

Summary of  
*A Survey of Deformable Modelling in  
Computer Graphics*

by Sarah F.F. Gibson and Brian Mirtich

Summary by Christopher Sewell

This paper is a pretty good overview of the major methods used in deformable modelling, including non-physical (geometric), mass-spring, and continuum finite element models.

## Non-Physical Models

One way to get deformations is to use geometric techniques. Rather than use any physically-based equations, you just directly impose the deformations according to your intuition. Several methods have been devised to help with this. Spline methods, such as Bezier curves, B-splines, and non-uniform rational B-splines (NURBS) allow you to define and deform curves by moving control points. The curve itself is determined by the control points – interpolating through some, tangent to the segments between some, etc., depending on the method.

Another technique called free-form deformations define a grid in which the object of interest is located, and the object is deformed according to transformations of the coordinate system. For example, multiplying the points of the object by a transformation matrix can rotate, translate, and scale the object. Other mappings can bend, taper, and otherwise warp the object.

Geometric techniques advantages

- very low computational demands
- direct control

Geometric techniques disadvantages

- accuracy limited by the skill of the designer
- many specific deformations can be difficult to express in terms of spline control points or free-form deformation mappings
- the process isn't really automated, making real-time deformations in response to unpredictable forces difficult.

## Mass-Spring Models

In a mass-spring system, you just discretize the object into a collection of  $N$  nodes interconnected with springs. At each node, Newton's Second Law holds:

$$m_i \ddot{x}_i = -\gamma_i \dot{x}_i + \sum_j g_{ij} + f_i \quad (1)$$

where  $m_i$  is the mass,  $\gamma_i$  the damping coefficient, and  $x_i$  the position of node  $i$ . The term  $\sum_j g_{ij}$  is the force exerted on node  $i$  by the spring between it and node  $j$ . For linear Hookean springs, this term can be written as  $\sum_j k_{ij} x_j$ , where  $k_{ij}$  is the stiffness coefficient of the spring between nodes  $i$  and  $j$ .

For a single node,  $x$  is a  $3 \times 1$  vector ( $x$ ,  $y$ , and  $z$  components). The above equation for a single node  $i$  can be aggregated into one system of equations for all  $N$  nodes, where the unknown is a  $3N \times 1$  vector  $x$ :

$$M\ddot{x} + C\dot{x} + Kx = f \quad (2)$$

where  $M$  and  $C$  are the  $3N \times 3N$  diagonal mass and damping matrices ( $M_{ii} = m_i$  and  $C_{ii} = \gamma_i$ ). The  $3N \times 3N$  banded matrix  $K$  gives the spring stiffness coefficients between each pair of nodes ( $K_{ij} = k_{ij}$ ), which is zero if there is no spring between them.

Rewriting this as a system of first-order differential equations,

$$\dot{v} = M^{-1}(-Cv - Kx + f) \quad (3)$$

$$\dot{x} = v \quad (4)$$

The inverse of  $M$  can be precomputed (and is trivial since it is diagonal). To run the simulation, we just take small timesteps, calculating  $\dot{v}$  and  $\dot{x}$  using the values of  $x$  and  $v$  from the previous timestep, and updating them,  $x_{t+1} = x_t + \dot{x}\Delta t$  and  $v_{t+1} = v_t + \dot{v}\Delta t$ . To model cutting, you can remove links, setting  $K_{ij} = 0$  between nodes where the cut takes place.

Some interesting notes on some applications of mass springs:

- Terzopoulos and Waters modelled faces using a three-layer mass spring mesh: a dermal layer, a fatty tissue layer, and a muscle layer. The muscle is attached to rigid bone, and the interface between the fatty tissue and the muscle is the fascia.
- So how do you decide what values to use for your spring constants? Koch derived the values from tissue densities on CT images.
- Terzopoulos modelled a melting system by associating a temperature with each node of hexahedral lattice. As temperature increases, spring stiffnesses increase until the melting point is reached, at which point the stiffness is set to zero. With the bond severed, the node becomes an independent glob simulated using a discrete fluid model.

Mass-spring advantages

- easy to construct
- low computational demands
- parallelize well
- more accurate than geometric techniques

Mass-spring disadvantages

- not very accurate physical model
- difficult to tune spring constants
- constraints such as incompressible volumes and thin surfaces are difficult to express
- difficult to model rigid objects, since large spring constants make the system of equations unstable, requiring small timesteps

## Finite Element Methods : Impact

First to review and summarize what we learned earlier about FEM, here is pseudocode for Impact, a simple FEM simulator, using two-node rod elements, pure linear elastic material, and explicit time stepping.

Input: list of nodes with their x,y,z coordinates; list of elements with their nodes, material type, and cross-sectional area; list of material properties (density, Young's Modulus); list of constraints (boundary constraint: fixed nodes); list of loads (location, force vector)

Initialization:

```
for each element
  mass = density*cross_sectional_area*length
  add half of mass to each node
  inertia =  $\begin{bmatrix} MR^2/2 & 0 & 0 \\ 0 & ML^2/12 & 0 \\ 0 & 0 & ML^2/12 \end{bmatrix}$ 
  add half of inertia to each node
```

Simulation loop:

```
for each timestep
  for each element
    strain =  $\frac{\Delta l}{l_0}$ 
    stress = Y*strain (since Y = stress/strain)
    force = stress*cross_sectional_area
    add [ force 0 0 ] to left node
    add [ -force 0 0 ] to right node
    if collision add repulsive forces to nodes
  for each node
    add external force from any load at the node
    accelerationin = force/mass
    accelerationrot = inertia-1.(forcerot - (velocityrot × (inertia · velocityrot)))
    velocity += acceleration · timestep
    displacement += velocity · timestep
```

Output: At each timestep, the displacement (from original position) of each node and the strain and stress of each element.

## Continuum Models and Finite Element Methods

The basic idea for a static model is that the system will reach equilibrium when its potential energy is at its minimum. So we need to come up with an expression for the potential energy and then set its derivative equal to zero and solve for the unknowns – the displacements of the nodes. The system consists of the deformable object and the loads that act on it. Thus, the total potential energy of the system is the strain energy of the deformed object minus the potential energy lost by the loads as it does the work of deforming the object. If  $\Lambda$  is the strain energy of the object,  $W$  is the work done by the loads, and  $\Pi$

is the total potential energy of the system,

$$\Pi = \Lambda - W \quad (5)$$

So we need to express  $\Lambda$  and  $W$  in terms of the unknowns, the displacements. We will write the equilibrium equation for one element, and then aggregate these equations for all elements into one big system.

Since I really like examples, we will consider along the way a simple element – a triangle with three nodes at  $(x_1, y_1, z_1)$ ,  $(x_2, y_2, z_2)$ , and  $(x_3, y_3, z_3)$  – to see what some of the matrices and equations look like for a real example.

First, we need an interpolation function that gives us the value of any function at any point  $(x,y,z)$  inside the element in terms of the values of the function at the nodes of the element. If  $\Phi$  is the function, we need parameters  $a_1$ ,  $a_2$ , and  $a_3$ , which are functions of the known node coordinates  $(x_1, y_1, z_1)$ ,  $(x_2, y_2, z_2)$ , ... and of the known values of the function  $\Phi_1, \Phi_2, \Phi_3$  at the nodes. For our 2-D triangle example, we need only worry about the  $(x,y)$  coordinates:

$$\Phi = a_1 + a_2x + a_3y \quad (6)$$

For each of the three nodes, we know  $x$ ,  $y$ , and  $\Phi(x, y)$ , so we have a system of three equations in three unknowns.

$$\Phi_1 = a_1 + a_2x_1 + a_3y_1 \quad (7)$$

$$\Phi_2 = a_1 + a_2x_2 + a_3y_2 \quad (8)$$

$$\Phi_3 = a_1 + a_2x_3 + a_3y_3 \quad (9)$$

We can solve for  $a_1$ ,  $a_2$ , and  $a_3$ , in terms of the known node coordinates and  $\Phi$  values at the nodes.

$$a_1 = [(x_2y_3 - x_3y_2) \Phi_1 + (x_3y_1 - x_1y_3) \Phi_2 + (x_1y_2 - x_2y_1) \Phi_3] / (2A) \quad (10)$$

$$a_2 = [(x_2 - x_3) \Phi_1 + (x_3 - x_1) \Phi_2 + (x_1 - x_2) \Phi_3] / (2A) \quad (11)$$

$$a_3 = [(x_3 - x_1) \Phi_1 + (x_1 - x_3) \Phi_2 + (x_2 - x_1) \Phi_3] / (2A) \quad (12)$$

where  $A$  is the area of the triangle (easily calculated as one-half the length times the base from the node coordinates).

Substituting these into (6), we have an expression for  $\Phi$  containing the variables  $x$  and  $y$  and the constants  $x_1, y_1, x_2, y_2, x_3, y_3, \Phi_1, \Phi_2$ , and  $\Phi_3$ . This expression can be rearranged into the form

$$\Phi = h_1\Phi_1 + h_2\Phi_2 + h_3\Phi_3 \quad (13)$$

where

$$h_1 = [(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y] / (2A) \quad (14)$$

$$h_2 = [(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y] / (2A) \quad (15)$$

$$h_3 = [(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y] / (2A) \quad (16)$$

One important  $\Phi$  function is the displacement function. We want to express the displacement  $u = [u \ v \ w]^T$  of a point (x,y,z) inside an element, in terms of the interpolation parameters and the displacements at the nodes. For our three-node triangle,

$$u = h_1u_1 + h_2u_2 + h_3u_3 \quad (17)$$

$$v = h_1v_1 + h_2v_2 + h_3v_3 \quad (18)$$

$$w = h_1w_1 + h_2w_2 + h_3w_3 \quad (19)$$

These equations can be written in matrix form  $u = HU$ , where H is  $3 \times 3N$  and U is  $3N \times 1$ , giving the  $3 \times 1$  displacement vector u. For our triangle,

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 & 0 & 0 \\ 0 & h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 & 0 \\ 0 & 0 & h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \\ u_3 \\ v_3 \\ w_3 \end{bmatrix} \quad (20)$$

Strain is a  $6 \times 1$  vector that relates changes in displacements to changes position. It is defined as

$$\epsilon = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial w}{\partial z} \\ \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{bmatrix} \quad (21)$$

So, for example, for our triangle, substituting the values of the h parameters from (14) to (16) into the expression for u in (17) and differentiating with respect to x,

$$\epsilon_1 = \frac{(y_2 - y_3)u_1 + (y_3 - y_1)u_2 + (y_1 - y_2)u_3}{2A} \quad (22)$$

The strain can be expressed as a matrix equation  $\epsilon = BU$ , where B is a  $6 \times 3N$  matrix. The (first, second, third) row of B is obtained by differentiating

(17) with respect to  $(x, y, z)$ , and then with respect to each of the  $3N$  node displacements. The fourth through sixth rows are the sums of two such differentiations, according to the definition of  $\epsilon$  in (21). For our triangle example,

$$\begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial w}{\partial z} \\ \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \\ \frac{\partial x}{\partial y} + \frac{\partial v}{\partial x} \end{bmatrix} = \begin{bmatrix} \frac{y_2 - y_3}{2A} & 0 & 0 & \frac{y_3 - y_1}{2A} & 0 & 0 & \frac{y_1 - y_2}{2A} & 0 & 0 \\ 0 & \frac{x_3 - x_2}{2A} & 0 & 0 & \frac{x_1 - x_3}{2A} & 0 & 0 & \frac{x_2 - x_1}{2A} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{x_3 - x_2}{2A} & 0 & 0 & \frac{x_1 - x_3}{2A} & 0 & 0 & \frac{x_2 - x_1}{2A} \\ 0 & 0 & \frac{y_2 - y_3}{2A} & 0 & 0 & \frac{y_3 - y_1}{2A} & 0 & 0 & \frac{y_1 - y_2}{2A} \\ \frac{x_3 - x_2}{2A} & \frac{y_2 - y_3}{2A} & 0 & \frac{x_1 - x_3}{2A} & \frac{y_3 - y_1}{2A} & 0 & \frac{x_2 - x_1}{2A} & \frac{y_1 - y_2}{2A} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \\ u_3 \\ v_3 \\ w_3 \end{bmatrix} \quad (23)$$

The strain energy over the body can be expressed as the product of the stress and strain vectors, integrated over the volume:

$$\Lambda = \frac{1}{2} \int_V \sigma^T \epsilon dV \quad (24)$$

Dimensional analysis sanity check: Stress is force per unit area, and strain is length per length (unitless). So the stress over a volume  $\frac{\text{kg}\cdot\text{m}}{\text{m}^2\cdot\text{s}^2} \cdot \text{m}^3$  gives  $\frac{\text{kg}\cdot\text{m}^2}{\text{s}^2}$ , a force times a distance, which is an energy.

Since stress and strain are related (for linear elasticity, by Young's modulus  $Y = \text{stress}/\text{strain}$ ), the stress can be expressed in terms of the strain,  $\sigma^T = \epsilon^T D$ , where  $D$  is the linear matrix that relates the stress and strain components. For linear elasticity, I think  $D$  can just be a diagonal matrix with Young's modulus along the diagonal. Thus we can rewrite (24) as

$$\Lambda = \frac{1}{2} \int_V \epsilon^T D \epsilon dV \quad (25)$$

Since  $\epsilon = BU$ ,

$$\Lambda = \frac{1}{2} \int_V U^T B^T D B U dV = \frac{1}{2} U^T \left( \int_V B^T D B dV \right) U \quad (26)$$

We've seen all of these matrices –  $B$ ,  $U$ , and  $D$ . The unknowns are the node displacements in  $U$ ; the known constants are in terms of node coordinates (known at each iteration) in  $B$  and material properties (such as Young's Modulus) in  $D$ .

The work done by the load can be expressed as the displacements (a distance) dotted with the applied forces, integrated over the volume of the object:

$$W = \int_V u \cdot f dV \quad (27)$$

These forces can include body forces over the whole volume (such as gravity),  $f_b$ ; surface forces (such as pressure or drag),  $f_s$ ; and concentrated loads at nodes  $i$ ,  $p_i$ .

$$W = \int_V u \cdot f_b dV + \int_\Gamma u \cdot f_s dS + \sum_i u_i \cdot p_i \quad (28)$$

Since  $u = HU$  (and  $\sum_i u_i \cdot p_i = U^T P$  for concentrated loads since no interpolation is needed exactly at the nodes),

$$W = \int_V U^T H^T \cdot f_b dV + \int_\Gamma U^T H^T \cdot f_s dS + U^T P \quad (29)$$

$$W = U^T \left( \int_V H^T \cdot f_b dV + \int_\Gamma H^T \cdot f_s dS + P \right) \quad (30)$$

The integrals evaluate to  $3N \times 1$  vectors  $F_b$  and  $F_s$ , since  $H^T$  is  $3N \times 3$  and  $f_b$  and  $f_s$  are  $3 \times 1$ . The integrals are approximated using numerical integration techniques (a topic for another day). If the forces vary across the volume or surface (i.e., they are a function of the coordinates), then these integrals must be re-evaluated at every timestep, greatly slowing the simulation.

$$W = U^T (F_b + F_s + P) \quad (31)$$

Substituting (26) and (31) into (5),

$$\Pi = \frac{1}{2} U^T \left( \int_V B^T D B dV \right) U + U^T (F_b + F_s + P) \quad (32)$$

Forgetting these are matrices for a second, we can differentiate with respect to the unknown  $U$ , getting

$$\frac{\partial \Pi}{\partial U} = \left( \int_V B^T D B dV \right) U + F_b + F_s + P \quad (33)$$

Since  $B^T$  is  $3N \times 6$ ,  $D$  is  $6 \times 6$ , and  $B$  is  $6 \times 3N$ , the integral evaluates to a  $3N \times 3N$  matrix  $K$ . The  $3N \times 1$  vectors  $F_b$ ,  $F_s$ , and  $P$  add up to a  $3N \times 1$  vector  $-F$ , giving us a standard system of linear equations  $KU = F$ , for which we can solve for the  $3N \times 1$  vector  $U$ , containing the unknown displacements for the three coordinates of all  $N$  nodes.

Since  $B$  is a function of the node positions, which change at each timestep, the integral in (33) must be numerically evaluated over the changing volume at

each timestep if deformations are not really small, greatly slowing the simulation.

## Dynamic Deformation

If you want not just the static equilibrium, but also the dynamics of the system, you can essentially combine the mass and damping of mass-springs with the equilibrium analysis of the above continuum analysis.

$$M\ddot{U} + C\dot{U} + KU = F \quad (34)$$

where  $K$ ,  $F$ , and  $U$  are as described in the preceding section.  $M$  is the mass matrix, composed from all elements, where the mass matrix for a single element is just

$$M = \int_V \rho H^T H dV \quad (35)$$

where  $H$  is as described in the previous section.  $C$  can be computed from damping parameters, but since these are usually hard to estimate, it is often just computed as a linear combination of  $M$  and  $K$ .

So now, after doing all the work to get the system (including numerical integrations over the volume and surface to get  $K$ ), instead of having a linear system to solve as in the static case, we have a system of first-order differential equations to solve, as in the mass-springs section.

$$\dot{v} = M^{-1}(-CV - KU + F) \quad (36)$$

$$\dot{U} = V \quad (37)$$

## FEM: The Good and the Bad

FEM advantages

- more physically realistic simulation
- fewer nodes needed than mass-springs, resulting in smaller linear system

FEM disadvantages

- forces must be numerically integrated over volume or surface at each timestep, requiring a lot of computation

- topology changes or large deformations changes B, requiring recomputing the large stiffness matrix K (and M for dynamic systems) throughout the simulation

## Approximate Continuum Models

The main idea in the FEM methods explained above was to calculate the total potential energy of the system, given by (5), and minimize it by settings its derivative to zero and solving for the unknowns – the displacements. A fairly complex model was devised for representing the strain energy  $\Lambda$  for a deformable object as a function of its displacements. The methods in this section try to give simpler (or at least faster to calculate) such functions. Although not really physically based, they try to give a “reasonable” measure of just how strained an object is given its displacements. For some reason, this strain energy function is denoted as  $V$  in this section of the paper, but I think it’s essentially the same as (or an approximation of)  $\Lambda$ , or at least it’s closely related.

We’ll go out of order and start by reviewing something we’ve already studied – Terzopoulos’s discretized deformation energy formulations using fundamental forms.

They use a function  $r(s)$  that transforms the parameters into a point in 3-D Euclidean space. You parameterize your volume, surface, or curve with  $s$ ; for a volume, it has three components  $(s_1, s_2, s_3)$ , for a surface, two  $(s_1, s_2)$ , and for a curve, one  $(s_1)$ . For example, for a perfect sphere, you could use the standard  $\rho$ ,  $\phi$ , and  $\theta$  parameters

$$r(s) = r(s_1, s_2, s_3) = r(\rho, \phi, \theta) = \begin{bmatrix} \rho \sin(\phi) \cos(\theta) & \rho \sin(\phi) \sin(\theta) & \rho \cos(\phi) \end{bmatrix} \quad (38)$$

As the sphere starts deforming, there is no longer an analytical form for  $r(s)$  – or at least not a very simple one – but there is still *some* mapping of parameter points to Euclidean points, and we’re going to discretize everything eventually anyway.

The first fundamental form relates changes in parameters to change in Euclidean distance.

$$dl^2 = \sum_{i,j} G_{ij} ds_i ds_j \quad (39)$$

Two volumes have the same shape if their first fundamental forms are equivalent.

The strain energy for elastic bodies is approximated by the weighted norm of the differences between the fundamental forms of the deformed body and the fundamental forms of the natural, undeformed body  $G^0$ .

$$V(r) = \int_V \|G - G^0\|_\alpha^2 ds_1 ds_2 ds_3 \quad (40)$$

So, intuitively, if in the undeformed body, a small change in the parameters results in a small change in distance – in other words, two points close together in parameter space are close together in Euclidean space – but now in the current configuration, a small change in the parameters results in a large change in distance – in other words, two points close together in parameter space have now moved far apart – then there is a lot of deformation going on, resulting in a lot of strain energy.

The first fundament form can be calculated by

$$G_{ij}(r(a)) = \frac{\partial r}{\partial a_i} \cdot \frac{\partial r}{\partial a_j} \quad (41)$$

For example, consider a perfect cylinder. Then, as we remember from cylindrical coordinates,  $r$  is the standard

$$r(s) = [ \rho \cos(\theta) \quad \rho \sin(\theta) \quad z ] \quad (42)$$

The partials are

$$\frac{\partial r}{\partial s_1} = \frac{\partial r}{\partial \rho} = [ \cos(\theta) \quad \sin(\theta) \quad 0 ] \quad (43)$$

$$\frac{\partial r}{\partial s_2} = \frac{\partial r}{\partial \theta} = [ -\rho \sin(\theta) \quad \rho \cos(\theta) \quad 0 ] \quad (44)$$

$$\frac{\partial r}{\partial s_3} = \frac{\partial r}{\partial z} = [ 0 \quad 0 \quad 1 ] \quad (45)$$

The components of  $G$  are

$$g_{11} = \frac{\partial r}{\partial s_1} \cdot \frac{\partial r}{\partial s_1} = [ \cos(\theta) \quad \sin(\theta) \quad 0 ] \cdot [ \cos(\theta) \quad \sin(\theta) \quad 0 ] = 1 \quad (46)$$

$$g_{12} = \frac{\partial r}{\partial s_1} \cdot \frac{\partial r}{\partial s_2} = [ \cos(\theta) \quad \sin(\theta) \quad 0 ] \cdot [ -\rho \sin(\theta) \quad \rho \cos(\theta) \quad 0 ] = 0 \quad (47)$$

$$g_{13} = \frac{\partial r}{\partial s_1} \cdot \frac{\partial r}{\partial s_3} = [ \cos(\theta) \quad \sin(\theta) \quad 0 ] \cdot [ 0 \quad 0 \quad 1 ] = 0 \quad (48)$$

$$g_{21} = \frac{\partial r}{\partial s_2} \cdot \frac{\partial r}{\partial s_1} = [ -\rho \sin(\theta) \quad \rho \cos(\theta) \quad 0 ] \cdot [ \cos(\theta) \quad \sin(\theta) \quad 0 ] = 0 \quad (49)$$

$$g_{22} = \frac{\partial r}{\partial s_2} \cdot \frac{\partial r}{\partial s_2} = \begin{bmatrix} -\rho \sin(\theta) & \rho \cos(\theta) & 0 \end{bmatrix} \cdot \begin{bmatrix} -\rho \sin(\theta) & \rho \cos(\theta) & 0 \end{bmatrix} = \rho^2 \quad (50)$$

$$g_{23} = \frac{\partial r}{\partial s_2} \cdot \frac{\partial r}{\partial s_3} = \begin{bmatrix} -\rho \sin(\theta) & \rho \cos(\theta) & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} = 0 \quad (51)$$

$$g_{31} = \frac{\partial r}{\partial s_3} \cdot \frac{\partial r}{\partial s_1} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \end{bmatrix} = 0 \quad (52)$$

$$g_{32} = \frac{\partial r}{\partial s_3} \cdot \frac{\partial r}{\partial s_2} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -\rho \sin(\theta) & \rho \cos(\theta) & 0 \end{bmatrix} = 0 \quad (53)$$

$$g_{33} = \frac{\partial r}{\partial s_3} \cdot \frac{\partial r}{\partial s_3} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} = 1 \quad (54)$$

Substituting into (39), we get

$$dl^2 = da_1^2 + \rho^2 da_2^2 + da_3^2 = d\rho^2 + \rho^2 d\theta^2 + dz^2 \quad (55)$$

So if there is a change in the radius, the square of the distance is  $d\rho^2$ ; if there is a change in the angle, the square of the distance (arc length) is  $\rho^2 d\theta^2$ ; and if there is a change in the height, the square of the distance is  $dz^2$ , which all check out with basic geometry. (Recall the formula arc length =  $r\theta$ .)

For a surface, instead of one 3\*3 matrix, we require two 2\*2 matrices, the first and second fundamental forms. The first fundamental form,  $G$ , is a measure of the amount of movement of a surface in the parameter plane. The second fundamental form,  $B$ , is a measure of the change in normal vector and the change of surface position at a surface point as a function of a small movement in the parameter space. The strain energy for a surface is then calculated as

$$V(r) = \int_{\Omega} \|G - G^0\|_{\alpha}^2 + \|B - B^0\|_{\beta}^2 da_1 da_2 \quad (56)$$

For more information about this method, including how to calculate the second fundamental form, see my summary of Terzopoulos's *Elastically Deformable Models*.

The snakes method seems to be essentially a simplified 1D analog of the method just described. (Terzopoulos is also one of the authors of the snakes paper.) The strain energy for a 1D "snake" is assumed to be related the squares of its first and second derivatives, integrated over the entire length

$$V = \frac{1}{2} \int \left[ \alpha(s) \left\| \frac{dv}{ds}(s) \right\|^2 + \beta(s) \left\| \frac{d^2v}{ds^2}(s) \right\|^2 \right] ds \quad (57)$$

where  $v(s)$  is the parameterized position, just like  $r(s)$ . Large derivatives intuitively correspond to large deformations. The first derivative term corresponds to axial deformations, while the second derivative term corresponds to bending deformations. The  $\alpha(s)$  and  $\beta(s)$  are just weighting functions.

By stepping from one end of the snake to the other by  $N$  small increments of  $s$ , the integral can be discretized into a finite sum

$$V = \frac{1}{2} \sum_{i=1}^N \left[ \alpha(s_i) \left\| \frac{dv}{ds}(s_i) \right\|^2 + \beta(s_i) \left\| \frac{d^2v}{ds^2}(s_i) \right\|^2 \right] \quad (58)$$

with the first and second derivatives calculated using standard finite differences

$$\frac{dv}{ds}(s_i) = \frac{v(s_i) - v(s_{i-1})}{h} \quad (59)$$

$$\frac{d^2v}{ds^2}(s_i) = \frac{\frac{dv}{ds}(s_{i+1}) - \frac{dv}{ds}(s_i)}{h} = \frac{\frac{v(s_{i+1}) - v(s_i)}{h} - \frac{v(s_i) - v(s_{i-1})}{h}}{h} = \frac{v(s_{i+1}) - 2v(s_i) + v(s_{i-1}))}{h^2} \quad (60)$$

In the paper, the  $h$  in the denominator of the finite difference equation for a second derivative is not squared, but according to the derivation above, and according to my numerical analysis books, it should be squared; I think the missing square is a typo in the paper.

One problem with the discretized deformation energy methods is that, like the mass-spring equations, the equations (40) can become ill-conditioned for rigid objects. One approach to dealing with this has been to separate the rigid body dynamics and the deformations into two components. The rigid body dynamics of the undeformed reference body can be simulated using much simpler standard non-deformable techniques, while the deformable component need only express the displacements relative to the undeformed reference body. Thus, rather than having to include the reference shape  $G_0$  in the equations (40), you can approximate the energy with low order partial derivatives of the displacement function, like the snake equation (57), generalized to 2D or 3D. This energy is minimized as the displacements from the rigid undeformed reference body approach zero, meaning the equations actually become better conditioned the more rigid it is. Terzopoulos implemented this idea using deformable superquadric ellipsoids that, like the snake equation (57), essentially integrate the weighted sums of the squares of the partial derivatives across the surface (rather than the length). I think there's a mistake in the equation in the paper, since both derivative terms are the same, but I didn't try to understand this part in great detail. If we're really interested, we can try reading Terzopoulos's paper on this.

## Low Degree of Freedom Models

Remember the basic mass-spring equation (2)? According to model analysis, the generalized eigenvalues and eigenvectors of the mass and stiffness matrices determine the natural frequencies and modes of vibration of the structure. Similar to how a regular eigenvalue problem for one matrix  $A$  finds vectors  $\phi$  and values  $\lambda$  such that  $A\phi = \lambda\phi$ , a (second-order) generalized eigenvalue problem for two matrices  $M$  and  $K$  finds vectors  $\phi$  and values  $\lambda$  such that  $K\phi = \lambda M\phi$ . A conceptually simple (though not numerically optimal) solution method is just to rewrite this as  $(M^{-1}K)\phi = \phi\lambda$ , and find the regular eigenvalues and eigenvectors for the matrix  $M^{-1}K$ . Note that we can view  $\phi$  as either a single eigenvector and  $\lambda$  as a single scalar, or  $\phi$  as the matrix whose columns are all the eigenvectors and  $\lambda$  as the diagonal matrix of all the eigenvalues. For the rest of this discussion, we will take the latter view.

Let's see what happens if we take this matrix of generalized eigenvectors  $\phi$  of  $M^{-1}K$ , multiply both sides of the equation by  $\phi^T$ , and let  $x = \phi\tilde{x}$ . Do it now, ask questions later. This gives us

$$\phi^T M \phi \ddot{\tilde{x}} + \phi^T C \phi \dot{\tilde{x}} + \phi^T K \phi \tilde{x} = \phi^T f \quad (61)$$

Since, by definition,  $A\phi = \phi\lambda$  for matrix of eigenvectors  $\phi$  and diagonal matrix of eigenvalues  $\lambda$ ,  $\phi^{-1}A\phi = \lambda$ . Since our mass and stiffness matrices are symmetric, the matrix of eigenvectors is orthogonal, so  $\phi^{-1} = \phi^T$ . Thus,  $\phi^T M \phi = \tilde{M}$  and  $\phi^T K \phi = \tilde{K}$ , where  $\tilde{M}$  and  $\tilde{K}$  are diagonal matrices. If the damping matrix is a linear combination of the mass and stiffness matrices, as is usually the case,  $\phi^T C \phi$  also gives a diagonal matrix  $\tilde{C}$ . So we can rewrite (54) (with  $\tilde{f} = \phi^T f$ ) as

$$\tilde{M}\ddot{\tilde{x}} + \tilde{C}\dot{\tilde{x}} + \tilde{K}\tilde{x} = \tilde{f} \quad (62)$$

Don't quite see how this all works, or don't believe it will give the same result as (2)? I offer my usual proof by example.

Let's say our (usually diagonal) mass and (usually symmetric) stiffness matrices are

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad (63)$$

$$K = \begin{bmatrix} 4 & 1 & 3 \\ 1 & 5 & 1 \\ 3 & 1 & 6 \end{bmatrix} \quad (64)$$

Let the damping matrix be a linear combination (simplest case, just element-wise addition) of  $M$  and  $K$

$$C = \begin{bmatrix} 5 & 1 & 3 \\ 1 & 7 & 1 \\ 3 & 1 & 9 \end{bmatrix} \quad (65)$$

We want the generalized eigenvectors and eigenvalues of M and K, so we find the standard eigenvectors and eigenvalues of

$$M^{-1}K = \begin{bmatrix} 4 & 1 & 3 \\ 0.5 & 2.5 & 0.5 \\ 1 & 0.3333 & 2 \end{bmatrix} \quad (66)$$

By eigenvector/value numerical techniques (a topic for another day), we get

$$\phi = \begin{bmatrix} -0.92443 & -0.707107 & 0.340298 \\ -0.224673 & 0 & -0.933451 \\ -0.308143 & 0.707107 & 0.113433 \end{bmatrix} \quad (67)$$

$$\lambda = \begin{bmatrix} 5.24304 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2.25696 \end{bmatrix} \quad (68)$$

We verify that indeed

$$\phi\lambda = M^{-1}K\phi = \begin{bmatrix} -4.84682 & -0.707107 & 0.768039 \\ -1.17797 & 0 & -2.10676 \\ -1.61561 & 0.707107 & 0.256013 \end{bmatrix} \quad (69)$$

We then calculate our new diagonal “tilde” matrices

$$\tilde{M} = \phi^T M \phi = \begin{bmatrix} 1.24038 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1.89706 \end{bmatrix} \quad (70)$$

$$\tilde{K} = \phi^T K \phi = \begin{bmatrix} 6.50337 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4.2816 \end{bmatrix} \quad (71)$$

$$\tilde{C} = \phi^T C \phi = \begin{bmatrix} 7.74376 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6.17866 \end{bmatrix} \quad (72)$$

Let’s say our initial values are

$$x_0 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad v_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad f = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} \quad (73)$$

Then, at our first timestep, according to (3),

$$\dot{v} = M^{-1}(-Cv - Kx + f) = \begin{bmatrix} -22 \\ -10.5 \\ -11.3333 \end{bmatrix} \quad (74)$$

If instead we solve according to the system in (62), with initial values

$$\tilde{x}_0 = \phi^{-1}x = \begin{bmatrix} -3.70564 \\ 2.82843 \\ -1.25068 \end{bmatrix} \quad \tilde{v}_0 = \phi^{-1}v = \begin{bmatrix} -1.85282 \\ 0.707107 \\ -0.625338 \end{bmatrix} \quad \tilde{f} = \phi^T f = \begin{bmatrix} -2.91449 \\ 0 \\ -0.95944 \end{bmatrix} \quad (75)$$

we get

$$\dot{\tilde{v}} = \tilde{M}^{-1}(-\tilde{C}\tilde{v} - \tilde{K}\tilde{x} + \tilde{f}) = \begin{bmatrix} 28.6464 \\ -4.24264 \\ 4.35367 \end{bmatrix} \quad (76)$$

Converting back to the non-tilde world,

$$\dot{v} = \phi \dot{\tilde{v}} = \begin{bmatrix} -22 \\ -10.5 \\ -11.3333 \end{bmatrix} \quad (77)$$

Outstanding. But what did all this do us? Since all of our matrices are diagonal, the each equation (matrix row) is independent, and the relative importance of their contributions are proportional to their eigenvalues. (Since M and K are assumed to be positive definite, usually related to being heavily-weighted along the diagonal, all eigenvalues are positive.) So, if we're pressed for time (as we always are in real-time), we can adaptively control our accuracy by using only the most important modes! It's like SVD for mass springs! In 3D systems, you can often get away with just 6 vibration modes for rigid body motion and a few more for linear strain and quadratic strain, discarding higher order deformations.

So in our example, what happens if we remove the equation for the smallest eigenvalue, 1. We can just remove the row and column corresponding to this eigenvalue (in this case, the middle row and column, since 1 is in the middle row and column of (68)) from each matrix, since the equations are independent, giving us the smaller system

$$\begin{aligned} \dot{\tilde{v}} = \tilde{M}^{-1}(-\tilde{C}\tilde{v} - \tilde{K}\tilde{x} + \tilde{f}) = \\ \begin{bmatrix} 1.24038 & 0 \\ 0 & 1.89706 \end{bmatrix}^{-1} \left( - \begin{bmatrix} 7.74376 & 0 \\ 0 & 6.17866 \end{bmatrix} \begin{bmatrix} -1.85282 \\ -0.625338 \end{bmatrix} \right. \\ \left. - \begin{bmatrix} 6.50337 & 0 \\ 0 & 4.2816 \end{bmatrix} \begin{bmatrix} -3.70564 \\ -1.25068 \end{bmatrix} + \begin{bmatrix} -2.91449 \\ -0.95944 \end{bmatrix} \right) = \begin{bmatrix} 28.6464 \\ 4.35367 \end{bmatrix} \quad (78) \end{aligned}$$

Converting back to the non-tilde world,

$$\dot{v} = \phi \dot{\tilde{v}} = \begin{bmatrix} -0.92443 & -0.340298 \\ -0.308143 & 0.113433 \end{bmatrix} \begin{bmatrix} 28.6464 \\ 4.35367 \end{bmatrix} = \begin{bmatrix} -25 \\ -10.5 \\ -8.3333 \end{bmatrix} \quad (79)$$

Pretty close, considering we're only using  $\frac{2}{3}$  of the data.

Turning now to dynamic global deformations, Witkin and Welch define a function  $f(x, t)$  that maps locations  $x$  of a point on the undeformed object to its location on the deformed object at each time point

$$f(x, t) = R(t)p(x) \quad (80)$$

The undeformed location  $x$  and the vector  $p$  do not depend on time, but the matrix  $R$  of the generalized coordinates does. The kinetic energy is calculated as  $\frac{1}{2}mv^2$ , where the masses are given and the velocities are obtained by differentiating (80). Apparently potential energy is more difficult to calculate in general; they say that Witkin and Welch give some examples of special-case potential energy functions they developed for various deformation dynamic behaviors. Given a kinetic and a potential energy, Lagrangian techniques (another topic for another day) are used to compute the dynamics (i.e., the coordinates as a function of time).

Apparently very similar to the hybrid models described in the previous section (such as Terzopoulos's superquadric ellipsoids), another method represents the potential (strain) energy of a deformable object from its local stretching and bending by integrating the weighted sums of the squares of low-order partial derivatives over its surface. If  $w(u, v)$  gives the positions in 3D of points using parameters  $u$  and  $v$  (just like  $r(s_1, s_2)$  in the notation we've seen previously), the potential energy  $E$  (presumably equivalent to the  $V$  we've seen in previous notation)

$$E = \int_{\Gamma} (\alpha_{11} \left\| \frac{\partial w}{\partial u} \right\|^2 + 2\alpha_{12} \frac{\partial w}{\partial u} \frac{\partial w}{\partial v} + \alpha_{22} \left\| \frac{\partial w}{\partial v} \right\|^2 + \beta_{11} \left\| \frac{\partial^2 w}{\partial u \partial u} \right\|^2 + 2\beta_{12} \left\| \frac{\partial^2 w}{\partial u \partial v} \right\|^2 + \beta_{22} \left\| \frac{\partial^2 w}{\partial v \partial v} \right\|^2 - 2f \cdot w) dudv \quad (81)$$

Constraints, such as control points, can be specified in a state vector  $x$ , and  $E$  is minimized subject to the constraints  $x$  using standard constrained optimization techniques (another topic for another day).

## Conclusions

Several of the main current challenges in deformable modelling are listed: measuring material properties (which has been done by a few people using mechanical testing and light interference techniques), performing quantitative comparisons and validations of simulation results to real objects, and improving speed enough to simulate deformations of large models with high accuracy in real-time. With many hours of pre-processing, the state-of-the-art performance in 1997 seems to be implied to be 8000 node FEM models at 15 frames per second. Parallel methods and hierarchical methods (e.g. modal analysis, multigrid FEM, and ChainMail) are said to be promising for increasing performance.