Flexible

SOFA

May 16, 2012

Abstract

1 Overview

In this plugin, our goal is to separate each stage of the simulation into individual components to improve the modularity of the simulator.
The numerical simulation of continuous deformable objects is based on a discrete number of independent degrees of freedom (DOFs) which we will call the nodes. They are kinematic primitives (can be points, frames, etc.).
Nodes are associated with shape functions which are combined to produce the displacement function of material points in the solid.
We introduce the following notations:

- $\bar{\mathbf{q}}$, $\mathbf{q}$ and $\mathbf{f}$: the initial positions, current positions, and forces of the nodes.
- $\mathbf{\Theta}$: the material coordinates of a point according the chosen parameterization of the solid.
- $\bar{\mathbf{p}}(\mathbf{\Theta})$, $\mathbf{p}(\mathbf{\Theta})$: the initial and current position of a point in space
- $\mathbf{u}(\mathbf{\Theta}) = \mathbf{p} - \bar{\mathbf{p}}$: the displacement of a point
- $w_i(\mathbf{\Theta})$: the shape function associated with node i

The local deformation is computed by differentiation with respect to material coordinates. The deformation gradient is: $F = \partial\mathbf{p}/\partial\mathbf{\Theta}$.
The elastic deformation is described using a strain measure based on the deformation gradient.
These three stages can be modeled using SOFA mappings:

$$\begin{tikzcd} \text{Nodes} \arrow[r, shift left=3, "\mathcal{J}"] \arrow[r, "\mathbf{J}"] & \text{Deformation gradients} \arrow[l, shift left=3, "\mathbf{J}^T"] \arrow[r, shift left=3, "\mathcal{J}"] \arrow[r, "\mathbf{J}"] & \text{Strains} \arrow[l, shift left=3, "\mathbf{J}^T"] \end{tikzcd} \tag{1}$$

The elastic potential energy is computed from the strain.
After spatial integration (quadrature), we obtain associated forces (total stress), that can be back propagated to the nodes using transposed jacobians.

$$\mathbf{f} = -\frac{\partial \mathcal{W}^T}{\partial \mathbf{q}} = -\mathbf{J}_0^T \mathbf{J}_1^T \int_{\mathcal{V}} \sigma \tag{2}$$

Force variations are updated at each mapping by combining material and geometric stiffnesses. For a mapping from p to c, we have:

$$\delta(\mathbf{f}_p) = (\mathbf{J}^T \mathbf{K}_c \mathbf{J} + \frac{\partial \mathbf{J}^T}{\partial \mathbf{q}_p} \mathbf{f}_c)\delta(\mathbf{q}_p) \tag{3}$$

2 Scene graph

- **State** = nodes
- Shape function
- **ELASTICITY:**
  - Gauss point sampler
  - **State** = deformation gradients
  - **Mapping** = deformation mapping
  - **MATERIAL:**
    - **State** = strains
    - **Force field**
    - **Mapping** = strain mapping
- **MASS:**
  - **State** = points
  - Mass
  - **Mapping** = deformation mapping
- **COLLISION:**
  - **State** = points
  - **Mapping** = deformation mapping
- **VISU:**
  - **State** = points
  - **Mapping** = deformation mapping

3 Shape functions

3.1 Shepard

Shepard shape functions correspond to inverse distance weights
(`http://en.wikipedia.org/wiki/Inverse_distance_weighting`).
They are defined as $w_i(\mathbf{\Theta}) = 1/||\mathbf{\Theta} - \mathbf{\Theta}_i||^p$ followed by normalization.

3.2 Barycentric

Barycentric shape functions are the barycentric coordinates of points inside cells (can be edges, triangles, quads, tetrahedra, hexahedra). They achieve first order consistency: $\mathbf{\Theta} = \sum w_i \mathbf{\Theta}_i$

3.3 Natural Neighbors

Natural neighbor interpolants are based on Voronoi diagrams. Currently, Voronoi diagrams are computed from an image (a rasterized object).

3.4 to do

- higher order FEM
- clarify material vs. spatial coordinates
-

4 Deformation mapping

4.1 linear mapping

Child positions are computed as a linear combination of parent node dofs. For instance, the mapping from points to points is : $\mathbf{p} = \sum_i w_i(\mathbf{q} - \bar{\mathbf{q}})$. The mapping from affine frames to points in homogeneous coordinates is : $\mathbf{p} = \sum_i w_i \mathbf{q}\bar{\mathbf{q}}^{-1}\bar{\mathbf{p}}$.

4.2 Extension mapping

4.3 Distance mapping

4.4 Log rigid mapping

4.5 Relative rigid mapping

4.6 Triangle deformation mapping

4.7 to do

- Moving Least squares
- non-linear skinning
- clarify material vs. spatial coordinates
- model plasticity/control using relative mappings
-

5 Strain mapping

5.1 Green-Lagrangian strain

The strain is mapped from the deformation gradient as : $\mathbf{E} = (F^T F - \mathbf{I})/2$. Here the strain is stored into vectors using Voigt notation. In 3d, we have: $\epsilon = [\epsilon_{xx}, \epsilon_{yy}, \epsilon_{zz}, 2\epsilon_{xy}, 2\epsilon_{yx}, 2\epsilon_{xz}]$ The energy conjugate SPK stress vector is $\sigma = [\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{yx}, \sigma_{xz}]$

5.2 Corotational strain

The rigid displacement $\mathbf{R}$ is first extracted from the deformation gradient using, for instance, polar or QR decomposition. Then, supposing that the non-rigid deformation $\mathbf{R}^T F$ is small enough, we can apply the Cauchy strain formulation: $\mathbf{E} = [\mathbf{R}^T F + F^T \mathbf{R})/2 - \mathbf{I}$.

5.3 Invariants of deformation tensor

The elastic energy of some materials are expressed using the three invariants of the right Cauchy deformation tensor $\mathbf{C} = F^T F$:

- $I1(\mathbf{C}) = trace(\mathbf{C})$
- $I2(\mathbf{C}) = (trace(\mathbf{C}^2) + trace(\mathbf{C})^2)/2$
- $I3(\mathbf{C}) = det(\mathbf{C})$

In practice, deviatoric invariants are used:

- $\tilde{I1}(\mathbf{C}) = I1(\mathbf{C})/det(F)^{2/3}$
- $\tilde{I2}(\mathbf{C}) = I2(\mathbf{C})/det(F)^{4/3}$

Invariants are homogeneous with energies, so we use their squared roots as the state vectors.

5.4 to do

- fix undefined invariants for inverted/flat elements
-
-

6 Materials

6.1 Hooke Force field

Hooke materials have linear strain/stress relationships: $\sigma = \mathbf{H}\epsilon$. The potential energy is $W = \int_{\mathcal{V}} \epsilon^T \mathbf{H} \epsilon / 2$.

6.2 Mooney Rivlin

The potential energy is $W = \int_{\mathcal{V}} [C1(I1 - 3) + C2(I2 - 3)]$, where $C1$ and $C2$ are material constants.

6.3 Volume preservation

Possible energy formulations for volume conservation are :

- $W = \frac{k}{2} \int_{\mathcal{V}} log(det(F))^2$
- $W = \frac{k}{2} \int_{\mathcal{V}} (det(F) - 1)^2$

where k is the bulk modulus.

6.4 to do

- merge with implemented fem materials (Costa, Arruda-Boyce, NeoHookean, Veronda)
-
-

7 Quadrature

Quadrature points are sampled using one of the GaussPointSampler component. Currently, samplers can take meshes or images as inputs.
Quadrature methods estimate integrals using a sum of weighted evaluations at sample positions: $\int_{\mathcal{V}} f(\bar{\mathbf{p}})d\mathcal{V} \approx \sum_i v_i f(\bar{\mathbf{p}}_i)$

7.1 Mid-point

The simplest method is to take one point per region and weight the value by its volume. This is exact only for constant functions (e.g., elastic energy in a first order tetrahedral FEM)

7.2 Gauss-Legendre

Several points are used to approximate the intergral of higher order functions. Currently only first order Gauss-Legendre quadrature on hexahedra is implemented.

7.3 Elastons

The idea is to decompose f on a basis: $f(\bar{\mathbf{p}}) = \mathbf{c}(\bar{\mathbf{p}}_0)(\bar{\mathbf{p}} - \bar{\mathbf{p}}_0)^*$, where $\mathbf{c}$ are the coeficients and $()^*$ the basis vector (for instance the first order polynomial basis $[1, x, y, z]$). The integral is then estimated as $\int_{\mathcal{V}} f(\bar{\mathbf{p}})d\mathcal{V} \approx \mathbf{c}(\bar{\mathbf{p}}_0) \int_{\mathcal{V}} (\bar{\mathbf{p}} - \bar{\mathbf{p}}_0)^* d\mathcal{V} = \sum_i v_i c_i(\bar{\mathbf{p}}_0)$ where the coeeficient v_i part can be precomputed using an arbitrary fine discretization.

7.4 to do

- Newton Cotes
- Finish implementation of elastons. Required instanciations for all force fields..
-

8 Requirements

SOFA Packages: The following must be enabled in sofa-local.prf

- Any special things that need to be enabled

SOFA Plugins: The following must be loaded in your SOFA instance

- Any plugins that must be loaded

9 Scene Settings

9.1 Required Settings

RequiredExampleSetting
A description of what this setting controls, and any special information the user should know about it. Required settings are those that need to be specified by the user in order for the component to function. If your component doesn't have any required settings, leave this section blank. The below "value type" is the type that is expected by the component. "Aliases" are other strings that can be specified in the scene file for this setting. These are defined in the code using "addAlias()".
Value Type - **string**
Aliases - requiredexamplesetting

9.2 Optional Settings

OptionalExampleSetting
A description of what this setting controls, and any special information the user should know about it. Optional settings can be specified by the user in order to change the default behaviour of the component. The below "Default Value" is the value given to the setting if the user doesn't specify anything.
Value Type - **bool**
Default Value - false
Aliases - optionalexample, optionalsetting

10 Scene Data

10.1 Required Data

RequiredExampleData
Data is usually a link to something in another component. Required data must be specified in order for the component to function.
Value Type - **ExampleType**

10.2 Optional Data

OptionalExampleData
Data is usually a link to something in another component.
Value Type - **ExampleType**
Aliases - OptData

10.3 Output data

Output data is generally not defined in the component, but is linked to by other components.

10.4 Example File

path/to/an/example/file.scn