



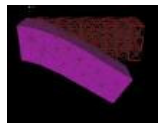
``Deformable Modeling``

Overview

- Problem Definition
- Inspirational Videos
- Geometric Methods
- Mass-springs
- Finite Element Method
- Approximate Continuum Methods
- Low Degree of Freedom Methods
- Boundary Element Method
- Long Element Method
- Matthias Teschner's Work
- Other Good Ideas
- Parameterizing Models

Defining the Problem

- We want to make non-rigid objects look and feel realistic when forces are applied
- Input
 - Forces
 - Over whole volume (e.g., gravity)
 - Over the surface (e.g., pressure, drag)
 - Concentrated loads (e.g., poking with haptic device)
 - Original, undeformed geometry (usually assumed here to be a triangular mesh)
- Output:
 - New, deformed geometry
 - Static equilibrium
 - At each time step (dynamic)
 - If using meshes, just need node displacements
 - Usually assumes invariant topology (e.g., no cutting)
- Haptics
 - Can just use updated node positions and use proxy or other method as usual
 - Don't need to do all deformation computation on haptics thread
 - Local models
 - Some algorithms we'll see (e.g., BEM, LEM) explicitly calculate haptic forces)

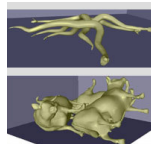
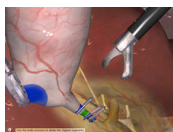
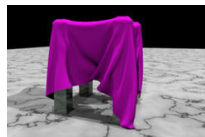


Definitions and Types of Deformations

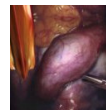
- Types of Deformations
 - Elastic – once forces are removed, object returns to original shape (like rubber)
 - Plastic – when forces are removed, object does not return to original shape (like denting plastic)
 - Linear – if a force f results in a deformation d , a force of $a*f$ will result in a deformation $a*d$ (like steel)
 - Isotropic – deforms in the same way in all directions (like metals)
 - Anisotropic – deforms in different ways in different directions (like wood)
- Other Fundamental Definitions
 - Stress – a pressure (force per unit area) within a body that balances an applied force
 - Strain – a fractional change in length ($\Delta l/l_0$) caused by a stress
 - Young's modulus – a parameter that relates stress to strain, $y = \text{stress}/\text{strain}$; for linear materials, it is constant over a large range of strains
 - Poisson's ratio - the ratio of the transverse to longitudinal strains produced by a stress along the longitudinal axis of the object
 - Completely compressible if $\nu=0$
 - Completely incompressible if $\nu=0.5$

Applications

- Computer animation
 - Cloth
 - Faces
 - Human and animal characters
- Surgical simulation
- CAD



Inspirational Videos



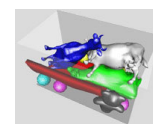
Doug James and Dinesh Pai
University of British Columbia
(now CMU and Rutgers)



Ron Fedkiw, Robert Bridson, and John Anderson
Stanford
not real-time



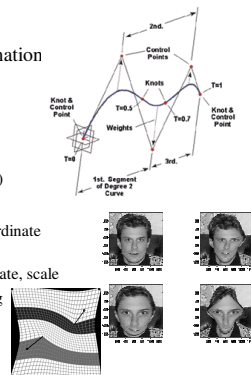
CO-ME Group
ETH Zurich



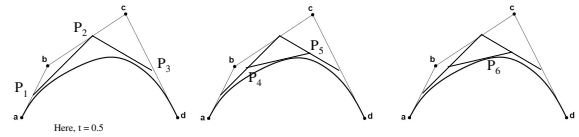
Matthias Teschner and others
University of Freiburg

Geometrical Techniques

- Non-physical models
- Allow artist to directly specify deformation each time point
- Deform curves using control points
 - Bezier curves
 - B-splines
 - Non-uniform rational B-splines (NURBS)
- Free-form deformations
 - Deform by transforming the object's coordinate system
 - Rigid-body transformations: rotate, translate, scale
 - Deformations: bending, tapering, warping



Bezier Curves



- For each point t on the curve, $0 < t < 1$, first find the point P_1 that is a fraction t from a to b , P_2 that is a fraction t from b to c , and P_3 that is a fraction t from c to d

$$P_1 = (1-t)A + tB \quad P_2 = (1-t)B + tC \quad P_3 = (1-t)C + tD$$

- Then find the point P_4 that is a fraction t from P_1 to P_2 , and P_5 that is a fraction t from P_2 to P_3

$$P_4 = (1-t)P_1 + tP_2 \quad P_5 = (1-t)P_2 + tP_3$$

- Finally, find the point P_6 on the curve that is a fraction t from P_4 to P_5

$$P_6 = (1-t)P_4 + tP_5 = A(1-t)^3 + 3B(1-t)^2t + 3C(1-t)t^2 + Dt^3$$

Geometrical Techniques : Evaluation

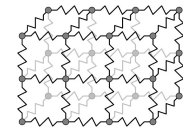


- Advantages
 - Very low computational demands
 - Direct control over deformations



- Disadvantages
 - Accuracy limited by 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



- Discretize the object into a collection of N nodes interconnected with springs
- For each node i , just using Newton's $F=ma$ law, with some velocity-dependent damping,

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

where m_i is the mass of node i , γ_i is its damping coefficient, x_i its displacement (relative to its rest position), \dot{x}_i its velocity, and \ddot{x}_i its acceleration

- 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, it is just $\sum_j k_{ij}(x_j - x_i)$, where k_{ij} is the stiffness coefficient for the spring between nodes i and j (zero if not connected by a spring)
- f_i is the total of any external forces (such as loads or poking objects) at the node
- Combining the equations for all N nodes, we get the system of equations

$$M\ddot{x} + C\dot{x} + Kx = f$$

where M and C are $3N \times 3N$ diagonal matrices, K is a symmetric, banded $3N \times 3N$ matrix, and x , \dot{x} , \ddot{x} , and f are $3N \times 1$ vectors

Solving the Mass-Spring System

- We can solve the system

$$M\ddot{x} + C\dot{x} + Kx = f$$

by rewriting it as a system of first-order differential equations

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

$$\dot{x} = v$$

- The inverse of M can be pre-computed (and is trivial since it is diagonal)
- Initialize $x_0 = 0$, $v_0 = 0$
- At each time-step (of duration Δt), calculate \dot{x} and \dot{v} in the equations above using the values of x and v from the previous time-step
- Update the displacement and velocity for the next time-step (Euler method)

$$x_{t+\Delta t} = x_t + \dot{x}\Delta t$$

$$v_{t+\Delta t} = v_t + \dot{v}\Delta t$$
- More accurate explicit integration methods, such as Runge-Kutta, may be used instead (speed/accuracy trade-off)

Implicit Integration

- In the previous slide, we used explicit integration to solve a differential equation

$$y' = f(t, y)$$

which only used values of the variables at time t_k to calculate new values at time t_{k+1} .

$$y_{k+1} = y_k + f(t_k, y_k)\Delta t$$

- Implicit integrators use values at t_k and at t_{k+1} to calculate new values at t_{k+1} .

$$y_{k+1} = y_k + f(t_{k+1}, y_{k+1})\Delta t$$

- Since we don't know values at t_{k+1} yet, this results in a (not necessarily linear) system of equations
- This method is provably more stable, and can take larger step-sizes, but each step is significantly more costly

Mass Springs : Evaluation



- Advantages
 - Easy to construct
 - Fairly low computational demands
 - Parallelize well
 - More accurate than geometric techniques
 - Can model cutting by just setting K_{ij} to zero

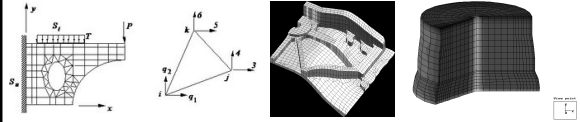


- Disadvantages
 - Not very accurate physical model
 - Difficult to tune spring constants – not related to a real material property
 - Constraints such as incompressible volumes and thin surfaces are difficult to express
 - Difficult to model rigid objects since large spring constants make the system of equations unstable, requiring small time-steps
 - Can go unstable if updated too slowly



NOT VERY ACCURATE

Finite Element Method



- Discretize model into a finite number of simple elements (such as tetrahedrons)
- Considers forces throughout entire volume, not just at nodes
- System will reach equilibrium when potential energy is minimized
- We express the potential energy of the system in terms of our unknowns – the node displacements
- Set derivative of potential energy function to zero to find its minimum

Potential Energy

- Total potential energy Π of the system consists of
 - Strain energy of the deformed object, Λ
 - Energy lost by the loads acting on the object as they do work by deforming it, W
- $\Pi = \Lambda - W$
- We will consider this equation for a single element, then aggregate the equations for all elements and solve

Interpolation Functions (1)

- We will need an expression that will give us the value of some function Φ for any point inside an element in terms of the value of that function at the nodes of the element
- Specifically, we will need this for Φ being the displacement function
- The expression obviously depends on the type of element
- As a simple example, for a 2D triangular element, we need to find parameters a_1 , a_2 , and a_3 such that, for any point (x,y) in the triangle,

$$\Phi(x,y) = a_1 + a_2x + a_3y$$

- We know the value of Φ at the nodes, so we have a system of three equations in three unknowns:

$$\begin{aligned} \Phi_1 &= a_1 + a_2x_1 + a_3y_1 \\ \Phi_2 &= a_1 + a_2x_2 + a_3y_2 \\ \Phi_3 &= a_1 + a_2x_3 + a_3y_3 \end{aligned}$$

Interpolation Functions (2)

- Solving this system gives the parameters in terms of the node coordinates and the function value at the nodes

$$\begin{aligned} a_1 &= [(x_2y_3 - x_3y_2)\Phi_1 + (x_3y_1 - x_1y_3)\Phi_2 + (x_1y_2 - x_2y_1)\Phi_3] / (2A) \\ a_2 &= [(x_2 - x_3)\Phi_1 + (x_3 - x_1)\Phi_2 + (x_1 - x_2)\Phi_3] / (2A) \\ a_3 &= [(x_3 - x_1)y_1\Phi_1 + (x_1 - x_2)y_2\Phi_2 + (x_2 - x_3)y_3\Phi_3] / (2A) \end{aligned}$$

where A is the area of the triangle (easily calculated using Heron's Formula or $1/2$ times cross product)
- Given these values, we can rearrange the equation $\Phi(x,y) = a_1 + a_2x + a_3y$ into the form $\Phi(x,y) = h_1\Phi_1 + h_2\Phi_2 + h_3\Phi_3$

$$\begin{aligned} h_1 &= [(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y] / (2A) \\ h_2 &= [(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y] / (2A) \\ h_3 &= [(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y] / (2A) \end{aligned}$$
- For a 3D tetrahedron, you would solve the system

$$\begin{aligned} \Phi &= a_1 + a_2x_1 + a_3y_1 + a_4z_1 \\ \Phi &= a_1 + a_2x_2 + a_3y_2 + a_4z_2 \\ \Phi &= a_1 + a_2x_3 + a_3y_3 + a_4z_3 \\ \Phi &= a_1 + a_2x_4 + a_3y_4 + a_4z_4 \end{aligned}$$

for a_1 , a_2 , a_3 , and a_4 , and then rewrite it in the form $\Phi(x,y,z) = h_1\Phi_1 + h_2\Phi_2 + h_3\Phi_3 + h_4\Phi_4$, where the Φ_i are functions of x,y,z , and of the four node coordinates (x_1, y_1, z_1) , (x_2, y_2, z_2) , (x_3, y_3, z_3) , and (x_4, y_4, z_4)

Displacement Function

- We are interested in a function $\Phi : (x,y,z) \rightarrow (u,v,w)$, where (u,v,w) is the displacement of the point (x,y,z)
- We have shown that we can express $\mathbf{u} = (u,v,w)$ as a linear combination of the displacements at the N nodes

$$\mathbf{u} = \mathbf{H}\mathbf{U}$$

where \mathbf{H} is a $3 \times 3N$ matrix and \mathbf{U} is a $3N \times 1$ vector

- For example, for a tetrahedron with $N=4$ nodes,

$$\begin{aligned} u &= h_1u_1 + h_2u_2 + h_3u_3 + h_4u_4 \\ v &= h_1v_1 + h_2v_2 + h_3v_3 + h_4v_4 \\ w &= h_1w_1 + h_2w_2 + h_3w_3 + h_4w_4 \end{aligned}$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 & 0 & 0 & h_4 & 0 & 0 \\ 0 & h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 & 0 & 0 & h_4 & 0 \\ 0 & 0 & h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 & 0 & 0 & h_4 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \\ u_3 \\ v_3 \\ w_3 \\ u_4 \\ v_4 \\ w_4 \end{bmatrix}$$

$$\mathbf{u} = \mathbf{H}\mathbf{U}$$

Strain

- Strain ϵ is a 6*1 vector that relates changes in displacements to changes in position

- An example using a triangle element:

$$\begin{aligned} u &= h_1 u_1 + h_2 u_2 + h_3 u_3 & h_1 &= [(x_2 y_3 - x_3 y_2) + (y_2 - y_3)x + (x_3 - x_2)y] / (2A) \\ v &= h_1 v_1 + h_2 v_2 + h_3 v_3 & h_2 &= [(x_3 y_1 - x_1 y_3) + (y_3 - y_1)x + (x_1 - x_3)y] / (2A) \\ w &= h_1 w_1 + h_2 w_2 + h_3 w_3 & h_3 &= [(x_1 y_2 - x_2 y_1) + (y_1 - y_2)x + (x_2 - x_1)y] / (2A) \end{aligned}$$

$$\text{So, } \epsilon_x = \frac{\partial u}{\partial x} = \frac{(y_2 - y_3)u_1 + (y_3 - y_1)u_2 + (y_1 - y_2)u_3}{2A}$$

- We can express the strain as a matrix equation $\epsilon = \mathbf{B}\mathbf{U}$, where \mathbf{B} is a 6*3N matrix, whose (first, second, third) row is obtained by differentiated with respect to (x,y,z), and then (for each column), with respect to each of the 3N node displacements

For a triangular element:

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

Strain Energy

- The strain energy Λ over the body can be expressed as the product of the stress σ and strain ϵ vectors integrated over the volume V :

$$\Lambda = \frac{1}{2} \int_V \sigma^T \epsilon \, dV$$

- Dimensional analysis sanity check: Stress is force per unit area, and strain is length per length (unitless). So this integrated over a volume gives

$$\frac{\text{kg} \cdot \text{m}}{\text{s}^2} \cdot \frac{1}{\text{m}^2} \cdot \frac{\text{m}}{\text{m}} \cdot \text{m}^3 = \frac{\text{kg} \cdot \text{m}^2}{\text{s}^2} = \frac{\text{kg} \cdot \text{m}}{\text{s}^2} \cdot \text{m}$$

which is a force times a distance, which is work or energy.

- Since stress and strain are related (for linear elasticity, Young's modulus $Y = \text{stress/strain}$), the stress can be expressed in terms of the strain

$$\sigma^T = \epsilon^T \mathbf{D}$$

where \mathbf{D} is a linear matrix that relates the stress and strain components (for linear elasticity, it can just have Young's modulus along the diagonal)

- Thus, we have

$$\Lambda = \frac{1}{2} \int_V \epsilon^T \mathbf{D} \epsilon \, dV$$

- Since $\epsilon = \mathbf{B}\mathbf{U}$,

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

(we can factor out \mathbf{U}^T because we are integrating over (x,y,z) not (u,v,w))

Elements

element type	# nodes	interpolation equation	interpolation functions
linear triangular area A	3	$\Phi = a_1 + a_2 x + a_3 y$	$h_1 = [(x_2 y_3 - x_3 y_2) + (y_2 - y_3)x + (x_3 - x_2)y] / (2A)$ $h_2 = [(x_3 y_1 - x_1 y_3) + (y_3 - y_1)x + (x_1 - x_3)y] / (2A)$ $h_3 = [(x_1 y_2 - x_2 y_1) + (y_1 - y_2)x + (x_2 - x_1)y] / (2A)$
bilinear rectangular width w height h area A	4	$\Phi = a_1 + a_2 x + a_3 y + a_4 xy$	$h_1 = (w + x_1 - x)(h + y_1 - y) / A$ $h_2 = (x - x_1)(h + y_1 - y) / A$ $h_3 = (w + x_1 - x)(y - y_1) / A$ $h_4 = (x - x_1)(y - y_1) / A$
quadratic triangular	6	$\Phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2$	see FEM text
Lagrangian	9	$\Phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^2 y + a_8 x y^2 + a_9 x^2 y^2$	see FEM text
tetrahedral	4	$\Phi = a_1 + a_2 x + a_3 y + a_4 z$	see FEM text
20-node brick	20	see FEM text	see FEM text

- Trade-off: elements with more nodes are more accurate, and can use fewer of them, but lead to larger and more complicated equations to solve
- For more complicated elements, such as the bilinear rectangle, the interpolation functions may not be linear in x, y, and z; thus, \mathbf{B} may include the variables (x,y,z), which will make it more difficult when we integrate an expression involving this matrix over the volume of the element

Work

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

$$\mathbf{W} = \int_V \mathbf{u} \cdot \mathbf{f} \, dV$$

- These forces can include body forces \mathbf{f}_b over the whole volume (such as gravity), surface forces \mathbf{f}_s (such as pressure or drag), and concentrated loads \mathbf{p}_i at nodes i

$$\mathbf{W} = \int_V \mathbf{u} \cdot \mathbf{f}_b \, dV + \int_V \mathbf{u} \cdot \mathbf{f}_s \, dS + \sum_i \mathbf{u}_i \cdot \mathbf{p}_i$$

- Since $\mathbf{u} = \mathbf{H}\mathbf{U}$, and, by the definition of dot product, $\mathbf{u} \cdot \mathbf{f} = \mathbf{u}^T \mathbf{f}$,

$$\mathbf{W} = \int_V \mathbf{U}^T \mathbf{H}^T \mathbf{f}_b \, dV + \int_V \mathbf{U}^T \mathbf{H}^T \mathbf{f}_s \, dS + \mathbf{U}^T \mathbf{P}$$

$$\mathbf{W} = \mathbf{U}^T \left(\int_V \mathbf{H}^T \mathbf{f}_b \, dV + \int_V \mathbf{H}^T \mathbf{f}_s \, dS + \mathbf{P} \right)$$

Putting it all Together

- We wanted to express the total potential energy $\Pi = \Lambda - \mathbf{W}$ as a function of the unknown node displacements
- We have done this:

$$\Pi = \frac{1}{2} \mathbf{U}^T \left(\int_V \mathbf{B}^T \mathbf{D} \mathbf{B} \, dV \right) \mathbf{U} - \mathbf{U}^T \left(\int_V \mathbf{H}^T \mathbf{f}_b \, dV + \int_V \mathbf{H}^T \mathbf{f}_s \, dS + \mathbf{P} \right)$$

- Taking the derivative with respect to \mathbf{U} , we get

$$\frac{\partial \Pi}{\partial \mathbf{U}} = \left(\int_V \mathbf{B}^T \mathbf{D} \mathbf{B} \, dV \right) \mathbf{U} - \left(\int_V \mathbf{H}^T \mathbf{f}_b \, dV + \int_V \mathbf{H}^T \mathbf{f}_s \, dS + \mathbf{P} \right)$$

- Dimensions: \mathbf{B}^T is 3N*6, \mathbf{D} is 6*6, and \mathbf{B} is 6*3N, so the first integral evaluates to a 3N*3N matrix \mathbf{K} . \mathbf{H}^T is 3N*3 and \mathbf{f}_b and \mathbf{f}_s are each 3*1, so the second two integrals evaluate to 3N*1 vectors. \mathbf{P} is also a 3N*1 vector, so the last three terms add up to a 3N*1 vector \mathbf{F} .
- Therefore, if we set this derivative to zero, we have a linear system

$$\mathbf{K}\mathbf{U} = \mathbf{F}$$

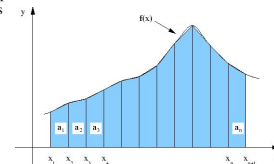
for the 3N unknown displacements in \mathbf{U} .

- This is the equation for a single element. We aggregate all the equations for all elements into one big linear system. Nodes are shared by multiple elements, so it will be of dimension less than 3MN, where M is the number of elements

Numerical Integration (Quadrature)

- To get the force vectors, we need to evaluate the integrals $\int_V \mathbf{H}^T \mathbf{f}_b \, dV$ and $\int_V \mathbf{H}^T \mathbf{f}_s \, dS$
- We also need to evaluate the integral $\int_V \mathbf{B}^T \mathbf{D} \mathbf{B} \, dV$
 - If \mathbf{B} is not a function of (x,y,z), such as for simple elements like a triangle or tetrahedron, the integral is trivial (just multiply by the volume)
 - If we assume small deformations, we can assume the node coordinates $(x_1, y_1, z_1) \dots (x_N, y_N, z_N)$ don't change, and we don't have to re-evaluate this integral at every time step

- Numerical integration is used to compute the definite integral of a function by sampling its value at various points along the region of interest rather than analytically finding the indefinite integral
- Simple example is the Trapezoid Rule, which is a member of the Newton-Cotes family of numerical quadrature rules



- Other, usually more accurate techniques, include Gaussian and Gauss-Kronrod Quadrature Rules

Solving Linear Systems

- Most basic general method is Gaussian elimination, which annihilates subdiagonal elements by adding linear multiples of rows to other rows to reduce the matrix to an upper triangular system, which can be solved by back-substitution

- For example, to solve the system $2x+3y=21$ and $4x+2y=22$,

$$\begin{bmatrix} 2 & 3 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 21 \\ 22 \end{bmatrix}$$

we annihilate the subdiagonal entry a_{21} in the first column by adding $-a_{21}/a_{11} = -4/2 = -2$ times the first row to the second row

$$\begin{bmatrix} 2 & 3 \\ 0 & -4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 21 \\ -20 \end{bmatrix}$$

- Now we can directly solve the last equation, $y = -20/-4 = 5$, and substitute this back into the first equation to get $x = (21-3*5)/2 = 3$
- Pivoting (row swapping) improves stability
- Gauss-Jordan elimination reduces matrix to a diagonal system
- In general, time complexity is $O(n^3)$
- Many tricks exist to solve systems much more quickly when the matrix has special properties, such as if it is sparse or banded

Dynamic Deformation

- To get dynamics rather than just static equilibrium, instead of solving the linear system $\mathbf{K}\mathbf{U}=\mathbf{F}$, solve the differential equation

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{C}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{F}$$

where \mathbf{K} , \mathbf{F} , and \mathbf{U} are defined the same as in the static case, \mathbf{M} is the mass matrix, calculated from \mathbf{H} and the density ρ

$$\mathbf{M} = \int_V \rho \mathbf{H}^T \mathbf{H} dV$$

and \mathbf{C} is computed from damping parameters or as a linear combination of \mathbf{M} and \mathbf{K}

- We can solve this in the same way as in mass springs

$$\dot{\mathbf{V}} = \mathbf{M}^{-1}(-\mathbf{C}\mathbf{V} - \mathbf{K}\mathbf{U} + \mathbf{F})$$

$$\dot{\mathbf{U}} = \mathbf{V}$$

- \mathbf{F} (and potentially \mathbf{K}) change at each time-step, so we have to re-solve this differential equation at each time step

IMPACT: An FEM Implementation (1)

Description: Simple example using two-node rod elements, pure linear elastic material, and explicit integration

Input: list of nodes with their (x,y,z) coordinates; list of elements with their nodes; material type; element cross-sectional area and density; Young's modulus for the material; list of constraints (boundary constraint: fixed nodes); list of loads (force vectors and locations)

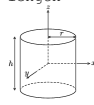
Output: At each time-step, the displacement (from original position) of each node and stress and strain of each element

Initialization:

```
for each element
  mass = density*cross_sectional_area*length
  add half of mass to each node
  inertia = [ MR^2/2  0  0
             0  ML^2/12  0
             0  0  ML^2/12 ]
  add half of inertia to each node
```



$$I = ML^2/12$$



$$I = MR^2/2$$

IMPACT: An FEM Implementation (2)

Simulation Loop:

```
for each timestep t = t+1
  for each element
    l_0 = |node1pos_{t-1} - node2pos_{t-1}|
    Δl = ||node1pos_t - node1pos_{t-1}| - |node2pos_t - node2pos_{t-1}||
    strain = Δl/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
    acceleration_inn = force/mass
    acceleration_rot = inertia^-1(force_rot - (velocity_rot *
      (inertia*velocity_rot)))
    velocity += acceleration*timestep
    displacement += velocity*timestep
```

FEM : Evaluation



- Advantages
 - Very physically realistic simulation
 - Fewer nodes usually needed than with mass-springs

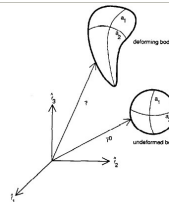
➡ HIGH ACCURACY



- Disadvantages
 - Forces must be numerically integrated over volume and/or surface at each time-step, requiring a lot of computation
 - Topology changes or large deformations change \mathbf{B} (and \mathbf{M} for dynamic deformation computation), requiring you to numerically integrate to recompute the large stiffness matrix \mathbf{K} (and \mathbf{M}) throughout the simulation

➡ TOO SLOW for real-time, especially haptics

Approximate Continuum Models



- Main idea: Is there a simpler way to estimate the strain energy Λ ?
- Still have some direct relationship to physics, but sacrifice some accuracy for speed

- Example: Demetri Terzopoulos's *Elastically Deformable Models*
- Parameterize the volume with some function $r(a_1, a_2, a_3)$
- Estimate $\Lambda(r)$ as $\int_V \|\mathbf{G} - \mathbf{G}^0\|_\alpha^2 da_1 da_2 da_3$
where \mathbf{G} is the "first fundamental form" of the deformed body and \mathbf{G}^0 is the "first fundamental form" of the reference, undeformed body

First Fundamental Form

- The first fundamental form relates changes in parameters to changes in Euclidean distance

$$dl^2 = \sum_{i,j} G_{ij} da_i da_j$$

- Two volumes have the same shape if their first fundamental forms are equivalent
- $\int_{\Omega} \|\mathbf{G} - \mathbf{G}^0\|_{\alpha}^2 da_1 da_2 da_3$ estimates the strain energy, because, 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 are now far apart), then there is a lot of deformation going on, resulting in a lot of strain energy
- The first fundamental form can be calculated

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

First Fundamental Form : Example

- For a cylinder, $r(a_1, a_2, a_3) = r(\rho, \theta, z) = [\rho \cos(\theta) \quad \rho \sin(\theta) \quad z]$
- The partial derivatives are

$$\frac{\partial r}{\partial a_1} = \frac{\partial r}{\partial \rho} = [\cos(\theta) \quad \sin(\theta) \quad 0] \quad \frac{\partial r}{\partial a_2} = \frac{\partial r}{\partial \theta} = [-\rho \sin(\theta) \quad \rho \cos(\theta) \quad 0] \quad \frac{\partial r}{\partial a_3} = \frac{\partial r}{\partial z} = [0 \quad 0 \quad 1]$$

- Thus we can calculate G as

$$\mathbf{G} = \begin{bmatrix} \frac{\partial r}{\partial a_1} \cdot \frac{\partial r}{\partial a_1} & \frac{\partial r}{\partial a_1} \cdot \frac{\partial r}{\partial a_2} & \frac{\partial r}{\partial a_1} \cdot \frac{\partial r}{\partial a_3} \\ \frac{\partial r}{\partial a_2} \cdot \frac{\partial r}{\partial a_1} & \frac{\partial r}{\partial a_2} \cdot \frac{\partial r}{\partial a_2} & \frac{\partial r}{\partial a_2} \cdot \frac{\partial r}{\partial a_3} \\ \frac{\partial r}{\partial a_3} \cdot \frac{\partial r}{\partial a_1} & \frac{\partial r}{\partial a_3} \cdot \frac{\partial r}{\partial a_2} & \frac{\partial r}{\partial a_3} \cdot \frac{\partial r}{\partial a_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \rho^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then, substituting into $dl^2 = \sum_{ij} G_{ij} da_i da_j$, we get

$$dl^2 = da_1^2 + \rho^2 da_2^2 + da_3^2 = d\rho^2 + \rho^2 d\theta^2 + dz^2$$

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 is $\rho^2 d\theta^2$ (recall arc length = $r\theta$), and if there is a change in height, the square of the distance is dz^2 , so everything checks out – changes in parameters are correctly mapped to changes in Euclidean distance

Surfaces and Snakes

- For a surface, we estimate the strain energy using two 2*2 matrices (using two parameters) instead of one 3*3 matrix (with three parameters)

$$\int_{\Omega} \|\mathbf{G} - \mathbf{G}^0\|_{\alpha}^2 + \|\mathbf{B} - \mathbf{B}^0\|_{\beta}^2 da_1 da_2$$

- The first fundamental form, \mathbf{G} , is a measure of the amount of movement of a surface in the parameter plane
- The second fundamental form \mathbf{B} is a measure of the change in the normal vector and the change of surface position at a surface point as a function of a small movement in the parameter space
- Whether using a volume or surface to get the strain energy estimate, which they call $\epsilon(r)$, they then use it in an equation similar to the mass-spring-damper equation

$$\frac{\partial}{\partial t} \left(\mu \frac{\partial r}{\partial t} \right) + \gamma \frac{\partial r}{\partial t} + \frac{\delta \epsilon(r)}{\delta r} = f(r, t)$$

- A related method is Terzopoulos's 1D deformable "snakes", parameterized by s
- Strain energy is estimated by squares of the first and second derivatives:

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

- First derivative measures axial deformations
- Second derivative measures bending deformations

Approximate Continuum Models : Evaluation



- Advantages
 - More realistic than mass-springs
 - More related to physics than mass-springs
 - Faster than FEM



- Disadvantages
 - Not as accurate as FEM
 - Not as directly related to physics as FEM
 - Not as fast as mass-springs
 - Equations become ill-conditioned for rigid objects, although Terzopoulos later proposed separating the rigid body dynamics and deformations into separate components

Low Degree of Freedom Models

- For the mass-spring model, the generalized eigenvalues and eigenvectors of the mass and stiffness matrices determine the natural frequencies and modes of vibration
- First order eigenvector ϕ and eigenvalue λ for a matrix \mathbf{A} are a vector and scalar such that $\mathbf{A} \phi = \lambda \phi$
- Second order eigenvector and eigenvalue for two matrices, such as \mathbf{K} and \mathbf{M} , are a vector and scalar such that $\mathbf{K} \phi = \lambda \mathbf{M} \phi$
- Simple (but numerically unstable) way to calculate second-order eigenvectors is just as first-order eigenvectors of $\mathbf{M}^{-1}\mathbf{K}$, since we can rewrite above equation as $(\mathbf{M}^{-1}\mathbf{K}) \phi = \lambda \phi$
- The matrix whose columns are the eigenvalues is Φ and the diagonal matrix whose diagonal elements are the eigenvalues is Λ
- We can take the mass-springs equation $\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{f}$ and multiply both sides by Φ^T and let $\mathbf{x} = \Phi \tilde{\mathbf{x}}$, to get $\Phi^T \mathbf{M} \Phi \ddot{\tilde{\mathbf{x}}} + \Phi^T \mathbf{C} \Phi \dot{\tilde{\mathbf{x}}} + \Phi^T \mathbf{K} \Phi \tilde{\mathbf{x}} = \Phi^T \mathbf{f}$
- Since \mathbf{M} and \mathbf{K} are symmetric positive definite, they are diagonalized by Φ : $\Phi^T \mathbf{M} \Phi = \tilde{\mathbf{M}}$ and $\Phi^T \mathbf{K} \Phi = \tilde{\mathbf{K}}$, where $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{K}}$ are diagonal
- If \mathbf{C} is a linear combination of \mathbf{M} and \mathbf{K} , we can compute a diagonal matrix $\Phi^T \mathbf{C} \Phi = \tilde{\mathbf{C}}$
- With these definitions, we have $\tilde{\mathbf{M}} \ddot{\tilde{\mathbf{x}}} + \tilde{\mathbf{C}} \dot{\tilde{\mathbf{x}}} + \tilde{\mathbf{K}} \tilde{\mathbf{x}} = \tilde{\mathbf{f}}$
- Since our matrices are now diagonal, each equation (row) is independent, and the relative importance of their contributions are proportional to their eigenvalues. Thus, we can adaptively control our accuracy by using only the M most important modes (by just dropping the rows and columns corresponding to the smallest $N - M$ eigenvalues), and solving the system $\tilde{\mathbf{M}} \ddot{\tilde{\mathbf{x}}} + \tilde{\mathbf{C}} \dot{\tilde{\mathbf{x}}} + \tilde{\mathbf{K}} \tilde{\mathbf{x}} = \tilde{\mathbf{f}}$, then convert our result back using $\mathbf{x} = \Phi \tilde{\mathbf{x}}$

Low Degree of Freedom Models : Evaluation



- Advantages
 - Allow adaptive control of accuracy / computation time trade-off by throwing away least important modes (similar to other eigenvalue-based compression schemes, such as SVD) when pressed for time



- Disadvantages
 - Only further decreases mass spring realism
 - Requires additional pre-computation

ArtDefo (Boundary Element Method)

- Developed by Doug James and Dinesh Pai
- Assumes linear deformations
 - The amount of deformation at node i due to a deformation u at node j is $H_{ij}u$
 - If node i moves x due to a deformation y at node j , it will move $2x$ in response to a deformation $2y$ at node j
 - Same for forces: a force p at node j will result in a force $G_{ij}p$ at node i
- The boundary integral formulation depends only on the geometry of the boundary; we don't need any elements in the interior of the volume



Pre-computation

- The basis of the model is a boundary integral formulation of Navier's equation

$$c\mathbf{u} + \int_{\Gamma} \mathbf{p}^* \mathbf{u} d\Gamma = \int_{\Gamma} \mathbf{u}^* \mathbf{p} d\Gamma$$

where c is a function of the boundary geometry, \mathbf{u} is the node displacements, \mathbf{p} is forces at the nodes, and \mathbf{u}^* and \mathbf{p}^* are fundamental solutions that depend only on known elasticity properties

- Through some complicated calculus, this integral equation is reduced to a matrix equation

$$\mathbf{H}\mathbf{u} = \mathbf{G}\mathbf{p}$$

where the matrices \mathbf{G} and \mathbf{H} essentially relate forces and displacements, respectively, at each node to forces and displacements at every other node

- These are each $3N \times 3N$ matrices
- The values for all elements of \mathbf{G} and \mathbf{H} are pre-computed based on material properties
 - Poisson's ratio (ν) – the ratio of the transverse to longitudinal strains produced by a stress along the longitudinal axis of the object
 - Completely compressible if $\nu=0$; completely incompressible if $\nu=0.5$
 - Shear modulus (G) – the ratio of shear stress to shear strain

Simulation

- For each node, we know *either* the displacement or the force
 - If the haptic device is in contact with the node, its displacement is wherever the device pushes it, but we need to calculate a force to push back on the haptic device
 - If the haptic device is not in contact with the node, its force is zero, but we need to calculate its displacement (due to the haptic device contacting other nodes)
- Thus, we have $3N$ unknowns – some of them in \mathbf{u} and some in \mathbf{p}
- By some linear algebra tricks, the matrix equation $\mathbf{H}\mathbf{u} = \mathbf{G}\mathbf{p}$ can be transformed into an equation $\mathbf{A}\mathbf{v} = \mathbf{z}$, where all of the known forces and displacements are in \mathbf{z} and all the unknown ones in \mathbf{v}
- \mathbf{A} is inverted, and the unknowns \mathbf{v} calculated as $\mathbf{A}^{-1}\mathbf{z}$
- Unfortunately, when the haptic device moves into or out of contact with a node, the set of known and unknown quantities changes, so \mathbf{A} also changes, and we have to re-invert it
- Since the contact state of only a few nodes will change in a time-step, \mathbf{A} won't change *very much*, so we can use the Sherman-Morrison-Woodbury formula, which computes an inverse of a matrix given the inverse of a similar matrix (with a rank- k change) in $O(n^2k)$ instead of $O(n^3)$ time
- Since the haptic device will likely move to neighbors of the nodes it is currently contacting, free CPU cycles can be used in real-time to pre-compute inverses of \mathbf{A} 's that are likely to arise soon

ArtDefo / BEM : Evaluation

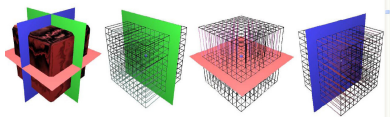


- Advantages
 - Faster than FEM, at least as long as not too many boundary conditions (contact nodes) change
 - Requires fewer elements than FEM (and none in interior)
 - Uses real physical properties (such as shear modulus and Poisson's ratio), unlike mass-springs
 - Doesn't go unstable if updated too slow, unlike mass-springs
 - Designed for haptic interaction



- Disadvantages
 - Still a lot of matrix inversions
 - Makes linear elastic assumption; can't have non-linearities
 - Can't model any forces that act in interior (only on boundary)

Long Element Method (LEM)



- Developed by Remis Balaniuk, Christian Laugier, Ivan Costa, and Kenneth Salisbury
- An object is meshed into "long elements" (rectangular prisms) along each of three orthogonal directions (i.e., x , y , and z axes)
- We solve a 1D deformation problem along each axis, which maps each cell in the original configuration to its new, deformed location

Equations for Each Long Element



- The external and internal pressures should balance

$$P_{\text{ext}} = P_{\text{int}}$$
- The external pressure includes the atmospheric pressure, P_{atm} , and the stress s caused by elongation, which can be expressed in terms of the strain, $s = E\Delta l/l$, where E is the modulus relating stress and strain

$$P_{\text{ext}} = P_{\text{atm}} + E\Delta l/l$$
- The internal pressure includes the fluid pressure, P_{fluid} , and the effect of gravity on the fluid, dgh , where d is the density, g gravity, and h the height at which the pressure is calculated

$$P_{\text{int}} = P_{\text{fluid}} + dgh$$
- We also include surface tension, as the deformation of an element propagates to its neighbors; the pressure resulting from this surface tension between nodes i and j is modeled as a spring

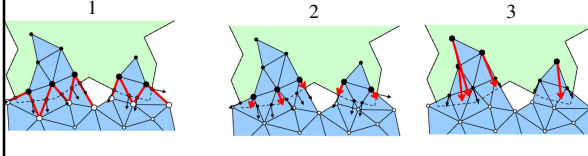
$$P_{\text{surface}} = k_s(\Delta l_i - \Delta l_j)/A_i$$
- We also include pressures P_{env} applied by the haptic device
- We define $\Delta P = P_{\text{fluid}} - P_{\text{atm}}$; Pascal's Principle states that an external pressure applied to a fluid in a closed container is transmitted undiminished throughout, so ΔP is the same for all elements
- Assuming node i has n neighbors, j_1 to j_n , the overall equation for node i is

$$(E_i/l_i + nk_s/A)\Delta l_i - k_s(\Delta l_{j_1} + \dots + \Delta l_{j_n})/A_i - \Delta P = d_i g h_i + P_{\text{env}}$$

The unknowns are the Δl_i for each of the N elements, and ΔP

Collision Response

1. After finding colliding (black) and non-colliding (white) points using collision detection algorithm, find "border edges" connected a black and a white point, and calculate intersection points (red) and surface normals
2. Approximate penetration depth and direction for all border points by averaging adjacent intersection points and normals
3. Propagate penetration depths and directions to all other colliding points by averaging values at adjacent nodes



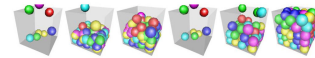
Teschner : Evaluation



- Advantages
 - Can control material properties by weighting distance, surface area, and volume preservation energies (e.g., high volume preservation and low distance preservation constants would be good for a water balloon)
 - Fast (especially collision detection)
 - Only the skinning is topology dependent; basic model allows for cutting

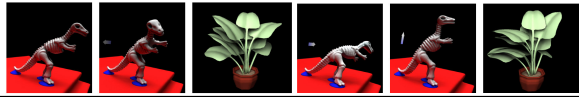


- Disadvantages
 - Not parameterized by real, measurable physical properties
 - Can miss some collisions



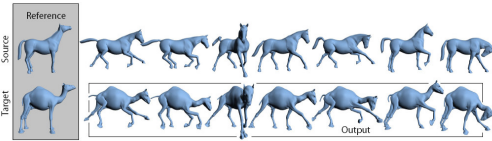
Other Good Ideas (1)

- "Precomputing Interactive Dynamic Deformable Scenes"
 - Doug James and Kayvon Fatahalian (CMU)
 - Choose a small set of "excitations" (impulse force, persistent force, and force duration) that are likely to occur during interactive use of a model
 - Run a (very long) FEM simulation with these excitations randomly applied (not letting it come to rest after each excitation)
 - Each time an excitation is applied, a new IRF (Impulse Response Function) database entry is created, consisting of the excitation, initial state when the excitation was applied (positions of all nodes), and the resulting trajectory (positions and velocities of all nodes for some time duration thereafter)
 - Then run the interactive simulation: each time one of these excitations is applied, find the IRF entry in the database with the most similar initial state to the current state, and just play back its stored trajectory
 - Compress IRF entries in database using SVD
- Phil Fong is building a database from excitation data recorded by probing real objects with the Phantom and recording deformations using projector and camera



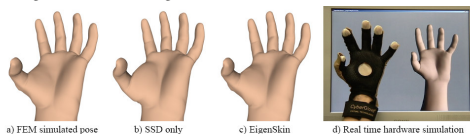
Other Good Ideas (2)

- "Deformation Transfer for Triangle Meshes"
 - Robert Sumner and Jovan Popovic (MIT)
 - Given a deformation that you like of some source mesh (perhaps obtained by FEM or range scanning), apply a similar deformation to a target mesh
 - For example, make a camel run like a horse
 - First pre-compute, for each target mesh triangle, the source mesh triangle it can be transformed to so as to optimize overall transform smoothness (nearby triangles have similar transforms), minimize overall transforms (so that the target mesh keeps its identity), and match up user specified control points
 - Solve a least-squares optimization problem to minimize absolute differences between the norms of the transformation matrices for matching triangles



Other Good Ideas (3)

- "Eigenskin"
 - Paul Kry, Doug James, Dinesh Pai
 - The problem: Say you have some deformable material (such as skin) attached to a rigid skeleton (such as bones), and want to model where the deformable material goes when the skeleton undergoes rigid body transformations
 - Using Skeletal Subspace Deformation (SSD), for each node of the skin mesh, its transformation is calculated as the weighted sum of the transformations of nearby bones (weights may just be proportional to distance)
 - Eigenskin learns the error in SSD as a function of bone transformations by comparing to lots of training data from off-line FEM simulations
 - It learns using SVD to determine the fundamental displacements
 - Then just simulate using SSD, but calculate the SSD error using the learned weights, and add a "fudge factor" to cancel out the error



Parameterizing Models

- Nearly all models discussed, except FEM, have parameters for the materials that are not directly related to real, measurable material properties (such as modulus)
- How do you choose good values for your material?
- Dan Morris has been working on this, attempting to learn the best values for the parameters of Teschner's model (k_D , k_S , k_V) by running an optimization algorithm (simulated annealing) with the function to minimize being the sum of the squares of the distances of the vertices of the mesh at equilibrium between an FEM simulation and Teschner's simulation using given these parameter values for a given applied force (or set of forces)