PHYSICS-BASED MODELING OF NONRIGID OBJECTS FOR VISION AND GRAPHICS

Dimitri N. Metaxas

A thesis submitted in conformity with the requirements
for the Degree of Doctor of Philosophy,
Graduate Department of Computer Science, in the
University of Toronto

PHYSICS-BASED MODELING OF NONRIGID OBJECTS FOR VISION AND GRAPHICS

Ph.D. 1992

Dimitri N. Metaxas

Graduate Department of Computer Science

University of Toronto

Abstract

This thesis develops a physics-based framework for 3D shape and nonrigid motion modeling for computer vision and computer graphics. In computer vision it addresses the problems of complex 3D shape representation, shape reconstruction, quantitative model extraction from biomedical data for analysis and visualization, shape estimation, and motion tracking. In computer graphics it demonstrates the generative power of our framework to synthesize constrained shapes, nonrigid object motions and object interactions for the purposes of computer animation.

Our framework is based on the use of a new class of dynamically deformable primitives which allow the combination of global and local deformations. It incorporates physical constraints to compose articulated models from deformable primitives and provides force-based techniques for fitting such models to sparse, noise-corrupted 2D and 3D visual data. The framework leads to shape and nonrigid motion estimators that exploit dynamically deformable models to track moving 3D objects from time-varying observations.

We develop models with global deformation parameters which represent the salient shape features of natural parts, and local deformation parameters which capture shape details. In the context of computer graphics, these models represent the physics-based marriage of the parameterized and free-form modeling

paradigms. An important benefit of their 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 Lagrange equations of motion that govern our models, augmented by constraints, make them responsive to externally applied forces derived from input data or applied by the user. This system of differential equations is discretized using finite element methods and simulated through time using standard numerical techniques. We employ these equations 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 adjust the translational, rotational, and deformational degrees of freedom such that the model evolves in a consistent fashion with the noisy data.

We demonstrate the interactive time performance of our techniques in a series of experiments in computer vision, graphics, and visualization.

Dedicated to my parents
Nikolas and Ada

Acknowledgements

I would like to thank my thesis supervisor, Demetri Terzopoulos, for his constant support and encouragement throughout this research. He was the best possible supervisor for this work, as he has a unique blend of technical skills, experience, and the imagination needed to pioneer and supervise research bridging two fields. In addition to being a true scholar, he is also a very approachable and friendly person.

I am grateful to my external committee member Professor Takeo Kanade of Carnegie Mellon University for his steadfast enthusiasm for the thesis, and for reading it meticulously. The other members of my committee, Professors Ted Davisson, Wayne Enright, Eugene Fiume, Evangelos Milios, and John Tsotsos deserve commendation for their patience with the many drafts they perused conscientiously. Their comments improved the clarity of the thesis. I also thank Professor Ahmed A. Shabana of the University of Illinois at Chicago who provided valuable feedback and advice on multibody dynamics and Dr. Richard Szeliski of Digital Equipment Corporation for important discussions on Kalman filtering. Furthermore, I would like to thank Professors Alain Fournier of the University of British Columbia and Tom Huang of the University of Illinois at Urbana-Champaign for their support and advice during this work.

I wish to thank Professor Susan Tupling for providing valuable data for my experiments by granting me access to the WATSMART system at the Biomechanics Lab. I thank Michael McCool and the others at the DGP Lab for helping me create high quality video of my computer animations.

Amongst my colleagues and friends, (now Professor) Niels da Vitoria Lobo stands out for his steadfast support of my ambitions and for his academic maturity and deep understanding of the intricacies of the world today. Additional acknowledgements go out to other people around the lab and the department, Sean Culhane, Jeremy Cooperstock, Isabel Cruz, Sven Dickinson, Gregory Dudek, Vincent Gogan, Tim McInerney, Robert Majka, Theodore Norvell, David Tonnesen, Luiz Velho, Gilbert Verghese and Dave Wilkes, with all of whom I developed warm friendships.

For financial support during my PhD, I thank the University of Toronto Open Fellowships office and the ITRC.

Finally, I thank my parents Nikolas and Ada Metaxas for all their love and encouragement throughout my studies and, most importantly for giving my sister Greta and me a thirst for knowledge and a desire for intellectual achievement.

Contents

List of Figures

Nomenclature

u	material coordinates
$\mathbf{x}$	position of a point on a deformable model wrt to an inertial frame Φ
$\mathbf{p}$	position of a point on the deformable model wrt to a model frame ϕ
$\mathbf{s}$	reference shape of deformable model
$\mathbf{d}$	local deformation of deformable model
$\mathbf{e}$	parameterized geometric primitive
$\mathbf{T}$	function defining parameterized global deformations
$\mathbf{q}_s$	vector of global deformation parameters
$\mathbf{J}$	Jacobian matrix wrt to global deformations
$\mathbf{S}$	basis matrix
$\mathbf{q}_d$	vector of local deformation parameters
$\mathbf{R}$	rotation matrix
$\mathbf{c}$	translation of model frame wrt to inertial frame
$\mathbf{q}_c$	vector of translation degrees of freedom
$\mathbf{q}_\theta$	vector of rotation degrees of freedom
$\mathbf{M}$	mass matrix
$\mathbf{D}$	stiffness matrix
$\mathbf{K}$	damping matrix
$\mathbf{g}_q$	generalized inertial forces
$\mathbf{f}_q$	generalized external forces
$\mathbf{q}$	vector of model generalized degrees of freedom
$\boldsymbol{\omega}$	model angular velocity
$\mathbf{L}$	Jacobian matrix wrt to model degrees of freedom
$\boldsymbol{\sigma}$	stress vector
$\boldsymbol{\epsilon}$	strain vector

N_i	finite element shape function
P	potential function
$\mathbf{f}$	external force
a	elasticity constant
$\mathbf{n}$	unit normal vector at given point of a surface
β_f	friction coefficient
$\mathbf{v}$	nodal velocity
α	damping coefficient
$\mathbf{M}$	matrix of central moments
ω_i	model stiffness parameters
$\mathbf{C}$	vector of point-to-point constraints
$\mathbf{f}_{g_c}$	generalized constraint forces
Δt	integration step
$\mathbf{w}$	uncorrelated modeling error noise
$\boldsymbol{v}$	uncorrelated measurement error noise
$\mathbf{P}$	error covariance matrix
$\mathbf{V}$	modeling error covariance matrix
$\mathbf{Q}$	measurement error covariance matrix

Chapter 1

Introduction

1.1 Problem Statement

This thesis develops a physics-based framework for 3D shape and nonrigid motion modeling for computer vision and computer graphics. The framework addresses a variety of difficult modeling problems common to both fields.

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 accurately[1] in terms of simple shape primitives and they may undergo motions that are nonrigid and 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.

The computer graphics literature, on the other hand, is replete with mathematical representations of solid objects. In particular, the field of solid modeling has

[1]The 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.

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.

This thesis proposes a new 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 our modeling primitives make them responsive to externally applied forces. External forces may be derived from input data, may arise from interactions 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 through time using standard numerical techniques.

Our physics-based 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.

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]

The thesis demonstrates the utility of our framework in both vision and graphics applications. The vision applications include shape reconstruction, quantitative model extraction from biomedical data for analysis and visualization, shape estimation, and motion tracking. The graphics applications exploit the generative power of our framework to synthesize constrained shapes, nonrigid object motions, and object interactions for the purposes of computer animation.

1.2 Illustrative Modeling Problems and Examples

We will now illustrate the spectrum of modeling problems that fall within the scope of this thesis with a brief series of examples.

The first example illustrates the application of our dynamic models to interactive shape design. Fig. 1.1 shows a snapshot of an interactive 3D world inhabited

[2] A 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. This usually means not less than 3 frames per second.

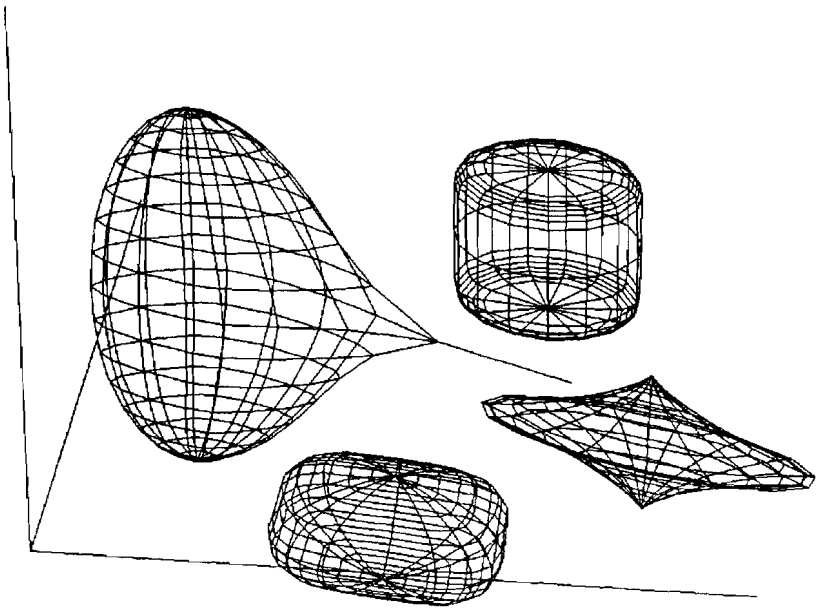


Figure 1.1: Interactions with deformable models.

by deformable models that we dub *deformable superquadrics*. The figure illustrates deformable superquadric shells covered by rectangular and triangular finite elements. Through mouse control, the user can initialize models, change their global deformation parameters, apply forces to them, and vary the viewpoint. The figure illustrates four deformable models with different settings for the global deformation parameters, resulting in qualitatively different shapes. The model at the left is being pulled by a stretchy spring (displayed as a line) activated and dragged using the mouse. The applied spring force causes local (free-form) and global (parameterized) deformations in the model. Thus, useful shapes may be designed by applying forces.

A fundamental problem in computer vision is the reconstruction of quantitative representations of 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 models often provide the additional constraints needed to produce a reasonable reconstruction, while they provide the ability to conform to nontrivial shapes. For example, Fig. 1.2 illustrates the fit-

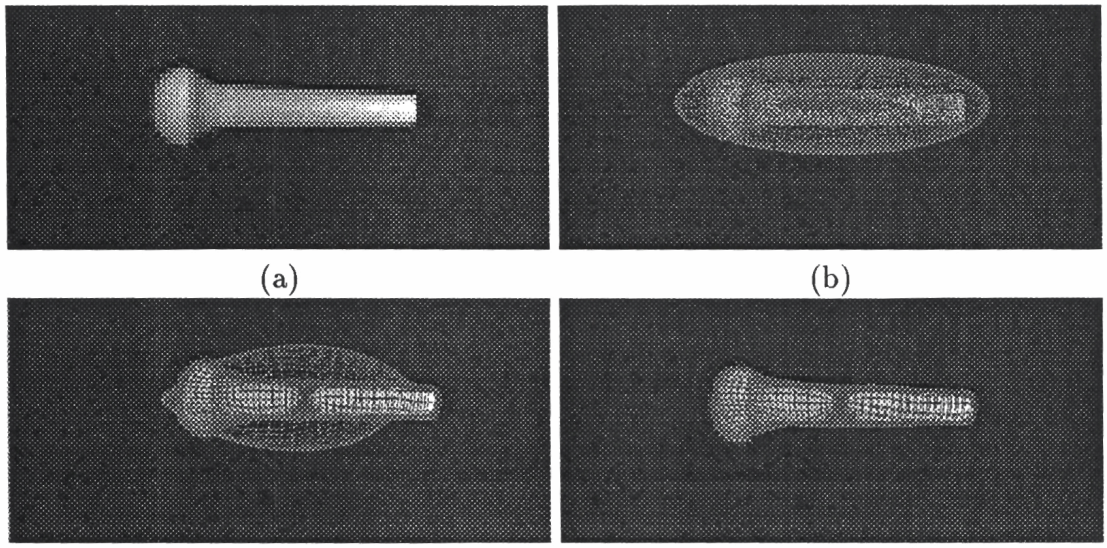


Figure 1.2: Fitting a deformable superquadric (b), (c), (d) to pestle image (a).

ting of a deformable superquadric to a monocular image of a pestle Fig. 1.2(a). As we explain in a subsequent chapter, the image is converted into a force field that acts on the model, deforming it such that it becomes consistent with the occluding boundary of the pestle in the image. Fig. 1.2(b) shows the initial state of the deformable superquadric displayed in wireframe projected onto the image. Fig. 1.2(c) shows an intermediate step in the fitting process as the image forces are deforming the model and Fig. 1.2(d) shows the final reconstructed model.

With the advent of low cost, real time visual sensors and image processing hardware, dynamic object representation and tracking tasks are gaining a significant presense in the vision literature. An interesting problem is to infer the shapes and motions of complex single- or multi-part deformable objects from sparse, noisy 3D observations. Such estimation problems are particularly challenging when they must be solved on-line, and Kalman filtering has become a popular approach for accomplishing this. Fig. 1.3 illustrates a new recursive shape and motion estimator which is developed in this thesis. The estimator incorporates constrained deformable superquadrics as Kalman filter system models. The figure illustrates a model composed of 5 connected deformable superquadrics. The estimator is applied to biomechanical data collected by 3D position sensors applied to the arms of

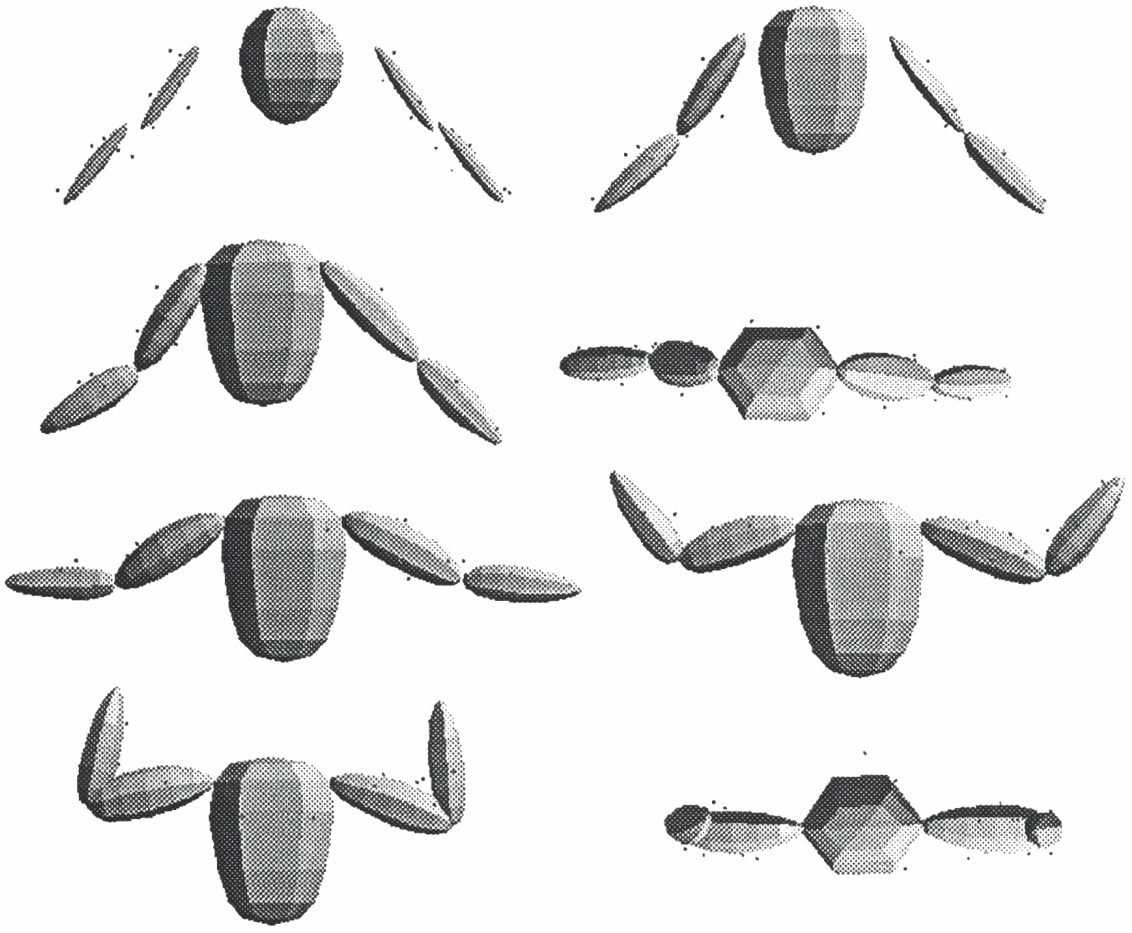


Figure 1.3: Tracking of raising and flexing human arm motion.

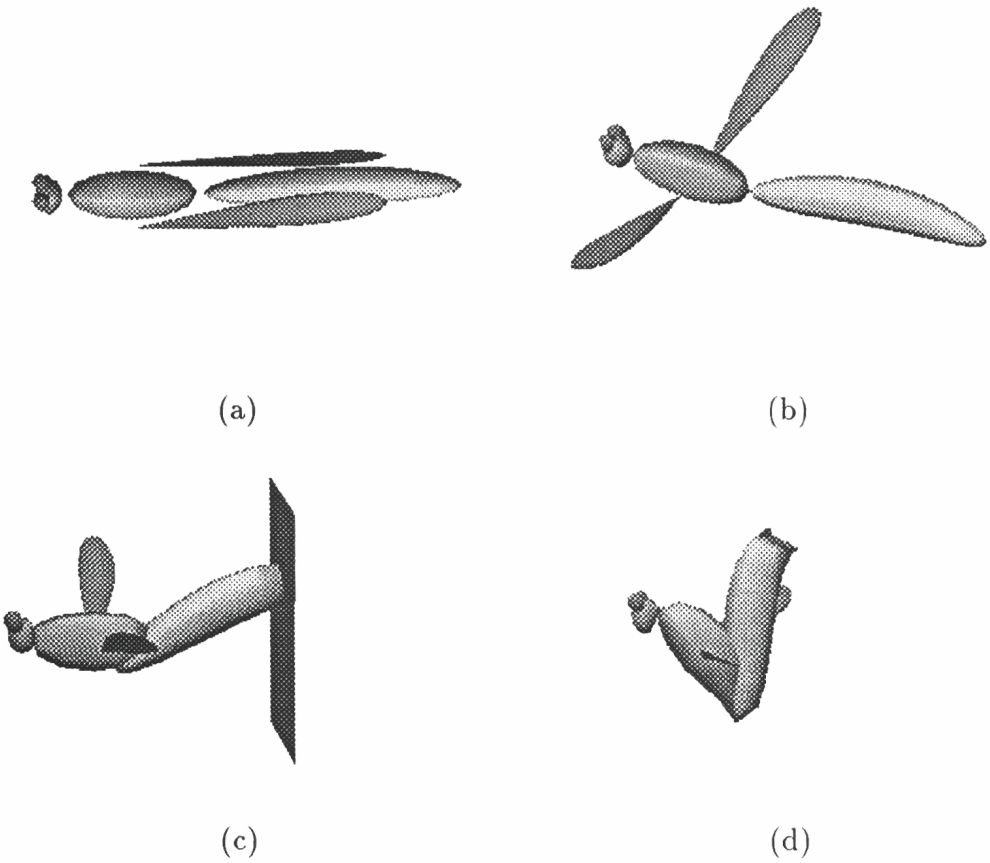


Figure 1.4: Self-assembly (a), articulation (b), and swatting (c), (d) of a dragonfly.

a human subject. Fig. 1.3(a) shows a view of the 3D data and the initial models. Fig. 1.3(b) shows an intermediate step of the fitting process driven by data forces from the first frame of the data sequence, while Figs. 1.3(c) and (d) show different views of the models fitted to the initial data. Figs. 1.3(e) and (f) show intermediate frames of the models tracking the nonrigid motion of the subject's flexing arms, while Figs. 1.3(g) and (h) show two views of the final position of the models.

The deformable modeling primitives and the constraint methods that we develop in this thesis, 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. 1.4, which shows the automatic construction of a minimalist dragonfly from its constituent deformable superquadric parts. Fig. 1.4(a) shows the disjoint parts in their 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. 1.4(b). In Fig. 1.4(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 final shape of the dragonfly is shown in Fig. 1.4(d).

1.3 Contributions

This thesis develops a framework that includes

1. a new class of dynamic deformable primitives which combine global and local deformation parameters,
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,
3. the development of physics-based constraints between these deformable primitives that may be used to synthesize complex articulated models,
4. a recursive technique for estimating shape and nonrigid motion from noise corrupted data based on applying Kalman filtering theory to our dynamic models,
5. new applications to visual estimation and graphics animation.

In the sequel, we elaborate on each of the above technical contributions, which have also been reported in [67, 38, 39, 40, 41, 42, 43, 44, 45, 46].

1.3.1 New Deformable Primitives

We create a new family of modeling primitives by developing a mathematical approach that allows the combination of global and local deformations. Our primitives include global deformation parameters which represent the salient shape

features of natural parts and local deformation parameters which capture shape details. More specifically, we develop hybrid models whose underlying geometric structure allows the combination of parametric models (superquadrics, spheres, cylinders), parameterized global deformations (bends, tapers, twists, shears, etc.) and local spline free-form deformations. In this way, the descriptive power of our models is a superset of the descriptive power of locally deformable models [65, 53], and globally deformable models [52, 72]. In the context of computer graphics, we incorporate characteristics of the parameterized and free-form modeling paradigms within a single physics-based model. An important benefit of the global/local descriptive power of these models in the context of computer vision is that it can potentially satisfy the often conflicting requirements of shape reconstruction and shape recognition. The local degrees of freedom of deformable models allow the reconstruction of fine scale structure and the natural irregularities of real world data, while the global degrees of freedom capture the salient features of shape that are innate to natural parts and appropriate for matching against object prototypes.

1.3.2 Systematic Formulation of Dynamic Primitives

Through the application of Lagrangian mechanics, we develop a method to systematically convert the geometric parameters of the solid primitive, the global (parameterized) and local (free-form) deformation parameters, and the six degrees of freedom of rigid-body motion into generalized coordinates or dynamic degrees of freedom. More precisely, our method applies generally across all well-posed geometric primitives and deformations, so long as their equations are differentiable. The distinguishing feature of our approach is that it combines the parameterized and free-form modeling paradigms within a single physical model. Thus our models exhibit correct mechanical behaviors and their various geometric parameters assume well-defined physical meanings in relation to prescribed mass distributions, elasticities, and energy dissipation rates. Furthermore, motivated by the requirements of real time vision and graphics applications we appropriately simplify the models and we use simple numerical integration techniques to achieve real time or

near real time simulation rates on available graphics workstations. [3]

1.3.3 Physics-based Constraints

To deal with constrained multipart objects such as articulated anthropomorphic bodies, we develop an efficient technique to implement hard point-to-point constraints between deformable primitives. These constraints are never violated, regardless of the magnitude of the forces experienced by the parts. Attempting to approximate such constraints with simple, stiff springs leads to instability. In our approach, we compute the constraint forces using a stabilized Lagrange multiplier technique [7]. Furthermore, we develop constraint techniques to synthesize primitive interactions in simulated physical environments. In particular, using the elastic properties of our models we calculate forces stemming from collisions and friction against impenetrable solid or deformable surfaces.

1.3.4 Recursive Estimation

We also exploit the constrained nonrigid motion synthesis capabilities of our models in order to estimate shape and motion from incomplete, noisy observations available sequentially over time.[4] Applying continuous nonlinear Kalman filtering theory, we construct a powerful new recursive estimator which employs the Lagrange equations of 3D nonrigid motion as a system model. 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. A Riccati procedure updates an error covariance matrix which transforms the forces in accordance with the system dynamics and the prior observation history. The transformed forces induce

[3] In many real-time applications the time required to arrive at a reasonable solution is often more critical than the accuracy of the solution.

[4] These sorts of inverse problems have been studied widely in the mathematics literature, where emphasis is placed on the development of sufficient conditions to guarantee that the problem is well-posed and a unique solution exists. We will not concern ourselves with existence and uniqueness in this thesis. The data sets that we will employ in practice permit our estimation techniques to converge to reasonable solutions.

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.[5] Furthermore, we show that the force-based tracking scheme proposed in [33, 70] and applied to deformable superquadrics in [67] amounts to a degenerate "Kalman filter" with a constant, unit error covariance matrix.

1.3.5 Applications to Vision and Graphics

We demonstrate the usefulness of the framework in several application areas in vision and graphics. These include quantitative 3D shape reconstruction from static monocular images and model fitting and tracking of biomechanics data for analysis and visualization. Our modeling framework also provides the necessary generative power to synthesize constrained shapes and nonrigid motions for the purposes of computer animation.

1.4 Thesis Outline

Chapter 2 summarizes the physics-based approaches in computer vision and computer graphics that are relevant to our work and compares them to our framework.
Chapter 3 presents the geometric formulation of our models. We describe a general technique for defining global and local geometric degrees of freedom and apply it to create a new class of deformable models that we dub *deformable superquadrics.*
Chapter 4 develops the kinematic and dynamic formulation of our 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 5 gives an overview of the finite element method that we employ to

[5] Our algorithms are designed to perform numerical integration steps perpetually as new data arrive. This allows them to react continuously to the incoming data. The delay between successive data inputs should be long enough, however, so that the integration method has time to achieve a steady state estimate for each data frame.

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 and graphics applications.

Chapter 6 presents techniques for converting visual data into forces that can be applied to deformable models in data fitting scenarios. Secondly, for 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 7 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 8 presents a technique to implement hard point-to-point constraints between deformable part models which should not be violated, regardless of the magnitude of the forces experienced by 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 9 presents computer vision and computer graphics experiments involving physics-based constraints.

Chapter 10 applies continuous nonlinear Kalman filtering theory to construct a recursive estimator which employs the Lagrange equations of 3D nonrigid motion that we have developed 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 11 draws conclusions from our work and gives a perspective of future research.

Chapter 2

Related Prior Work

This chapter discusses related prior work in both computer vision and computer graphics. In the context of computer vision we will compare our models to other related models as well as previous recursive estimation techniques for shape and motion estimation. In the context of computer graphics we will compare our framework with other physics-based approaches in terms of modeling power, generality, constraint formulation, and interactions with the physical simulated physical environments.[1]

2.1 Computer Vision

2.1.1 Local, Global, and Physics-Based Models

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 [9][62][65]. Generalized spline techniques underlie the notion of regularization and its application to a variety of reconstruction problems in early vision [55][64]. The many degrees of freedom and local deformation properties of gener-

[1]There is a close relationship between our models and other approaches used for rigid articulated bodies and other constraints in robotics and other application areas. The difference is that our approach deals with deformable objects.

alized 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 [8] 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 manageable 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 [12]. Marr and Nishihara [36] 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 [70] 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 [49][50] proposes

the use of a simpler part model with scalar parameters, the superquadric ellipsoid with parameterized deformations [3] (the notion of a superquadric was introduced by Hein [21]). 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 [24][25][61].

Pentland [50, 51], following the physics-based approach of Terzopoulos, Witkin and Kass [70], proposes an alternative method for fitting deformable part models based on superquadric ellipsoids. Inspired by modal analysis, a technique for analyzing the vibrations of linear mechanical systems under periodic forcing conditions described by Bathe and Wilson [6], he applies to superquadrics polynomial approximation to the deformation "modes" of a 21 node element. Pentland's modeling primitives are not fully dynamic in that the underlying superquadric parameters do not respond to forces and are not fitted to data through force interactions. The deformation modes may make the method efficient for the recovery of smooth, symmetrically deformed parts. On the down side, global deformation modes lack an obvious physical meaning, and they make it difficult to deal with nonlinearities and boundary conditions. Moreover, the representation of complex shapes requires many modes, rendering Pentland's scheme no more efficient than a nodal finite element solution [6].

In this thesis we combine the best of the Terzopoulos, Witkin, and Kass models [70] and Pentland's models [51] into our deformable primitives. The coupling of rigid-body and deformation dynamics is similar to that described in [70], but our formulation accommodates global deformations defined by fully nonlinear parametric equations. Hence, our models are more general than the restrictive, linearly deformable ones in [72, 2] and quadratically deformable ones in [52, 57].

2.1.2 Recursive Estimation of Shape and Nonrigid Motion

Kalman filtering techniques described by Gelb [22] have been applied in the vision literature to the estimation of surface depth [37, 27], dynamic features (e.g., Deriche and Faugeras [15]), and rigid motion parameters [19, 10, 11] of objects from image sequences. To date, these Kalman filters have been discrete filters with the simplest possible system models, usually constant velocity assumptions (e.g., Pentland and Horowitz [51]). Additional restrictions such as constant error covariance matrices may also be found in the literature [51]. 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 filters that we develop in this paper incorporate rather sophisticated Lagrange equations of 3D nonrigid motion as system models (simpler dynamic models for the estimation of rigid body motion parameters [20, 19] or 2D nonrigid motion parameters developed by Szeliski and Terzopoulos [63] 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 proposed in [33, 70] and applied to deformable superquadrics by Terzopoulos and Metaxas [67] amounts to a degenerate "Kalman filter" with a constant, unit error covariance matrix.

2.2 Computer Graphics

2.2.1 Physics-Based Modeling

In the computer graphics literature, mathematical representations of solid objects has been the commonplace. The field of solid modeling [30] has developed ge-

ometric 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 [58] define local deformations of solid primitives as ambient space warps) becomes particularly evident when one faces the problem of realistically animating deformable objects. Physically-based models have been pursued for this purpose (e.g., [69, 26, 54, 65, 48, 14, 23, 52, 32, 13]). These methods introduce realistic physical behaviors into free-form geometric models of solids or their surfaces.

Our 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 [70, 68], but it is much more general, in part, because we must accommodate geometric primitives. Our treatment of global deformation dynamics is similar to Witkin and Welch's [72] 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 [4]). The implicitly defined models of Sclaroff and Pentland [57] may be viewed as complementary to the global parametric deformations defined in the thesis; however, our 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 elements techniques described by Kardestuncer [31].

2.2.2 Constraint Methods

Several researchers have proposed physics-based constraint methods for working with and controlling animations involving rigid and nonrigid primitives [5, 54, 53, 71, 72]. We describe a method for computing generalized constraint forces between our deformable models which is based on the constraint stabilization technique of Baumgarte [7, 73]. The resulting constraint satisfaction algorithm is efficient and

stable for point-to-point constraints. It allows the construction and animation of articulated objects composed of rigid or nonrigid parameterized parts. Our constraint algorithm is a generalization of work on physics-based constraints between rigid parts developed by Barzel and Barr [5] and linearly deformable parts developed by Witkin and Welsch [72]. As in [5], it can support the assembly of complex objects satisfying constraints from inappropriately shaped and mispositioned deformable parts that do not initially satisfy the constraints. Furthermore, to simulate interactions of our models with the physical world we adopt a force-based approach to collisions which is equivalent to the velocity based technique used by [1, 2] in case of rigid and globally deformable models. Simplifying the model by ignoring some of the physics and using simple numerical integration techniques for our constrained motion equations we can achieve real or interactive time animations on graphics workstations, particularly for the numerical modeling of hollow "deformable shells" which are used in many computer vision and computer graphics applications.

Chapter 3

Geometry of Deformable Models

This chapter develops a technique for creating classes of hybrid 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 well-posed 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 special case of 3D surface models.

3.1 Hybrid Models

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

The positions of points on the model relative to an inertial frame of reference

[1] For the case of a 3D "shell," $\mathrm{u} = (u, v, 1)$

Φ in space are given by a vector-valued, time varying function of u:

$$\mathbf{x}(\mathrm{u},t) = (x_1(\mathrm{u},t), x_2(\mathrm{u},t), x_3(\mathrm{u},t))^{\mathsf{T}}, \tag{3.1}$$

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

$$\mathbf{x} = \mathbf{c} + \mathbf{R}\mathbf{p}, \tag{3.2}$$

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, $\mathbf{p}(\mathrm{u},t)$ 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 $\mathbf{s}(\mathrm{u},t)$ and a displacement function $\mathbf{d}(\mathrm{u},t)$:

$$\mathbf{p} = \mathbf{s} + \mathbf{d}. \tag{3.3}$$

Fig. 3.1 illustrates the model geometry.

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. In particular we dub *deformable superquadrics* deformable models which use superquadrics as parameterized primitives.

3.1.1 Global deformations

We define the reference shape as

$$\mathbf{s} = \mathbf{T}(\mathbf{e}(\mathrm{u};\, a_0, a_1, \ldots);\, b_0, b_1, \ldots) = \mathbf{T}(\mathbf{e}; \mathbf{b}). \tag{3.4}$$

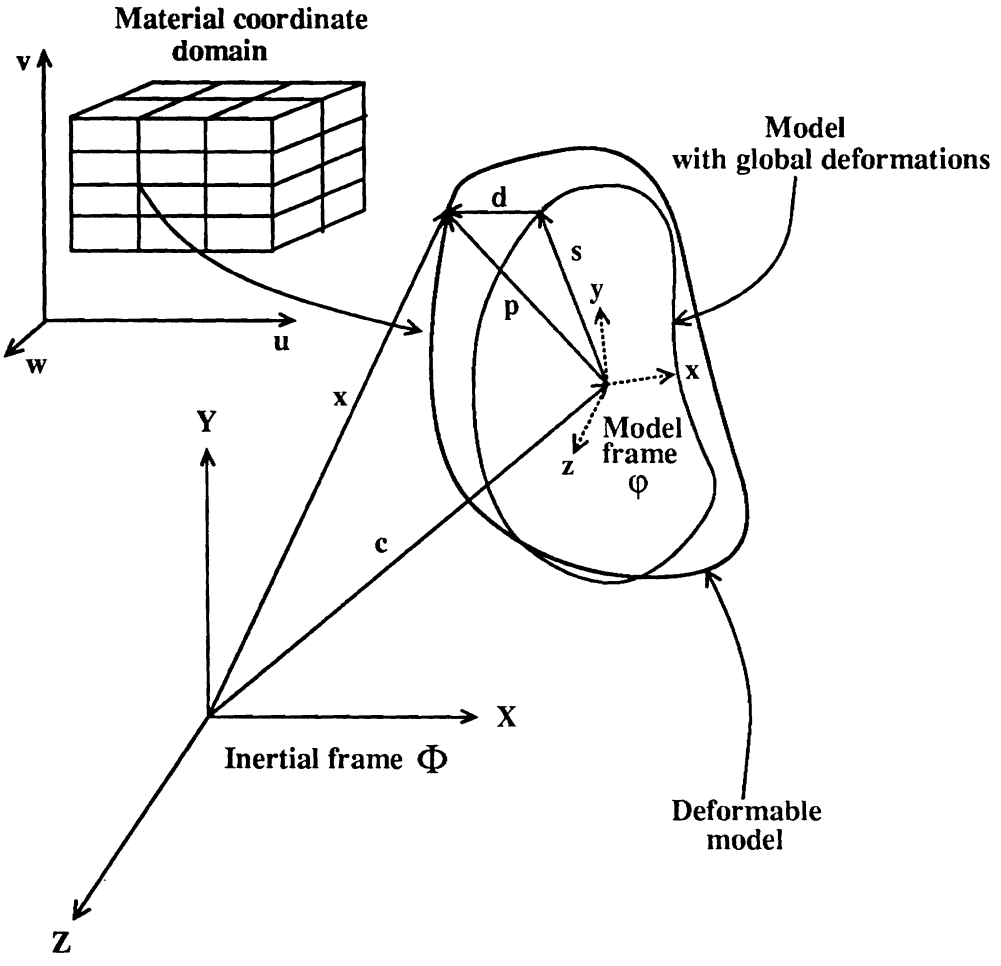


Figure 3.1: Geometry of deformable model.

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

$$\mathbf{q}_s = (a_0, a_1, \ldots, b_0, b_1, \ldots)^{\mathsf{T}}. \tag{3.5}$$

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

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

$$\mathbf{J} = \frac{\partial \mathbf{s}}{\partial \mathbf{q}_s} \tag{3.6}$$

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

Example 1: Superquadric ellipsoid

We consider a 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

$$\mathbf{e} = a_0 w \begin{pmatrix} a_1 C_u{}^{\epsilon_1} C_v{}^{\epsilon_2} \\ a_2 C_u{}^{\epsilon_1} S_v{}^{\epsilon_2} \\ a_3 S_u{}^{\epsilon_1} \end{pmatrix}, \tag{3.7}$$

where $-\pi/2 \le u \le \pi/2$, $-\pi \le v < \pi$, $0 \le w \le 1$, and $S_u{}^{\epsilon} = \mathrm{sgn}(\sin u)|\sin u|^{\epsilon}$, $C_u{}^{\epsilon} = \mathrm{sgn}(\cos u)|\cos u|^{\epsilon}$, and similarly for $C_v{}^{\epsilon}$ and $S_v{}^{\epsilon}$. Here, $a_0 \ge 0$ is a scale parameter, $0 \le a_1, a_2, a_3 \le 1$, are aspect ratio parameters, and $\epsilon_1, \epsilon_2 \ge 0$ are "squareness" parameters [3, 4].

We collect the parameters in **s** into the parameter vector

$$\mathbf{q}_s = (a_0, a_1, a_2, a_3, \epsilon_1, \epsilon_2)^\mathsf{T}. \tag{3.8}$$

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

$$\begin{aligned}
\mathbf{J}_{11} &= wa_1 C_u{}^{\epsilon_1} C_v{}^{\epsilon_2}, \\
\mathbf{J}_{12} &= a_0 w C_u{}^{\epsilon_1} C_v{}^{\epsilon_2}, \\
\mathbf{J}_{15} &= a_0 w a_1 \ln(|\cos u|) C_u{}^{\epsilon_1} C_v{}^{\epsilon_2}, \\
\mathbf{J}_{16} &= a_0 w a_1 \ln(|\cos v|) C_u{}^{\epsilon_1} C_v{}^{\epsilon_2}, \\
\mathbf{J}_{21} &= wa_2 C_u{}^{\epsilon_1} S_v{}^{\epsilon_2}, \\
\mathbf{J}_{23} &= a_0 w C_u{}^{\epsilon_1} S_v{}^{\epsilon_2}, \\
\mathbf{J}_{25} &= a_0 w a_2 \ln(|\cos u|) C_u{}^{\epsilon_1} S_v{}^{\epsilon_2}, \\
\mathbf{J}_{26} &= a_0 w a_2 \ln(|\sin v|) C_u{}^{\epsilon_1} S_v{}^{\epsilon_2}, \\
\mathbf{J}_{31} &= wa_3 S_u{}^{\epsilon_1}, \\
\mathbf{J}_{34} &= a_0 w S_u{}^{\epsilon_1}, \\
\mathbf{J}_{35} &= a_0 w a_3 \ln(|\sin u|) S_u{}^{\epsilon_1}.
\end{aligned} \tag{3.9}$$

Example 2: Tapering/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 [4] and [61], 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 3[2] of the superquadric **e** into a single parameterized deformation **T**,

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

and express the reference shape as

$$\mathbf{s} = \mathbf{T}(\mathbf{e}, t_1, t_2, b_1, b_2, b_3) = \begin{pmatrix} (\frac{t_1 e_3}{a_0 a_3 w} + 1)\ e_1 + b_1\ cos(\frac{e_3 + b_2}{a_0 a_3 w}\pi b_3) \\ (\frac{t_2 e_3}{a_0 a_3 w} + 1)\ e_2 \\ e_3 \end{pmatrix}, \tag{3.10}$$

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

$$\mathbf{q}_s = (a_0, a_1, a_2, a_3, \epsilon_1, \epsilon_2, t_1, t_2, b_1, b_2, b_3)^\mathsf{T}. \tag{3.11}$$

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

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

$$
\begin{aligned}
\mathbf{J}_{25} &= t_2 \ln(|\sin u|) {S_u}^{\epsilon_1} a_0 w a_2 {C_u}^{\epsilon_1} {S_v}^{\epsilon_2} + (t_2 {S_u}^{\epsilon_1} + 1) a_0 w a_2 \ln(|\cos u|) {C_u}^{\epsilon_1} {S_v}^{\epsilon_2}, \\
\mathbf{J}_{35} &= a_0 w a_3 \ln(|\sin u|) {S_u}^{\epsilon_1}, \\
\mathbf{J}_{16} &= (t_1 {S_u}^{\epsilon_1} + 1) a_0 w a_1 \ln(|\cos v|) {C_u}^{\epsilon_1} {C_v}^{\epsilon_2}, \\
\mathbf{J}_{26} &= (t_2 {S_u}^{\epsilon_1} + 1) a_0 w a_2 \ln(|\sin v|) {C_u}^{\epsilon_1} {S_v}^{\epsilon_2}, \\
\mathbf{J}_{17} &= {S_u}^{\epsilon_1} a_0 w a_1 {C_u}^{\epsilon_1} {C_v}^{\epsilon_2}, \\
\mathbf{J}_{28} &= {S_u}^{\epsilon_1} a_0 w a_2 {C_u}^{\epsilon_1} {S_v}^{\epsilon_2}, \\
\mathbf{J}_{19} &= \cos(r), \\
\mathbf{J}_{1\,10} &= -\frac{b_1 b_3}{a_0 w a_3} \pi \sin(r), \\
\mathbf{J}_{1\,11} &= -b_1 \pi \sin(r)\, r,
\end{aligned}
\tag{3.12}
$$

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

Example 3: Tapering/bending/shearing/twisting

We now generalize the model defined in example 2 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.,

$$\mathbf{T}(\mathbf{e};\mathbf{b}) = \mathbf{T}_2(\mathbf{T}_1(\mathbf{T}_3(\mathbf{e};\mathbf{b}))). \tag{3.13}$$

The reference shape is therefore defined as

$$\mathbf{s} = \mathbf{T}(\mathbf{e};\mathbf{b}). \tag{3.14}$$

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

The transformation along axis 3 is given by

$$\mathbf{r}_3 = \mathbf{T}_3(\mathbf{e};\mathbf{b}) = \begin{pmatrix} A_3\cos(\phi_3) - B_3\sin(\phi_3) \\ A_3\sin(\phi_3) + B_3\cos(\phi_3) \\ e_3 \end{pmatrix}, \tag{3.15}$$

where

$$A_3 = [(t_3^1 e_3/a_0 a_3 w) + 1]\ e_1 + b^{a1}_3 \cos[(e_3 + b^{l1}_3)/(a_0 a_3 w)\pi b^{r1}_3] + e_3 s_3^1, \tag{3.16}$$

$$B_3 = [(t_3^2 e_3/a_0 a_3 w) + 1]\ e_2 + b^{a2}_3 \cos[(e_3 + b^{l2}_3)/(a_0 a_3 w)\pi b^{r2}_3] + e_3 s_3^2 \tag{3.17}$$

and

$$\phi_3 = (e_3/a_0 a_3 w)\pi\tau_3. \tag{3.18}$$

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

$$\mathbf{r}_1 = \mathbf{T}_1(\mathbf{T}_3(\mathbf{e};\mathbf{b})) = \begin{pmatrix} \mathbf{r}_{3_3} \\ A_1\cos(\phi_1) - B_1\sin(\phi_1) \\ A_1\sin(\phi_1) + B_1\cos(\phi_1) \end{pmatrix}, \tag{3.19}$$

where

$$A_1 = [(t_1^2 \mathbf{r}_{3_1}/a_0 a_1 w) + 1]\ \mathbf{r}_{3_2} + b^{a2}_1 \cos[(\mathbf{r}_{3_1} + b^{l2}_1)/(a_0 a_1 w)\pi b^{r2}_1] + \mathbf{r}_{3_1} s_1^2, \tag{3.20}$$

$$B_1 = [(t_1^3 \mathbf{r}_{3_1}/a_0 a_1 w) + 1]\ \mathbf{r}_{3_3} + b^{a3}_1 \cos[(\mathbf{r}_{3_1} + b^{l3}_1)/(a_0 a_1 w)\pi b^{r3}_1] + \mathbf{r}_{3_1} s_1^3 \tag{3.21}$$

and

$$\phi_1 = (\mathbf{r}_{3_1}/a_0 a_1 w)\pi\tau_1. \tag{3.22}$$

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 $b^{a}{}_1^j$, $b^{l}{}_1^j$, $b^{r}{}_1^j$ 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

$$\mathbf{s} = \mathbf{T}_2(\mathbf{T}_1(\mathbf{T}_3(\mathbf{e};\mathbf{b}))) = \begin{pmatrix} A_2\cos(\phi_2) - B_2\sin(\phi_2) \\ \mathbf{r}_{1_2} \\ A_2\sin(\phi_2) + B_2\cos(\phi_2) \end{pmatrix}, \tag{3.23}$$

where

$$A_2 = [(t_2^3\mathbf{r}_{1_2}/a_0a_2w) + 1]\ \mathbf{r}_{1_3} + b^{a}{}_2^3\cos[(\mathbf{r}_{1_2} + b^{l}{}_2^3)/(a_0a_2w)\pi b^{r}{}_2^3] + \mathbf{r}_{1_2}s_2^3, \tag{3.24}$$

$$B_2 = [(t_2^1\mathbf{r}_{1_2}/a_0a_2w) + 1]\ \mathbf{r}_{1_1} + b^{a}{}_2^1\cos[(\mathbf{r}_{1_2} + b^{l}{}_2^1)/(a_0a_2w)\pi b^{r}{}_2^1] + \mathbf{r}_{1_2}s_2^1, \tag{3.25}$$

and

$$\phi_2 = (\mathbf{r}_{1_2}/a_0a_2w)\pi\tau_2. \tag{3.26}$$

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 $b^{a}{}_2^j$, $b^{l}{}_2^j$, $b^{r}{}_2^j$ 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

$$\mathbf{q}_s = (a_0, a_1, a_2, a_3, \epsilon_1, \epsilon_2, \tau_i, t_i^j, s_i^j, b^{a}{}_i^j, b^{l}{}_i^j, b^{r}{}_i^j)^\top, \tag{3.27}$$

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.

3.1.2 Local Deformations

In general [59], 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) $b_j(\mathrm{u})$

$$\mathbf{d} = \sum_i \mathbf{S}_i\mathbf{q}_{d_i}, \tag{3.28}$$

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* [29] are employed. These methods express the displacement $\mathbf{d}$ in terms of a finite number of basis functions. In this case the series of (3.28) are truncated leading to

$$\mathbf{d} = \sum_{i=1}^{n} \mathbf{S}_i \mathbf{q}_{d_i}. \tag{3.29}$$

We can rewrite (3.29) in the compact matrix form

$$\mathbf{d} = \mathbf{S}\mathbf{q}_d, \tag{3.30}$$

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 $\mathbf{q}_{d_i}$

$$\mathbf{q}_d = (\ldots, \mathbf{q}_{d_i}^{\top}, \ldots)^{\top}. \tag{3.31}$$

In this thesis, we will be using the finite element method [74], a special case of the Rayleigh-Ritz method, 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 in this thesis will be given in a later chapter.

3.2 Summary

In this chapter we provided the geometric formulation of our models. We described a general technique for defining global and local geometric degrees of freedom and applied it to create a new class of deformable models that we dub *deformable superquadrics.* Our technique is general and can be applied to any class of parameterized geometric primitives and deformations as long as their underlying equations are differentiable.

Chapter 4

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 $\mathbf{q}$-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 the 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. Finally, we present a force-based estimation technique for shape and nonrigid motion estimation in computer vision applications.

4.1 Kinematics

From (3.2), the velocity of a point on the model is given by

$$\begin{aligned} \dot{\mathbf{x}} &= \dot{\mathbf{c}} + \dot{\mathbf{R}}\mathbf{p} + \mathbf{R}\dot{\mathbf{p}} \\ &= \dot{\mathbf{c}} + \mathbf{B}\dot{\boldsymbol{\theta}} + \mathbf{R}\dot{\mathbf{s}} + \mathbf{RS}\dot{\mathbf{q}}_d, \end{aligned} \tag{4.1}$$

where $\boldsymbol{\theta} = (\ldots, \theta_i, \ldots)^\top$ is the vector of rotational coordinates of the model and $\mathbf{B} = [\ldots \partial(\mathbf{Rp})/\partial\theta_i \ldots]$. Furthermore,

$$\dot{\mathbf{s}} = \left[\frac{\partial \mathbf{s}}{\partial \mathbf{q}_s}\right] \dot{\mathbf{q}}_s \;=\; \mathbf{J}\dot{\mathbf{q}}_s, \tag{4.2}$$

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 kinematics compactly as

$$\mathbf{x} = \mathbf{c} + \mathbf{R}(\mathbf{s} + \mathbf{d}) = \xi(\mathbf{q}) \tag{4.3}$$

and

$$\dot{\mathbf{x}} = [\mathbf{I}\ \mathbf{B}\ \mathbf{RJ}\ \mathbf{RS}]\dot{\mathbf{q}} = \mathbf{L}\dot{\mathbf{q}}, \tag{4.4}$$

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,

$$\mathbf{q} = (\mathbf{q}_c^\top, \mathbf{q}_\theta^\top, \mathbf{q}_s^\top, \mathbf{q}_d^\top)^\top, \tag{4.5}$$

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

4.1.1 Computation of R and B Using Quaternions

We represent $\mathbf{q}_\theta$ 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 [73].

A quaternion $[s, \mathbf{v}_q]$ with unit magnitude [60],

$$\|[s, \mathbf{v}_q]\| = s^2 + \mathbf{v}_q^\top \mathbf{v}_q = 1, \tag{4.6}$$

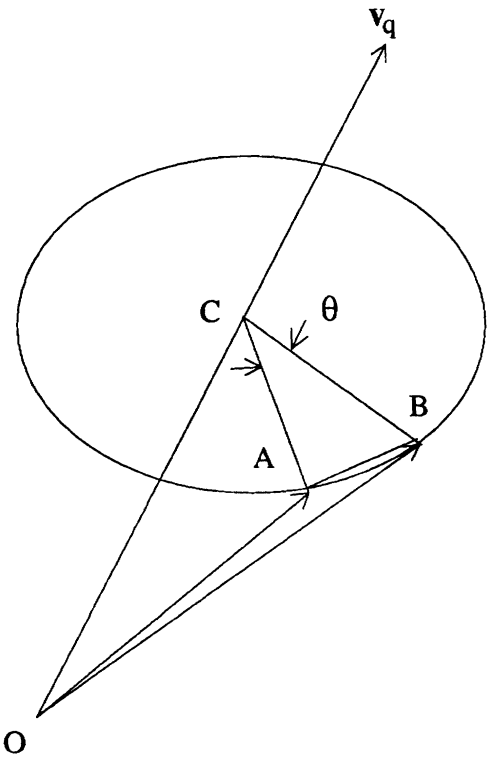


Figure 4.1: Rotation by angle θ through an axis of rotation $\mathbf{v}_q$.

specifies a rotation of the model from its reference position through an angle $\theta = 2\cos^{-1} s$ around an axis aligned with vector $\mathbf{v}_q = (v_1, v_2, v_3)^\top$ as shown in Fig. 4.1.

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

$$\mathbf{R} = \begin{bmatrix} 1-2(v_2^2+v_3^2) & 2(v_1v_2 - sv_3) & 2(v_1v_3 + sv_2) \\ 2(v_1v_2 + sv_3) & 1-2(v_1^2+v_3^2) & 2(v_2v_3 - sv_1) \\ 2(v_1v_3 - sv_2) & 2(v_2v_3 + sv_1) & 1-2(v_1^2+v_2^2) \end{bmatrix}. \tag{4.7}$$

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

$$\mathbf{R}^\top = \mathbf{R}^{-1}. \tag{4.8}$$

The matrix $\mathbf{B}$ is given by [59] (see Appendix A for a proof)

$$\mathbf{B}(u) = -\mathbf{R}\ \tilde{\mathbf{p}}(u)\ \mathbf{G}, \tag{4.9}$$

where $\mathbf{R}$ represents the rotation matrix at time t, $\tilde{\mathbf{p}}(u)$ is the dual 3×3 matrix of the position vector $\mathbf{p}(u) = (p_1, p_2, p_3)^\mathsf{T}$ (see (3.3)) defined as

$$\tilde{\mathbf{p}}(u) = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}, \tag{4.10}$$

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

$$\mathbf{G} = 2 \begin{bmatrix} -v_1 & s & v_3 & -v_2 \\ -v_2 & -v_3 & s & v_1 \\ -v_3 & v_2 & -v_1 & s \end{bmatrix}. \tag{4.11}$$

4.2 Dynamics

In computer vision 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 $\mathbf{q}_\theta$ 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.

4.2.1 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

$$\frac{d}{dt}\left(\frac{\partial \mathcal{T}}{\partial \dot{\mathbf{q}}}\right)^{\mathsf{T}} - \left(\frac{\partial \mathcal{T}}{\partial \mathbf{q}}\right)^{\mathsf{T}} + \left(\frac{\partial \mathcal{F}}{\partial \dot{\mathbf{q}}}\right)^{\mathsf{T}} + \delta_{\mathbf{q}}\mathcal{E} = \mathbf{f}_q. \tag{4.12}$$

These equations can be written in the form

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{g}_q + \mathbf{f}_q, \tag{4.13}$$

where **M**, **D**, and **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 $\mathbf{f}_q(\mathrm{u}, t)$ are the generalized external forces associated with the degrees of freedom q of the model.

In the following subsections we derive from (4.12) formulas for the matrices and vectors in (4.13).

4.2.2 Kinetic Energy: Mass Matrix

The kinetic energy T of the model is given by

$$\mathcal{T} = \frac{1}{2}\int \mu \dot{\mathbf{x}}^{\mathsf{T}}\dot{\mathbf{x}}\, d\mathrm{u} = \frac{1}{2}\dot{\mathbf{q}}^{\mathsf{T}}\left[\int \mu \mathbf{L}^{\mathsf{T}}\mathbf{L}\, d\mathrm{u}\right]\dot{\mathbf{q}} = \frac{1}{2}\dot{\mathbf{q}}^{\mathsf{T}}\mathbf{M}\dot{\mathbf{q}}, \tag{4.14}$$

where $\mathbf{M} = \int \mu \mathbf{L}^{\mathsf{T}}\mathbf{L}\, d\mathrm{u}$ is the symmetric mass matrix of the object and $\mu(\mathrm{u})$ is the mass density of the object. Using the expression for **L** from (4.4), we can rewrite **M** as a block symmetric matrix as follows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{cc} & \mathbf{M}_{c\theta} & \mathbf{M}_{cs} & \mathbf{M}_{cd} \\ & \mathbf{M}_{\theta\theta} & \mathbf{M}_{\theta s} & \mathbf{M}_{\theta d} \\ & & \mathbf{M}_{ss} & \mathbf{M}_{sd} \\ \text{symmetric} & & & \mathbf{M}_{dd} \end{bmatrix}, \tag{4.15}$$

where

$$\begin{aligned}
\mathbf{M}_{cc} &= \int \mu \mathbf{I}\, du, & \mathbf{M}_{\theta s} &= \int \mu \mathbf{B}^\top \mathbf{R}\mathbf{J}\, du, \\
\mathbf{M}_{c\theta} &= \int \mu \mathbf{B}\, du, & \mathbf{M}_{\theta d} &= \int \mu \mathbf{B}^\top \mathbf{R}\mathbf{S}\, du, \\
\mathbf{M}_{cs} &= \mathbf{R}\int \mu \mathbf{J}\, du, & \mathbf{M}_{ss} &= \int \mu \mathbf{J}^\top \mathbf{J}\, du, \\
\mathbf{M}_{cd} &= \mathbf{R}\int \mu \mathbf{S}\, du, & \mathbf{M}_{sd} &= \int \mu \mathbf{J}^\top \mathbf{S}\, du, \\
\mathbf{M}_{\theta\theta} &= \int \mu \mathbf{B}^\top \mathbf{B}\, du, & \mathbf{M}_{dd} &= \int \mu \mathbf{S}^\top \mathbf{S}\, du.
\end{aligned} \tag{4.16}$$

4.2.3 Calculation of Acceleration and Inertial Forces

The acceleration of a point $\mathbf{x}$ on the deformable model is shown in [59] to satisfy

$$\ddot{\mathbf{x}} = \mathbf{L}\ddot{\mathbf{q}} + \dot{\mathbf{L}}\dot{\mathbf{q}}, \tag{4.17}$$

where

$$\dot{\mathbf{L}}\dot{\mathbf{q}} = \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{R}\mathbf{p}) + 2\boldsymbol{\omega} \times \mathbf{R}\dot{\mathbf{p}}. \tag{4.18}$$

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

$$\dot{\mathbf{p}} = \dot{\mathbf{s}} + \dot{\mathbf{d}} = \mathbf{J}\dot{\mathbf{q}}_s + \mathbf{S}\dot{\mathbf{q}}_d, \tag{4.19}$$

and the angular velocity of the deformable model with respect to the world coordinate system using [59]

$$\boldsymbol{\omega} = \mathbf{Q}\dot{\boldsymbol{\theta}}, \tag{4.20}$$

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

$$\mathbf{Q} = 2\begin{bmatrix} -v_1 & s & -v_3 & v_2 \\ -v_2 & v_3 & s & -v_1 \\ -v_3 & -v_2 & v_1 & s \end{bmatrix}. \tag{4.21}$$

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

[59]

$$\delta \mathbf{W}_I = \int \mu \delta \mathbf{x}^\top \ddot{\mathbf{x}}\, d\mathrm{u} = \int \delta \mathbf{q}^\top \mathbf{L}^\top (\mathbf{L}\ddot{\mathbf{q}} + \dot{\mathbf{L}}\dot{\mathbf{q}})\, d\mathrm{u} = \delta \mathbf{q}^\top (\mathbf{M}\ddot{\mathbf{q}} - \mathbf{g}_q), \tag{4.22}$$

where the generalized mass matrix is

$$\mathbf{M} = \int \mu \mathbf{L}^\top \mathbf{L}\, d\mathrm{u} \tag{4.23}$$

and the generalized inertial forces are

$$\mathbf{g}_q = -\int \mu \mathbf{L}^\top \dot{\mathbf{L}}\dot{\mathbf{q}}\, d\mathrm{u}. \tag{4.24}$$

Using (4.14) and (4.22), the first two terms of (4.12) which express inertial forces can be written as

$$\frac{d}{dt}\left(\frac{\partial \mathcal{T}}{\partial \dot{\mathbf{q}}}\right)^\top - \left(\frac{\partial \mathcal{T}}{\partial \mathbf{q}}\right)^\top = \mathbf{M}\ddot{\mathbf{q}} - \mathbf{g}_q, \tag{4.25}$$

where

$$\begin{aligned} \mathbf{g}_q &= -\dot{\mathbf{M}}\dot{\mathbf{q}} + \frac{1}{2}\left[\frac{\partial}{\partial \mathbf{q}}(\dot{\mathbf{q}}^\top \mathbf{M}\dot{\mathbf{q}})\right]^\top \\ &= -\int \mu \mathbf{L}^\top \dot{\mathbf{L}}\dot{\mathbf{q}}\, d\mathrm{u} \end{aligned} \tag{4.26}$$

gives the centrifugal and Coriolis forces [59].

4.2.4 Energy Dissipation: Damping Matrix

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

$$\mathcal{F} = \frac{1}{2}\int \gamma \dot{\mathbf{x}}^\top \dot{\mathbf{x}}\, d\mathrm{u}, \tag{4.27}$$

where $\gamma(u)$ is a damping density. Since it has the same form as (4.14) we can rewrite (4.27) as follows:

$$\mathcal{F} = \frac{1}{2}\dot{\mathbf{q}}^{\top}\mathbf{D}\dot{\mathbf{q}}, \tag{4.28}$$

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

Using (4.14), we express the third term of (4.12)

$$\frac{\partial \mathcal{F}}{\partial \dot{\mathbf{q}}} = \mathbf{D}\dot{\mathbf{q}}. \tag{4.29}$$

4.2.5 Strain Energy: Stiffness Matrix

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

$$\mathbf{K} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & & \mathbf{K}_{ss} & \mathbf{K}_{sd} \\ \text{symmetric} & & & \mathbf{K}_{dd} \end{pmatrix}. \tag{4.30}$$

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 (4.30). 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

$$\mathcal{E}_{s_i} = \frac{1}{2} k_{s_i} (\mathbf{q}_{s_i} - \mathbf{q}_{s_{i0}})^2, \tag{4.31}$$

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.,

$$\mathbf{K}_{ss} = \text{diag}(k_{s_i}) \tag{4.32}$$

Differentiating (4.31) with respect to $\mathbf{q}_{s_i}$ we derive the global elastic force associated with parameter $\mathbf{q}_{s_i}$

$$\mathbf{f}_{s_i} = k_{s_i} (\mathbf{q}_{s_i} - \mathbf{q}_{s_{i0}}). \tag{4.33}$$

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 [68], suitable for $\mathbf{C}^0$ continuous model

where the function $w_{00}(\mathrm{u})$ controls the local magnitude and $w_{10}(\mathrm{u})$, $w_{01}(\mathrm{u})$ 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 [64]

$$\begin{aligned}\mathcal{E}_p(\mathbf{d}) &= \int w_{20}\left(\frac{\partial^2 \mathbf{d}}{\partial u^2}\right)^2 + w_{11}\left(\frac{\partial^2 \mathbf{d}}{\partial u \partial v}\right)^2 + w_{02}\left(\frac{\partial^2 \mathbf{d}}{\partial v^2}\right)^2 + \\ &\quad w_{10}\left(\frac{\partial \mathbf{d}}{\partial u}\right)^2 + w_{01}\left(\frac{\partial \mathbf{d}}{\partial v}\right)^2 + w_{00}\mathbf{d}^2\, d\mathrm{u} \end{aligned} \tag{4.35}$$

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 $w_{ij}(\mathrm{u}) = w_{ij}$.

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

$$\mathcal{E} = \int \boldsymbol{\sigma}^\top \boldsymbol{\epsilon}\, d\mathrm{u}, \tag{4.36}$$

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

$$\boldsymbol{\epsilon} = \mathcal{P}\mathbf{d}, \tag{4.37}$$

where $\mathcal{P}$ is a differential operator that is derived from the local deformation strain

energy (e.g., 4.34 and 4.35). In terms of the generalized local coordinates $\mathbf{q}_d$ we can rewrite (4.37) as

$$\boldsymbol{\epsilon} = \mathcal{P}\mathbf{S}\mathbf{q}_d. \tag{4.38}$$

We can also express the relation between the stress $\boldsymbol{\sigma}$ and strains $\boldsymbol{\epsilon}$ as

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\epsilon}, \tag{4.39}$$

where the symmetric matrix $\mathbf{D}$ is derived from the local deformation strain energy. Substituting (4.38) into (4.39) yields

$$\boldsymbol{\sigma} = \mathbf{D}\mathcal{P}\mathbf{S}\mathbf{q}_d. \tag{4.40}$$

Substituting (4.38) and (4.40) into (4.36) we get

$$\mathcal{E} = \int \mathbf{q}_d^\top (\mathcal{P}\mathbf{S})^\top \mathbf{D}\mathcal{P}\mathbf{S}\mathbf{q}_d \, d\mathrm{u}, \tag{4.41}$$

where we utilize the symmetry of $\mathbf{D}$. Since $\mathbf{q}_d$ depends only on time we can rewrite (4.41) as

$$\mathcal{E} = \mathbf{q}_d^\top \left(\int (\mathcal{P}\mathbf{S})^\top \mathbf{D}\mathcal{P}\mathbf{S} \, d\mathrm{u} \right) \mathbf{q}_d \tag{4.42}$$

or in compact form

$$\mathcal{E} = \mathbf{q}_d^\top \mathbf{K}_{dd} \mathbf{q}_d, \tag{4.43}$$

where

$$\mathbf{K}_{dd} = \int (\mathcal{P}\mathbf{S})^\top \mathbf{D}\mathcal{P}\mathbf{S} \, d\mathrm{u} \tag{4.44}$$

is the symmetric positive definite local deformation stiffness matrix.

In a subsequent 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 (4.31) and (4.43) the fourth term of (4.12), the variation of $\mathcal{E}$ with respect

to $\mathbf{q}$, can be written

$$\delta_{\mathbf{q}}\mathcal{E} = \mathbf{K}\mathbf{q}. \tag{4.45}$$

4.2.6 External Forces and Virtual Work

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

$$\delta\mathbf{W}_F = \int \mathbf{f}^\top \mathbf{L}\delta\mathbf{q} = \mathbf{f}_q^\top \delta\mathbf{q}, \tag{4.46}$$

where

$$\mathbf{f}_q^\top = \int \mathbf{f}^\top \mathbf{L}\, d\mathrm{u} = (\mathbf{f}_c^\top, \mathbf{f}_\theta^\top, \mathbf{f}_s^\top, \mathbf{f}_d^\top), \tag{4.47}$$

and

$$\begin{array}{llll} \mathbf{f}_c^\top = \int \mathbf{f}^\top d\mathrm{u}, & \mathbf{f}_s^\top = \int \mathbf{f}^\top \mathbf{R}\mathbf{J}\, d\mathrm{u}, \\ \mathbf{f}_\theta^\top = \int \mathbf{f}^\top \mathbf{B}\, d\mathrm{u}, & \mathbf{f}_d^\top = \int \mathbf{f}^\top \mathbf{R}\mathbf{S}\, d\mathrm{u}, \end{array} \tag{4.48}$$

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

4.3 Choosing the Order of the Motion Equations

For convenience, we rewrite the second-order equations (4.13) in standard dynamical system form, as the coupled set of first-order equations

$$\dot{\mathbf{u}} = \mathbf{F}\mathbf{u} + \mathbf{g}, \tag{4.49}$$

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

$$\mathbf{u} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{q} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} -\mathbf{M}^{-1}\mathbf{D} & -\mathbf{M}^{-1}\mathbf{K} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \mathbf{M}^{-1}(\mathbf{g}_q + \mathbf{f}_q) \\ \mathbf{0} \end{bmatrix}. \tag{4.50}$$

A full implementation and simulation of the above dynamic equations would be appropriate for physics-based animation where it is important to achieve real-

istic motion synthesis [68]. 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 (4.13) (or, equivalently, (4.49) and (4.50)) is appropriate in such a case, since the mass term provides intertia; 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 zero, which nonetheless preserves useful first-order dynamics that achieve the equilibrium fit.

Setting the mass density $\mu(\mathrm{u})$ (see eq. (4.14)) to zero in (4.13) results in the first-order dynamic system

$$\mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{f}_\mathbf{q} \tag{4.51}$$

(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 (4.51) in the standard form (4.49), where

$$\mathbf{u} = \mathbf{q}, \quad \mathbf{F} = -\mathbf{D}^{-1}\mathbf{K}, \quad \mathbf{g} = \mathbf{D}^{-1}\mathbf{f}_\mathbf{q}. \tag{4.52}$$

[1] In 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.

4.4 Summary

In this chapter we described the kinematic and dynamic formulation of our models. We also described 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. This method depends on the calculation of a Jacobian matrix which requires that all the relevant model equations are differentiable. Furthermore, we presented a general technique for deriving the stiffness matrix associated with the elastic properties of the model from a given deformation energy expression.

Chapter 5

Finite Element Implementation

This chapter presents the finite element method that we will employ to discretize the Lagrange equations of motion for both vision and graphics applications. We state the criteria for choosing the appropriate elements for a given problem, we give various finite element tessellations of the parametric space of a deformable model, we show examples of finite elements and finally we present the approximation to the Lagrange equations of motion using finite elements.

5.1 Finite Elements

In this thesis we will be using the finite element method [6], a special case of the Rayleigh-Ritz method, 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

$$\mathbf{d}^j = \sum_{i=1}^{n} N_i^j(\mathrm{u})\mathbf{q}_{d_i}^j = \mathbf{N}^j\mathbf{q}_d^j, \tag{5.1}$$

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, $\mathbf{q}_d^j = (\mathbf{q}_{d_1}^{j^\top}, \ldots, \mathbf{q}_{d_n}^{j^\top})^\top$ and n is the number of the element's nodes.

Using (5.1) we can express the displacement $\mathbf{d}$ anywhere within the deformable model as

$$\mathbf{d} = \mathbf{S}\mathbf{q}_d, \tag{5.2}$$

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 $\mathbf{f}(u)$ within an element j, then we can extrapolate it to the nodes of the element using the formula

$$\mathbf{f}_i = N_i(\mathrm{u})\mathbf{f}(u), \tag{5.3}$$

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

5.1.1 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 [28]

1. *Completeness:* A complete polynomial[1] 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.

5.1.2 Various Model Tessellations

We will give two possible tessellations of the material coordinate system u $= (u, v, 1)$ into finite element domains for the case of a deformable superquadric ellipsoid. The first is illustrated in Fig. 5.1. The need for quadrilateral and triangular elements is evident. Equation (3.7) 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 later we use the second representation shown in Fig. 5.2 where each of the rectangular elements has been replaced by two triangular elements. We use such a representation in applications requiring shape accuracy.

[1] In 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 $l = (p+1)(p+2)(p+3)/6$.

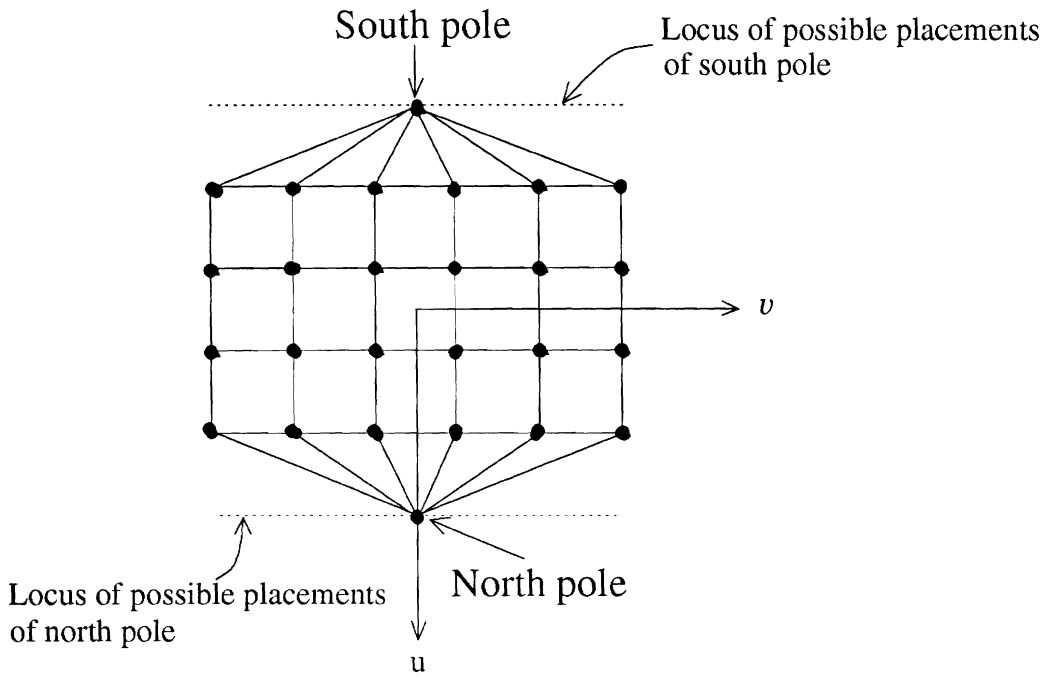


Figure 5.1: Model tessellation in material coordinates: rectangular and triangular elements.

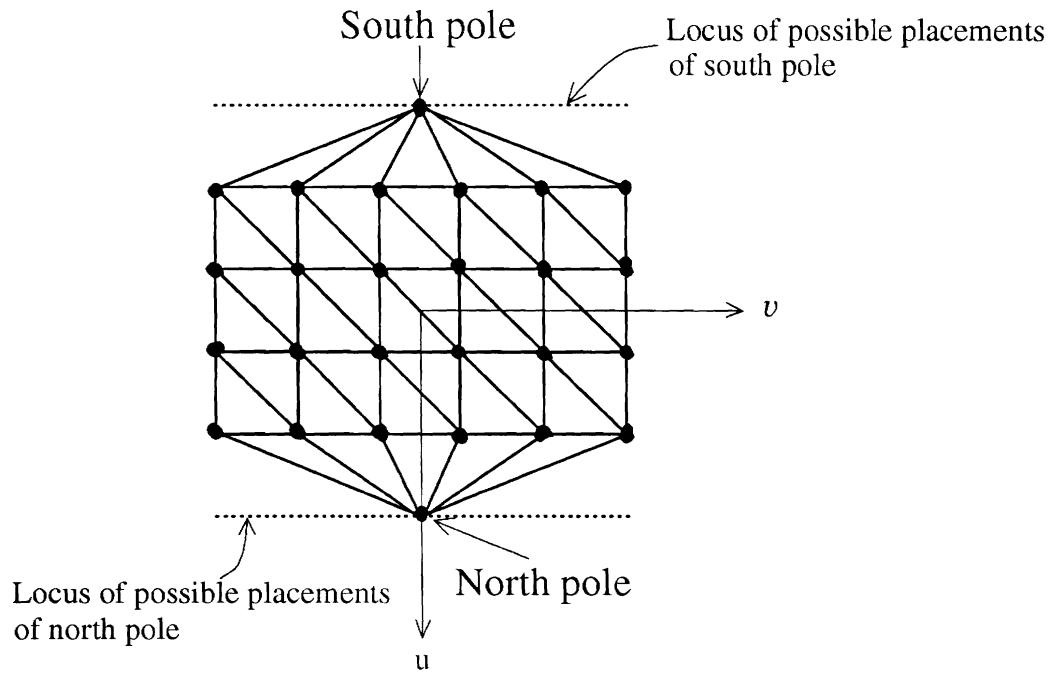


Figure 5.2: Model tessellation in material coordinates: triangular elements.

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 thesis. We will be using elements with C^0 or C^1 continuity[2] 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 (ξ, η). Furthermore, due to the model discretization in material coordinates u, we give the relationship between the reference and the material coordinates.

5.1.3 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. 5.1 we use bilinear quadrilateral and south and north linear triangular elements. For the second tessellation shown in Fig. 5.2 we use north, south and mid-region linear triangular elements.

Bilinear Quadrilateral Elements

The nodal shape functions of the bilinear quadrilateral element (Fig. 5.3) are

$$N_i(\xi, \eta) = \frac{1}{4}(1 + \xi\xi_i)(1 + \eta\eta_i), \tag{5.4}$$

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

$$\xi = \frac{2}{a}(u - u_c), \qquad \eta = \frac{2}{b}(v - v_c), \tag{5.5}$$

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

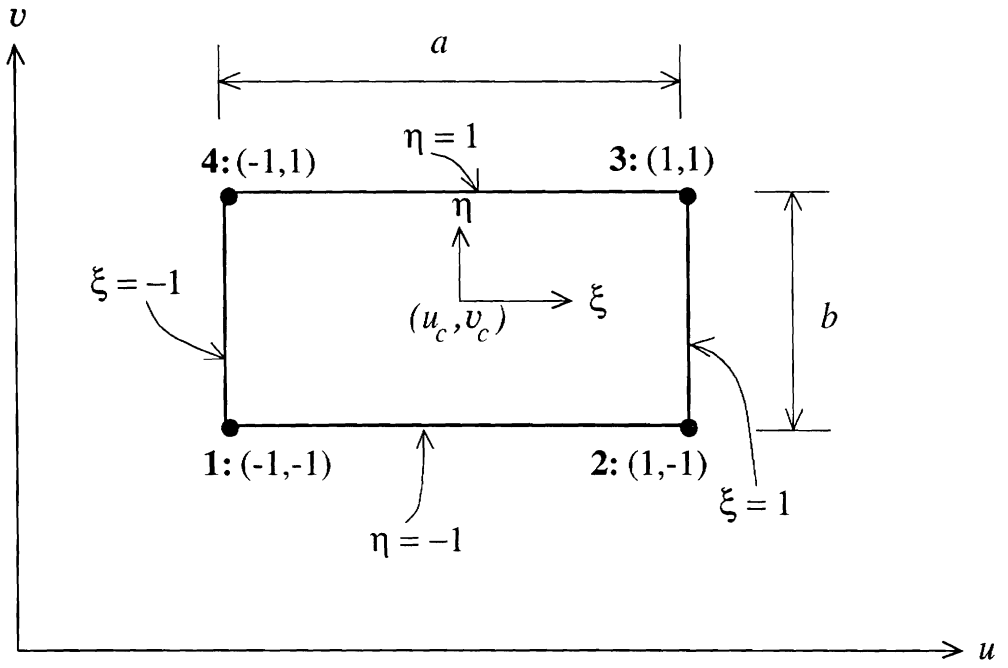


Figure 5.3: Bilinear quadrilateral element. The four nodes are numbered.

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

$$\frac{\partial N_i}{\partial u} = \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial u} + \frac{\partial N_i}{\partial \eta}\frac{\partial \eta}{\partial u} = \frac{1}{2a}\xi_i(1 + \eta\eta_i), \tag{5.6}$$

$$\frac{\partial N_i}{\partial v} = \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial v} + \frac{\partial N_i}{\partial \eta}\frac{\partial \eta}{\partial v} = \frac{1}{2b}(1 + \xi\xi_i)\eta_i, \tag{5.7}$$

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

$$\iint_{E_j} f(u, v)\, du\, dv = \int_{-1}^{1}\int_{-1}^{1} f(\xi, \eta)\frac{ab}{4}\, d\xi\, d\eta. \tag{5.8}$$

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

North Pole Linear Triangular Elements

The nodal shape functions for the north pole linear triangular element (Fig. 5.4) are

$$N_1(\xi,\eta) = 1-\xi-\eta, \tag{5.9}$$

$$N_2(\xi,\eta) = \xi, \tag{5.10}$$

$$N_3(\xi,\eta) = \eta. \tag{5.11}$$

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

$$\xi = \frac{1}{a}(u-u_1), \tag{5.12}$$

$$\eta = \frac{1}{b}(v-v_1), \tag{5.13}$$

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 (5.6) and (5.7) yields

$$\frac{\partial N_1}{\partial u} = -\frac{1}{a}; \qquad \frac{\partial N_2}{\partial u} = \frac{1}{a}; \qquad \frac{\partial N_3}{\partial u} = 0; \tag{5.14}$$

$$\frac{\partial N_1}{\partial v} = -\frac{1}{b}; \qquad \frac{\partial N_2}{\partial v} = 0; \qquad \frac{\partial N_3}{\partial v} = \frac{1}{b}; \tag{5.15}$$

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

$$\iint_{E_j} f(u,v)\,du\,dv = \int_0^1 \int_0^{(1-\eta)} f(\xi,\eta)ab\,d\xi\,d\eta. \tag{5.16}$$

We approximate such integrals using Radau quadrature rules.

South Pole Linear Triangular Elements

The nodal shape functions for the south pole linear triangular element (Fig. 5.5) are

$$N_1(\xi,\eta) = 1-\xi, \tag{5.17}$$

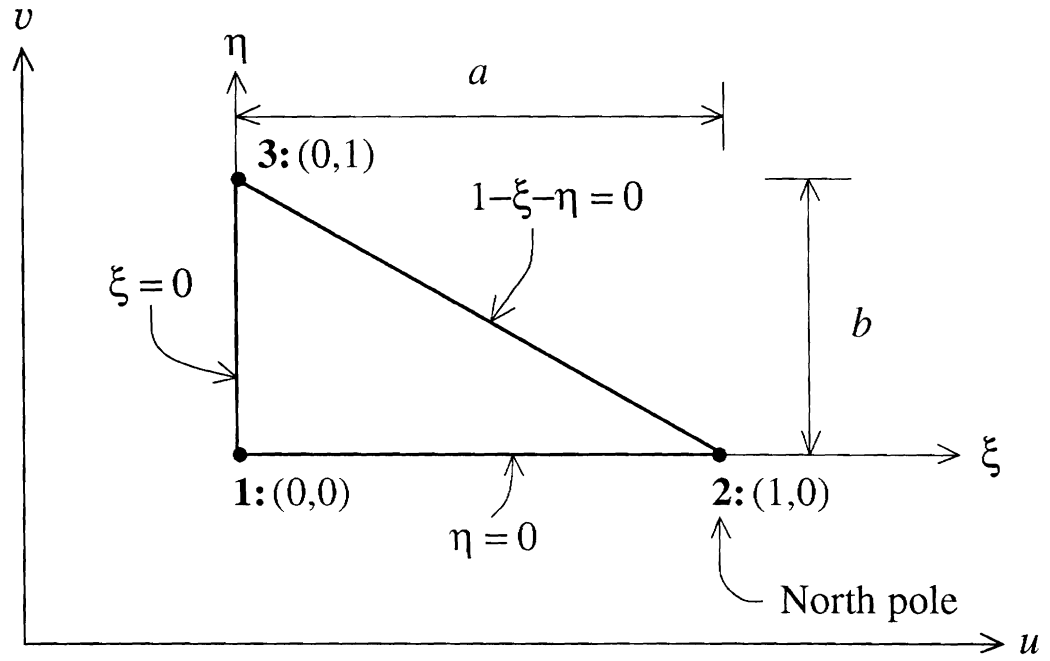


Figure 5.4: North pole linear triangular element. The three nodes are numbered.

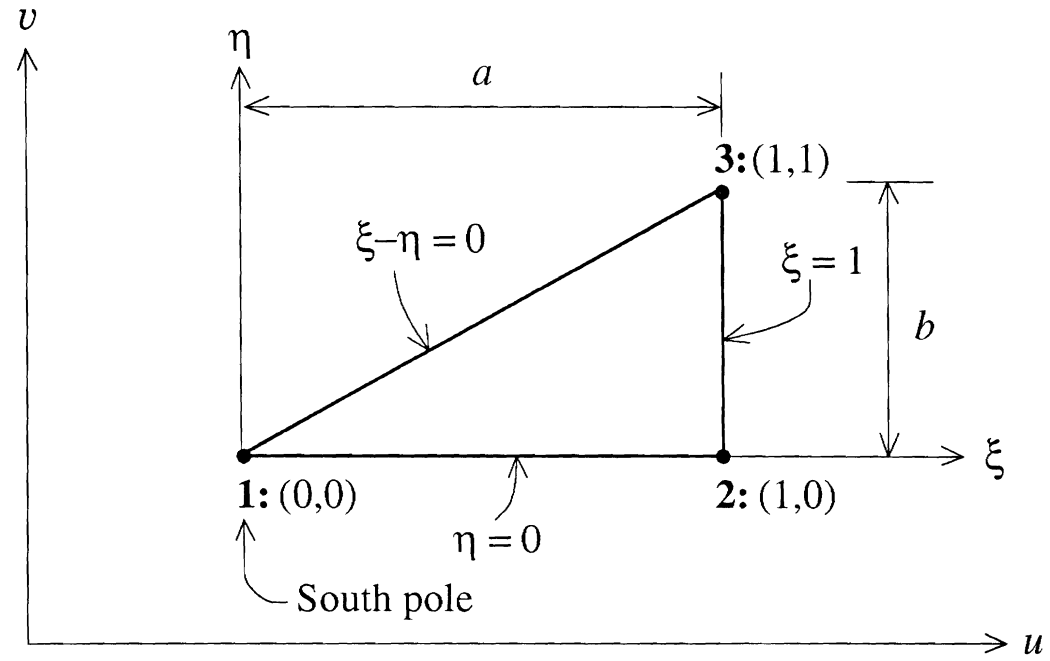


Figure 5.5: South pole linear triangular element. The three nodes are numbered.

$$N_2(\xi,\eta) = \xi - \eta, \tag{5.18}$$

$$N_3(\xi,\eta) = \eta. \tag{5.19}$$

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

$$\xi = \frac{1}{a}(u - u_1), \tag{5.20}$$

$$\eta = \frac{1}{b}(v - v_1), \tag{5.21}$$

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 (5.6) and (5.7) yields

$$\frac{\partial N_1}{\partial u} = -\frac{1}{a}; \qquad \frac{\partial N_2}{\partial u} = \frac{1}{a}; \qquad \frac{\partial N_3}{\partial u} = 0; \tag{5.22}$$

$$\frac{\partial N_1}{\partial v} = 0; \qquad \frac{\partial N_2}{\partial v} = -\frac{1}{b}; \qquad \frac{\partial N_3}{\partial v} = \frac{1}{b}; \tag{5.23}$$

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

$$\iint_{E_j} f(u,v)\,du\,dv = \int_0^1 \int_0^\xi f(\xi,\eta)ab\,d\eta\,d\xi. \tag{5.24}$$

We approximate such integrals using Radau quadrature rules.

Mid-Region Triangular Elements

The nodal shape functions for the mid-region linear triangular element (Fig. 5.6) are

$$N_1(\xi,\eta) = 1 - \eta, \tag{5.25}$$

$$N_2(\xi,\eta) = \xi + \eta - 1, \tag{5.26}$$

$$N_3(\xi,\eta) = 1 - \xi. \tag{5.27}$$

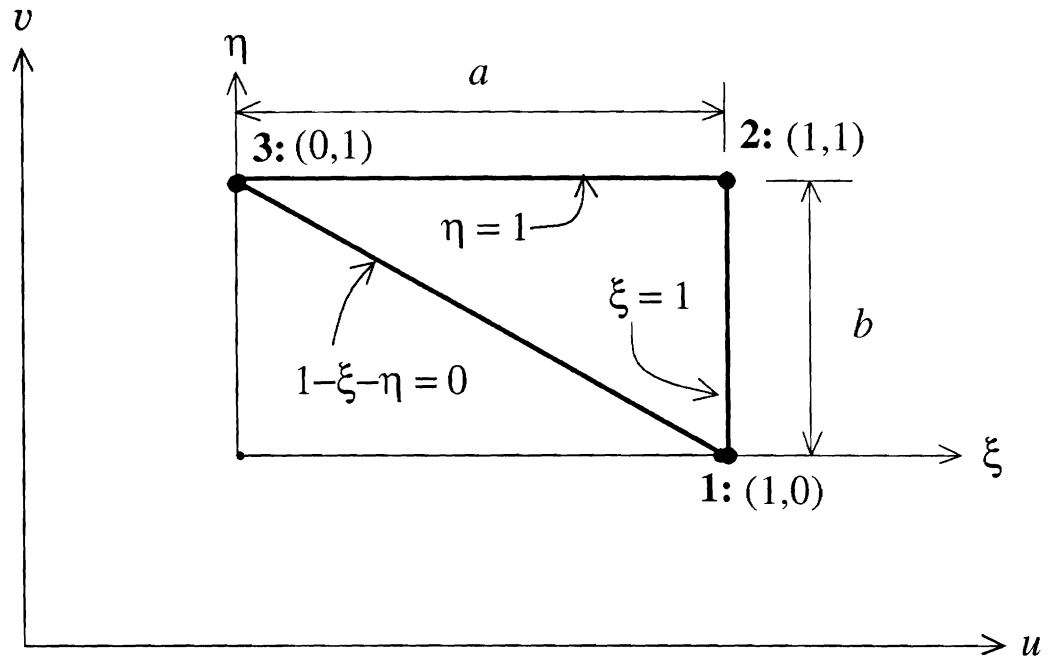


Figure 5.6: Mid-region triangular element. The three nodes are numbered.

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

$$\xi = \frac{1}{a}(u - u_3), \tag{5.28}$$

$$\eta = \frac{1}{b}(v - v_1), \tag{5.29}$$

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 (5.6) and (5.7) yields

$$\frac{\partial N_1}{\partial u} = 0; \qquad \frac{\partial N_2}{\partial u} = \frac{1}{a}; \qquad \frac{\partial N_3}{\partial u} = -\frac{1}{a}; \tag{5.30}$$

$$\frac{\partial N_1}{\partial v} = -\frac{1}{b}; \qquad \frac{\partial N_2}{\partial v} = \frac{1}{b}; \qquad \frac{\partial N_3}{\partial v} = 0; \tag{5.31}$$

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

$$\iint_{E_j} f(u,v)\,du\,dv = \int_0^1 \int_{(1-\xi)}^1 f(\xi,\eta)ab\,d\eta\,d\xi. \tag{5.32}$$

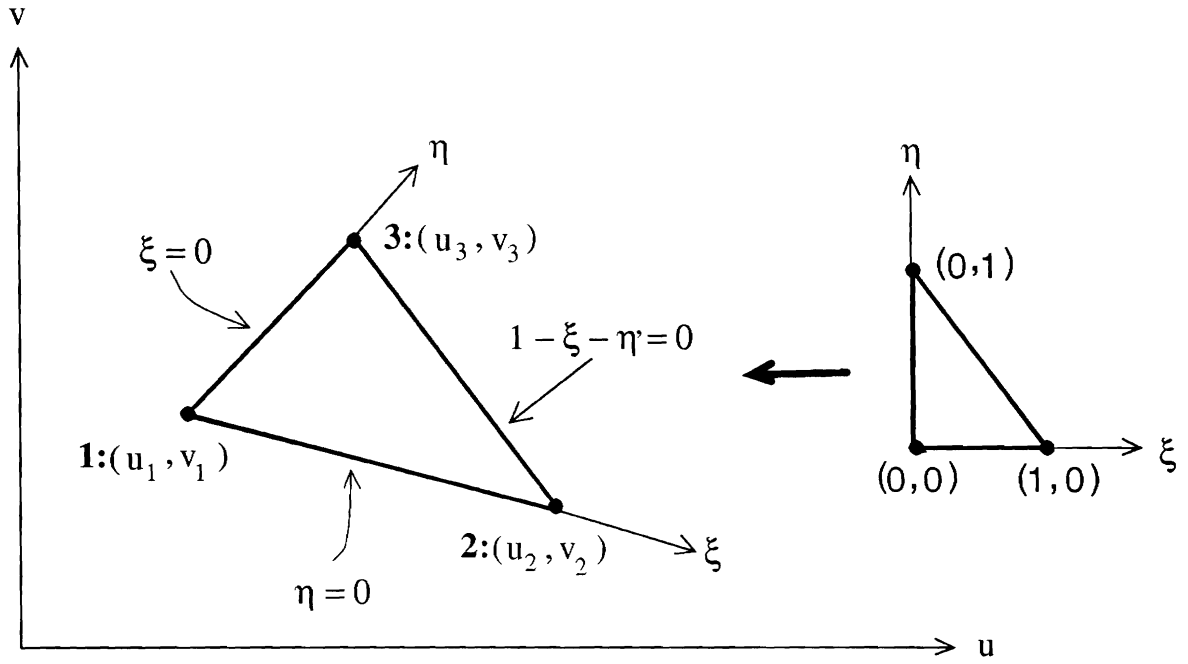


Figure 5.7: C^1 Continuous Triangular Element. The three nodes are numbered.

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.

5.1.4 C^1 Triangular Elements

Such elements can be used for the tessellation shown in Fig. 5.2 for problems requiring C^1 continuity.

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

$$u = (1-\xi-\eta)u_3 + \xi u_1 + \eta u_2 \tag{5.33}$$

$$v = (1-\xi-\eta)v_3 + \xi v_1 + \eta v_2, \tag{5.34}$$

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

$$\mathbf{q}_{d_i}(t) = \left[\mathbf{d}_i^\top, \frac{\partial \mathbf{d}_i^\top}{\partial u}, \frac{\partial \mathbf{d}_i^\top}{\partial v}, \frac{\partial^2 \mathbf{d}_i^\top}{\partial u^2}, \frac{\partial^2 \mathbf{d}_i^\top}{\partial u \partial v}, \frac{\partial^2 \mathbf{d}_i^\top}{\partial v^2}\right]^\top. \tag{5.35}$$

Concatenating the $\mathbf{q}_{d_i}$ at each of the three nodes of element j, the 18-dimensional element nodal vector $\mathbf{q}_d^j = [\mathbf{q}_{d_1}^\top, \mathbf{q}_{d_2}^\top, \mathbf{q}_{d_3}^\top]^\top$ is obtained.

The 18 nodal shape functions $\mathbf{N}_i(\xi, \eta)$ are given by the following formulas. For node 1

$$\mathbf{N}_1 = \lambda^2(10\lambda - 15\lambda^2 + 6\lambda^3 + 30\xi\eta(\xi+\eta)), \qquad \mathbf{N}_2 = \xi\lambda^2(3 - 2\lambda - 3\xi^2 + 6\xi\eta),$$
$$\mathbf{N}_3 = \eta\lambda^2(3 - 2\lambda - 3\eta^2 + 6\xi\eta). \qquad \mathbf{N}_4 = \tfrac{1}{2}\xi^2\lambda^2(1 - \xi + 2\eta),$$
$$\mathbf{N}_5 = \xi\eta\lambda^2, \qquad \mathbf{N}_6 = \tfrac{1}{2}\eta^2\lambda^2(1 + 2\xi - \eta),$$

for node 2

$$\mathbf{N}_7 = \xi^2(10\xi - 15\xi^2 + 6\xi^3 + 15\eta^2\lambda), \qquad \mathbf{N}_8 = \tfrac{\xi^2}{2}(-8\xi + 14\xi^2 - 6\xi^3 - 15\eta^2\lambda),$$
$$\mathbf{N}_9 = \tfrac{\xi^2\eta}{2}(6 - 4\xi - 3\eta - 3\eta^2 + 3\xi\eta), \qquad \mathbf{N}_{10} = \tfrac{\eta^2}{4}(2\xi(1-\xi)^2 + 5\eta^2\lambda),$$
$$\mathbf{N}_{11} = \tfrac{\xi^2\eta}{2}(-2 + 2\xi + \eta + \eta^2 - \xi\eta), \qquad \mathbf{N}_{12} = \tfrac{\xi^2\eta^2\lambda}{4} + \tfrac{\xi^3\eta^2}{2},$$

and for node 3

$$\mathbf{N}_{13} = \eta^2(10\eta - 15\eta^2 + 6\eta^3 + 15\xi^2\lambda), \qquad \mathbf{N}_{14} = \tfrac{\xi\eta^2}{2}(6 - 3\xi - 4\eta - 3\xi^2 + 3\xi\eta),$$
$$\mathbf{N}_{15} = \tfrac{\eta^2}{2}(-8\eta + 14\eta^2 - 6\eta^3 - 15\xi^2\lambda), \qquad \mathbf{N}_{16} = \tfrac{\xi^2\eta^2\lambda}{4} + \tfrac{\xi^2\eta^3}{2},$$
$$\mathbf{N}_{17} = \tfrac{\xi\eta^2}{2}(-2 + \xi + 2\eta + \xi^2 - \xi\eta), \qquad \mathbf{N}_{18} = \tfrac{\eta^2}{4}(2\eta(1-\eta)^2 + 5\xi^2\lambda),$$

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.,

$$\iint_{E_j} f(u, v)\, du\, dv. \tag{5.36}$$

The element is complete up to fourth-order and its shape functions contain three fifth-order terms [16]. The shape functions are C^∞ continuous within elements and they ensure C^1 continuity between elements. Since (4.35) contains up to second order derivatives, the element is conforming. Therefore such an element is appropriate for a thin plate under tension deformation energy [47].

5.1.5 Approximation of the Lagrange Equations

We may approximate the Lagrange equations of motion (4.13) 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 4) are assembled from matrices computed within each of the elements[3] , i.e.,

$$\mathbf{M} = \sum_j \mathbf{M}^j_{q\times q}; \quad \mathbf{D} = \sum_j \mathbf{D}^j_{q\times q}; \quad \mathbf{K} = \sum_j \mathbf{K}^j_{q\times q}; \quad \mathbf{f}_q = \sum_j \mathbf{f}^j_q; \quad \mathbf{g}_q = \sum_j \mathbf{g}^j_q, \tag{5.37}$$

where $\mathbf{M}^j_{q\times q}$, $\mathbf{D}^j_{q\times q}$, $\mathbf{K}^j_{q\times q}$, $\mathbf{f}^j_q$ and $\mathbf{g}^j_q$ are the appropriately expanded [29] 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

[3]The 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.

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,

$$\mathbf{M}_{dd} = \int \mu \mathbf{N}^{j^T} \mathbf{N}^j \, d\mathrm{u}, \tag{5.38}$$

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

5.1.6 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 (4.34).

We discretize the model in material coordinates u using finite elements. We can derive $\mathbf{K}_{dd}$ as an assembly of the local stiffness matrices $\mathbf{K}^j_{dd}$ associated with each element domain $E_j \subset \mathrm{u}$. Since $\mathbf{d}(\mathrm{u}, t) = \mathbf{d}^j(\mathrm{u}, t) = [d^j_1(\mathrm{u}, t), d^j_2(\mathrm{u}, t), d^j_3(\mathrm{u}, t)]^\mathsf{T}$, we can rewrite the membrane spline deformation energy (4.34) on E_j as the sum of component energies

$$\mathcal{E}^j(\mathbf{d}) = \mathcal{E}^j(d^j_1) + \mathcal{E}^j(d^j_2) + \mathcal{E}^j(d^j_3), \tag{5.39}$$

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

$$\mathcal{E}^j(d^j_k) = \int_{E_j} w^j_{10} \left(\frac{\partial d_k}{\partial u}\right)^2 + w^j_{01} \left(\frac{\partial d_k}{\partial v}\right)^2 + w^j_{00} d_k^2 \, d\mathrm{u}. \tag{5.40}$$

In accordance to the theory of elasticity, (5.40) can be written in the form

$$\mathcal{E}^j(d^j_k) = \int_{E_j} \boldsymbol{\sigma}^{j^\mathsf{T}}_k \boldsymbol{\epsilon}^j_k \, d\mathrm{u}, \tag{5.41}$$

where

$$\boldsymbol{\epsilon}^j_k = \left[\frac{\partial d^j_k}{\partial u}, \frac{\partial d^j_k}{\partial v}, d^j_k\right]^\mathsf{T} \tag{5.42}$$

is the strain vector and

$$\boldsymbol{\sigma}_k^j = \mathbf{D}_k^j \boldsymbol{\epsilon}_k^j = \begin{pmatrix} w_{10}^j & 0 & 0 \\ 0 & w_{01}^j & 0 \\ 0 & 0 & w_{00}^j \end{pmatrix} \boldsymbol{\epsilon}_k^j \tag{5.43}$$

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

$$\boldsymbol{\sigma}^j = \mathbf{D}^j \boldsymbol{\epsilon}^j = \begin{pmatrix} \mathbf{D}_1^j & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2^j & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_3^j \end{pmatrix} \begin{pmatrix} \boldsymbol{\epsilon}_1^j \\ \boldsymbol{\epsilon}_2^j \\ \boldsymbol{\epsilon}_3^j \end{pmatrix}, \tag{5.44}$$

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, ..., n, where n is the number of nodes associated with element E_j. Hence, we can write (5.42) as

$$\boldsymbol{\epsilon}_k^j = \sum_{i=1}^{n} \boldsymbol{\gamma}_i^j (q_{d_k}^j)_i = \boldsymbol{\Gamma}_k^j \mathbf{q}_{d_k}^j, \tag{5.45}$$

where

$$\boldsymbol{\gamma}_i^j = \left[\frac{\partial N_i^j}{\partial u}, \frac{\partial N_i^j}{\partial v}, N_i^j \right]^\mathsf{T}, \tag{5.46}$$

$$\boldsymbol{\Gamma}_k^j = \left(\boldsymbol{\gamma}_1^j \, \boldsymbol{\gamma}_2^j \, \ldots \, \boldsymbol{\gamma}_n^j \right) \tag{5.47}$$

and

$$\mathbf{q}_{d_k}^j = [(q_{d_k}^j)_1, (q_{d_k}^j)_2, \ldots, (q_{d_k}^j)_n]^\mathsf{T}. \tag{5.48}$$

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

$$\boldsymbol{\epsilon}^j = \begin{pmatrix} \boldsymbol{\epsilon}_1^j \\ \boldsymbol{\epsilon}_2^j \\ \boldsymbol{\epsilon}_3^j \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Gamma}_1^j & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Gamma}_2^j & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Gamma}_3^j \end{pmatrix} \begin{pmatrix} \mathbf{q}_{d_1}^j \\ \mathbf{q}_{d_2}^j \\ \mathbf{q}_{d_3}^j \end{pmatrix} = \boldsymbol{\Gamma}^j \mathbf{q}_d^j, \tag{5.49}$$

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

$$\mathbf{K}_{dd}^j = \int_{E_j} \mathbf{\Gamma}^{j\top} \mathbf{D}^j \mathbf{\Gamma}^j \, du = \mathrm{diag}\Big(\int_{E_j} \mathbf{\Gamma}_k^{j\top} \mathbf{D}_k^j \mathbf{\Gamma}_k^j \, du \Big) = \mathrm{diag}(\mathbf{K}_{dd_k}^j). \tag{5.50}$$

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

$$\boldsymbol{\epsilon}_d^j = \left[d_k^j, \frac{\partial d_k^j}{\partial u}, \frac{\partial d_k^j}{\partial v}, \frac{\partial^2 d_k^j}{\partial u^2}, \frac{\partial^2 d_k^j}{\partial u \partial v}, \frac{\partial^2 d_k^j}{\partial v^2} \right]^\top, \tag{5.51}$$

$$\mathbf{D}_k^j = \begin{bmatrix} w_{00}^j & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{10}^j & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{01}^j & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{20}^j & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{11}^j & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{02}^j \end{bmatrix}, \tag{5.52}$$

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

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

$$\mathbf{K}_{44} = ab(\frac{w_1}{3a^2} + \frac{w_1}{3b^2} + \frac{w_0}{9}), \tag{5.53}$$

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

$$\begin{aligned}
\mathbf{K}_{11} &= \frac{ab}{2}(\frac{w_1}{a^2} + \frac{w_0}{6}), \\
\mathbf{K}_{12} = \mathbf{K}_{21} &= \frac{ab}{2}(-\frac{w_1}{a^2} + \frac{w_0}{12}), \\
\mathbf{K}_{13} = \mathbf{K}_{31} &= \frac{ab}{24}w_0, \\
\mathbf{K}_{22} &= \frac{ab}{2}(\frac{w_1}{b^2} + \frac{w_1}{a^2} + \frac{w_0}{6}), \\
\mathbf{K}_{23} = \mathbf{K}_{32} &= \frac{ab}{2}(-\frac{w_1}{b^2} + \frac{w_0}{12}), \\
\mathbf{K}_{33} &= \frac{ab}{2}(\frac{w_1}{b^2} + \frac{w_0}{6}).
\end{aligned} \tag{5.54}$$

5.2 Summary

In this chapter we gave an overview of the finite element method that we employ to discretize the Lagrange equations of motion. In particular, we gave 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 a given problem in vision or graphics applications.

Chapter 6

Applied Forces

This chapter describes various techniques for computing external forces and applying them to deformable models. In computer vision applications such forces originate from datapoints or image potentials, and are assigned to points on the deformable model [70]. 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) [66]. We also describe our force-based technique for shape and nonrigid motion estimation.

6.1 Computer Vision Applications

In the dynamic model fitting process the data are transformed into an externally applied force distribution $\mathbf{f}(\mathrm{u}, t)$. Using (4.48), 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 forces[1] obtained through gradients of potential functions. For 3D range data we assign long-range forces[2] based on distances between data points and the model's surface.

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

[2]Long 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.

6.1.1 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 $I(x, y)$, we construct the potential function [70]

$$P(x, y) = \|\nabla(G_\sigma * I)\|, \tag{6.1}$$

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

$$\mathbf{f} = \beta \nabla P(\Pi \mathbf{x}) \tag{6.2}$$

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 5 for more details).

Algorithm 1

We select those nodes from the finite elements whose vertical distance from the image plane shown in Fig. 6.1, is less than an experimentally defined threshold ϵ. To each of those nodes we apply the forces (6.2).

Algorithm 2

We first compute the intersection points between the edges of the finite elements and the image plane shown in Fig. 6.2. We then assign to each intersection point $\mathbf{u}_i$ the force (6.2). We then extrapolate this force to each of the two corresponding

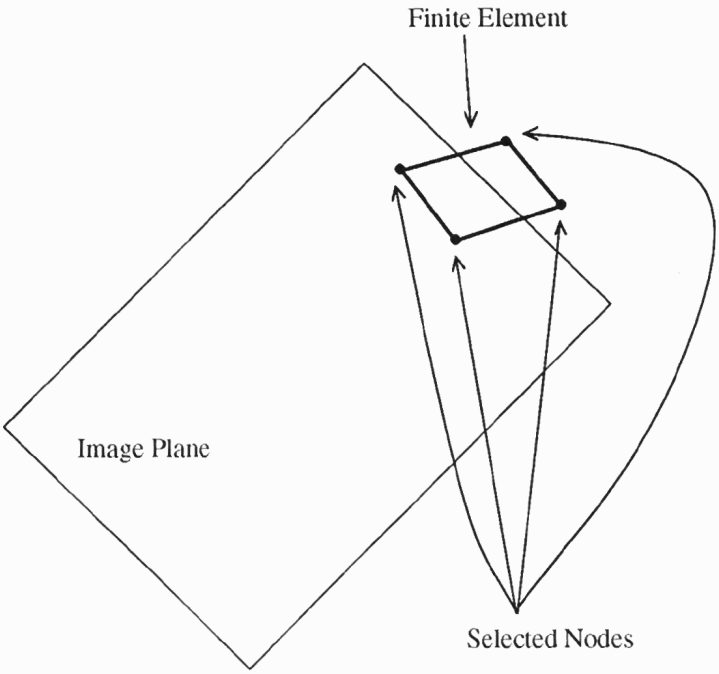


Figure 6.1: Application of image forces to nodes.

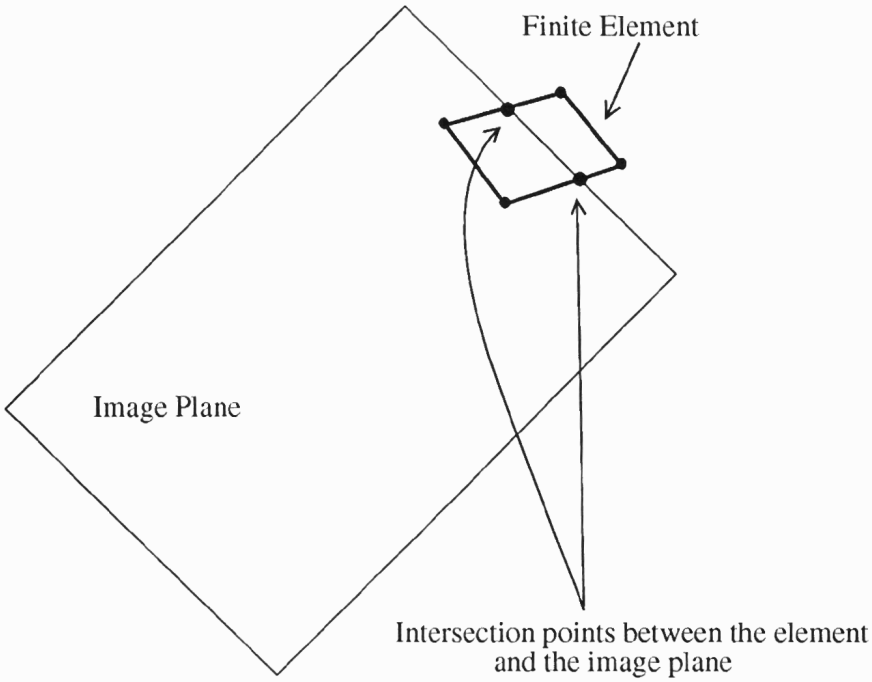


Figure 6.2: Accurate application of image forces to nodes.

nodes of the triangular element using the shape functions

$$\mathbf{f}_k = N_k(\mathrm{u}_i)\mathbf{f}(\mathrm{u}_i), \tag{6.3}$$

where $k \in (1,2,3)$, 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 (6.2) 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 $k = 1,\ldots,4$ that surround (x,y). We then consider these pixels as the nodes of the quadrilateral finite element of Fig. 5.3, with $a = b = 1$, in order to define a bilinear interpolant in the region between the pixels; i.e., using (5.4) and (5.5), the interpolant is given by

$$\nabla P(x,y) = \sum_{k=1}^{4} N_k(2(x-x_c), 2(x-y_c))\nabla P_k, \tag{6.4}$$

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.

6.1.2 Long-range Forces

Alternatively, we may define long-range forces

$$\mathbf{f}(\mathrm{u}_r) = \beta \left\| \mathbf{z} - \mathbf{x}(\mathrm{u}_z) \right\|, \tag{6.5}$$

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

$$\mathbf{x}(\mathrm{u}_z) = \sum_{i=1}^{4} N_i(2(u_z - u_c)/a, 2(v_z - v_c)/b)\mathbf{x}_i, \tag{6.6}$$

where the $\mathbf{x}_i$ are the nodal positions (see (5.4) and (5.5)). The equivalent forces on each of the four nodes are

$$\mathbf{f}_i = N_i(2(u_z - u_c)/a, 2(v_z - v_c)/b)\mathbf{f}(\mathrm{u}_z). \tag{6.7}$$

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 u_z to minimize the distance d between a given datapoint $\mathbf{z}$ and the model, where

$$d(\mathrm{u}) = \|\mathbf{z} - \mathbf{x}(\mathrm{u})\|. \tag{6.8}$$

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

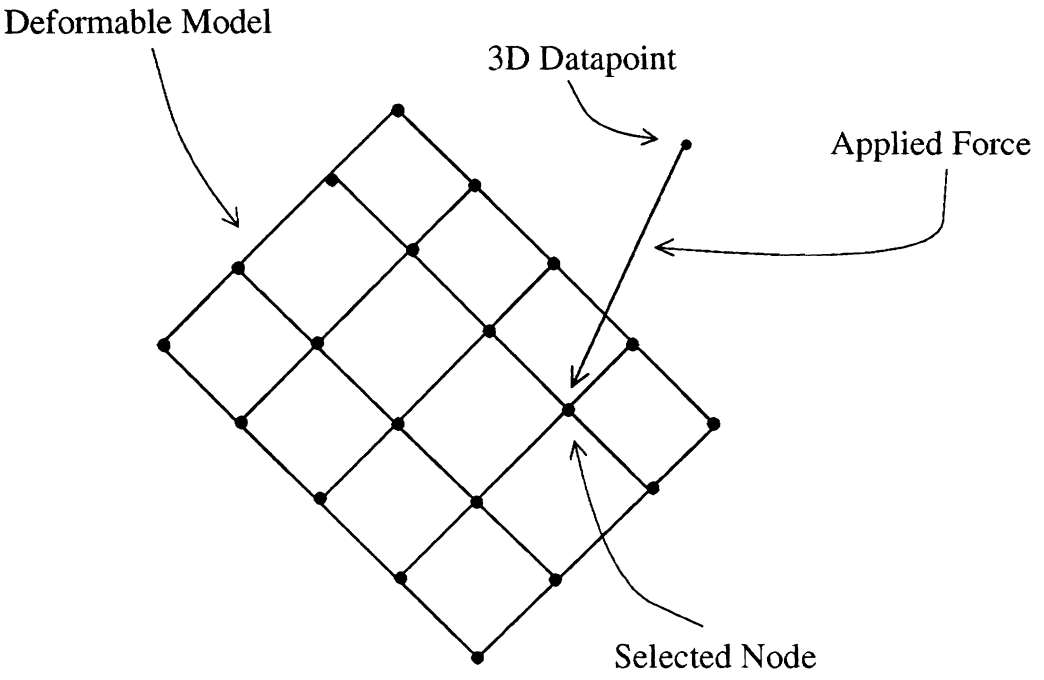


Figure 6.3: Force application to the model node with minimal distance from a 3D datapoint.

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.

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. 6.3. 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. 6.4), we project the force at each time step onto the unit tangent vectors $(\partial \mathbf{x}/\partial u)/\left\|\partial \mathbf{x}/\partial u\right\|$ and $(\partial \mathbf{x}/\partial v)/\left\|\partial \mathbf{x}/\partial v\right\|$ to the surface at the current point of influence $P_0 = \mathrm{u}_0$, and we migrate the point in

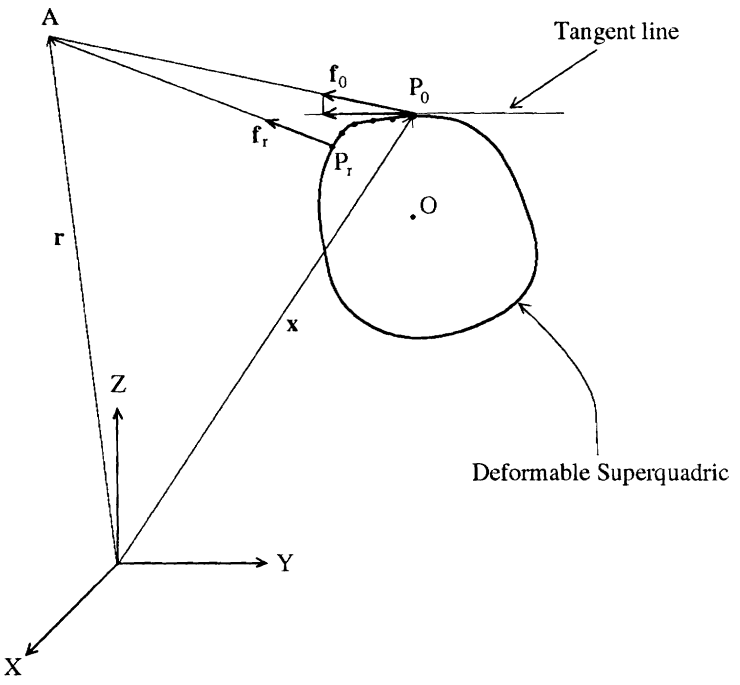


Figure 6.4: Migration of data force point of influence over model surface.

the 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 $d(\mathrm{u})$ from a given datapoint. We then use a minimization procedure[3] within each of the elements that the previously selected model node belongs as shown in Fig. 6.5. We perform the minimization within each element j using the element's shape functions

$$d(\mathrm{u}) = \min^j \|\mathbf{z} - \mathbf{x}(\mathrm{u})\| = \min^j \left\| \mathbf{z} - \sum_{i=1}^{n} N_i(\mathrm{u})\mathbf{x}_i \right\|, \tag{6.9}$$

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

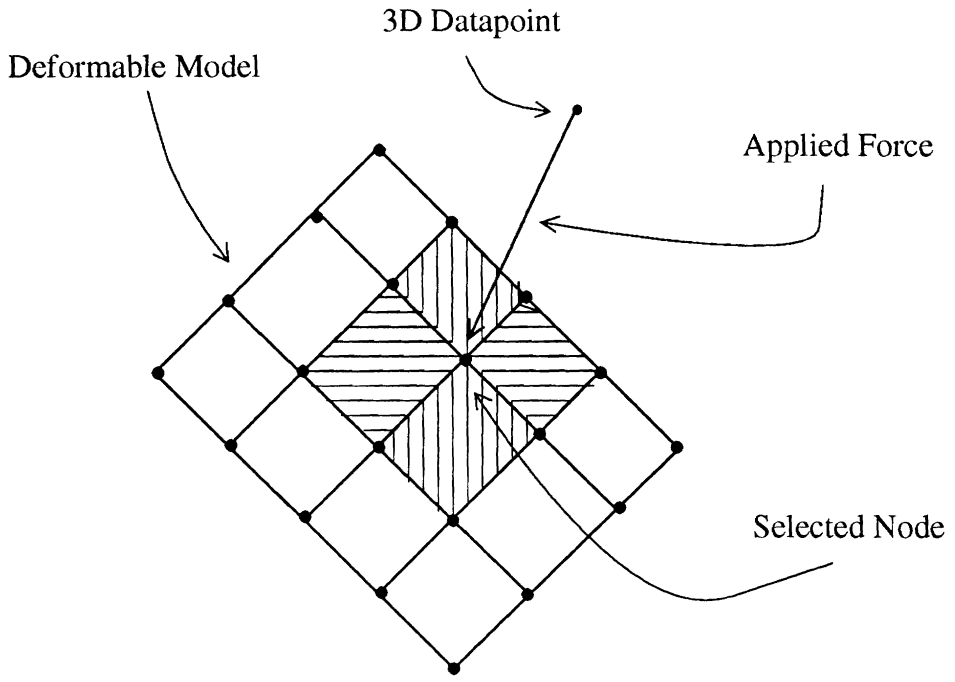


Figure 6.5: Force application to a model point with minimal distance from a 3D datapoint. The model point lies within the elements (shaded) which share the model node with minimal distance from the 3D datapoint.

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 (6.9). 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 $\Theta(mn)$.

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. 6.6. The intersection of this line with the model surface defines a point $\mathbf{x}(\mathrm{u})$ on the model to which this force will be exerted. The force is then computed using (6.5). This kind of force is only suitable for convex models because otherwise the radial line might intersect

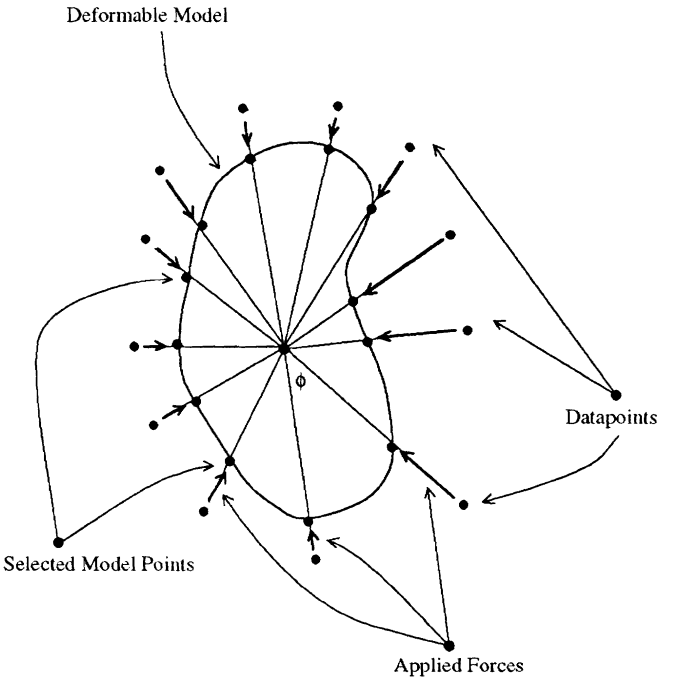


Figure 6.6: Application of radial forces to model points.

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.

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.

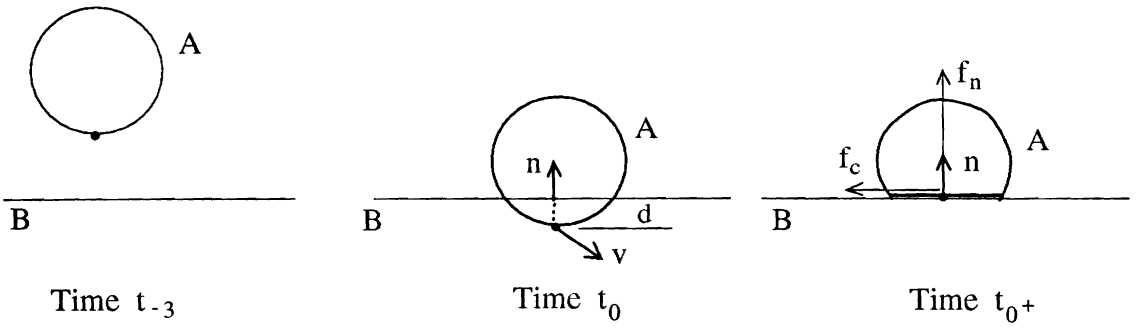


Figure 6.7: Collision of a deformable with a rigid model.

6.2 Computer Graphics Applications

We compute forces stemming from collisions of deformable with rigid or deformable models using the following two algorithms.

6.2.1 Collisions of Deformable with Rigid Models

Suppose that a deformable model is colliding with a rigid model as illustrated in Fig. 6.7. At every time step we check whether a node has penetrated the rigid model. In Fig. 6.7 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.

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

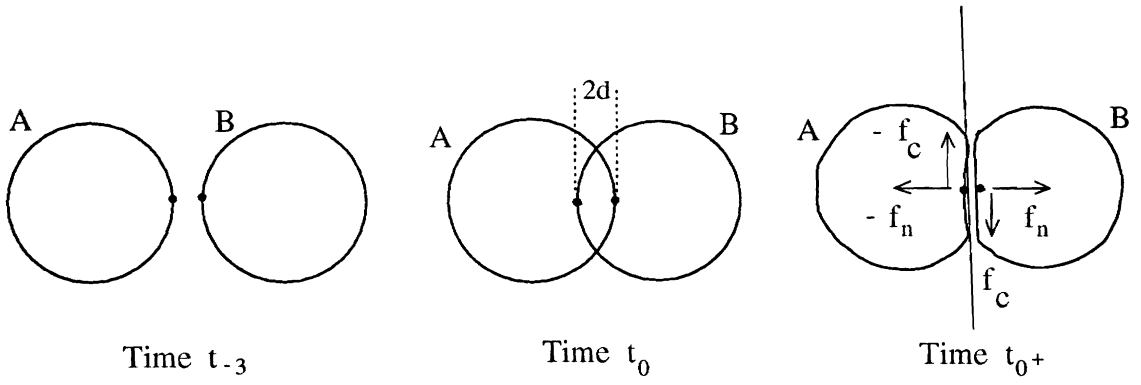


Figure 6.8: Collision of two deformable models.

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

$$\mathbf{f}_{q_e} = \mathbf{K}\mathbf{q}_d, \tag{6.10}$$

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

$$\mathbf{f}_n = a(\mathbf{f}_i \cdot \mathbf{n})\mathbf{n}, \tag{6.11}$$

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

$$\mathbf{f}_c = -\beta_f \|\mathbf{f}_n\| \frac{\mathbf{v}_p}{\|\mathbf{v}_p\|}, \tag{6.12}$$

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

6.2.2 Collisions between Deformable Models

Let's assume that two deformable models will penetrate one another as shown in Fig. 6.8. 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. 6.8).

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

$$\|\mathbf{f}_c\| = b_f \|\mathbf{f}_n\| \tag{6.13}$$

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.

6.3 Force-Based Estimation

We present our force-based estimation technique for shape and nonrigid motion in computer vision 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 (4.49) driven by these forces.

More specifically, let the observation vector $\mathbf{z}(t)$ denote time-varying input data. Using (4.3) we can relate $\mathbf{z}(t)$ to the model's state vector $\mathbf{u}(t)$ as follows:

$$\mathbf{z} = \mathbf{h}(\mathbf{u}) + \mathbf{v}, \tag{6.14}$$

where $\mathbf{v}(t)$ represents uncorrelated measurement errors as a zero mean white noise process with known covariance $\mathbf{V}(t)$, i.e., $\mathbf{v}(t) \sim \mathbf{N}(\mathbf{0}, \mathbf{V}(t))$[5]. In force-based estimation, the rate of change of the estimated model state vector $\hat{\mathbf{u}}$ is given from (4.49) by

$$\dot{\hat{\mathbf{u}}} = \mathbf{F}\hat{\mathbf{u}} + \mathbf{g}. \tag{6.15}$$

Furthermore, according to (4.4), (4.3), and (6.14), the driving function $\mathbf{g}$ in (4.50) includes the term

$$\mathbf{f}_q = \mathbf{H}^\top \mathbf{V}^{-1}(\mathbf{z} - \mathbf{h}(\hat{\mathbf{u}})), \tag{6.16}$$

where

$$\mathbf{H} = \left.\frac{\partial \mathbf{h}(\mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{u}=\hat{\mathbf{u}}}. \tag{6.17}$$

Matrix $\mathbf{H}$ maps the 3-space observation forces $(\mathbf{z} - \mathbf{h}(\hat{\mathbf{u}}))$ scaled by $\mathbf{V}^{-1}$, to $\mathbf{q}$-

[5]For 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 (4.3) evaluated at $\mathbf{u}_k$. If $\mathbf{z}$ includes velocity estimates, then we also use (4.4)

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.

6.4 Summary

In this chapter we presented techniques for converting visual data into forces that can be applied to deformable models in data fitting scenarios. Secondly, for computer animation purposes, we described two algorithms which use the elastic properties of our models to calculate forces stemming from collisions and friction against impenetrable solid or deformable surfaces. Finally, we presented our force-based algorithm used in shape and nonrigid motion estimation applications.

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

Chapter 7

Model Implementation

In this chapter we first present a technique to integrate the motion equations (4.13) and (4.51) and describe the initialization of our models. Second we present various experiments in computer vision and computer graphics applications. In computer vision we present experiments with real and synthetic data to demonstrate the performance of our framework in deformable model fitting to static 2D and 3D data and motion 3D data, using our force-based shape estimation algorithm. 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.

7.1 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 response[1] by employing standard numerical methods to integrate (4.13). The simplest method is the first-order Euler procedure which updates the

[1]See the Introduction for definitions.

state vector over a time step Δt from time t to $t + \Delta t$ according to the formulas

$$\begin{aligned} \ddot{\mathbf{q}}^{(t)} &= \mathbf{M}^{-1}(\mathbf{g}_q^{(t)} + \mathbf{f}_q^{(t)} - \mathbf{K}\mathbf{q}^{(t)} - \mathbf{D}\dot{\mathbf{q}}^{(t)}) \\ \dot{\mathbf{q}}^{(t+\Delta t)} &= \dot{\mathbf{q}}^{(t)} + \Delta t\ \ddot{\mathbf{q}}^{(t)} \\ \mathbf{q}^{(t+\Delta t)} &= \mathbf{q}^{(t)} + \Delta t\ \dot{\mathbf{q}}^{(t+\Delta t)}. \end{aligned} \tag{7.1}$$

For the simplified equations of motion (4.51), the Euler procedure becomes

$$\mathbf{q}^{(t+\Delta t)} = \mathbf{q}^{(t)} + \Delta t\ \mathbf{D}^{-1}\left(\mathbf{f}_q^{(t)} - \mathbf{K}\mathbf{q}^{(t)}\right). \tag{7.2}$$

The following details about our numerical solution are noteworthy. First, we represent the rotation component of the models using quaternions (see Chapter 4). This simplifies the updating of $\mathbf{q}_\theta$ 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 $\mathbf{q}_d$ [67]. Third, for added efficiency without significant loss of accuracy, we may lump masses[2] to obtain a diagonal $\mathbf{M}$, and we may assume mass-proportional damping, i.e. $\mathbf{D} = \alpha\mathbf{M}$ where α is the damping coefficient [31]. We can therefore parallelize on multiprocessor computers the updating of the model degrees of freedom and the computation of $\mathbf{K}\mathbf{q}$.

7.2 Model Initialization

We will now discuss various ways of initializing our models in computer vision applications. Our model fitting algorithm based on (4.13) and (4.51) does not require user intervention beyond the initialization phase. The initialization step

[2]In 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.

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 ealry-vision system. We therefore expect that the initialization can be automated using available image segmentation techniques. The reaction-diffusion segmentation process of Kimia, Tannenbaum, and Zucker [34], the dynamic segmentation technique of Leonardis, Gupta and Bajcsy [35] and the qualitative shape recovery and segmentation technique based on primitive aspects of Dickinson, Pentland and Rosenfeld [17, 18] appear promising for this purpose[3].

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 [61]. In case of 3D data we first compute the matrix of central moments

$$\mathcal{M} = \frac{1}{m}\sum_{i=1}^{N}\begin{bmatrix} (r_{i2}-\bar{r}_2)^2+(r_{i3}-\bar{r}_3)^2 & -(r_{i2}-\bar{r}_2)(r_{i1}-\bar{r}_1) & -(r_{i3}-\bar{r}_3)(r_{i1}-\bar{r}_1) \\ & (r_{i1}-\bar{r}_1)^2+(r_{i3}-\bar{r}_3)^2 & -(r_{i3}-\bar{r}_3)(r_{i2}-\bar{r}_2) \\ \text{symmetric} & & (r_{i1}-\bar{r}_1)^2+(r_{i2}-\bar{r}_2)^2 \end{bmatrix}, \quad (7.3)$$

where $\mathbf{r}_i = (r_{i1}, r_{i2}, r_{i3})^\top$ and $\bar{\mathbf{r}} = (\bar{r}_1, \bar{r}_2, \bar{r}_3)^\top$ are the 3D coordinates and 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 [61], 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.

[3]The development of segmentation algorithms is beyond the scope of this thesis.

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 since there is no automatic formal way so far of setting them.[4] Furthermore, the Euler step we used in the shape and nonrigid motion experiments amounted to $20 - 100$ algorithm steps required for a model to fit each data frame.

7.3 Computer Vision Experiments

7.3.1 Static Shape Recovery

We will present examples of fitting deformable superquadric models to 2D real image data and 3D real range data from the NRCC 3D image database [56]. The Euler method time step was 4.0×10^{-5} s and we used a unit damping matrix $\mathbf{D}$.

2D Image Data

Fig. 7.1 shows the various steps of fitting a deformable superquadric to a 120×128, 256-intensity monocular image Fig. 7.1(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. 7.1(b) shows the potential function $P(x, y)$ generated from the image by computing the magnitude of the intensity gradient. Fig. 7.1(c) shows the initial state of the deformable superquadric 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 $\mathbf{q}_\theta$ 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. 7.1(d) shows an intermediate step in the fitting process which simulates the

[4]The development of formal techniques to automatically set them is a topic of future research.

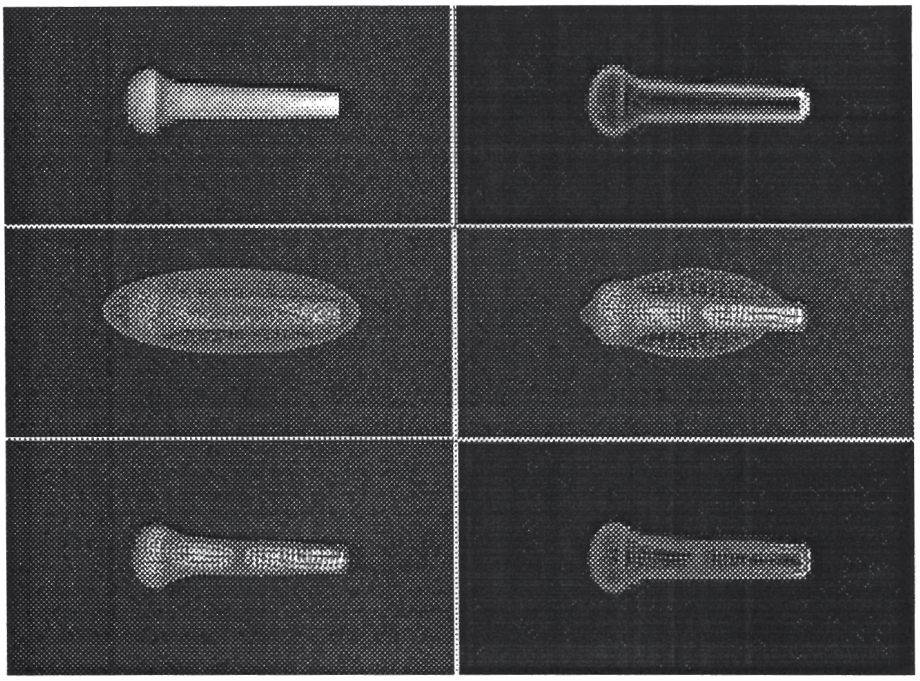
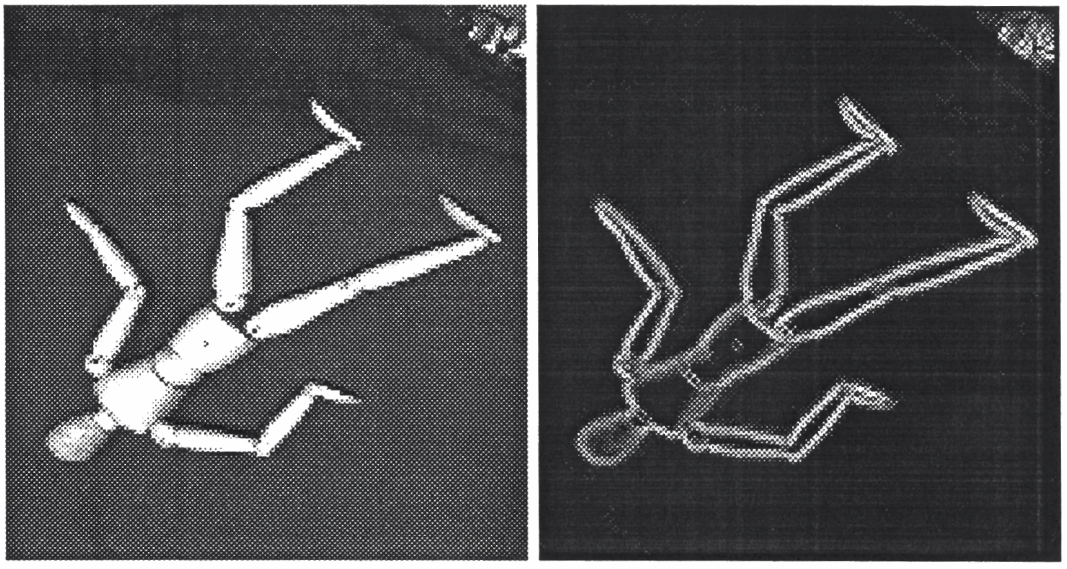
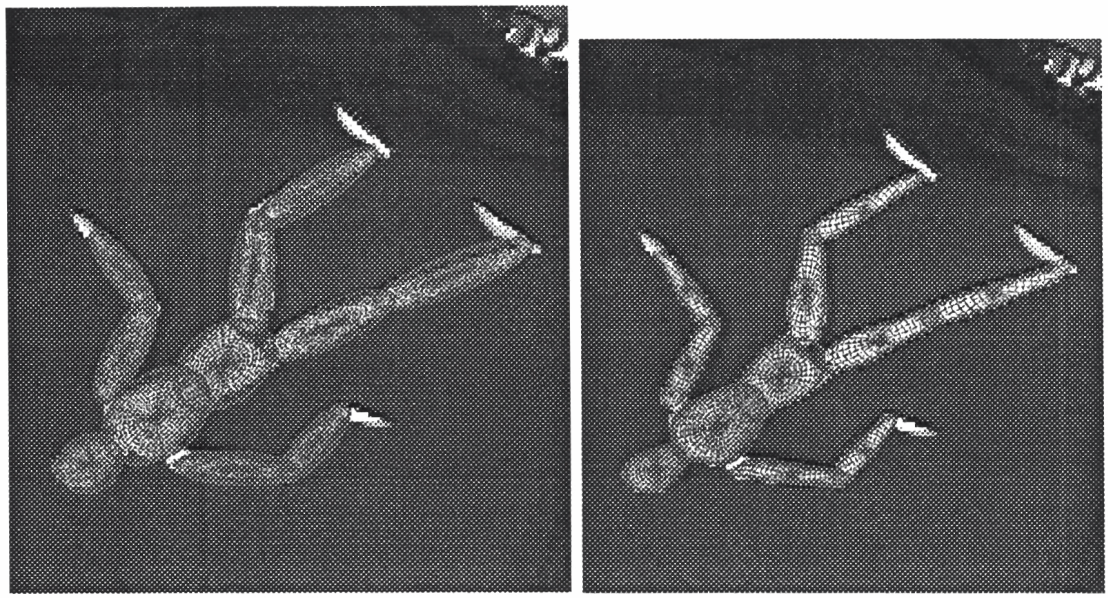


Figure 7.1: Fitting a deformable superquadric to pestle image.



(a) (b)

Figure 7.2: (a) Doll image. (b) Doll potential.



(a) (b)

Figure 7.3: (a) Initialization of deformable superquadrics to doll image. (b) Fitting deformable superquadrics to doll image.

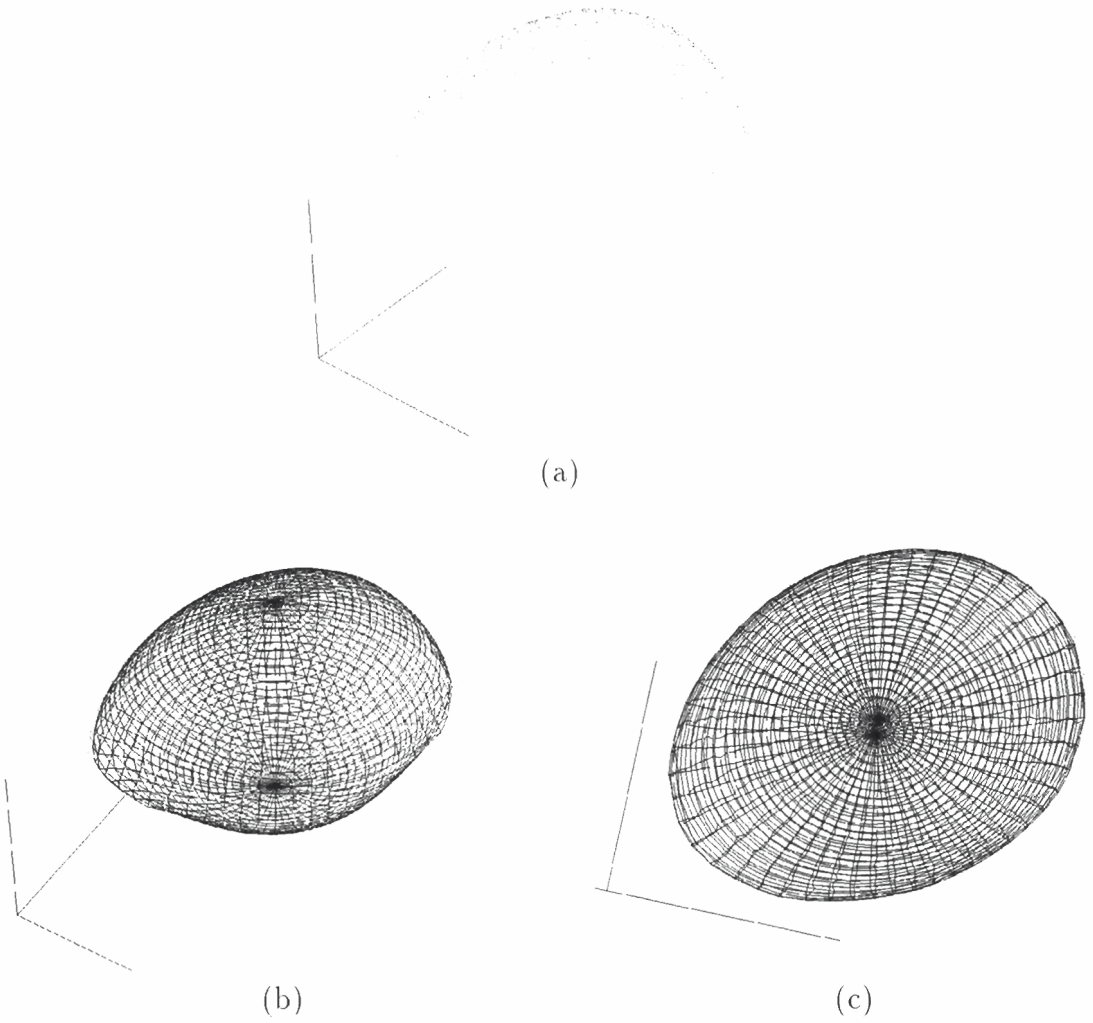
equations of motion using stiffness parameters $w_0 = 1.0 \times 10^{-6}$ and $w_1 = 4.0 \times 10^{-2}$. Using an orthogonal projection Π, nodes of the model whose positions $\mathbf{x}$ in space lie near the image plane ($|x_3| < 0.2$) are subject to a force directed parallel to the image plane:

$$\mathbf{f} = \beta \left(\frac{\partial P}{\partial x}, \frac{\partial P}{\partial y}, 0 \right)^{\mathsf{T}}, \tag{7.4}$$

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

The forces deform the model and Figs. 7.1(e) and (f) show the final state of the model at equilibrium, superimposed on the image and the potential, respectively.

In the second experiment we use the image of a doll shown in Fig. 7.2(a) whose potential is shown in Fig. 7.2(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. 7.3(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. 7.3(b).



(a)

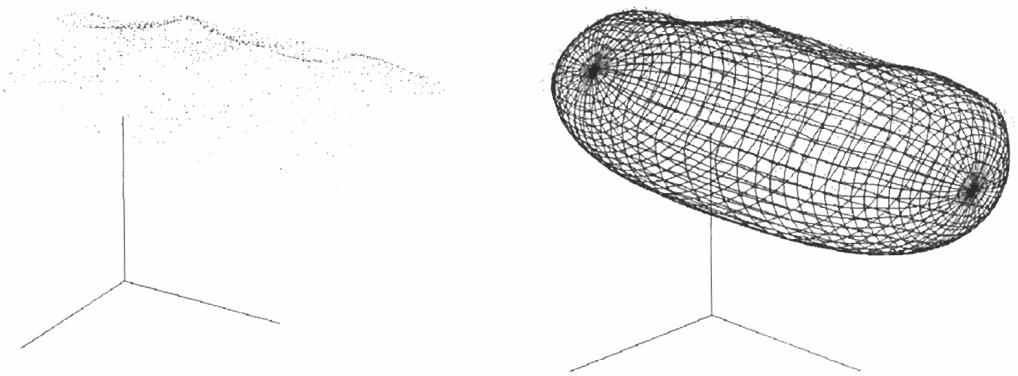
(b) (c)

Figure 7.4: Fitting deformable superquadric (b), (c) to egg range data (a).

3D Range Data

3D data generally provide greater constraint in the model fitting process than do 2D image projections. The following experiments utilize range data from the NRCC 3D image database [56]. 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.

In the first experiment we fit a deformable superquadric model with 2,603 nodes to 3D data sparsely sampled from the upper "hemisphere" of an egg (from



(a) (b)

Figure 7.5: Fitting deformable superquadric (b) to mug range data (a).

range map EGG 1 CAT # 233). Fig. 7.4(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. 7.4(b) shows the fitted deformable superquadric at equilibrium and Fig. 7.4(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. 7.5(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. 7.5(b) shows the fitted deformable superquadric 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 superquadric.

7.3.2 Shape and Nonrigid Motion Estimation

We have carried out various shape and nonrigid motion estimation experiments utilizing synthetic data. The synthetic data sets consist of time-varying 3D positions of points sampled from the surfaces of synthesized deformable superquadrics undergoing nonrigid motions in response to externally applied forces.

In the experiments, we couple the models to the data points, indicated by dark dots in the forthcoming figures, by searching for the nearest node of the model to each datapoint. This brute-force method for assigning data points to model points, is simple and robust. Despite the method's inefficiency compared to the other methods proposed in [67], our algorithms execute at rates of 2–3 seconds per frame of data on a single processor of a Silicon Graphics 4D-340VGX workstation, including the 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 $\mathbf{D}$. We present results which utilize our force-based estimation technique for shape and nonrigid motion estimation.

The following model fitting experiments utilize data that were synthesized by deformable superquadrics producing nonrigid motions in response to externally applied forces. The synthetic data consists of time-varying 3D positions of points sampled from the surfaces of models. We indicate datapoints by dark dots in the forthcoming figures. We tested the performance of our technique in various examples including sparse and noisy data.

In the first experiment we fit a deformable superquadric model with 27 nodes to 27 3D datapoints sampled over time (20 frames) from a squash-like deformable superquadric object undergoing global deformations. Applied forces make impart nonrigid motions on the squash so that an initially positive b_1 bend along the x axis becomes negative over time. Figs. 7.6(a) and (b) show two views of the datapoints and the initial model which is an ellipsoid ($\mathbf{q}_s = (2.7, 0.1, 0.2, 0.7, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.2)^\mathsf{T}$). We did not position the initial model at the center of gravity of the data in order to demonstrate the performance of the model fitting algorithm

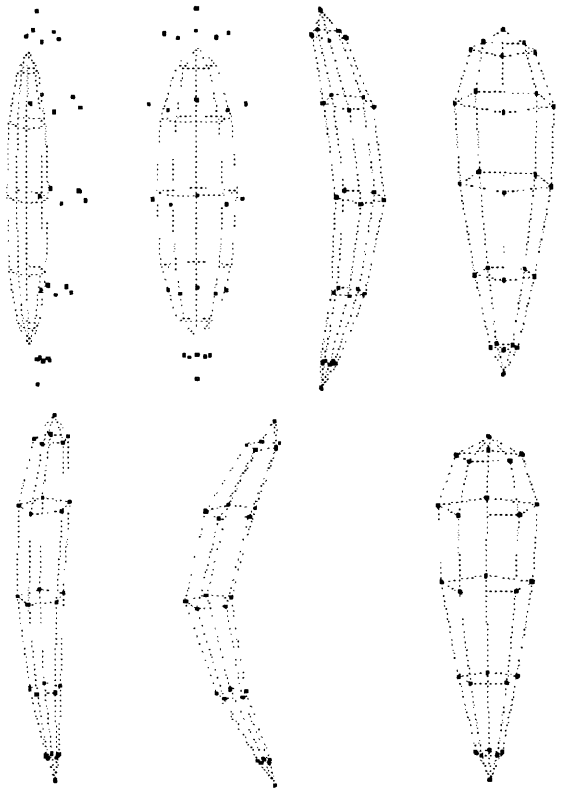


Figure 7.6: Tracking of globally deformable squash shaped object.

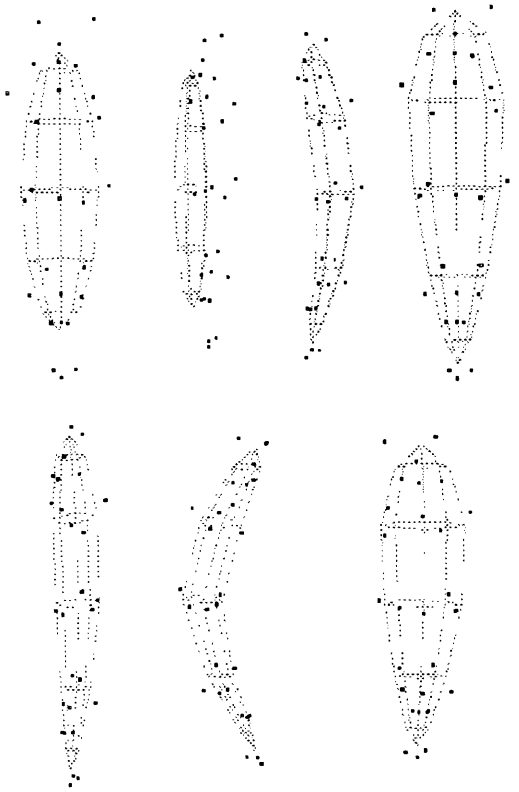


Figure 7.7: Tracking of globally deformable squash shaped object using noisy data.

when the initial estimate is poor. Figs. 7.6(c) and (d) show two views of the model fitted to the initial data. Fig. 7.6(e) is an intermediate time frame of the model tracking the motion of the squash, while Figs. 7.6(f) and (g) show two views of the final position of the model.

In the second experiment we add $\pm 15\%$ noise, with the sign chosen randomly, to the motion range data of the first experiment and fit a deformable superquadric whose initial parameters and position are the same as in the first experiment. Fig. 7.7 is analogous to Fig. 7.6. This experiment demonstrates that the global deformations successfully captured the gross shape indicated by data corrupted by significant noise.

In the third experiment we fit a deformable superquadric model with 27 nodes to 6 3D datapoints sampled from the surface of a squash-like model undergoing only global deformations exactly as in the first experiment. The initial position and other parameters are identical. Fig. 7.8 is analogous to Fig. 7.6. The additional

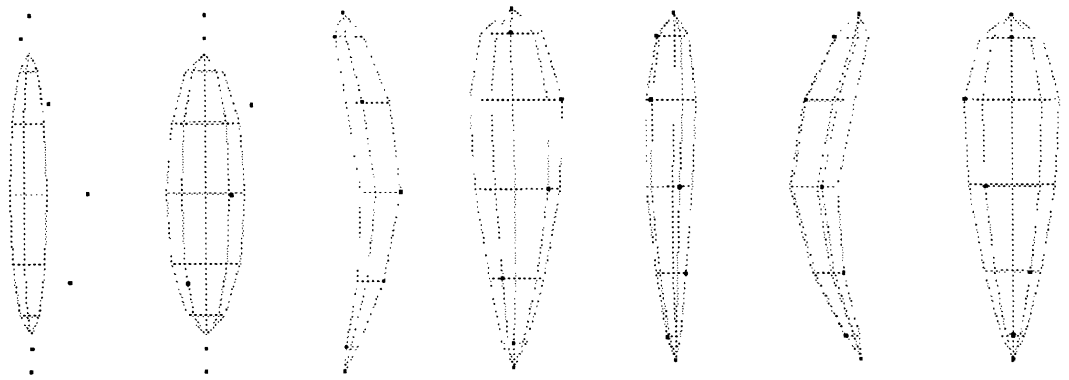


Figure 7.8: Tracking of globally deformable squash model using sparse data.

Fig. 7.8(c′) shows an intermediate step in fitting the model to the initial datapoints. The fit is nearly identical even though there are far fewer datapoints. This is to be expected, since only a few datapoints are required to constrain sufficiently the global deformational degrees of freedom of the model for it to capture the overall shape of the object.

The above experiments indicate that global deformations require relatively few datapoints to abstract the shapes of objects.

7.4 Computer Graphics Experiments

We have created several real time physics-based animation examples involving models made from deformable superquadric 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. 7.9(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 superquadric 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 superquadric which causes it to rotate, translate and deform. The elastic parameters of the deformable superquadric

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 $\mathbf{q}_s = (7.0, 0.4, 0.4, 1.0, 2.5, 1.0)^\mathsf{T}$.

Fig. 7.10 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. 7.11 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. 7.12 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. 7.13(a-i) show an elastic "banana" dropping on a box and subsequently colliding with two planes. The elastic parameters of the "banana" were $w_0 = 65.0$ and $w_1 = 85.0$, the Euler step 0.0025, the nodal mass 7.4, the damping

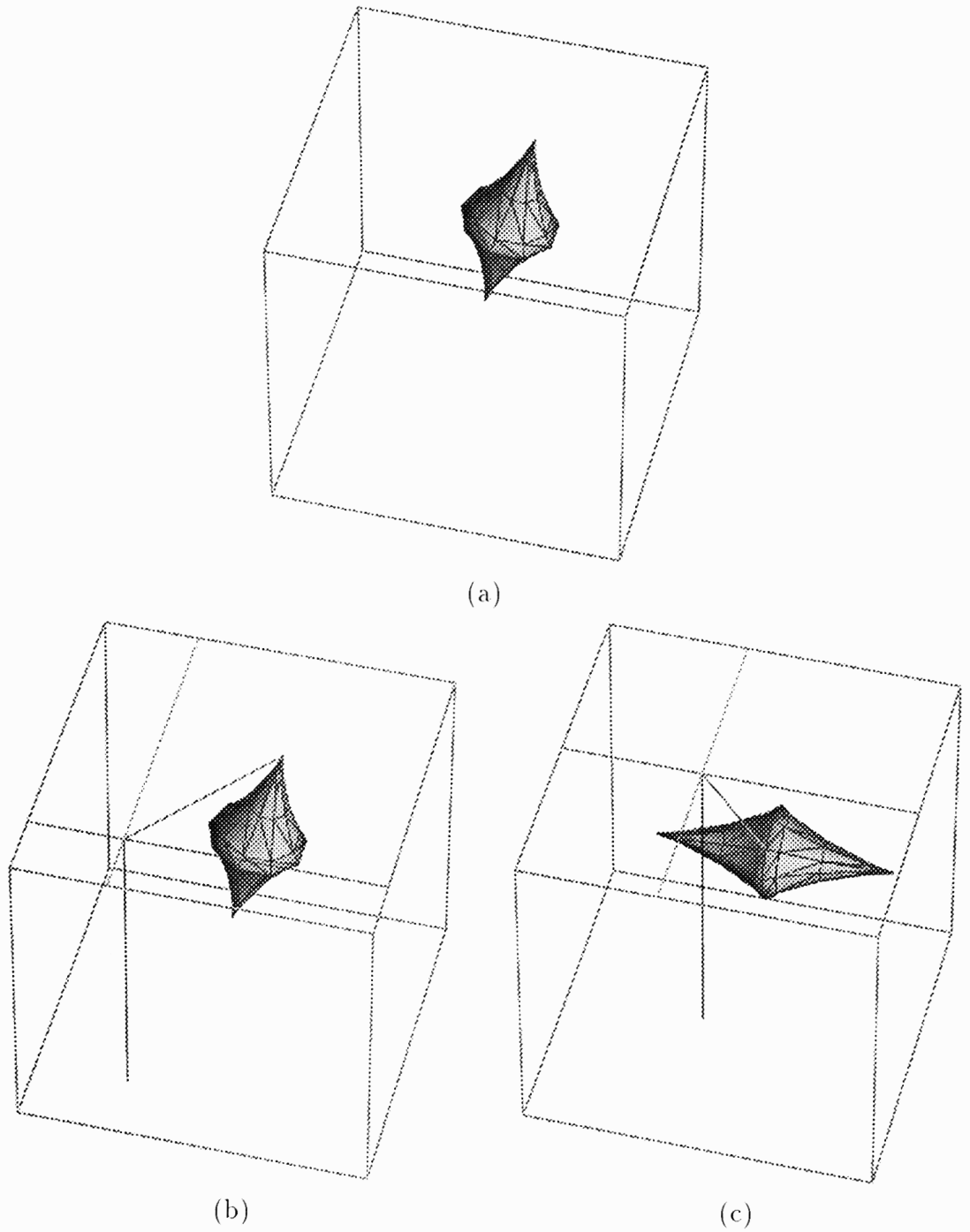


Figure 7.9: Interactive Deformable Superquadric.

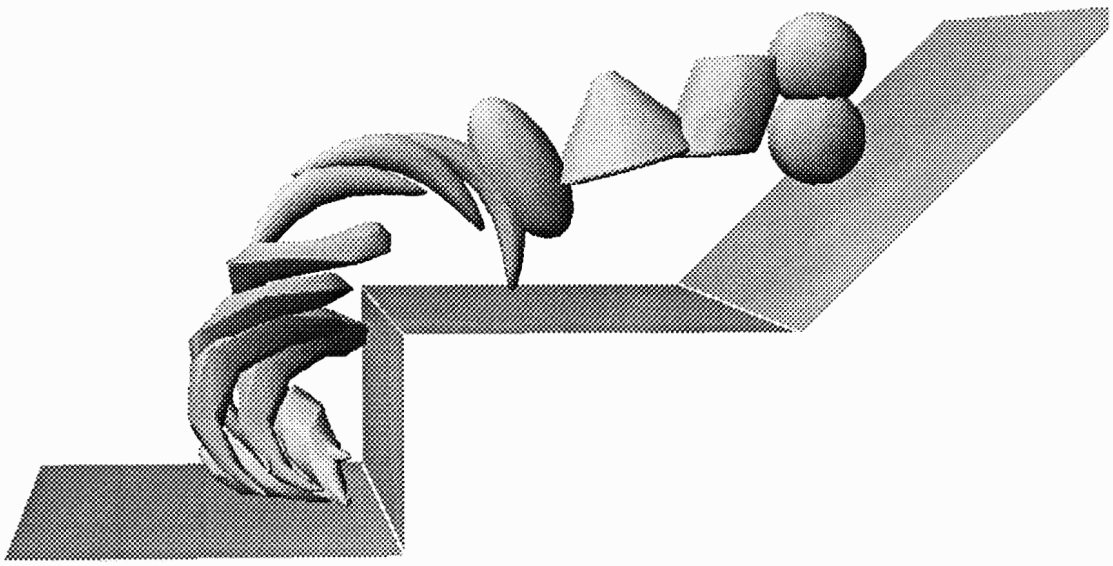


Figure 7.10: Morphing Shell.

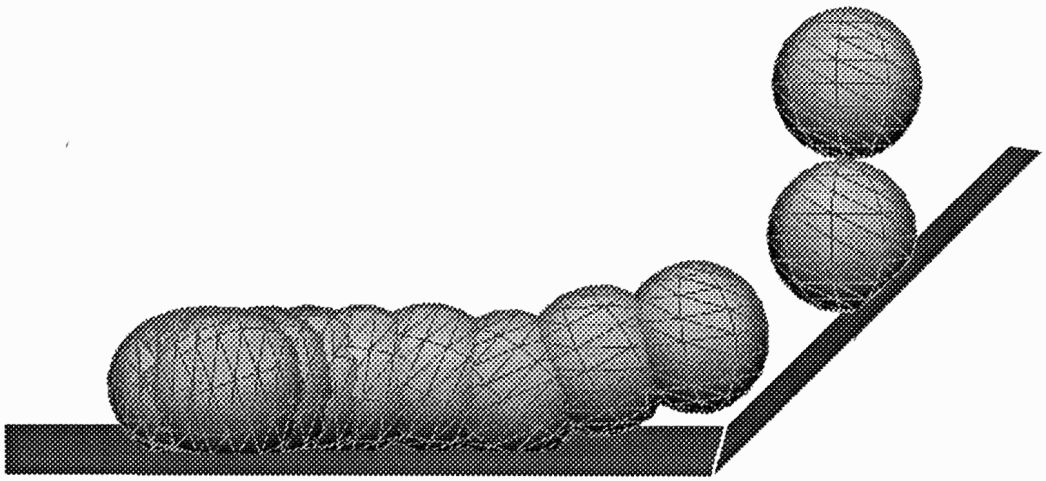


Figure 7.11: "Jello-ball" dropped on a plane.

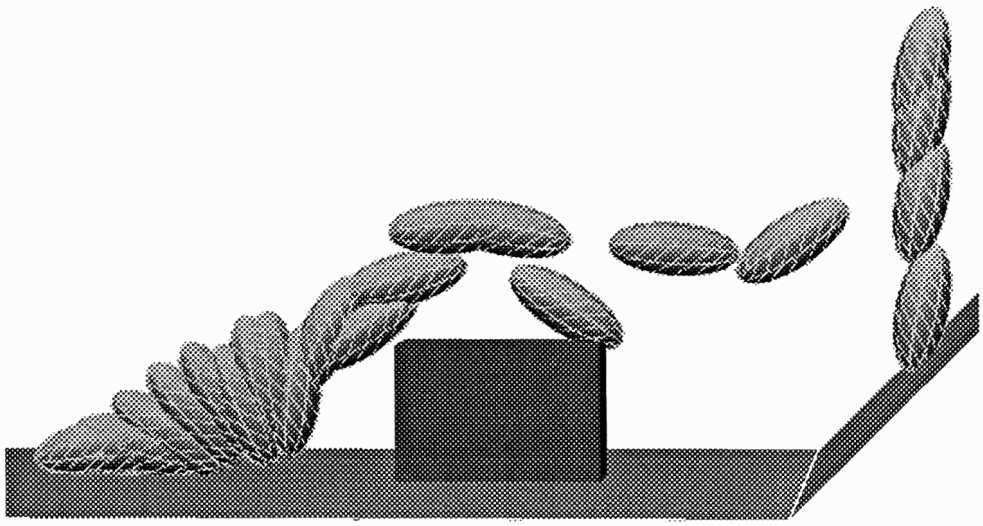


Figure 7.12: Elastic "rugby-ball" dropped on a plane.

coefficient $\nu = 1.47$, the plane elasticity 90.0 and the friction coefficient 0.15. Most of the "banana" mass is concentrated at its ends. The highly nonuniform mass distribution causes exaggerated swings as the "banana" collides with the planes.

Figs. 7.14(a-i) illustrate collisions of two locally deformable balls with planes and a spring loaded see-saw, while Fig. 7.15 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 energy" associated with the rotation of the see saw. Its formulation has the same form as the stiffness energy associated with global deformations we presented in chapter 4.

7.5 Summary

In this chapter we first described the integration schemes we use to simulate the differential equations of motion in parallel and the algorithm for initial placement of our models. Secondly, we presented some computer vision and computer graphics

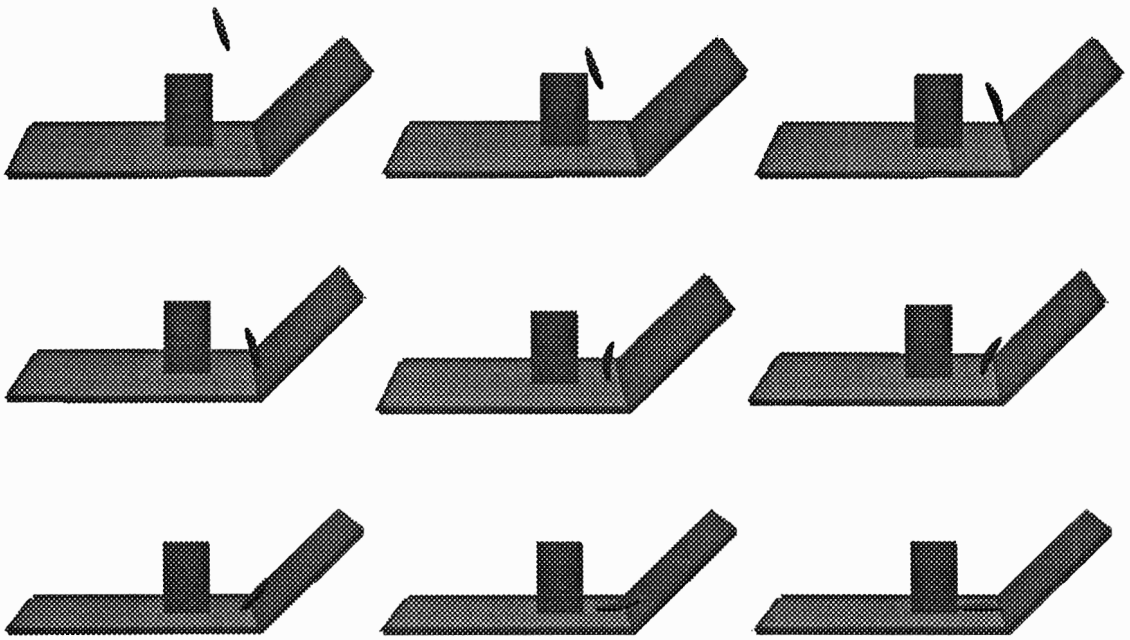


Figure 7.13: Elastic "banana" dropped on a box.

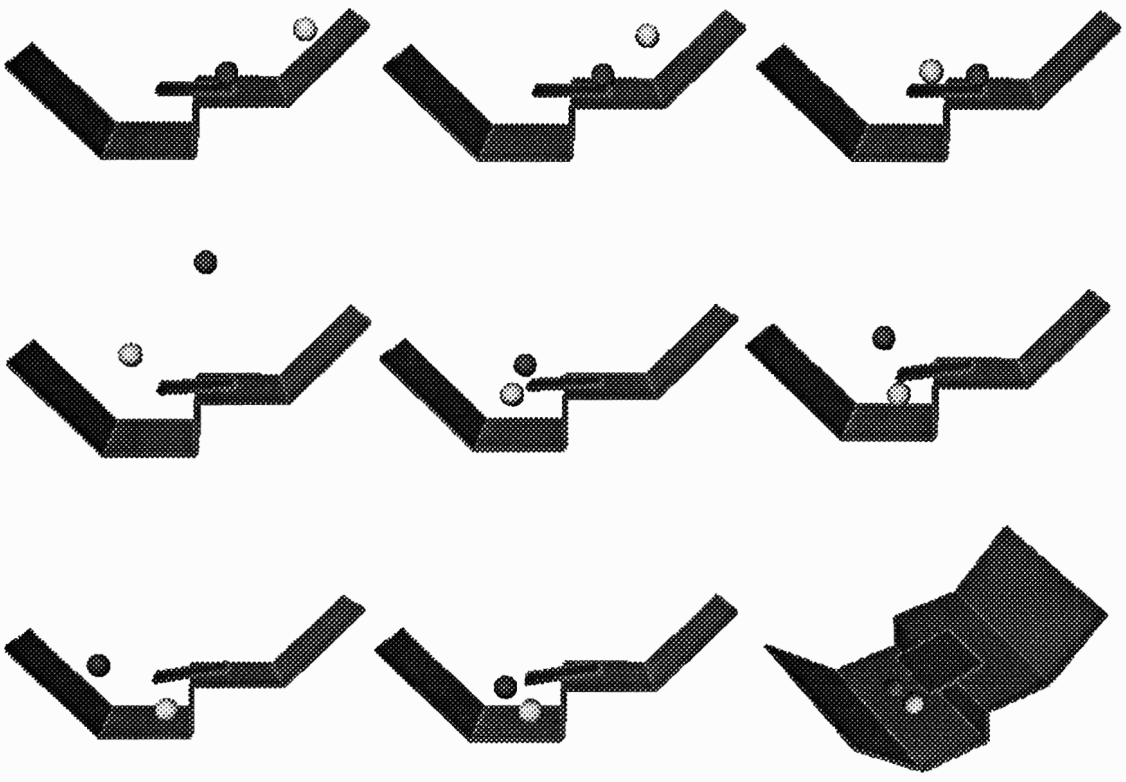


Figure 7.14: Collisions of deformable balls with planes and a spring loaded see-saw.

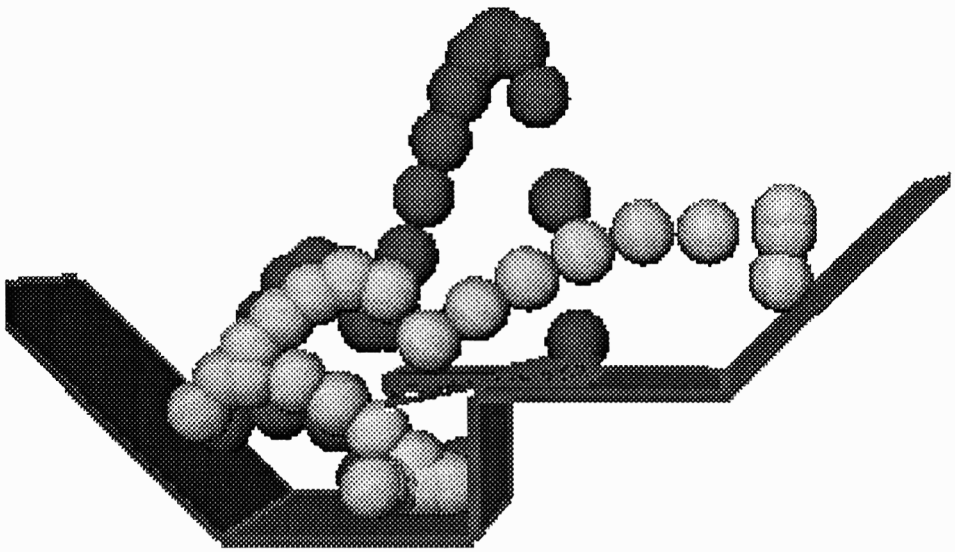


Figure 7.15: Strobed motion sequence of previous example.

experiments that test the modeling methods developed to this point.

Chapter 8

Constrained Nonrigid Motion

This chapter extends the equations of motion (4.13) to account for the motions of composite models with interconnected deformable parts which are constrained not to separate.

8.1 Holonomic Constraints and Lagrange Multipliers

Shabana [59] describes the well-known Lagrange multiplier method for multibody 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 $\mathbf{q}_i$, $\mathbf{g}_{q_i}$, and $\mathbf{f}_{q_i}$ associated with each part $i = 1, \ldots, \bar{n}$. 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

$$\mathbf{C}(\mathbf{q}, t) = \mathbf{0}, \tag{8.1}$$

where

$$\mathbf{C} = [\mathbf{C}_1^\top, \mathbf{C}_2^\top, \ldots, \mathbf{C}_{\bar{k}}^\top]^\top \tag{8.2}$$

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

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{g}_q + \mathbf{f}_q + \mathbf{f}_{g_c}, \tag{8.3}$$

while in case of the simplified model, the equations become

$$\mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{f}_q + \mathbf{f}_{g_c}. \tag{8.4}$$

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

$$\dot{\mathbf{u}} = \mathbf{F}\mathbf{u} + \mathbf{g}, \tag{8.5}$$

where $\mathbf{u}$, $\mathbf{F}$ have exactly the same for as in (4.50) and (4.52) for second and first order systems respectively. The vector $\mathbf{g}$ is now given by

$$\mathbf{g} = \begin{bmatrix} \mathbf{M}^{-1}(\mathbf{g}_q + \mathbf{f}_q + \mathbf{f}_{g_c}) \\ \mathbf{0} \end{bmatrix}, \tag{8.6}$$

for second-order systems and

$$\mathbf{g} = \mathbf{D}^{-1}(\mathbf{f}_q + \mathbf{f}_{g_c}), \tag{8.7}$$

for first-order systems.

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

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{g}_q + \mathbf{f}_q - \mathbf{C}_{\mathbf{q}}^{\mathsf{T}}\boldsymbol{\lambda}, \tag{8.8}$$

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

$$\mathbf{f}_{g_c} = -\mathbf{C}_\mathbf{q}^\top \boldsymbol{\lambda}. \tag{8.9}$$

The term $\mathbf{C}_\mathbf{q}^\top$ is the transpose of the constraint Jacobian matrix and

$$\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1^\top, \ldots, \boldsymbol{\lambda}_{\bar{n}}^\top)^\top \tag{8.10}$$

is the vector of Lagrange multipliers that must be determined.

Equation (8.8) comprises fewer equations than unknowns. To obtain the additional equations, we differentiate (8.1) twice with respect to time

$$\ddot{\mathbf{C}}(\mathbf{q}, t) = \mathbf{0}, \tag{8.11}$$

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

$$\boldsymbol{\gamma} = \mathbf{C}_\mathbf{q}\ddot{\mathbf{q}} = -\mathbf{C}_{tt} - (\mathbf{C}_\mathbf{q}\dot{\mathbf{q}})_\mathbf{q}\dot{\mathbf{q}} - 2\mathbf{C}_{\mathbf{q}_t}\dot{\mathbf{q}}. \tag{8.12}$$

Appending this equation to (8.8) and rearranging terms, we arrive at the augmented equations of motion

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_\mathbf{q}^\top \\ \mathbf{C}_\mathbf{q} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{D}\dot{\mathbf{q}} - \mathbf{K}\mathbf{q} + \mathbf{g}_q + \mathbf{f}_q \\ \boldsymbol{\gamma} \end{bmatrix}. \tag{8.13}$$

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 (8.13) 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.

8.2 Stabilized Constraints

To remedy these two problems, we apply a method proposed by Baumgarte which stabilizes the constrained equations through linear feedback control [7, 73]. The method replaces the differential equation (8.12) 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

$$\ddot{\mathbf{C}} + 2\alpha\dot{\mathbf{C}} + \beta^2\mathbf{C} = \mathbf{0}, \tag{8.14}$$

where α and β are stabilization factors modify (8.13) as follows

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_\mathbf{q}^\top \\ \mathbf{C}_\mathbf{q} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{D}\dot{\mathbf{q}} - \mathbf{K}\mathbf{q} + \mathbf{g}_q + \mathbf{f}_q \\ \boldsymbol{\gamma} - 2\alpha\dot{\mathbf{C}} - \beta^2\mathbf{C} \end{bmatrix}. \tag{8.15}$$

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

$$\mathbf{C}(\mathbf{q},t) = \mathbf{C}(\mathbf{q},0)e^{-\alpha t} \tag{8.16}$$

which, for given α, has the quickest asymptotic decay towards constraint satisfaction $\mathbf{C} = \mathbf{0}$. A caveat in applying the constraint stabilization method is that it introduces 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 also in numerically stiff equations.

8.3 Fast Point-to-Point Constraint Force Computation

The general Lagrange multiplier method described above is potentially expensive for our models, since the matrix in (8.13) can be large, depending on the number of finite elements used in the model discretization. We have devised a specialized solver for the unknown constraint forces $\mathbf{f}_{g_c}$ for point-to-point constraints. The specialized method requires the solution of linear systems of size proportional to the number of constraints, which is usually small. In this sense, it is similar to the dynamic constraint technique of [5]; however, it is suitable for nonrigid parts. We derive the method for second-order dynamic systems (4.13) and for first-order dynamic systems (4.51).

8.3.1 Second Order Dynamic Systems

We will start by giving simple examples of multipart objects with constraints. This will clarify the general algorithm described later.

Single Constraint

Fig. 8.1 illustrates two parts, 1 and 2. We constrain points A and B to be in contact, and must compute the constraint forces $\mathbf{f}_c(t)$ at point A and $-\mathbf{f}_c(t)$ at point B needed to accomplish this. From (4.13), the motion equations of the parts are

$$\ddot{\mathbf{q}}_1 = \mathbf{M}_1^{-1}(\mathbf{g}_{q_1} + \mathbf{f}_{q_1} + \mathbf{f}_{g_{c_A}} - \mathbf{K}_1\mathbf{q}_1 - \mathbf{D}_1\dot{\mathbf{q}}_1), \tag{8.17}$$

$$\ddot{\mathbf{q}}_2 = \mathbf{M}_2^{-1}(\mathbf{g}_{q_2} + \mathbf{f}_{q_2} + \mathbf{f}_{g_{c_B}} - \mathbf{K}_2\mathbf{q}_2 - \mathbf{D}_2\dot{\mathbf{q}}_2), \tag{8.18}$$

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

$$\mathbf{f}_{g_{c_A}} = \mathbf{L}_A^T\mathbf{f}_c, \qquad \mathbf{f}_{g_{c_B}} = -\mathbf{L}_B^T\mathbf{f}_c, \tag{8.19}$$

and $\mathbf{L}_A, \mathbf{L}_B$ are computed using (4.4).

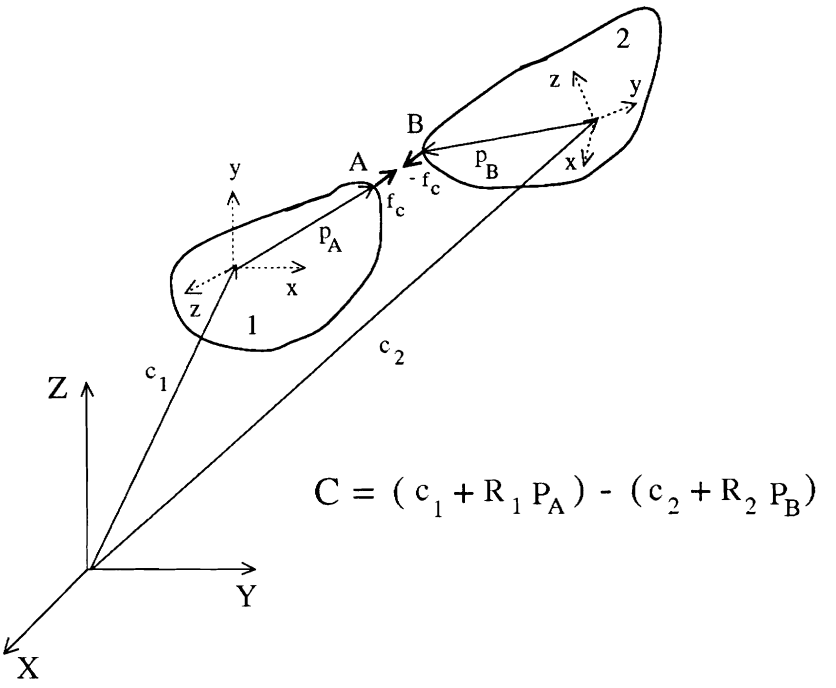


Figure 8.1: Point-to-point constraint.

From (4.4) and (4.3), the constraint equation and its time derivatives are

$$\begin{aligned} \mathbf{C} &= \mathbf{x}_A - \mathbf{x}_B = (\mathbf{c}_1 + \mathbf{R}_1\mathbf{p}_A) - (\mathbf{c}_2 + \mathbf{R}_2\mathbf{p}_B) \\ \dot{\mathbf{C}} &= \mathbf{L}_A\dot{\mathbf{q}}_1 - \mathbf{L}_B\dot{\mathbf{q}}_2 \\ \ddot{\mathbf{C}} &= \mathbf{L}_A\ddot{\mathbf{q}}_1 + \dot{\mathbf{L}}_A\dot{\mathbf{q}}_1 - \mathbf{L}_B\ddot{\mathbf{q}}_2 - \dot{\mathbf{L}}_B\dot{\mathbf{q}}_2. \end{aligned} \tag{8.20}$$

Replacing these expressions into Baumagarte's equation (8.14) (with $\alpha = \beta$), we obtain the linear equations

$$\mathbf{N}\mathbf{f}_c + \mathbf{r} = \mathbf{0}, \tag{8.21}$$

for $\mathbf{f}_c$, where the 3×3 matrix $\mathbf{N}$ is

$$\mathbf{N} = (\mathbf{L}_A\mathbf{M}_1^{-1}\mathbf{L}_A^\top + \mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{L}_B^\top) \tag{8.22}$$

and the vector $\mathbf{r}$ is

$$\mathbf{r} = \dot{\mathbf{L}}_A\dot{\mathbf{q}}_1 - \dot{\mathbf{L}}_B\dot{\mathbf{q}}_2 + 2\alpha\dot{\mathbf{C}} + \alpha^2\mathbf{C} + \mathbf{L}_A\mathbf{M}_1^{-1}(\mathbf{g}_{q_1} + \mathbf{f}_{q_1} - \mathbf{K}_1\mathbf{q}_1 - \mathbf{D}_1\dot{\mathbf{q}}_1) -$$

$$\mathbf{L}_B\mathbf{M}_2^{-1}(\mathbf{g}_{q_2} + \mathbf{f}_{q_2} - \mathbf{K}_2\mathbf{q}_2 - \mathbf{D}_2\dot{\mathbf{q}}_2). \tag{8.23}$$

Two Constraints

Figure 8.2 illustrates three parts, 1, 2 and 3 of an 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 (4.13), the motion equations of the parts are

$$\begin{aligned}\ddot{\mathbf{q}}_1 &= \mathbf{M}_1^{-1}(\mathbf{g}_{q_1} + \mathbf{f}_{q_1} + \mathbf{f}_{g_{c_A}} - \mathbf{K}_1\mathbf{q}_1 - \mathbf{D}_1\dot{\mathbf{q}}_1) \\ &= \mathbf{M}_1^{-1}(\mathbf{f}_{g_{c_A}} + \mathbf{V}_1),\end{aligned} \tag{8.24}$$

$$\begin{aligned}\ddot{\mathbf{q}}_2 &= \mathbf{M}_2^{-1}(\mathbf{g}_{q_2} + \mathbf{f}_{q_2} + \mathbf{f}_{g_{c_B}} + \mathbf{f}_{g_{c_C}} - \mathbf{K}_2\mathbf{q}_2 - \mathbf{D}_2\dot{\mathbf{q}}_2) \\ &= \mathbf{M}_2^{-1}(\mathbf{f}_{g_{c_B}} + \mathbf{f}_{g_{c_C}} + \mathbf{V}_2),\end{aligned} \tag{8.25}$$

$$\begin{aligned}\ddot{\mathbf{q}}_3 &= \mathbf{M}_3^{-1}(\mathbf{g}_{q_3} + \mathbf{f}_{q_3} + \mathbf{f}_{g_{c_D}} - \mathbf{K}_3\mathbf{q}_3 - \mathbf{D}_3\dot{\mathbf{q}}_3) \\ &= \mathbf{M}_3^{-1}(\mathbf{f}_{g_{c_D}} + \mathbf{V}_3),\end{aligned} \tag{8.26}$$

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

$$\begin{aligned}\mathbf{f}_{g_{c_A}} &= \mathbf{L}_A^\top\mathbf{f}_{c_1}, \quad & \mathbf{f}_{g_{c_B}} &= -\mathbf{L}_B^\top\mathbf{f}_{c_1}, \\ \mathbf{f}_{g_{c_C}} &= \mathbf{L}_C^\top\mathbf{f}_{c_2}, \quad & \mathbf{f}_{g_{c_D}} &= -\mathbf{L}_D^\top\mathbf{f}_{c_2},\end{aligned} \tag{8.27}$$

and $\mathbf{L}_A, \mathbf{L}_B, \mathbf{L}_C, \mathbf{L}_D$, are computed using (4.4).

From (4.4) and (4.3), the two constraint equations and their time derivatives are

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

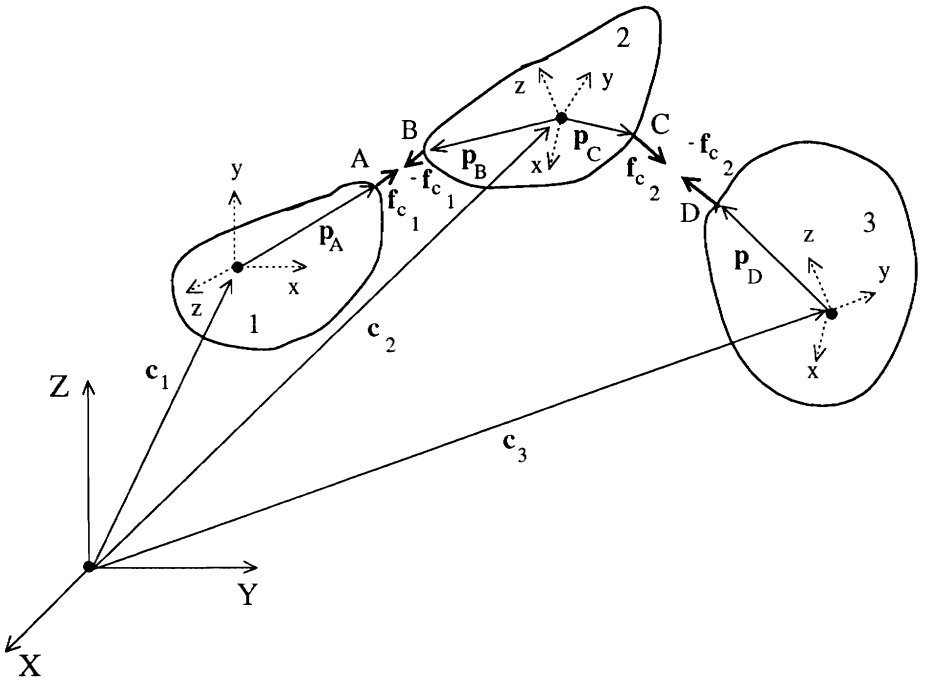


Figure 8.2: Two Point-to-point constraints.

$$\ddot{\mathbf{C}}_1 = \mathbf{L}_A\ddot{\mathbf{q}}_1 + \dot{\mathbf{L}}_A\dot{\mathbf{q}}_1 - \mathbf{L}_B\ddot{\mathbf{q}}_2 - \dot{\mathbf{L}}_B\dot{\mathbf{q}}_2, \tag{8.28}$$

and

$$\begin{aligned}
\mathbf{C}_2 &= \mathbf{x}_C - \mathbf{x}_D = (\mathbf{c}_2 + \mathbf{R}_2\mathbf{p}_C) - (\mathbf{c}_3 + \mathbf{R}_3\mathbf{p}_D) \\
\dot{\mathbf{C}}_2 &= \mathbf{L}_C\dot{\mathbf{q}}_2 - \mathbf{L}_D\dot{\mathbf{q}}_3 \\
\ddot{\mathbf{C}}_2 &= \mathbf{L}_C\ddot{\mathbf{q}}_2 + \dot{\mathbf{L}}_C\dot{\mathbf{q}}_2 - \mathbf{L}_D\ddot{\mathbf{q}}_3 - \dot{\mathbf{L}}_D\dot{\mathbf{q}}_3.
\end{aligned} \tag{8.29}$$

Replacing these expressions into Baumagarte's equation (8.14) (with $\alpha = \beta$), we obtain the linear system of equations

$$\begin{aligned}
(\mathbf{L}_A\mathbf{M}_1^{-1}\mathbf{L}_A^\top + \mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{L}_B^\top)\mathbf{f}_{c_1} - (\mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{L}_C^\top)\mathbf{f}_{c_2} + \mathbf{r}_{11} &= \mathbf{N}_1\mathbf{f}_c + \mathbf{r}_1 = \mathbf{0} \\
-(\mathbf{L}_C\mathbf{M}_2^{-1}\mathbf{L}_B^\top)\mathbf{f}_{c_1} + (\mathbf{L}_C\mathbf{M}_2^{-1}\mathbf{L}_C^\top + \mathbf{L}_D\mathbf{M}_3^{-1}\mathbf{L}_D^\top)\mathbf{f}_{c_2} + \mathbf{r}_{22} &= \mathbf{N}_2\mathbf{f}_c + \mathbf{r}_2 = \mathbf{0},
\end{aligned} \tag{8.30}$$

with unknowns the constraint forces $\mathbf{f}_c = (\mathbf{f}_{c_1}^\top, \mathbf{f}_{c_2}^\top)^\top$, where the 3×1 vectors $\mathbf{r}_{11}$

and $\mathbf{r}_{22}$ are

$$\mathbf{r}_{11} = \dot{\mathbf{L}}_A\dot{\mathbf{q}}_1 - \dot{\mathbf{L}}_B\dot{\mathbf{q}}_2 + 2a\dot{\mathbf{C}}_1 + b^2\mathbf{C}_1 + \mathbf{L}_A\mathbf{M}_1^{-1}\mathbf{V}_1 - \mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{V}_2, \tag{8.31}$$

$$\mathbf{r}_{22} = \dot{\mathbf{L}}_C\dot{\mathbf{q}}_2 - \dot{\mathbf{L}}_D\dot{\mathbf{q}}_3 + 2a\dot{\mathbf{C}}_2 + b^2\mathbf{C}_2 + \mathbf{L}_C\mathbf{M}_2^{-1}\mathbf{V}_2 - \mathbf{L}_D\mathbf{M}_3^{-1}\mathbf{V}_3, \tag{8.32}$$

and

$$N_1 = \begin{pmatrix} \mathbf{L}_A\mathbf{M}_1^{-1}\mathbf{L}_A^\top + \mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{L}_B^\top & -(\mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{L}_C^\top) \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{8.33}$$

$$N_2 = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ -(\mathbf{L}_C\mathbf{M}_2^{-1}\mathbf{L}_B^\top) & \mathbf{L}_C\mathbf{M}_2^{-1}\mathbf{L}_C^\top + \mathbf{L}_D\mathbf{M}_3^{-1}\mathbf{L}_D^\top \end{pmatrix}, \tag{8.34}$$

$$\mathbf{r}_1 = \begin{pmatrix} \mathbf{r}_{11} \\ \mathbf{0} \end{pmatrix}, \tag{8.35}$$

$$\mathbf{r}_2 = \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_{22} \end{pmatrix}. \tag{8.36}$$

Combining (8.30) we arrive at the linear system

$$\mathbf{N}\mathbf{f}_c + \mathbf{r} = (\mathbf{N}_1 + \mathbf{N}_2)\mathbf{f}_c + (\mathbf{r}_1 + \mathbf{r}_2) = \mathbf{0}, \tag{8.37}$$

which we solve using an LU decomposition algorithm.

Multiple Constraints

For multiple point-to-point constraints the constraint force 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 multipart model's constraint force vector

$\mathbf{f}_c = (\mathbf{f}_{c_1}^\top, \mathbf{f}_{c_2}^\top, \ldots, \mathbf{f}_{c_{\bar{k}}}^\top)^\top$ and express the equation for constraint forces $\mathbf{f}_{c_i}$ as in (8.30):

$$\mathbf{N}_i \mathbf{f}_c + \mathbf{r}_i = \mathbf{0}, \tag{8.38}$$

where

$$\mathbf{r}_i = (\mathbf{0}^\top, \ldots, \mathbf{r}_{ii}^\top, \ldots, \mathbf{0}^\top)^\top \tag{8.39}$$

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

$$\mathbf{N}\mathbf{f}_c + \mathbf{r} = \mathbf{0}, \tag{8.40}$$

where

$$\mathbf{N} = \sum_{i=1}^{\bar{k}} \mathbf{N}_i \tag{8.41}$$

and

$$\mathbf{r} = \sum_{i=1}^{\bar{k}} \mathbf{r}_i. \tag{8.42}$$

Based on the form of equations (8.65) we can devise the following algorithm to automatically compute the entries of the above matrix $\mathbf{N}$.

Algorithm to Compute N

The algorithm is as follows:

- *Step 0*: Divide $\mathbf{N}$ into submatrices as follows

$$\mathbf{N} = \begin{bmatrix} \mathbf{N}_{11} & . & . & . & \mathbf{N}_{1\bar{k}} \\ & & & & \\ . & . & & & . \\ . & & \mathbf{N}_{ij} & & . \\ . & & & . & . \\ & & & & \\ \mathbf{N}_{\bar{k}1} & . & . & . & \mathbf{N}_{\bar{k}\bar{k}} \end{bmatrix}, \tag{8.43}$$

where $\mathbf{N}_{ij}$ are 3×3 submatrices and $\bar{k}$ is the number of constraints.

Set $\mathbf{N} = \mathbf{0}$.

- *Step 1*: For each constraint l, $0 \leq l \leq \bar{k}$ do

  The constraint requires that i_A and j_B of two deformable parts i and j should always be in contact. Let's also assume that the pointer to part i is marked as *one*, while the pointer to part j is marked as *two*.

  - *Step 1-0*: Set

$$\mathbf{N}_{ll} = \mathbf{L}_{i_A}\mathbf{M}_i^{-1}\mathbf{L}_{i_A}^{\top} + \mathbf{L}_{j_B}\mathbf{M}_j^{-1}\mathbf{L}_{j_B}^{\top}. \tag{8.44}$$

  - *Step 1-1*: For each constraint $0 \leq m \leq \bar{k}$ do

    If constraint m involves point i_J which belongs to part i, and the pointer to part i is marked as *one*:

$$\mathbf{N}_{lm} = \mathbf{L}_{i_A}\mathbf{M}_i^{-1}\mathbf{L}_{i_J}^{\top}. \tag{8.45}$$

    else if the pointer to part i is marked as *two*:

$$\mathbf{N}_{lm} = -\mathbf{L}_{i_A}\mathbf{M}_i^{-1}\mathbf{L}_{i_J}^{\top}. \tag{8.46}$$

  - *Step 1-2*: For each constraint $0 \leq r \leq \bar{k}$ do

    If constraint r which involves point j_J which belongs to part j, and the pointer to part j is marked as *one*:

$$\mathbf{N}_{lr} = -\mathbf{L}_{j_B}\mathbf{M}_j^{-1}\mathbf{L}_{j_J}^{\top}. \tag{8.47}$$

    else if the pointer to part j is marked as *two*:

$$\mathbf{N}_{lr} = \mathbf{L}_{j_B}\mathbf{M}_j^{-1}\mathbf{L}_{j_J}^{\top}. \tag{8.48}$$

The pattern of nonzero entries in $\mathbf{N}$ depends on the connectivity of the parts.

8.3.2 First Order Dynamic Systems

In first order systems the simplified Lagrange equations of motion take the form

$$\mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{f}_q. \tag{8.49}$$

Since the simplified Lagrange equations of motion do not involve acceleration, we approximate the second derivative of the constraint $\ddot{\mathbf{C}}$ in Baumgarte's differential equation of the constraint as follows

$$\ddot{\mathbf{C}}(\mathbf{q}, t, \Delta t) \approx \frac{\dot{\mathbf{C}}(\mathbf{q}, t) - \dot{\mathbf{C}}(\mathbf{q}, t - \Delta t)}{\Delta t}. \tag{8.50}$$

Baumgarte's technique is implemented in the same way as for second order systems with a few minor modifications. We arrive at the general algorithm through an example involving an object with three parts and two constraints.

8.3.3 Two Constraints

Figure 8.2 illustrates three parts, 1, 2 and 3 of an 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 (4.51), the motion equations of the parts are

$$\begin{aligned} \dot{\mathbf{q}}_1 &= \mathbf{D}_1^{-1}(\mathbf{f}_{q_1} + \mathbf{f}_{g_{c_A}} - \mathbf{K}_1\mathbf{q}_1) \\ &= \mathbf{D}_1^{-1}(\mathbf{f}_{g_{c_A}} + \mathbf{V}_1), \end{aligned} \tag{8.51}$$

$$\begin{aligned} \dot{\mathbf{q}}_2 &= \mathbf{D}_2^{-1}(\mathbf{f}_{q_2} + \mathbf{f}_{g_{c_B}} + \mathbf{f}_{g_{c_C}} - \mathbf{K}_2\mathbf{q}_2) \\ &= \mathbf{D}_2^{-1}(\mathbf{f}_{g_{c_B}} + \mathbf{f}_{g_{c_C}} + \mathbf{V}_2), \end{aligned} \tag{8.52}$$

$$\dot{\mathbf{q}}_3 = \mathbf{D}_3^{-1}(\mathbf{f}_{q_3} + \mathbf{f}_{g_{c_D}} - \mathbf{K}_3\mathbf{q}_3)$$

$$= \mathbf{D}_3^{-1}(\mathbf{f}_{g_{c_D}} + \mathbf{V}_3), \tag{8.53}$$

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

$$\begin{aligned} \mathbf{f}_{g_{c_A}} &= \mathbf{L}_A^\top \mathbf{f}_{c_1}, \quad & \mathbf{f}_{g_{c_B}} &= -\mathbf{L}_B^\top \mathbf{f}_{c_1}, \\ \mathbf{f}_{g_{c_C}} &= \mathbf{L}_C^\top \mathbf{f}_{c_2}, \quad & \mathbf{f}_{g_{c_D}} &= -\mathbf{L}_D^\top \mathbf{f}_{c_2}, \end{aligned} \tag{8.54}$$

and $\mathbf{L}_A, \mathbf{L}_B, \mathbf{L}_C, \mathbf{L}_D$, are computed using (4.4).

From (4.4) and (4.3), the two constraint equations and their time derivatives are

$$\begin{aligned} \mathbf{C}_1 &= \mathbf{x}_A - \mathbf{x}_B = (\mathbf{c}_1 + \mathbf{R}_1\mathbf{p}_A) - (\mathbf{c}_2 + \mathbf{R}_2\mathbf{p}_B) \\ \dot{\mathbf{C}}_1 &= \mathbf{L}_A\dot{\mathbf{q}}_1 - \mathbf{L}_B\dot{\mathbf{q}}_2 \\ \ddot{\mathbf{C}}_1 &= \frac{\dot{\mathbf{C}}_1 - \dot{\mathbf{C}}_1(t-\Delta t)}{\Delta t}, \end{aligned} \tag{8.55}$$

and

$$\begin{aligned} \mathbf{C}_2 &= \mathbf{x}_C - \mathbf{x}_D = (\mathbf{c}_2 + \mathbf{R}_2\mathbf{p}_C) - (\mathbf{c}_3 + \mathbf{R}_3\mathbf{p}_D) \\ \dot{\mathbf{C}}_2 &= \mathbf{L}_C\dot{\mathbf{q}}_2 - \mathbf{L}_D\dot{\mathbf{q}}_3 \\ \ddot{\mathbf{C}}_2 &= \frac{\dot{\mathbf{C}}_2 - \dot{\mathbf{C}}_2(t-\Delta t)}{\Delta t}. \end{aligned} \tag{8.56}$$

Replacing these expressions into Baumagarte's equation (8.14) (with $\alpha = \beta$), we obtain the linear system of equations

$$\begin{aligned} (\frac{1}{\Delta t} + 2a)[(\mathbf{L}_A\mathbf{D}_1^{-1}\mathbf{L}_A^\top + \mathbf{L}_B\mathbf{D}_2^{-1}\mathbf{L}_B^\top)\mathbf{f}_{c_1} - (\mathbf{L}_B\mathbf{D}_2^{-1}\mathbf{L}_C^\top)\mathbf{f}_{c_2}] + \mathbf{r}_{11} &= \mathbf{N}_1\mathbf{f}_c + \mathbf{r}_1 = \mathbf{0} \\ (\frac{1}{\Delta t} + 2a)[-(\mathbf{L}_C\mathbf{C}_2^{-1}\mathbf{L}_B^\top)\mathbf{f}_{c_1} + (\mathbf{L}_C\mathbf{C}_2^{-1}\mathbf{L}_C^\top + \mathbf{L}_D\mathbf{D}_3^{-1}\mathbf{L}_D^\top)\mathbf{f}_{c_2}] + \mathbf{r}_{22} &= \mathbf{N}_2\mathbf{f}_c + \mathbf{r}_2 = \mathbf{0}, \end{aligned} \tag{8.57}$$

where

$$\mathbf{r}_{11} = b^2\mathbf{C}_1 - \frac{\dot{\mathbf{C}}_1(t-\Delta t)}{\Delta t} + (\frac{1}{\Delta t} + 2a)(\mathbf{L}_A\mathbf{D}_1^{-1}\mathbf{V}_1 - \mathbf{L}_B\mathbf{D}_2^{-1}\mathbf{V}_2), \tag{8.58}$$

$$\mathbf{r}_{22} = b^2\mathbf{C}_2 - \frac{\dot{\mathbf{C}}_2(t-\Delta t)}{\Delta t} + (\frac{1}{\Delta t} + 2a)(\mathbf{L}_C\mathbf{D}_2^{-1}\mathbf{V}_2 - \mathbf{L}_D\mathbf{D}_3^{-1}\mathbf{V}_3), \tag{8.59}$$

and

$$\mathbf{N}_1 = (\frac{1}{\Delta t} + 2a)\begin{pmatrix} \mathbf{L}_A\mathbf{D}_1^{-1}\mathbf{L}_A^\top + \mathbf{L}_B\mathbf{D}_2^{-1}\mathbf{L}_B^\top & -(\mathbf{L}_B\mathbf{D}_2^{-1}\mathbf{L}_C^\top) \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{8.60}$$

$$\mathbf{N}_2 = (\frac{1}{\Delta t} + 2a)\begin{pmatrix} \mathbf{0} & \mathbf{0} \\ -(\mathbf{L}_C\mathbf{D}_2^{-1}\mathbf{L}_B^\top) & \mathbf{L}_C\mathbf{D}_2^{-1}\mathbf{L}_C^\top + \mathbf{L}_D\mathbf{D}_3^{-1}\mathbf{L}_D^\top \end{pmatrix}, \tag{8.61}$$

$$\mathbf{r}_1 = \begin{pmatrix} \mathbf{r}_{11} \\ \mathbf{0} \end{pmatrix}, \tag{8.62}$$

$$\mathbf{r}_2 = \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_{22} \end{pmatrix}. \tag{8.63}$$

Combining (8.57) we arrive at the linear system

$$\mathbf{N}\mathbf{f}_c + \mathbf{r} = (\mathbf{N}_1 + \mathbf{N}_2)\mathbf{f}_c + (\mathbf{r}_1 + \mathbf{r}_2) = \mathbf{0}, \tag{8.64}$$

which we solve using an LU decomposition algorithm.

The main difference with the previously described second-order systems is the replacement of the mass matrix $\mathbf{M}$ with the damping matrix $\mathbf{D}$ and the multiplicative coefficient $(\frac{1}{\Delta t} + 2a)$ in the various expressions.

Multiple Constraints

For multiple point-to-point constraints the constraint force 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 $3\bar{k} \times 3\bar{k}$,

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 multipart model's constraint force vector $\mathbf{f}_c = (\mathbf{f}_{c_1}^\top, \mathbf{f}_{c_2}^\top, \ldots, \mathbf{f}_{c_{\bar{k}}}^\top)^\top$ and express the equation for constraint forces $\mathbf{f}_{c_i}$ as in (8.30):

$$\mathbf{N}_i \mathbf{f}_c + \mathbf{r}_i = \mathbf{0}, \tag{8.65}$$

where

$$\mathbf{r}_i = (\mathbf{0}^\top, \ldots, \mathbf{r}_{ii}^\top, \ldots, \mathbf{0}^\top)^\top \tag{8.66}$$

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

$$\mathbf{N}\mathbf{f}_c + \mathbf{r} = \mathbf{0}, \tag{8.67}$$

where

$$\mathbf{N} = \sum_{i=1}^{\bar{k}} \mathbf{N}_i \tag{8.68}$$

and

$$\mathbf{r} = \sum_{i=1}^{\bar{k}} \mathbf{r}_i. \tag{8.69}$$

Based on the form of equations (8.65) we can devise the following algorithm to automatically compute the entries of the above matrix $\mathbf{N}$.

Algorithm to Compute N

The algorithm is as follows:

- *Step 0*: Divide $\mathbf{N}$ into submatrices as in (8.43).

  Set $\mathbf{N} = \mathbf{0}$.

- *Step 1*: For each constraint l, $0 \leq l \leq \bar{k}$ do

  The constraint requires that i_A and j_B of two deformable parts i and j should always be in contact. Let's also assume that the pointer to part i is marked as *one*, while the pointer to part j is marked as *two*.

– *Step 1-0*: Set

$$\mathbf{N}_{ll} = (\frac{1}{\Delta t} + 2a)(\mathbf{L}_{i_A}\mathbf{M}_i^{-1}\mathbf{L}_{i_A}^{\top} + \mathbf{L}_{j_B}\mathbf{M}_j^{-1}\mathbf{L}_{j_B}^{\top}). \tag{8.70}$$

– *Step 1-1*: For each constraint $0 \leq m \leq \bar{k}$ do

If constraint m involves point i_J which belongs to part i, and the pointer to part i is marked as *one*:

$$\mathbf{N}_{lm} = (\frac{1}{\Delta t} + 2a)\mathbf{L}_{i_A}\mathbf{M}_i^{-1}\mathbf{L}_{i_J}^{\top}. \tag{8.71}$$

else if the pointer to part i is marked as *two*:

$$\mathbf{N}_{lm} = -(\frac{1}{\Delta t} + 2a)\mathbf{L}_{i_A}\mathbf{M}_i^{-1}\mathbf{L}_{i_J}^{\top}. \tag{8.72}$$

– *Step 1-2*: For each constraint $0 \leq r \leq \bar{k}$ do

If constraint r which involves point j_J which belongs to part j, and the pointer to part j is marked as *one*:

$$\mathbf{N}_{lr} = -(\frac{1}{\Delta t} + 2a)\mathbf{L}_{j_B}\mathbf{M}_j^{-1}\mathbf{L}_{j_J}^{\top}. \tag{8.73}$$

else if the pointer to part j is marked as *two*:

$$\mathbf{N}_{lr} = (\frac{1}{\Delta t} + 2a)\mathbf{L}_{j_B}\mathbf{M}_j^{-1}\mathbf{L}_{j_J}^{\top}. \tag{8.74}$$

The pattern of nonzero entries in $\mathbf{N}$ depends on the connectivity of the parts.

8.4 Integrating the Constrained Motion Equations

In integrating the constrained motion equations, at each time step we may solve (8.13) for $\ddot{\mathbf{q}}^{(t)}$ and $\boldsymbol{\lambda}^{(t)}$ with known $\mathbf{q}^{(t)}$ and $\dot{\mathbf{q}}^{(t)}$, and then we integrate $\ddot{\mathbf{q}}$ and $\dot{\mathbf{q}}$

from t to $t + \Delta t$ to obtain $\dot{\mathbf{q}}^{(t+\Delta t)}$ and $\mathbf{q}^{(t+\Delta t)}$, respectively, using the Euler method ((7.2) or (7.1)) .

In applying the fast point-to-point constraint algorithm of Section 8.3, at each time step we assemble $\mathbf{N}$ and $\mathbf{r}$ and compute the constraint forces $\mathbf{f}_c$ by solving (8.21) 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 as before.

8.5 Summary

In this chapter we presented a technique to implement hard point-to-point constraints between deformable part models which should not be violated, regardless of the magnitude of the forces experienced by the parts. We then developed a stabilized Lagrange multiplier method.

Chapter 9

Experiments with Constraints

In this chapter we present various experiments involving computer vision and computer graphics applications of our models and constraint algorithms. We first present vision experiments that demonstrate the performance of our framework in complex deformable model fitting to 3D motion data, using our force-based shape estimation algorithm. Next we present dynamic graphical simulations of complex models that interact with each other and their simulated physical environments through constraints, collisions, gravity, and friction against impenetrable surfaces.

9.1 Computer Vision Experiments

We have carried out various shape and nonrigid motion estimation experiments utilizing 3D human motion data and synthetic data. The synthetic data sets consist of time-varying 3D positions of points sampled from the surfaces of synthesized deformable superquadrics undergoing nonrigid motions in response to externally applied forces. The human motion data were collected using WATSMART, a commercial non-contact, 3D motion digitizing and analysis system. Using multiple optoelectric measurement cameras, it can track as many as 64 infrared light emitting diode markers attached to various body parts of a moving subject. It 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 calcu-

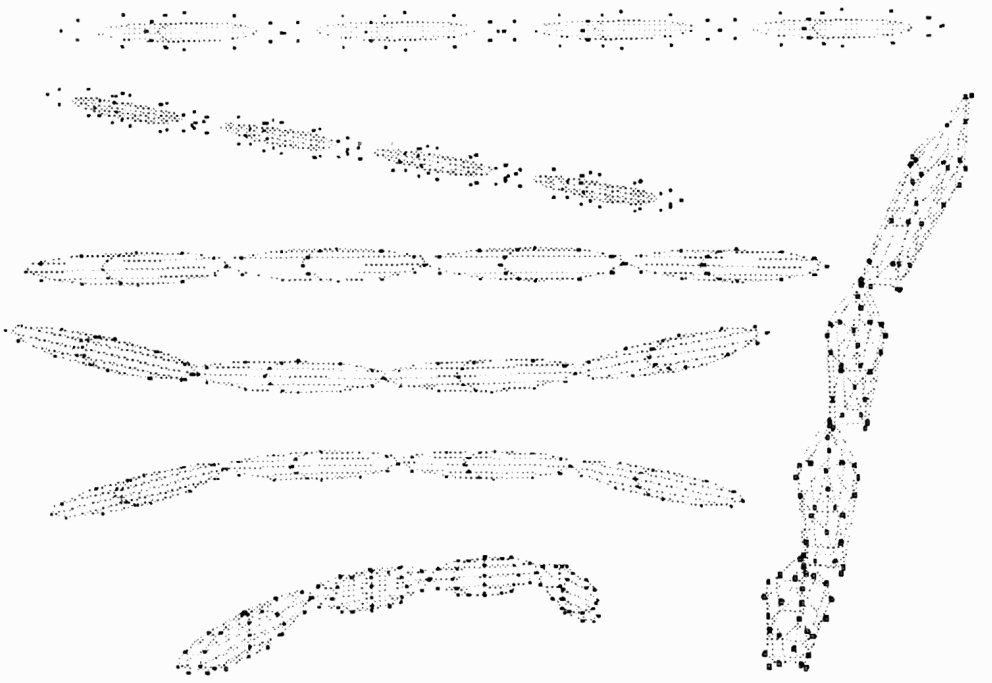


Figure 9.1: Tracking of four globally deformable superquadrics in a row.

lated 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 nearest node of the model to each datapoint. This brute-force method for assigning data points to model points, is simple and robust. Despite the method's inefficiency compared to the other methods proposed in [67], our algorithms execute at rates of 2–3 seconds per frame of data on a single processor of a Silicon Graphics 4D-340VGX workstation, including the 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 $\mathbf{D}$. We present results which utilize either our force-based or our recursive estimation technique for shape and nonrigid motion estimation.

In the first experiment we track the motion of an articulated chain of four deformable superquadrics constrained to be in end-to-end contact. The data-

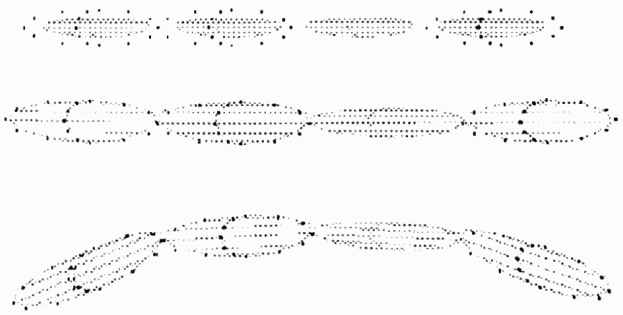


Figure 9.2: Tracking of occluded four globally deformable superquadrics in a row.

points were synthesized by simulating the motion of a connected chain of four deformable superquadrics that may undergo only global deformations, and sampling 27 datapoints from each superquadric "link" through time (47 frames). To demonstrate the performance of our constrained motion tracking algorithm, we initialize the deformable superquadric parts, each with 27 nodes, as shown in Figs. 9.1(a) and (b). The initial part models are ellipsoidal parameters ($\mathbf{q}_s = (1.6, 1.0, 0.2, 0.8, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0)^T$). Fig. 9.1(c) shows a view of the parts fitted to the datapoints in the first time frame. The constraint forces help position the four parts in the correct configuration. Figs. 9.1(d) and (e) show a frame of the 4 constrained deformable superquadrics tracking the upward motion of the chain datapoints, while Figs. 9.1(f) and (g) show them tracking downward motion.

In the second experiment shown in Fig. 9.2 we track the motion of an articulated chain of four deformable superquadrics constrained to be in end-to-end contact. The datapoints were synthesized by simulating the motion of a connected chain of four deformable superquadrics that may undergo only global deformations, and sampling through time (47 frames) 27 datapoints from each superquadric "link," except the third, whose data points are missing due to occlusion. To demonstrate the performance of our constrained motion tracking algorithm, we initialize the deformable superquadric parts, each with 27 nodes, as shown in Figs. 9.2(a). The initial part models are ellipsoidal parameters ($\mathbf{q}_s =$

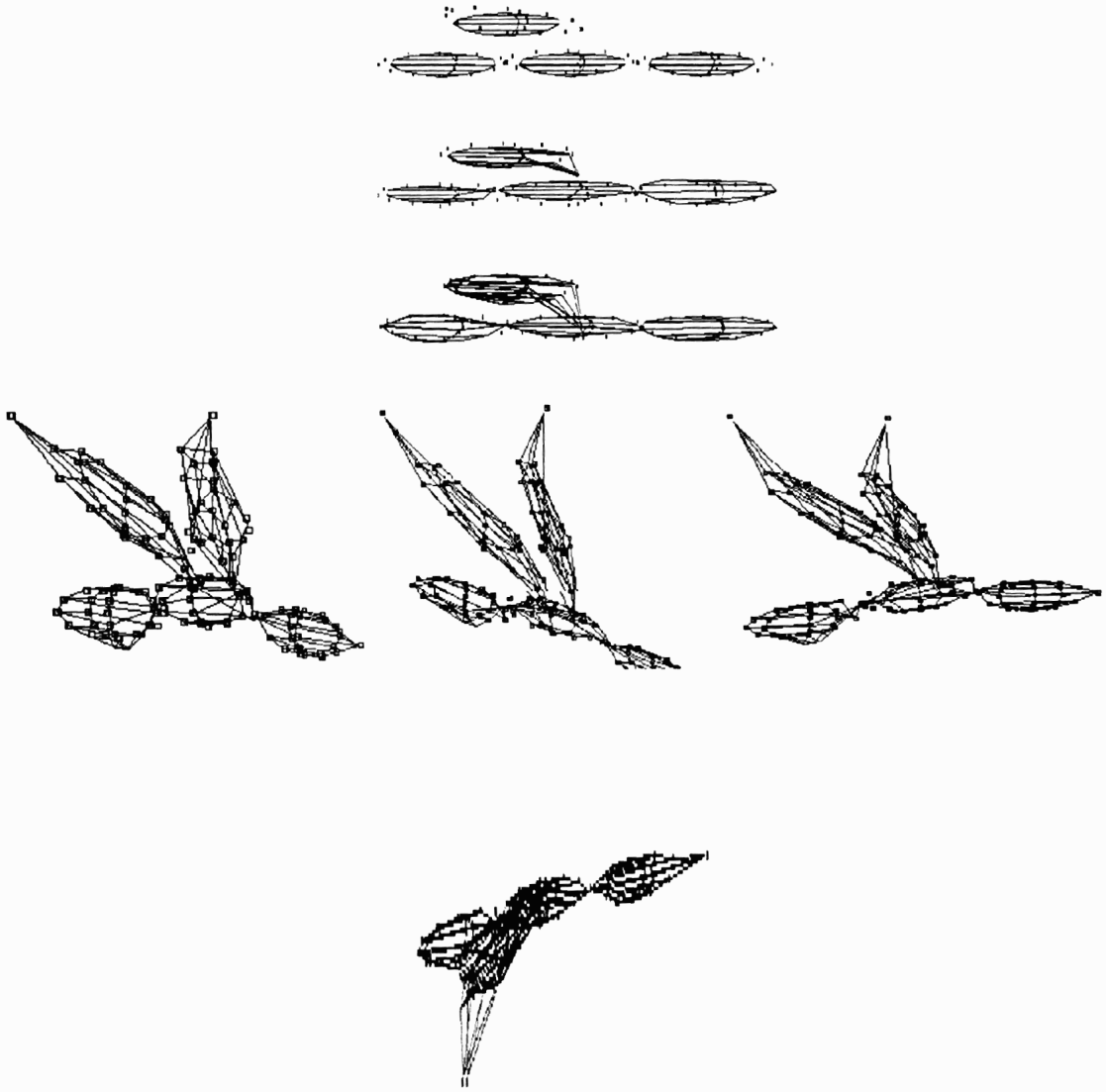


Figure 9.3: Tracking of an insect's parts.

$(1.6, 1.0, 0.2, 0.8, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0)^T$). Fig. 9.2(b) shows a view of the parts fitted to the datapoints in the first time frame. The constraint forces help position the four parts in the correct configuration. Fig. 9.2(c) shows a subsequent frame of the superquadric chain tracking downward motion. The constraint forces deform the occluded part to connect the chain properly and they impart on the occluded part a motion that is consistent with the motion of its neighbors. Of course the shape of the occluded part is inaccurate, since no data are available to deform it to the correct cross section.

Finally we report two more experiments in which we used our force-based tracking algorithm. Fig. 9.3 illustrates the tracking of an articulated "insect" consisting of five deformable superquadric parts, each having 27 nodes. We again synthesized datapoints by simulating the motions of constrained deformable superquadrics undergoing global and local deformations and sampling 27 datapoints from each superquadric through time (36 frames). The nonrigid motions were imparted by three forces, a force applied to each wing tip to make the wings flap and a force applied to the nose to pull the insect forward through space. The local deformation stiffness parameters were set to $w_0 = 0.1$ and $w_1 = 0.5$. To demonstrate the performance of our constrained motion algorithm, we initialize the deformable superquadrics models to ellipsoidal shapes ($\mathbf{q}_s = (2.0, 0.8, 0.07, 0.07, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.5)^T$) as shown in Fig. 9.3(a). Fig. 9.3(b) shows an intermediate step in fitting the models to the datapoints associated with the first time frame, while Fig. 9.3(c) shows the final fit to these initial datapoints. Again, the constraint forces help configure the five parts correctly, as is evident from the final deformations of the wings. Fig. 9.3(d) shows three time frames of the fitted insect model tracking the time-varying datapoints as the wings are open upwards, while Fig. 9.3(e) shows a frame of the motion with the wings dipping downward.

Fig. 9.4 illustrates a sparse datapoint version of the previous experiment. We use the same parameter settings, but now there are only 7 datapoints per superquadric. Again, the constraint forces help link the five superquadrics into the correct configuration. The shapes of the wings in Fig. 9.4(d) differ from the shapes in the previous insect experiment. The sparse data do not provide enough constraints for the local deformations to recover the exact shapes.

The above experiments indicate the tradeoffs between local and and global deformations in the sparse data case. Global deformations require relatively few datapoints to abstract the shapes of objects. By contrast, local deformations can provide a more accurate approximation to the exact shape of a complex object, but their recovery generally requires more data. The symbiosis of local and global deformations within our dynamic models appears to offer the best of both worlds.

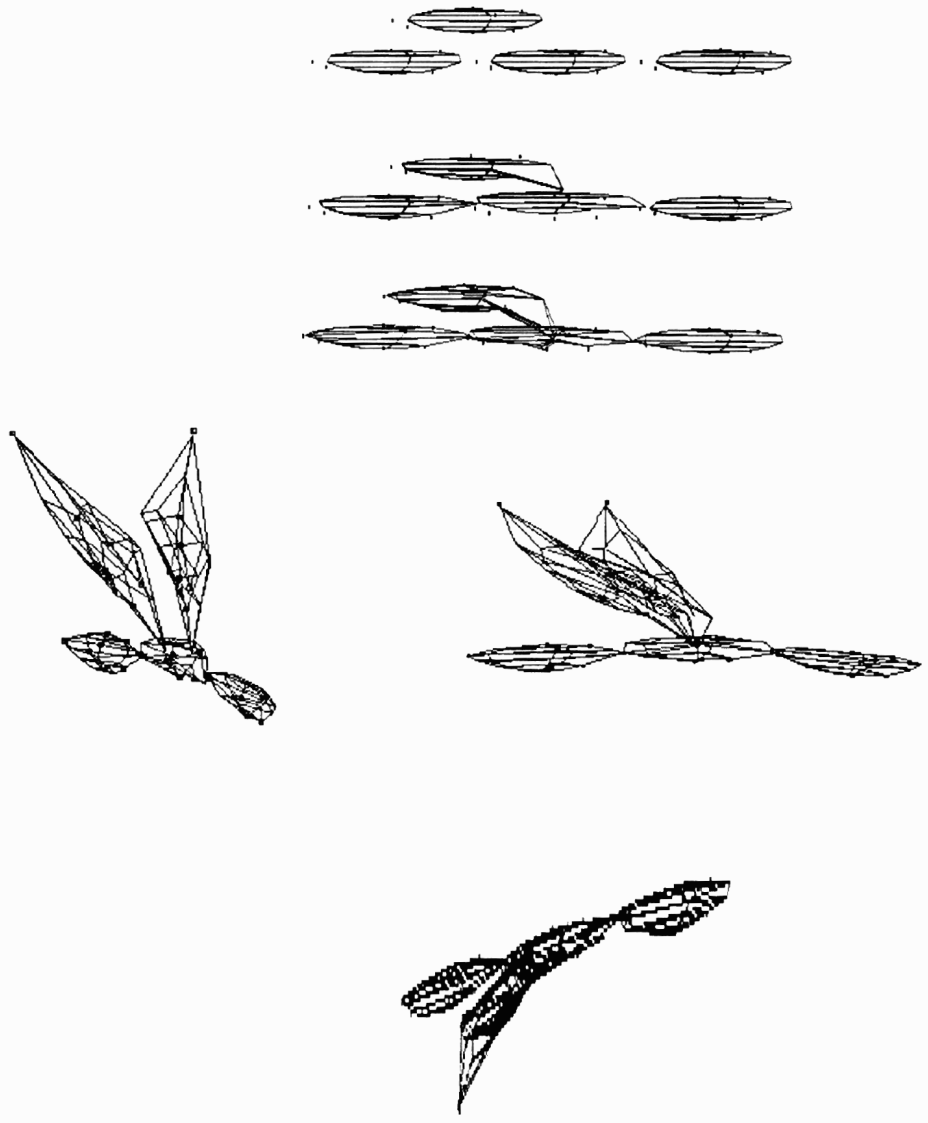


Figure 9.4: Tracking of an insect's parts using sparse data.

9.2 Computer Graphics Experiments

We have created several real time physics-based animation examples involving constrained complex models made from deformable superquadric primitives.

Fig. 9.5 shows several frames from an animation of two deformable superquadric "balloons" suspended in gravity by flexible but inextensible strings attached to a ceiling using point-to-point constraints. Point-to-point constraints also connect the balloons to the strings. 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. 9.5(a) shows the initial configuration with the left balloon pulled to the side. Gravity is activated in Fig. 9.5(b). The balloons deform under their own weight. The left balloon swings to the right, colliding inelastically with its neighbor in Fig. 9.5(c), thereby transferring some of its kinetic energy. In Figs. 9.5(d–f) the balloons collide repeatedly until all the kinetic energy is dissipated. The collisions are implemented using reaction constraints [54] between multiple deformable bodies. Fig. 9.6 shows a similar scenario involving three balloons. By deforming, the middle balloon cushions the left balloon from the blow of the collision. It therefore swings a shorter distance than the left balloon did in the 2-balloon animation.

Fig. 9.7 shows the automatic construction of a minimalist dragonfly from its constituent deformable superquadric parts. Fig. 9.7(a) shows the disjoint parts in their initial configurations. The elastic parameters were $w_0 = 0.1$ and $w_1 = 0.5$, the Euler step 0.0031, the nodal mass 2.0 and the damping coefficient $\nu = 1.6$. After activating our constraint algorithm, the model self-assembles (in a similar fashion to the self-assembling models in [71, 5]) to form the articulated dragonfly shown in Figs. 9.7(b–c). 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 Figs. 9.7(d–f). An impenetrable plane appears in Fig. 9.7(g) to swat the dragonfly in the rear (Fig. 9.7(h)). The body

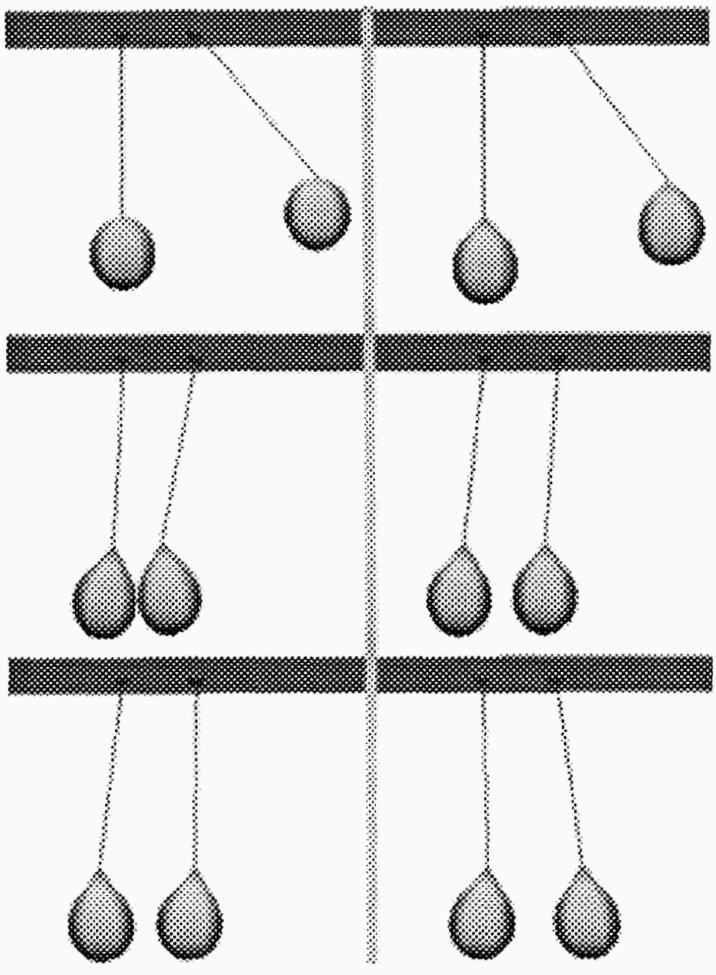


Figure 9.5: Two balloon pendulums.

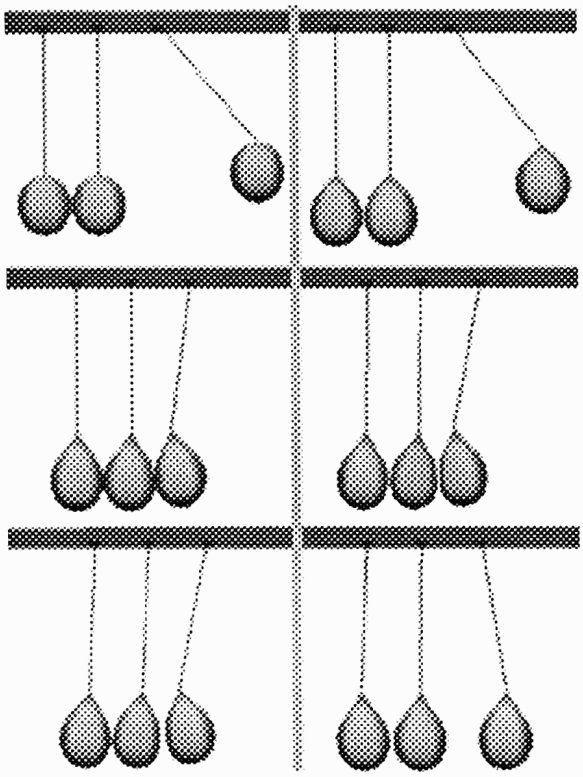


Figure 9.6: Three balloon pendulums.

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. 9.7(i).

Fig. 9.8 illustrates the self-assembly and subsequent animation of a snowman. Figs. 9.8(a–b) show two views of the disjoint deformable superquadric body parts of a snowman. The elastic parameters of each part were $w_0 = 0.1$ and $w_1 = 0.5$, the Euler step 0.001, the nodal mass 1.0, the damping coefficient $\nu = 200.0$, the plane elasticity 200.0 and the friction coefficient 0.15. The snowman self-assembles when the constraints are activated (Fig. 9.8(c)). 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 (Figs. 9.8(d–f)).

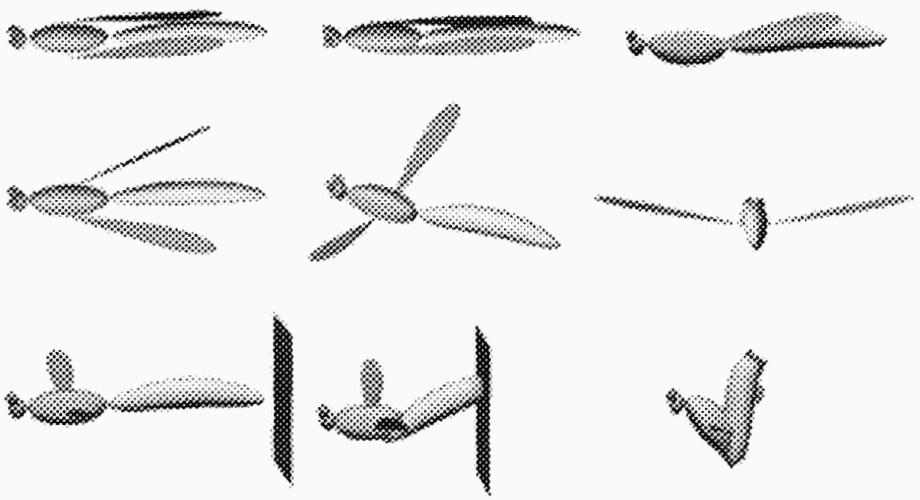


Figure 9.7: Self-assembly, articulation, and swatting of a dragonfly.

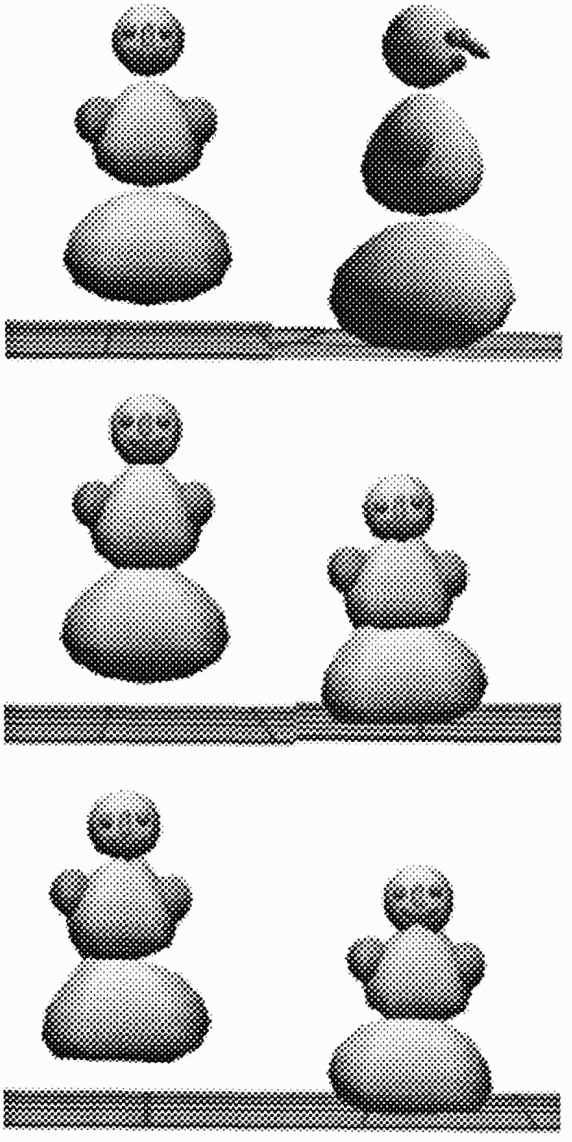


Figure 9.8: Self-assembly and animated hopping of a snowman.

Chapter 10

Shape and Nonrigid Motion Estimation

This chapter describes 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.

10.1 Recursive Estimation

The force based estimation scheme described in Chapter 6 employs data forces (6.16) 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 [22]. 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 (4.49) and (6.14) the nonlinear Kalman filter equations for our dynamic model take the form

$$\begin{aligned} \dot{\mathbf{u}} &= \mathbf{F}\mathbf{u} + \mathbf{g}_1 + \mathbf{w} \\ \mathbf{z} &= \mathbf{h}(\mathbf{u}) + v, \end{aligned} \tag{10.1}$$

where

$$\mathbf{g}_1 = \begin{bmatrix} \mathbf{M}^{-1}(\mathbf{g}_q + \mathbf{f}_{g_c}) \\ \mathbf{0} \end{bmatrix} \tag{10.2}$$

($\mathbf{g}_1 = \mathbf{D}^{-1}\mathbf{f}_{g_c}$ 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., $E[\mathbf{w}(t)\mathbf{v}^\top(\tau)] = 0$) is

$$\dot{\hat{\mathbf{u}}} = \mathbf{F}\hat{\mathbf{u}} + \mathbf{g}_1 + \mathbf{P}\mathbf{H}^\top\mathbf{V}^{-1}\left(\mathbf{z} - \mathbf{h}(\hat{\mathbf{u}})\right), \tag{10.3}$$

[1] Kalman filtering is optimal for linear system and measurement models, assuming the associated noise processes are Gaussian [22]. 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.

where $\mathbf{H}$ is computed using (6.17). The expression $\mathbf{G}(t) = \mathbf{P}\mathbf{H}^\top\mathbf{V}^{-1}$ is known as the Kalman gain matrix. The symmetric error covariance matrix $\mathbf{P}(t)$ is the solution of the matrix Riccati equation

$$\dot{\mathbf{P}} = \mathbf{F}\mathbf{P} + \mathbf{P}\mathbf{F}^\top + \mathbf{Q} - \mathbf{P}\mathbf{H}^\top\mathbf{V}^{-1}\mathbf{H}\mathbf{P}. \tag{10.4}$$

Note that the term $\mathbf{P}\mathbf{H}^\top\mathbf{V}^{-1}\,(\mathbf{z}-\mathbf{h}(\hat{\mathbf{u}}))$ in (10.3) is a generalized force. It results from the instantaneous disparity between the measurements $\mathbf{z}$ and the estimated state of the model $\hat{\mathbf{u}}$. For unit covariance matrix $\mathbf{P}(t) = \mathbf{I}$, this term reduces to the generalized data forces (6.16) which are applied to dynamic models in force-based estimation [67, 70]. 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 $\hat{\mathbf{u}}$ [22]. 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 (10.3) 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.

10.2 Kalman Filter Implementation

The continuous Kalman estimation equations (10.3) and (10.4) can be integrated using standard numerical techniques; the simplest being Euler's method.

The cost of computing (10.4) 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 [22]. 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 (4.49) 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 (10.4), 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 $\mathbf{K}$ [67]; 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 $\mathbf{Q}$ [22]. The latter are therefore considered as filter tuning parameters. If the entries of $\mathbf{Q}$ are inappropriate for the particular application, the

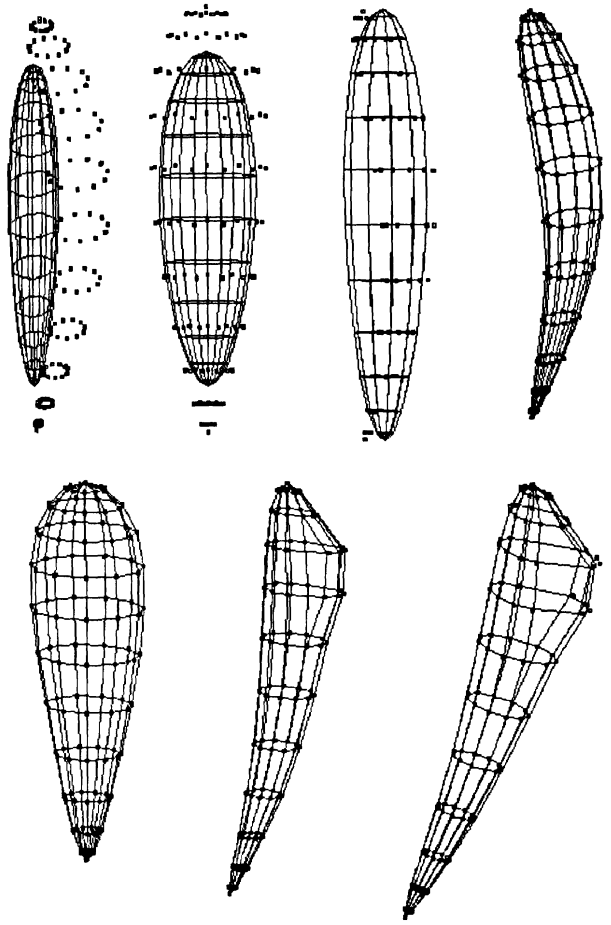


Figure 10.1: Tracking of fully deformable squash shaped object.

filter may diverge or converge to the wrong value [11]. This is due to the nonlinear measurement equations (10.1). The entries of $\mathbf{Q}$ should therefore be tuned so that the filter converges to the correct solution.

10.3 Recursive Estimation of Shape and Nonrigid Motion

This section presents several quantitative experiments involving synthetic data. We employ motion sequences of squash shaped objects in which we used our Kalman estimation algorithm. In Fig. 10.1 we fit a deformable superquadric model with 123 nodes to 123 3D datapoints. The datapoints are sampled over

20 frames from the surface of a synthesized squash-like model undergoing both global and local deformations in response to a time-varying force applied at the upper left of the squash. Figs. 10.1(a) and (b) show two views of the range data and the initial model. The local deformation stiffness parameters of the model were $w_0 = 0.1$ and $w_1 = 0.5$ (see [67]). The initial model was an ellipsoid ($\mathbf{q}_s = (2.7, 0.1, 0.2, 0.7, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.2)^\mathsf{T}$). The diagonal entries of the initial covariance matrix $\mathbf{P}(0)$ were set to 1.0 while the off-diagonal entries used in the estimation of local deformations were set to 0.4. We also set $\mathbf{Q} = 0.1\ \mathbf{I}$ and $\mathbf{V} = 0.1\ \mathbf{I}$. In order to demonstrate the performance of the model fitting algorithm, we did not initialize the part models at the center of gravity of the data. Fig. 10.1(c) shows an intermediate step of the fitting process driven by data forces from the first frame of the motion sequence, while Figs. 10.1(d) and (e) show the model fitted to the initial data, with visible tapering and bending global deformations. Fig. 10.1(f) shows an intermediate frame of the model tracking the nonrigid motion of the squash, while Fig. 10.1(g) shows the final position of the squash; the local deformations are now readily apparent.

To assess the performance of our estimation algorithm quantitatively, we conducted a series of error analysis experiments with data points from the above experiment at 100% data density and also decimated to 75%, 50% and 25% data densities. We corrupted the data with zero-mean independent Gaussian noise of variance 0, 1, 5, and 10. We initialized the estimator as in the above experiment. In each run we compute the average error per frame in the estimated translation, rotation, global and local deformation, as well as the error in the estimated nodal positions of the model using the formula

$$err_{\mathbf{a}} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\mathbf{a}_{e_i} - \mathbf{a}_{T_i})^\mathsf{T}(\mathbf{a}_{e_i} - \mathbf{a}_{T_i})}, \tag{10.5}$$

where $\mathbf{a}$ stands for $\mathbf{q}_c$, $\mathbf{q}_\theta$, $\mathbf{q}_s$, $\mathbf{q}_d$, or $\mathbf{x}$, the subscripts e and T denote estimated and true values respectively, and n is the number of frames used in the experiment—in this case, 20.

The following tables show the error analysis results for the various data percentages and noise variances.

Variance	Translation Error	Rotation Error	Global Error	Local Error	Position Error
10	0.06	0.02	0.19	0.65	0.71
5	0.04	0.005	0.18	0.43	0.44
1	0.039	0.003	0.117	0.1	0.09
0	0.025	0.002	0.011	0.03	0.014

Table 1. Noise experiment. Data density 100%.

Variance	Translation Error	Rotation Error	Global Error	Local Error	Position Error
10	0.062	0.027	0.22	0.67	0.86
5	0.05	0.007	0.19	0.48	0.49
1	0.04	0.004	0.17	0.37	0.29
0	0.039	0.0034	0.15	0.27	0.24

Table 2. Noise experiment. Data density 75%.

Variance	Translation Error	Rotation Error	Global Error	Local Error	Position Error
10	0.07	0.03	0.25	0.99	1.07
5	0.06	0.015	0.19	0.66	0.62
1	0.05	0.005	0.18	0.48	0.39
0	0.045	0.0036	0.19	0.43	0.32

Table 3. Noise experiment. Data density 50%.

Fig. 10.2 shows a detailed plot, over the 20 frames, of the error in the estimated nodal positions from the 75% data density, noise variance 1 experiment. Given an inaccurate initial model, the error drops sharply as the algorithm rapidly estimates the correct nodal positions. The modest increase in error up to frame 8 and its subsequent decrease is due to the increasing size of the local deformation implied by the data. The filter tunes after 8 frames (local deformations are now significant) and we observe a subsequent decrease in the frame error.

Variance	Translation Error	Rotation Error	Global Error	Local Error	Position Error
10	0.092	0.036	0.29	1.03	1.24
5	0.09	0.023	0.25	0.7	0.95
1	0.089	0.016	0.2	0.6	0.64
0	0.068	0.013	0.17	0.57	0.53

Table 4. Noise experiment. Data density 25%.

The above error analysis experiments first demonstrate that the recovery of translation, rotation and global deformations is more robust than the recovery of local deformations in the presence of noise. Secondly, the Kalman filter ignores some of the noise by not forcing the local deformations to interpolate the data-points. It produces reasonably accurate shape estimation by keeping the overall position estimation error small. We observed that the reliability of the estimated motion parameters began to fail at data densities of around 25% and variances around 10.

The following experiment indicates the performance of the method for non-Gaussian noise[2] and when there is no one-to-one correspondence between the model nodes and the data points. In Fig. 10.3 we add uniform noise, by perturbing by $\pm 5\%$ (with randomly chosen sign) the noiseless value of the 123 motion range data

[2]The estimation of accurate noise models from input data is beyond the scope of this thesis.

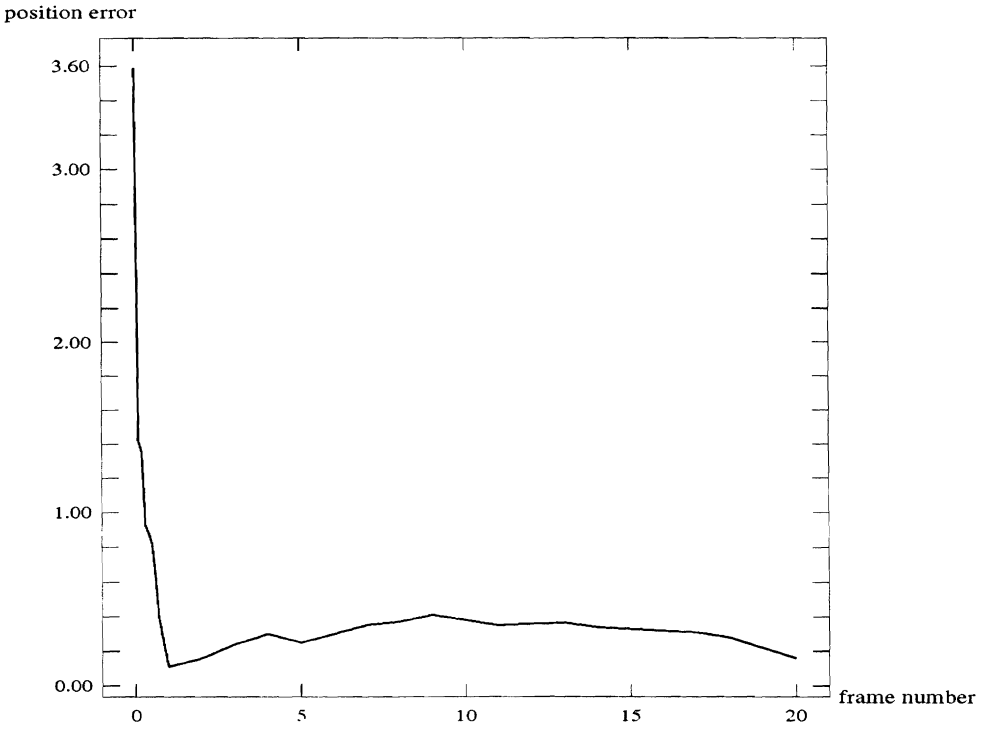


Figure 10.2: Position error per frame with 75% of the data and noise variance 1.0.

point. We fit a deformable superquadric with 81 nodes whose parameters and initial position are the same as in the first experiment, except for the initial ellipsoid parameters which were ($\mathbf{q}_s = (2.8, 0.1, 0.2, 0.8, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.2)^T$) and we set $\mathbf{V} = 6.0\,\mathbf{I}$ to account for noise. Fig. 10.3 is similar to Fig. 10.1. Figs. 10.3(a) and (b) show two views of the range data and the initial model. Fig. 10.3(c) shows an intermediate step of the fitting process driven by data forces from the first frame of the motion sequence. Figs. 10.3(d) and (e) show the model fitted to the initial data, with visible tapering and bending global deformations. Fig. 10.3(f) shows an intermediate frame of the model tracking the nonrigid motion of the squash, while Figs. 10.3(g) and (h) show the final position of the squash.

We conducted three more experiments involving real 3D data from arm and torso motions of a human subject using the WATSMART system. Approximately 120 time frames were used in this 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 superquadric part to model the arms and

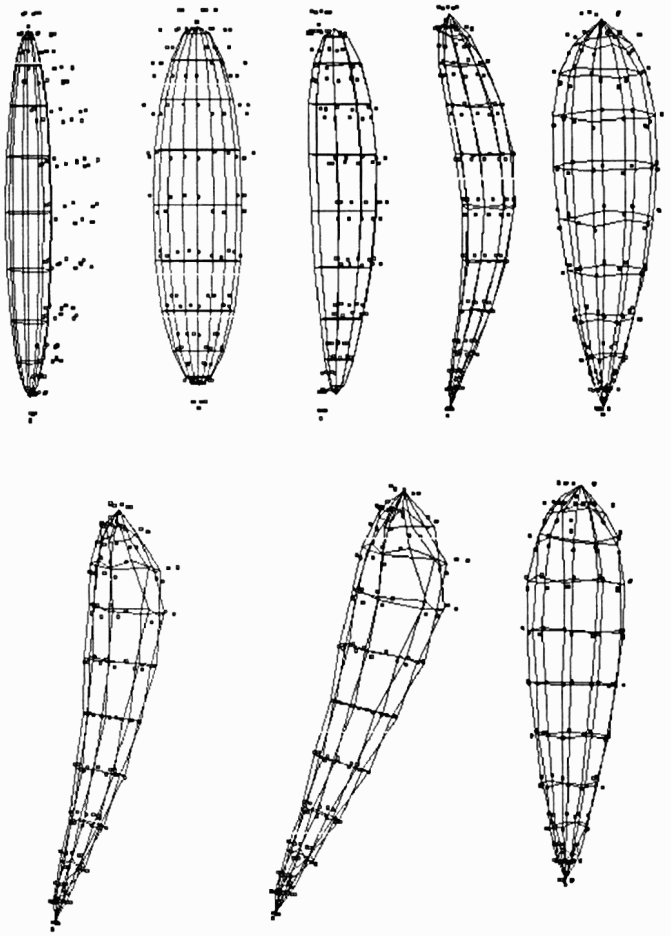


Figure 10.3: Tracking of fully deformable squash shaped object with noise.

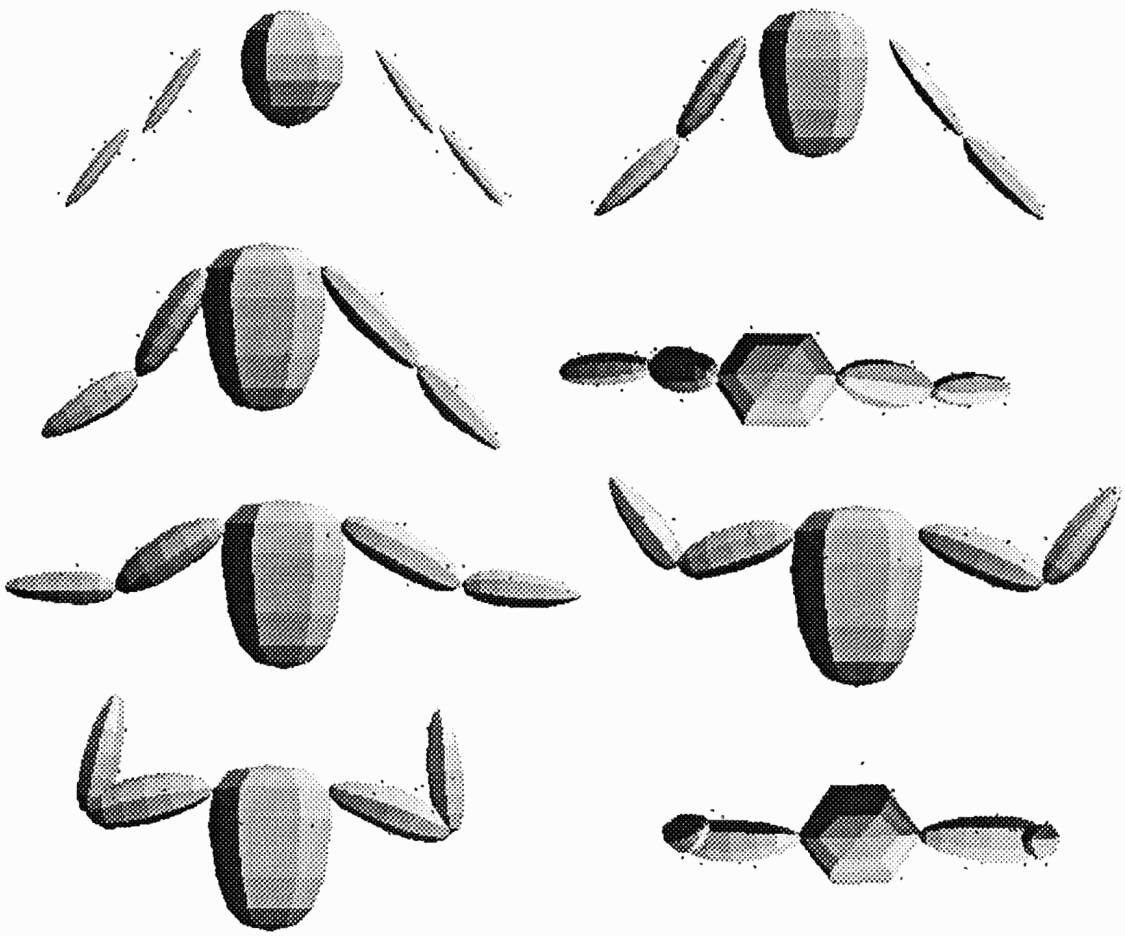


Figure 10.4: Tracking of raising and flexing human arm motion.

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 [61]. 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 this experiment, since the available data were very sparse, i.e., about 6 to 13 points per deformable superquadric part.

Fig. 10.4 shows shape and motion estimation results involving an articulated multipart model composed of 5 connected deformable superquadrics using data collected from the raising and flexing motion of the two arms of a human subject. Fig. 10.4(a) shows a view of the range data and the initial models. Fig. 10.4(b) shows an intermediate step of the fitting process driven by data forces from the first frame of the motion sequence, while Figs. 10.4(c) and (d) show the models fitted to the initial data. Figs. 10.4(e) and (f) show intermediate frames of the models tracking the nonrigid motion of the arms, while Figs. 10.4(g) and (h) show two views of the final position of the models.

In Fig. 10.5 we fit 3 deformable superquadrics to data collected from the rotating motion of the right arm of a human subject. Figs. 10.5(a) and (b) show two viewpoints of the range data and the initial models. Figs. 10.5(c) and (d) show an intermediate step of the fitting process driven by data forces from the first frame of the motion sequence, while Figs. 10.5(e) and (f) show the models fitted to the initial data. Each of the remaining three frames show two viewpoints each, of the models tracking the rotational motion of the arm.

In Fig. 10.6 we fit 3 deformable superquadrics to data collected from the up-down motion of the right arm of a human subject. Figs. 10.6(a) and (b) show two viewpoints of the range data and the initial models. Fig. 10.6(c) shows an intermediate step of the fitting process driven by data forces from the first frame of the motion sequence, while Figs. 10.6(d) and (e) show the models fitted to the

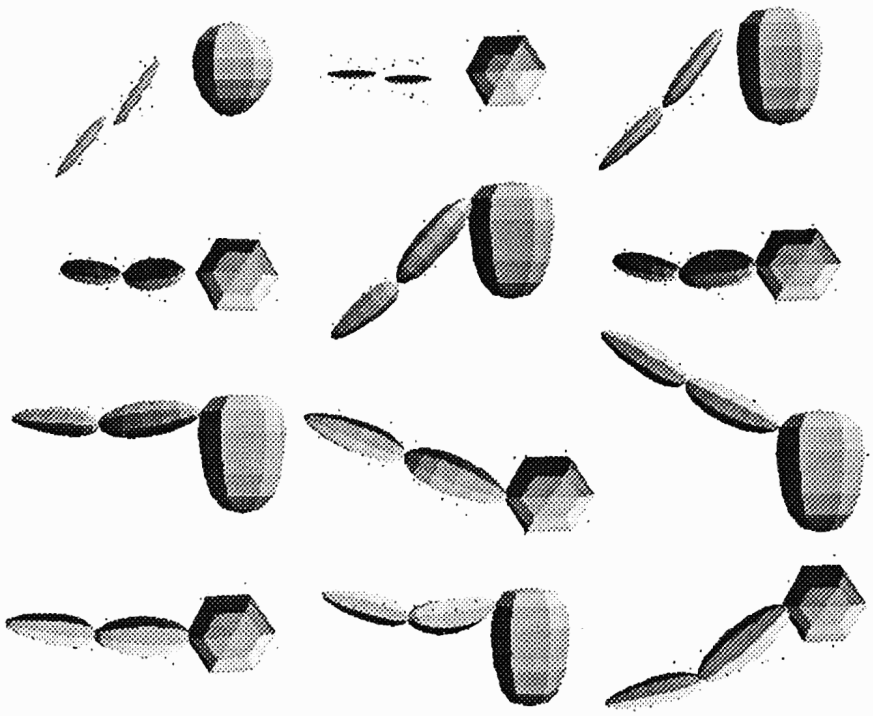


Figure 10.5: Tracking of rotating human arm motion.

initial data. Fig. 10.6(f) shows an intermediate frame of the models tracking the nonrigid motion of the arm, while Figs. 10.6(g) and (h) show two viewpoints 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.

10.4 Summary

In this chapter we applied continuous nonlinear Kalman filtering theory, to construct a recursive estimator which employs the Lagrange equations of 3D nonrigid motion that we have developed as a system model. This estimator allows the recovery of shape and nonrigid motion in the presence of noise. We also described a computationally efficient implementation of the Kalman filter equations. Finally, we presented computer vision experiments involving shape and nonrigid motion

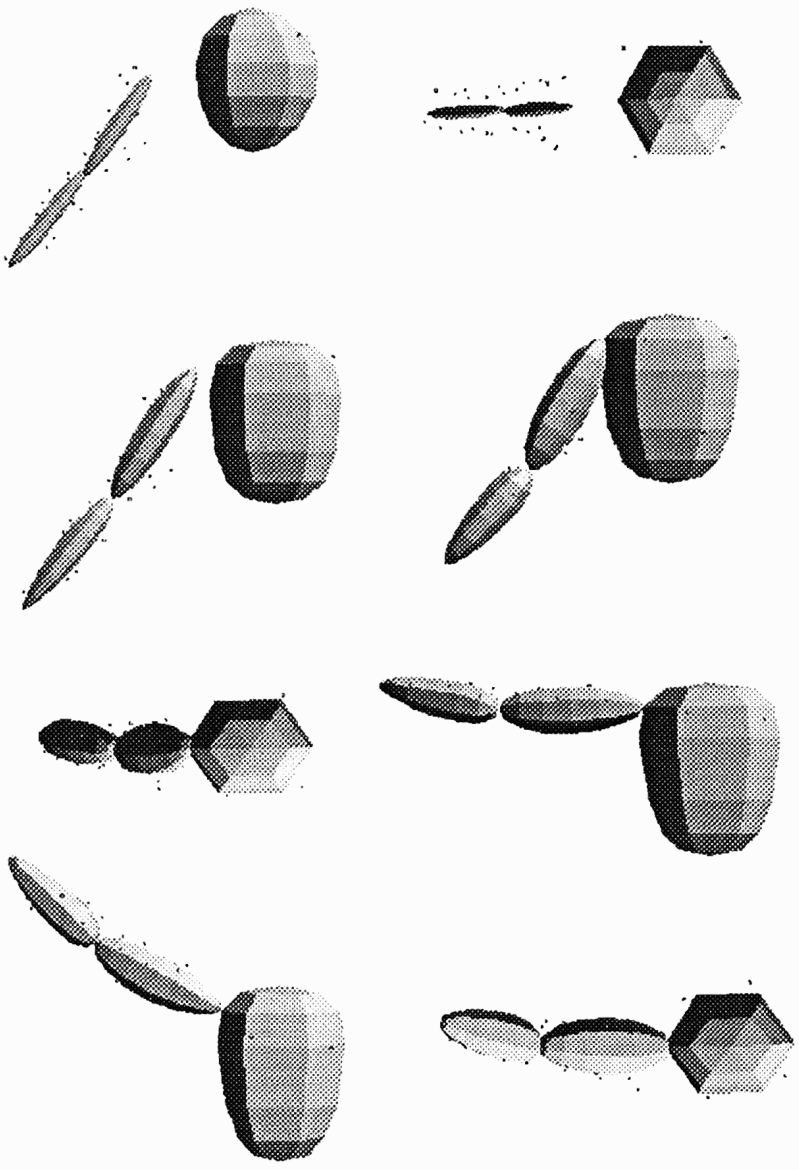


Figure 10.6: Tracking of up-down human arm motion.

estimation from 3D range data, using the recursive estimation techniques based on Kalman filtering.

Chapter 11

Conclusions

We conclude by giving a summary of the thesis and proposing directions for future research.

11.1 Summary

This thesis developed a physics-based framework for 3D shape and nonrigid motion modeling for computer vision and computer graphics. The framework features a new class of dynamic deformable part models. It incorporates physical constraints to compose articulated models from deformable parts and provides force-based techniques for fitting such models to sparse, noise-corrupted 2D and 3D visual data. The framework leads to estimators that exploit dynamic deformable models to track moving 3D objects from time-varying observations. The thesis demonstrated the usefulness of the framework in several application areas. These include quantitative model extraction from biomedical data for analysis and visualization. The modeling framework also provides the necessary generative power to synthesize constrained shapes and nonrigid motions for the purposes of computer animation.

We developed models with global deformation parameters which represent the salient shape features of natural parts, and local deformation parameters which capture shape details. Our approach is general and can be used to combine parameterized geometric primitives, parameterized global deformations, and local

deformations. In the context of computer graphics, these models combine the parameterized and free-form modeling paradigms. An important benefit of their global/local descriptive power in the context of computer vision is that it can potentially satisfy the conflicting requirements of shape reconstruction and shape recognition.

The Lagrange equations of motion that govern our models, augmented by constraints, make them responsive to externally applied forces derived from input data or stemming from interactions of our deformable models with the physical world or from the user. This system of differential equations is discretized using finite element methods and simulated through time using standard numerical techniques.

We also developed a recursive technique for 3D shape modeling and nonrigid motion estimation from incomplete, noise-corrupted, time-varying observations. Assimilating our dynamic models within continuous nonlinear Kalman filtering theory, we derived recursive algorithms to estimate translation, rotation, and deformation parameters of nonrigid objects. Our estimators are much more complex and powerful than previous Kalman filter based estimators described in the vision literature (including our earlier force-based estimators [67] which may be viewed as degenerate "Kalman filters" with unit error covariance matrices).

The formulations in this thesis were purposefully general, since the general models may become important in off-line analysis tasks and in interactive tasks on faster computers. Nonetheless, we showed that it is possible to ease significantly the computational burden of our recursive estimation algorithms with minimal loss of accuracy through various means, such as through state decoupling. We are confident that with these reasonable simplifications our framework can support fast, on-line estimation of nonrigid motion parameters on currently available vision hardware for real-time applications.

We demonstrated the interactive rate time performance of our techniques in a series of experiments in computer vision, graphics, and visualization.

11.2 Future Work

In the area of Physics-Based Modeling and Simulation, it is possible to conduct further research into biomedical applications of the techniques that we have developed. Such applications include modeling and simulation of the physical properties of various body tissues and organs such as skin, muscles, and bones and the reconstruction of internal or external body parts from MRI and CT data. The approach promises to be useful, for instance, in assessing patient morphology, surgical planning, prostheses, fabrication, and athletic fitness. With the experience gained from the modeling and simulation of flexible objects, we would also like to move towards the simulation of other complex physical phenomena such as fluid motion and turbulence to support visualization that will promote their better understanding.

In Computer Vision, we plan to extend models and techniques in various directions. We would like to explore methods for the estimation of the local elastic parameters of our models. This is needed to improve the reconstruction of complex deformable objects, such as a heart, since such objects do not have uniform elastic properties. We would also like to show the usefulness of our models in recognition tasks and we would work on the development and implementation of fast 2D versions of our models for the purposes of real-time active vision. Finally, we would like to work towards the development of an autonomous vision system, through the integration of visual modules, including the one developed in this thesis. A very challenging problem in vision apart from improving existing modules which attempt to solve restricted subsets of the vision problem, is the question of integrating these modules to achieve a general purpose performance that will be of practical significance.

In Computer Graphics and Animation, our next step would be research into the creation of tools and the further development of our algorithms for faster and more realistic animations. Then we would investigate the application of control theory to the models developed in this thesis. This promises to be critical to the creation of complicated animations in a user-friendly way.

Appendix A

Derivation of B matrix

Shabana [59] has shown that

$$\dot{\mathbf{R}}\mathbf{p} = \mathbf{R}(\boldsymbol{\omega}_\phi \times \mathbf{p}), \tag{12.1}$$

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 o.

Furthermore, in [59] 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

$$\boldsymbol{\omega}_\phi = \mathbf{G}\dot{\mathbf{q}}_\theta, \tag{12.2}$$

where $\mathbf{G}$ is a 3×4 matrix whose definition is based on the value of the quaternion $\mathbf{q}_\theta = [s, \mathbf{v}_q]$,

$$\mathbf{G} = 2\begin{bmatrix} -v_1 & s & v_3 & -v_2 \\ -v_2 & -v_3 & s & v_1 \\ -v_3 & v_2 & -v_1 & s \end{bmatrix}. \tag{12.3}$$

Therefore, we can rewrite the vector $\dot{\mathbf{R}}\mathbf{p}$ using (12.2) as follows

$$\begin{aligned} \dot{\mathbf{R}}\mathbf{p} &= \mathbf{R}(\boldsymbol{\omega}_\phi \times \mathbf{p}) = -\mathbf{R}(\mathbf{p} \times \boldsymbol{\omega}_\phi) \\ &= -\mathbf{R}\ \tilde{\mathbf{p}}\ \mathbf{G}\ \dot{\mathbf{q}}_\theta, \end{aligned} \tag{12.4}$$

where $\tilde{\mathbf{p}}$ is the dual 3×3 matrix of the position vector $\mathbf{p}(u) = (p_1, p_2, p_3)^\top$ (see (3.3)) defined as

$$\tilde{\mathbf{p}}(u) = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}. \qquad (12.5)$$

Comparing (12.4) and (4.1), we identify matrix $\mathbf{B}$ as

$$\mathbf{B} = -\mathbf{R}\ \tilde{\mathbf{p}}\ \mathbf{G}. \qquad (12.6)$$

Appendix B

Integration Rules

B.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

$$I = \int_{-1}^{1} f(\xi) d\xi. \tag{13.7}$$

Then using the Gauss-Legendre rules we sample the function $f(\xi)$ at some predetermined sampling position $\xi = \bar{\xi}$ and multiply the results by some predetermined weights W_i. Thus for a n-point rule

$$I_n = \sum_{i=1}^{n} W_i \ f(\bar{\xi}_i). \tag{13.8}$$

The following table shows the sampling positions and weights for the first four Gauss-Legendre rules [29].

n	$\bar{\xi}_i$	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 1. Gauss-Legendre integration rules

Note that an n-point rule exactly integrates a polynomial of degree $(2n-1)$ or less exactly [29].

For an isoparametric quadrilateral let $g(\xi,\eta)$ be the function we want to integrate in the *two-dimensional* parameter space. Then using (13.8)

$$\begin{aligned}\int_{-1}^{1}\int_{-1}^{1} g(\xi,\eta)d\xi\ d\eta &= \int_{-1}^{1}\sum_{i=1}^{n} W_i\ g(\bar{\xi}_i,\eta)\ d\eta \\ &= \sum_{i=1}^{n}\sum_{i=1}^{n} W_iW_j\ g(\bar{\xi}_i,\bar{\eta}_i), \qquad (13.9)\end{aligned}$$

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 $(\bar{\xi}_i,\bar{\eta}_i)$.

B.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 [29]

$$\int_E g(\xi,\eta)\ d\xi\ d\eta = \sum_{i=1}^{n} W_i g(\bar{\xi}_i,\bar{\eta}_i), \qquad (13.10)$$

where the weights and the sampling positions are given in the following table.

n	i	$\bar{\xi}_i$	$\bar{\eta}_i$	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 2. Weights and sampling positions for numerical integration over triangular regions (Radau rules).

Bibliography

[1] D. Baraff. Coping with Friction for Non-penetrating Rigid Body Simulation. *Computer Graphics (Proc. SIGGRAPH)*, 25(4):31–40, 1991.

[2] D. Baraff and A. Witkin. Dynamic Simulation of Non-penetrating Flexible Bodies. *Computer Graphics (Proc. SIGGRAPH)*, 26(2):303–308, 1992.

[3] A. Barr. Superquadrics and Angle-Preserving Transformations. *IEEE Computer Graphics and Applications*, 1:11–23, 1981.

[4] A. Barr. Global and Local Deformations of Solid Primitives. *Computer Graphics*, 18:21–30, 1984.

[5] R. Barzel and A. Barr. A Modeling System Based on Dynamic Constraints. *Computer Graphics*, 22(4):179–188, 1988. Proc. ACM SIGGRAPH'88.

[6] K.-J. Bathe and E.L. Wilson. *Numerical Methods in Finite Element Analysis*. Prentice–Hall, Englewood Cliffs, NJ, 1976.

[7] J. Baumgarte. Stabilization of Constraints and Integrals of Motion in Dynamical Systems. *Computer Methods in Applied Mechanics and Engineering*, 1:1–16, 1972.

[8] I. Biederman. Human Image Understanding: Recent Research and Theory. *Computer Vision, Graphics, and Image Processing*, 32:29–73, 1985.

[9] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, 1987.

[10] T. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):90–98, January 1986.

[11] T. J. Broida, S. Chandrashekhar, and R. Chellapa. Recursive 3-D Motion Estimation from a Monocular Image Sequence. *IEEE Transactions on Aerospace and Electronic Systems*, 26(4):639–656, July 1990.

[12] R. Brooks. Symbolic Reasoning among 3D Models and 2D Images. *Artificial Intelligence*, 17:285–348, 1981.

[13] G. Celniker. Deformable Curve and Surface Finite Elements for Free-form Shape Design. *Computer Graphics (Proc. SIGGRAPH)*, 25:257–266, 1991.

[14] J.E. Chadwick, D.R. Haumann, and R.E. Parent. Layered Construction for Animated Deformable Characters. *Computer Graphics (Proc. SIGGRAPH)*, 23(3):243–252, 1989.

[15] R. Deriche and O. Faugeras. Tracking Line Segments. *Image and Vision Computing*, 8(4):261–270. November 1990.

[16] G. Dhatt and G. Touzot. *The Finite Element Method Displayed.* Wiley, New York, 1984.

[17] 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):174–198, February 1992.

[18] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld. From Volumes to Views: An Approach to 3-D Object Recognition. *Computer Vision Graphics and Im-*

age Processing:Image Understanding,special issue on CAD-based vision, 55(2), March 1992.

[19] E. D. Dickmannns and Volker Graefe. Applications of Dynamic Monocular Machine Vision. *Machine Vision and Applications*, 1:241–261, 1988.

[20] E. D. Dickmannns and Volker Graefe. Dynamic Monocular Machine Vision. *Machine Vision and Applications*, 1:223–240, 1988.

[21] M. Gardiner. The Superellipse: A Curve Between the Ellipse and the Rectangle. *Scientific American*, 213:222–234, 1965.

[22] A. Gelb. *Applied Optimal Estimation*. MIT Press, Cambridge, MA, 1974.

[23] 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):21–30, 1989.

[24] 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.

[25] 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.

[26] D. Haumann. Modeling the Physical Behavior of Flexible Objects. *Topics in Physically-Based Modeling, Barr, A., et al. (eds.), ACM SIGGRAPH '87 Course Notes*, 17, 1987.

[27] J. Heel. Temporally Integrated Surface Reconstruction. In *IEEE Third International Conference on Computer Vision (ICCV'90), Osaka, Japan, Dec.*, pp. 292–295, 1990.

[28] E. Hinton and D. Owen. *Finite Element Programming.* Academic Press, 1977.

[29] E. Hinton and D. Owen. *An Introduction to Finite Element Computations.* Pineridge Press, Swansea, U.K., 1979.

[30] C.M. Hoffmann. *Geometric and Solid Modeling.* Morgan-Kaufmann, Palo Alto, 1989.

[31] H. Kardestuncer. *Finite Element Handbook.* McGraw–Hill, New York, 1987.

[32] M. Kass and G. Miller. Rapid. Stable Fluid Dynamics for Computer Graphics. *Computer Graphics (Proc. SIGGRAPH)*, 24(4):49–57, 1990.

[33] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[34] 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.

[35] 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.

[36] D. Marr and H. K. Nishihara. Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes. *Proc. Roy. Soc. London, B*, 200:269–294, 1978.

[37] L. Matthies, T. Kanade, and R. Szeliski. Kalman Filter-based Algorithms for Estimating Depth from Image Sequences. *International Journal of Computer Vision*, 3:209–236, 1989.

[38] 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.

[39] 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.

[40] D. Metaxas and D. Terzopoulos. A Physically-based Framework for Shape and Nonrigid Motion Estimation. *Proc. IJCAI-91 Workshop on Dynamic Scene Understanding, Sydney, Australia*, August 1991.

[41] D. Metaxas and D. Terzopoulos. Recursive Estimation of Nonrigid Shape and Motion. *Proc. IEEE Motion Workshop, Princeton, NJ*, pp. 306–311, October 1991.

[42] D. Metaxas and D. Terzopoulos. Shape and Nonrigid Motion Estimation Through Physics-Based Synthesis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1992. in press.

[43] 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.

[44] D. Metaxas and D. Terzopoulos. Kalman Filter Based Estimation of Shape and Nonrigid Motion. *Proc. Canadian Conference on Electrical and Computer Engineering (CCECE-92), Toronto, Ontario*, September 1992.

[45] 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.

[46] 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, in press.

[47] T. McInerney. Finite Element Techniques for Fitting Deformable Models to 3D Data. *MSc Thesis, Dept. of Computer Science, University of Toronto, Toronto, CA*, 1992.

[48] G. Miller. The Motion Dynamics of Snakes and Worms. *Computer Graphics (Proc. SIGGRAPH)*, 20(4):169–178, 1988.

[49] A. Pentland. Perceptual Organization and the Representation of Natural From. *Artificial Intelligence*, 28:293–331, 1986.

[50] A. Pentland. Automatic Extraction of Deformable Part Models. *International Journal of Computer Vision*, pp. 107–126, 1990.

[51] A. Pentland and B. Horowitz. Recovery of Non-rigid Motion and Structure. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):730–742, 1991.

[52] A. Pentland and J. Williams. Good Vibrations: Modal Dynamics for Graphics and Animation. *Computer Graphics (Proc. SIGGRAPH)*, 23(3):215–222, 1989.

[53] 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.

[54] J. Platt and A. Barr. Constraint Methods for Flexible Objects. *Computer Graphics (Proc. SIGGRAPH)*, 22(4):279–288, 1989.

[55] T. Poggio, V. Torre, and C. Koch. Computational Vision and Regularization Theory. *Nature*, 317(6035):314–319, 1985.

[56] M. Rioux and L. Cournoyer. The NRCC Three-Dimensional Image Data Files. Technical Report CNRC No 29077, National Research Council of Canada, 1988.

[57] S. Sclarof and A. Pentland. Generalized Implicit Functions for Computer Graphics. *Computer Graphics (Proc. SIGGRAPH)*, 25(4):247–250, 1991.

[58] T. W. Sederberg and S. R. Parry. Free-form Deformation of Solid Geometric Primitives. *Computer Graphics (Proc. SIGGRAPH)*, 20(4):151–160, 1986.

[59] A.A. Shabana. *Dynamics of Multibody Systems.* Wiley, New York, 1989.

[60] K. Shoemake. Animating Rotations with Quaternion Curves. *Computer Graphics (Proc. SIGGRAPH)*, 19(3):245–254, 1985.

[61] F. Solina and R. Bajcsy. Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(2):131–146, 1990.

[62] R. Szeliski. *Bayesian Modeling of Uncertainty in Low-Level Vision.* Kluwer Academic Publishers, Boston, MA, 1989.

[63] 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.

[64] D. Terzopoulos. Regularization of Inverse Visual Problems Involving Discontinuities. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(4):413–424, 1986.

[65] D. Terzopoulos. The Computation of Visible-Surface Representations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(4):417–438, 1988.

[66] D. Terzopoulos and K. Fleischer. Deformable Models. *The Visual Computer*, 4(6):306–331, 1988.

[67] D. Terzopoulos and D. Metaxas. Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991. See also Proc. Third International Conference on Computer Vision (ICCV'90), Osaka, Japan, Dec. 1990, pp. 606-615.

[68] D. Terzopoulos and A. Witkin. Physically Based Models with Rigid and Deformable Components. *IEEE Computer Graphics and Applications*, 8(6):41–51, 1988.

[69] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-Seeking Models and 3D Object Reconstruction. *International Journal of Computer Vision*, 1(3):211–221, 1987.

[70] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on Deformable Models: Recovering 3D Shape and Nonrigid motion. *Artificial Intelligence*, 36(1):91–123, 1988.

[71] A. Witkin, K. Fleischer, and A. Barr. Energy Constraints on Parameterized Models. *Computer Graphics (Proc. SIGGRAPH)*, 21(4):225–232, 1987.

[72] A. Witkin and W. Welch. Fast Animation and Control of Nonrigid Structures. *Computer Graphics (Proc. SIGGRAPH)*, 24(4):243–252, 1990.

[73] J. Wittenberg. *Dynamics of Systems of Rigid Bodies.* Tubner, Stuttgart, 1977.

[74] O. Zienkiewicz. *The Finite Element Method.* McGraw-Hill, 1977.