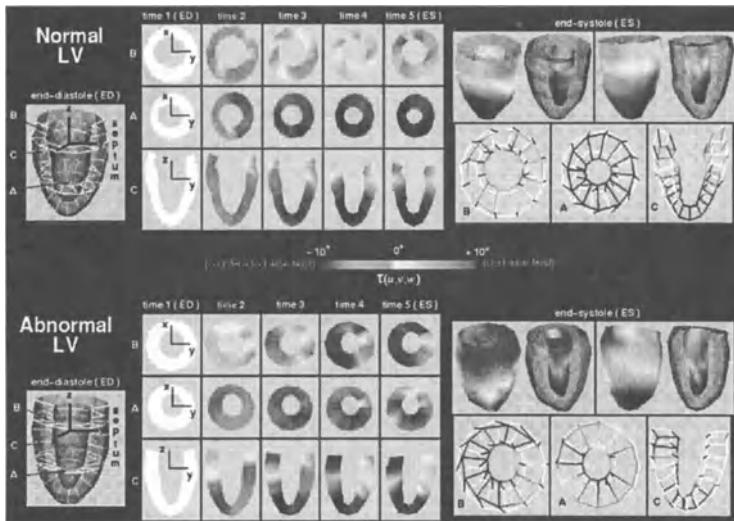


Figure 13.19 Quantitative visualization of the twisting parameter function for normal and abnormal LVs.

In Fig. 13.19 we illustrate the twisting motion of the two LVs at two short axis views (A and B) and one long axis view (C). In our coloring method, white corresponds to zero twist, while positive (clockwise) twist, and negative (counter-clockwise) twist are depicted with a different amount of gray. During systole, for each LV (normal and abnormal) the first two rows on the left illustrate the variation in twisting on the two short axis views, while the third row on the left shows the variation in twisting on the long axis view (notice that the y axis always points towards the septum). Finally, for each of the two LVs, the rows on the right depict at end-systole the twisting motion of the LV on the inner and outer walls (top row), and on the A, B and C cross-sections (bottom row) where white meshes depict the cross-section at end-diastole and black lines depict point correspondences. The maximum twists are depicted in dark gray (10 and -10 degrees). From this coloring we can clearly visualize the nonuniform twisting motion throughout each of the LVs. It also becomes clear that the abnormal LV twists more and its twisting motion is more non-uniform. We can also observe the changes in the myocardium thickness during systole. Additional color-based analysis of the LV shape and motion can be found at the author's World Wide Web page: <http://www.cis.upenn.edu/~dnm>.

Based on Gauss' theorem (O'Neil [206]), we also calculated the volume of the blood pool over time to compute ejection-fraction. The ejection-fraction of the normal LV was 61%, while for the abnormal it was 52%.

All the above presented results verify quantitatively qualitative knowledge about the LV known to physicians. We are currently in the process of conducting a series of experiments with normal and abnormal hearts with various types of disease in order to investigate the relationship between the extracted parameter functions and heart disease.

VISUALIZING RESPIRATORY MECHANICS BASED ON ANATOMICAL AND PHYSIOLOGICAL MODELING

We present a new direction in medical applications which stems from our recent efforts to couple PBM methods and physiology modeling to visualize respiratory mechanics. This effort is hoped to result in future sophisticated virtual environments that can be used in both clinical applications and in medical training.

Understanding relationships between anatomy and physiology is critical to the education of medical professionals. The computer affords a unique opportunity to investigate these relationships by allowing students to interact with simulated patients. Recently, computer graphics techniques and hardware have developed sufficiently to enable opportunities for creating realistic virtual environments. The enormous success of flight simulators to train pilots attests to the potential value of virtual environments. Biomedical researchers ([78, 246, 248]) feel that similar success can be attained for biomedical applications, such as for surgical training.

In addition to its potential value in surgical training, a virtual patient environment can serve to demonstrate physiological and pathophysiological concepts. The operator (student) can change physiological parameters to simulate different conditions, and see the results of that change (both direct and indirect). Some physiological simulations (e.g., [56, 75, 248]) output their results in formats similar to the output of equipment found in actual environments, or with simplified anatomical representations. We feel that some areas, such as respiratory mechanics, lend themselves better to explanation via demonstration of their effects on anatomy (i.e., a 3D presentation), and hence we focus on integrating realistic physiological simulation with 3D anatomical models.

This chapter describes the methods we are applying to model respiratory mechanics for quiet, normal breathing and a few types of pneumothoraces.¹ This involves integrating a 3D deformable object simulation with physiological process equations. We present results showing pressure/volume relationships dur-

¹A *pneumothorax* is the pathological presence of air in the intrapleural space, the space between the visceral and parietal pleura. Pneumothoraces typically result from trauma to the chest.

ing simulation of normal breathing and open sucking chest wounds. The values of model variables are consistent with qualitative expectations and are sufficiently precise for clinical confirmation.

The aim of this research is to link physiological models that express mechanical behavior (and ultimately chemical and neurological functions) with a visually-and structurally-accurate anatomical model. Satava [246] envisions the impact of Virtual Reality on medical education as helping students to understand basic anatomy or important physiological principles. We see the impact of our effort as the means for illustrating and predicting *relationships* between anatomy and physiology in an interactive environment.

14.1 RELATED WORK

Computer-based medical modeling has a long history in the bioengineering community [236]. Emerson et al [80] present an extensive bibliography on the subject of Virtual Reality in medical education. While there is a growing body of multimedia medical instruction programs, most of them are based on simple physiological models. More comprehensive programs for physiological models (e.g., [56, 75]) display their (scalar) results in graphs or with simple graphical output [248]. However, none of the above attempts model graphically the anatomical changes.

An exciting prospect for medical education involves potential uses for the data collected in the Visible Human Project [1]. The Visible Human Project is an endeavor to create a complete, anatomically detailed, three-dimensional representation of the male and female body, from CAT, MR, and cryosection images. Part of the long-term goals for the Project is to perform basic research into representation of structures, particularly encoding the connection between structural-anatomical to functional-physiological knowledge.

14.2 BASIC ANATOMY AND PHYSIOLOGY

The lung and the chest wall (a unit comprised of the ribcage, diaphragm-abdominal combination, and the respiratory muscles) function as viscoelastic structures.

In normal anatomy, the lung and chest wall are separated by a thin layer of fluid called the intrapleural fluid that transmits forces between the two. The fluid acts to resist their separation in a similar way that two moist pieces of glass resist separation. Also, the fluid allows the lung to slide along the chest wall. The pressure in this space, the pleural (or intrapleural) pressure, is subatmo-

spheric. The slightly negative pressure keeps the lung inflated and in contact with the chest wall. At the end of quiet exhalation (Functional Residual Capacity [FRC]), the elastic tendency for the lung to recoil from the chest wall is exactly balanced by the chest wall's elastic tendency to recoil from the lung [93].

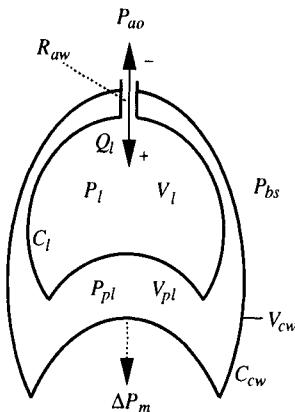


Figure 14.1 Physiological variables.

Fig. 14.1 shows some of the physiological variables and parameters involved. Volumes are denoted as V , pressures as P , and flows as Q . Resistance is denoted R , and compliance C . Subscripts indicate to which entity the variable pertains, where l is for the lung, cw is for the chest wall, pl is for the intrapleural space, ao is the airway opening, and aw is the airway. The symbol ΔP_m represents the total muscle effort cast as an effective pressure difference.

Inhalation is an active process, where quiet exhalation is passive. During inhalation, the muscles of respiration (principally the diaphragm) contract. This increases the volume of the thoracic cavity (V_{cw}) and decreases the pleural pressure (P_{pl}). That decrease causes the lung (V_l) to expand further. As the lungs expand, the airway pressure (P_l) decreases slightly which triggers air flow (Q_l) into the lungs. At the end of inhalation, the muscles relax and the recoil of the lung pulls the chest wall back to its resting balance. The airway pressure becomes slightly positive resulting in air flow out of the lungs.

A *pneumothorax* is a typical injury resulting from penetrating trauma to the chest. In Fig. 14.2, we show the variables and parameters for one type of pneumothorax, an *open sucking chest wound*, a condition in which the hole into the intrapleural space remains open. This allows air flow into and out of the intrapleural space.

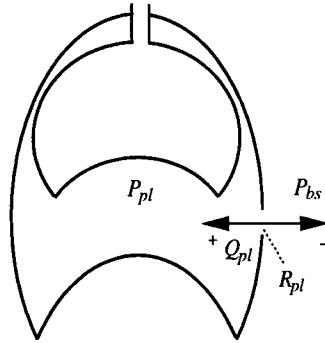


Figure 14.2 Physiological variables for a pneumothorax.

14.3 METHODS

To depict respiratory maneuvers in a virtual environment, we need to couple accurate, 3D anatomical models with physiological models that describe their mechanical behavior. If we were only interested in displaying the respiratory system, we might get by with scripted animation. However, since we are interested in simulating and ultimately predicting interactions among body systems, we need physical object models that respond to and transmit forces. Because the anatomical components are principally soft tissue, that response must include object deformation.

Object deformation must be accompanied by physiological response. Since this is a requirement for surgical simulation, we might expect to see this addressed in the literature. However, Dumay [78] points out that current, first generation surgical simulators generally focus on rendering anatomy without addressing commensurate physiological or pathophysiological models, though they recognize their importance.

This section describes our methodologies for anatomical and physiological modeling. The crux of this work is the description of the perspective we take for simulating respiratory mechanics and how we apply this to 3D modeling.

14.3.1 Anatomical Modeling

To meet real-time performance demands, we have chosen to pursue our modeling using viscoelastic, lumped-parameter models [253]. In this approach, one models surfaces or volumes as a collection of nodes and edges between nodes. Mass is concentrated at the nodes with damped springs along the edges.

The system state is described by the position and velocity of the node collection. For every two nodes that are connected by an edge in our object, we model the

edge as a damped spring and compute forces on those nodes as follows. Given position vectors for nodes **a** and **b**, the edge between them is the vector **l** such that $\mathbf{l} = \mathbf{a} - \mathbf{b}$. We record the resting length of each edge, storing it in r . We compute the force \mathbf{f}_{ab} on node **a** from interaction with node **b** through a viscoelastic element as

$$\mathbf{f}_{ab} = - \left[k_s(|\mathbf{l}| - r) + k_d \frac{\mathbf{l} \cdot \mathbf{l}}{|\mathbf{l}|} \right] \frac{\mathbf{l}}{|\mathbf{l}|}. \quad (14.1)$$

For node **b**, the force \mathbf{f}_{ba} is $\mathbf{f}_{ba} = -\mathbf{f}_{ab}$. The constants k_s and k_d are elastic and damping coefficients, respectively. Nodes have a mass (m), position (\mathbf{p}), and velocity ($\dot{\mathbf{p}}$). Applying Newton's Second Law, $m_i \ddot{\mathbf{p}}_i = \sum \mathbf{f}_{ij}$ for node i and its connections to each node j ($i \neq j$), we compute each node's acceleration to yield velocity and position using standard numerical methods.

Force Categories

In our virtual environment, objects deform and move in response to forces. We organize these forces into three categories: internal, contact, and body forces.

1. *Internal forces* are those produced from deformation through the damped springs as the object tries to return to its resting shape.
2. *Contact forces* are those that apply to surfaces as a result of pressure gradients across a surface or contact among bodies. We need to assign a 3D interpretation for force given scalar pressure differences. Currently, we apply pressure gradients along vectors that approximate the resulting shape change we want for the lung.
3. *Body forces* are those acting on all bodies in a space, such as gravity. Incorporating these types of forces is easy for force-based modeling approaches because they can be added directly to the sum of current forces on each node.

14.3.2 Physiological Modeling

We are designing our virtual respiratory system from the 'bottom up'—focusing first on modeling the mechanics of breathing, which in turn can provide the support for higher-level respiratory physiology such as biochemical and electrical activity. This higher-level physiology in turn may influence the mechanical processes by, for example, increasing respiration rate or airway dimensions such as resistance in response to hypoxia or hypercapnia.

Respiratory Mechanics

Mechanics describes the relation among forces, displacements, and their derivatives. *Respiratory mechanics* is the study of mechanics applied to the respiratory system.

In reviewing respiratory mechanics modeling, Ligas and Primiano [166] distinguish two modeling paradigms: classical and formal mechanical modeling. The distinction refers to the form of the model and variables. *Classical mechanical models* represent forces and displacements as scalar values, such as volume to represent generalized displacement and pressure differences to represent generalized forces. *Formal mechanical models* express forces and displacements as vectors, applying principles of continuum mechanics.

One important distinction between these approaches is that classical models are based on *observable* variables such as pleural pressure change, lung volume change, airway pressure, etc. Formal mechanical models, on the other hand, are based on more detailed measurements that may not be readily accessible (e.g., specific tissue segment displacements). We therefore model the physiology using the classical approach.

The basic equation for classical models relates a sum of pressure differences to a function of volume, its derivatives, and derivatives of pressure differences [166]:

$$\sum \Delta P = f(V, \dot{V}, \ddot{V}, \dots, \Delta \dot{P}). \quad (14.2)$$

In the classical model formulation, the respiratory system is modeled as a series of deformable or rigid compartments through and into which gas flows. The variable ΔP represents a pressure difference, typically between the ends of a compartment or across a surface.

For our initial effort, we consider the lung as a single-compartment, viscoelastic structure. Its ΔP to V relationship will therefore include volume and only its first derivative. We linearize (14.2) by truncating the Taylor series expansion about an operating point (P_0, V_0) :

$$\sum (\Delta P - \Delta P_0) = \partial f / \partial V (V - V_0) + \partial f / \partial \dot{V} (\dot{V} - \dot{V}_0). \quad (14.3)$$

Following [166], we represent the change about an operating point in lowercase letters. This reduces Equation (14.3) to the general form for our models:

$$\sum \Delta p = (\partial f / \partial V) v + (\partial f / \partial \dot{V}) \dot{v}. \quad (14.4)$$

In Equation (14.4), the coefficient of volume is compliance and the coefficient of the volume's first derivative is resistance.

It is important to recognize that this approach assumes that the *change* in a particular physiological variable is the same regardless of physical position in the system. It does not mean, however, that the absolute value is the same. For example, in an upright torso, the pleural pressure at the lung apex is less than the pressure at the base of the lung. Our approach merely assumes that for small maneuvers, the *change* in pleural pressure is approximately the same.

The rest of this section describes the models we have implemented using constant compliances and resistances.

Normal, Quiet Breathing

We modeled both lungs as a single compartment during quiet, normal breathing (see Section 14.3.4). The lung and chest wall are modeled as viscoelastic structures. We model the physiology with three equations. The first (14.5) expresses a volume constraint (14.5), namely that the changes in the volume of chest wall are equal to the volume changes in the lung, since the change in the volume of the intrapleural fluid is zero. The second (14.6) expresses the generalized force balance² on the chest wall (14.6). The third (14.7) expresses the generalized force balance on the lung (14.7).

$$v_{cw} = v_l \quad (14.5)$$

$$(p_{pl} - p_{bs}) + \Delta p_m = v_{cw}/C_{cw} + R_{cw}\dot{v}_{cw} \quad (14.6)$$

$$(p_{ao} - p_{pl}) = v_l/C_l + R_{aw}\dot{v}_l, \quad (14.7)$$

where the variables are pressures (p) and volumes (v) of the lung (l), chest wall (cw), pleural space (pl), body surface (bs), and airway (ao [airway opening] and aw [within the airway]). Since we model the lung and the chest wall as viscoelastic structure we represent their resistance and compliance with R and C , respectively. The term Δp_m represents the generalized force (an effective pressure difference) exerted by the chest wall muscles. For the cases we simulate, we used the following parameters:

²generalized forces are modeled as pressure differences, while the lung is a viscoelastic structure.

$$\begin{aligned}
 R_{aw} &= 1.7 \frac{\text{cm H}_2\text{O}}{\text{liters/sec}} \\
 R_{cw} &= 0.3 \frac{\text{cm H}_2\text{O}}{\text{liters/sec}} \\
 C_l &= 0.2 \text{ liters/cm H}_2\text{O} \\
 C_{cw} &= 0.15 \text{ liters/cm H}_2\text{O}.
 \end{aligned}$$

We used airway resistance (R_{aw}) and lung compliance (C_l) from the literature[4], and estimated the chest wall parameters.

Pneumothoraces

Pneumothoraces, or the pathological presence of gas within the intrapleural space, can result from penetrating trauma to the chest. Any more than a small amount of air in the intrapleural space will impede normal lung function. This can be life-threatening.

Even though our methodology in general applies to many other cases of lung pathologies, in the work presented here we will demonstrate our modeling for the *open sucking chest wound*. An open sucking chest wound is a pneumothorax in which the passage between the source of gas and the intrapleural space remains open during inhalation and exhalation. As the amount of air in the intrapleural space increases, the lung collapses and impedes respiration, in addition to other potential serious consequences resulting from mediastinal displacement (we briefly describe this at the end of this section in our discussion of the tension pneumothorax).

The penetrating chest wall hole between the body surface and the intrapleural space results in flow into the intrapleural space according to the pressure gradient across the interface:

$$P_{pl} - P_{bs} = -Q_{pl}R_{pl}, \quad (14.8)$$

where Q_{pl} is the flow of gas into the intrapleural space (neglecting gas compressibility, $Q_{pl} = \dot{V}_{pl}$), and R_{pl} is the flow resistance of the interface (see Fig. 14.2).

The chest wall and lung equations, (14.10) and (14.11), are the same as our earlier chest wall and lung equations (14.6) and (14.7), respectively, but since now that the intrapleural volume can be non-zero, we introduce the constraint in (14.9) that considers that both the lung and the intrapleural volume can vary. By linearizing (14.8) based on (14.3) we get (14.12).

$$v_{cw} = v_l + v_{pl} \quad (14.9)$$

$$(p_{pl} - p_{bs}) + \Delta p_m = v_{cw}/C_{cw} + R_{cw}\dot{v}_{cw} \quad (14.10)$$

$$p_{ao} - p_{pl} = v_l/C_l + R_{aw}\dot{v}_l \quad (14.11)$$

$$p_{pl} - p_{bs} = -R_{pl}\dot{v}_{pl}. \quad (14.12)$$

For the experiments, we use the same resistance and compliance values from the normal, quiet breathing example. We document the particular value of R_{pl} used in each experiment later. The operating points we use are

$$\begin{aligned} P_{atm} &= 1029.81 \text{ cm H}_2\text{O} \\ P_{bs_0} &= P_{atm} \\ P_{ao_0} &= P_{atm} \\ P_{pl_0} &= 1022.31 \text{ cm H}_2\text{O} \\ \Delta P_{m_0} &= 0 \\ V_{cw_0} &= \text{FRC}_{cw} \\ V_{l_0} &= \text{FRC}_l \\ V_{pl_0} &= 0 \\ \dot{V}_{cw_0} &= 0 \\ \dot{V}_{l_0} &= 0 \\ \dot{V}_{pl_0} &= -\frac{P_{pl_0} - P_{bs_0}}{R_{pl}}. \end{aligned}$$

where FRC_l is the volume at functional residual capacity of the gas space enclosed by the lung, approximately 2.2 liters and FRC_{cw} is the volume at functional residual capacity of the space enclosed by the chest wall.

We mentioned above that the open sucking chest wound can cause displacement of the mediastinum. Mediastinal displacement is a classical late sign of another type of pneumothorax, the *tension pneumothorax*. The tension pneumothorax is a prototypical example of the conditions we want to model because it involves pathological mechanical interactions that result in physiological disturbances across body systems. In a tension pneumothorax, the passage between the source of gas and the intrapleural space remains open during inhalation, but closes during exhalation. This can occur when a flap of tissue acts to block air from leaving the intrapleural space. Pressure in the intrapleural space increases, pushing against the mediastinum and the opposite lung. As a result of increased pressure on the vena cava in the mediastinum, the vein collapses and impedes

venous return. To describe this phenomenon, we plan to include in the model two lungs, each enclosed in its own hemi-thorax, with a mediastinum between them.

14.3.3 Integration

Our approach combines 3D deformable body modeling with scalar pressure and volume changes. The critical step is the manner in which we apply the scalar values to the 3D anatomical modeling. To keep the simulations in correspondence, we need to verify that the pressure/volume relationships in the scalar equations are reflected in the 3D models.

We currently apply the transmural pressure ($P_l - P_{pl}$, the alveolar pressure minus the pleural pressure) to the 3-D lung surface along vectors that approximate the shape change. The diaphragm as a separate entity does not contract, rather the chest wall (a unit representing the structures attached to the parietal pleura) moves with the effective pressure difference ΔP_m .

The correspondence in 3D with volume is direct. Under certain pressure values, the objects should have the corresponding volume as predicted by the model equations. Getting this right involves determining and assigning spring coefficients to enforce the distending-pressure/change-in-volume relationship.

14.3.4 Implementation Details

We present some details in modeling the anatomy and the physiology.

Anatomical Modeling

Our lung model is a deformable shell. The inner and outer layers are connected by damped springs. We assigned a constant k_s (spring coefficient) to the surface elements. For the elements connecting the inner layer with the surface, we assigned the k_s 's to be inversely proportional to the distance between the inner layer and the lung surface. This is a first approximation that enables different parts of the lung to expand and contract differently. The damping coefficient k_d can be computed from k_s , assuming that the lung parenchyma is roughly critically damped. Introducing correct elastic properties is beyond the scope of the work presented here. We currently can only preassign the spring coefficients.

Another approximation is our simulation of the intrapleural fluid. We want a method that approximates its function with as little computational overhead as possible. Currently, we simulate the incompressibility of the intrapleural fluid which results in a constant intrapleural volume, by enforcing at every step during the simulation a distance between the lung and the chest wall. This

distance may vary from step to step to ensure that the intrapleural volume remains constant³. When a node on the lung violates this distance, we zeroize the velocity in the direction of the surface normal. In this way, the lung can slide with respect to the chest wall, but not penetrate it. We do not model viscous shear forces, but they could be added for additional realism.

Physiological Modeling

Our physiological model assumes a single-compartment lung. We use the above described linearized models for our simulation, and our structures are assumed to have constant resistances and compliances. Also, we assume that air is incompressible, which means that air flow is equal to the change in volume for our structures, i.e., $Q = \dot{V}$ (see (14.8)).

14.4 RESULTS

This section presents our preliminary results for simulation of quiet, normal breathing and an open sucking chest wound. As the simulation proceeds, the operator sees the effects on the 3-D chest wall and lungs as well as plots of the concurrently-computed physiological variables.

14.4.1 Normal, Quiet Breathing

We simulated (14.5)-(14.7) to produce the results in Fig. 14.3. The lung and chest wall expansion in (a) are equal (600 ml) since the change in intrapleural volume is zero. In Fig. 14.3(b), the intrapleural pressure excursion is about 3 cm H₂O. The alveolar pressure is calculated as $p_l = p_{ao} - R_{aw}\dot{v}_L$. In (c), we estimated the muscle effort ΔP_m as a quarter sine wave. Fig. 14.4 shows the lung and chest wall during inhalation. Notice the increased chest wall size in (b) and the corresponding increase in lung size.

14.4.2 Open Sucking Chest Wound

For the open sucking chest wound, we need to specify the resistance of the passage R_{pl} between the gas source and the intrapleural space. The graphs in Figs. 14.5 and 14.6 show how air from the outside enters the intrapleural space, at the same time that air within the lung is released. We demonstrate this occurring both with no muscle activity Δp_m (Fig. 14.5) and with regular muscle effort (ΔP_m equivalent to the effort in the normal breathing example, Fig. 14.6). For both examples, $R_{pl} = 70$, which is about forty times the resistance of the airway (or between one-third and half the airway radius, since resistance is

³The intrapleural volume is simply the product of this distance with the lung's surface.

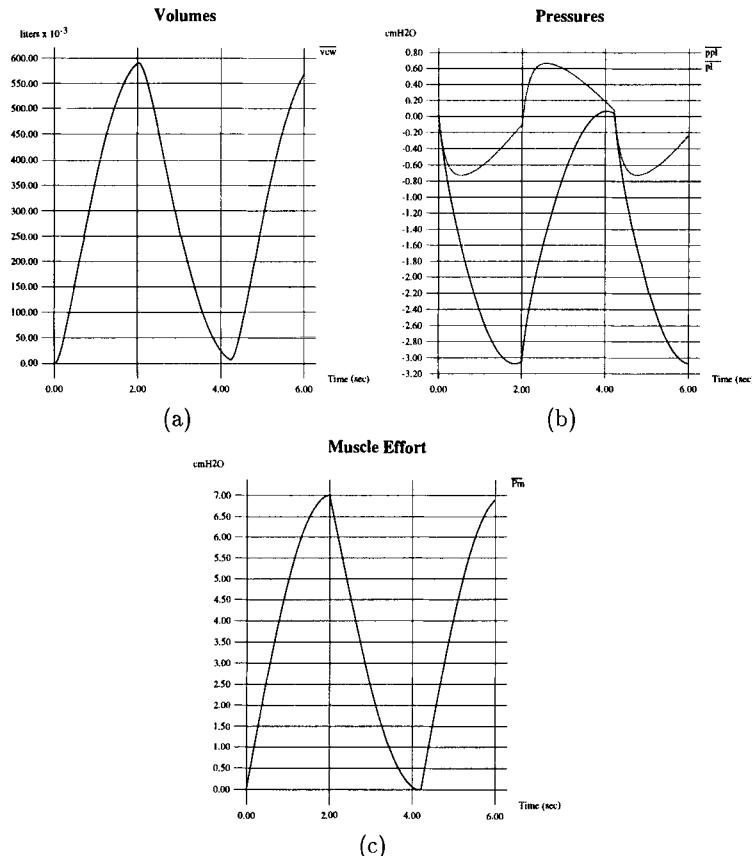


Figure 14.3 Normal, quiet breathing for a single lung compartment. In (a), the change in lung and chest wall volumes are equal, reaching approximately 600 ml. In (b), we present the change in alveolar and intrapleural pressures. In (c), we show the muscle effort ΔP_m .

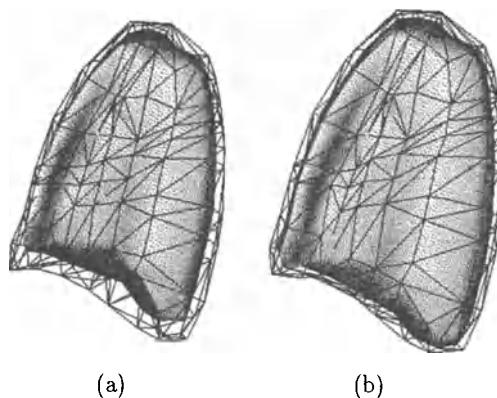


Figure 14.4 3D models showing inhalation, (a) is starting shape, (b) is during inhalation.

inversely proportional to radius to the fourth power [236]). Fig. 14.7 shows the 3D models for the pneumothorax with no muscle effort.

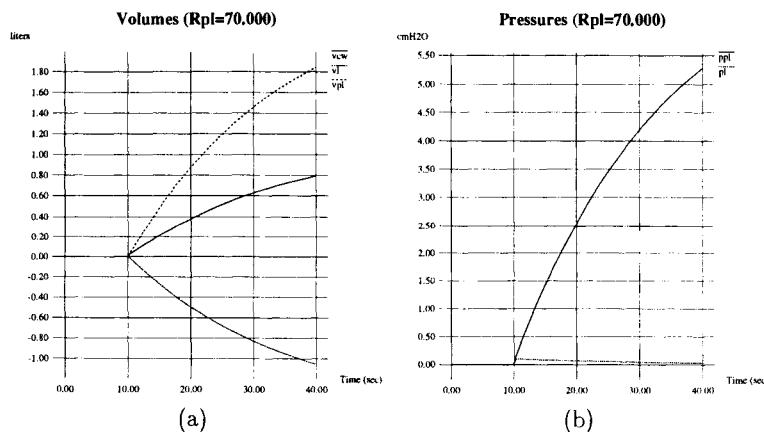


Figure 14.5 No muscle activity, passive loss of air from lung. Resistance for hole into intrapleural space is about forty times that for normal airway. In (a), the lung volume decreases, the intrapleural volume increases (from zero), and the chest wall volume increases. In (b), note the pleural pressure increasing significantly. Also, the alveolar pressure rises slightly then decreases toward P_{bs} .

Fig. 14.5(a) shows the impact on change of volumes resulting from the open sucking chest wound. Note that the chest wall volume increases because P_{pl} is increasing (shown in (b)).

While we have shown constant-amplitude muscle effort, one might imagine how the model could be used with a control module that adjusted the rate or mag-

nitude of the muscle effort based on a sensitivity to abnormal blood gas levels of O_2 or CO_2 .

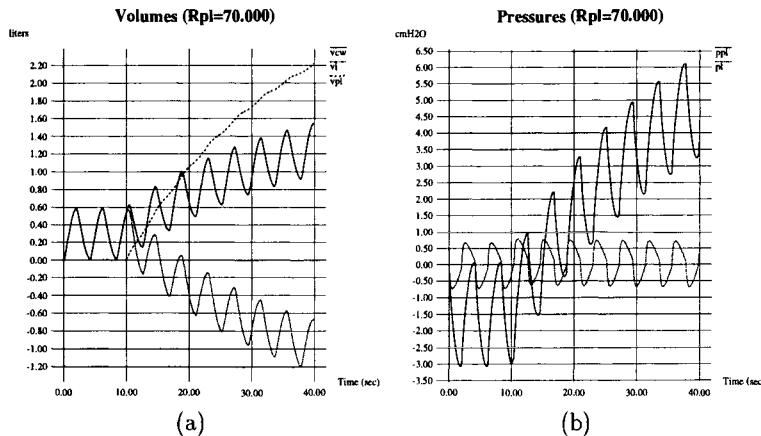


Figure 14.6 Steady muscle activity with loss of air from lung. Resistance for hole into intrapleural space is about twenty times that for normal airway. In (a), the lung volume decreases, the intrapleural volume increases (from zero), and the chest wall volume increases. In (b), we see the alveolar pressure and pleural pressure.

Fig. 14.6 shows the volumes and pressures responding to an active muscle pressure generator. Lung deflation is slightly faster than in Fig. 14.5(a). Note that the excursions (amplitude) of the lung and chest wall are constant, as a result of assuming constant compliances and a constant-amplitude muscle effort (ΔP_m). Finally, Fig. 14.7 shows in 3D the lung and the chest in case of a pneumothorax without muscle effort.

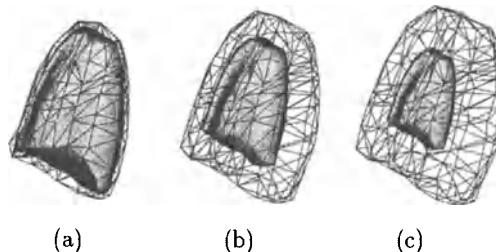


Figure 14.7 3D models showing pneumothorax without muscle effort, (a) is starting shape, (b) is during deflation, and (c) is fully deflated.

We are currently in the process of increasing the repertoire of lung pathologies we can model with our system.

RECURSIVE DYNAMICS AND ADAPTIVE CONTROL FOR ANIMATING ARTICULATED FIGURES

Although physics-based simulation is guaranteed to generate natural looking motion, it provides the user with very little control over the results produced. In kinematically controlled animation, the user is responsible for every aspect of the generated motion; however, no guarantees can be made about the result's physical correctness. In the method we present we provide the direct control of kinematic animation to dynamically generated motion, hence taking advantage of the strengths of both techniques.

Given a particular task, the degrees of freedom (DOFs) of a figure can be classified as primary or secondary. Primary DOFs are instrumental in achieving the goal and distinguish that task from all others. Secondary DOFs do not involve a particular goal and usually move involuntarily; however, they significantly enhance the overall realism of the motion. For example, for a human who is casually bending forward (Fig. 15.1), primary are the DOFs of the back. The arm DOFs can be considered as secondary since it is not imperative for their motion to meet any goal. In such a scenario, a user should be required to specify only the way the back is bending and automatically be provided with realistic arm motion. In this chapter we present an animation system which, using a robust adaptive control scheme, can link the user specified kinematic requirements on the motion of the primary DOFs to a fully automated dynamic simulation of the whole figure. For improved performance, the dynamics of the articulated figure are modeled based on a recursive dynamic formulation as opposed to the use of Lagrange multipliers presented in Chapter 7.

The desired motion of the primary DOFs is prescribed in the form of kinematic trajectories. Trajectories can be defined for either joint displacements or segment positions and orientations. For example, to animate a human figure, the user might choose to provide the desired knee and hip joint trajectory to generate a leg motion, and the position and orientation trajectory of the hand segment to generate an arm motion. Since ultimately the figure motion is generated dynamically, primary DOFs need to be actuated—provided with a force or torque which will cause them to move. In our system, actuator forces and torques which attempt to drive the primary DOFs along the desired trajectories are computed through a self-correcting dynamic control system based on

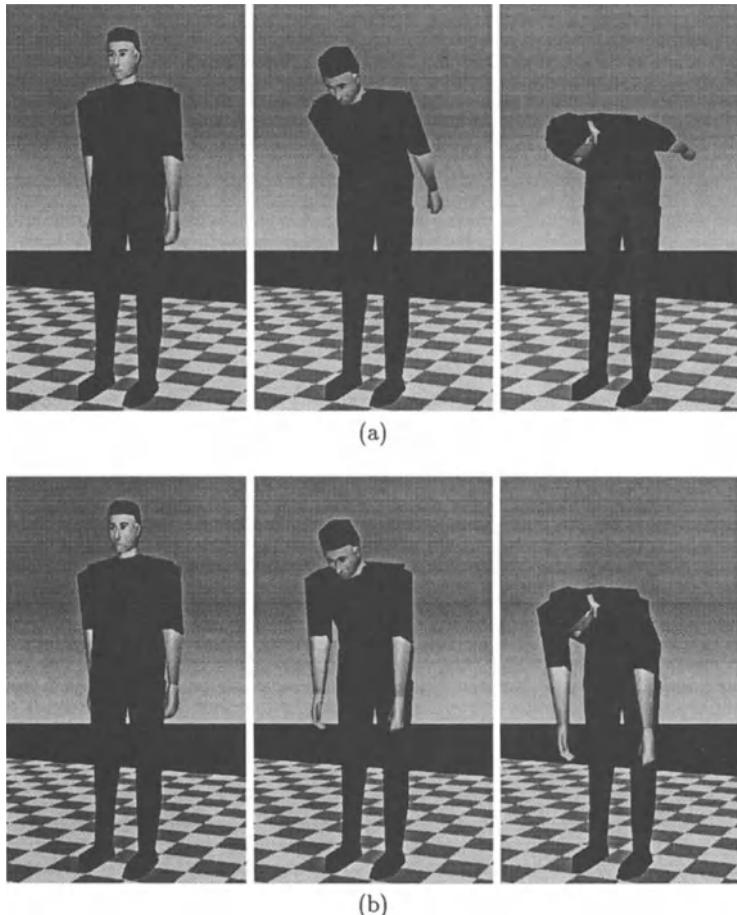


Figure 15.1 Bending the back: (a) Pure kinematic control of the back, and (b) Dynamic simulation of the upper body. Notice how gravity positions the arms naturally in (b).

the *Model Reference Adaptive Control (MRAC)* paradigm. The robust MRAC scheme, while being simple to implement and computationally efficient, allows complex trajectories to be consistently followed. Furthermore, it provides the user with direct control over the speed and type of response of the actuated DOFs to sudden changes in their desired state.

The dynamic controllers will generate forces and torques to actuate the primary DOFs of a figure. The only other forces acting on the figure are external forces due to gravity and collisions. Collision handling is usually the bottleneck of interactive dynamic animation. In our system we devised an algorithm which breaks up the collision response problem into two distinct phases—an impact and a contact phase. The impact phase occurs when two objects first collide and results in an instantaneous change of the objects' velocities. The contact phase continues for as long as the objects are touching and produces the contact force between them that will prevent interpenetration. This two stage approach gives physically accurate collision behavior with relatively low computational cost. With all the forces acting on a figure known, a forward dynamics simulator completes the animation procedure and computes the figure motion.

The advantages of using dynamic simulation for animating a figure whose desired motion of the primary DOFs is already specified are the following: (1) Physical validity of the motion is guaranteed by the dynamic simulation which handles collisions and joint limits, (2) Natural passive motion of secondary DOFs is automatically generated, leaving the animator free to direct specific aspects of the active motion, (3) Feasibility of motions can be studied given realistic actuator limits. A desired motion will be followed as closely as possible within the specified physical constraints of the system, and (4) Segment position and orientation trajectories provide powerful tools which do not suffer from the occasional jerkiness present in inverse kinematics solutions. They can be particularly useful in following trajectories provided by sensors of a motion-capture system.

Our system maintains low computational cost and small user involvement in the dynamic simulation. A recursive forward dynamics technique based on the work of Featherstone [83] was enhanced to effectively handle articulated figure collisions. There is no overhead in simulating the motion of new figures since their dynamic equations need not be derived symbolically. The dynamic control scheme used to generate driving forces for the simulation adjusts automatically to its assigned task and needs no special tuning from the user. In fact, generating an animation with our system requires no more effort than any kinematics-based system would.

The system described in this work is built as an extension to *Jack[®]*, the human modeling and simulation package developed at the University of Pennsylvania [220, 14]. The user can take advantage of the advanced interactive manipulation features of *Jack[®]* to specify complex kinematic trajectories and achieve enhanced motion through the dynamic simulation.

The remaining sections in this chapter describe the animation system in detail with special emphasis given to the collision response and the dynamic control algorithms. We provide the equations used to handle impact between articulated figures and a step-by-step account of how to build a MRAC system for trajectory following.

15.0.3 Related Work

Researchers have tried to tackle problems in generating physically correct purposeful motion for computer animation in a variety of ways and for a variety of applications. Raibert, Hodgins and their colleagues have developed dynamic control algorithms that deal with motion of machines and simulated humans [228, 117, 116]. Their controllers were hand-crafted to assure successful and natural looking motion for the models they used. Dynamic controllers tailored for particular simulated figures were also used by McKenna and Zeltzer [191] and Bruderlin and Calvert [42]. In recent years, a number of techniques for automatically generating motion for particular behaviors have been presented [292, 54] which automatically determine an optimal trajectory, a suitable control algorithm or even a morphological structure for the simulated system [262]. These approaches completely free the user from specifying the details of the motion but unfortunately their use is limited to simple systems and basic behaviors.

Our work does not deal directly with generating a particular behavior on a certain figure. Instead we provide a general system for animating articulated figures by giving the user a substantial amount of low level control. Isaacs and Cohen [126] presented a system which blends kinematic and dynamic motion in a figure by removing the kinematically controlled DOFs from the dynamic formulation for the figure. Barzel and Barr [25] control the motion of figures through constraint forces although their algorithm is not particularly suited for articulated figures with complex joints since it provides no direct control over joint angles. Our work has similarities to the work of Lamouret and Gascuel [153] who use dynamic controllers to drive motion along kinematically specified paths in space and to synchronize the animation of different objects. However, in their system, articulated figure motion cannot be described through desired joint trajectories thus complicating user control of complex figures. Furthermore, the Proportional Derivative (PD) control scheme used for the actuators lacks the robustness and ability to adapt to a variety of dynamical systems and motions provided by adaptive control, as we will explain in a section 15.4.

15.1 SYSTEM DESCRIPTION

Figure 15.2 shows the overall structure of the animation system. The user input is mostly limited to defining the objects and their geometry and providing the desired motion trajectories for the primary DOFs. Given the specified trajectories and the figure state computed by the dynamic simulator, the MRAC

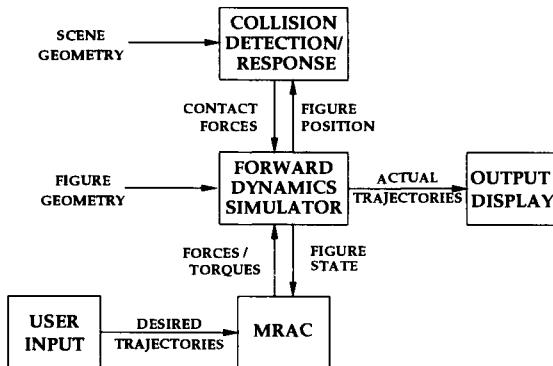


Figure 15.2 The overall structure of the animation system.

calculates the forces and torques to be applied at the actuated DOFs. The collision response module interacts with the dynamic simulator to prevent penetrations between figures.

15.1.1 Figure Representation

A typical environment contains one or more *figures*. Each figure consists of rigid *segments* with defined geometric shape connected together through joints in a tree structure. Every segment has only one *parent segment* to which it is connected through its *parent joint*. The only exception is the *root segment* of the figure, which has no parent. Joints connecting the segments can have up to six degrees of freedom (*DOFs*), three translational and three rotational.

Attachment points on a segment are called *sites*. The root joint of each segment connects the segment's *root site* to one of its parent's sites. Every site is the origin of a Cartesian system attached rigidly to the segment. The Cartesian system attached to the root site is the *local frame* of the segment. The environment has a fixed coordinate system called the *inertial frame*.

Any kinematics based animation system should have the above parameterization to generate motion. For a dynamic simulation, the mass (or density) of each segment should be provided. Assuming uniform mass distribution, the center of mass and inertia tensor of each segment can be automatically computed from the segment's geometry. For added realism, a damping element can be added to the joints to simulate mechanical energy dissipation due to joint friction or muscular stiffness.

15.1.2 Desired Kinematic Trajectories

Trajectories can be specified for both joints and sites. A joint trajectory describes joint displacements over time and a site trajectory position or orientation over time. Joints or sites that have been assigned a trajectory are called *actuated* since there a force or torque acting on them will make them follow that trajectory. A joint actuator is internal to the figure and plays a role similar to a muscle or a motor. A site actuator acts much like a string on a marionette, providing an external force at the site. Typically, the user decides to power the primary DOFs which are the most important to a particular motion and allows the secondary DOFs to follow naturally. Kinematic trajectories reflect a desired motion and can be constructed in advance or while the animation is progressing. An interesting source for site trajectories is motion-captured data which can be used to drive the motion of a simulated articulated figure if sites are placed on the same positions on the figure as sensors on the real body. The advantage of using a dynamic simulation instead of playing back the raw kinematic data in this case is that we prevent physically incorrect situations (such as segment interpenetration) due either to inconsistencies between the real and the simulated world or sensor noise.

15.2 EFFICIENT FORWARD DYNAMICS

A forward dynamics simulator computes the figure motion when all external and internal forces are known. For an interactive system, the simulator should be computationally efficient and able to handle arbitrary complex figures without user involvement. The forward dynamic simulator in our system is based directly on Featherstone's Articulated Body Method (ABM) [83], an efficient recursive procedure which accommodates a variety of joint types. The algorithm runs in time proportional to the number of DOFs for any articulated figure without closed loops. Contrary to the equivalent closed form Lagrangian formulation with implicit joint constraints ([59]), no symbolic preprocessing stage is necessary to develop the equations of motion and there is therefore no additional overhead for each new figure. Moreover, solutions of the closed form of the Lagrange equations typically have at least quadratic complexity.

15.2.1 The Spatial Notation

The ABM algorithm is formulated in *spatial notation* which uses 6-dimensional vectors to account simultaneously for the linear and angular components of physical quantities, such as velocities, accelerations and forces. The use of spatial quantities leads to elegant solutions of rigid-body dynamics problems with a significantly reduced number of equations. Although a complete coverage of spatial algebra can be found in Featherstone's book, the brief introduction to spatial quantities in the remainder of this section should be sufficient for following the notation used in our work.

The *spatial velocity* $\hat{\mathbf{v}}$ (the $\hat{\cdot}$ denotes spatial quantities) of a rigid body with respect to an origin O is represented by the vector $[\omega^T \mathbf{v}_O^T]^T$ where \mathbf{v}_O is the linear velocity of a point moving with the object, instantaneously coincident with O and ω is the object's angular velocity.

The *joint axis* is a spatial vector which defines the direction and nature of motion allowed by the joint. The relative velocity of two bodies is obtained by multiplying the joint axis by the scalar joint velocity. A revolute joint which allows rotation about an axis \mathbf{s} is represented as $\hat{\mathbf{s}} = [\mathbf{s}^T \mathbf{0}^T]^T$. A prismatic joint which allows pure translation along axis \mathbf{s} has the form $\hat{\mathbf{s}} = [\mathbf{0}^T \mathbf{s}^T]^T$.

Any *force* acting on a rigid body can be represented as a vector \mathbf{f} acting along a line passing through the origin O together with a couple τ_O . In spatial notation, the force is written: $\hat{\mathbf{f}} = [\mathbf{f}^T \tau_O^T]^T$.

The *spatial rigid-body inertia* $\hat{\mathbf{I}}$ is a 6×6 matrix which transforms the spatial velocity $\hat{\mathbf{v}}$ of a rigid body into its spatial momentum $\hat{\mathbf{P}}$ through the equation $\hat{\mathbf{P}} = \hat{\mathbf{I}}\hat{\mathbf{v}}$.

A *spatial transformation matrix* ${}_P\mathbf{X}_O$ is a 6×6 matrix transforming a spatial quantity from frame O to frame P .

Finally, the *spatial transpose* of a vector $\hat{\mathbf{a}} = [\mathbf{a}^T \mathbf{a}_0^T]^T$ is a 1×6 vector $\hat{\mathbf{c}}^S = [\mathbf{a}_0^T \mathbf{a}^T]$.

15.2.2 Articulated Figure Structure

In its most general form, an articulated structure in Featherstone's formulation consists of links connected together with single degree of freedom joints, each with a corresponding joint axis $\hat{\mathbf{s}}$. A multiple degree of freedom joint can be represented as a chain of the appropriate number of single degree of freedom joints. The state of a figure can be fully described by the position, $\hat{\mathbf{p}}_R$, and velocity, $\hat{\mathbf{v}}_R$, of the root link and the displacement, q_l , and its derivative, \dot{q}_l , of the joint attached to each link l . The velocity of any link l , $\hat{\mathbf{v}}_l$, can be recursively obtained from the velocity of its parent link p , $\hat{\mathbf{v}}_p$, the joint axis $\hat{\mathbf{s}}$ and \dot{q}_l using:

$$\hat{\mathbf{v}}_l = \hat{\mathbf{v}}_p + \dot{q}_l \hat{\mathbf{s}}_l.$$

In the following sections we will describe the additions to the basic form of the forward dynamic simulator needed to complete our animation system.

15.3 COLLISION HANDLING

One of the most important features of a dynamical simulation is the ability to automatically handle collisions between objects. Modeling impact and contact

is the bottleneck of a dynamic simulation. There are two distinct aspects to collision handling: *detection* and *response*. In a simulated world, collisions between segments can be detected by examining the segment geometry, position and orientation. This process needs to be repeated at every step of the simulation and can be very costly for geometrically complex environments. When a collision is detected, appropriate action must be taken to ensure proper response. In a physics-based simulation, repulsive forces between the colliding rigid objects prevent interpenetration of the objects.

Several algorithms, varying in sophistication and efficiency, exist for detecting collision between polygonal surfaces [196]. In our system we use a fast bounding box check to pick which segment pairs could be in contact. Only for these segment pairs, a more accurate technique based on the work of [100] is applied to obtain the contact position if one exists.

The most popular technique for dealing with collision response in a dynamically simulated environment is inserting a fictitious spring-damper element at the contact point between the two segments and allowing small penetrations of the segment geometries [228, 116, 191]. The repulsive force is a function of the penetration distance and velocity. A spring-damper element is easy to implement and computationally inexpensive. Ideally, a spring should be stiff enough to allow only minimal penetration between objects even when they collide with high velocities. However, stiff springs lead to a stiff dynamical system that requires slow integration. Choosing the spring and damper constants is non-trivial since a good choice depends on the physical properties and collision velocities of the colliding objects. An interesting but more involved technique for the computation of contact forces between rigid non-articulated bodies without using spring-damper elements is described in [20].

The method we use for dealing with collision response is automated and has small effect on the dynamic simulator's integration rate. We divide the collision of two objects in two stages, the *impact* and the *contact* stage. At the impact stage, which lasts an infinitesimally short time, the velocity of the involved objects changes instantaneously whereas their position and orientation remain constant. Once a new collision is detected during the course of the forward dynamic simulation, we temporarily halt the simulation, compute new object velocities from their current velocities, and restart the simulation. When the simulation restarts, the objects are still in contact since their position and orientation did not change during the impact. If the collision is elastic and the objects separate in the next time instant, then no further steps need to be taken. If however the colliding objects do not separate and for as long as they stay in contact, repulsive forces are generated to prevent interpenetration.

For the contact stage we use a spring-damper element at every contact point. However, springs generating contact forces after impact need not be stiff. A mild spring can adequately prevent interpenetration of objects since the relative velocity of the contact points is always zero after the impact. A reasonable

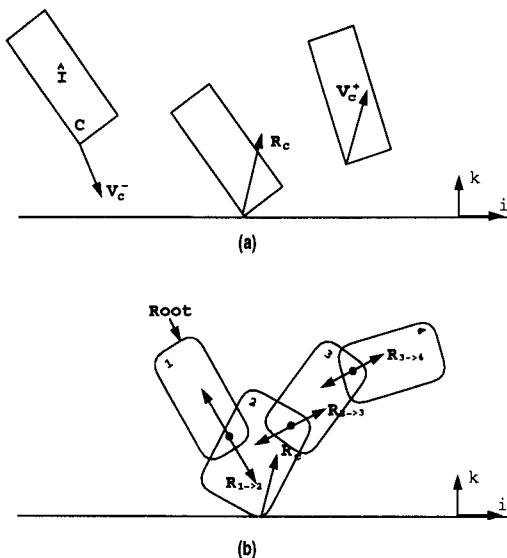


Figure 15.3 Impact of (a) a simple figure and (b) an articulated figure with a fixed surface. The collision impulse is R_c . The law of action and reaction holds for impulses between connected segments in a figure.

choice for the spring and damper constants can work for a variety of objects and motions.

In our simulated environment there can be collisions between two dynamically animated figures as well as between a dynamical and a stationary (non-dynamical) figure such as a wall or a floor. Two segments can come in contact in multiple contact points. The following section provides a description of the equations needed for the impact stage of our collision response algorithm.

15.3.1 Impact stage

Impact occurs when two objects collide and results in an instantaneous change in their velocities. Using the principle of conservation of momentum one can formulate an analytical solution for the impact between two arbitrary articulated figures. For strong collisions, an analytical solution is advantageous compared to the spring-damper approach, because it bypasses the problem of large collision forces. At the time of the impact, a linear system of equations is formulated and solved to obtain the post-impact velocities of the figures. The equations presented in this section grew out of the work in [196] and are adapted to match the dynamic figure representation required by Featherstone's algorithm. All the quantities in this section are assumed to be expressed in the inertial frame.

As a simple first case, we will consider a rigid object colliding with a fixed flat surface (see Fig. 15.3(a)). If $\hat{\mathbf{I}}$ is the spatial inertia of the object then the relationship between the spatial velocity $\hat{\mathbf{v}}^-$ before and $\hat{\mathbf{v}}^+$ after the impact is given by

$$\hat{\mathbf{I}}\hat{\mathbf{v}}^+ = \hat{\mathbf{I}}\hat{\mathbf{v}}^- + \hat{\mathbf{R}}_c, \quad (15.1)$$

where $\hat{\mathbf{R}}_c = [\mathbf{R}_c^T \ \mathbf{t}_c^T]^T$ is the unknown collision impulse from the surface to the object. If O is the origin of the object and C is the collision point, then the velocity of point C is given by

$$\hat{\mathbf{v}}_C = [\mathbf{v}_C^T \ \boldsymbol{\omega}^T]^T = {}_C\mathbf{X}_O \hat{\mathbf{v}}.$$

where ${}_C\mathbf{X}_O$ is a transformation from point O to point C . The collision frame is defined by three orthogonal unit vectors, \vec{k} perpendicular to the contact surface and \vec{i} and \vec{j} in the plane of the surface. With the elasticity coefficient of the collision $e \in [0, 1]$ we use the following relationship to compute the perpendicular to the surface post-impact velocity of the contact point

$$\mathbf{v}_C^+ \cdot \vec{k} = e(\mathbf{v}_C^- \cdot \vec{k}). \quad (15.2)$$

When $e = 0$ the collision is inelastic and when $e = 1$ the collision is perfectly elastic. If there is no friction at the contact point then there can be no impulse along the \vec{i} and \vec{j} directions

$$\mathbf{R}_c \cdot \vec{i} = \mathbf{R}_c \cdot \vec{j} = 0 \quad (15.3)$$

and if we assume infinite friction then the velocity along the surface will be zero

$$\mathbf{v}_C \cdot \vec{i} = \mathbf{v}_C \cdot \vec{j} = 0. \quad (15.4)$$

For the time being we do not handle arbitrary friction coefficients during the impact stage.

So far we have found nine linear equations (six from (15.1), one from (15.2) and two from (15.3) or (15.4)) with twelve unknowns (six for $\hat{\mathbf{v}}^+$ and six for $\hat{\mathbf{R}}_c$). The remaining three equations are obtained from the dependency of the linear and angular components of $\hat{\mathbf{R}}_c$ which take the form

$$\mathbf{t}_c = \mathbf{r}_C \times \mathbf{R}_c, \quad (15.5)$$

where \mathbf{r}_C is the position of point C in the inertial frame.

This basic principle of momentum conservation can be readily extended from the simple solid object to an arbitrary articulated figure (see Fig. 15.3(b)). For each segment l , the general form of (15.1) is

$$\hat{\mathbf{I}}_l \hat{\mathbf{v}}_l^+ = \hat{\mathbf{I}}_l \hat{\mathbf{v}}_l^- + \hat{\mathbf{R}}_{p \rightarrow l} + \sum_{all \ d} (-\hat{\mathbf{R}}_{l \rightarrow d}) + \sum_{all \ c} \hat{\mathbf{R}}_c. \quad (15.6)$$

A segment l receives an impulse $\hat{\mathbf{R}}_{p \rightarrow l}$ from its parent, p , through their connecting joint. From the law of action and reaction, p receives an impulse $-\hat{\mathbf{R}}_{p \rightarrow l}$

of equal magnitude but opposite direction, and hence for each segment l we need to add the impulses received from all its children d , $-\hat{\mathbf{R}}_{l \rightarrow d}$. Finally, each active contact point c on the segment contributes an external impulse $\hat{\mathbf{R}}_c$. A joint cannot transfer impulse in a direction along its axis \hat{s} , and therefore for each degree of freedom f of the root joint of segment l we need an equation of the form

$$\hat{s}_f^S \hat{\mathbf{R}}_{p \rightarrow l} = 0. \quad (15.7)$$

To compute the joint velocities right after the impact, for each DOF f we also need the equation which relates the velocity of a segment to the velocity of its parent

$$\hat{\mathbf{v}}_l^+ = \hat{\mathbf{v}}_p^+ + \sum_f \dot{q}_f \hat{s}_f. \quad (15.8)$$

The equations relating to the contact point velocity and impulse are the same as in the simple rigid object case. For each contact point c in every segment we need (15.2), (15.3) or (15.4) and (15.5). The equations described form a complete system which can be solved to obtain the post impact segment and joint velocities. Although for a complex figure the system to be solved is large, the coefficient matrix is sparse. A sparse matrix technique can be used to achieve efficiency.

Only minor modifications to the above scheme are required to generalize it for impacts between two dynamic figures or between segments of the same figure: when an impulse $\hat{\mathbf{R}}_c$ is applied to one of the two contacting segments, an impulse $-\hat{\mathbf{R}}_c$ is applied to the other. The absolute velocity of the contact point, $\hat{\mathbf{v}}_C$, is replaced by the relative velocity of the contact points of the two segments. We have also worked out the details for handling different types of common contacts between polygonal surfaces, involving edges or planes instead of single points which is beyond the scope of this work presented here.

15.4 DYNAMIC CONTROL

An animation system needs to provide means for controlling the motion to be generated. Contrary to kinematics based systems where motion is driven directly by setting kinematic trajectories such as object positions and joint angles over time, physics-based systems use as input only forces and torques. For a user, specifying a desired object trajectory in Cartesian coordinates is generally more intuitive than giving the force that would make the object move along it. If it were not for the dynamic controllers, forward dynamics would almost be useless for generating purposeful motion! A dynamic controller links the direct kinematic control with realistic looking dynamic animations.

A dynamic controller acts on a specific dynamic system ranging in complexity from a single joint to multi-DOF articulated figure. The controller and the dynamic system together are called a *control system*. The inputs to the controller are the desired values of the system's variables, and the output is a set of forces

or torques which, when applied to the system, attempt to create the desired effect on the actual values of the same variables. The goal of the control system is to minimize the discrepancy between the desired and the actual values. For example, if the dynamic system is a human arm and the desired motion is the elbow flexion, then a dynamic controller can produce the required torque at the elbow joint to flex it as desired.

There are two main classes of controllers, the *open loop* and the *feedback* controllers. Open loop controllers base their output only on the desired state of the dynamic system and are much less powerful than the feedback controllers which take into account the actual state of the system as well. A typical example of an open loop control strategy for articulated figures is *inverse dynamics* which, given the desired joint angles, velocities and accelerations of the figure, computes the required internal torques to achieve them.

The simplest to implement and most commonly used type of closed loop controller is the *Proportional Derivative (PD)* controller [126, 228, 116, 117]. The output torque of the PD controller is proportional to the difference in position and velocity between the desired and the actual state, which makes it behave similarly to a spring-damper system

$$f = K_p(x_{desired} - x_{actual}) - K_d(\dot{x}_{desired} - \dot{x}_{actual}).$$

Since a PD controller assumes nothing about the dynamical characteristics of the system to which it is applied, its successful performance relies heavily on the fine tuning of its two parameters, namely, the proportional and the derivative gains, K_p and K_d . Tuning of the controller needs to be done through trial and error and the gains depend on the characteristics of both the system and the desired motion. Gains that work fine for a slow motion could be inappropriate for a motion involving high accelerations and vice versa. An animation system using PD controllers would require laborious manual tuning of the gains to successfully achieve a desired motion.

15.4.1 Model Reference Adaptive Control

The *adaptive* control systems used in this work have evolved from the need to implement high performance control systems that assume little about the dynamic characteristics of the system to be controlled. The fundamental characteristic of adaptive control systems is the presence of a supplementary feedback loop acting upon a performance index of the control system. Among the many solutions which have been proposed to make a control system 'adaptive', a special class called *Model Reference Adaptive Control (MRAC)*, uses the innovative idea of a reference model to specify the desired system performance [154, 264, 265].

The advantages of using MRAC to generate the forces and torques driving a figure along kinematically specified trajectories are many: (1) MRAC systems

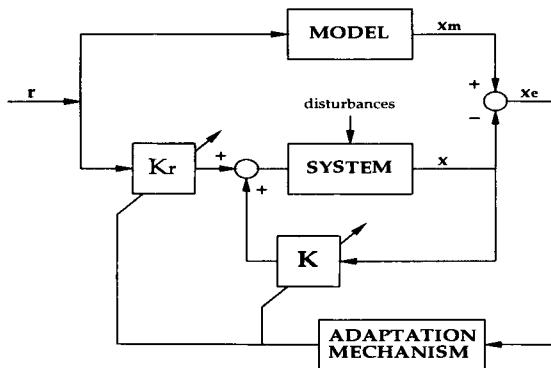


Figure 15.4 The structure of a MRAC. The model and the actual system run in parallel. The difference in their state is fed to the adaptation mechanism which subsequently modifies the gains \mathbf{K} and K_r .

are easy to implement and computationally efficient, (2) Since MRAC assumes little about the system it is being applied to, one basic controller design can be used to successfully control a variety of dynamic systems, (3) The self-adjusting nature of the controller relieves the user from explicitly setting gains or other non-intuitive parameters, and (4) MRAC design gives the user direct control over the ideal behavior of the controlled system. For example, an animator could vary the speed in which figures should react to changes in their desired motion, thus simulating a variety of muscular response times for animated characters.

Fig. 15.4 shows a typical configuration of a model reference adaptive control system. The reference model which specifies a given index of performance in terms of inputs and model states is a part of the control system itself. The MRAC tries to force the state of the system \mathbf{x} to follow the model state x_m even in the presence of disturbances or open loop differences. The adaptation mechanism modifies the control variables \mathbf{K} and K_r , and synthesizes an auxiliary input signal to assure good model following.

The dynamics of an articulated figure with n DOFs can be written as

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{x} + \mathbf{B}(\mathbf{q})\mathbf{u}, \quad (15.9)$$

where \mathbf{q} is the $n \times 1$ vector of the root segment position and the joint displacements, $\mathbf{x} = [\mathbf{q}^T \dot{\mathbf{q}}^T]^T$ is the $2n \times 1$ state vector and \mathbf{u} is the vector of external and internal forces acting on the figure. Matrices \mathbf{A} and \mathbf{B} can be symbolically derived from the figure structure and a coupled MRAC can be designed to control the motion of any number of the figure's DOFs.

Alternatively, the same dynamic system with n DOFs can be written as n 1-DOF dynamic simpler systems each one dealing with a single DOF q of the original system. The equation of each simple system can be written as

$$\dot{\mathbf{x}}_q = \mathbf{A}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{x}_q + \mathbf{B}(\mathbf{q})u_q \quad (15.10)$$

or

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_{q1}(\mathbf{q}) & -a_{q2}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ b_{q1}(\mathbf{q}) \end{bmatrix} u_q.$$

The n simple systems are coupled and therefore scalars a_{q1} , a_{q2} and b_{q1} depend on the values of all the DOFs.

This set of dynamic systems can be controlled using a *decoupled* MRAC approach. A simple controller can be attached to each DOF of the figure that needs to behave in a specified manner. Decoupled MRACs have been proven to be robust and nearly as effective as the coupled MRAC while at the same time they are easier to implement and are computationally more efficient. For each DOF q of the figure, we construct a *reference model* system which has the form

$$\dot{\mathbf{x}}_{mq} = \mathbf{A}_{mq}\mathbf{x}_{mq} + \mathbf{B}_{mq}r_q, \quad (15.11)$$

and the model following error for q is defined as

$$\mathbf{x}_{eq} = \mathbf{x}_{mq} - \mathbf{x}_q. \quad (15.12)$$

To complete the MRAC, we define the control input u_q of (15.10) to be

$$u_q = \mathbf{K}_q(t)\mathbf{x}_q + K_{rq}(t)r_q, \quad (15.13)$$

where $\mathbf{K}_q(t)$ is a 1×2 vector and $K_{rq}(t)$ scalar. The objective of the MRAC is to force \mathbf{x}_{eq} asymptotically to zero in a controlled fashion and achieves that by setting the proper values of the adaptive gains \mathbf{K}_q and K_{rq} .

We devote the following section to present a step by step procedure on how to synthesize a MRAC to do trajectory following for a single DOF of a dynamic figure. The decoupled MRAC approach is elegant for controlling multiple DOFs at the same time, can be achieved by assigning a different MRAC to each DOF.

15.4.2 Sample MRAC synthesis

This section gives a “cookbook” description for a procedure which results in a successful MRAC synthesis. Model Reference Adaptive Control is a powerful technique and the interested reader should consult the original published work in [154, 264, 265] for more details.

Assume that a joint trajectory is given from which we can extract the joint angle and velocity at any time instant. The goal is to synthesize a MRAC which generates the torque within specified limits which would move the actual joint close to the trajectory in a controlled manner. The steps to be followed are

1. *Design the reference model:* The dynamical systems we are interested in are driven by finite forces and therefore cannot change their state instantaneously. The greater the force applied to the system, the faster its state

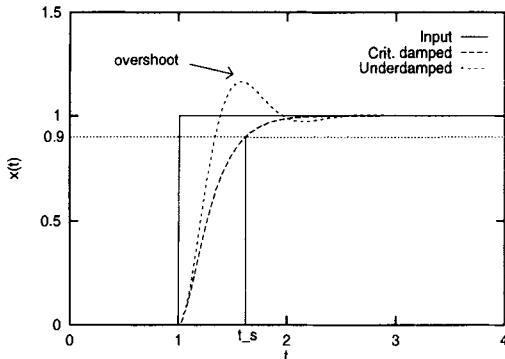


Figure 15.5 Different responses to a step input. Underdamped motion reaches the desired state faster but overshoots. Critically damped motion provides the fastest response without overshooting. The settling time t_s is shown for the critically damped case.

changes. Given a particular goal position, if the force supplied is too low, the system will move too slowly. If the force is too great, the system will reach the goal quickly, but will not be able to stop in time and will overshoot, as shown in Fig. 15.5. The model system will act as an index of performance of the actual system. We therefore need to design it so that it behaves the way we would like the actual system to behave. The MRAC will try to minimize the difference between the model and the actual system's state, thus driving the actual system as closely as possible to its ideal behavior. The model system is defined from (15.11) where

$$\mathbf{A}_{mq} = \begin{bmatrix} 0 & 1 \\ -a_{mq1} & -a_{mq2} \end{bmatrix}, \quad \mathbf{B}_{mq} = \begin{bmatrix} 0 \\ b_{mq} \end{bmatrix}.$$

For good trajectory following, a critically damped behavior of the model system gives the fastest response possible with no overshoot. Critically damped behavior can be achieved by setting $a_{mq1} = b_{mq} = \lambda^2$, $a_{mq2} = 2\lambda$ with $\lambda = 4/t_s$, where t_s is the required *settling time* of the system [151]. The settling time can be set by the user to adjust the speed of response of the controlled system. Typically, the settling time is defined as the time taken for the system to reach within 10% of its desired value. The lower the settling time, the faster the system has to move to its desired state which translates to higher force or torque generated by the controller. By setting lower settling time for a figure's actuated DOFs, an animator can achieve quick responses. A high settling time would give a sluggishly moving figure.

2. *Generate the model input:* With the model system defined, the controller is ready. The model and the actual systems will be integrated at the same time to observe the difference in their states. At each time instant, the input r_q to the model system is determined by inverting the reference model dynamics

$$r_q = \mathbf{B}_{mq}^+ (\dot{\mathbf{x}}_{dq} - \mathbf{A}_{mq} \mathbf{x}_{dq}), \quad (15.14)$$

where $\mathbf{B}_{mq}^+ = [0 \ 1/b_{mq}]$. This input will instantaneously drive the model system towards the desired state x_{dq} obtained from the user provided desired trajectory of DOF q .

3. *Compute Control Signal:* The control signal u_q is computed from the following equations ([154])

$$\begin{aligned}\mathbf{x}_{eq} &= \mathbf{x}_{mq} - \mathbf{x}_q \\ y &= \begin{bmatrix} 16 & 4 \\ t_s^2 & t_s \end{bmatrix} \mathbf{x}_{eq},\end{aligned}\quad (15.15)$$

$$\mathbf{K}_q(t) = \int_0^t \alpha y \mathbf{x}_q^T d\tau + \beta y \mathbf{x}^T, \quad (15.16)$$

$$K_{rq}(t) = \int_0^t \alpha y r d\tau + \beta y r, \quad (15.17)$$

$$u_q = \mathbf{K}_q(t) + K_{rq}(t)r_q.$$

(15.16) and (15.17) are the heart of the adaptation process and guarantee the asymptotic convergence of \mathbf{x}_{eq} to 0 [264]. Constants α and β depend on the characteristics of the dynamic system but can be chosen within a wide range of values and do not have a significant effect on the trajectory following performance of the controller.

To enhance the realism of a dynamic animation it is necessary to limit the outputs of the controllers just as the outputs of mechanical motors and human muscles are limited. In the MRAC scheme, when the control signal computed from (15.13) is higher than the maximum allowed torque then the integration in (15.16) and (15.17) must be halted and the old values of the integrals should be kept. The output of the controller is set to the maximum allowed value and the integration is stopped until the signal returns to within its specified range.

15.5 RESULTS

We have conducted a series of experiments to test our system and determine possible limitations. We have applied the forward dynamic simulator with the collision handling module to a variety of articulated figures, from simple pendulums to a human figure model. For simple articulated figures (e.g., ten segments), computation time is not a problem; typically, simulations can go faster than real time. As figures get more complicated, the running time increases, but interactive rates can still be maintained, especially when the number of impacts is not excessive. Impact equations for a complex figure result in a large linear system which is expensive to solve. However, impacts do not cause a decrease in the integration time step like stiff springs do; furthermore, they preserve the numerical stability of the dynamic system much better. A difficult test case for the collision response system is the simulation of a soldier falling on his back after being shot in the chest. The velocity of the soldier's back

when he hits the ground combined with the large mass of his body give rise to high impact forces. Very slow integration would be required if springs were used to handle the impact. Our system handles the impact and the contact at the back and the legs without a significant delay. The motion shown in Fig. 15.6 is generated by exerting a large initial impulse on the chest of the soldier, which results in his fall to the ground.

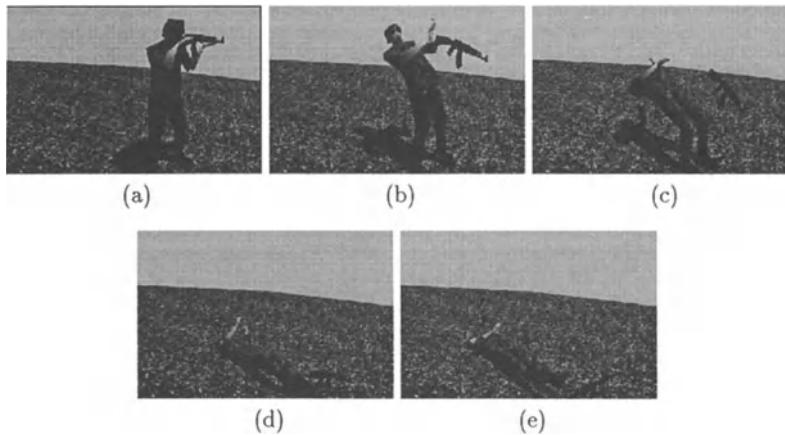


Figure 15.6 A wounded soldier. The soldier receives a large impulse at his upper chest and falls on the ground.

The adaptive controllers proved to be powerful in driving the controlled DOFs along desired trajectories. A number of trajectories were used to drive the joints and sites of articulated figures. As expected, for smooth desired trajectories there was almost no error (Fig. 15.7). On discontinuous trajectories like the one shown in Fig. 15.8, the error varies depending on the choice of the settling time t_s of the model reference system. Lower settling time forces faster response to changes in the state, while higher settling time makes the system take longer to adjust and results to greater error. For both figures, we used a human figure model which was dynamically animated from the waist up. We actuated the waist joint and we allowed the rest of the joints (arms and upper torso) to move freely. Despite the disturbances created by the free motion of the rest of the joints, the waist followed the desired trajectory accurately. This shows the power of the MRAC to continuously adapt to the system it controls and to overcome external disturbances.

Fig. 15.9 demonstrates an attempt to follow a physically incorrect trajectory which involves penetration between the arms and a table. Although kinematically the motion is valid, when played back dynamically the contacts are detected and penetration is prevented. A motion trajectory is originally provided for the shoulders and the elbow. Once the contact forces prevent any further motion, the output of the joint controllers reaches its maximum value and stays there, stubbornly trying to make the arms follow the desired motion.

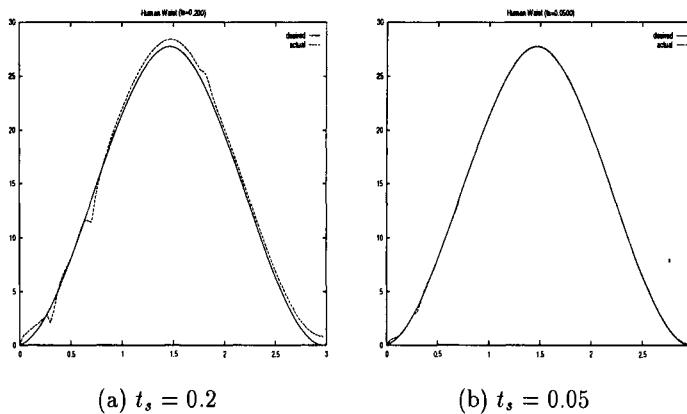


Figure 15.7 Trajectory following for the human waist on a smooth motion. The lower the value of the settling time t_s , the more accurate the following.

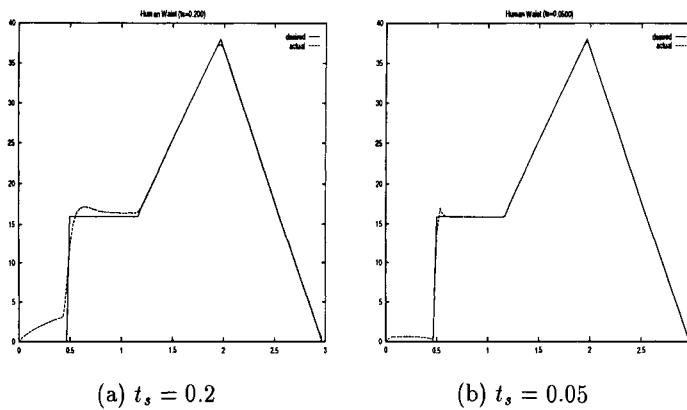


Figure 15.8 Trajectory following for the human waist on a discontinuous motion. By varying t_s one gets different response time of the joint to changes in the desired state.

Finally, in Fig. 15.10 we show five steps of an interactive crawler crane simulation. The user, using the panel shown in Fig. 15.11, controls the motion of the crane (left and right crawler velocity, cabin rotation, lower and upper boom angles) in order to lift and place the pillar on top of another pillar. The motion of the heavy (10 tons) pillar attached to the end of the crane's cable is generated in real time, thus giving the user realistic visual feedback of his actions. The crane is modeled as an articulated figure with three joints (cabin, lower boom, upper boom). Furthermore, the cable is connected to the end of the upper boom with a spherical joint and the load is connected to the other end of the cable with another spherical joint. A translational joint is placed at the top of the cable segment to allow the user to lift/lower the load. Realistic

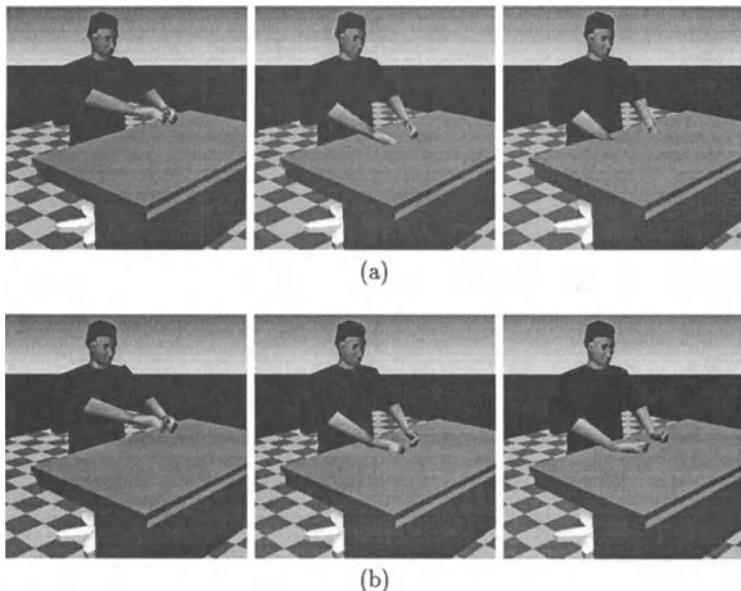


Figure 15.9 Following a physically incorrect motion trajectory: (a) Pure kinematics allows the motion, whereas (b) Dynamic simulation prevents inter-penetration between the arms and the desk.

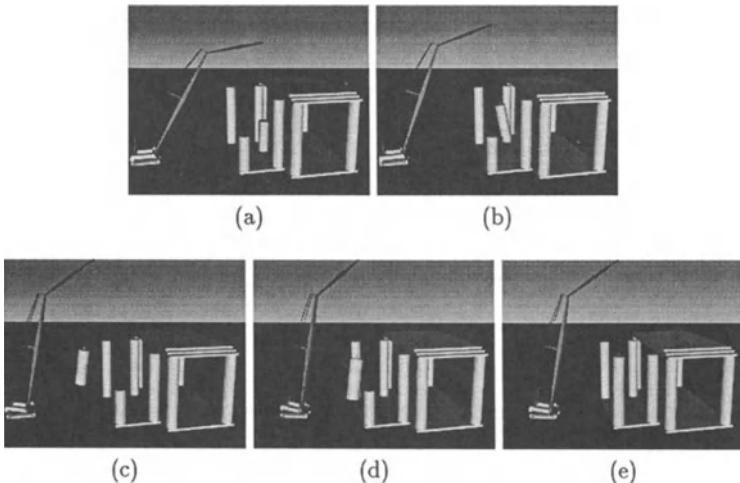


Figure 15.10 Interactive crawler crane simulation: (a-e) Five snapshots from a simulation where based on user defined crane motions, the crane lifts and places a 10 ton pillar on top of another pillar.

tic mass values are assigned to each segment of the crane. A dynamic adaptive

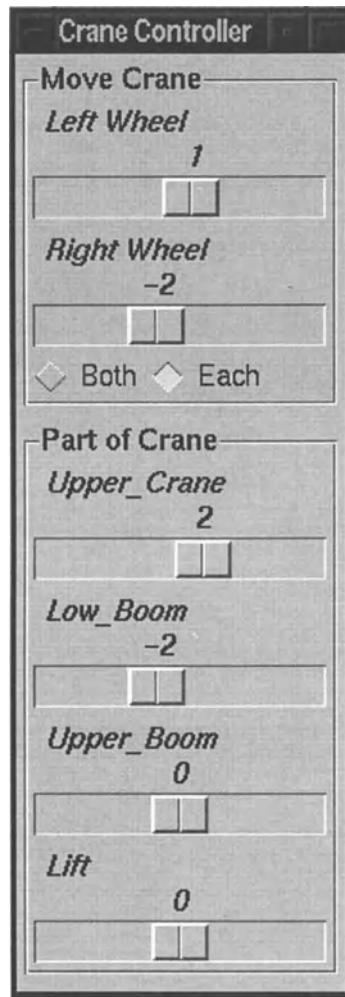


Figure 15.11 Interactive crawler crane simulation: User control panel.

controller regulates the torque (or force) at each joint of the model and provides the user with control over the crane's function.

Based on the above very encouraging results, we plan to extend our system to allow for more complex dynamic simulations and enhanced user interaction capabilities.

ANIMATING LIQUIDS

So far all the PBM methods presented have dealt with modeling phenomena involving rigid and viscoelastic single/multiple objects. In this chapter we present a PBM method for modeling a different class of physical phenomena and in particular liquids. In addition, we also demonstrate how to model simple interactions between buoyant rigid objects and a liquid.

Some of the most breathtaking animations in recent years have been generated by modeling the interaction between light and water. Effects such as caustic shading, reflection, refraction, and internal scattering have been addressed in some detail, with realistic results [203, 280, 288]. One characteristic of that work however, has been that the motion of the water surface is approximated by a non physics-based function. Suggested methods have included parametric functions [90] and sinusoidal phase functions [173, 213]. Two exceptions to this are the papers by Kass and Miller, and Chen and Lobo. Kass and Miller use a fast approximation to the two-dimensional shallow water equations to simulate surface waves in water of varying depth [138]. Their model allows for the reflection and refraction of waves, and takes account of mass transport, but it does not address the full range of three-dimensional motion found in a liquid. Such motion includes rotational and pressure based effects responsible for the much of a fluid's characteristic behavior. They also cannot easily incorporate dynamic objects or buoyant effects into the model, because the velocity of the fluid is known only on the surface, and internal pressure is not calculated at all. Chen and Lobo go further towards a physics-based fluid methodology by solving a simplified form of the Navier-Stokes equations in two dimensions [51]. However, they assume that the fluid has zero depth, and calculate the elevation of the surface solely from the instantaneous pressure. This allows them to perform some interaction between moving objects and the flow field, but restricts the class of problems that can be solved using the method. Notably, obstacle geometry must be two-dimensional, and although the surface height is varied for animation, they treat the fluid as being completely flat during the calculation. Therefore, convective wave effects, mass transport, and submerged obstacles are not covered by their technique.

Comprehensive models of fluid motion do exist, and there are a variety of tools for solving them in the field of Computational Fluid Dynamics (CFD). These methods generally involve direct simulation techniques to get accurate fluid motion. Unfortunately, in any direct simulation technique the temporal resolution is strongly coupled to the spatial resolution. Thus, if the spatial resolution doubles, the temporal resolution must also be doubled so that the solution does not move more than one spatial sample per time step. This gives running times proportional to the fourth power of the resolution, so most of these techniques will scale poorly. Furthermore, an animator needs a fairly clear understanding of the system of equations being solved so that he or she can set initial and boundary conditions to get the desired results. An ideal fluid simulator for graphics applications would apply the correct conditions automatically based on the underlying geometry. CFD methods also resist external control, making it difficult to force a particular motion from a fluid, unless it is a natural consequence of the system. These restrictions are an inherent part of the fluid modeling problem. The question arises whether it is possible to accurately model realistic fluid motion while keeping within acceptable efficiency bounds for Computer Graphics.

In this chapter we present a solution to the Navier-Stokes equations for modeling liquid motion, that satisfies many of an animator's needs. Realism is provided through a finite difference approximation to the incompressible Navier-Stokes equations. This gives rise to a complete pressure and velocity profile of the simulated environment. This profile is then used to determine the behavior of free surfaces, and is loosely coupled to the Lagrange equations of motion to include buoyant rigid objects into a scene. The range of behaviors accounted for include wave effects such as refraction, reflection and diffraction, together with rotational motion such as eddies and vorticity. Furthermore, velocity and pressure are strongly coupled within the model. This means that even the simplest animation exhibits subtle realistic behavior not available using previous computer-graphics fluid models.

Usability has also been a strong motivation for this work. The Navier-Stokes equations are solved over a coarse, rectangular mesh containing an arbitrary distribution of submerged or semi-submerged obstacles. Boundary conditions for the mesh are generated automatically by constraining the free variables at an obstacle-fluid or air-fluid boundary. This low resolution calculation together with homogeneous boundary conditions leads to a relatively efficient determination of fluid velocity and internal pressure. Detail is achieved by using the velocity field to concentrate attention on regions of interest, i.e., the fluid surface. The surface is represented as either a chain of massless marker particles, or a height field. The markers are carried around the mesh by convection, and can have arbitrary connectivity, accounting for multiple colliding surfaces in a scene.

Consideration is also given to controlling the overall behavior of the fluid. Liquid sources or sinks (known as inflow and outflow boundaries) can be included anywhere in the environment. They allow liquid to flow (or be forced) into

a scene, or flow out at a natural rate. A time dependent pressure field may also be applied to the fluid surface. Thus, the effects of a strong wind can be simulated and initial waves be driven realistically. The output from the system is a polygonal surface or height field, both of which can be rendered using many of the techniques presented by researchers in recent years [90, 138, 203, 213, 288].

This chapter is organized as follows: We begin by describing the Navier-Stokes equations, which can be used to accurately determine the motion of a viscous liquid in three dimensions. We show how these equations can be solved to give the complete pressure and velocity profile of a flow, and that if care is taken during discretization of the scene, this can be done efficiently and automatically for an arbitrary environment. Section 16.3 then focuses attention on the liquid free surface. The dynamic position of multiple surfaces can be delineated without restriction by the convection of massless particles. For the special, but common case of a flow without overturning, we derive a height field equation that couples velocity and surface elevation to the Navier-Stokes equations to generate a mesh suitable for spline-based rendering. A method for loosely coupling the Lagrange equations of motion to the flow pressure profile is described in Section 16.4. This method is used to include buoyant rigid objects into a scene. The complete algorithm for our technique is given in Section 16.5. Following this, (Section 16.6) two methods for controlling the fluid motion by constraining velocity and pressure boundary conditions are presented. Examples are given to simulate different speeds of flow and wind driven waves. We conclude with a description of several example animations that have been made using our system, together with a discussion of the procedure followed in each case.

16.1 NAVIER-STOKES EQUATIONS

The motion of a fluid at any point within a flow is completely described by a set of non-linear equations known as the momentum or Navier-Stokes equations. In three dimensions, for an incompressible fluid such as water, these equations can be written as

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z} &= -\frac{\partial p}{\partial x} + g_x + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) \\ \frac{\partial v}{\partial t} + \frac{\partial vu}{\partial x} + \frac{\partial v^2}{\partial y} + \frac{\partial vw}{\partial z} &= -\frac{\partial p}{\partial y} + g_y + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right) \\ \frac{\partial w}{\partial t} + \frac{\partial uw}{\partial x} + \frac{\partial vw}{\partial y} + \frac{\partial w^2}{\partial z} &= -\frac{\partial p}{\partial z} + g_z + \nu \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right), \end{aligned} \quad (16.1)$$

where u , v , w are velocities in the x , y , z directions respectively, p is the local pressure, g , gravity, and ν is the kinematic viscosity of the fluid. Although they may seem daunting at first sight, these equations have very humble origins. They are derived from Newton's Second Law which states that momentum is always conserved. The Navier-Stokes equations simply account for all momentum exchange possibilities within a fluid. Specifically, the terms on the left

hand side of the equations account for changes in velocity due to local fluid acceleration and convection. The right hand terms take account of acceleration due to the force of gravity (or any body force \mathbf{g}), acceleration due to the local pressure gradient, ∇p , and drag due to the kinematic viscosity, ν , or thickness of the fluid. Together with appropriate boundary conditions and the constraint that not only momentum, but also mass should be conserved (see Section 16.2.1), the Navier-Stokes equations can be used to accurately simulate fluid phenomena.

16.2 SOLVING THE NAVIER-STOKES EQUATIONS

Despite the complexity of a system of differential equations such as (16.1), it is possible to solve it in an intuitive way, using standard analysis tools [87]. The first step is to discretize both the equations and the environment that we want to model. There are a number of ways to do this, but it is important to keep four things in mind:

- In a typical graphics application involving liquids, there are likely to be numerous boundaries between the liquid and other objects, and between the liquid and the surrounding medium. Computation cost can be minimized if such interfaces are homogeneously incorporated into the model instead of being treated as special cases.
- Generality is everything. Users of the system need to be able to specify environment geometry quickly, and without referring to the underlying equations for the correct boundary conditions.
- It must be possible to apply some external control to the system so that the animator can accurately specify how the liquid will behave.
- The range of motion that can be animated using the technique should include the set of effects available with existing computer-graphics methods, and extend it by adding new, interesting, and useful behavior.

With some thought, a good discretization that provides a solution to the first of these constraints also provides solutions to the other three. In the following sections we present a numerical solution to the Navier-Stokes equations. The technique combines a low resolution 3D calculation to determine velocity and pressure fields within the liquid, with a height field equation that is used to precisely track the position of a free surface. At all times during the computation, boundary conditions due to solid obstacles and the fluid surface are homogeneous, and their application is transparent to the user.

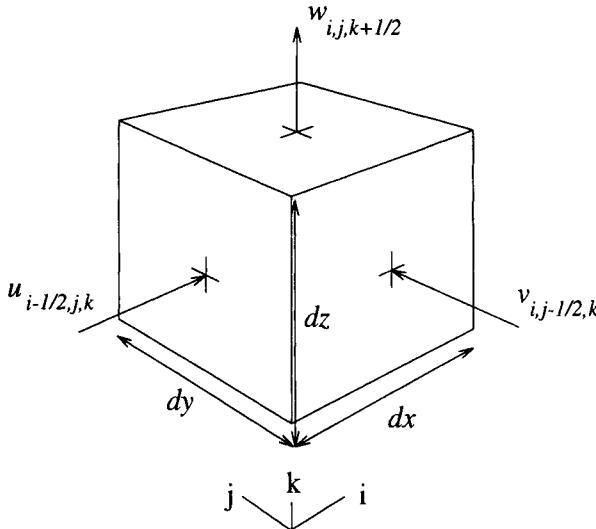


Figure 16.1 Location of staggered velocity components on a typical cell (i, j, k) .

16.2.1 Discretization

We solve (16.1) across the entire environment. Solid obstacles and the atmosphere are treated as fluid, but with special properties that remain constrained throughout the calculation. The computation domain is first divided into a fixed rectangular grid aligned with a Cartesian coordinate system. The u , v , and w velocities are defined at the centers of each face of a cell and referenced locally (see Fig. 16.1) while pressure, p , is defined in the center of each cell. Note from the figure that $v_{i,j-1/2,k} \equiv v_{i,(j-1)+1/2,k}$. At the start of the calculation the contents of each cell are determined. A cell may either contain a *Solid* obstacle, be *Full* of fluid, be a *Surface* cell on the boundary between the liquid and surrounding medium, or be *Empty*. In all four cases, the velocity and pressure fields are defined everywhere.

This discretization leads to an explicit finite difference approximation of (16.1) in the form [109]

$$\begin{aligned}
 \tilde{u}_{i+1/2,j,k} = & u_{i+1/2,j,k} + \delta t \{ (1/\delta x)[(u_{i,j,k})^2 - (u_{i+1,j,k})^2] \\
 & + (1/\delta y)[(uv)_{i+1/2,j-1/2,k} - (uv)_{i+1/2,j+1/2,k}] \\
 & + (1/\delta z)[(uw)_{i+1/2,j,k-1/2} - (uw)_{i+1/2,j,k+1/2}] + g_x \\
 & + (1/\delta x)(p_{i,j,k} - p_{i+1,j,k}) + (\nu/\delta x^2)(u_{i+3/2,j,k} \\
 & - 2u_{i+1/2,j,k} + u_{i-1/2,j,k}) + (\nu/\delta y^2)(u_{i+1/2,j+1,k} \\
 & - 2u_{i+1/2,j,k} + u_{i+1/2,j-1,k}) + (\nu/\delta z^2)(u_{i+1/2,j,k+1} \\
 & - 2u_{i+1/2,j,k} + u_{i+1/2,j,k-1}) \},
 \end{aligned} \tag{16.2}$$

for each velocity component u, v , and w of cell i, j, k . Although this system of equations is complex, the solution process is straightforward. In choosing an explicit formulation, we have made a tradeoff between ease of implementation and the maximum stable timestep for numerical integration. However, by approximating the environment using a relatively coarse grid (see Section 16.2.2), the condition that [87]

$$1 > \max[u \frac{\delta t}{\delta x}, v \frac{\delta t}{\delta y}, w \frac{\delta t}{\delta z}] \quad (16.3)$$

be satisfied everywhere for the solution to be stable, can be met with a timestep that is still sufficiently large to animate motion efficiently (see Section 16.7).

To move the solution forward, velocities and pressures from the previous iteration are taken directly from individual cells and plugged into (16.2) to give the new velocities for the current iteration ($\tilde{u}, \tilde{v}, \tilde{w}$). In some cases, velocities are required that do not lie on cell faces, in which case they are averaged over the nearest available values, e.g., $u_{i,j,k} = \frac{1}{2}(u_{i+1/2,j,k} + u_{i-1/2,j,k})$, and the square of a quantity, e.g., u^2 at (i, j, k) , is the square of the average, $(u_{i,j,k})^2$.

The new velocities are labeled with a tilde because the direct application of (16.2) does not ensure physical correctness. Due to the rectangular discretization of the environment, each individual cell may not explicitly satisfy the criteria that mass be conserved and that the fluid is incompressible. Also, the new pressure field needs to be determined. These constraints are satisfied simultaneously by solving the mass conservation, or continuity equation [87],

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \quad (16.4)$$

which essentially says that the net fluid flow into or out of a cell is zero.

Consider a cell i, j, k . The divergence of fluid within the cell, or “missing mass”, is given by [201]

$$\begin{aligned} D_{i,j,k} &= -((1/\delta x)(u_{i+1/2,j,k} - u_{i-1/2,j,k}) \\ &\quad + (1/\delta y)(v_{i,j+1/2,k} - v_{i,j-1/2,k})) \\ &\quad + (1/\delta z)(w_{i,j,k+1/2} - w_{i,j,k-1/2})). \end{aligned} \quad (16.5)$$

Notice that this is a finite difference approximation to the continuity equation. A positive D therefore, represents an influx of fluid, and in the real world would correspond to an increase in cell pressure and subsequent increase in fluid outflow from the cell. Similarly, a negative D lowers internal pressure and increases inflow from neighboring cells. Thus if the change in cell pressure is scaled according to the divergence in the cell, and the face cell velocities are adjusted according to the change in pressure, the cell can be made to satisfy (16.4). The change in pressure for a cell is

$$\delta p = \beta D, \quad (16.6)$$

where β is given by [201]

$$\beta = \beta_0 / 2\delta t (\frac{1}{\delta x^2} + \frac{1}{\delta y^2} + \frac{1}{\delta z^2}), \quad (16.7)$$

and β_0 is a relaxation coefficient within the range [1,2]. The cell face velocities are then updated according to δp such that

$$\begin{aligned} u_{i+1/2,j,k} &= u_{i+1/2,j,k} + (\delta t / \delta x) \delta p, \\ u_{i-1/2,j,k} &= u_{i-1/2,j,k} - (\delta t / \delta x) \delta p, \\ v_{i,j+1/2,k} &= v_{i,j+1/2,k} + (\delta t / \delta y) \delta p, \\ v_{i,j-1/2,k} &= v_{i,j-1/2,k} - (\delta t / \delta y) \delta p, \\ w_{i,j,k+1/2} &= w_{i,j,k+1/2} + (\delta t / \delta z) \delta p, \\ w_{i,j,k-1/2} &= w_{i,j,k-1/2} - (\delta t / \delta z) \delta p, \end{aligned} \quad (16.8)$$

and the cell pressure is updated according to

$$\tilde{p}_{i,j,k} = p_{i,j,k} + \delta p. \quad (16.9)$$

Use of the above equations satisfies (16.4) for a single cell, but neighboring cells may now have a non-zero divergence [201]. In order for the whole mesh to simultaneously satisfy (16.4), the pressure and velocities are first adjusted using (16.5)–(16.8) for every cell in the grid. This procedure is then repeated until all cells in the flow field have a divergence less than some prescribed small ϵ . With a β_0 of 1.7 and ϵ of 0.0001, the examples shown converged in 3–6 sweeps on average. Once convergence is achieved, the fluid is considered to be locally incompressible and the velocity and pressure fields are complete for buoyant object inclusion and the start of the next cycle.

16.2.2 Boundary Conditions

The boundary conditions for our model are set automatically once the contents of each cell in the mesh have been determined. They are also homogeneous. That means that once they have been set, the Navier-Stokes equations can be applied blindly without determining exactly where surfaces or obstacles lie. This makes for cheap computation because boundary conditions need only be checked once at the beginning of an iteration, rather than for the velocity component calculation of each cell. A boundary is an interface between the fluid and a solid obstacle, or between the fluid and atmosphere, or a point at which fluid flows into or out of the system. In all cases, generalizing assumptions about the shape of static obstacles, and the position of free surfaces can greatly reduce the amount of work that we have to do, without compromising accuracy or realism.

Stationary Obstacles

Consider Fig. 16.2, which shows an obstacle and a free surface. We assume that the walls of an obstacle are always coincident with the face of a computational

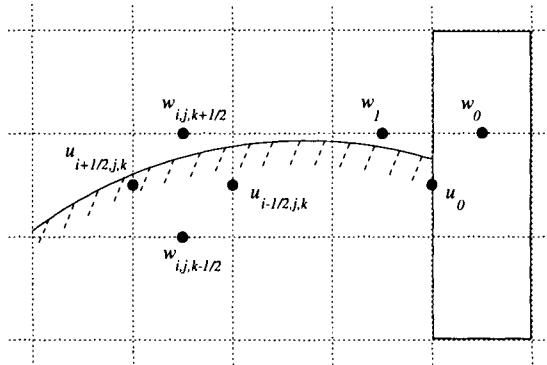


Figure 16.2 Setting boundary conditions on the free surface and across obstacle boundaries.

cell. It then becomes a trivial process to set correct solid obstacle boundary conditions, that is, velocity and pressure for use in the finite difference expressions. For example, the component of fluid velocity normal to the face of a non-permeable obstacle is zero. Because obstacle and cell faces are coincident, the normal velocities are set directly ($u_0 = 0$ in the figure). In the case of a non-slip obstacle which exerts a drag on the fluid, the tangential velocity at the boundary is also zero. This is set indirectly by making the tangential cell face velocity inside the boundary cell equal and opposite to that outside in the fluid ($w_0 = -w_1$). Finally, the pressure in the boundary cell, which is also needed for the finite difference calculation, is set equal to the pressure in the adjacent fluid cell, preventing any acceleration across the boundary.

Another useful type of obstacle is a free-slip boundary. The treatment of pressure and velocity is the same as for a non-slip boundary except that the inner tangential velocity is set equal to that outside in the fluid ($w_0 = w_1$). A free-slip boundary can be thought of as a plane of symmetry for motion tangential to it, thus it provides a convenient way to bound a flow field.¹

Inflow and Outflow

Fluid can easily flow into or out of the system by virtue of inflow or outflow boundary cells. For inflow, the required input velocity is set on the cell faces and held fixed throughout the calculation. In the case of an outflow boundary, velocities are initially set equal to the tentative velocity field in adjacent fluid cells and then allowed to relax without constraint during the pressure iteration step. This ensures that fluid can flow freely out of the system without causing any upstream artifacts.

¹All of the examples shown are bounded by a layer of free-slip boundary cells so that sides of a scene which are *open* do not effect the flow.

Free Surface

Boundary conditions also need to be set on the free surface. When (16.2) is applied to a surface cell, velocities and pressures are needed from adjacent empty cells. We assume that for most applications, if the wavelength of any disturbance is longer than a few inches, forces due to surface tension will be negligible. We then relax the constraint that we need to know exactly where in a cell the surface lies. Thus, if any part of a free surface passes through a cell, that cell is labeled as a *Surface* cell, and the equation of continuity (16.4) is used to set boundary velocities. Consider a two dimensional surface cell which is surrounded on three sides by cells containing fluid. The velocity on the remaining surface side is set so that the divergence D of the fluid in the cell is zero. So referring to Fig. 16.2,

$$w_{i,j,k+1/2} = w_{i,j,k-1/2} - (\delta z / \delta x)(u_{i+1/2,j,k} - u_{i-1/2,j,k}). \quad (16.10)$$

If the cell had two sides which face an empty cell, we require that $\partial u / \partial x$ and $\partial w / \partial z$ both vanish separately, that is that each open side velocity equals the velocity of the side of the cell opposite it. This also satisfies (16.4). Finally, for the case in which three sides are open, the side opposite the fluid carries the velocity of that side, while the remaining two sides follow freely the effects of the body force and do not otherwise change. A three dimensional surface cell has velocity components set in an analogous fashion, leading to 64 distinct *Empty-Fluid* configurations. The pressure in a surface cell is set to the applied atmospheric pressure or forcing pressure function (see Section 16.6.2).

16.3 TRACKING FLUID POSITION

We have described a method for solving the full Navier-Stokes equations over a finite difference mesh. From the mesh we want to generate a smooth and accurate representation of the actual fluid surface position. We also want to track the motion of such a surface over time, so that we can adjust the contents of the mesh accordingly (i.e., *Full*, *Surface*, or *Empty*). Finally, to avoid aliasing, the resolution of the surface should not be restricted by the coarse resolution of the mesh. With these goals in mind three methods of surface identification have been developed, each of which is useful for a particular class of liquid phenomena.

16.3.1 Marker Particles

The simplest and most functional way to track fluid position in 2D is to convect massless marker particles with local fluid velocity. In this way particles are continuously introduced at inflow boundaries and removed if they cross an outflow boundary, and can splash and flow freely. A particle's new position is found using an area weighting interpolation over the four nearest cell velocities (See Fig. 16.3) and multiplying the resultant velocity by the current timestep. The finite difference mesh is then labeled as follows:

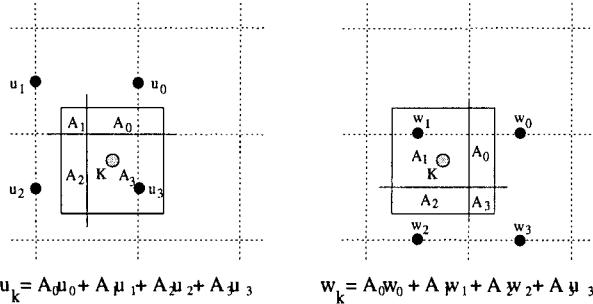


Figure 16.3 Area weighting interpolation scheme for determining local fluid velocity for a marker k .

- A cell containing no particles is *Empty*.
- A cell containing at least one particle that is adjacent to an *Empty* cell is a *Surface* cell.
- A cell containing at least one particle that is not a *Surface* cell is a *Full* cell.

The use of marker particles can highlight the full range of internal fluid motion such as rotation and splashing at a greater resolution than the finite difference mesh. It is important to note that the particles do not represent a mass of fluid. They are used to define the position of the surface only, and have no effect on the calculation. Frames from two dimensional animations using marker particles are shown in Fig. 16.4. Fig. 16.4(b) shows that an initial pulse of water has struck the sides of a concrete tank and has been projected up into the air. This jet eventually overturns and crashes back down into the growing pool. Marker particles are ideal for animating violent phenomena such as overturning waves because they define the position of the fluid exactly, regardless of how complex the surface has become. Fig. 16.4 is discussed in more detail in Section 16.7.

16.3.2 Free Surface Particles

Marker particles can also be used to precisely delineate any free-surfaces in a scene. Instead of appearing within every cell containing fluid, a grid of markers is placed along the boundaries between fluid and obstacles or air. This grid is convected with local velocity as before. However, the number, distribution, and connectivity of particles are allowed to change dynamically as the position of the surface changes. The rules for removing and adding particles are simple. If two particles become too close together, delete both of them and connect their neighbors. If two particles become too far apart, insert a new particle on the link between them. This ensures that the surface always remains continuous and that colliding surfaces are smoothly connected. In two dimensions

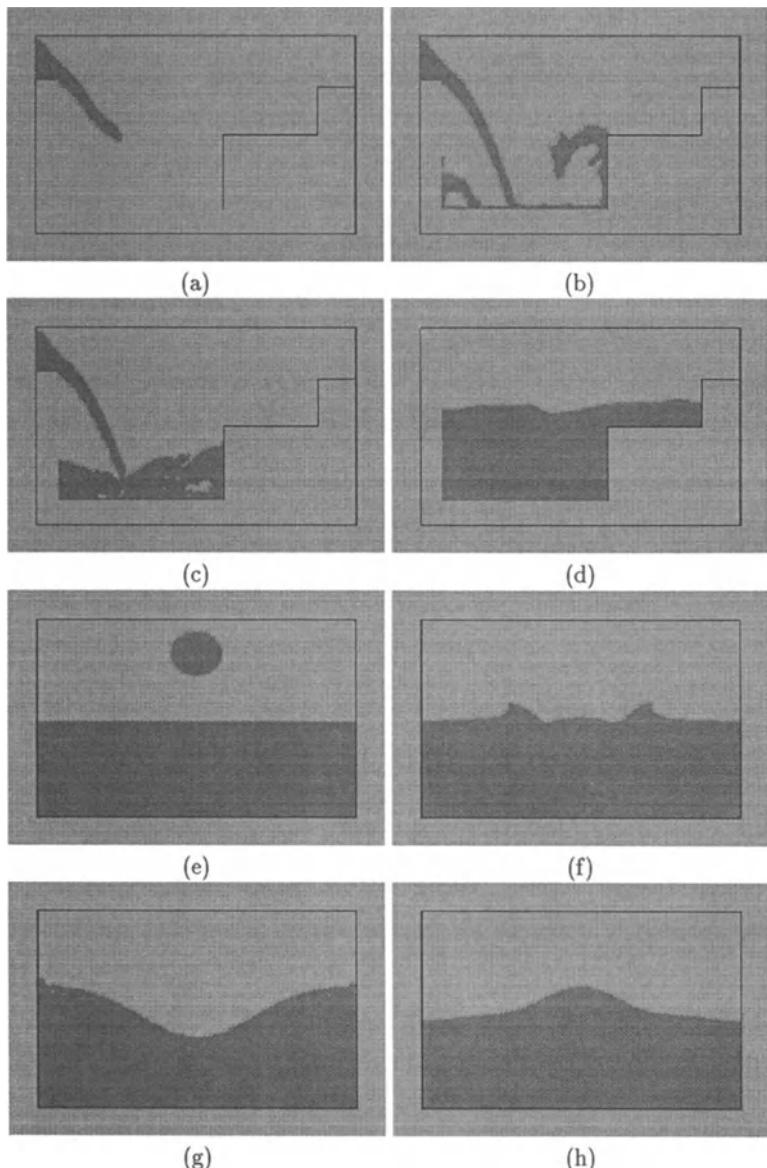


Figure 16.4 Frames from two dimensional animations making use of marker particles. A jet of water splashes into a concrete tank (a-d). A drop of water splashes into a shallow pool (e-h).

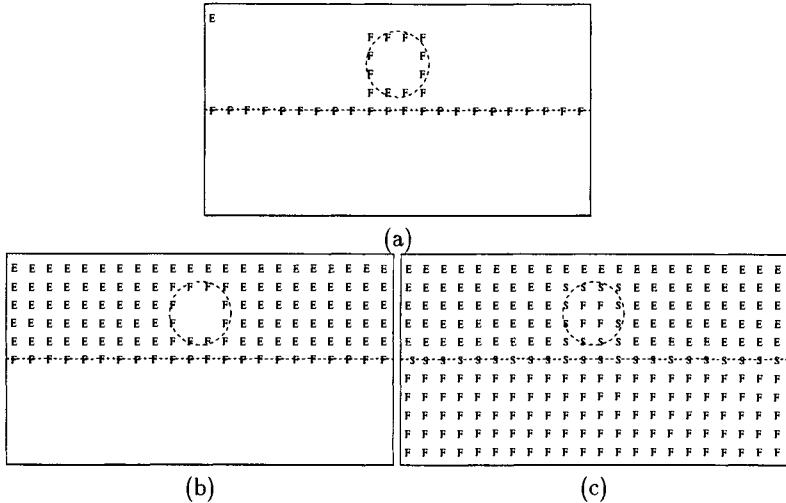


Figure 16.5 Region growing used to identify cell contents when using free surface particles. Initial state showing *Surface* cells S (a). After growing *Empty* cells E (b). Final state after growing *Full* cells F (c).

this method is particularly useful because it is fast and can easily account for multiple surfaces.

Cell configuration in 2D for each cycle is determined using a region growing algorithm. Firstly, all the cells that contain a surface marker are set to *Full* and all other cells are flagged as *Unknown* (See Fig. 16.5(a)). An *Empty* region is then grown from a cell in the mesh, usually the corner above the origin, which is known to be empty of particles at all times during the computation. Growing advances from this starting cell using a 4-connected search. In this way, any *Unknown* cells that are adjacent to an existing *Empty* cell are also set to *Empty*. When there are no more adjacent *Empty*-*Unknown* pairs (Fig. 16.5(b)), a second region is grown by setting *Unknown* cells that are adjacent to a *Full* cell to *Full*. This process is repeated, alternating between *Empty* and *Full* regions, always growing until a boundary or *Full* cell is hit. Finally, the cells that contain the original surface markers are set to *Surface* (Fig. 16.5(c)).

16.3.3 Height Field

Liquid in the real world often has a surface that is single valued. Examples of this are puddles, rivers, or the ocean (as long as there are no overturning waves). For such cases the position of the surface can be calculated without using marker particles because we no longer need to track the complex geometry caused by overturning. We define the surface height along the y axis, at the center of each vertical column of cells in the three-dimensional mesh. The change in local surface elevation at each timestep is determined by the local

fluid velocity, that is, by the vertical component of the fluid motion plus the horizontal convection of the surface elevation from adjacent cell columns,

$$\frac{\partial h}{\partial t} = w - u\left(\frac{\partial h}{\partial x}\right) - v\left(\frac{\partial h}{\partial y}\right), \quad (16.11)$$

where h is the surface height. This equation can be approximated by a finite difference expression [201]

$$\begin{aligned} h_{i,j}^{t+\delta t} = & h_{i,j}^t + \delta t \{ \bar{w}_{i,j,k}^{t+\delta t} \\ & + \frac{(h_{i-1,j}^t - h_{i+1,j}^t)}{4\delta x} (u_{i+1/2,j,k}^{t+\delta t} + u_{i-1/2,j,k}^{t+\delta t}) \\ & + \frac{(h_{i,j-1}^t - h_{i,j+1}^t)}{4\delta y} (v_{i,j+1/2,k}^{t+\delta t} + v_{i,j-1/2,k}^{t+\delta t}) \}. \end{aligned} \quad (16.12)$$

This expression is used to update the position of the height field once the velocity and pressure fields have been calculated. It is important to note that despite superficial similarities to the method used by Kass *et al.* in [138], the height field equation is very different. Here, surface elevation is driven by the underlying fluid velocity. Therefore, velocity or pressure disturbances anywhere in the fluid volume can affect the surface (see Examples). Cell configuration for the height field approach is trivial. Cells crossed by the height field are *Surface* cells while those above it are *Empty*, and those below it are *Full*.

For dramatic effects such as crashing waves or splashing, the height field can be combined with the marker particles. Whenever the vertical velocity of the surface is greater than some positive threshold, a set of particles are introduced just below the surface and the local fluid velocity is used to set their initial velocity. The particles are then removed from the Navier-Stokes calculation and affected only by gravity, wind, and air resistance. For more realism there is a small probability that airborne particles will separate over time, giving the appearance of dispersing spray. Foam is introduced whenever spray collides with the height field. Foam has a random life span proportional to the size of its parent spray particle. There is interesting discussion of topics related to the use of particle systems for fluid animation in [90, 103, 193].

16.4 BUOYANCY

Rigid dynamic objects can be included in a scene using the velocities and pressures calculated using the Navier-Stokes equations. Specifically, we assume that each rigid object is discretized and consists of a set of nodes n_i . For each model surface node n_i which is within the fluid, the force acting on this node is calculated based on the following formula

$$\mathbf{f}_{n_i} = -\nabla p_i dV_i + m_i \mathbf{g}, \quad (16.13)$$

where dV_i is a volume associated with the submerged node of the object and ∇p_i is the gradient vector of the pressure. Each component of ∇p_i is computed in discrete form as

$$(\nabla p_i)_{x_j} = \frac{p_{n_i} - p_{n_i,x_j}}{\delta x_j}, \quad j = 1, 2, 3, \quad (16.14)$$

where p_{n_i} is the pressure in the cell containing n_i , and p_{n_i,x_j} is the pressure in the previous cell in the x_j direction. Also, g is the gravitational acceleration, and m_i is the nodal mass assuming lumped masses. The total force on the object due to the fluid motion and gravity is given in discrete form by

$$\mathbf{f}_{fluid} = \sum_i \mathbf{f}_{n_i}. \quad (16.15)$$

Based on the total force acting on each node, we compute the generalized external forces \mathbf{f}_q (total force and torque acting on the object) and we compute its motion based on the Lagrange equations of motion (see also Chapter 3)

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} = \mathbf{f}_q + \mathbf{g}_q, \quad (16.16)$$

where \mathbf{M} and \mathbf{D} are the object's generalized mass and damping matrices, \mathbf{q} are the model translational and rotational degrees of freedom, and \mathbf{g}_q are the generalized coriolis and centrifugal forces. The mass matrix, \mathbf{M} , is derived directly from the object in question [178], and is unaffected by the fluid model. The damping matrix, \mathbf{D} , also has the same form as in Chapter 3, but with the damping coefficients adjusted proportional to the relative velocity between a node, n_i , and the local fluid.

In order to handle collisions of the floating objects with static obstacles, we also apply the techniques developed in [178, 180] for collision detection and collision force computation.

The floating objects that we used in the examples are small compared to the mesh size and therefore it is possible to make the simplifying assumption that they do not effect the water flow. Thus, they act like large marker particles moving and rotating according to local forces. For the objects to influence the motion of the fluid, more sophisticated techniques need to be employed.

16.5 SUMMARY OF THE NAVIER-STOKES ALGORITHM

The complete algorithm for solving the Navier-Stokes equations and tracking the fluid surface can be summarized in the following steps;

1. Define obstacles and starting fluid configuration, and place dynamic objects.

2. Set initial pressure and velocity conditions.
3. Determine cell contents depending on the method used to track the surface.
4. Set up boundary conditions for the free surface and obstacle cells.
5. Compute $\tilde{u}, \tilde{v}, \tilde{w}$ for all *Full* cells.
6. Perform the pressure iteration for all *Full* cells.
7. Recalculate boundary velocities for *Surface* cells.
8. Update the position of the surface and objects.
9. Go to step 3.

16.6 CONTROL

An important part of the animation process is specifying how objects in a scene will move. Doing this for a fluid surface is difficult because the governing equations (16.1) are strongly coupled and non-linear. Large scale behavior of the system can be controlled by altering various constants such as gravity and viscosity, but it is difficult to specify a motion then solve backwards to find the correct boundary conditions to cause it. However, there are two places in our algorithm where coercion can be applied to the fluid. These can easily be exploited to yield effective methods for controlling the fluid surface.

16.6.1 Inflow and outflow velocities

A time dependent function can be used to determine the rate at which fluid is pumped into a scene or the rate at which it is allowed to exit, producing a variety of effects. For example, a broken dam initially generates a high input rate, but tails off exponentially as the water level drops. Also, for animating a river scene, a varying inflow and outflow rate will simulate different classes (speed, turbulence) of water flow without requiring any changes to the environment model.

16.6.2 Surface pressure history

Perhaps the most natural way to specify surface behavior is to model nature. As wind blows across a liquid surface, small, low pressure vortices induce a local change in surface elevation. This in turn, disturbs the airflow over the surface, changing the pressure. Gravity then provides a restoration force for the initial perturbation which results in oscillation. Thus, over time, the process is amplified and a wind driven wave is born. A similar effect can be achieved in a shorter time by applying a forcing pressure history to the free surface during the

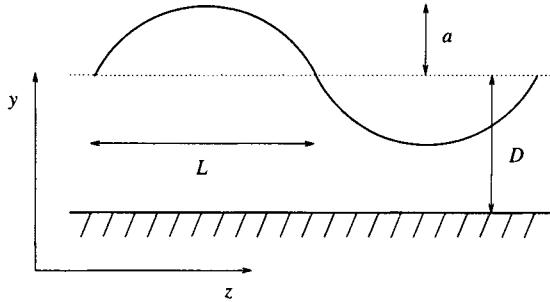


Figure 16.6 Deriving constants for an applied pressure function.

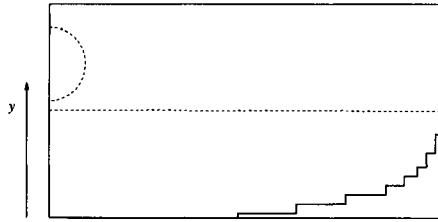


Figure 16.7 Starting configuration for 2D marble soup animation.

Navier-Stokes computation. This may be constant, time dependent, or depend on the present height of the surface. For example, in two dimensions, interesting waves can be developed using the forcing function

$$p_{\text{applied}}(z) = \frac{A + B \cos(Cz - \omega t)}{\delta t}, \quad (16.17)$$

where p_{applied} is the pressure within a *Surface* cell, $A/\delta t$ is the mean pressure, and B and C are constants derived from the desired wave motion. From Fig. 16.6, if $2L$ is the wavelength of the oscillation and D is the mean depth of the fluid, then,

$$B = a \sqrt{-\frac{gD}{C}}, \quad (16.18)$$

and

$$C = \frac{\pi}{L}, \quad (16.19)$$

where a is the wave amplitude, and g is gravity. Such a function is used to set the applied pressure boundary condition on the free surface (See section 16.2).

16.7 EXAMPLES

We present a number of examples to show different aspects of the system described. Running times are given for a Silicon Graphics Crimson R4000. They

do not vary linearly with the size of each problem because other factors, such as the total number of *Full* cells present, or the speed of the flow, make a larger contribution to the amount of CPU time required.

The first example (Fig. 16.4(a-d)), is a two dimensional animation of a water jet splashing into a concrete tank. The water motion was calculated over a 30x40 grid of cells, and marker particles were used to delineate fluid position. Two input rates were specified; water inflow and particle inflow. The jet had a velocity of 0.8 ms^{-1} and new particles were introduced at the inflow boundary at a rate of 500 particles per second. It is important to note that the only overhead associated with the marker particles is the cost of moving and displaying them. A relatively sparse distribution of particles was used in this case to clearly show that the model can account for colliding surfaces, overturning waves, and arbitrary splashing. A later frame from the animation shows that after the jet is turned off the vortices in the tank slow down and the surface starts to settle (Fig. 16.4(d)). This animation ran for 4500 iterations in just over sixteen minutes. The same grid size (30x40) was used again to animate a splashing drop (Fig. 16.4(e-h)). Figs. 16.4(e) and (f) show the starting configuration of the drop and its initial impact with the surface. The waves caused by the collision travel out to the sides of the pool (Fig. 16.4 (g)), and are reflected back to give the characteristic fluid rebound at the epicenter of the splash. Particle density was set at 25 per cell. This animation ran for 2800 iterations in twelve minutes, slightly slower than the water jet example above because the average number of *Full* cells per iteration was higher.

If the scene geometry is rotationally symmetric, computationally cheap two dimensional calculations can be made using linked chains of markers. The resulting profiles can then be used to define a surface of revolution. While this method is not a physically accurate way of generating three dimensional motion, it does give some simple effects which are good enough for graphics applications. Fig. 16.9 shows two frames from an animation of a rigid marble dropping into a bowl of thick soup. The actual calculation was performed in two dimensions by setting the z axis resolution to 1. The curved side of the bowl was approximated as a series of steps, and a semicircular drop of liquid was aligned along the y axis (see Fig. 16.7) to represent the marble. The drop was given an initial velocity of -0.2 ms^{-1} , and the viscosity of the soup was set relatively high (0.003). Two chains of particles were used to represent the free surfaces in the scene, 50 for the drop and 150 for the soup. The calculation was run for two thousand iterations at a resolution of 15x30x1 taking twelve minutes. The scene was rendered directly from the positions of the markers. Each chain was used to define the profile of a surface of revolution, which was smoothed using a series of bicubic splines. Finally, the marble and other objects were added, and the whole scene rendered using Pixar's PhotoRealistic RenderMan [282]. The liquid surface was colored using a straightforward environment map, taking account of Fresnel's law [58] to calculate the fraction of light reflected toward the camera, or transmitted to the bottom of the bowl.

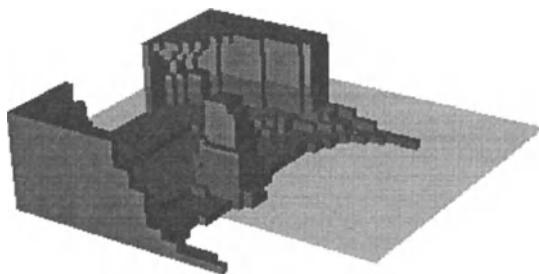


Figure 16.8 Calculation environment for the Moonlight Cove.

The soup example clearly shows some of the advantages of our model. The liquid drop for the calculation is the same size as the marble object, so after impact the mean surface level has risen correctly. Also, the coupling between pressure and velocity develops as a non-linear oscillation which continues long after the wave due to the collision has subsided. Previous computer graphics fluid models would have accounted for the surface wave, but not for the accompanying pressure wave which is responsible for most of the final motion.

The first full 3D example is an animation titled *Moonlight Cove* (Fig. 16.10). A 50x15x40 mesh was used to finely resolve the effect of two large ocean waves crashing into a shallow cove. Submerged rocks, and an irregular sea bottom, focus the waves into the center of the cove, causing a number of interesting features on the water surface. The wave becomes steeper as the water depth decreases, and eddies and pressure waves appear to the left of, and behind the initial obstacle (Fig. 16.10(b)).

Setting up the scene was straightforward and proceeded in two stages. First, a voxel based editor was used to define the initial distribution of rocks and water (Fig. 16.8). The last plane of cells opposite the cove were then designated as inflow cells, with inflow velocity defined as

$$u = u + a\omega^2 \cos\omega t, \quad (16.20)$$

where a was the desired wave amplitude and ω the desired wave frequency. The calculation was run for $2/\omega$ seconds, then the inflow cells were changed to outflow and water allowed to leave the system at its natural rate. This approach resulted in two full waves while allowing the added water volume to flow back out of the scene once the waves had been reflected. The animation took two and a half hours to complete and ran for 20,000 iterations.

RenderMan was also used to render this example. Two spline meshes were used; one generated from the surface height field, and another from the distribution

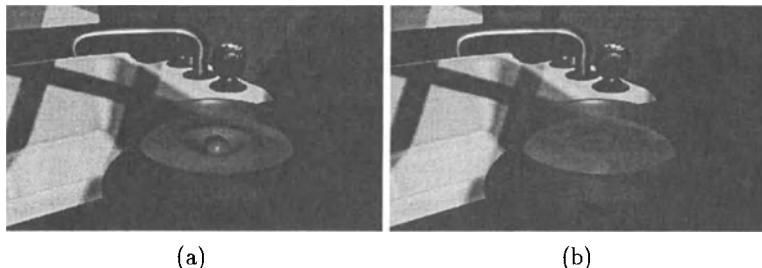


Figure 16.9 A marble dropping into a bowl of thick soup. Initial collision (a). Oscillation due to coupling between pressure and velocity (b).

of boundary cells. The water surface was rendered as a glass-like object with small disturbances generated using the long crested wave model suggested in [247]. Detail in the rocks was provided using a displacement map and suitable noise function on the spline surface.

A second 3D example is shown in Fig. 16.11. A pressure wave caused by an opening sluice gate is forced along a channel. The wave travels over a submerged pyramid, is reflected through ninety degrees, and finally crashes over a semi-submerged rock. The importance of calculating pressure and velocity throughout the whole volume is clear from the pictures. Fig. 16.11(a) shows the trailing wave that builds up behind the submerged obstacle as water flows around it. This trailing wave survives until the original wave crashes over the semi-submerged obstacle (Fig. 16.11(b)) and flows out of the system. The mean water level has also risen because of the inflow from the sluice gate. This example was calculated over a 40x12x40 grid, took an hour to compute, and ran for 8,000 iterations. For speed, standard Silicon Graphics hardware routines were used for rendering in this case.

The frames in Fig. 16.12 show screen shots from an animation involving buoyant objects. Water flows into a closed container carrying soda cans along with it. When the flow is turned off, the cans gather at the far corner of the container because the walls in this example were set as non-slip so the tangential fluid velocity is zero. This simulates the effect that objects tend to gather in stagnant parts of a flow. The water motion was precomputed in thirty minutes over a 30x10x20 grid. The soda cans were added later using an interactive editor which takes a precomputed velocity and pressure field, and calculates the forces on an object within the mesh. In this way, many different shapes and sizes of object can be experimented with, without having to re-do the fluid calculation.

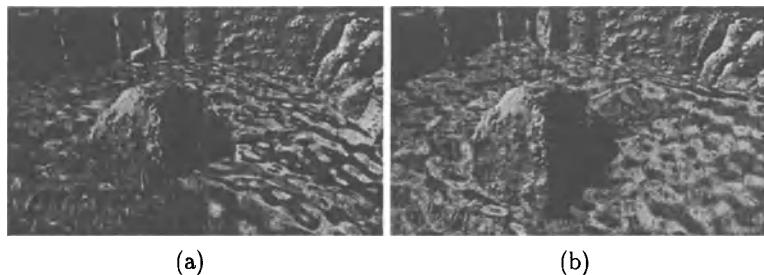


Figure 16.10 Moonlight Cove. Two ocean waves crash into a shallow cove. Pressure and velocity effects throughout the water volume manifest themselves at the surface (a,b).

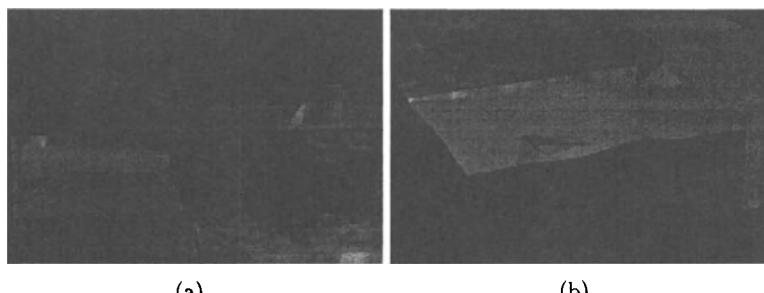


Figure 16.11 A pressure wave travels over a submerged obstacle (a), is reflected by ninety degrees and crashes over a semi-submerged rock (b).

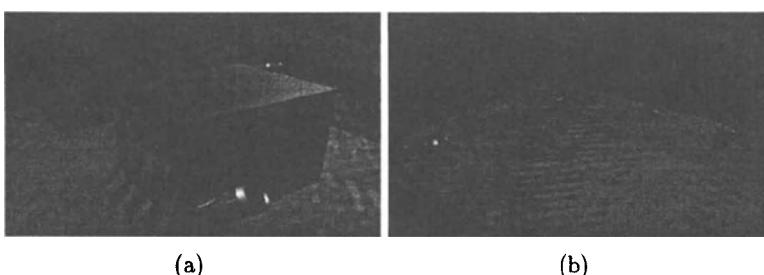


Figure 16.12 Dynamic objects. Soda cans are carried along with the incoming water, colliding with obstacles (a), and getting caught in local eddies (b).

CONCLUSIONS

In this book we have presented our PBM framework and have demonstrated its usefulness in shape and motion modeling, segmentation and estimation. We have demonstrated the application of our framework to a variety of problems in computer vision, graphics and medical imaging. The unique feature of our framework is that we have developed a theory upon which certain important aspects of the problems of segmentation, shape and motion estimation and modeling are treated in a unified and successful way. We strongly believe that an integrated approach to these problems will allow the future solution of more complex problems in computer vision, graphics and medical imaging.

However, there are many more difficult problems in the above areas that need to be addressed. Such problems include the discovery of new methods for segmenting arbitrary scenes, tracking multiple objects with significant occlusion, recognizing objects and behaviors, modeling the motions and behaviors of living organisms, modeling the anatomy and physiology of humans, internal organ shape and motion estimation, and illness diagnosis from various imaging modalities. In the future, we plan to significantly extend this framework in order to address more challenging problems in the above domains. In the following, we will elaborate briefly on some of our new directions.

Most of the image-based shape and motion recovery methods in computer vision, are based on the use of contours (occluding and/or internal) and stereo or multiple view information. However, the use of other cues such as optical flow and shape from shading, has not been integrated in these methods. These other cues are very important in order to increase the accuracy and reliability of the object shape and motion estimation. We have recently proposed a theory for the integration of optical flow information within a deformable model framework [63]. Based on this approach, the optical flow is treated as a non-holonomic constraint, i.e., it constrains the velocity of the model parameters, within a deformable model framework. As a result, we can simultaneously use edges and optical flow to improve our shape and motion estimation results. We have applied this framework to the estimation of facial shape and motion including expressions. Another important and novel aspect of this work is that

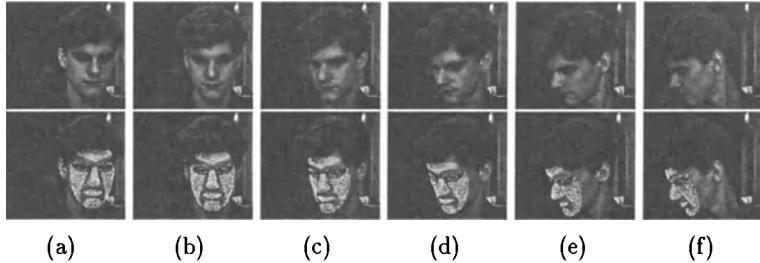


Figure 17.1 A face motion tracking example.

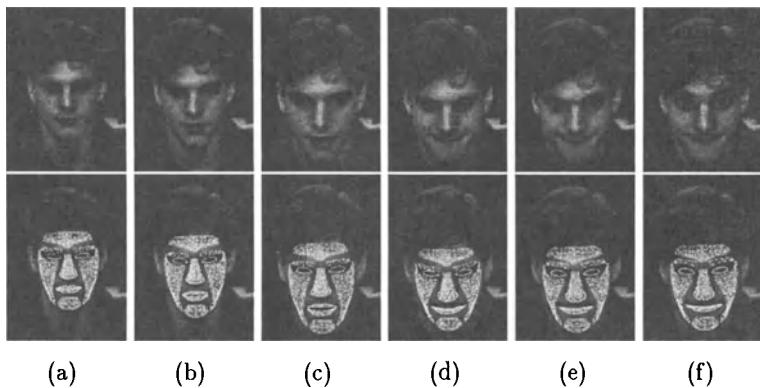


Figure 17.2 A face motion and expression tracking example.

we have constructed a deformable face mask whose parameterization is based on facial measurements statistics, taken by anthropometrists [82].

The following three figures show results from fitting a deformable mask (the mask's deformable parts are clearly visible) to the face of two subjects performing a variety of head motions and facial expressions. Notice that the shape of the deformable mask continuously changes based on the subject's facial expressions. Fig. 17.1, shows frames from a sequence where the subject rotates his head by a large amount. The face motion is successfully tracked given this significant rotation and self-occlusion. Fig. 17.2 shows the same subject making a series of face motions: opening the mouth in (b) and (c), smiling in (d) and (e), and raising the eyebrows in (e) and (f). In each case, the motion parameter values change appropriately, and at the correct times. Fig. 17.3 shows a different subject (with a very different face shape) turning her head and opening her mouth from (d) to (f). The face model is able to capture the different shape of her face, as well as the motion.

In computer graphics and animation, the creation of tools and the further development of our algorithms for faster and more realistic animations will be the major focus of our work. This includes control theory and motion capture



Figure 17.3 Another face motion and expression tracking example.

techniques, the modeling of a variety of physical phenomena, and the behavior modeling of living organisms.

In medical imaging, the improved modeling of the physical properties of various body tissues and organs and the reconstruction of internal or external body parts from various imaging modalities will allow significant advances in medical education and in clinical applications. Assessing a patient's condition, surgical planning, prostheses, athletic fitness and early diagnosis of possibly fatal diseases will undoubtedly benefit from these new methods.

A

APPENDIX

A.1 DERIVATION OF B MATRIX

Shabana [253] has shown that

$$\dot{\mathbf{R}}\mathbf{p} = \mathbf{R}(\boldsymbol{\omega}_\phi \times \mathbf{p}), \quad (\text{A.1})$$

where \mathbf{R} is the rotation matrix, \mathbf{p} is the position of a point on the deformable model with respect to the model frame ϕ and $\boldsymbol{\omega}_\phi$ is the angular velocity of the deformable model with respect to ϕ .

Furthermore, in [253] it is shown that $\boldsymbol{\omega}_\phi$ can be written in terms of the time derivative of the quaternion $\mathbf{q}_\theta = [s, \mathbf{v}_q]$ representing the rotation at time t as follows

$$\boldsymbol{\omega}_\phi = \mathbf{G}\dot{\mathbf{q}}_\theta, \quad (\text{A.2})$$

where \mathbf{G} is a 3×4 matrix whose definition is based on the value of the quaternion $\mathbf{q}_\theta = [s, \mathbf{v}_q]$,

$$\mathbf{G} = 2 \begin{bmatrix} -v_1 & s & v_3 & -v_2 \\ -v_2 & -v_3 & s & v_1 \\ -v_3 & v_2 & -v_1 & s \end{bmatrix}. \quad (\text{A.3})$$

Therefore, we can rewrite the vector $\dot{\mathbf{R}}\mathbf{p}$ using (A.2) as follows

$$\begin{aligned} \dot{\mathbf{R}}\mathbf{p} &= \mathbf{R}(\boldsymbol{\omega}_\phi \times \mathbf{p}) = -\mathbf{R}(\mathbf{p} \times \boldsymbol{\omega}_\phi) \\ &= -\mathbf{R} \tilde{\mathbf{p}} \mathbf{G} \dot{\mathbf{q}}_\theta, \end{aligned} \quad (\text{A.4})$$

where $\tilde{\mathbf{p}}$ is the dual 3×3 matrix of the position vector $\mathbf{p}(u) = (p_1, p_2, p_3)^T$ (see (2.3)) defined as

$$\tilde{\mathbf{p}}(u) = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}. \quad (\text{A.5})$$

Comparing (A.4) and (3.1), we identify matrix \mathbf{B} as

$$\mathbf{B} = -\mathbf{R} \tilde{\mathbf{p}} \mathbf{G}. \quad (\text{A.6})$$

A.2 INTEGRATION RULES

A.2.1 Gauss-Legendre Integration Rules

We use the Gauss-Legendre integration rules when we use isoparametric quadrilateral elements (e.g., bilinear quadrilateral elements). We first give the relevant formulas for *one-dimensional* functions and then we extend them to *two-dimensional* functions.

Suppose we have a function $f(\xi)$ which must be integrated over an *one-dimensional* element so that we are trying to find

$$I = \int_{-1}^1 f(\xi) d\xi. \quad (\text{A.7})$$

Then using the Gauss-Legendre rules we sample the function $f(\xi)$ at some predetermined sampling position $\xi = \bar{\xi}_i$ and multiply the results by some predetermined weights W_i . Thus for a n -point rule

$$I_n = \sum_{i=1}^n W_i f(\bar{\xi}_i). \quad (\text{A.8})$$

Table A.1 gives the sampling positions and weights for the first four Gauss-Legendre rules [114]. Note that an n -point rule exactly integrates a polynomial of degree $(2n - 1)$ or less exactly [114].

For an isoparametric quadrilateral let $g(\xi, \eta)$ be the function we want to integrate in the *two-dimensional* parameter space. Then using (A.8)

$$\begin{aligned} \int_{-1}^1 \int_{-1}^1 g(\xi, \eta) d\xi d\eta &= \int_{-1}^1 \sum_{i=1}^n W_i g(\bar{\xi}_i, \eta) d\eta \\ &= \sum_{i=1}^n \sum_{j=1}^n W_i W_j g(\bar{\xi}_i, \bar{\eta}_j), \end{aligned} \quad (\text{A.9})$$

n	ξ_i	W_i
1	0.0	2.0
2	± 0.577530	1.0
3	0.0	$8/9$
	± 0.774597	$5/9$
4	± 0.861136	0.347855
	± 0.339981	0.652145

Table A.1 Gauss-Legendre integration rules.

n	i	ξ_i	$\bar{\eta}_i$	W_i
1	1	1/3	1/3	1/2
3	1	1/2	0	1/6
	2	1/2	1/2	1/6
	3	0	1/2	1/6
7	1	0	0	1/40
	2	1/2	0	1/15
	3	1	0	1/40
	4	1/2	1/2	1/15
	5	0	1	1/40
	6	0	1/2	1/15
	7	1/3	1/3	9/40

Table A.2 Weights and sampling positions for numerical integration over triangular regions (Radau rules).

where W_i and W_j are the weight coefficients and $g(\bar{\xi}_i, \bar{\eta}_i)$ corresponds to the value of the function sampled at the sampling point $(\bar{\xi}_i, \bar{\eta}_i)$.

A.2.2 Radau Integration Rules

For isoparametric triangular elements we use the Radau rules to approximate the integral of a function $g(\xi, \eta)$ in the *two-dimensional* parametric space of the element. These rules have the following form [114]

$$\int_E g(\xi, \eta) d\xi d\eta = \sum_{i=1}^n W_i g(\bar{\xi}_i, \bar{\eta}_i), \quad (\text{A.10})$$

where the weights and the sampling positions are given in Table A.2.

REFERENCES

- [1] M. J. Ackerman, "The visible human project", *Journal of Biocommunication*, 18(2), pp. 14, 1991.
- [2] O. P. Agrawal, "Dynamic analysis of multi-body systems using tangent coordinates", In E. Bautista, J. Garcia-Lomas, and A. Navarro, editors, *The Theory of Machines and Mechanisms*, pp. 533-536, Pergamon Press, Oxford, 1987.
- [3] K. Akita, "Image sequence analysis of real world human motion", *Pattern Recognition*, 17, pp. 73-83, 1984.
- [4] P. F. Altman, J. F. Gibson, and C.C. Wang, *Handbook of respiration*, Philadelphia: Saunders Company, 1958.
- [5] A. Amini and J. Duncan, "Pointwise tracking of left-ventricular motion in 3D", Proc. IEEE Workshop on Visual Motion, Princeton, NJ, 294-298, 1991.
- [6] T. Arts, P.C. Veenstra, and R. S. Reneman, "Epicardial deformation and left ventricle wall mechanics during ejection in the dog", *Am. J. Physiol.*, 243, H379-H390, 1982.
- [7] T. Arts, W. C. Hunter, A. Douglas, M. M. Muijtjens, and R. S. Reneman, "Description of the deformation of the left ventricle by a kinematic model", *J Biomechanics*, 25(10), pp. 1119-1127, 1992.
- [8] T. Arts, W. C. Hunter, A. Douglas, M. M. Muijtjens, J. W. Corsel, and R. S. Reneman, "Macroscopic three-dimensional motion patterns of the left ventricle", *Advances in Experimental Medicine and Biology*, 346, pp. 383-392, 1993.
- [9] L. Axel and L. Dougherty, "Heart wall motion: Improved method of spatial modulation of magnetization for MR imaging", *Radiology*, 172, pp. 349-350, 1989.
- [10] L. Axel, F. Goncalves, and D. Bloomgarden, "Regional heart wall motion: Two-dimensional analysis and functional imaging of regional heart wall motion with magnetic resonance imaging", *Radiology*, 183, pp. 745-750, 1992.
- [11] L. Axel, D. Bloomgarden, C-N. Cheng, D. Kraitchman, and A. A. Young, "SPAMMVU: A Program for the analysis of dynamic tagged MRI", in: Book of Abstracts: Society of Magnetic Resonance in Medicine, 724, 1993.

- [12] H. Azhari, M. Buchalter, S. Sideman, E. Shapiro, and R. Beyar, “A conical model to describe the nonuniformity of the left ventricular twisting motion”, *Annals of Biomedical Engineering*, 20, pp. 149–165, 1992.
- [13] F. Azuola, N. I. Badler, P. Ho, I. A. Kakadiaris, D. Metaxas, and B. Ting, “Building anthropometry-based virtual human models”, In Proc. of the IMAGE VII Society Conference, Tucson, AZ, June 1994.
- [14] N. I. Badler, C. B. Phillips, and B. L. Webber, *Simulating humans: Computer graphics animation and control*, Oxford University Press, New York, 1993. ISBN 0-19-507359-2.
- [15] C. L. Bajaj, F. Bernandini, and G. Xu, “Automatic reconstruction of surfaces and scalar fields from 3D scans”, Proc. Computer Graphics (Siggraph’95), pp. 109–118, Los Angeles, CA, 1995.
- [16] R. Bajcsy and C. Broit, “Matching of deformed images”, Proc. of 6th ICPR, Munich, pp. 351–353, 1982.
- [17] R. Bajcsy and S. Kovacic, “Multiresolution elastic matching”, CVGIP, 46, pp. 1–21, 1989.
- [18] D. Baraff, “Coping with friction for non-penetrating rigid body simulation”, Computer Graphics (Proc. SIGGRAPH), 25(4), pp. 31–40, 1991.
- [19] D. Baraff and A. Witkin, “Dynamic simulation of non-penetrating flexible bodies”, Computer Graphics (Proc. SIGGRAPH), 26(2), pp. 303–308, 1992.
- [20] D. Baraff, “Fast contact force computation for nonpenetrating rigid bodies”, Proc. of SIGGRAPH ’94, pp. 23–34, Orlando, Florida, July 24–29, 1994.
- [21] D. Baraff, “Linear-time dynamics using Lagrange multipliers”, Proc. of SIGGRAPH ’96, pp. 137–146, New Orleans, 1996
- [22] E. Bardinet, N. Ayache, and L. D. Cohen, “Fitting of iso-surfaces using superquadrics and free-form deformations”, Proc. of IEEE Workshop on Biomedical Image Analysis, Seattle, WA, pp. 184–193, 1994.
- [23] A. Barr, “Superquadrics and angle-preserving transformations”, *IEEE Computer Graphics and Applications*, 1, pp. 11–23, 1981.
- [24] A. Barr, “Global and local deformations of solid primitives”, *Computer Graphics*, 18, pp. 21–30, 1984.
- [25] R. Barzel and A. Barr, “A modeling system based on dynamic constraints”, *Computer Graphics*, 22(4), pp. 179–188, 1988. Proc. ACM SIGGRAPH’88.
- [26] R. Barzel, *Physically-based modeling for computer graphics: A structured approach*, Academic Press, San Diego, 1992.
- [27] K.-J. Bathe and E.L. Wilson, *Numerical methods in finite element analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1976.

- [28] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems", *Computer Methods in Applied Mechanics and Engineering*, 1, pp. 1-16, 1972.
- [29] R. Beyar, S. Sideman, "Effect of the twisting motion on the non-uniformities of transmural fiber mechanics and energy demands - A theoretical study", *IEEE Trans. Biomed. Eng.*, 32, pp. 764-769, 1985.
- [30] I. Biederman, "Human image understanding: recent research and theory", *Computer Vision, Graphics, and Image Processing*, 32, pp. 29-73, 1985.
- [31] I. Biederman, "Recognition-by-components: a theory of human image understanding", *Psychological Review*, 94, pp. 115-147, April 1987.
- [32] T. Binford, "Visual perception by computer", In *IEEE Conference on Systems and Control*, December 1971.
- [33] A. G. Bharatkumar, K. E. Daigle, M. G. Pandy, Qin Cai, and J. K. Aggarwal, "Lower limb kinematics of human walking with the medial axis transformation", In *Proc. of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pp. 70-76, Austin, TX, November 11-12 1994.
- [34] M. Black and Y. Yacoob, "Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion", In Proc. IEEE Fifth International Conference on Computer Vision, ICCV '95, pp. 374-381, 1995.
- [35] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge, MA, 1987.
- [36] A. Blake, R. Curwen, and A. Zisserman, "Affine-invariant contour tracking with automatic control of spatiotemporal scale", In *Proc. IEEE 4th International Conference on Computer Vision*, pp. 502-507, 1993.
- [37] J. H. Bramble, J. E. Pasciak, and J. Xu, "The analysis of multigrid algorithms with nonnested spaces of noninherited quadratic forms", *Mathematics of Computation* 56(193), pp. 1-34, 1991.
- [38] T. Broida and R. Chellappa, "Estimation of object motion parameters from noisy images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1), pp. 90-98, January 1986.
- [39] T. J. Broida, S. Chandrashekhar, and R. Chellappa, "Recursive 3D motion estimation from a monocular image sequence", *IEEE Transactions on Aerospace and Electronic Systems*, 26(4), pp. 639-656, July 1990.
- [40] R. Brooks, "Symbolic Reasoning among 3D models and 2D images", *Artificial Intelligence*, 17:285-348, 1981.
- [41] R. W. Brower, H. T. Katen, and G. J. Meester, "Direct method for determining regional myocardial shortening after bypass surgery from radiopaque markers in man", *Am. J. Cardiol.*, 41, pp. 1222-1229, 1978.

- [42] A. Bruderlin and T. Calvert, "Goal-directed, dynamic animation of human walking", In Computer Graphics (Proc. SIGGRAPH), volume 23, pp. 233–242. ACM, July 1989.
- [43] J. Canny, "A computational approach to edge detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), pp. 679–698, November 1986.
- [44] G. Celniker, "Deformable curve and surface finite elements for free-form shape design", Computer Graphics (Proc. SIGGRAPH), 25, pp. 257–266, 1991.
- [45] J.E. Chadwick, D.R. Haumann, and R.E. Parent, "Layered construction for animated deformable characters", Computer Graphics (Proc. SIGGRAPH), 23(3), pp. 243–252, 1989.
- [46] M. Chan and D. Metaxas, "Physics-based object pose and shape estimation from multiple views", In Proc. IEEE 12th International Conference on Pattern Recognition, volume I, pp. 326–330, 1994.
- [47] M. Chan, D. Metaxas, and S. Dickinson, "A new approach to tracking 3d objects in 2d image sequences", In 1994 Proc. of the 12th National Conference on Artificial Intelligence, pp. 960–965, 1994.
- [48] C. O. Chang and P. E. Nikravesh, "An Adaptive constraint violation stabilization method for dynamic analysis of mechanical systems", Journal of Mech., Trans., and Auto. in Design, 107, pp. 488–492, 1985.
- [49] C. W. Chen and T. S. Huang, "Epicardial motion and deformation estimation from coronary artery bifurcation points", Proc. of 3rd International Conference on Computer Vision, Osaka, Japan, pp. 456–459, 1990.
- [50] Z. Chen and H. J. Lee, "Knowledge-guided visual perception of 3D human gait from single image sequence", *IEEE Transactions on Systems, Man and Cybernetics*, 22(2), pp. 336–342, 1992.
- [51] J. Chen, and N. Lobo, "Toward interactive-rate simulation of fluids with moving obstacles using the Navier-Stokes equations", Graphical Models and Image Processing, March 1995, pp. 107–116, 1995.
- [52] L. D. Cohen and I. Cohen, "A finite element method applied to new active contour models and 3D reconstruction from cross sections", Proc. 3rd International Conference on Computer Vision, Osaka, Japan, pp. 587–591, 1990.
- [53] L. Cohen and I. Cohen, "Finite element methods for active contour models and balloons for 2D and 3D images", Pattern Analysis and Machine Intelligence, 15(11), pp. 1131–1147, 1993.
- [54] M. F. Cohen, "Interactive spacetime control for animation", In Edwin E. Catmull, editor, Computer Graphics (SIGGRAPH '92 Proceedings), volume 26, pp. 293–302, July 1992.

- [55] I. Cohen, L. Cohen, and N. Ayache, "Using deformable surfaces to segment 3D images and infer differential structure", *CVGIP: Image Understanding*, 56(2), pp. 242-263, 1992.
- [56] T.J. Coleman and J.E. Randell, "HUMAN—a comprehensive physiological model", *Physiologist*, 26(1), pp. 15-21, 1983.
- [57] S. A. Cover, N. F. Ezquerra, J.F. O'Brien, and et al, "Interactively deformable models for surgery simulation", *IEEE Computer Graphics & Applications*, 13(6), pp. 68-75, 1993.
- [58] R. Cook, and K. Torrance, "A reflectance model for computer graphics", *ACM Transactions on Graphics*, 1(1), pp. 7-24, 1982.
- [59] J. J. Craig, *Introduction to robotics: mechanics and control*, Addison-Wesley, 1989.
- [60] D. DeCarlo and D. Metaxas, "Blended deformable models", *IEEE PAMI*, 18(4), pp. 834-839, April, 1996 (See also Proc. Computer Vision and Pattern Recognition Conference'94, Seattle, Washington, June 1994).
- [61] D. DeCarlo and D. Metaxas, "Adaptive shape evolution using blending", In Proc. IEEE Fifth International Conference on Computer Vision, pp. 834-839, Cambridge, MA, June 1995.
- [62] D. DeCarlo, J. Kaye, D. Metaxas, J.R. Clarke, B. Webber, and N. Badler, "Integrating anatomy and physiology for behavior modeling", in *Medicine Meets Virtual Reality III: Interactive Technology and the New Paradigm for Healthcare*, RM Satava, K Morgan, HB Sieburg, *et al* eds., 1995.
- [63] D. DeCarlo and D. Metaxas, "The integration of optical flow and deformable models with applications to human face shape and motion estimation", In Proc. CVPR '96, pp. 231-238, 1996.
- [64] H. Delingette, M. Hebert and K. Ikeuchi, "Shape representation and image segmentation using deformable surfaces", *Image and Vision Computing*, Vol. 10, No. 3, pp.132-144, April 1992.
- [65] H. Delingette, M. Hebert and K. Ikeuchi, "A spherical representation for the recognition of curved objects", Proc. of IEEE Fourth International Conference on Computer Vision, Berlin, Germany, May 1993.
- [66] H. Delingette, "Simplex meshes: A general representation for 3d shape reconstruction", Technical Report No. 2214, INRIA, Sophia Antipolis, France, 1994.
- [67] T.S. Jr. Denney, J. L. Prince, "3D displacement field reconstruction on an irregular domain from planar tagged cardiac MR images", Proc. IEEE Workshop on Motion of Non-Rigid and Articulated Objects, pp. 172-177, Austin, TX, 1994.
- [68] R. Deriche and O. Faugeras, "Tracking line segments", *Image and Vision Computing*, 8(4), pp. 261-270, November 1990.

- [69] G. Dhatt and G. Touzot, *The finite element method displayed*, Wiley, New York, 1984.
- [70] S. Dickinson, A. Pentland, and A. Rosenfeld, “A representation for qualitative 3-D object recognition integrating object-centered and viewer-centered models”, In K. Leibovic, editor, *Vision: A Convergence of Disciplines*, Springer Verlag, New York, 1990.
- [71] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld, “3-D Shape recovery using distributed aspect matching”, *IEEE Trans. Pattern Analysis and Machine Intelligence, special issue on Interpretation of 3-D Scenes*, 14(2), pp. 174–198, February 1992.
- [72] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld, “From volumes to views: An approach to 3 object recognition”, *Computer Vision Graphics and Image Processing:Image Understanding, special issue on CAD-based vision*, 55(2), March 1992.
- [73] S. Dickinson, H. Christensen, J. Tsotsos, and G. Olofsson, “Active object recognition integrating attention and viewpoint control”, In Proc. ECCV '94, Stockholm, Sweden, May 1994.
- [74] S. Dickinson and D. Metaxas, “Integrating qualitative and quantitative shape recovery”, *International Journal of Computer Vision*, 13(3), pp. 1–20, 1994.
- [75] C. J. Dickinson, *A computer model of human respiration*, Baltimore: University Park Press, 1977.
- [76] E. D. Dickmanns and V. Graefe, “Dynamic monocular machine vision”, *Machine Vision and Applications*, 1, pp. 223–240, 1988.
- [77] E. D. Dickmanns and V. Graefe, “Applications of dynamic monocular machine vision”, *Machine Vision and Applications*, 1, pp. 241–261, 1988.
- [78] A. C. M. Dumay, “Triage simulation in a virtual environment”, in *Medicine Meets Virtual Reality III: Interactive Technology and the New Paradigm for Healthcare*, RM Satava, K Morgan, HB Sieburg, et al eds., pp. 101-111, 1995.
- [79] P. Ekman and W. Friesen, *The facial action coding system*, Consulting Psychologist Press, Inc., 1978.
- [80] T. Emerson, J. Prothero, and S. Weghorst, “Medicine and virtual reality: A guide to the literature”, Tech Report B-94-2, Human Interface Technology Lab of the University of Washington: 1994. Available electronically at <ftp://ftp.u.washington.edu>.
- [81] I. Essa and A. Pentland, “Facial expression recognition using a dynamic model and motion energy”, In Proc. IEEE Fifth International Conference on Computer Vision, ICCV '95, pp. 360–367, 1995.
- [82] L. G. Farkas, *Anthropometry of the Head and Face*, Raven Press, 1994.

- [83] R. Featherstone, *Robot dynamics algorithms*, Kluwer Academic Publishers, 1987.
- [84] J. Feldmar, N. Ayache, and F. Betting, “3D-2D projective registration of free-form curves and surfaces”, In Proc. of the IEEE Fifth International Conference on Computer Vision [227], pp. 549–556, Cambridge, MA 1995.
- [85] F. Ferrie, J. Lagarde, and P. Whaite, “Darboux frames, snakes and superquadrics: Geometry from the bottom up”, *IEEE Pattern Analysis and Machine Intelligence*, 15(8), pp. 771–784, 1993.
- [86] S. E. Fischer, G. C. McKinnon, M. B. Scheidegger, W. Prins, D. Meier, and P. Boesiger, “True myocardial motion tracking”, *Magnetic Resonance in Medicine*, 31, pp. 401–413, 1994.
- [87] C. A. J. Fletcher, *Computational techniques for fluid dynamics*, Springer Verlag, Sydney, 1990.
- [88] N. Foster and D. Metaxas, “Visualization of dynamic fluid simulations”, *Engineering Computations*, 12, pp. 109–124, 1995.
- [89] N. Foster and D. Metaxas, “Realistic animation of liquids”, In Proc. Graphics Interface ’96, pp. 204–212, 1996 (also to appear in Graphical Models and Image Processing (GMIP)).
- [90] A. Fournier, and W. Reeves, “A simple model of ocean waves”, Proc. of SIGGRAPH ’86, in *Computer Graphics*, 20(3), pp. 75–84, 1986.
- [91] D. Fribolet, I. E. Magnin, and D. Revel, “Assessment of a model for overall left ventricular three-dimensional motion from MRI data”, *International Journal of Cardiac Imaging*, 8, pp. 175–190, 1992.
- [92] D. Fribolet, I. E. Magnin, C. Mathieu, A. Pommert, and K. H. Hoehne, “Assessment and visualization of the curvature of the left ventricle from 3D medical images”, *Computerized Medical Imaging and Graphics*, 17(4/5), pp. 257–262, 1993.
- [93] W. F. Ganong, *Review of Medical Physiology.*, 16 ed., Norwalk, CT: Appleton & Lange, 1993.
- [94] M. Gardiner, “The superellipse: A curve between the ellipse and the rectangle”, *Scientific American*, 213, pp. 222–234, 1965.
- [95] D. M. Gavrilla and L. S. Davis, “Towards 3-D model-based tracking and recognition of human movement”, In Proc. of the IEEE International Workshop on Face and Gesture Recognition, Zurich, Switzerland, June 1995.
- [96] J. C. Gee, “Probabilistic matching of deformed images”, PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, 1996.
- [97] A. Gelb, *Applied optimal estimation*, MIT Press, Cambridge, MA, 1974.

- [98] D. Gennery, "Visual tracking of known three-dimensional objects", *International Journal of Computer Vision*, 7(3), pp. 243–270, 1992.
- [99] D. N. Ghista and H. S. Hamid, "Finite element stress analysis of the human left ventricle whose irregular shape is developed from single plane cineangiogram", *Computer Programs in Biomedicine*, 7, pp. 219-231, 1977.
- [100] D.W. Johnson, E.G. Gilbert, and S.S. Keerthi, "A fast procedure for computing the distance between objects", *IEEE Journal of Robotics and Automation*, 1988.
- [101] P.E. Gill, W. Murray, and M.H. Wright, *Practical optimization*, Academic Press, London, 1981.
- [102] L. Goncalves, E. Di Bernandom, E. Ursella, and P. Perona, "Monocular tracking of the human arm in 3D", In Proc. of the IEEE Fifth International Conference on Computer Vision [227], pp. 764–770, 1995, Cambridge, MA 1995.
- [103] M. E. Goss, "A Real-time particle system for display of ship wakes", *IEEE Computer Graphics and Applications*, 10(3), pp. 30-35, 1990.
- [104] J.-P Gourret, N. Thalmann, and D. Thalmann, "Simulation of object and human skin deformations in a grasping task", *Computer Graphics (Proc. SIGGRAPH)*, 23(3), pp. 21–30, 1989.
- [105] A.D. Gross and T.E. Boult, "Error of fit measures for recovering parametric solids", In *Second International Conference on Computer Vision*, pp. 690–694, 1988.
- [106] J. M. Guccione and A. D. McCulloch, "Finite element modeling of ventricular mechanics", in: L. Glass, P. Hunter, A. McCulloch, eds. *Theory of Heart: biomechanics, biophysics, and nonlinear dynamics of cardiac function*, New York, Springer-Verlag, pp. 121-144, 1991
- [107] A. Gupta and R. Bajcsy, "Part description and segmentation using contour, surface, and volumetric primitives", In *Sensing and Reconstruction of Three-Dimensional Objects and Scenes, Proc. SPIE 1260*, B. Girod (ed.), pp. 203–214, 1990.
- [108] A. J. Hanson, "Hyperquadrics: smoothly deformable shapes with convex polyhedral bounds", *Computer Vision, Graphics, and Image Processing*, 44, pp. 191–210, 1988.
- [109] F. H. Harlow and J. E. Welch, "Numerical calculation of time-dependent viscous incompressible flow", *Phys. Fluids*, 8, pp. 2182–2189, 1965.
- [110] D. C. Harrison, A. Goldblatt, E. Braunwald, G. Glick, and D. T. Mason, "Studies on cardiac dimensions in intact unanesthetized man. 1. Description of techniques and their validation", *Circ. Res.*, 13, pp. 448-455, 1963.

- [111] D. Haumann, "Modeling the physical behavior of flexible objects", *Topics in Physically-Based Modeling*, A. Barr, et al. (eds.), *ACM SIGGRAPH '87 Course Notes*, 17, 1987.
- [112] J. Heel, "Temporally integrated surface reconstruction", In *IEEE Third International Conference on Computer Vision (ICCV'90)*, Osaka, Japan, Dec., pp. 292–295, 1990.
- [113] E. Hinton and D. Owen, *Finite element programming*, Academic Press, 1977.
- [114] E. Hinton and D. Owen, *An introduction to finite element computations*, Pineridge Press, Swansea, U.K., 1979.
- [115] M. W. Hirsch, *Differentiable topology*, Springer-Verlag, 1991.
- [116] J. K. Hodgins, P. K. Sweeney, and D. G. Lawrence, "Generating natural-looking motion for computer animation", In Proc. of Graphics Interface '92, pp. 265–272, May 1992.
- [117] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien, "Animating human athletics", In Proc. SIGGRAPH 95, pp. 71–78, August 1995.
- [118] C.M. Hoffmann, *Geometric and solid modeling*, Morgan-Kaufmann, Palo Alto, 1989.
- [119] D. Hogg, "Model-based vision: A program to see a walking person", *Image and Vision Computing*, 1(1), pp. 5–20, 1983.
- [120] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction", Proc. Computer Graphics (SIGGRAPH'94), pp. 295–302, Orlando, Florida, 1994.
- [121] B.K.P. Horn, *Robot Vision*, McGraw-Hill, 1986.
- [122] T. S. Huang, "Modeling, analysis and visualization of nonrigid object motion", In Proc. IEEE 10th International Conference on Pattern Recognition, volume 1, pp. 361–364, 1990.
- [123] W. C. Huang, and D. Goldgof, "Adaptive-Size physically-based models for nonrigid motion analysis", Proc. IEEE Computer Vision and Pattern Recognition Conference (CVPR'92), pp. 833–835, Champaign, Illinois, June 1992.
- [124] W. C. Huang and D. Goldgof, "Adaptive-size meshes for rigid and non-rigid shape analysis and synthesis", *IEEE Transactions on Pattern Analysis*, 15(6), pp. 611–616, 1993.
- [125] *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Seattle, WA, June 21-23 1994. IEEE Computer Society Press, New York, NY.

- [126] P. M. Isaacs and M. F. Cohen, "Mixed methods for complex kinematic constraints in dynamic figure animation", *The Visual Computer*, 4(6), pp. 296–305, Dec. 1988.
- [127] R. F. Janz and R. J. Waldron, "Predicted effect of chronic apical aneurysms on the passive stiffness of the human left ventricle", *Circ. Res.*, 42(2), pp. 255-263, 1978.
- [128] T. Joshi and J. Ponce, "Hot curves for modeling and recognition of smooth curved 3D objects", In IEEE Computer Society Conference on Computer Vision and Pattern Recognition [125].
- [129] I. A. Kakadiaris, D. Metaxas, and R. Bajcsy, "Active part-decomposition, shape and motion estimation of articulated objects: A physics-based approach", In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 980-984, Seattle, WA, June 21-23 1994.
- [130] I. A. Kakadiaris, D. Metaxas, and R. Bajcsy, "Active motion-based segmentation of human body outlines", In Proc. of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects, pp. 50-56, Austin, TX, November 11-12 1994.
- [131] I. A. Kakadiaris and D. Metaxas, "3D human body model acquisition from multiple views", In Proc. of the IEEE Fifth International Conference on Computer Vision, pp. 618-623, Boston, MA, June 20-23, 1995.
- [132] I. A. Kakadiaris and D. Metaxas, "Model based estimation of 3D human motion with occlusion based on active multi-viewpoint selection", In Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 81-87, San Francisco, CA, June 1996.
- [133] I. A. Kakadiaris, "Motion-based part identification, shape and motion estimation of complex multi-part objects: Application to human body tracking", Technical Report MS-CIS-96-207, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, August 1996.
- [134] C. Kambhamettu C. and D. Goldgof, "Point correspondence recovery in non-rigid motion", Proc. of IEEE Conference on Computer Vision and Pattern Recognition, Champaign, Illinois, pp. 222-227, 1992.
- [135] H. Kardestuncer, *Finite element handbook*, McGraw-Hill, New York, 1987.
- [136] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models", Proc. IEEE First International Conference on Computer Vision, pp. 259–268, 1987.
- [137] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models", *International Journal of Computer Vision*, 1(4), pp. 321–331, 1988.
- [138] M. Kass and G. Miller, "Rapid, stable fluid dynamics for computer graphics", Computer Graphics (Proc. SIGGRAPH), 24(4), pp. 49–57, 1990.

- [139] J. Kaye, D. Metaxas, J. R. Clarke, and B. L. Webber, "Lung modeling: Integrating anatomy and physiology", In Proc. of the First International Conference on Medical Robotics and Computer-Assisted Surgery, Pittsburgh, PA, September 1994.
- [140] J. Kaye, D. Metaxas, J. R. Clarke, B. L. Webber, and N. Badler, "Integrating anatomy and physiology for behavior modeling", in Medicine Meets Virtual Reality III, San Diego, 1995.
- [141] J. Kaye, D. Metaxas, and F.P. Primiano, "Modeling respiratory mechanics in VR", in the Proc. of the Nineteenth Symposium on Computer Applications for Medical Care (SCAMC-95), New Orleans, October, pp. 488-92, 1995.
- [142] J. Kaye, F.P. Primiano, and D. Metaxas, "Anatomical and physiological simulation for respiratory mechanics", in the Proc. of the Second Conference on Medical Robotics and Computer-Assisted Surgery (MRCAS-95), Baltimore, November 1995. Also appears with same title in J. Img Guid Surg, 1(3), pp. 164-171, 1995
- [143] J. Kaye, D. Metaxas and F.P. Primiano, " Modeling cardiopulmonary interactions in trauma", in the Proc. of IMAGE 96, the IMAGE Society, Scottsdale, Arizona, June 1996.
- [144] H. C. Kim, B. G. Min, M. M. Lee, J. D. Seo, Y. W. Lee, and M. C. Han, "Estimation of local cardiac wall deformation and regional wall stress from biplane coronary cineangiograms", *IEEE Trans. Biomed. Eng.*, 32, pp. 503-511, 1985.
- [145] B.B. Kimia, A. Tannenbaum, and S.W. Zucker, "Toward a computational theory of shape: An overview", Technical Report TR-CIM-89-13, Computer Vision and Robotics Laboratory, 1989.
- [146] J. Koenderink and A. van Doorn, "The internal representation of solid shape with respect to vision", Biological Cybernetics, 32, pp. 211-216, 1979.
- [147] E. Koh, D. Metaxas and N. I. Badler, "Hierarchical shape representation using locally adaptive finite elements", Proc. Third European Conference on Computer Vision (ECCV'94, pp. 441-446, Stockholm, Sweden, May, 1994.
- [148] E. Kokkevis, D. Metaxas and N. I. Badler, "Autonomous animation and control of four-legged animals", Proc. Graphics Interface'95, pp. 10-17, Quebec City, Quebec, Canada, May 1995.
- [149] E. Kokkevis, D. Metaxas and N. I. Badler, "User controlled physics-based animation of articulated figures", Proc. Computer Animation'96, pp. 16-26, Geneva, Switzerland, 1996.
- [150] J. J. Kuch and T. S. Huang, "Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration", In Proc. of the Fifth International Conference on Computer Vision [227], pp. 666-672, 1995.

- [151] B.C. Kuo, *Automatic control systems*, Prentice Hall, 1991.
- [152] S. Kurakake and R. Nevatia, “Description and tracking of moving articulated objects”, In *Proc. of the IEEE International Conference on Pattern Recognition*, pp. 491–495, 1992.
- [153] A. Lamouret and M.-P. Gascuel, “Scripting interactive physically-based motions with relative paths and synchronization”, In *Proc. of Graphics Interface '95*, pp. 18–25, May 1995.
- [154] I.D. Landau, *Adaptive control, the model reference approach*, Marcel Dekker, New York, 1979.
- [155] A. Lanitis, C.J. Taylor, and T.F. Cootes, “A unified approach for coding and interpreting face images”, In Proc. of the IEEE Fifth International Conference on Computer Vision, ICCV '95, pp. 368–373, 1995.
- [156] R. Szeliski S. Lavallée, and L. Brunie, “Matching 3-D smooth surfaces with their 2-D projections using 3-D distance maps”, In SPIE, Geometric Methods in Computer Vision, 1991.
- [157] J. Lee, R. Haralick and L. Shapiro, “Morphologic edge detection”, IEEE Journal of Robotics and Automation, 3(2), pp. 211–216, 1979.
- [158] A. Leonardis, A. Gupta, and R. Bajcsy, “Segmentation as the search for the best description of the image in terms of primitives”, In *IEEE Third International Conference on Computer Vision (ICCV'90), Osaka, Japan, Dec.*, pp. 121–125, 1990.
- [159] A. Leonardis, F. Solina, and A. Macerl, “A direct recovery of superquadric models in range images using recover-and-select paradigm”, In Proc. European Conference on Computer Vision, ECCV '94, pp. 309–318, 1994.
- [160] M. K. Leung and Y. H. Yang, “Human body motion segmentation in a complex scene”, *Pattern Recognition*, 20(1), pp. 55–64, 1987.
- [161] M. K. Leung and Y. H. Yang, “A region based approach for human body motion analysis”, *Pattern Recognition*, 20(3), pp. 321–339, 1987.
- [162] M. K. Leung and Y. H. Yang, “An empirical approach to human body analysis”, Technical Report 94-1, University of Saskatchewan, Saskatchewan, Canada, 1994.
- [163] Maylor K. Leung and Yee-Hong Yang, “First sight: A human body outline labeling system”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4), pp. 369–377, April 1995.
- [164] H. Li, P. Roivainen, and R. Forchheimer, “3-D motion estimation in model-based facial image coding”, *PAMI*, 15(6), pp. 545–555, June 1993.
- [165] C. Liao and G. Medioni, “Simultaneous segmentation and approximation of complex patterns”, In Proc. CVPR '94, pp. 617–623, 1994.

- [166] J. R. Ligas and F.P. Primiano, Jr., "Respiratory mechanics", in *Encyclopedia of Medical Instrumentation*, J.G. Webster, editor, vol. 4, John Wiley & Sons: New York, 2550-2573, 1988.
- [167] D. Lowe, "Fitting parameterized three-dimensional models to images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5), pp. 441-450, 1991.
- [168] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modeling with front propagation: A level set approach", *IEEE Pattern Analysis and Machine Intelligence*, 17(2), 1995.
- [169] R. Mann, A. Jepson, and J. M. Siskind, "Computational perception of scene dynamics", In Bernard Buxton and Roberto Cipolla, editors, *Fourth European Conference on Computer Vision*, Lecture Notes in Computer Science, pp. 528-539, Cambridge, UK, April 14-18 1996.
- [170] D. Manocha and J. F. Canny, "A new approach for surface intersection", *International Journal of Computational Geometry and Applications*, pp. 491-516, 1991.
- [171] D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes", *Proc. Roy. Soc. London, B*, 200, pp. 269-294, 1978.
- [172] L. Matthies, T. Kanade, and R. Szeliski, "Kalman Filter-based algorithms for estimating depth from image sequences", *International Journal of Computer Vision*, 3, pp. 209-236, 1989.
- [173] N. Max, "Vectorized procedural models for natural terrain: Waves and islands in the sunset", Proc. of SIGGRAPH '81, in Computer Graphics, 15(3), pp. 317-324, 1981.
- [174] G. D. Meier, A. A. Bove, W. P. Santamore, P. R. Lynch, "Contractile function in canine right ventricle", *Am. J. Physiol.*, 239, H794-H804, 1980.
- [175] D. Metaxas and D. Terzopoulos, "Shape representation and recovery using deformable superquadrics", *Proc. International Workshop on Visual Form, Capri, Italy*, pp. 389-398, May 1991.
- [176] D. Metaxas and D. Terzopoulos, "Constrained deformable superquadrics and nonrigid motion tracking", *Proc. IEEE Computer Vision and Pattern Recognition Conference (CVPR'91), Hawaii*, pp. 337-343, June 1991.
- [177] D. Metaxas and D. Terzopoulos, "Recursive estimation of nonrigid shape and motion", Proc. IEEE Motion Workshop, Princeton, NJ, pp. 306-311, October 1991.
- [178] D. Metaxas and D. Terzopoulos, "Dynamic deformation of solid primitives with constraints", Computer Graphics (Proc. ACM Siggraph'92, Chicago, IL, July), Vol. 26(2), pp. 309-312, 1992.

- [179] D. Metaxas and D. Terzopoulos, “Flexible multibody dynamics techniques for shape and nonrigid motion estimation and synthesis”, Proc. ASME Symposium on the Dynamics of Flexible Multibody Systems: Theory and Experiment, Anaheim, CA, pp. 147-155, November 1992.
- [180] D. Metaxas, *Physics-based modeling of nonrigid objects for vision and graphics*, Ph.D. Thesis, Department of Computer Science, University of Toronto, 1992.
- [181] D. Metaxas and D. Terzopoulos, “Shape and nonrigid motion estimation through physics-based synthesis”, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-15(6), pp. 580-591, June 1993.
- [182] D. Metaxas and D. Terzopoulos, “Nonrigid multibody dynamics for model synthesis and estimation”, Proc. 3rd Pan American Congress of Applied Mechanics, Sao Paulo, Brazil, January 4-8, 1993.
- [183] D. Metaxas and S. Dickinson, “Integration of quantitative and qualitative techniques for deformable model fitting from orthographic, perspective, and stereo projections”, In Proc. IEEE Fourth International Conference on Computer Vision, pp. 641-649, 1993.
- [184] D. Metaxas and I. A. Kakadiaris, “Elastically adaptive deformable models”, In Proc. European Conference on Computer Vision (ECCV’96), pp. 550-559, Cambridge, UK, April 1996.
- [185] D. Metaxas and I. A. Kakadiaris, ““Smart” deformable models”, In Proc. of the Ninth Image and Multidimensional Signal Processing (IMDSP) Workshop, Belize City, March 3 - 6, 1996.
- [186] D. Metaxas, E. Koh, and N. I. Badler, “Multi-level shape representation using global deformations and locally adaptive finite elements”, International Journal of Computer Vision, to appear.
- [187] D. Metaxas and E. Koh, “Flexible multibody dynamics and adaptive finite element techniques for model synthesis and estimation”, *Journal of Computer Methods in Applied Mechanics and Engineering*, 136(1-2), pp. 1-25, 1996.
- [188] T. McInerney, *Finite element techniques for fitting deformable models to 3D data, MSc Thesis, Dept. of Computer Science, University of Toronto, Toronto, CA*, 1992.
- [189] T. McInerney and D. Terzopoulos, “A finite element model for 3d shape reconstruction and non-rigid motion tracking”, In Proc. of the International Conference on Computer Vision (ICCV’93), pp. 518-523, Berlin, May, 1993.
- [190] T. McInerney and D. Terzopoulos, “Topologically adaptable snakes”, Proc. IEEE Fifth International Conference on Computer Vision (ICCV’95), pp. 840-845, Cambridge, MA, June 1995.

- [191] M. McKenna and D. Zeltzer, "Dynamic simulation of autonomous legged locomotion", In Computer Graphics (Proc. SIGGRAPH), volume 24. ACM, August 1990.
- [192] G. Miller, "The motion dynamics of snakes and Worms", Computer Graphics (Proc. SIGGRAPH), 20(4), pp. 169–178, 1988.
- [193] G. Miller and A. Pearce, "Globular dynamics: A connected particle system for animating viscous fluids", Computers and Graphics, 13(3), pp. 305–309, 1989.
- [194] S. K. Mishra and D. B. Goldgof, "Motion analysis and modeling of epicardial surfaces from point and line correspondences", Proc. of IEEE Workshop on Visual Motion, Princeton, NJ, pp. 300-305, 1991.
- [195] C. Moore, W. O'Dell, E. McVeigh, and E. Zerhouni, "Calculation of three-dimensional left ventricular strains form biplanar tagged MR images", *J Mag Res Imag*, 2, pp. 165-175, 1992.
- [196] M. Moore and J. Wilhelms, "Collision detection and response for computer animation", In John Dill, editor, Computer Graphics (Proc. SIGGRAPH '88), 22, pp. 289–298, August 1988.
- [197] Y. Moses, D. Reynard, and A. Blake, "Robust real time tracking and classification of facial expressions", In Proc. IEEE Fifth International Conference on Computer Vision, ICCV '95, pp. 296–301, 1995.
- [198] NASA, The anthropometry source book. Technical Report NASA Reference Publication 1024, Johnson Space Center, Houston, TX, 1978.
- [199] C. Nastar and N. Ayache, "Spatio-temporal analysis of nonrigid motion from 4D data", Proc. IEEE Workshop on Motion of Non-Rigid and Articulated Objects, pp. 146-151, Austin, TX, November 1994.
- [200] S. Neumann, "Modeling acute hemorrhage in the human cardiovascular system", *Annals of Biomedical Engineering*, 21 (Supplement 1), pp. 13, Abstract 64, October, 1993.
- [201] B. D. Nichols, "Calculating three dimensional free surface flows in the vicinity of submerged and exposed structures", *J. Comp. Phys.*, 12, pp. 234–245, 1973.
- [202] N. Nilsson, *Principles of artificial intelligence*, Morgan Kaufman Publishers, Inc., Los Altos, CA, 1980.
- [203] T. Nishita, and E. Nakamae, "Method of displaying optical effects within water: Using the accumulation buffer", Proc. of SIGGRAPH '94, (July 1994), pp. 24–29, 1994.
- [204] S. A. Niyogi and E. H. Adelson, Analyzing and recognizing walking figures in XYT, In IEEE Computer Society Conference on Computer Vision and Pattern Recognition [125], pp. 469–474.

- [205] W. G. O'Dell, C. C. Moore, and E. R. McVeigh, "Displacement field fitting approach to calculate 3D deformations from parallel-tagged MR images", *J. Magn. Reson. Imag.*, 3(P), P208, 1993.
- [206] P. V. O'Neil, *Advanced engineering mathematics*, 2nd ed., Belmont, Calif., Wadsworth Pub. Co., 1987.
- [207] J. O'Rourke and N. I. Badler, "Model-based image analysis of human motion using constraint propagation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6), pp. 522–536, 1980.
- [208] J. Park, D. Metaxas, and A. Young, "Deformable models with parameter functions: Application to heart-wall modeling", Proc. IEEE Computer Vision and Pattern Recognition, Seattle, WA, pp. 437-442, 1994.
- [209] J. Park, D. Metaxas, A. Young A, and L. Axel, "Model-based analysis of cardiac motion from tagged MRI data", Proc. Seventh Annual IEEE Symposium on Computer-Based Medical Systems, Winston-Salem, North Carolina, pp. 40-45, 1994.
- [210] J. Park, D. Metaxas, and L. Axel, "Volumetric deformable models with parameter functions: A New approach to the 3D motion analysis of the LV from MRI-SPAMM", Proc. of IEEE Fifth International Conference on Computer Vision, Cambridge, Massachusetts, pp. 700-705, 1995.
- [211] J. Park, D. Metaxas, A. A. Young, and L. Axel, "Deformable models with parameter functions for cardiac motion analysis from tagged MRI data", *IEEE Trans. on Medical Imaging*, June, 1996.
- [212] J. Park, D. Metaxas, A. A. Young, and L. Axel, "Analysis of left ventricular wall motion based on volumetric deformable models and MRI-SPAMM", *Medical Image Analysis Journal*, 1(1), pp. 53-71, April, 1996.
- [213] D. Peachy, "Modeling waves and surf", Proc. of SIGGRAPH '86, in Computer Graphics, 20(3), pp. 65–74, 1986.
- [214] A. Pentland, "Perceptual organization and the representation of natural form", *Artificial Intelligence*, 28, pp. 293–331, 1986.
- [215] A. Pentland and J. Williams, "Good vibrations: Modal dynamics for graphics and animation", Computer Graphics (Proc. SIGGRAPH), 23(3), pp. 215–222, 1989.
- [216] A. Pentland, "Automatic extraction of deformable part models", *International Journal of Computer Vision*, pp. 107–126, 1990.
- [217] A. Pentland and B. Horowitz, "Recovery of non-rigid motion and structure", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7), pp. 730–742, 1991.
- [218] A. Pentland and S. Sclaroff, "Closed-Form solutions for physically based shape modeling and recognition", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7), pp. 715–729, 1991.

- [219] A. Pentland, B. Horowitz, and S. Sclaroff, “Non-rigid motion and structure from contour”, Proc. of IEEE Workshop on Visual Motion, Princeton, NJ, pp. 288–293, 1991.
- [220] C. B. Phillips and N. I. Badler, “Interactive behaviors for bipedal articulated figures”, In Thomas W. Sederberg, editor, Computer Graphics (Proc. SIGGRAPH '91), volume 25, pp. 359–362, July 1991.
- [221] B. Placek, “Contribution to the solution of the equations of motion of the discrete dynamical system with holonomic constraints”, In E. Bautista, J. Garcia-Lomas, and A. Navarro, editors, The Theory of Machines and Mechanisms, pp. 379–382, Pergamon Press, Oxford, 1987.
- [222] S. Platt and N. I. Badler, “Animating facial expressions”, Computer Graphics 15(3), pp. 245–252, Aug, 1981.
- [223] J. Platt, *Constraint methods for neural networks and computer graphics*, PhD Thesis, Dept. of Computer Science, California Institute of Technology, Pasadena, CA (Caltech-CS-TR-89-07), 1989.
- [224] J. Platt and A. Barr, “Constraint methods for flexible objects”, Computer Graphics (Proc. SIGGRAPH), 22(4), pp. 279–288, 1989.
- [225] T. Poggio, V. Torre, and C. Koch, “Computational vision and regularization theory”, *Nature*, 317(6035), pp. 314–319, 1985.
- [226] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical recipes in C : The art of scientific computing*, Cambridge, UK, Cambridge University Press, 1992.
- [227] *Proceedings of the Fifth IEEE International Conference on Computer Vision*, Boston, MA, June 20–22 1995.
- [228] M. H. Raibert and J. K. Hodgins, “Animation of dynamic legged locomotion”, In Thomas W. Sederberg, editor, Computer Graphics (Proc. SIGGRAPH '91), volume 25, pp. 349–358, July 1991.
- [229] N. Raja and A. Jain, “Recognizing geons from superquadrics fitted to range data”, *Image and Vision Computing*, 10(3), pp. 179–190, 1992.
- [230] R.J. Qian and T.S. Huang, “Motion analysis of articulated objects with applications to human ambulatory patterns”, In *DARPA92*, pp. 549–553, 1992.
- [231] H. Qin and D. Terzopoulos, “D-NURBS: A physics-based framework for geometric design”, *IEEE Transactions on Visualization and Computer Graphics*, 2(1), pp. 85–96, March 1996.
- [232] P. J. Ramadge and W. M. Wonham, “The control of discrete event systems”, Proc. of the IEEE, 77(1), pp. 81–97, January 1989.
- [233] J. S. Rankin, P. A. McHale, C. E. Artentzen, D. Ling, J. C. Greenfield, and R. W. Anderson, “The three-dimensional dynamic geometry of the left ventricle in the conscious dog”, *Circ. Res.*, 39, pp. 304–313, 1976.

- [234] J. M. Rehg and T. Kanade, “Visual tracking of high DOF articulated structures: An application to human hand tracking”, In Jan-Olof Eklundth, editor, *Third European Conference on Computer Vision*, pp. 35–46, Stockholm, Sweden, May 2-6 1994.
- [235] J. R. Rehg and T. Kanade, “Model-based tracking of self-occluding articulated objects”, In Proc. of the Fifth IEEE International Conference on Computer Vision [227], Cambridge, MA, 1995.
- [236] V. C. Rideout, *Mathematical and computer modeling of physiological systems*, Biophysics and Bioengineering Series, ed. A. Noordergraaf, Prentice Hall: Eaglewood Cliffs, New Jersey, 1991.
- [237] M. Rioux and L. Cournoyer, *The NRCC three-dimensional image data files*, Technical Report CNRC No 29077, National Research Council of Canada, 1988.
- [238] M. C. Rivara, Algorithms for refining triangular grids suitable for adaptive and multigrid techniques, International Journal for Numerical Methods in Engineering, 20, pp. 745–756, 1984.
- [239] W. Rogers, E. Shapiro, J. Weiss, M. Buchalter, F. Rademakers, M. Weisfeldt, and E. Zerhouni, “Quantification of and correction for left ventricular systolic long-axis shortening by magnetic resonance tissue tagging and slice isolation”, *Circulation*, 84, pp. 721–731, 1991.
- [240] K. Rohr, “Towards model-based recognition of human movements in image sequences”, *Computer Vision, Graphics, and Image Processing: Image Understanding*, 59(1), pp. 94–115, January 1994.
- [241] I. G. Rosenberg, and F. Stenger, “A lower bound on the angles of triangles constructed by bisecting the longest side”, *Math. Comp.*, 29, pp. 390–395, 1975.
- [242] E. Ruspini, Numerical methods for fuzzy clustering. *Information Science*, (6), pp. 273–284, 1972.
- [243] K. Russell, T. Starner, and A. Pentland, “Unencumbered virtual environments”, In *IJCAI-95 Workshop on Entertainment and AI/Alife*, IEEE Computer Society Press, 1995.
- [244] M. Rutishauser, M. Stricker, and M. Trobina, “Merging range images of arbitrarily shaped objects”, In Proc. CVPR ’94, pp. 573–580, 1994.
- [245] P. Saint-Marc and G. Medioni, “Adaptive smoothing: A general tool for early vision, IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(6), pp. 514–529, 1991.
- [246] R. M. Satava, “Medicine 2001: The king is dead”, in *Medicine Meets Virtual Reality III: Interactive Technology and the New Paradigm for Healthcare*, RM Satava, K Morgan, HB Sieburg, et al eds., IOS Press: Amsterdam, pp. 334–339, 1995.

- [247] B. Schachter, "Long crested wave models", Computer Graphics and Image Processing, 12, (February 1980) , pp. 187–201, 1980.
- [248] H. A. Schwid, A flight simulator for general anesthesia training. *Computers and Biomedical Research*, 20, pp. 64-75, 1987.
- [249] S. Sclaroff and A. Pentland, "Generalized implicit functions for computer graphics", Computer Graphics (Proc. SIGGRAPH), 25(4), pp. 247–250, 1991.
- [250] S. Sclaroff and A. Pentland, "On modal modeling for medical images: Underconstrained shape description and data compression", Proc. of IEEE Workshop on Biomedical Image Analysis, Seattle, WA, pp. 70-79, 1994.
- [251] F. Schmitt, B. Barsky and W-H Du, "An adaptive subdivision method for surface-fitting from sampled data", Proc. Siggraph'86, pp. 179–188, Dallas, TX, August 1986.
- [252] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric primitives", Computer Graphics (Proc. SIGGRAPH), 20(4), pp. 151–160, 1986.
- [253] A.A. Shabana, *Dynamics of multibody systems*, Wiley, New York, 1989.
- [254] J. Rismantab-Sany and A. A. Shabana, "On the numerical solution of differential/algebraic equations of motion of deformable mechanical systems with nonholonomic constraints", Computers and Structures, 33(4), pp. 1017-1029, 1989.
- [255] A. A. Shabana, "Constrained motion of deformable bodies", International Journal for Numerical Methods in Engineering, 32, pp. 1813-1831, 1991.
- [256] P. Shi, A. Amini, G. Robinson, A. Sinusas, C. T. Constable, and J. Duncan, "Shape-based 4D left ventricular myocardial function analysis", Proc. of IEEE Workshop on Biomedical Image Analysis, Seattle, WA, pp. 88-97, 1994.
- [257] P. Shi, G. Robinson, T. Constable, A. Sinusas, and J. Duncan, "A model-based integrated approach to track myocardial deformation using displacement and velocity constraints", Proc. of IEEE Fifth International Conference on Computer Vision, Cambridge, Massachusetts, pp. 687-692, 1995.
- [258] K. Shoemake, "Animating rotations with quaternion curves", Computer Graphics (Proc. SIGGRAPH), 19(3), pp. 245–254, 1985.
- [259] K. Siddiqi, K. Tresness, and B. B. Kimia, "Parts of visual form: Ecological and psychological aspects", Technical Report LEMS-104, Brown University, February 1994.
- [260] E. Simoncelli, "Design of multi-dimensional derivative filters", In *ICIP '94*, 1994.

- [261] E. Simoncelli, E. Adelson, and D. Heeger, “Probability distributions of optical flow”, In Proc. CVPR ’91, pp. 310–315, 1991.
- [262] K. Sims, “Evolving virtual creatures”, In Proc. of SIGGRAPH ’94, pp. 24–29, Orlando, Florida, July 1994.
- [263] 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), pp. 131–146, 1990.
- [264] D.P. Stoten and H. Benchoubane, “Empirical studies of an MRAC algorithm with minimal control synthesis”, *International Journal of Control*, 51(4), pp. 823–849, 1990.
- [265] D.P. Stoten and H. Benchoubane, “The decentralized minimal control synthesis algorithm”, *International Journal of Control*, 56(4), pp. 967–983, 1992.
- [266] M. Stynes, “On fast convergence of the bisection method for all triangles”, *Math. Comp.*, 35, pp. 1195–1201, 1980.
- [267] R. Szeliski, *Bayesian modeling of uncertainty in low-level vision*, Kluwer Academic Publishers, Boston, MA, 1989.
- [268] R. Szeliski and D. Terzopoulos, “Physically-based and probabilistic modeling for computer vision”, In *SPIE Vol. 1570 Geometric Methods in Computer Vision*, pp. 140–152, San Diego, July 1991. Society of Photo-Optical Instrumentation Engineers.
- [269] R. Szeliski, D. Tonnesen, and D. Terzopoulos, “Modeling surfaces of arbitrary topology with dynamic particles”, In Proc. CVPR ’93, pp. 82–87, 1993.
- [270] H. Tanaka and F. Kishino, “Adaptive mesh generation for surface reconstruction: Parallel hierarchical triangulation without discontinuities”, Proc. CVPR’93, pp. 88–94, NY, 1993.
- [271] R. Tennant and C. J. Wiggers, “The effect of coronary occlusion on myocardial contraction”, *Am J Physiol*, 112, pp. 351–361, 1935.
- [272] D. Terzopoulos, “Regularization of inverse visual problems involving discontinuities”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(4), pp. 413–424, 1986.
- [273] D. Terzopoulos, A. Witkin, and M. Kass, “Symmetry-seeking models and 3D object reconstruction”, *International Journal of Computer Vision*, 1(3), pp. 211–221, 1987.
- [274] D. Terzopoulos, “The computation of visible-surface representations”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(4), pp. 417–438, 1988.

- [275] D. Terzopoulos, A. Witkin, and M. Kass, “Constraints on deformable models: Recovering 3D shape and nonrigid motion”, *Artificial Intelligence*, 36(1), pp. 91–123, 1988.
- [276] D. Terzopoulos and A. Witkin, “Physically based models with rigid and deformable components”, *IEEE Computer Graphics and Applications*, 8(6), pp. 41–51, 1988.
- [277] D. Terzopoulos and K. Fleischer, “Deformable models”, *The Visual Computer*, 4(6), pp. 306–331, 1988.
- [278] D. Terzopoulos and D. Metaxas, “Dynamic 3D models with local and global deformations: Deformable superquadrics”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7), pp. 703–714, 1991. See also Proc. IEEE Third International Conference on Computer Vision (ICCV’90), Osaka, Japan, Dec. 1990, pp. 606–615.
- [279] D. Terzopoulos and K. Waters, “Analysis and synthesis of facial image sequences using physical and anatomical models”, *IEEE Pattern Analysis and Machine Intelligence*, 15(6), pp. 569–579, 1993.
- [280] P. Ts’o, and B. Barsky, “Modeling and rendering waves: Wave-Tracing using Beta-Splines and Reflective and Refractive Texture Mapping”, *ACM Transactions on Graphics*, 6(3), pp. 191–214, 1987.
- [281] J. K. Tsotsos, “Knowledge organization and its role in representation and interpretation for time-varying data: The ALVEN system”, in M. Fischler and O. Firschein, eds. *In Readings in Computer Vision*, Morgan Kaufmann, pp. 498–514, 1989.
- [282] S. Upstill, *The RenderMan companion*, Addison Wesley, New York, 1990.
- [283] M. Vasilescu, and D. Terzopoulos, “Adaptive meshes and shells: Irregular triangulation, discontinuities, and hierarchical subdivision”, Proc. IEEE Computer Vision and Pattern Recognition Conference (CVPR’92), pp. 829–832, Champaign, Illinois, June 1992.
- [284] B.C. Vemuri and A. Radisavljevic, “From global to local, a continuum of shape models with fractal priors”, Proc. CVPR’93, pp. 307–313, NY, 1993.
- [285] B. C. Vemuri and A. Radisavljevic, “Multiresolution stochastic hybrid shape models with fractal priors”, *ACM Transactions on Graphics*, 13(2), pp. 177–207, 1994.
- [286] F. L. Villarreal, L. K. Waldman, and W. Y. W. Lew, “A technique for measuring regional two-dimensional finite strains in canine left ventricle”, *Circ Res* 62, pp. 711–721, 1988.
- [287] L. K. Waldman, Y. C. Fung, and J. W. Covell, “Transmural myocardial deformation in the canine left ventricle: Normal in vivo three-dimensional finite strains”, *Circ Res*, 57, pp. 152–163, 1985.

- [288] M. Watt, "Light-water interaction using backward beam tracing", Proc. of SIGGRAPH '90, in Computer Graphics 24, pp. 377–385, 1990.
- [289] J. A. Webb and J. K. Aggarwal, "Structure from motion of rigid and joined objects", In Proc. of the International Joint Conference on Artificial Intelligence, pp. 686–691, 1981.
- [290] R. A. Wehage and E. J. Haug, "Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems", ASME J. Mechanical Design, 104, pp. 247-255, 1981.
- [291] A. Witkin, K. Fleischer, and A. Barr, "Energy constraints on parameterized models",, *Computer Graphics (Proc. SIGGRAPH)*, 21(4), pp. 225–232, 1987.
- [292] A. Witkin and M. Kass, "Spacetime constraints", *Computer Graphics (Proc. SIGGRAPH '88)*, 22, pp. 159–168, August 1988.
- [293] A. Witkin and W. Welch, "Fast animation and control of nonrigid structures", *Computer Graphics (Proc. SIGGRAPH)*, 24(4), pp. 243–252, 1990.
- [294] J. Wittenberg, *Dynamics of systems of rigid bodies*, Tubner, Stuttgart, 1977.
- [295] Y. Yacoob and L.S. Davis, "Computing spatio-temporal representations of human faces", In Proc. CVPR '94, pp. 70–75, 1994.
- [296] M. Yamamoto and K. Koshikawa, "Human motion analysis based on a robot arm model", In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 664–665, Hawaii, June 3–6 1991.
- [297] A. Young, "Epicardial deformation from coronary cineangiograms", In L. Glass, P. Hunter, A. McCulloch, eds., *Theory of Heart: biomechanics, biophysics, and nonlinear dynamics of cardiac function*, New York, Springer-Verlag, pp. 175–207, 1991.
- [298] A. A. Young, P. J. Hunter, and B. H. Smaill, "Estimation of epicardial strain using the motions of coronary bifurcations in biplane cineangiography", *IEEE Trans Biomed Eng*, 39, pp. 526-531, 1992.
- [299] A. A. Young and L. Axel, "Three-dimensional motion and deformation of the heart wall", *Radiology*, 185, pp. 241-247, 1992.
- [300] A. A. Young, L. Axel, L. Dougherty, D. K. Bogen, and C. S. Parenteau, "Validation of tagging with MR imaging to estimate material deformation", *Radiology*, 188, pp. 101-108, 1993.
- [301] A. A. Young, D. L. Kraitchman, and L. Axel, "Deformable models for tagged MR Images: reconstruction of two- and three-dimensional heart wall motion", Proc. of IEEE Workshop on Biomedical Image Analysis, pp. 317-323, Seattle, WA, 1994.

- [302] A. A. Young, C. M. Kramer, V. A. Ferrari, L. Axel, and N. Reichek, "Three-dimensional left ventricular deformation in hypertrophic cardiomyopathy", *Circulation*, 90(2), pp. 854-867, 1994.
- [303] A.L. Yuille, D.S. Cohen, and P. Hallinan, "Feature extraction from faces using deformable templates", *International Journal of Computer Vision*, 8, pp. 104-109, 1992.
- [304] A. A. Young, D. L. Kraitchman, L. Dougherty, and L. Axel, "Tracking and finite element analysis of stripe deformation in magnetic resonance tagging", *IEEE Trans. on Medical Imaging*, 14(3), pp. 413-421, 1995.
- [305] E. A. Zerhouni, D. M. Parish, W. J. Rogers, and A. Yang, "Human heart: Tagging with MR imaging - a method for noninvasive assessment of myocardial motion", *Radiology*, 169, pp. 59-63, 1988.
- [306] O. Zienkiewicz, *The Finite element method*, McGraw-Hill, 1977.

INDEX

- Acceleration, 31
ACRONYM, 18
Active model nodes, 142
Adaptive control, 15, 244
Adaptive Euler integration, 66
Adaptive finite elements, 95, 97
Adaptive Runge-Kutta integration, 66
Adaptive subdivision, 96, 98
Anatomical modeling, 219, 222
Anatomy, 220
Angular velocity, 31
Animation, 253
Applied forces, 53
Articulated objects, 15, 233
Aspect graph representation, 127
Axial blending, 113
Basis functions, 25
Basis matrix, 39
Baumgarte stabilization, 77
Bending rigidity, 34
Bending, 22–23, 95, 131–132, 151
Bisection operation, 99
Blended shapes, 113
Blending function, 113
Blending regions, 113
Blending, 111
Boundary conditions, 259
Buoyancy, 265
 C^0 elements, 42
 C^1 triangular elements, 47
Centrifugal forces, 31–32
Collisions, 60, 235, 239
Complete polynomial, 40
Composed model, 153
Composition function, 154
Computational fluid dynamics, 254
Computer graphics, 11, 71, 84, 97, 178, 268, 274
Computer vision, 4, 67, 90, 96, 144, 178, 273
Conforming operation, 100
Conformity, 103
Constrained motion, 77
Constraint forces, 82
Constraint methods, 77
Controlling fluids, 254, 267
Convergence, 40
Coriolis accelerations, 31
Coriolis forces, 32
CT data, 193
Cue integration, 273
Curvature, 99, 195
Cyberware digitizer, 104
Damping density, 32
Damping matrix, 30, 67
Data fitting, 36
Deformable face mask, 274
Deformable model fitting, 67
Deformable phantoms, 196
Deformable superquadric, 92, 95, 131
Deformation strain energy, 30
Degrees of freedom, 27
Discrete event systems, 168
Discretization errors, 40
Displacement, 25
Dynamic constraints, 11
Dynamic impact, 241
Dynamics, 27, 199
Efficient point-to-point constraints, 83
Elastic properties, 33
Element strain vector, 50
Element stress vector, 50
Error covariance matrix, 87
Euler angle, 28
Euler integration, 65
Euler step, 72
Exhalation, 221
Extended Kalman filter, 88
External forces, 35

- Facial motion estimation, 273
- Facial shape estimation, 273
- Fast constraint force computation, 80
- Featherstone, 235
- Finite element implementation, 39
- Finite element method, 25
- Finite element subdivision, 98
- Finite elements, 14, 27, 191, 193–194
- Fluid animations, 268
- Fluid tracking, 261
- Force based estimation, 88
- Force computation, 156
- Force-based estimation, 63
- Forces, 27, 53, 200
- Frictional force, 61
- Fuzzy clustering, 156
- Galerkin method, 25
- Gauss-Legendre quadrature rules, 43
- Generalized coordinates, 27–28, 118
- Generalized cylinders, 18
- Generalized force, 89
- Generalized inertial forces, 32
- Generalized mass matrix, 32
- Geometry of deformable models, 17
- Gimbal lock, 28
- Global deformations, 17, 20, 95
- Global parameterization, 112
- Global strain energy, 33
- Heart analysis, 15, 191
- Heart motion analysis, 208
- Height field, 264
- Hermite polynomial, 117
- Hierarchical aspect representation, 125
- Holes and indentations, 116
- Holonomic constraints, 14, 77
- Hookean energy, 33
- Human body part acquisition, 169, 171
- Human body tracking, 175
- Hybrid deformable models, 17, 20
- Hybrid models, 97
- Image data, 67, 132
- Implantation methods, 194
- Incomplete data, 112
- Inertia, 36
- Inhalation, 221
- Instant rotation center, 167
- Integration of qualitative and PBM methods, 125
- Intensity gradients, 53
- Isoparametric formulation, 42
- Jacobian matrix, 21, 42
- Jacobian, 28
- Kalman filter implementation, 89
- Kalman filter, 15, 87
- Kalman gain, 88
- Kinematics, 27
- Kinetic energy, 31
- Lagrange equations, 30, 48, 87
- Lagrange multipliers, 11, 77, 233
- Lagrangian dynamics, 27
- Left ventricular wall motion, 191
- Liquids, 12, 253
- Ljapunov stability, 79
- Loaded membrane energy, 49
- Loaded membrane, 34, 47, 67
- Local deformations, 20, 95
- Local strain energy, 34
- Long range forces, 56
- Magnetic tagging, 9, 191
- Marker particles, 261
- Mass density, 31
- Mass matrix, 30
- Matrix of central moments, 66
- Medical imaging, 9, 211, 229, 275
- Membrane energy, 34
- Modal analysis, 18
- Modal deformations, 193
- Model discretization, 133
- Model evolution, 111, 118
- Model geometry, 28
- Model implementation, 65
- Model kinematics, 28
- Model tessellation, 40
- Modeling errors, 40
- Motion estimation, 27, 87, 138
- MRI data, 193
- MRI-SPAMM, 191, 196
- Multi-level shape representation, 95

- Multi-view integration, 135
Multibody objects, 77, 82
Multiple constraints, 82
Navier-Stokes equations, 16, 253,
 255
Noise, 112
Non-degeneracy, 103
Non-invasive imaging, 191
Nonholonomic constraint, 273
Normal breathing, 219, 225, 229
NRCC database, 70
Object recognition, 17, 112
Object-centered models, 127
Occlusion, 36
Optical flow, 273
Order of the motion equations, 36
Orthogonal matrix, 29
Parameter functions, 191, 198, 204
Parameter graphs, 213
Parameterized global
 deformations, 20
Parameterized primitives, 20
Parametric composition, 153
Part segmentation, 158
Perspective projection, 132
Physiological modeling, 219, 223
Physiology modeling, 11
Physiology, 220
Pneumothorax, 219, 226, 229
Point-to-point constraints, 82
Potential functions, 53
Prediction, 143
Proportional Derivative control,
 236
Quadrilateral elements, 42
Qualitative shape estimation, 6,
 127
Qualitative shape recovery, 128
Quantitative shape recovery, 6
Quaternion, 28, 66
Radau quadrature rules, 44
Radial forces, 59
Radiopaque markers, 194
Rayleigh dissipation functional, 32
Range data, 70, 104, 120
Rayleigh-Ritz method, 25
Recursive dynamics, 12, 233
Recursive estimation, 88
Reference shape, 20
Respiratory mechanics, 11, 219,
 224
Riccati equation, 89
Rotation matrix, 20
Rotation, 31
Runge-Kutta integration, 66
Shape accuracy, 112
Shape blending, 15
Shape coverage, 111–112
Shape estimation, 4, 9, 36, 87, 111,
 120
Shape functions, 39, 42, 59, 95
Shearing, 23
Short range forces, 53
Singularity, 28
Smoothness, 103
Solid modeling, 18
Sparse data, 112
Spline models, 17
Spring-mass systems, 193
Stability analysis, 121
Stabilization factors, 79
Stabilized constraints, 79
State estimation, 88
State variables, 87
Stiffness matrix, 30, 33, 49–50
Strain energy, 33
Strain vector, 50
Stress vector, 50
Subdivision algorithm, 102
Suboptimal filter design, 89
Superquadric, 18, 21, 67
Surface models, 192
Surgical training, 219
Symbolic representation, 111
System model, 87
Tapering, 22–23, 95, 131, 153
Temporal correspondence, 191
Tensions, 34
Thin plate under tension, 34, 51
Topologically adaptive models,
 111, 161
Tracking, 7, 138, 143
Triangular elements, 42, 97
Twisting, 23, 192
Ultrasonic crystals, 194

- Uncorrelated measurement noise,
 - 88
- Uncorrelated system noise, 88
- Viewer-centered models, 127
- Virtual work, 32, 35
- Volumetric models, 192–193, 198
- Voxel representation, 195
- WATSMART system, 90
- Weighting functions, 34