Shape Matching Element Method

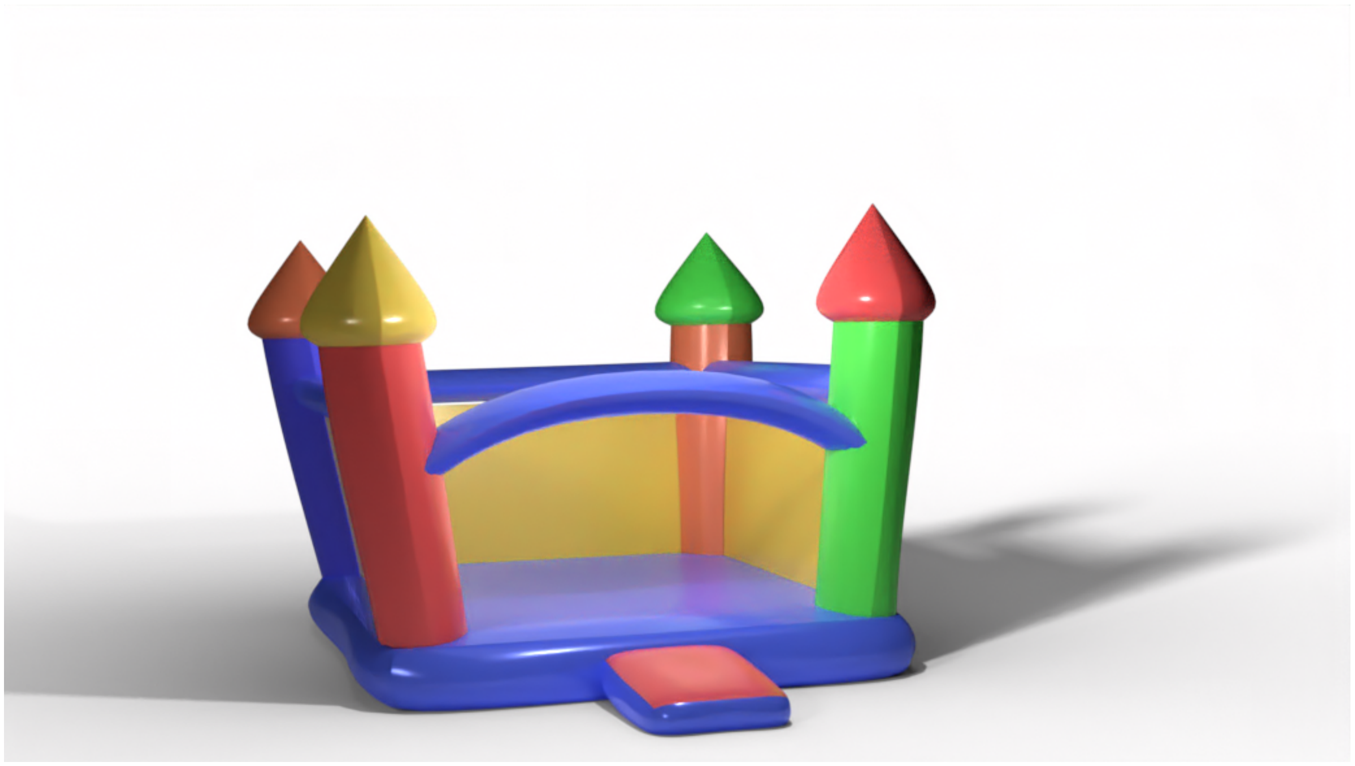

Figure 1: tmp

ACM Reference Format:
. 2020. Shape Matching Element Method. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 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.

<insert 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.

1.1 Shape Matching:

In shape matching, authors seek a polynomial transformation, $\mathbf{\Pi}$ that minimizes $\sum_i ||\mathbf{\Pi M(X)} - \mathbf{p_i}||^2$ where $\mathbf{M(X)}$ 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.

1.2 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

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

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

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

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

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

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 $\mathbf{q}$.

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

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

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

1.3 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:

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

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:

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

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

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

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.

1.4 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

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

and the undeformed position

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

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

1.5 Deformation Gradient

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

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

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

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.

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

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.

1.6 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

$$L = T - V \tag{15}$$

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

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

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

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

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

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

1.7 Stability term

1.8 Projection Operator Weighting

$$\begin{aligned} \mathbf{w}(\mathbf{X}) = \min_{\mathbf{w}} \quad & \mathbf{w}^T \Theta(\mathbf{X}) \mathbf{w} \\ \text{s.t.} \quad & 0 \leq w_i \leq 1 \\ & \sum_{i}^{n} w_i = 1 \end{aligned} \tag{23}$$

where

$$\Theta(\mathbf{X}) = \operatorname{diag}\left(\frac{1}{\theta_1}, \frac{1}{\theta_2}, \ldots, \frac{1}{\theta_N}\right) \tag{24}$$

2 RELATED WORK

2.1 Shape Matching Related Papers

(1) *Shape Matching* [?]: Meshless simulation by fitting a polynomial describing the shapes deformation using only the nodal values.
(2) *Lattice Shape Matching* [?]: Voxelize model to construct a lattice of cubes. Use these lattice cubes to construct overlapping shape matching regions (just like clustered shape matching?). The original mesh is deformed using trilinear interpolation of lattice vertex positions.
(3) *Robust Real-Time Deformation of Incompressible Surface Meshes* [?]: Shape matching on trimeshes with overlapping regions (clustered shape matching). Adds an additional volume preservation constraint. Position based dynamics approach to satisfying volume preservation.
(4) *Shape-Up: Shaping Discrete Geometry with Projections* [?]: Shape constraints by least squares fitting (like in shape matching). They have some "proximity function" indicating distance to least-squares fit, then uses projection operators to minimize proximity function (pretty much just shape matching).
(5) *Shape Matching with Oriented Particles* [?]: More general form for shape matching, permitting wider range of motion. Also they use shape matching projection operators for skinning, much like we do. For each skinning point, they specify weights with up to 4 particles (each with their own projection operator)
(6) *Fast Adaptive Shape Matching Deformations* [?]: Essentially same thing as Lattice Shape Matching, but instead they use an octree instead of a basic voxel grid for shape matching. It's not super significant to mention this paper, but it does make clear that much of the followup work after shape matching never didn't emphasize it's utility as a meshless boundary only method. They kept converting it to a mesh-based method!
(7) *A Geometric Deformation Model for Stable Cloth Simulation* [?] Shape matching for cloth simulation.

2.2 Other Meshless Methods in Graphics

(1) *Point Based Animation* [?]: Purely particle based, MLS to approximate derivatives. Appears to be among the earliest meshless methods in graphics based on continuum mechanics.
(2) *Position Based Dynamics* [?]: Operates directly on particle positions by forming set of constraints and solving for particle positions that satisfy these constraints. Meshless
(3) *Projective Dynamics* [?]: Similar to position based dynamics but solves the constraints implicitly by minimizing energy potentials. Mesh-based. Global solver unlike PBD which satisfies constraints locally using Gauss-Seidel.

2.3 Virtual Element and other Element Methods

(1) *Mimetic Finite Differences* [?] [?] (they double dipped!): Considered a close relative to VEM and framed as the predecessor to VEM (in VEM papers). I still haven't read on MFD yet. From PolyDDF: "extension of finite volume and finite difference techniques to polygons that first discretizes a prime operator (typically, the gradient or the divergence) via a boundary integral, and then derives other operators by mimicking continuous structural properties."
(2) *Basic principles of Virtual Element Method* [?] Original VEM paper
(3) *The Hitchhiker's Guide to Virtual Element Method* [?]: More understandable versions of original VEM paper.

(4) *Discrete Differential Operators on Polygonal Meshes* [?]: Extends VEM to do discrete differential geometry on arbitrary polygonal meshes.
(5) *FLexible Simulation of Deformable Models using Discontinuous Galerkin FEM* [?]: Uses ordinary hexahedral elements, but uses a cut cell-based approach to support arbitrary polyhedra on the surface.
(6) *Generalizing the finite element method: Diffuse approximation and diffuse elements* [?] Predecessor to Element Free Galerkin. FEM interpolation replaced with a local Moving Least Square interpolation.
(7) *Element-free Galerkin methods* [?]: similar to DEM, but more accurate gradients (not sure of all the differences). In MLS methods, they solve least squares for each particle in the domain, weighting nearby particles with a Gaussian-like density function. In contrast to us, we only compute least squares fitting on the boundary, and then precompute some weighting for each particle to the projection operators on the boundaries.
(8) *Unified Simulation of Elastic Rods, Shells, and Solids* [?]: Propose Generalized moving least squares (GMLS) to resolve limitation of MLS shape functions that require many particles in the support of a point (that are not coplonar).

2.4 Physics based Skinning

(1) *Skinning Siggraph Course* [?]: The linear weighting of polynomials at the exterior is very similar to skinning.
(2) *Linear Subspace Design for Real-Time Shape Deformation* [?]: Linear deformation subspace that uses linear blend skinning and generalized barycentric coordinates. Similarly, we have "handles" but they are represented by entire NURBS patches, and our coordinates are the output of a polynomial, whereas barycentric coordinates are linear (I only skimmed this paper, not sure if this description is fair).
(3) *Complementary dynamics* [?]: Physics based skinning, orthogonality constraint can be seen as similar to our stability term (conformity term, error term, whatever it's called :))
(4) *Physically-Based Character Skinning* [?]: Linear blend skinning with multiple layers of skin simulated via oriented particle shape matching and position based dynamics to enforce distance constraints (avoiding unwanted intersections).

2.5 Isogeometric Analysis

(1) *Isogeometric Analysis Book* [?]: The book everyone references when they write Isogeometric analysis in their papers.
(2) *Dynamic NURBS* [?]: Outline of how to simulate on NURBS with the control points as the degrees of freedom. Method used in our work.
(3) *XCAD* [?]: Optimize CAD models. CAD embedded in hexahedral mesh, complex integration strategy. Uncut hexahedral elements simulated ordinarily, cut elements use XFEM that add additional DOF to account for new element shapes.
(4) *Development of a quadratic finite element formulation based on the XFEM and NURBS* [?]: XFEM to handle curve surface integration of NURBs patches. Complex subdividing of "X-Elements" to produce cut cells along NURBS surfaces.
(5) *A NURBS enhanced extended finite element approach for unfitted CAD analysis* [?]: Pretty much the same as the quadratic, but allows higher-order approximation and better handling of interface.
(6) *A NURBS-based interface-enriched generalized finite element method for problems with complex discontinuous gradient fields* [?]: Similar to other XFEM NURBS approaches. Uses NURBS-based enrichment functions with cut cells. Additional DOFs added to handle discontinuities.
(7) *A NURBS-based generalized finite element scheme for 3D simulation of heterogeneous materials* [?]: Similar to previous "NIGFEM" paper above, but now in 3D.
(8) *Swept Volume Parameterization for Isogeometric* [?] To provide volumetric simulation of NURBS they introduce a new NURBS volume parameterization (B-Spline Volumes ... jesus christ)
(9) *A finite volume method on NURBS geometries and its application in isogeometric fluid–structure interaction* [?]: Combines NURBS paramaterization with finite volume method (requiring a mesh for the volume).
(10) *NURBS-Enhanced Finite Element Method (NEFEM)* [?]: Similar to the above example. They run an order FVM simulation and deal with the interface in a complicated manner (could this be considered XFEM?).

2.6 Quadrature

(1) *A new method for meshless integration in 2D and 3D Galerkin meshfree methods* [?]: Strategy we use for integrating over CAD model volumes. Raycast along single dimension, find intersections, generate quadrature points in the intervals inside the object.
(2) Adaptive image-based intersection [?]: related to our meshless integration strategy in that we could use this to account for errors in the above approach (increase ray density where we estimate high error to be)
(3) Efficient and accurate numerical quadrature for immersed boundary methods [?]: Finite Cell Method. "Immerses" a shape in a set of cells (mesh!) and computes quadrature over this. To handle curved surface they use an octree to subdivide to the desired level of accuracy.
(4) Higher-Order Finite Elements for Embedded Simulation [?]: Another Finite Cell method like the above, but with a new quadrature generation method (the ones with the circles in the triangles)
(5) *Highly accurate surface and volume integration on implicit domains by means of moment-fitting* [?] and [?]: XCAD paper extends upon this method still need to read these

2.7 Misc

(1) FEM simulation of 3D deformable solids [?]
(2) Fusion 360 Gallery [?] : source of some models

Dump of some equations

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

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

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