Shape Matching Element Method

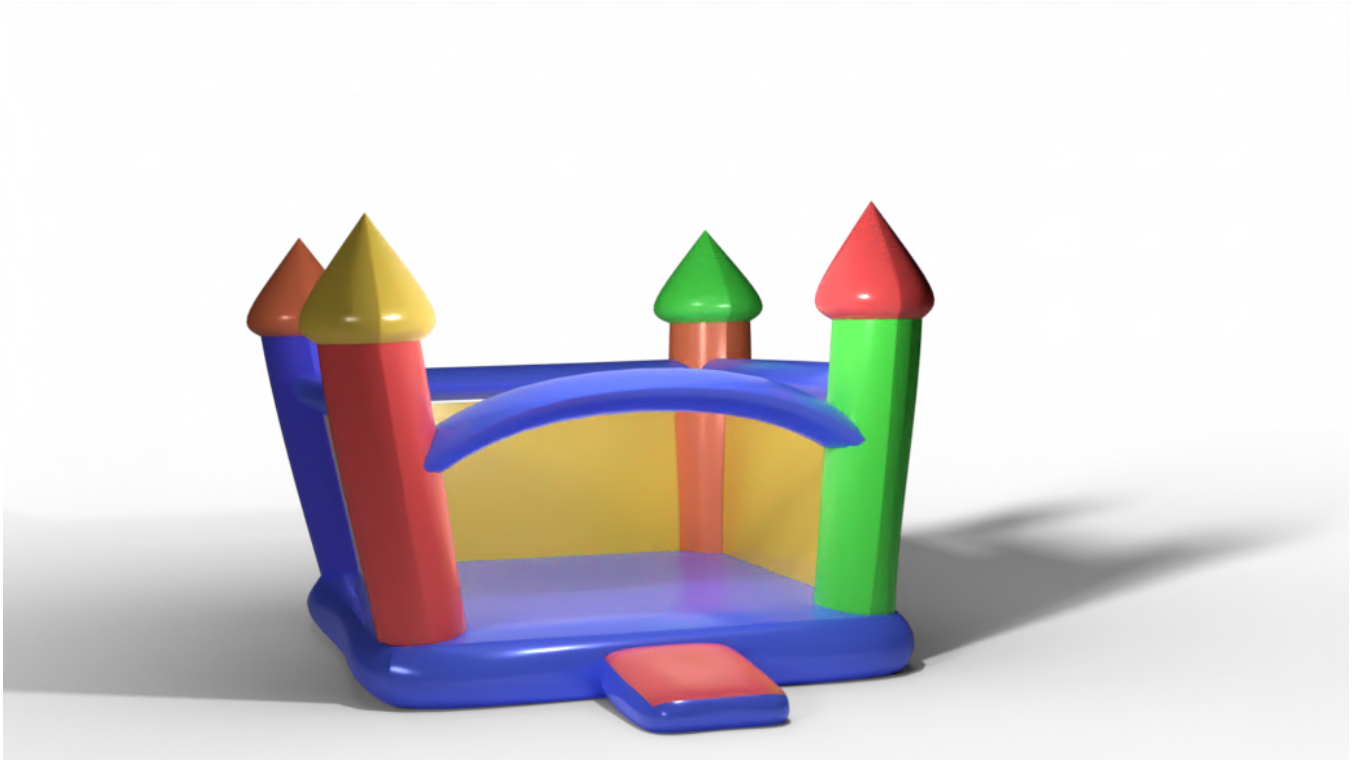

Figure 1: tmp

ACM Reference Format:
. 2020. Shape Matching Element Method. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A pointform description of the general idea

- A D-NURBS approach to simulation. Use a Jacobian to map from UV positions in a NURBS parametric space to world space positions on the surface.
- **Shape Matching**: From the sample world space positions on a single NURBS surface, we solve a least squares problem to fit a *projection operator* to the surface. This projection operator maps monomials in the undeformed space, to the least squares estimate of the deformed positions for the given monomials.
- With DNURBS and Shape Matching VEM we have two mappings: a UV to undeformed world space with our NURBS Jacobian, then an undeformed to deformed mapping with the projection operators. This lets us form a set of generalized coordinates in terms of our control points of the NURBS surface (section 1.4).
- For an arbitrary position in space (not necessarily on the surface), we can build a projection operator unique to this point via a weighted sum of the projection operators of the projection operators for each NURBS (eq 5). From this we can reconstruct an estimate of the deformed position from the monomial basis defined using its undeformed position.
- For any point in space, we need to compute a set of weights to the projection operators defined on the surfaces with the following criteria: the weights sum to 1, are nonzero, and depend on the distances to the surfaces (See section 1.8).
- With this mapping for undeformed to deformed positions, this yields a simple definition for the deformation gradient (eq 12).
- Armed with our deformation gradient, we next define the Lagrangian, but this basic form isn't sufficient. We augment

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

our kinetic and potential energies with a stability term (ref VEM). This accounts for cases in which the position of the nodal values don't match their estimated positions defined by the projection operator mapping.
- **Raycasting Quadrature**: To compute the volume integrals we make use of raycasting quadrature approach (ref) where we uniform sample a YZ grid and shoot rays in the X direction. We then use the points of intersection to define integrations bounds for 1D integrals along these rays, which we integrate numerically.
- With our augmented lagrangian, we use the Euler-Lagrange equation to arrive at equations of motion. Our generalized coordinates give us direct updates on the control points of the NURBS. Currently we are using linearly-implicit Euler for time integration.
- **Handling trimmed NURBS**: For this case we update the original control points, but we must only sample points within the boundary. To compute these points we use an approach similar to the raycasting quadrature but in 2D. In the paremtric domain for a T-NURBS we sample uniforly in the V direction and shoot rays along the U axis. At each region of intersection, we sample more values along the ray. The final result is a set of UV coordinates that respect the boundary defined by NURBS curves.

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 $\sum_i ||\mathbf{\Pi M}(\mathbf{X}) - \mathbf{p_i}||^2$ 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.

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

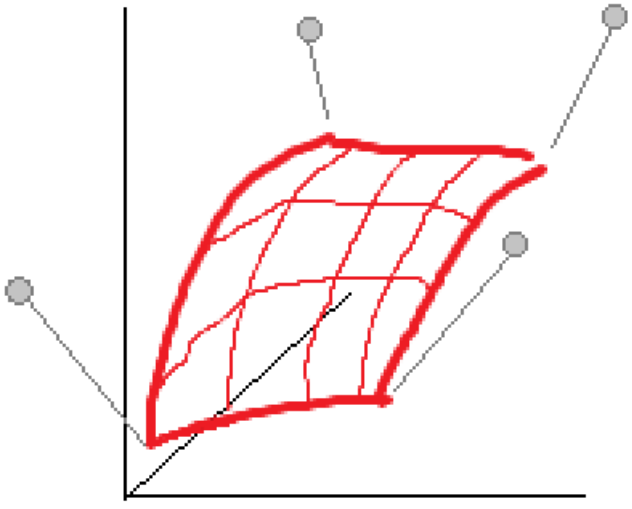

Figure 2: The NURBs model uses control points, shown in grey, as the original degrees of freedom.

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 $\mathbf{J}_{\hat{\Pi}}(\mathbf{X})$ 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

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

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

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

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

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

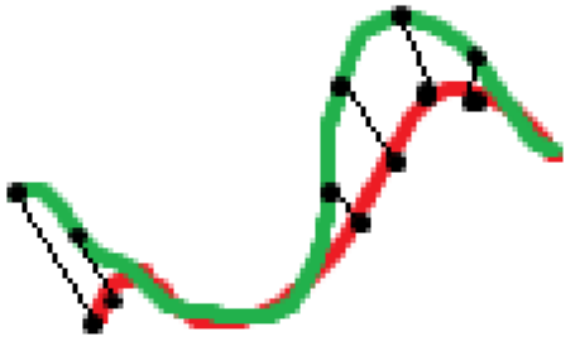

Figure 3: A figure showing the spring energy error for each sampled point on the NURBS curve. I should change the colors of the lines to show higher energies where the error is higher.

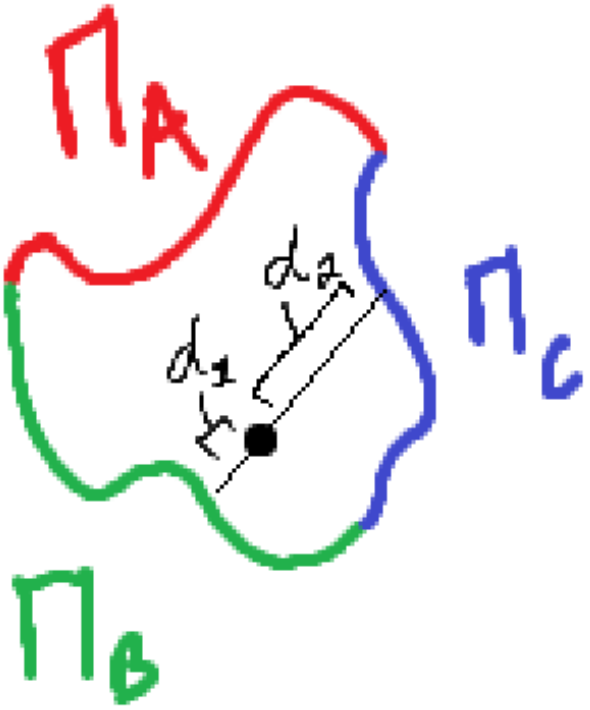

Figure 4: A 2D example showing how θ is computed for Π_B. d_1 shows the distance to the closest point on Π_B to x and d_2 shows the distance to the surface hit by the ray in the opposite direction.

where

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

In computing the theta values, we require that they a approach 1 at the target surface and fall off to zero as a point becomes closer to other surfaces. To compute this for some surface, we shoot a ray from the point to the nearest point on the surface. Then in the opposite direction we find the nearest intersection to any other surfaces. We compute the theta with the following linear function

$$theta = \max(1.0 - \frac{d_1}{\min(d_2, D)}, 0.0) \tag{25}$$

where D is a fixed cut-off distance.

1.9 Integrating the volume

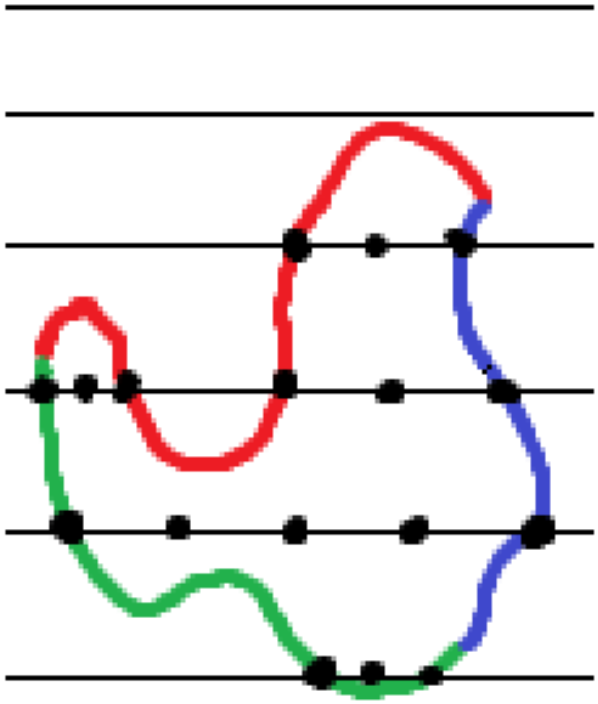

Figure 5: An example raycasting quadrature on a 2D NURBS curve.

2 RELATED WORK

2.1 Shape Matching Related Papers

(1) *Shape Matching* [Müller et al. 2005]: Meshless simulation by fitting a polynomial describing the shapes deformation using only the nodal values.
(2) *Lattice Shape Matching* [Rivers and James 2007]: 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* [Diziol et al. 2011]: 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* [Bouaziz et al. 2012]: 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* [Müller and Chentanez 2011]: 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* [Steinemann et al. 2008]: 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* [Stumpp et al. 2008] Shape matching for cloth simulation.

2.2 Other Meshless Methods in Graphics

(1) *Point Based Animation* [Müller et al. 2004]: 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* [Müller et al. 2007]: Operates directly on particle positions by forming set of constraints and solving for particle positions that satisfy these constraints. Meshless
(3) *Projective Dynamics* [Bouaziz et al. 2014]: 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* [Brezzi et al. 2005] [Lipnikov et al. 2014] (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* [Veiga et al. 2012] Original VEM paper
(3) *The Hitchhiker's Guide to Virtual Element Method* [Beirão da Veiga et al. 2014]: More understandable versions of original VEM paper.
(4) *Discrete Differential Operators on Polygonal Meshes* [De Goes et al. 2020]: Extends VEM to do discrete differential geometry on arbitrary polygonal meshes.
(5) *FLexible Simulation of Deformable Models using Discontinuous Galerkin FEM* [Kaufmann et al. 2008]: 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* [Nayroles et al. 1992] Predecessor to Element Free Galerkin. FEM interpolation replaced with a local Moving Least Square interpolation.
(7) *Element-free Galerkin methods* [Belytschko et al. 1994]: 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* [Martin et al. 2010]: 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* [Jacobson et al. 2014]: The linear weighting of polynomials at the exterior is very similar to skinning.
(2) *Linear Subspace Design for Real-Time Shape Deformation* [Wang et al. 2015]: 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* [Zhang et al. 2020]: 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* [Deul and Bender 2013]: 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* [Cottrell et al. 2009]: The book everyone references when they write Isogeometric analysis in their papers.

(2) *Dynamic NURBS* [Terzopoulos and Qin 1994]: Outline of how to simulate on NURBS with the control points as the degrees of freedom. Method used in our work.
(3) *XCAD* [Hafner et al. 2019]: 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* [Haasemann et al. 2011]: 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* [Legrain 2013]: 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* [Safdari et al. 2015]: 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* [Safdari et al. 2016]: Similar to previous "NIGFEM" paper above, but now in 3D.
(8) *Swept Volume Parameterization for Isogeometric* [Aigner et al. 2009] 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* [Heinrich et al. 2012]: Combines NURBS paramaterization with finite volume method (requiring a mesh for the volume).
(10) *NURBS-Enhanced Finite Element Method (NEFEM)* [Sevilla et al. 2008]: 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* [Khosravifard and Hematiyan 2010]: 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 [Wang et al. 2012]: 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 [Kudela et al. 2015]: 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 [Longva et al. 2020]: 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* [Müller et al. 2013] and [Müller et al. 2017]: XCAD paper extends upon this method still need to read these

2.7 Misc

(1) TRACKS: Toward Directable Thin Shells [Bergou et al. 2007]: Petrov-Galerkin test functions for weak-form constraints that handles artifacts due to pointwise constraints.
(2) FEM simulation of 3D deformable solids [Sifakis and Barbic 2012]
(3) Fusion 360 Gallery [Willis et al. 2020] : source of some models

REFERENCES

Martin Aigner, Christoph Heinrich, Bert Jüttler, Elisabeth Pilgerstorfer, Bernd Simeon, and Anh-Vu Vuong. 2009. Swept Volume Parameterization for Isogeometric Analysis. 19–44. https://doi.org/10.1007/978-3-642-03596-8_2

L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. 2014. The Hitchhiker's Guide to the Virtual Element Method. *Mathematical Models and Methods in Applied Sciences* 24, 08 (2014), 1541–1573. https://doi.org/10.1142/S021820251440003X arXiv:https://doi.org/10.1142/S021820251440003X

T. Belytschko, Y. Y. Lu, and L. Gu. 1994. Element-free Galerkin methods. *Internat. J. Numer. Methods Engrg.* 37, 2 (1994), 229–256. https://doi.org/10.1002/nme.1620370205 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620370205

Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. 2007. TRACKS: Toward Directable Thin Shells. *ACM Trans. Graph.* 26, 3 (July 2007), 50–es. https://doi.org/10.1145/1276377.1276439

Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. 2012. Shape-Up: Shaping Discrete Geometry with Projections. *Comput. Graph. Forum* 31, 5 (Aug. 2012), 1657–1667. https://doi.org/10.1111/j.1467-8659.2012.03171.x

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4, Article 154 (July 2014), 11 pages. https://doi.org/10.1145/2601097.2601116

Franco Brezzi, Konstantin Lipnikov, and Valeria Simoncini. 2005. A family of mimetic finite difference methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences* 15 (04 2005). https://doi.org/10.1142/S0218202505000832

J. Cottrell, Thomas Hughes, and Yuri Bazilevs. 2009. *Isogeometric Analysis: Toward integration of CAD and FEA.* https://doi.org/10.1002/9780470749081.ch7

Fernando De Goes, Andrew Butts, and Mathieu Desbrun. 2020. Discrete Differential Operators on Polygonal Meshes. *ACM Trans. Graph.* 39, 4, Article 110 (July 2020), 14 pages. https://doi.org/10.1145/3386569.3392389

Crispin Deul and Jan Bender. 2013. Physically-Based Character Skinning. https://doi.org/10.2312/PE.vriphys.vriphys13.025-034

R. Diziol, J. Bender, and D. Bayer. 2011. Robust Real-Time Deformation of Incompressible Surface Meshes. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vancouver, British Columbia, Canada) *(SCA '11)*. Association for Computing Machinery, New York, NY, USA, 237–246. https://doi.org/10.1145/2019406.2019438

G. Haasemann, M. Kästner, S. Prüger, and V. Ulbricht. 2011. Development of a quadratic finite element formulation based on the XFEM and NURBS. *Internat. J. Numer. Methods Engrg.* 86, 4-5 (2011), 598–617. https://doi.org/10.1002/nme.3120 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.3120

Christian Hafner, Christian Schumacher, Espen Knoop, Thomas Auzinger, Bernd Bickel, and Moritz Bächer. 2019. X-CAD: Optimizing CAD Models with Extended Finite Elements. *ACM Trans. Graph.* 38, 6, Article 157 (Nov. 2019), 15 pages. https://doi.org/10.1145/3355089.3356576

Ch. Heinrich, B. Simeon, and St. Boschert. 2012. A finite volume method on NURBS geometries and its application in isogeometric fluid–structure interaction. *Mathematics and Computers in Simulation* 82, 9 (2012), 1645 – 1666. https://doi.org/10.1016/j.matcom.2012.03.008

Alec Jacobson, Zhigang Deng, Ladislav Kavan, and J. P. Lewis. 2014. Skinning: Real-Time Shape Deformation (Full Text Not Available). In *ACM SIGGRAPH 2014 Courses* (Vancouver, Canada) *(SIGGRAPH '14)*. Association for Computing Machinery, New York, NY, USA, Article 24, 1 pages. https://doi.org/10.1145/2614028.2615427

Peter Kaufmann, Sebastian Martin, Mario Botsch, and Markus Gross. 2008. Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*

(Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 105–115.

Amir Khosravifard and Mohammad Rahim Hematiyan. 2010. A new method for meshless integration in 2D and 3D Galerkin meshfree methods. *Engineering Analysis with Boundary Elements* 34, 1 (2010), 30 – 40. https://doi.org/10.1016/j.enganabound.2009.07.008

László Kudela, Nils Zander, Tino Bog, Stefan Kollmannsberger, and Ernst Rank. 2015. Efficient and accurate numerical quadrature for immersed boundary methods. *Advanced Modeling and Simulation in Engineering Sciences* 2 (06 2015). https://doi.org/10.1186/s40323-015-0031-y

Grégory Legrain. 2013. A NURBS enhanced extended finite element approach for unfitted CAD analysis. *Computational Mechanics* 52 (04 2013). https://doi.org/10.1007/s00466-013-0854-7

Konstantin Lipnikov, Gianmarco Manzini, and Mikhail Shashkov. 2014. Mimetic Finite Difference Method. *J. Comput. Phys.* 257 (Jan. 2014), 1163–1227. https://doi.org/10.1016/j.jcp.2013.07.031

Andreas Longva, Fabian Löschner, Tassilo Kugelstadt, José Antonio Fernández-Fernández, and Jan Bender. 2020. Higher-Order Finite Elements for Embedded Simulation. *ACM Trans. Graph.* 39, 6, Article 181 (Nov. 2020), 14 pages. https://doi.org/10.1145/3414685.3417853

Sebastian Martin, Peter Kaufmann, Mario Botsch, Eitan Grinspun, and Markus Gross. 2010. Unified Simulation of Elastic Rods, Shells, and Solids. *ACM Trans. Graph.* 29, 4, Article 39 (July 2010), 10 pages. https://doi.org/10.1145/1778765.1778776

B. Müller, S. Krämer-Eis, F. Kummer, and M. Oberlack. 2017. A high-order discontinuous Galerkin method for compressible flows with immersed boundaries. *Internat. J. Numer. Methods Engrg.* 110, 1 (2017), 3–30. https://doi.org/10.1002/nme.5343 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.5343

B. Müller, F. Kummer, and M. Oberlack. 2013. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *Internat. J. Numer. Methods Engrg.* 96, 8 (2013), 512–528. https://doi.org/10.1002/nme.4569 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.4569

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109 – 118. https://doi.org/10.1016/j.jvcir.2007.01.005

Matthias Müller and Nuttapong Chentanez. 2011. Solid Simulation with Oriented Particles. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 92, 10 pages. https://doi.org/10.1145/1964921.1964987

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless Deformations Based on Shape Matching. *ACM Trans. Graph.* 24, 3 (July 2005), 471–478. https://doi.org/10.1145/1073204.1073216

M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. 2004. Point Based Animation of Elastic, Plastic and Melting Objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Grenoble, France) *(SCA '04)*. Eurographics Association, Goslar, DEU, 141–151. https://doi.org/10.1145/1028523.1028542

B. Nayroles, G. Touzot, and P. Villon. 1992. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics* 10 (1992), 307–318.

Alec R. Rivers and Doug L. James. 2007. FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation. In *ACM SIGGRAPH 2007 Papers* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 82–es. https://doi.org/10.1145/1275808.1276480

Masoud Safdari, Ahmad R. Najafi, Nancy R. Sottos, and Philippe H. Geubelle. 2015. A NURBS-based interface-enriched generalized finite element method for problems with complex discontinuous gradient fields. *Internat. J. Numer. Methods Engrg.* 101, 12 (2015), 950–964. https://doi.org/10.1002/nme.4852 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.4852

Masoud Safdari, Ahmad R. Najafi, Nancy R. Sottos, and Philippe H. Geubelle. 2016. A NURBS-based generalized finite element scheme for 3D simulation of heterogeneous materials. *J. Comput. Phys.* 318 (2016), 373 – 390. https://doi.org/10.1016/j.jcp.2016.05.004

Ruben Sevilla, Sonia Mendez, and Antonio Huerta. 2008. Nurbs-enhanced finite element method (NEFEM). *Internat. J. Numer. Methods Engrg.* 76 (10 2008), 56–83. https://doi.org/10.1002/nme.2311

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses* (Los Angeles, California) *(SIGGRAPH '12)*. Association for Computing Machinery, New York, NY, USA, Article 20, 50 pages. https://doi.org/10.1145/2343483.2343501

Denis Steinemann, Miguel A. Otaduy, and Markus Gross. 2008. Fast Adaptive Shape Matching Deformations. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 87–94.

Thomas Stumpp, Jonas Spillmann, Markus Becker, and Matthias Teschner. 2008. A Geometric Deformation Model for Stable Cloth Simulation. *VRIPHYS 2008 - 5th Workshop on Virtual Reality Interactions and Physical Simulations*, 39–46. https://doi.org/10.2312/PE/vriphys/vriphys08/039-046

Demetri Terzopoulos and Hong Qin. 1994. Dynamic NURBS with Geometric Constraints for Interactive Sculpting. *ACM Trans. Graph.* 13, 2 (April 1994), 103–136. https://doi.org/10.1145/176579.176580

L. Veiga, Franco Brezzi, Andrea Cangiani, G. Manzini, L. Marini, and Alessandro Russo. 2012. Basic principles of Virtual Element Methods. *Mathematical Models and Methods in Applied Sciences* 23 (11 2012). https://doi.org/10.1142/S0218202512500492

Bin Wang, François Faure, and Dinesh K. Pai. 2012. Adaptive image-based intersection volume. *ACM Trans. Graph.* 31, 4 (2012), 97:1–97:9. https://doi.org/10.1145/2185520.2185593

Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. 2015. Linear Subspace Design for Real-Time Shape Deformation. *ACM Trans. Graph.* 34, 4, Article 57 (July 2015), 11 pages. https://doi.org/10.1145/2766952

Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2020. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Reconstruction. *arXiv preprint arXiv:2010.02392* (2020).

Jiayi Eris Zhang, Seungbae Bang, David I. W. Levin, and Alec Jacobson. 2020. Complementary dynamics. *ACM Transactions on Graphics* 39, 6 (Nov 2020), 1–11. https://doi.org/10.1145/3414685.3417819

Dump of some equations

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

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

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