March 1993

Fast Dynamic Point-to-Point Constraint Algorithm for Deformable Bodies

Dimitris Metaxas
University of Pennsylvania

Fast Dynamic Point-to-Point Constraint Algorithm for Deformable Bodies

Abstract

This paper develops a general approach for the efficient modeling of dynamic point-to-point constraints in deformable multibody objects for the purposes of computer animation and motion estimation. Based on a stabilized Lagrange multiplier technique we devise an algorithm for the efficient computation of constraint forces necessary for the modeling of hard point-to-point constraints. Through our algorithm we compute the constraint forces by solving a usually smaller linear system of the order of the number of constraints in the deformable multibody object. We construct multi-body deformable objects from a new family of physics-based modeling primitives that we have developed. These primitives can undergo free motions as well as parameterized and free-form deformations. We demonstrate the performance of our algorithm in a serious of computer vision and computer graphic applications.

Fast Dynamic Point-to-Point Constraint Algorithm for Deformable Bodies

Dimitri Metaxas[1]
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Abstract

This paper develops a general approach for the efficient modeling of dynamic point-to-point constraints in deformable multibody objects for the purposes of computer animation and motion estimation. Based on a stabilized Lagrange multiplier technique we devise an algorithm for the efficient computation of constraint forces necessary for the modeling of hard point-to-point constraints. Through our algorithm we compute the constraint forces by solving a usually small linear system of the order of the number of constraints in the deformable multibody object. We construct multibody deformable objects from a new family of physics-based modeling primitives that we have developed. These primitives can undergo free motions as well as parameterized and free-form deformations. We demonstrate the performance of our algorithm in a series of computer vision and computer graphics applications.

1 Introduction

This paper develops an an efficient approach to approximating the dynamic behavior of nonrigid multibody objects for simulation, visualization, and estimation. We propose an approach for creating dynamic deformable models with intuitive physical behaviors by combining parameterized geometric primitives (e.g., superquadrics), parameterized geometric deformations (e.g., tapering, bending, shearing, twisting), and local free-form deformations (e.g., finite element shape functions).

Our approach makes use of some of the features of Shabana's formulation of multibody dynamics [5]. Applying Lagrangian dynamics and the finite element method, we convert the translational, rotational, and geometric degrees of freedom of our models into generalized coordinates of equations of motion governing the model's response to applied forces. The method is general and does not depend on the particular choice of geometric primitives and deformations, as long as the Jacobian of the geometric functions is computable.

[1]Assistant Professor.

Using the above primitives as building blocks we develop a constraint algorithm, based on Baumgarte's stabilization method [1], that allows point-to-point constraints among these dynamic models. Our algorithm computes the unknown constraint forces by solving a linear system of equations whose size is proportional to the number of constraints present in the articulated object. This allows the efficient construction and dynamic simulation of articulated objects with rigid or flexible parts. The constraint algorithm computes generalized constraint forces and adds them to the generalized external forces in the Lagrange equations of motion of the multibody system. Finally, we integrate these equations using standard numerical techniques.

We demonstrate the efficiency of our techniques for the synthesis and visualization of shape and nonrigid motion, as well as for the estimation of elastically deformable models from visual data. For added efficiency, we lump masses to obtain diagonal mass matrices and we employ mass-proportional damping.

In the following sections we first give an overview of our dynamic formulation based on our new modeling primitives and then we describe our dynamic point-to-point constraint algorithm.

2 Parameterized Geometric Primitives with Global and Local Deformations

We consider surface models whose intrinsic (material) coordinates are $\mathbf{u} = (u, v)$, defined on a bounded domain Ω. The positions of points on the model relative to an inertial frame of reference Φ in space are given by a vector-valued, time varying function of u:

$$\mathbf{x}(\mathbf{u}, t) = (x_1(\mathbf{u}, t), x_2(\mathbf{u}, t), x_3(\mathbf{u}, t))^T, \tag{1}$$

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

$$\mathbf{x} = \mathbf{c} + \mathbf{R}\mathbf{p}, \tag{2}$$

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

$$\mathbf{p} = \mathbf{s} + \mathbf{d}. \tag{3}$$

Figure 1 illustrates the model geometry. We define the reference shape as

$$\mathbf{s} = \mathbf{T}(\mathbf{e};\, b_1, b_2, \ldots) \tag{4}$$

where $\mathbf{T}$ is a global deformation, a function dependent on the parameters b_i. This is applied to a parametric function of material coordinates

$$\mathbf{e} = \mathbf{e}(\mathbf{u};\, a_1, a_2, \ldots) \tag{5}$$

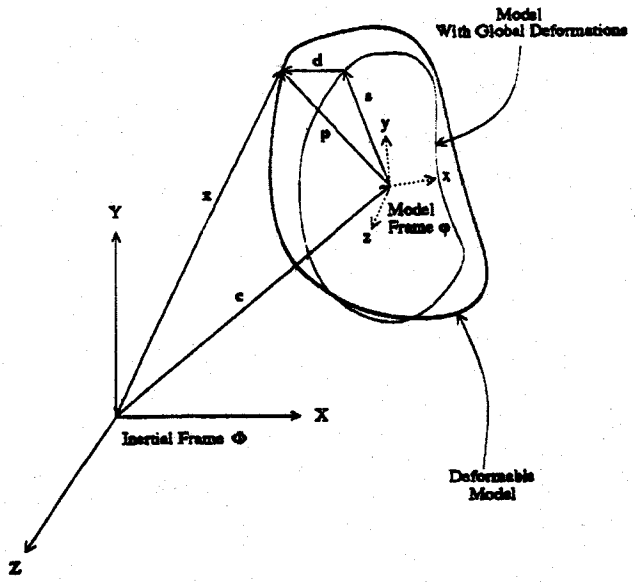

Figure 1: Geometry of deformable model.

which defines a geometric primitive whose parameters are a_i. [1] We collect both sets of global parameters into the vector of global generalized coordinates

$$\mathbf{q}_s = (a_1, a_2, \ldots, b_1, b_2, \ldots)^T. \tag{6}$$

In [4] we give examples of the parametric equations of superquadric ellipsoids with tapering, bending, shearing, and twisting deformations.

We express the displacement $\mathbf{d}$ as a linear combination of basis functions $\mathbf{b}_i(\mathrm{u})$

$$\mathbf{d} = \sum_i \mathbf{S}_i^T \mathbf{q}_i, \tag{7}$$

where the diagonal matrix $\mathbf{S}_i$ is formed from the basis functions while $\mathbf{q}_i$ are time dependent generalized coordinates. The basis functions can be local or global; however, finite element basis functions [2] are the natural choice for representing local deformations. We associate a displacement vector $\mathbf{q}_i$ with each node i of the model. Collecting the generalized coordinates into a vector $\mathbf{q}_d = (\ldots, \mathbf{q}_i^T, \ldots)^T$, we write

$$\mathbf{d} = \mathbf{S}\mathbf{q}_d, \tag{8}$$

where $\mathbf{S}$ is the shape matrix whose entries are the finite element basis functions. In [4] we describe the finite elements that we employ to create deformable superquadrics.

[1] Both $\mathbf{T}$ and $\mathbf{e}$ are assumed to be differentiable. Note that $\mathbf{T}$ will usually be a composite deformation which can be a sequence of simpler deformations.

3 Kinematics and Dynamics

To convert parameterized geometric models into physical models that respond dynamically to forces, we first consider the kinematics implied by the geometry, then we introduce mass, damping, and elasticity into the model to derive its mechanics.

The velocity of points on the model is given by,

$$\dot{\mathbf{x}} = \dot{\mathbf{c}} + \dot{\mathbf{R}}\mathbf{p} + \mathbf{R}\dot{\mathbf{p}} = \dot{\mathbf{c}} + \mathbf{B}\dot{\boldsymbol{\theta}} + \mathbf{R}\dot{\mathbf{s}} + \mathbf{R}\mathbf{S}\dot{\mathbf{q}}_d, \tag{9}$$

where $\boldsymbol{\theta} = (\ldots, \theta_i, \ldots)^T$ is the vector of rotational coordinates and $\mathbf{B} = [\ldots \partial(\mathbf{R}\mathbf{p})/\partial\theta_i \ldots]$. Furthermore, $\dot{\mathbf{s}} = [\partial \mathbf{s}/\partial \mathbf{q}_s]\dot{\mathbf{q}}_s = \mathbf{J}\dot{\mathbf{q}}_s$, where $\mathbf{J}$ is the Jacobian of $\mathbf{s}$ with respect to the global parameter vector $\mathbf{q}_s$.

We can therefore write the model kinematics compactly as

$$\mathbf{x} = \mathbf{c} + \mathbf{R}(\mathbf{s} + \mathbf{d}) = \mathbf{h}(\mathbf{q}), \tag{10}$$

and

$$\dot{\mathbf{x}} = [\mathbf{I}\ \mathbf{B}\ \mathbf{R}\mathbf{J}\ \mathbf{R}\mathbf{S}]\dot{\mathbf{q}} = \mathbf{L}\dot{\mathbf{q}}, \tag{11}$$

where $\mathbf{q} = (\mathbf{q}_c^T, \mathbf{q}_\theta^T, \mathbf{q}_s^T, \mathbf{q}_d^T)^T$, with $\mathbf{q}_c = \mathbf{c}$ and $\mathbf{q}_\theta = \boldsymbol{\theta}$ is the total generalized coordinate vector for the model.

Next, we introduce a mass distribution over the model's material domain $\mathbf{u} \in \Omega$ and assume that the material is subject to velocity-dependent damping while in motion. We also assume that the material is subject to elastic or viscoelastic deformations. Applying Lagrangian dynamics, we obtain second-order equations of motion which take the general form

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{g}_q + \mathbf{f}_q, \tag{12}$$

where $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ are the mass, damping, and stiffness matrices, respectively, where $\mathbf{g}_q$ are generalized inertial centrifugal and Coriolis forces arising from the dynamic coupling between the local and global deformation generalized coordinates, and where $\mathbf{f}_q(\mathbf{u}, t)$ are the generalized external forces associated with the generalized coordinates of the model. In [4] we derive (12) along with expressions for the relevant matrices and force vectors.

4 Holonomic Constraints and Lagrange Multipliers

In this section we extend the equations of motion (12) to account for the motions of composite models with interconnected deformable parts which are constrained not to separate. Shabana [5] 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

$$\mathbf{C}(\mathbf{q}, t) = \mathbf{0}, \tag{13}$$

where

$$\mathbf{C} = [\mathbf{C}_1^T, \mathbf{C}_2^T, \ldots, \mathbf{C}_{\bar{k}}^T]^T \tag{14}$$

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

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{g}_q + \mathbf{f}_q + \mathbf{f}_{g_c}. \tag{15}$$

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

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{g}_q + \mathbf{f}_q - \mathbf{C}_\mathbf{q}^T\boldsymbol{\lambda}, \tag{16}$$

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

$$\mathbf{f}_{g_c} = -\mathbf{C}_\mathbf{q}^T\boldsymbol{\lambda}. \tag{17}$$

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

$$\boldsymbol{\lambda} = (\lambda_1^T, \ldots, \lambda_{\bar{n}}^T)^T \tag{18}$$

is a vector of Lagrange multipliers that must be determined.

Equation (16) comprises fewer equations than unknowns. To obtain the additional equations, we differentiate (13) twice with respect to time

$$\ddot{\mathbf{C}}(\mathbf{q}, t) = \mathbf{0}, \tag{19}$$

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

$$\boldsymbol{\gamma} = \mathbf{C}_\mathbf{q}\ddot{\mathbf{q}} = -\mathbf{C}_{tt} - (\mathbf{C}_\mathbf{q}\dot{\mathbf{q}})_\mathbf{q}\dot{\mathbf{q}} - 2\mathbf{C}_{\mathbf{q}_t}\dot{\mathbf{q}}. \tag{20}$$

Appending this equation to (16) and rearranging terms, we arrive at the augmented equations of motion

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_\mathbf{q}^T \\ \mathbf{C}_\mathbf{q} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{D}\dot{\mathbf{q}} - \mathbf{K}\mathbf{q} + \mathbf{g}_q + \mathbf{f}_q \\ \boldsymbol{\gamma} \end{bmatrix}. \tag{21}$$

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 (21) 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.

5 Stabilized Constraints

To remedy these two problems, we apply a method proposed by Baumgarte which stabilizes the constrained equations through linear feedback control [1, 6]. The method replaces the differential equation (20) 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

$$\ddot{\mathbf{C}} + 2\alpha\dot{\mathbf{C}} + \beta^2\mathbf{C} = \mathbf{0}, \tag{22}$$

where α and β are stabilization factors modify (21) as follows

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_\mathbf{q}^T \\ \mathbf{C}_\mathbf{q} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{D}\dot{\mathbf{q}} - \mathbf{K}\mathbf{q} + \mathbf{g}_q + \mathbf{f}_q \\ \boldsymbol{\gamma} - 2\alpha\dot{\mathbf{C}} - \beta^2\mathbf{C} \end{bmatrix}. \tag{23}$$

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

$$\mathbf{C}(\mathbf{q}, t) = \mathbf{C}(\mathbf{q}, 0)e^{-\alpha t} \tag{24}$$

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.

6 Fast Point-to-Point Constraint Force Computation

The general Lagrange multiplier method described above is potentially expensive for our models, since the matrix in (21) 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.

We derive the method for second-order dynamic systems (12). We will start by giving a simple example of multipart objects with constraints. This will clarify the general algorithm described later.

6.1 Multibody Object with Two Constraints

Figure 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 (12), the motion equations of the parts are

$$\begin{aligned} \ddot{\mathbf{q}}_1 &= \mathbf{M}_1^{-1}(\mathbf{g}_{q_1} + \mathbf{f}_{q_1} + \mathbf{f}_{g_{c_A}} - \mathbf{K}_1\mathbf{q}_1 - \mathbf{D}_1\dot{\mathbf{q}}_1) \\ &= \mathbf{M}_1^{-1}(\mathbf{f}_{g_{c_A}} + \mathbf{V}_1), \end{aligned} \tag{25}$$

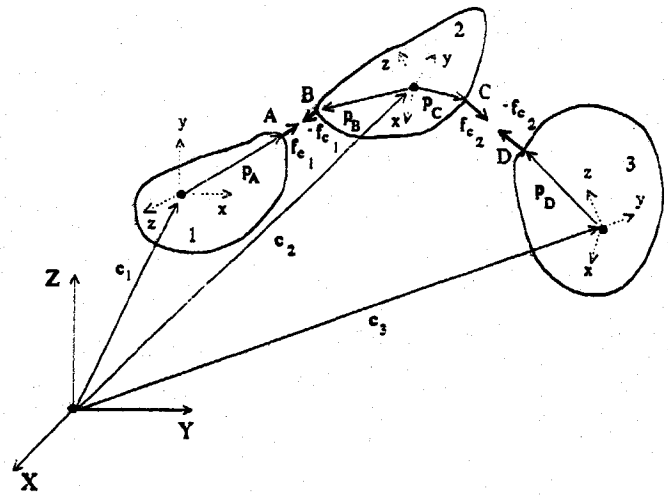

Figure 2: Two Point-to-point constraints.

$$\begin{aligned}\ddot{\mathbf{q}}_2 &= \mathbf{M}_2^{-1}(\mathbf{g}_{q_2} + \mathbf{f}_{q_2} + \mathbf{f}_{g_{c_B}} + \mathbf{f}_{g_{c_C}} - \mathbf{K}_2\mathbf{q}_2 - \mathbf{D}_2\dot{\mathbf{q}}_2) \\ &= \mathbf{M}_2^{-1}(\mathbf{f}_{g_{c_B}} + \mathbf{f}_{g_{c_C}} + \mathbf{V}_2), \end{aligned} \tag{26}$$

$$\begin{aligned}\ddot{\mathbf{q}}_3 &= \mathbf{M}_3^{-1}(\mathbf{g}_{q_3} + \mathbf{f}_{q_3} + \mathbf{f}_{g_{c_D}} - \mathbf{K}_3\mathbf{q}_3 - \mathbf{D}_3\dot{\mathbf{q}}_3) \\ &= \mathbf{M}_3^{-1}(\mathbf{f}_{g_{c_D}} + \mathbf{V}_3), \end{aligned} \tag{27}$$

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

$$\begin{aligned}\mathbf{f}_{g_{c_A}} &= \mathbf{L}_A^T\mathbf{f}_{c_1}, \quad & \mathbf{f}_{g_{c_B}} &= -\mathbf{L}_B^T\mathbf{f}_{c_1}, \\ \mathbf{f}_{g_{c_C}} &= \mathbf{L}_C^T\mathbf{f}_{c_2}, \quad & \mathbf{f}_{g_{c_D}} &= -\mathbf{L}_D^T\mathbf{f}_{c_2}, \end{aligned} \tag{28}$$

and $\mathbf{L}_A, \mathbf{L}_B, \mathbf{L}_C, \mathbf{L}_D$, are computed using (11).

From (11) and (10), the two constraint equations and their time derivatives are

$$\begin{aligned}\mathbf{C}_1 &= \mathbf{x}_A - \mathbf{x}_B = (\mathbf{c}_1 + \mathbf{R}_1\mathbf{p}_A) - (\mathbf{c}_2 + \mathbf{R}_2\mathbf{p}_B) \\ \dot{\mathbf{C}}_1 &= \mathbf{L}_A\dot{\mathbf{q}}_1 - \mathbf{L}_B\dot{\mathbf{q}}_2 \\ \ddot{\mathbf{C}}_1 &= \mathbf{L}_A\ddot{\mathbf{q}}_1 + \dot{\mathbf{L}}_A\dot{\mathbf{q}}_1 - \mathbf{L}_B\ddot{\mathbf{q}}_2 - \dot{\mathbf{L}}_B\dot{\mathbf{q}}_2, \end{aligned} \tag{29}$$

and

$$\begin{aligned}\mathbf{C}_2 &= \mathbf{x}_C - \mathbf{x}_D = (\mathbf{c}_2 + \mathbf{R}_2\mathbf{p}_C) - (\mathbf{c}_3 + \mathbf{R}_3\mathbf{p}_D) \\ \dot{\mathbf{C}}_2 &= \mathbf{L}_C\dot{\mathbf{q}}_2 - \mathbf{L}_D\dot{\mathbf{q}}_3 \\ \ddot{\mathbf{C}}_2 &= \mathbf{L}_C\ddot{\mathbf{q}}_2 + \dot{\mathbf{L}}_C\dot{\mathbf{q}}_2 - \mathbf{L}_D\ddot{\mathbf{q}}_3 - \dot{\mathbf{L}}_D\dot{\mathbf{q}}_3. \end{aligned} \tag{30}$$

Replacing these expressions into Baumagarte's equation (22) (with $\alpha = \beta$), we obtain the linear system of equations

$$\begin{aligned}(\mathbf{L}_A\mathbf{M}_1^{-1}\mathbf{L}_A^T + \mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{L}_B^T)\mathbf{f}_{c_1} - (\mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{L}_C^T)\mathbf{f}_{c_2} + \mathbf{r}_{11} &= \mathbf{N}_1\mathbf{f}_c + \mathbf{r}_1 = \mathbf{0} \\ -(\mathbf{L}_C\mathbf{M}_2^{-1}\mathbf{L}_B^T)\mathbf{f}_{c_1} + (\mathbf{L}_C\mathbf{M}_2^{-1}\mathbf{L}_C^T + \mathbf{L}_D\mathbf{M}_3^{-1}\mathbf{L}_D^T)\mathbf{f}_{c_2} + \mathbf{r}_{22} &= \mathbf{N}_2\mathbf{f}_c + \mathbf{r}_2 = \mathbf{0}, \end{aligned} \tag{31}$$

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

$$\mathbf{r}_{11} = \dot{\mathbf{L}}_A \dot{\mathbf{q}}_1 - \dot{\mathbf{L}}_B \dot{\mathbf{q}}_2 + 2a\dot{\mathbf{C}}_1 + b^2\mathbf{C}_1 + \mathbf{L}_A\mathbf{M}_1^{-1}\mathbf{V}_1 - \mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{V}_2, \tag{32}$$

$$\mathbf{r}_{22} = \dot{\mathbf{L}}_C \dot{\mathbf{q}}_2 - \dot{\mathbf{L}}_D \dot{\mathbf{q}}_3 + 2a\dot{\mathbf{C}}_2 + b^2\mathbf{C}_2 + \mathbf{L}_C\mathbf{M}_2^{-1}\mathbf{V}_2 - \mathbf{L}_D\mathbf{M}_3^{-1}\mathbf{V}_3, \tag{33}$$

and

$$N_1 = \begin{pmatrix} \mathbf{L}_A\mathbf{M}_1^{-1}\mathbf{L}_A^T + \mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{L}_B^T & -(\mathbf{L}_B\mathbf{M}_2^{-1}\mathbf{L}_C^T) \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{34}$$

$$N_2 = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ -(\mathbf{L}_C\mathbf{M}_2^{-1}\mathbf{L}_B^T) & \mathbf{L}_C\mathbf{M}_2^{-1}\mathbf{L}_C^T + \mathbf{L}_D\mathbf{M}_3^{-1}\mathbf{L}_D^T \end{pmatrix}, \tag{35}$$

$$\mathbf{r}_1 = \begin{pmatrix} \mathbf{r}_{11} \\ \mathbf{0} \end{pmatrix}, \tag{36}$$

$$\mathbf{r}_2 = \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_{22} \end{pmatrix}. \tag{37}$$

Combining (31) we arrive at the linear system

$$\mathbf{N}\mathbf{f}_c + \mathbf{r} = (\mathbf{N}_1 + \mathbf{N}_2)\mathbf{f}_c + (\mathbf{r}_1 + \mathbf{r}_2) = \mathbf{0}, \tag{38}$$

which we solve using an *LU* decomposition algorithm.

6.2 Multibody Object with 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 $\mathbf{f}_c = (\mathbf{f}_{c_1}^T, \mathbf{f}_{c_2}^T, \ldots, \mathbf{f}_{c_{\bar{k}}}^T)^T$ and express the equation for constraint forces $\mathbf{f}_{c_i}$ as in (31):

$$\mathbf{N}_i\mathbf{f}_c + \mathbf{r}_i = \mathbf{0}, \tag{39}$$

where

$$\mathbf{r}_i = (\mathbf{0}^T, \ldots, \mathbf{r}_{ii}^T, \ldots, \mathbf{0}^T)^T \tag{40}$$

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

$$\mathbf{N}\mathbf{f}_c + \mathbf{r} = \mathbf{0}, \tag{41}$$

where

$$\mathbf{N} = \sum_{i=1}^{\bar{k}} \mathbf{N}_i \tag{42}$$

and

$$\mathbf{r} = \sum_{i=1}^{\bar{k}} \mathbf{r}_i. \tag{43}$$

Based on the form of equations (39) we can devise the following algorithm to automatically compute the entries of the above matrix **N**.

6.2.1 Algorithm to Compute N

The algorithm is as follows:

- *Step 0*: Divide **N** into submatrices as follows

$$\mathbf{N} = \begin{bmatrix} \mathbf{N}_{11} & . & . & . & \mathbf{N}_{1\bar{k}} \\ . & . & & & . \\ . & & \mathbf{N}_{ij} & & . \\ . & & & . & . \\ \mathbf{N}_{\bar{k}1} & . & . & . & \mathbf{N}_{\bar{k}\bar{k}} \end{bmatrix}, \tag{44}$$

  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

$$\mathbf{N}_{ll} = \mathbf{L}_{i_A}\mathbf{M}_i^{-1}\mathbf{L}_{i_A}^T + \mathbf{L}_{j_B}\mathbf{M}_j^{-1}\mathbf{L}_{j_B}^T. \tag{45}$$

  - *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*:

$$\mathbf{N}_{lm} = \mathbf{L}_{i_A}\mathbf{M}_i^{-1}\mathbf{L}_{i_J}^T. \tag{46}$$

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

$$\mathbf{N}_{lm} = -\mathbf{L}_{i_A}\mathbf{M}_i^{-1}\mathbf{L}_{i_J}^T. \tag{47}$$

  - *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*:

$$\mathbf{N}_{lr} = -\mathbf{L}_{j_B}\mathbf{M}_j^{-1}\mathbf{L}_{j_J}^T. \tag{48}$$

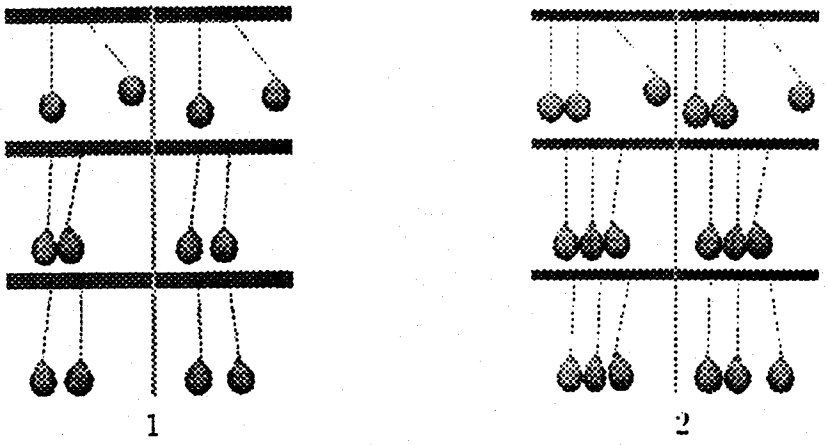

Figure 3: Balloon pendulums

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

$$\mathbf{N}_{lr} = \mathbf{L}_{j_B}\mathbf{M}_j^{-1}\mathbf{L}_{j_J}^T. \tag{49}$$

The pattern of nonzero entries in N depends on the connectivity of the parts.

In applying the fast point-to-point constraint algorithm of Section 6, at each time step we assemble N and r and compute the constraint forces $\mathbf{f}_c$ by solving (41) 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.

7 Animation and Estimation Examples

We have created several physics-based animations and shape estimation examples involving models made from deformable superquadric primitives. The primitives interact with one another and with their simulated physical environments through constraints, collisions, gravity, and friction.

Figure 3-1 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. Figure 3-1(a) shows the initial configuration with the left balloon pulled to the side. Gravity is activated in Figure 3-1(b). The balloons deform under their own weight. The left balloon swings to the right, colliding inelastically with its neighbor in Figure 3-1(c), thereby transferring some of its kinetic energy. In Figures 3-1(d–f) the balloons collide repeatedly until all the kinetic energy is dissipated. The collisions are implemented using reaction constraints between multiple deformable bodies [3]. Figure 3-2 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.

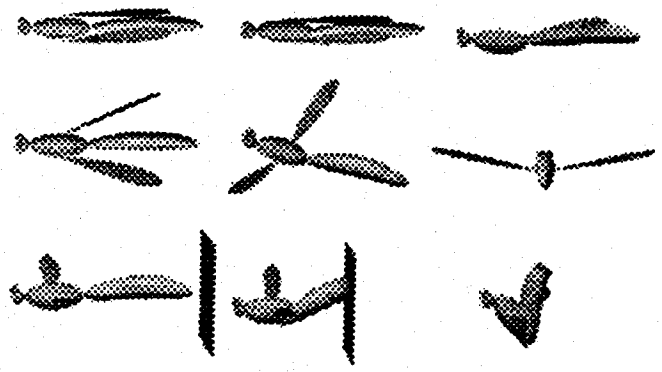

Figure 4: Self-assembly, articulation, and swatting of a dragonfly.

Figure 4 shows the automatic construction of a minimalist dragonfly from its constituent deformable superquadric parts. Figure 4(a) shows the disjoint parts in their initial configurations. [2] After activating our constraint algorithm, the model self-assembles to form the articulated dragonfly shown in Figures 4(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 Figures 4(d-f). An impenetrable plane appears in Figure 4(g) to swat the dragonfly in the rear (Figure 4(h)). The body parts deform in response to the blow, but the point-to-point constraints continue to hold them together. The mangled dragonfly is shown in Figure 4(i).

In Figure 5 we fit 5 deformable superquadrics to data collected from the raising and flexing motion of the two arms of a human subject (approximately 120 frames). The human motion 3D data were collected using WATSMART, a non-contact, three-dimensional motion digitizing and analysis system [3]. Figure 5(a) shows a view of the range data and the initial models. Figure 5(b) shows an intermediate step of the model estimation process driven by data forces from the first frame of the motion sequence, while Figures 5(c) and (d) show the models estimated from the initial data. Figures 5(e) and (f) show intermediate frames of the models tracking the nonrigid motion of the arms, while Figures 5(g) and (h) show two views of the final position of the estimated models (see [3, 4] for more details).

[2] 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 $\nu = 1.0$.

[3] We used an Euler step equal to 4.0×10^{-5}, a zero mass matrix M and a unit damping matrix D. In this way the deformable models have no inertia and therefore stop moving as soon as they fit the data and no external forces are exerted on them.

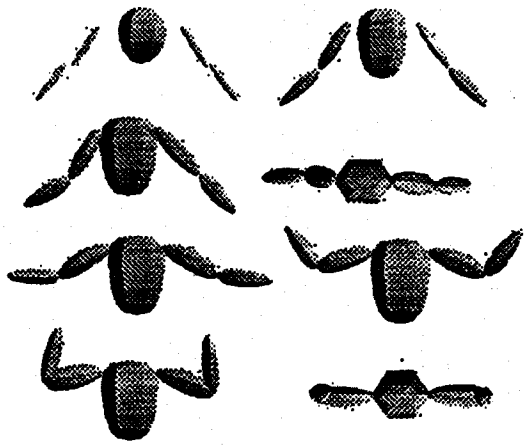

Figure 5: Tracking of raising and flexing human arm motion.

8 Conclusion

In this paper we first presented our new class of dynamic solid models that have direct ties to standard, parameterized geometric primitives. Unlike their geometric predecessors, however, our models will deform in physically intuitive ways under the control of Lagrange equations of motion. Second, we described a constraint algorithm suitable for our deformable multibody models, which is based on Baumgarte's stabilization method. For point-to-point constraints, the algorithm computes the unknown constraint forces by solving a linear system of equations whose size is of the order of the number of constraints. This permits the efficient construction and dynamic simulation of articulated objects with rigid or flexible parts.

We are currently exploring the addition of other kinds of constraints to our framework, as well as more accurate algorithms for dealing with collisions of multibody objects.

References

[1] Baumgarte, J., (1972) "Stabilization of constraints and integrals of motion in dynamical systems," *Comp. Meth. in Appl. Mech. and Eng.*, 1, 1–16.

[2] Kardestuncer, H., (ed.), (1987) *Finite element handbook*, McGraw–Hill, New York.

[3] Metaxas, D., and Terzopoulos, D., (1992) "Shape and nonrigid motion estimation," *IEEE Patt. Anal. and Mach. Intell.*, to appear.

[4] D. Metaxas, Physics-Based Modeling of Nonrigid Objects for Vision and Graphics, Ph.D. Thesis, Department of Computer Scinece, University of Toronto, 1992.

[5] Shabana, A., (1989) *Dynamics of multibody systems*, Wiley, New York.

[6] Wittenburg, J., (1977) *Dynamics of systems of rigid bodies*, Tubner, Stuttgart.