Shape Matching Element Method

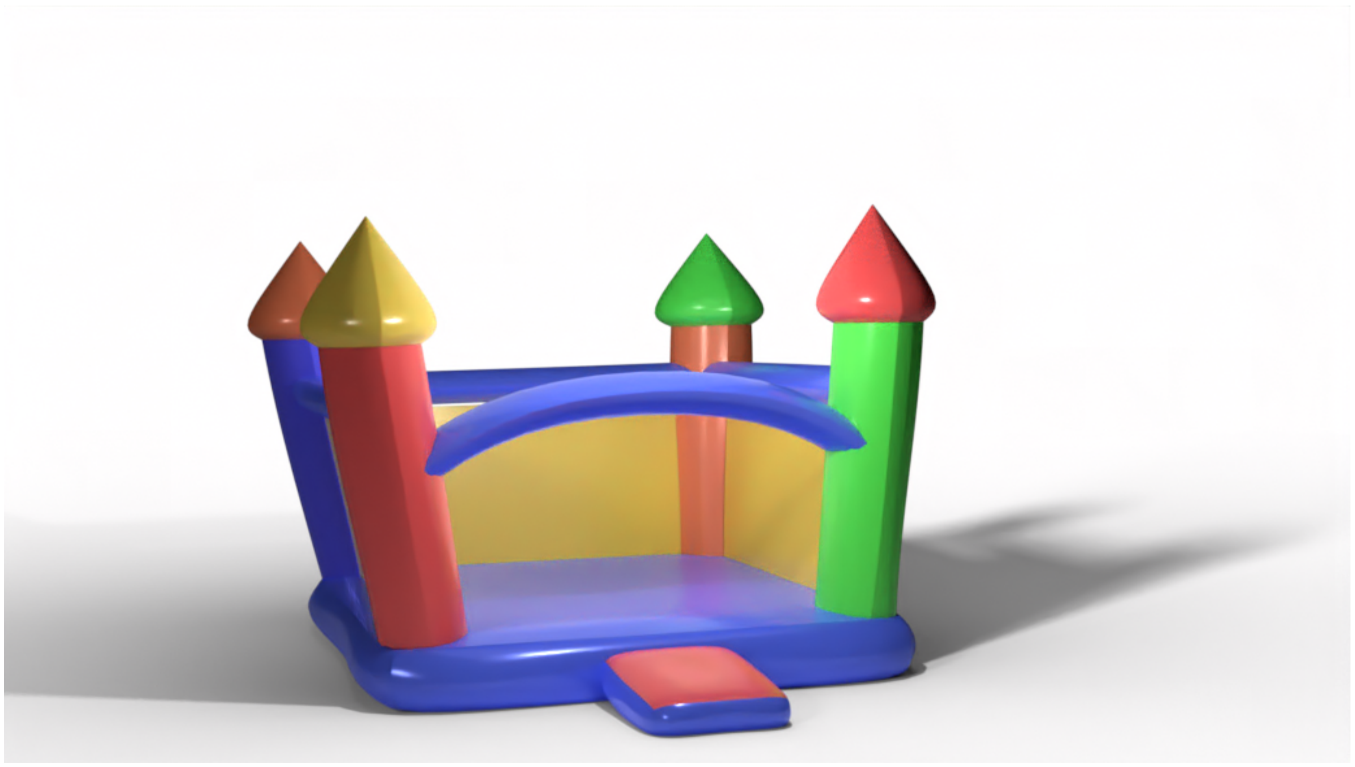

Figure 1: tmp

ACM Reference Format:
. 2020. Shape Matching Element Method. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

We have $\mathbf{x} = (x_0, x_1)$ and $\mathbf{X} = (X_0, X_1)$ as deformed and undeformed positions, respectively.

<Exposition about FEM >

In our case, the function we are trying to evaluate is deformation with respect to the center of mass. In VEM, instead of reconstructing values of our function using nodal values, we seek to find a a set of polynomial coefficients that map some arbitrary undeformed position to a polynomial approximation of its deformed position. This set of polynomial coefficients is referred to as the projection operator, $\mathbf{\Pi}$. The key point here is that $\mathbf{\Pi}$ is constructed strictly using positions defined on the boundary, avoiding the need for explicit representation of the interior.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Shape Matching:

In shape matching, authors seek a polynomial transformation, $\mathbf{\Pi}$ that minimizes $\sum_i ||\mathbf{\Pi}\mathbf{M}(\mathbf{X}) - \mathbf{p_i}||^2$ where $\mathbf{M}(\mathbf{X})$ is the monomial basis of the i-th boundary vertex and $\mathbf{p_i}$ is current displacement from the center of mass for the i-th point. We argue this paper can be interpreted as a sort of virtual element method.

Reconstructing deformed positions

A general form for the monomial basis takes the following form <insert here>. For example, in 2D with quadratic deformation we have:

$$\mathbf{M}(\mathbf{X}) = \begin{bmatrix} X_0 - \bar{X}_0 \\ X_1 - \bar{X}_1 \\ (X_0 - \bar{X}_0)^2 \\ (X_1 - \bar{X}_1)^2 \\ (X_0 - \bar{X}_0)(X_1 - \bar{X}_1) \end{bmatrix} \tag{1}$$

We write the estimated deformed position $\mathbf{x}$ of some arbitrary undeformed position $\mathbf{X}$ as

$$\mathbf{x}(\mathbf{X}) = \hat{\mathbf{\Pi}}\mathbf{M}(\mathbf{X}) + \bar{\mathbf{x}} \quad (2)$$

where $\hat{\mathbf{\Pi}}$ is the blended projection operator for this particular point:

$$\hat{\mathbf{\Pi}}(\mathbf{X}) = \sum_{i}^{|S|} w_i(\mathbf{X})\mathbf{\Pi}_i \quad (3)$$

The bar symbol indicates the center of mass. For example, $\bar{\mathbf{x}}$ is the deformed object's center of mass, which we compute as the mean of the boundary values

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i}^{n} x_i \quad (4)$$

Here w_i is some weighting function for the i-th shape. There are many possible forms for this weighting function. Right now we compute these weights by forming the following optimization: <insert here> In order to form the equations of motion, we need to re-express the above equation in terms of the nodal values, $\mathbf{q}$. To arrive it these we note that each $\mathbf{\Pi}_i$ is a function of $\mathbf{q}$.

$$\mathbf{\Pi}_i(\mathbf{q}) = \mathbf{P}(\mathbf{E_i q})\mathbf{M}(\mathbf{E_i q_0})^{\dagger} \quad (5)$$

I'm not a fan of what I wrote here. The MLS-MPM paper presents a similar idea when talking about EFG, and I like their notation a bit better This is the shape matching problem's least squares solution where $M(q_0)^{\dagger}$ is the Moore-Penrose pseudo-inverse of the monomial vectors stacked row-wise (add example equation). The matrix $\mathbf{E}_i$ selects the degrees of freedom from $\mathbf{q}$ associated with the i-th shape. The function $\mathbf{P}(\mathbf{q})$ evaluates the displacement of these values from the current center of mass. We observe that the above equation is linear in $\mathbf{q}$ so we may rewrite the equation for $\mathbf{x}$ as

$$\mathbf{x}(\mathbf{X}) = \mathbf{J}_{\hat{\mathbf{\Pi}}}(\mathbf{X})\mathbf{q} \quad (6)$$

Simulation on NURBs models We used the D-NURBS model (ref) for simulating on NURBs surfaces, giving an approach to simulating directly on the degrees of freedom of parametric NURBs models. The equation for a NURBs curve takes the form:

$$\mathbf{x}(u) = \frac{\sum_{i=1}^{n} \mathbf{p}_i w_i B_{i,k}(u)}{w_i B_{i,k}(u)} \quad (7)$$

where u is the parametric coordinate, $\mathbf{p}_i$ is the i-th control point with n total control points. $B_{i,k}(u)$ is the B-spline basis function for the $i-th$ control point with degree $k-1$. NURBs curves represent a generalization of B-splines with the addition of the weight w_i for each control point. In our case, we primarily work with NURBs surfaces, which are a generalization of the tensor-product of two B-splines:

$$\mathbf{x}(u,v) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{p}_{i,j} w_{i,j} B_{i,k}(u) B_{j,l}(v)}{w_{i,j} B_{i,k}(u) B_{j,l}(v)} \quad (8)$$

where we now have two parameters u, v, and $m \cdot n$ control points. In D-NURBs, we see that $\mathbf{x}(u,v)$ is linear in the control points, $\mathbf{p}$, so we can rewrite the above equation as

$$\mathbf{x}(u,v,\mathbf{p}) = \mathbf{J}(u,v)\mathbf{p}. \quad (9)$$

Here $\mathbf{J}$ is the Jacobian matrix that maps the control points to some world space position on the surface for some u, v pair. <elaborate on the form of J>. Now with these generalized coordinates, we may perform dynamics directly on the degrees of freedom of our NURBs models.

Generalized Coordinates
With D-NURBs simulation layered on the VEM simulation, we have two Jacobians: $\mathbf{J}(u,v)$ mapping controls points, $\mathbf{p}$ onto surface positions, and then $\mathbf{J}_{\hat{\mathbf{\Pi}}}(\mathbf{X})$ which maps surface positions $\mathbf{q}$ to their deformed positions. The Jacobian $\mathbf{J}(u,v)$ may be precomputed, and each point sampled on a surface of a NURBs has an associated $\mathbf{J}(u,v)$. With these two Jacobians we may express a single u, v pair's deformed position as

$$\mathbf{x}(u,v,\mathbf{p}) = \mathbf{J}_{\hat{\mathbf{\Pi}}}(\mathbf{X})\mathbf{J}(u,v)\mathbf{p} \quad (10)$$

and the undeformed position

$$\mathbf{X}(u,v) = \mathbf{J}(u,v)\mathbf{p_0} \quad (11)$$

where $\mathbf{p_0}$ is the initial values for the control points.

Deformation Gradient

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \frac{\partial \hat{\mathbf{\Pi}}\mathbf{M}(\mathbf{X})}{\partial \mathbf{X}} = \hat{\mathbf{\Pi}}\frac{\partial \mathbf{M}(\mathbf{X})}{\partial \mathbf{X}} \quad (12)$$

The $\frac{\partial \mathbf{M}(\mathbf{X})}{\partial \mathbf{X}}$ terms may be precomputed and using the previous $\mathbf{M}(\mathbf{X})$ example, we have

$$\frac{\partial \mathbf{M}(\mathbf{X})}{\partial \mathbf{X}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2(X_0 - \bar{X}_0) & 0 \\ 0 & 2(X_1 - \bar{X}_1) \\ (X_1 - \bar{X}_1) & (X_0 - \bar{X}_0) \end{bmatrix} \quad (13)$$

In computing the forces and the stiffness matrix, we require the gradient of the deformation gradient with respect to the configuration, $\frac{\partial \mathbf{F}}{\partial \mathbf{q}}$. Usually the form of this gradient is simple. For example, in linear tetrahedral FEM each quadrature point's deformation map only involves the four surrounding nodal values, simplifying this gradient. In our case, the presence of the projection operator $\hat{\mathbf{\Pi}}$ means that each quadrature point is computed using the nodal values of the entire model.

$$\frac{\partial \mathbf{F}}{\partial \mathbf{q}} = \frac{\partial \mathbf{P}}{\partial q}\mathbf{M}(\mathbf{q_0})^{\dagger}\frac{\partial \mathbf{M}(\mathbf{X})}{\partial \mathbf{X}} \quad (14)$$

Naively handled, this gives us a $\frac{\partial \mathbf{F}}{\partial \mathbf{q}}$ of size $3x|q|$. However, many of the columns of have values all near zero, allowing us to prune many columns for each quadrature point. Currently, I'm computing the infinity norm of each column and prune all except some fixed number of columns.

Variational Mechanics
We have now provided all the necessary ingredient to produce equations of motion for deformable bodies, which we derive via Euler-Lagrange equation. We begin with the Lagrangian

$$L = T - V \quad (15)$$

where T is the kinetic energy, and V is the potential energy. From the Euler-Lagrange equation we have:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{p}}} + \frac{\partial L}{\partial \mathbf{p}} = 0 \quad (16)$$

$$V = \int_{\Omega} \psi(\mathbf{F}(\mathbf{X})dX \quad (17)$$

then discretizing this over m quadrature points, we have

$$V = \sum_{i}^{m} \psi(\mathbf{F}(\mathbf{X_i}) A_i \tag{18}$$

$$\frac{\partial L}{\partial \mathbf{q}} = \sum_{i}^{m} \frac{\partial V_i}{\partial \mathbf{q}} = \sum_{i}^{m} \frac{\partial \psi(\mathbf{F_i})}{\partial \mathbf{F}} \frac{\partial \mathbf{F_i}}{\partial \mathbf{q}} A_i \tag{19}$$

Next we have the equation of the kinetic energy

$$T = \frac{1}{2} \int_{\Omega} \rho \dot{\mathbf{x}}^T \dot{\mathbf{x}} dV \tag{20}$$

which we rewrite in terms of our control points configuration

$$T = \frac{1}{2} \int_{\partial\Omega} \dot{\mathbf{p}}^T \mathbf{J}(u,v)^T \mathbf{M} \mathbf{J}(u,v) \dot{\mathbf{p}} du dv \tag{21}$$

change notation to not use M for the monomial basis where

$$\mathbf{M} = \int_{\Omega} \rho \mathbf{J}_{\hat{\mathbf{\Pi}}}(\mathbf{X})^T \mathbf{J}_{\hat{\mathbf{\Pi}}}(\mathbf{X}) \tag{22}$$

From the potential energy (which uses dF/dq) and this potential energy, we discretize in time and right now I'm using linearly implicit backward euler

Stability term

Dump of some equations

$$\mathbf{M}(\mathbf{X}) = (X_0 - \bar{X}_0, X_1 - \bar{X}_1)$$

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i}^{n} x_i$$

$$\bar{\mathbf{X}} = \frac{1}{n} \sum_{i}^{n} X_i$$