Phong Deformation: A better C^0 interpolant for embedded deformation

DOUG L. JAMES, Pixar Animation Studios and Stanford University

Physics-based simulations of deforming tetrahedral meshes are widely used to animate detailed embedded geometry. Unfortunately most practitioners still use linear interpolation (or other low-order schemes) on tetrahedra, which can produce undesirable visual artifacts, e.g., faceting and shading artifacts, that necessitate increasing the simulation's spatial resolution and, unfortunately, cost.

In this paper, we propose *Phong Deformation,* a simple, robust and practical vertex-based quadratic interpolation scheme that, while still only C^0 continuous like linear interpolation, greatly reduces visual artifacts for embedded geometry. The method first averages element-based linear deformation models to vertices, then barycentrically interpolates the vertex models while also averaging with the traditional linear interpolation model. The method is a fast, robust, and easily implemented replacement for linear interpolation that produces visually better results for embedded deformation with irregular tetrahedral meshes.

CCS Concepts: • **Computing methodologies** → *Modeling and simulation*; *Physical simulation.*

Additional Key Words and Phrases: Embedded deformation, tetrahedra, interpolation, cell-to-vertex reconstruction.

ACM Reference Format:
Doug L. James. 2020. Phong Deformation: A better C^0 interpolant for embedded deformation. *ACM Trans. Graph.* 39, 4, Article 56 (July 2020), 9 pages. https://doi.org/10.1145/3386569.3392371

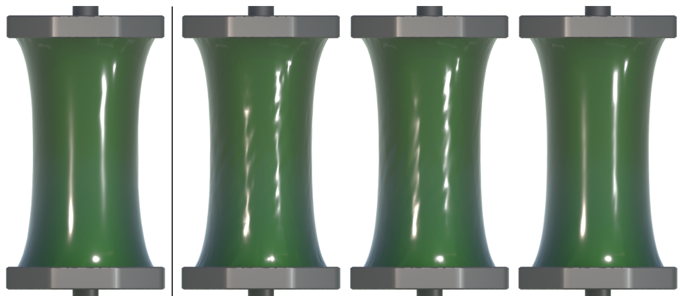

Fig. 1. **Bad embedded deformations make simulations look worse!** (FarLeft) An elastic cylinder is stretched between two plates in this Houdini FEM simulation (70,476 tetrahedra) and looks reasonable. However, embedded deformation of a high-resolution cylinder mesh (30,720 triangles) produces unsightly visual artifacts when using popular displacement interpolants: (MidLeft) Houdini FEM's default "attribute transfer" approach, or (MidRight) traditional linear interpolation. Such artifacts can lead practitioners to unnecessarily increase the FEM simulation resolution (which is very costly), or decrease the tri-mesh resolution, to hide artifacts. (FarRight) In contrast, our proposed Phong Deformation scheme provides a fast, high-quality embedded deformation with minimal artifacts.

1 INTRODUCTION

Physics-based animations of detailed geometric models are often performed by embedding the latter in simulated tetrahedral meshes, with coarse tetrahedral meshes used for speed whenever possible. By far the most common approach for transferring deformations from tetrahedral simulation meshes to high-resolution embedded meshes is linear interpolation within each tetrahedron, and for good reason: it is simple, fast, and trivial to implement. Unfortunately, linear interpolation produces only C^0 continuous displacement fields that have unsightly visual artifacts, such as faceting, in deformations and shading (see Figure 1). Popular alternatives, such as kernel-based scattered data interpolation schemes like Houdini's "attribute transfer" [SideFX 2020] can reduce faceting but introduce other artifacts. Animators seeking to avoid such artifacts are forced to increase the simulation resolution and lament the cost of FEM solvers, and/or reduce the resolution of embedded meshes. While higher continuity than C^0 is not strictly necessary when deforming embedded meshes, reducing interpolation errors for embedded deformation would benefit practitioners everywhere.

In this paper, we introduce *Phong Deformation,* a simple and practical interpolation scheme that, while still only C^0 continuous, greatly reduces visual artifacts for embedded deformations using coarse tetrahedral meshes (see Figures 1 and 3). Phong Deformation is, like its counterpart, linear interpolation, inherently simple, fast, robust, and easy to implement. However it is far more effective at reducing interpolation artifacts due to its quadratic deformation. Mathematically, a robust cell-to-vertex reconstruction is first performed to estimate second-order-accurate deformation gradients, and related linear deformation models, at vertices. Embedded deformation then involves barycentrically interpolating these vertex deformers across the element while also averaging with the traditional barycentric (linear) interpolation model (see Figure 2). Phong Deformation is fast, trivial to implement, affine invariant, and it produces visually better results than linear interpolation in practice, in part because the latter is only a second-order accurate interpolation scheme, whereas we show that Phong Deformation can achieve third-order accuracy, i.e., $O(h^2)$ vs $O(h^3)$ truncation error. The Phong deformer is similar to classical 10-node tetrahedral quadratic interpolants, but its novel construction is entirely vertex based and so it avoids the computation and storage of intermediate edge midpoint values, and it has a simple evaluation based on barycentric interpolation of vertex quantities. Finally, a key contribution is the robust regularized estimation of vertex gradients such that Phong Deformation gracefully degrades from cubic- to second-order accuracy (depending on what neighbor data is available), and never fails in practice.

Author's address: Doug L. JamesPixar Animation Studios; Stanford University, djames@cs.stanford.edu.

0730-0301/2020/7-ART56
https://doi.org/10.1145/3386569.3392371

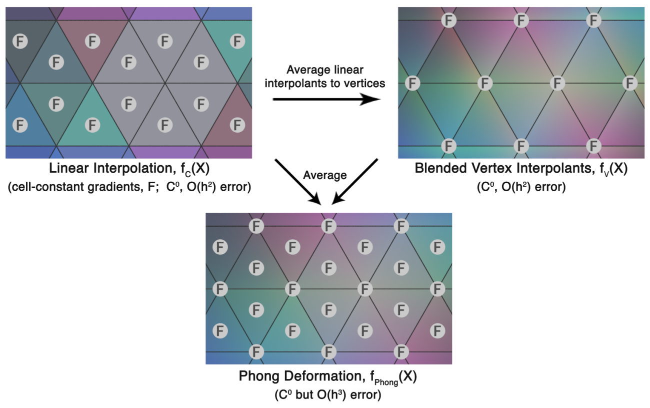
F
Linear Interpolation, $f_C(X)$
(cell-constant gradients, F; C^0, $O(h^2)$ error)
Average linear interpolants to vertices
Average
Blended Vertex Interpolants, $f_V(X)$
(C^0, $O(h^2)$ error)
Phong Deformation, $f_{Phong}(X)$
(C^0 but $O(h^3)$ error)

Fig. 2. **Overview of Phong Deformation construction**

2 RELATED WORK

Embedded Deformation. Driven by increasing geometric complexity and advances in computing and tetrahedral meshing [Hu et al. 2018; Labelle and Shewchuk 2007], the embedded simulation of detailed geometric models using irregular tetrahedral meshes with low-order isoparametric elements has become pervasive in computer graphics, and a core part of industrial animation systems, e.g., Houdini's FEM and Vellum solvers [SideFX 2020] and Ziva VFX's tet-embedded FEM solvers [Ziva Dynamics 2020]. Major benefits of embedded deformation include the ability to deform arbitrary geometric assets, and the speed that comes with using relatively coarse simulation meshes. Fundamental research advances include decades of real-time deformable simulations using coarse tetrahedral models [Capell et al. 2002; Debunne et al. 2001; Müller and Gross 2004]; simulation support for topological changes during cutting and fracture [Molino et al. 2004], and adaptively (re)meshed tetrahedra for embedded modeling of elastoplastic deformations [Wicke et al. 2010; Wojtan and Turk 2008]. Embedded deformation is the central part of lattice-based deformers [MacCracken and Joy 1996; Muller et al. 2004; Patterson et al. 2012; Rivers and James 2007; Sederberg and Parry 1986], coarsened simulation models [Kharevych et al. 2009; Nesme et al. 2009], and some dimensional model reduction implementations [Barbič and James 2005].

In most works using tetrahedral meshes the embedded deformation is reconstructed using linear interpolation due to (a) its low cost for detailed meshes, (b) its second-order accuracy (i.e., linear reproducing), and (c) obviousness—the shape function often matches the underlying isoparametric (finite element) analysis. Unfortunately, linear interpolation can produce visual artifacts due to deformation gradient discontinuities. Limiting embedded mesh resolution, or using subdivision [DeRose et al. 1998], or multi-resolution displacement mapping [James and Pai 2003] can hide artifacts to some extent, but do not address the core reconstruction deficiency. Alternative interpolation schemes include scattered data methods such as Shepard's interpolation (low-quality), radial basis functions (e.g., metaballs), and thin-plate splines [Nelles 2013; Wendland 2004]. These general-purpose methods do not exploit the tetrahedral mesh structure, and can struggle to achieve higher-order interpolation accuracy while remaining fast for detailed meshes, in part by using local computations that avoid (global) linear solves at runtime. Moving Least Squares (MLS) methods [Levin 1998] can achieve local data interpolation and C^∞ continuity, but require special data structures, and are more expensive for dense volume deformations, e.g., it must solve a linear system (and/or cache weights) at each evaluation point. For reference, Houdini's tetrahedral FEM and Vellum implementations [SideFX 2020] use a point-based attribute transfer (e.g., Point Deform SOP) scheme that involves distance-based metaball weighting—a practical yet only first-order accurate interpolation scheme. In contrast, our approach is meant to be simple, fast, and local, but also third-order accurate when possible.

Generalized barycentric coordinate schemes have been devised for interpolation on nonconvex 2D polygonal or 3D polyhedral domains, such as Mean Value Coordinates [Floater 2003; Ju et al. 2005], Harmonic Coordinates [Joshi et al. 2007], etc. However, such schemes are typically used with "cages" and simplify to linear interpolation within a single tetrahedron, whereas we propose a C^0 quadratic interpolant for vertex data on tetrahedral meshes.

In recent years, various schemes have been proposed to smoothly deform embedded or coarsely "rigged" geometry. For example, Sumner et al. [2007] proposed a general-purpose embedded deformation model based on a graph-based mesh skinning interpolant estimated using optimization; node deformers are similar to our linear vertex deformers, however, the Shepard interpolation of scattered nodes is only first-order accurate, and therefore, while smooth, it is second-order accurate at best. Huang et al. [2008] optimizes a modified barycentric interpolation scheme for a tetrahedral control mesh that is analogous to a linear-blend skinning scheme with displacement and transform data located at mesh vertices; the method is best suited for low-resolution control meshes, since a linear solve is required at each time step to solve the global optimization problem. Unfortunately, this nonlocal nature is undesirable for simulation applications that require detailed meshes with low per-element reconstruction costs, e.g., the simple cylinder in Figure 1 has 70k tetrahedra, whereas a character can have millions of elements. Requiring a global matrix solve (or back substitution) just to interpolate the solution at each timestep is a barrier to adoption in terms of implementation overhead, and performance.

Phong Shading and Tessellation. Our method is inspired, in part, by Phong shading [Phong 1975], and especially Phong Tessellation for triangle surface deformation [Boubekeur and Alexa 2008]. The latter quadratically deforms a triangle mesh using vertex normals to reduce polygonal artifacts of coarse meshes; it amounts to blending each barycentric triangle position with the linearly interpolated position on each vertex's tangent plane model. Phong Deformation is analogous but performs a volumetric deformation, with reconstructed vertex deformation gradient models replacing vertex tangent planes; blending is performed to combine linear interpolation with a blended vertex deformation model. We are able to show formally that Phong Deformation is a third-order accurate interpolation scheme under certain conditions. Related approaches exist that use cubic spline patches, such as PN Triangles [Vlachos et al. 2001], but their volumetric spline analogues are less appealing for our problem.

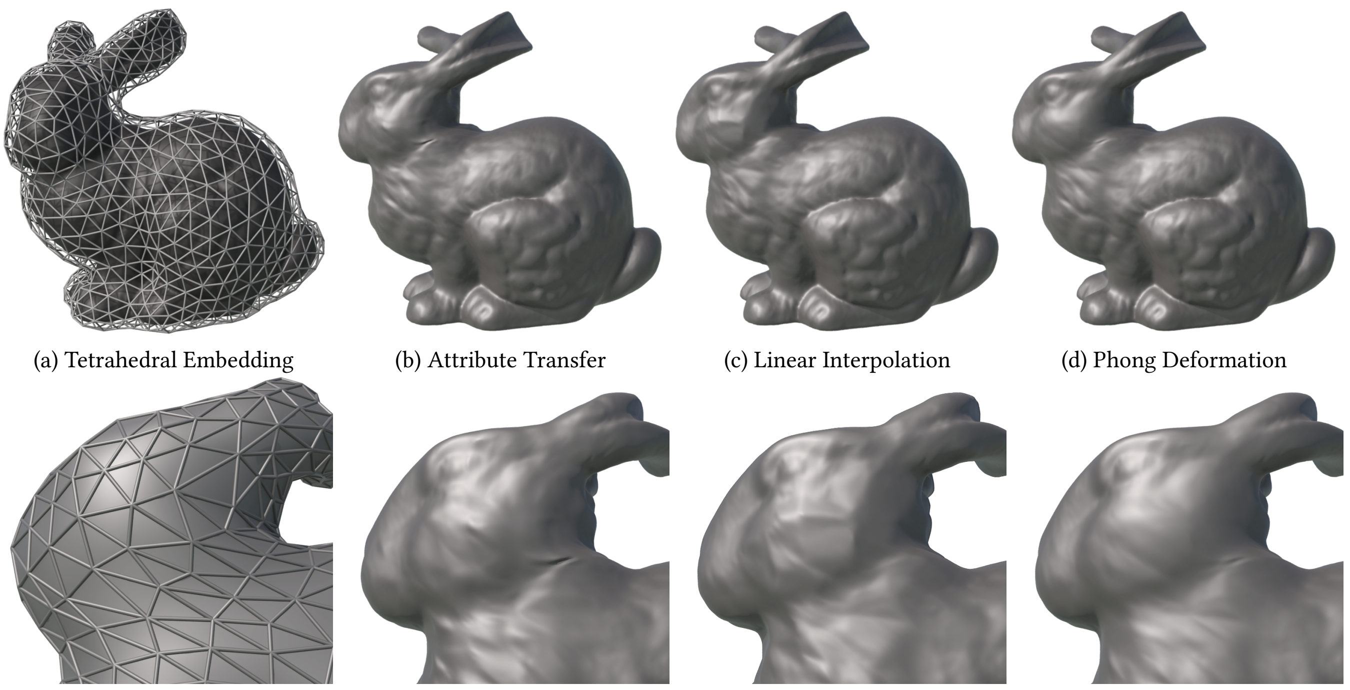

Fig. 3. **A bunny head shake** simulated by deforming (a) a tetrahedral mesh (14,763 tets, 2796 nodes) with Houdini FEM using spring control forces (on the face and bottom). The embedded triangle mesh (69,664 tri, 34,834 vtx) is deformed using three techniques with varying interpolation artifacts near the face and neck regions: (b) Houdini's default attribute transfer tends to produce irregular bumpy artifacts, (c) linear interpolation produces characteristic faceted artifacts, and (d) Phong Deformation has faceted but significantly reduced visual artifacts. The rendered mesh is subdivided once (278k tri).

Higher-order interpolation on tetrahedral meshes. C^1 and higher continuity schemes are possible on tetrahedral meshes, and a natural replacement for deficient C^0 schemes, but they are vastly more complicated than linear interpolation and, unfortunately, impractical. For example, discrete C^1 interpolants on tetrahedral meshes require complex schemes to ensure continuity [Alfeld 1984, 1985], require the user to provide both vertex data and many higher derivatives—the latter not usually being available—and can be computationally expensive (see [Alfeld and Sorokina 2009] for a recent comparison of methods). Higher-order C^m interpolation on tetrahedra is possible, but requires the user provide or estimate higher-order quantities, e.g., C^{2m} data at vertices [Xu 2001], which is impractical in computer animation. Furthermore, unlike in the case of higher-order interpolation on regular grid-based hexahedral meshes where reusable stencils exist [Patterson et al. 2012], the significant computation and memory overhead of estimating higher-order schemes can not be amortized for irregular tetrahedral meshes. In contrast, we propose a simple C^0 quadratic interpolant that is more accurate than linear interpolation, and still cheap to evaluate; the scheme interpolates the original vertices, but only approximates gradients.

Our novel quadratic interpolant is vertex based and avoids midpoint values used by classical 10-node quadratic interpolants on tetrahedra [Zienkiewicz et al. 2005], which has practical benefits such as a simpler implementation and lower memory usage (discussed further in §3.3). Other advanced visualization techniques include *data-bounded* quadratic interpolants that limit interpolation accuracy to avoid over-shooting of bounded quantities [Berzins 2000], but these are unnecessary for embedded deformation.

Higher-order deformation analysis. It is also possible to improve the smoothness properties of the underlying deformation analysis, such as by using higher-order finite elements. Unfortunately, for irregular tetrahedral meshes, C^1 finite elements are impractical and rarely used, although constructions exist using ninth-degree polynomials [Ženíšek 1973]. Higher-order (but still C^0) finite element deformation schemes are common in engineering and are used in graphics, e.g., quadratic elements [Bargteil and Cohen 2014]; while they can reduce simulation error and visual artifacts, they come with higher simulation costs that reduce the number of elements possible. In this work, we target the more widespread low-order tetrahedral deformation models; we assume a deformed tetrahedral mesh is provided, and simply attempt to improve the C^0 interpolation scheme used to rapidly transfer deformations to detailed embedded geometry.

Vertex deformation gradients. Phong Deformation requires the estimation of vertex deformation gradients from available cell-centered deformation gradients using appropriate cell-to-vertex weights–analogous to weights for estimating vertex normals from facet normals [Max 1999]. Vertex deformation gradients can be estimated from the displaced vertices of the immediately surrounding tetrahedra using weighted least squares, however care must be taken to achieve robustness and second-order accuracy. In the finite volume community, cell-vertex reconstruction methods are commonly used to consistently interpolate tetrahedral centroid data to vertices, that we use to interpolate tetrahedral deformation gradients to vertices. Second-order interpolation methods (that can reconstruct linear polynomial functions) are commonly estimated using linear least squares [Frink 1994; Holmes and Connell 1989; Rausch et al. 1992] with extensions for interpolants with positive weights [Costa et al. 2014]. In this work, we adapt the variant of Chandrashekar and Garg [2013] consisting of modified Shepard's inverse-distance interpolation with weights constrained to reconstruct linear polynomial functions, since it is sufficient and fast to precompute on the undeformed mesh, e.g., one 3x3 matrix solve per tet-mesh vertex.

3 PHONG DEFORMATION

In this section, we introduce the Phong Deformation model in the context of tetrahedral meshes.

3.1 Background and Notation

Consider a tetrahedron with undeformed (material space) vertex positions $\boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3$, that is deformed by a smooth function,

$$\boldsymbol{x} = \mathrm{f}(\boldsymbol{X}), \tag{1}$$

to yield deformed vertex positions $\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$, where $\boldsymbol{x}_i = \mathrm{f}(\boldsymbol{X}_i)$.

Linear interpolation is commonly used to approximate the deformation of an interior point, $\boldsymbol{X}$, to its displaced position

$$\bar{\boldsymbol{x}}(\boldsymbol{X}) = \sum_{i=0}^{3} \beta_i(\boldsymbol{X})\, \boldsymbol{x}_i \qquad \text{[Linear interpolation]} \tag{2}$$

where $\beta_i = \beta_i(\boldsymbol{X})$ are the barycentric coordinates such that

$$\boldsymbol{X} = \sum_{i=0}^{3} \beta_i\, \boldsymbol{X}_i, \qquad \text{and} \qquad 1 = \sum_{i=0}^{3} \beta_i \tag{3}$$

(dropping explicit summation over $i = 0, 1, 2, 3$ hereafter). Given $\boldsymbol{X}$, the barycentric coordinates of the embedding in an element are precomputed by solving a 3×3 linear system,

$$(\beta_1, \beta_2, \beta_3) = \mathsf{V}^{-1} (\boldsymbol{X} - \boldsymbol{X}_0) \tag{4}$$

$$\beta_0 = 1 - \beta_1 - \beta_2 - \beta_3 \tag{5}$$

where V is the undeformed 3×3 edge basis matrix,

$$\mathsf{V} = [(\boldsymbol{X}_1 - \boldsymbol{X}_0) \;\; (\boldsymbol{X}_2 - \boldsymbol{X}_0) \;\; (\boldsymbol{X}_3 - \boldsymbol{X}_0)]\,. \tag{6}$$

We can also interpret linear interpolation $\bar{\boldsymbol{x}}(\boldsymbol{X})$ as a linear model in terms of the element's 3×3 deformation gradient, $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}$ evaluated at the cell center, $\boldsymbol{F}_C$:

$$\mathrm{f}_C(\boldsymbol{X}) = \boldsymbol{c} + \boldsymbol{F}_C\, (\boldsymbol{X} - \boldsymbol{C}), \tag{7}$$

where $\boldsymbol{c} = \sum_i \boldsymbol{x}_i/4$ is the deformed position of the undeformed centroid, $\boldsymbol{C} = \sum_i \boldsymbol{X}_i/4$. The linear tetrahedral element's deformation gradient is $\boldsymbol{F}_C = \mathsf{v}\, \mathsf{V}^{-1}$, where v is the deformed edge basis matrix,

$$\mathsf{v} = [(\boldsymbol{x}_1 - \boldsymbol{x}_0) \;\; (\boldsymbol{x}_2 - \boldsymbol{x}_0) \;\; (\boldsymbol{x}_3 - \boldsymbol{x}_0)]\,. \tag{8}$$

For general deformations, $\boldsymbol{F}_C$ can be interpreted as a second-order accurate approximation of $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}$ at the element's centroid, $\boldsymbol{C}$.

3.2 Phong Deformer

For each vertex i of the tetrahedron, we define a linear deformer with vertex deformation gradient, $\boldsymbol{F}_i$, so that

$$\mathrm{f}_i(\boldsymbol{X}) = \boldsymbol{x}_i + \boldsymbol{F}_i\, (\boldsymbol{X} - \boldsymbol{X}_i), \tag{9}$$

interpolates the deformation at vertex i and has matching vertex gradient. We then barycentrically interpolate these vertex deformers, $\mathrm{f}_i(\boldsymbol{X})$, across the element:

$$\mathrm{f}_V(\boldsymbol{X}) = \sum_i \beta_i\, \mathrm{f}_i(\boldsymbol{X}) = \sum_i \beta_i \Big(\boldsymbol{x}_i + \boldsymbol{F}_i\, (\boldsymbol{X} - \boldsymbol{X}_i)\Big). \tag{10}$$

Unfortunately it turns out that this $\mathrm{f}_V(\boldsymbol{X})$ interpolant is only second-order accurate, similar to linear interpolation. Serendipitiously, by blending these two cell- and vertex-based deformers,

$$(1 - \alpha)\, \mathrm{f}_C(\boldsymbol{X}) + \alpha\, \mathrm{f}_V(\boldsymbol{X}), \tag{11}$$

the result can be improved (analogous to the case of Phong Tessellation [Boubekeur and Alexa 2008]). We show in §5 that when $\alpha = 1/2$ it actually provides a third-order accurate approximation, which is our proposed Phong Deformation model:

$$\mathrm{f}_{\text{Phong}}(\boldsymbol{X}) \equiv \frac{\mathrm{f}_C(\boldsymbol{X}) + \mathrm{f}_V(\boldsymbol{X})}{2}. \tag{12}$$

Other useful forms are

$$\mathrm{f}_{\text{Phong}}(\boldsymbol{X}) = \frac{\bar{\boldsymbol{x}}(\boldsymbol{X}) + \mathrm{f}_V(\boldsymbol{X})}{2} = \frac{1}{2}\left[\sum_i \beta_i \boldsymbol{x}_i + \sum_i \beta_i \mathrm{f}_i(\boldsymbol{X})\right] \tag{13}$$

$$= \bar{\boldsymbol{x}}(\boldsymbol{X}) + \frac{1}{2}\sum_i \beta_i(\boldsymbol{X})\, \boldsymbol{F}_i\, (\boldsymbol{X} - \boldsymbol{X}_i), \tag{14}$$

and the piecewise linear deformation gradient,

$$\nabla\, \mathrm{f}_{\text{Phong}}(\boldsymbol{X}) = \frac{1}{2}\left[\boldsymbol{F}_C + \sum_i \Big(\beta_i(\boldsymbol{X})\; \boldsymbol{F}_i + \mathrm{f}_i(\boldsymbol{X})\; \nabla \beta_i^T\Big)\right]. \tag{15}$$

3.3 Properties

C^0 continuous and interpolating. Since both linear interpolation $\mathrm{f}_C(\boldsymbol{X})$ and the vertex model $\mathrm{f}_V(\boldsymbol{X})$ are C^0 continuous and preserve vertex values, $\boldsymbol{x}_i$, the Phong Deformation model is also C^0 continuous and preserves vertex values.

Third-order accurate. For elements of scale h, traditional linear interpolation is $O(h^2)$ accurate, whereas we show in §5 that Phong Deformation is $O(h^3)$ accurate.

Affine invariance. Since both linear interpolation $\mathrm{f}_C(\boldsymbol{X})$ and the vertex deformer $\mathrm{f}_V(\boldsymbol{X})$ exhibit affine invariance, i.e., the shape resulting from transforming the input points $\mathrm{f}(\boldsymbol{X}; \{\boldsymbol{T}\boldsymbol{x}_i\})$ produces the same result as transforming the output, $\boldsymbol{T}\mathrm{f}(\boldsymbol{X}; \{\boldsymbol{x}_i\})$, we can conclude that the Phong Deformation result is also affine invariant.

Quadratic correction. The Phong deformer is simply a quadratic correction (in β) to the linear interpolant (linear in β) as the correction to $\bar{\boldsymbol{x}}(\boldsymbol{X})$ in (14) is quadratic in β: since $\boldsymbol{X} = \sum_j \beta_j \boldsymbol{X}_j$,

$$\mathrm{f}_{\text{Phong}}(\boldsymbol{X}) = \bar{\boldsymbol{x}}(\boldsymbol{X}) + \frac{1}{2}\sum_{i,j} \beta_i \beta_j\, \boldsymbol{F}_i\, (\boldsymbol{X}_j - \boldsymbol{X}_i). \tag{16}$$

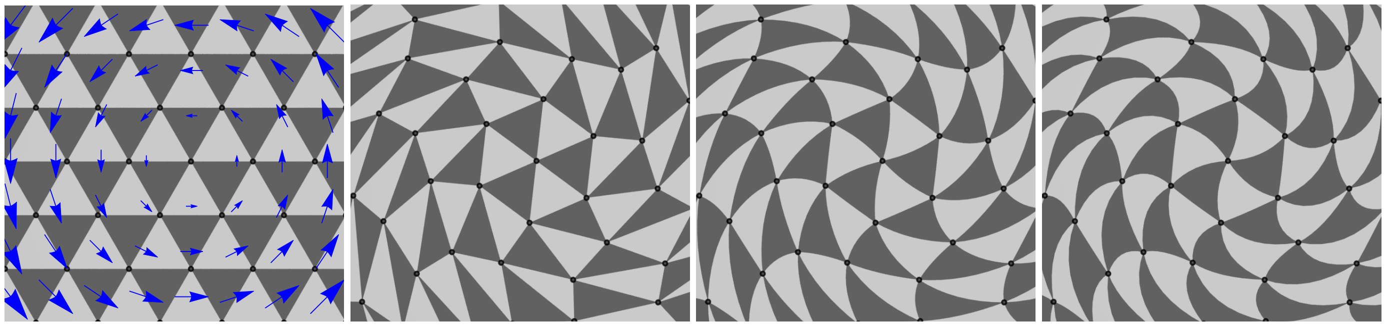
Triangle Tessellation w/ Twist
(1) Linear Interpolation, f_C
(2) Phong Deformation, f_{Phong}
(3) Blended vertex deformer, f_V

Fig. 4. **2D embedded deformation of a triangle tessellation** using (1) linear interpolation, f_C; (2) Phong Deformation, $f_{Phong} = \frac{f_C+f_V}{2}$, and (3) the blended vertex deformer, f_V. Triangle colors indicate the coarse-mesh embedding; a twist rotation (with $\theta \propto r$) is sampled at coarse nodes (in black). While all three interpolation schemes are only C^0 continuous, Phong Deformation's quadratic deformations do a better job of hiding discontinuity artifacts.

Correction only depends on variation in F_i. By properties of the barycentric coordinates (3), for any constant matrix $\boldsymbol{F}$,

$$0 = \sum_i \beta_i\,(\boldsymbol{X} - \boldsymbol{X}_i) = \sum_i \beta_i\; \boldsymbol{F}\,(\boldsymbol{X} - \boldsymbol{X}_i), \tag{17}$$

so it follows that the quadratic correction is only sensitive to variations in the vertex deformation gradients, $\boldsymbol{F}_i$. Therefore we could write the correction as

$$\bar{x}(\boldsymbol{X}) + \frac{1}{2}\sum_i \beta_i\,\Delta\boldsymbol{F}_i\,(\boldsymbol{X} - \boldsymbol{X}_i) \tag{18}$$

for, e.g., $\Delta\boldsymbol{F}_i = \boldsymbol{F}_i - \boldsymbol{F}_C$ on a tetrahedron, to illustrate that only gradient changes contribute to the quadratic correction. Consequently, if all vertex gradients are equal, the quadratic correction is zero, and Phong Deformation simplifies to linear interpolation.

Half-gradient interpretation. Interestingly, the Phong Deformation model can be interpreted as the interpolated linear vertex deformer model $f_V(\boldsymbol{X})$ in (10), but with "half-gradients," $\boldsymbol{H}_i = \boldsymbol{F}_i/2$:

$$f_{Phong}(\boldsymbol{X}) = \sum_i \beta_i\Big(\boldsymbol{x}_i + \boldsymbol{H}_i\,(\boldsymbol{X} - \boldsymbol{X}_i)\Big). \tag{19}$$

This also provides a straightforward recipe to apply Phong Deformation to other 2D and 3D settings.

For example, on *hexahedral grids*, β_i are the trilinear interpolation weights of $\boldsymbol{X}$ within the cell, $\boldsymbol{H}_i$ are the vertex half-gradients estimated from cell-centered half-gradients using the cell-to-vertex reconstruction algorithm in §4 (or centered differences for interior nodes of regular grids). We have verified this achieves third-order accuracy for regular grids with second-order accurate $\boldsymbol{H}_i$; for isolated cells with constant $\boldsymbol{H}_i$, $f_{Phong}(\boldsymbol{X})$ simplifies to $\bar{\boldsymbol{x}}(\boldsymbol{X})$—trilinear interpolation.

A vertex-based quadratic interpolant without midpoints. Given the four vertex deformation gradients, $\boldsymbol{F}_i$, one could alternately estimate and store $\boldsymbol{x}_m$ at the six edge midpoints using Hermite interpolation [Alfeld 1984], and then use classical 10-node quadratic interpolation based on Lagrange interpolating polynomials [Zienkiewicz et al. 2005] to deform the embedded geometry. Given an edge $(\boldsymbol{X}_0, \boldsymbol{X}_1) \longrightarrow (\boldsymbol{x}_0, \boldsymbol{x}_1)$ with vertex gradients $\boldsymbol{F}_0$ and $\boldsymbol{F}_1$, the Hermite-interpolated midpoint value is

$$\begin{aligned}
\boldsymbol{x}_m &= \frac{1}{2}(\boldsymbol{x}_0 + \boldsymbol{x}_1) + \frac{1}{8}(\boldsymbol{x}_0' - \boldsymbol{x}_1') && (20)\\
&= \frac{1}{2}(\boldsymbol{x}_0 + \boldsymbol{x}_1) + \frac{1}{8}\left(\boldsymbol{F}_0(\boldsymbol{X}_1 - \boldsymbol{X}_0) - \boldsymbol{F}_1(\boldsymbol{X}_1 - \boldsymbol{X}_0)\right) && (21)\\
&= \frac{1}{2}(\boldsymbol{x}_0 + \boldsymbol{x}_1) + \frac{1}{2}\left(\frac{\boldsymbol{F}_0}{2}\frac{\boldsymbol{X}_1 - \boldsymbol{X}_0}{2} + \frac{\boldsymbol{F}_1}{2}\frac{\boldsymbol{X}_0 - \boldsymbol{X}_1}{2}\right) && (22)\\
&= \sum_{i\in\{0,1\}} \beta_i\Big(\boldsymbol{x}_i + \boldsymbol{H}_i\,(\boldsymbol{X}_m - \boldsymbol{X}_i)\Big) && (23)
\end{aligned}$$

where the last line matches the half-gradient Phong Deformation formula (19). Therefore, by uniqueness of the 10-dof quadratic polynomial, the Phong Deformer and the 10-node quadratic interpolant will produce mathematically identical quadratic functions.

However, there are practical benefits of the vertex-based Phong Deformation scheme: (1) it avoids the extra step of computing and storing midpoint values, which (2) reduces the amount of data that must be cached and accessed (tetrahedral meshes can have $> 7\times$ as many edges as vertices making midpoint storage worse than just adding vertex gradients), and (3) the vertex-based Phong Deformation scheme with linear barycentric interpolation leads to a much simpler implementation without the need for edge-based data structures to store/access midpoint attributes. So, the 10-node interpolant is a natural choice for isoparametric finite element analysis, but Phong Deformation is better suited to quadratic reconstruction.

4 VERTEX GRADIENT ESTIMATION

The Phong deformer requires robust estimation of vertex deformation gradients, $\boldsymbol{F}_i$, in order to interpolate the vertex deformation models $f_i(\boldsymbol{X})$. Given the newly deformed vertex positions, we compute all tetrahedral cell gradients, $\boldsymbol{F}_k$, then perform a cell-to-vertex reconstruction to estimate each vertex gradient, $\boldsymbol{F}_i$, using a modification of the second-order accurate scheme of [Chandrashekar and Garg 2013]. The latter is based on a weighted Shepard interpolation scheme of the form

$$\boldsymbol{F}_i \approx \frac{\sum_{k\in C_i}\frac{w_k}{r_k}\,\boldsymbol{F}(\boldsymbol{r}_k)}{\sum_{j\in C_i}\frac{w_j}{r_j}} = \sum_{k\in C_i} w_k'\,\boldsymbol{F}(\boldsymbol{r}_k), \tag{24}$$

where the sums run over all tetrahedral cells C_i adjacent to the vertex i, with relative material-frame vertex-to-centroid vector $\boldsymbol{r}_k$, of magnitude r_k, and unit vector $\hat{\boldsymbol{r}}_k = \boldsymbol{r}_k / r_k$; here w_k are interpolation weights (w'_k are the final normalized weights used at runtime), and $\boldsymbol{F}_k = \boldsymbol{F}(\boldsymbol{r}_k)$ is the gradient at the centroid of the k^{th} adjacent cell.

The interpolant (24) trivially reproduces constant functions, but to enforce second-order accuracy the weights must also reproduce linear functions, and thus satisfy the three consistency conditions:

$$\sum_k w_k \, \hat{\boldsymbol{r}}_k = 0. \tag{25}$$

Minimizing $1/2 \sum_k (w_k - 1)^2$ subject to (25) has the solution

$$w_k = 1 + \boldsymbol{\lambda} \cdot \hat{\boldsymbol{r}}_k \tag{26}$$

where the Lagrange multipliers $\boldsymbol{\lambda}$ satisfy

$$\left[\sum_k \hat{\boldsymbol{r}}_k \hat{\boldsymbol{r}}_k^T \right] \boldsymbol{\lambda} = - \sum_k \hat{\boldsymbol{r}}_k \quad \Leftrightarrow \quad \boldsymbol{A}\,\boldsymbol{\lambda} = \boldsymbol{b}. \tag{27}$$

This second-order interpolation scheme was proposed in [Chandrashekar and Garg 2013], and works well for finite volume methods on good-quality tetrahedral grids, and, because of the objective, tends to encourage (but not guarantee) positive weights.

Regularization. Unfortunately, for our irregular tetrahedral meshes, this procedure can fail when $\boldsymbol{A}$ is ill-conditioned or singular, such as for surface vertices with insufficient neighboring tetrahedra (see Figure 5 for an example with just two tetrahedra). In such cases, the conditions for second-order accurate interpolation (25) can not be satisfied, and a compromise is needed. We use Tikhonov regularization to robustly solve $(\boldsymbol{A} + \varepsilon \boldsymbol{I})\, \boldsymbol{\lambda} = \boldsymbol{b}$, where $\boldsymbol{A}$ is the symmetric positive semi-definite matrix, and ε is the regularization parameter. We observe that a wide range of ε values produce good results on average but small ε values can cause over-fitting near the surface (see Figure 6); we used $\varepsilon = 1$ in our examples. This method works even in the extreme case of a single adjacent tetrahedron where the reconstructed vertex gradient is simply the element's value. We experimented with using additional k-nearest neighbors to avoid tet-deficient cases, such as for the line-like licorice mesh, however, we found that it provided a smoothing effect but tended to degrade numerical accuracy in general.

Normalized weights. Finally, the normalized cell-to-vertex interpolation weights $w'_k = (w_k / r_k) / (\sum_{j \in C_i} \frac{w_j}{r_j})$ are precomputed and cached for each vertex, then reused at runtime to reconstruct vertex gradients $\boldsymbol{F}_i$ prior to Phong Deformation of the embedded mesh.

5 ERROR ANALYSIS

We now show that the Phong Deformer provides a piecewise quadratic approximation to the deformation field. Given the messiness of the 3D truncation error analysis, we first provide a simpler 1D analysis for illustrative purposes.

5.1 Motivation: 1D Case

Analyzing our scheme in the one-dimensional setting provides a simple illustration of its benefits, and also allows us to derive truncation error estimates to clarify what is happening. The simplest setting that provides insight is for uniformly gridded data, with spacing h; let X_i samples be at $\{0, \pm h, \pm 2h, \ldots\}$. Analogous vertex gradients

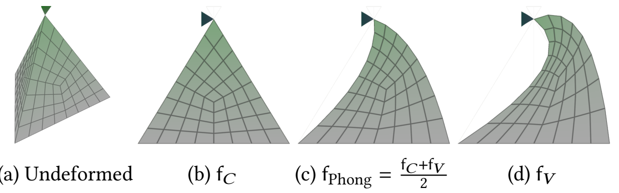

Fig. 5. **Animated vertex deformation gradient:** (a) In this two tetrahedra mesh, the small top element is rotated 90^o to animate a single vertex deformation gradient (via regularized cell-to-vertex averaging) on the large bottom fixed-vertex element. While (b) linear interpolation f_C only uses cell-constant gradients, (c) the Phong Deformer f_{Phong} exhibits smooth deformation coupling to the larger element, whereas (d) the blended vertex deformer f_V produces an exaggerated response. (see video for animations)

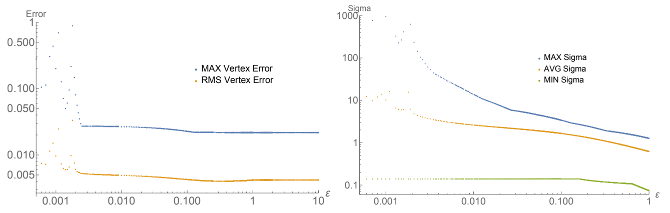

Fig. 6. **Dependence on regularization parameter, ε:** (Left) Maximum and average vertex error shows weak dependence, whereas (Right) large cell-to-vertex weights w' (over-fitting) occurs for small ε values, as shown by a normalized deviation measure, $\sigma = \sqrt{\sum_{k=1 \ldots n} (w'_k n - 1)^2 / n}$. Results are for the "Cell" example using the Kelvinlet deformer (shown later in Figure 8).

can be interpreted as averaging element/edge gradients obtained from centered differences, e.g., $f'_0 \equiv f'(0) \approx \frac{f'(h/2) + f'(-h/2)}{2} \approx \left(\frac{f_1 - f_0}{h} + \frac{f_0 - f_{-1}}{h} \right) / 2$. Consider the element interval $X \in [0, h]$, and let $\beta = X/h \in [0, 1]$.

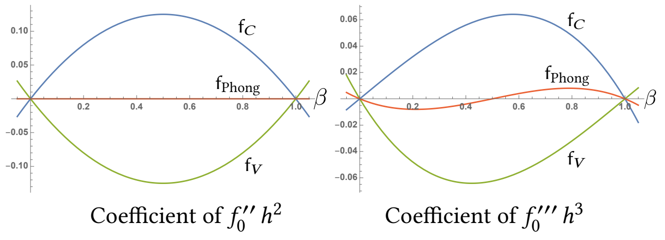

Fig. 7. **Truncation error analysis of 1D uniform-grid case:** Plots of the truncation error coefficients of the (Left) $f''_0 \, h^2$ and (Right) $f'''_0 \, h^3$ terms vs β (where $X = \beta h$) are shown for (f_C in **blue**) linear interpolation; (f_V in **green**) the linearly interpolated vertex-gradient model; and (f_{Phong} in **orange**) the Phong Deformation model which is their average ($\alpha = 1/2$). Observe two things: (Left) the proposed Phong model's $O(h^2)$ error term vanishes entirely making it third-order accurate, and (Right) the $O(h^3)$ error term's coefficient is much smaller. Although the Phong model is still merely C^0 continuous, its smaller truncation error is an improvement over piecewise linear interpolation.

A standard truncation error analysis of linear interpolation, $f(X) \approx (1-\beta)f_0 + \beta f_1$, shows second-order error:

$$-\frac{1}{2}(\beta-1)\beta f_0'' h^2 - \frac{1}{6}\beta(\beta^2-1) f_0''' h^3 + O(h^4). \tag{28}$$

The linearly interpolated vertex-gradient model is

$$f(X) \approx (1-\beta)\bar{f}_0(X) + \beta \bar{f}_1(X) \tag{29}$$

$$\bar{f}_i(X) = f_i + (X - X_i) f_i', \tag{30}$$

and has truncation error:

$$\frac{1}{2}(\beta-1)\beta f_0'' h^2 - \frac{1}{6}(\beta-2)(\beta-1)\beta f_0''' h^3 + O(h^4) \tag{31}$$

which is also only second-order accurate. Interestingly the coefficient of the $O(h^2)$ term is identical to linear interpolation, *except that it has the opposite sign.*

Our Phong deformation model combines these two previous models using a α-blended version with truncation error:

$$\frac{1}{2}(2\alpha-1)(\beta-1)\beta f_0'' h^2 - \frac{1}{6}(\beta-1)\beta(-3\alpha+\beta+1) f_0''' h^3 + O(h^4)$$

for which the leading-order term vanishes when $\alpha = 1/2$:

$$-\frac{1}{12}(\beta-1)\beta(2\beta-1) f_0''' h^3 + O(h^4). \tag{32}$$

Plots of the coefficients of these expansions are shown in Figure 7, and reveal that the leading nonzero coefficient (h^3) is also smaller.

Of course, in 1D one could use Hermite interpolation of the interval's end-point function and gradient values to obtain an $O(h^4)$ approximation, so this $O(h^3)$ interpolant scheme is uncompetitive in 1D, and merely serves to provide intuition for the 3D case.

5.2 Error Analysis: 3D Case

To show that the Phong Deformation model has $O(h^3)$ truncation error, it is sufficient to consider a canonical tetrahedron with vertices at $(0,0,0)$, $(h,0,0)$, $(0,h,0)$, $(0,0,h)$, with function values f_0, f_1, f_2, f_3, and gradients $\boldsymbol{F}_0$, $\boldsymbol{F}_1$, $\boldsymbol{F}_2$, $\boldsymbol{F}_3$. Consider the first component of the f field, denoted by $u = u(\boldsymbol{X})$. To perform error analysis, we again expanded all functions and gradients about the origin. The second-order truncation error of the linear interpolant is then

$$E_{\text{Linear}} = -\left(\sum_{i<j} \beta_i\beta_j\, \partial_i\partial_j u + \frac{1}{2}\sum_i (\beta_i - 1)\beta_i\, \partial_i^2 u\right) h^2 + O(h^3)$$

where $\beta_i = X_i/h,\ i = 1,2,3$, and all derivatives of u are evaluated at the origin. Similar to the 1D case, the interpolated vertex-gradient model also has $O(h^2)$ error, and its coefficients have *equal but opposite signs* as those of the linear error. The Phong Deformation model averages these two $O(h^2)$ interpolants (and so cancels the $O(h^2)$ error terms) yielding an $O(h^3)$ truncation error:

$$E_{\text{Phong}} = -\left(\beta_1\beta_2\beta_3\, \partial_1\partial_2\partial_3 u + \frac{1}{4}\sum_{i\neq j} \beta_i(2\beta_i-1)\beta_j\, \partial_i^2\partial_j u + \frac{1}{12}\sum_i (\beta_i-1)\beta_i(2\beta_i-1)\, \partial_i^3 u\right) h^3 + O(h^4).$$

Notice that the 3D error simplifies to the 1D case when $u = u(X)$.

Finally, this analysis assumes that the vertex gradients, $\boldsymbol{F}_i$, are exact, but the Phong Deformer uses approximate vertex gradients. It follows that the deformer is still $O(h^3)$ accurate provided that the vertex gradients are $O(h^2)$ accurate, which can be achieved using the second-order accurate cell-to-vertex reconstruction scheme (§4) when sufficient neighbors exist.

6 RESULTS

Please see the accompanying video for animations and other results.

Implementation. Phong Deformation was implemented in Houdini (v17.0), with runtime evaluation done in VEX[1] code using Attribute Wrangles. Micro-timings are difficult to do precisely in Houdini, however its Performance Monitor for the Bunny (a representative example) indicates Phong Deformation evaluation costed about 10x as much as trivial linear interpolation, but just 3% of a FEM step (ABE2 w/ 2 substeps), or 5% of the subsequent Loop subdivision and normal calculation. Approximately 10% of Phong Deformation runtime is spent estimating tetrahedra deformation gradient models, 34% performing their cell-to-vertex weighted averages, and about 33% of the time evaluating the actual f_C and f_V displacements.

FEM Examples. Several three-way comparisons have been made with popular interpolation schemes (attribute transfer (Houdini's Point Deform SOP), linear interpolation, and Phong Deformation) on FEM simulations generated using Houdini. Please see the accompanying figures for details. These examples are "rubber toy impact" (see Figure 9), the "licorice twist" (see Figure 11), the "bunny head shake" (see Figure 3), and the cylinder stretch and twist tests (see Figure 1 and video). In all cases, we observed that the Phong Deformation results were an improvement upon the prior popular methods. Please see the video for supporting animations. Unfortunately these comparisons are primarily visual in nature, and therefore we estimated interpolation separately.

Error Analysis. To assess the error of Phong Deformation, we have performed a detailed truncation error analysis in §5 to establish third-order accuracy provided second-order accurate vertex gradients are available. Numerical experiments were performed to support this third-order accurate convergence rate and those of other schemes (see Figure 10 (Left)). In addition, we confirmed that Phong Deformation's fell back to second-order accuracy when insufficient neighbors were available, e.g., neighbors missing on one side (see Figure 10 (Right)). Finally, numerical estimation and visual depiction of interpolation error was conducted for an instrumented "cell" model using a pinch Kelvinlet deformation, which revealed a 2.9× reduction in ℓ_2 vertex error (see Figure 8).

7 CONCLUSION

We have introduced Phong Deformation, a simple extension of linear interpolation for embedded mesh deformation on tetrahedral meshes. The method interpolates deformation gradients of deformed tetrahedra to vertices using precomputed cell-to-vertex weights and uses them to achieve quadratic interpolation. The scheme is fast, simple to implement, robust, and achieves clear improvements over linear interpolation due to improved accuracy that results in a reduction in visual artifacts for embedded deformation.

[1]Houdini compiles VEX code with VCC to produce multithreaded C++ code.

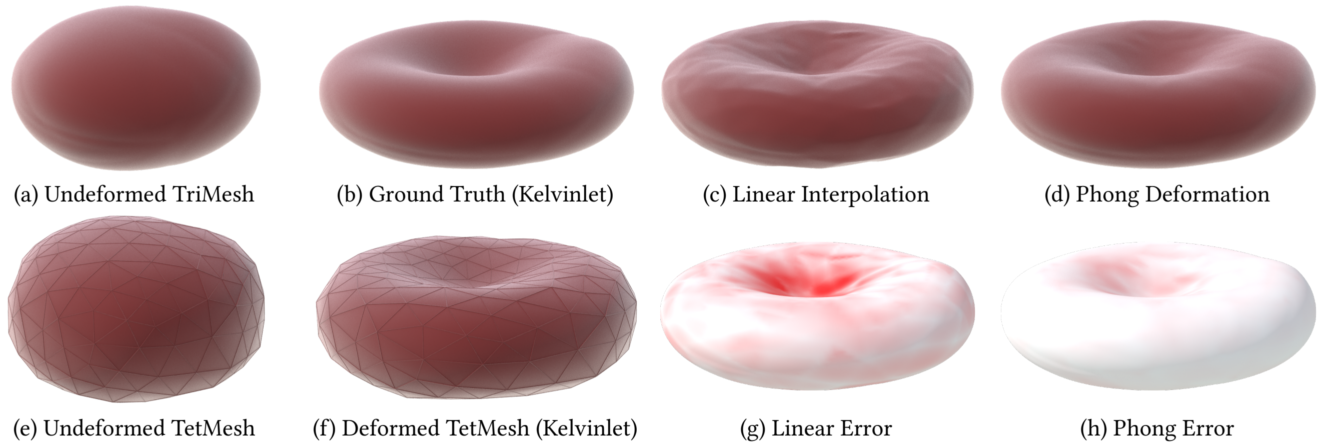

Fig. 8. **Numerical error reduction:** (a) A cell-like mesh (24,000 tri, 12,252 vtx) is embedded in (e) a coarse tetrahedral mesh (1307 tets, 341 nodes)), both of which are deformed (b,f) using an incompressible pinch Kelvinlet deformation [De Goes and James 2017, eq. 17]. We compared the embedded deformation of triangle mesh (a) resulting from the TetMesh deformation e⟶f using two methods: (c) linear interpolation exhibits visible faceting artifacts, whereas (d) Phong Deformation is clearly better. Plots of Euclidean vertex errors (g,h) (computed using the ground truth TriMesh (b)) reveal reduced errors for Phong Deformation (≈2.9× reduction in ℓ_2 error).

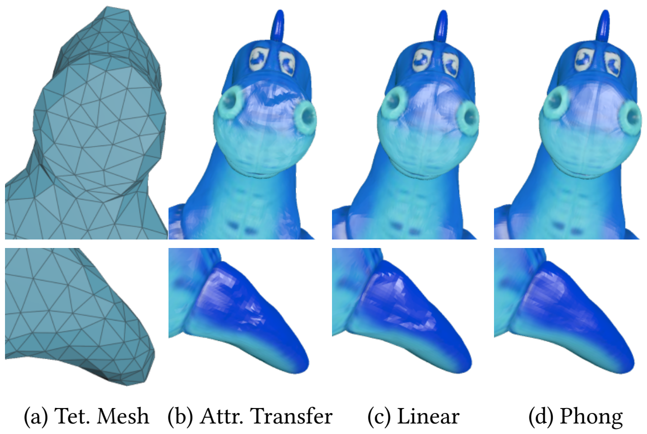

Fig. 9. **Rubber toy face plant:** An embedded triangle mesh (shown "flat shaded", 51,416 tri, 51,436 vtx) is deformed by its (a) coarse tetrahedral Houdini FEM model (3511 tets, 989 nodes) impacting an invisible ground plane. Close-up renderings of the face and fin show unsightly embedded-mesh artifacts for both (b) Houdini's default attribute transfer scheme and (c) linear interpolation, whereas (d) Phong Deformation produces fewer visual artifacts. Please see the video for related animations.

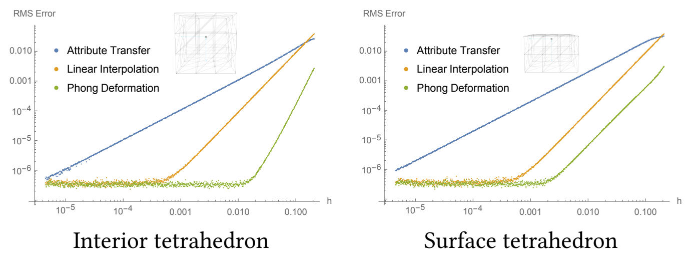

Fig. 10. **Convergence error analysis** versus h (average tetrahedra edge length) exhibits different slopes for competing methods in this log-log plot: (Left) rapid $O(h^3)$ convergence of Phong Deformation, versus the slower $O(h^2)$ behavior of Linear Interpolation, and the $O(h)$ Attribute Transfer interpolant in Houdini ("Point Deform SOP" with "radius" parameter hand tuned for best results). The test scenario consists of 25 points deformed using an analytical bending and twisting deformer, embedded in a scaled tetrahedral lattice (see inset mesh) translated to achieve isobarycentric test-point locations. Single-precision calculations in Houdini result in the noise floor seen below 10^{-6}. (Right) For surface elements (missing neighbors on top side) we observed slower convergence for Phong Deformation (only $O(h^2)$) due to deficient vertex gradients, however the error was still over 10× smaller than linear interpolation.

Limitations and Future Work. As mentioned, the accuracy of vertex deformation gradients can suffer at vertices with missing tetrahedral neighbors, and the regularized scheme may fail to achieve third-order accuracy. The quadratically deformed geometry can extend outside the deformed tetrahedron, and may be undesirable in certain cases, e.g., collisions processed with linearly embedded geometry may exhibit contact inconsistencies when displayed using the quadratic Phong deformer. The Phong Deformation scheme can be applied to triangles, tetrahedra in higher dimensions, and hexahedra (see §3.3). Future work might generalize the technique to higher dimensions, and models with varying co-dimensions.

ACKNOWLEDGMENTS

I thank the anonymous reviewers for their useful criticism and suggestions for improvement, support from the Pixar Research Group, and Ante Qu and Xinru Hua for proofreading the final manuscript.

REFERENCES

Peter Alfeld. 1984. A Discrete C^1 Interpolant for Tetrahedral Data. *The Rocky Mountain journal of mathematics* (1984), 5–16.
Peter Alfeld. 1985. Multivariate perpendicular interpolation. *SIAM J. Numer. Anal.* 22, 1 (1985), 95–106.
Peter Alfeld and Tatyana Sorokina. 2009. Two tetrahedral C1 cubic macro elements. *Journal of Approximation Theory* 157, 1 (2009), 53 – 69.
Jernej Barbič and Doug L James. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph.* 24, 3 (2005), 982–990.

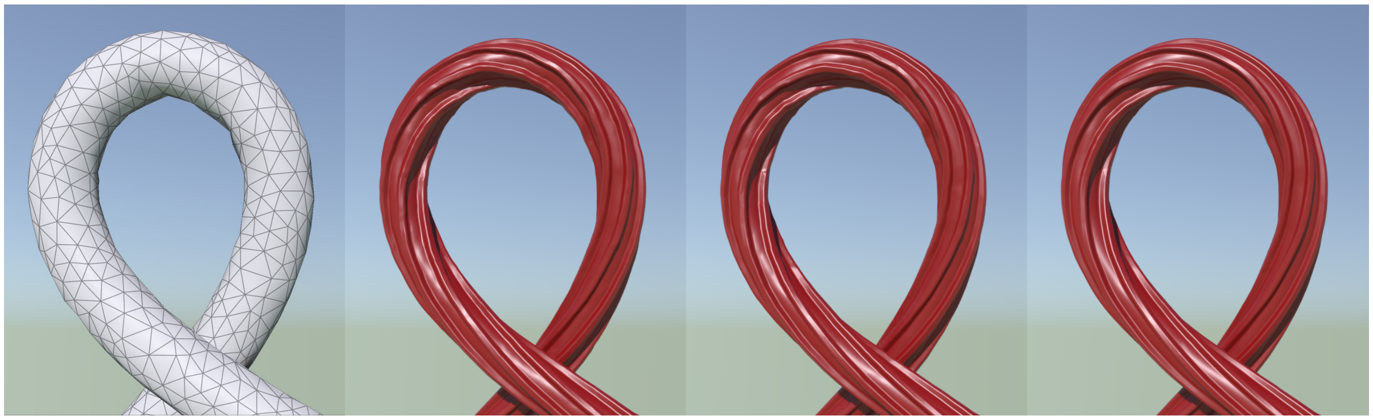

Fig. 11. **Licorice twist:** A triangulated licorice (386,954 tri, 193,477 vtx) is twisted into a simple loop by its embedding in (a) a deforming tetrahedral mesh (8077 tets, 1783 nodes). The deformed triangle mesh resulting from (b) Houdini's attribute transfer and (c) linear interpolation exhibit clear visual artifacts in this smooth-shaded render, whereas (d) Phong Deformation generally reduces visual artifacts.

Adam W. Bargteil and Elaine Cohen. 2014. Animation of Deformable Bodies with Quadratic Bézier Finite Elements. *ACM Trans. Graph.* 33, 3, Article 27 (June 2014).

Martin Berzins. 2000. A data-bounded quadratic interpolant on triangles and tetrahedra. *SIAM Journal on Scientific Computing* 22, 1 (2000), 177–197.

Tamy Boubekeur and Marc Alexa. 2008. Phong Tessellation. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 141.

Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popoviundefined. 2002. Interactive Skeleton-Driven Dynamic Deformations. *ACM Trans. Graph.* 21, 3 (July 2002), 586–593.

Praveen Chandrashekar and Ashish Garg. 2013. Vertex-centroid finite volume scheme on tetrahedral grids for conservation laws. *Computers & Mathematics with Applications* 65, 1 (2013), 58 – 74.

Ricardo Costa, Stéphane Clain, and Gaspar J. Machado. 2014. New cell–vertex reconstruction for finite volume scheme: Application to the convection–diffusion–reaction equation. *Computers & Mathematics with Applications* 68, 10 (2014), 1229 – 1249.

Fernando De Goes and Doug L. James. 2017. Regularized Kelvinlets: Sculpting Brushes Based on Fundamental Solutions of Elasticity. *ACM Trans. Graph.* 36, 4, Article 40 (July 2017), 11 pages.

Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H Barr. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proc. of ACM SIGGRAPH 2001*. ACM, 31–36.

Tony DeRose, Michael Kass, and Tien Truong. 1998. Subdivision surfaces in character animation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. 85–94.

Michael S Floater. 2003. Mean value coordinates. *Computer Aided Geometric Design* 20, 1 (2003), 19–27.

Neal Frink. 1994. Recent progress toward a three-dimensional unstructured Navier-Stokes flow solver. In *32nd Aerospace Sciences Meeting and Exhibit*. 61.

D Holmes and S Connell. 1989. Solution of the 2D Navier-Stokes equations on unstructured adaptive grids. In *9th Computational Fluid Dynamics Conference*. 1932.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral meshing in the wild. *ACM Trans. Graph.* 37, 4 (2018), 60–1.

Jin Huang, Lu Chen, Xinguo Liu, and Hujun Bao. 2008. Efficient mesh deformation using tetrahedron control mesh. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*. ACM, 241–247.

Doug L James and Dinesh K Pai. 2003. Multiresolution Green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Transactions on Graphics (TOG)* 22, 1 (2003), 47–82.

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic coordinates for character articulation. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 71.

Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 561–566.

Lily Kharevych, Patrick Mullen, Houman Owhadi, and Mathieu Desbrun. 2009. Numerical coarsening of inhomogeneous elastic materials. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–8.

François Labelle and Jonathan R. Shewchuk. 2007. Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles. *ACM Trans. Graph.* 26, 3 (July 2007), 57–es.

David Levin. 1998. The approximation power of moving least-squares. *Math. Comp.* 67, 224 (1998), 1517–1531.

Ron MacCracken and Kenneth I Joy. 1996. Free-form deformations with lattices of arbitrary topology. In *ACM SIGGRAPH Computer Graphics*. ACM, 181–188.

Nelson Max. 1999. Weights for computing vertex normals from facet normals. *Journal of Graphics Tools* 4, 2 (1999), 1–6.

Neil Molino, Zhaosheng Bao, and Ron Fedkiw. 2004. A virtual node algorithm for changing mesh topology during simulation. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 385–392.

Matthias Müller and Markus Gross. 2004. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*. 239–246.

Matthias Muller, Matthias Teschner, and Markus Gross. 2004. Physically-based simulation of objects represented by surface meshes. In *Proceedings Computer Graphics International, 2004*. IEEE, 26–33.

Oliver Nelles. 2013. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media.

Matthieu Nesme, Paul G Kry, Lenka Jeřábková, and François Faure. 2009. Preserving topology and elasticity for embedded deformable models. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 52.

Taylor Patterson, Nathan Mitchell, and Eftychios Sifakis. 2012. Simulation of complex nonlinear elastic bodies using lattice deformers. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–10.

Bui Tuong Phong. 1975. Illumination for computer generated pictures. *Commun. ACM* 18, 6 (1975), 311–317.

Russ D Rausch, John T Batina, and Henry TY Yang. 1992. Spatial adaptation of unstructured meshes for unsteady aerodynamic flow computations. *AIAA Journal* 30, 5 (1992), 1243–1251.

Alec R Rivers and Doug L James. 2007. FastLSM: Fast lattice shape matching for robust real-time deformation. In *ACM Transactions on Graphics (TOG)*, Vol. 26. ACM, 82.

Thomas W Sederberg and Scott R Parry. 1986. Free-form deformation of solid geometric models. *ACM SIGGRAPH Computer Graphics* 20, 4 (1986), 151–160.

SideFX. 2020. Houdini Engine. http://www.sidefx.com.

Robert W Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded deformation for shape manipulation. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 80.

Alex Vlachos, Jörg Peters, Chas Boyd, and Jason L Mitchell. 2001. Curved PN triangles. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*. 159–166.

Holger Wendland. 2004. *Scattered Data Approximation*. Cambridge University Press.

Martin Wicke, Daniel Ritchie, Bryan M Klingner, Sebastian Burke, Jonathan R Shewchuk, and James F O'Brien. 2010. Dynamic local remeshing for elastoplastic simulation. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 49.

Chris Wojtan and Greg Turk. 2008. Fast viscoelastic behavior with thin features. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 47.

Guo-Liang Xu. 2001. Tetrahedral C^m Interpolation by Rational Functions. *Journal of Computational Mathematics* 19, 2 (2001), 131–138.

Alexander Ženíšek. 1973. Polynomial approximation on tetrahedrons in the finite element method. *Journal of Approximation Theory* 7, 4 (1973), 334–351.

Olek C Zienkiewicz, Robert L Taylor, and Jian Z Zhu. 2005. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier.

Ziva Dynamics. 2020. Ziva VFX. https://zivadynamics.com.