Stochastic Computation of Barycentric Coordinates

FERNANDO DE GOES, Pixar Animation Studios, USA
MATHIEU DESBRUN, Inria / Ecole Polytechnique, France

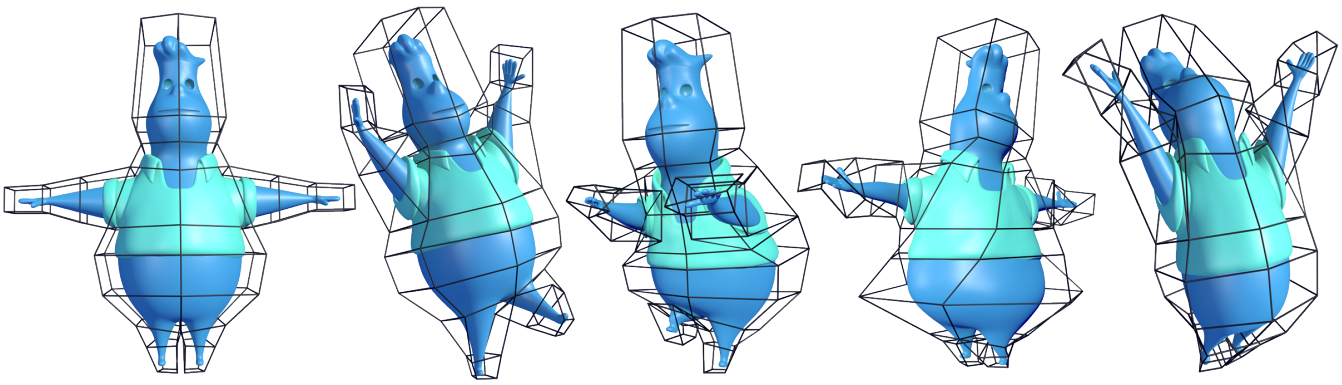

Fig. 1. **Stochastic Harmonic Coordinates:** We introduce a flexible and efficient approach to evaluate various types of barycentric coordinates stochastically, without requiring any volumetric discretization, large memory footprint, or custom solves. In this example, we reproduce harmonic coordinates [Joshi et al. 2007] on a 3D character made of multiple disconnected components (body, hair, and shirt) with a total of 30.8k points, deformed via a quadrangulated cage mesh with 88 control vertices. The left image shows the initial configuration on which we compute our stochastic harmonic coordinates, which took 2.9 s using 50 samples per query and a single denoising step, while the other images illustrate the resulting cage deformations. ©Pixar

This paper presents a practical and general approach for computing barycentric coordinates through stochastic sampling. Our key insight is a reformulation of the kernel integral defining barycentric coordinates into a weighted least-squares minimization that enables Monte Carlo integration without sacrificing linear precision. Our method can thus compute barycentric coordinates directly at the points of interest, both inside and outside the cage, using just proximity queries to the cage such as closest points and ray intersections. As a result, we can evaluate barycentric coordinates for a large variety of cage representations (from quadrangulated surface meshes to parametric curves) seamlessly, bypassing any volumetric discretization or custom solves. To address the archetypal noise induced by sample-based estimates, we also introduce a denoising scheme tailored to barycentric coordinates. We demonstrate the efficiency and flexibility of our formulation by implementing a stochastic generation of harmonic coordinates, mean-value coordinates, and positive mean-value coordinates.

CCS Concepts: • **Computing methodologies** → **Shape modeling**.

Additional Key Words and Phrases: Barycentric coordinates, cage interpolation, numerical integration, Monte Carlo methods, linear precision.

Authors' addresses: F. de Goes, fernando@pixar.com, Pixar Animation Studios, 1200 Park Avenue, Emeryville, CA, USA; M. Desbrun, mathieu.desbrun@inria.fr, Inria Saclay & LIX (IPP), 1 rue Honoré d'Estienne d'Orves, Palaiseau, France.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0730-0301/2024/7-ART42 $15.00
https://doi.org/10.1145/3658131

ACM Reference Format:
Fernando de Goes and Mathieu Desbrun. 2024. Stochastic Computation of Barycentric Coordinates. *ACM Trans. Graph.* 43, 4, Article 42 (July 2024), 13 pages. https://doi.org/10.1145/3658131

1 INTRODUCTION

The key premise behind barycentric coordinates lies in their ability to express any point in space as a weighted combination of the vertices of a cage mesh. This property, often denoted as linear precision, makes the resulting coordinates a powerful tool to extend a function defined on the cage to arbitrary points in space, with applications in a wide range of fields from geometric modeling and scattered data interpolation to computational mechanics [Hormann and Sukumar 2017]. Within the graphics community, barycentric coordinates are routinely deployed to compute free-form deformations using the cage vertices as a sparse set of manipulation handles.

While barycentric coordinates have easy-to-evaluate expressions in 2D [Meyer et al. 2002; Floater et al. 2006; Weber et al. 2009], computing similar coordinates in 3D still poses various practical challenges. Even if a few closed-form expressions for 3D coordinates exist, they assume strict restrictions on the input cage. For instance, the method proposed by Budninskiy et al. [2016] requires the cage to be a convex polytope and limits the coordinate evaluation to the interior of the cage. The popular 3D mean-value coordinates [Ju et al. 2005; Floater et al. 2005] possess an algebraic construction only when the cage is a watertight triangulation, and extensions of these coordinates involve costly computations such as numerical corrections to accommodate quadrangular cages [Thiery et al. 2018] and visibility tests to guarantee non-negative weights [Lipman et al. 2007]. Another common strategy for computing 3D coordinates is through an auxiliary volumetric discretization of the cage interior

combined with iterative solves [Joshi et al. 2007; Zhang et al. 2014; Dodik et al. 2023]. However, these volumetric approaches impose the burden of 3D meshing the inside of the cage and a large memory footprint to store the coordinates on the volumetric mesh.

In this work, we introduce a new method for efficiently computing barycentric coordinates that is agnostic to the type of cage discretization (be it a triangle or quad mesh, with boundaries or not, or even just a polygon soup), can handle query points both inside and outside the cage, and is free of any intermediate volumetric representation. At its core, our approach employs the Reproducing Kernel Particle Method (RKPM) [Liu et al. 1995] to reformulate barycentric coordinates as a weighted moving least-squares minimization defined directly on the points of interest. This new formulation enables the evaluation of 3D coordinates through numerical integration via Monte Carlo techniques without sacrificing linear precision. Additionally, our construction can easily incorporate denoising routines that eliminate the typical high-frequency noise caused by sampling strategies while still producing linear-precise coordinates. We showcase the versatility of our technique by presenting a stochastic generation of harmonic coordinates [Joshi et al. 2007], mean-value coordinates [Ju et al. 2005; Thiery et al. 2018], and positive mean-value coordinates [Lipman et al. 2007], all of them built upon simple proximity queries on the cage geometry such as closest points and ray intersections.

2 RELATED WORK

We start by providing a brief overview of prior work on barycentric coordinates, focusing exclusively on methods closely related to our formulation. We point the reader to recent surveys for more details [Floater 2015; Weber 2017; Hormann and Sukumar 2017].

Mean-Value Coordinates. Floater [2003] introduced closed-form mean-value coordinates on star-shaped 2D polygons, which was later generalized to arbitrary polygons by Hormann and Floater [2006]. Analytical expressions for mean-value coordinates in 3D were derived by Ju et al. [2005] and Floater et al. [2005] concurrently, both assuming closed triangular cages. Lipman et al. [2007] proposed positive mean-value coordinates as a variant preventing negative weights caused by cage concavities through visibility tests. More recently, Thiery et al. [2018] noticed that mean-value coordinates generated by triangulated cages can lead to severe interpolation artifacts, and devised an extension to quadrangulated cages via a custom quadrature-based correction. We instead draw inspiration from the theoretical work of Belyaev [2006], which described a general construction of barycentric coordinates as a simple modification of the Gordon-Wixon interpolation scheme [Gordon and Wixom 1974]. In particular, Belyaev [2006] showed that the (transfinite) mean-value coordinates correspond to a special case of the Gordon-Wixon scheme that averages values linearly interpolated between cage points. Our approach complements this formulation by presenting a unified numerical routine that can evaluate any transfinite barycentric kernel (including mean-value coordinates and its variants) via stochastic sampling by properly aggregating ray intersections against the cage geometry.

Harmonic Coordinates. Joshi et al. [2007] designed harmonic coordinates as a generalized barycentric coordinate that interpolates the basis function associated with each cage vertex through space by solving a Laplacian equation. While this construction offers desirable properties such as shape-awareness and non-negative coordinates, it relies on an indirect evaluation that involves 3D meshing the cage interior followed by volumetric solves. Alternatively, Martin et al. [2008] approximated harmonic coordinates by placing radial basis functions on the cage boundary via the method of fundamental solutions. Instead, Weber and Gotsman [2010] computed harmonic coordinates via a dense linear system derived through a boundary element method, but restricted to the case of simply-connected cages in 2D. More recently, Sawhney and Crane [2020] and Sugimoto et al. [2023] advocated for Monte Carlo methods to evaluate harmonic functions free of any volumetric discretization or costly solve. While the inherent noise of Monte Carlo methods can be reduced using boundary value caching [Miller et al. 2023], these schemes are inadequate for computing harmonic coordinates since the results lack linear precision even if a large number of samples is used (Figure 2 left). Moreover, these techniques assume that the harmonic functions fade to zero when evaluated outside the cage domain, thus producing exterior harmonic coordinates that fail to enforce partition of unity. With our construction, we can extend the Walk-on-Spheres algorithm [Sawhney and Crane 2020] in order to generate valid harmonic coordinates both inside and outside the cage. Additionally, our approach offers accurate gradient estimates and can retrofit existing denoising strategies to barycentric coordinates while still retaining linear precision.

Projection Methods. Instead of constructing linear-precise coordinates directly, some methods have considered projection routines that correct flawed coordinate estimates as a post-processing step. For instance, Hormann and Sukumar [2008] and Chang et al. [2023] employed statistical models as optimization objectives that project prior functions into barycentric coordinates. In contrast, we enforce linear precision by recasting the construction of barycentric coordinates based on the Reproducing Kernel Particle Method (RKPM) [Liu et al. 1995]. In graphics, RKPM has found previous applications in multi-phase [Chen et al. 2020] and fluid simulation [Westhofen et al. 2023], but we are not aware of its usage for cage interpolation. With our RKPM-based formulation, numerical approximations of barycentric coordinates can be converted into a weighted moving least-squares minimization that ensures linear precision exactly. Therefore, our approach can be seen as a more general form of projection in which kernel-based weights are introduced in order to encapsulate various types of barycentric coordinates and sampling strategies. Finally, we point out that our work can also be seen as an extension of the moving least-squares coordinates [Manson and Schaefer 2010] to arbitrary weighting functions, enabling the stochastic computation of various types of barycentric coordinates across diverse cage representations.

3 PRELIMINARIES

Before presenting our contributions, we summarize some definitions and notations necessary for our formulation. Hereafter, we adopt the

convention of using letters with Sans-serif font (e.g., u) to indicate d-sized vectors versus bold font (e.g., $\mathbf{u}$) to encode $(d+1)$-sized vectors. For more concise expressions, we also make use of homogeneous coordinates. To this end, we reserve the symbols x and y to denote points in $\mathbb{R}^d$ and encode their respective homogeneous coordinates as the $(d+1)$-sized vectors $\mathbf{x}=[\mathsf{x}\,;\,1]$ and $\mathbf{y}=[\mathsf{y}\,;\,1]$.

Cage Primitive: We define a cage primitive C in $\mathbb{R}^d$ as a list of n control vertices in general position so that their linear combinations span $\mathbb{R}^d$. For each control vertex v, we assign a position x_v and a basis function $\phi_v : C \to \mathbb{R}$. We further assume that these basis functions $\{\phi_v\}$ define a partition of unity over the cage, i.e., $\sum_v \phi_v(\mathsf{y}) = 1 \;\forall\, \mathsf{y} \in C$. Observe that our cage definition covers the typical case of simplicial cages such as 2D polygons and 3D triangle meshes, but it also provides a more general representation that detaches the cage discretization of any meshing requirement. For instance, when the cage is a triangle (resp., quad) mesh, the basis function ϕ_v is one at x_v, zero at any other control vertex, and piecewise-linear (resp., bilinear) over each cage element. Based on this cage definition, our work also applies to parametric curves, point clouds, meshes with boundaries, non-manifold geometry, or even a combination thereof.

Kernel Interpolation: Equipped with vertex-based basis functions, we can now set scalar values $\{g_v\}$ to each control vertex v of the cage C to construct a cage function $g : C \to \mathbb{R}$ through

$$g(\mathsf{y}) = \sum\nolimits_v \phi_v(\mathsf{y})\, g_v \quad \forall\, \mathsf{y} \in C. \tag{1}$$

To further interpolate the cage function g to the inside (and, possibly, the outside) of the cage domain, we introduce a kernel function $\kappa : \mathbb{R}^d \times C \to \mathbb{R}$ (with properties to be reviewed next) that determines the influence of each point y from the cage primitive C over an arbitrary point $\mathsf{x} \in \mathbb{R}^d$. We can now compute a function f that extends the cage function g to the rest of $\mathbb{R}^d$ via

$$f(\mathsf{x}) = \int_C \kappa(\mathsf{x},\mathsf{y})\, g(\mathsf{y}) d\mathsf{y} \quad \forall\, \mathsf{x} \in \mathbb{R}^d. \tag{2}$$

By substituting Eq. (1) into Eq. (2), we can rewrite the cage interpolation as a weighted sum of vertex values of the form $f(\mathsf{x}) = \sum_v \alpha_v(\mathsf{x}) g_v$, where the *coordinate function* $\alpha_v : \mathbb{R}^d \to \mathbb{R}$ extends the basis function ϕ_v for each control vertex v throughout $\mathbb{R}^d$ by accounting for the kernel influence over the cage, with

$$\alpha_v(\mathsf{x}) = \int_C \kappa(\mathsf{x},\mathsf{y})\, \phi_v(\mathsf{y}) d\mathsf{y} \quad \forall\, \mathsf{x} \in \mathbb{R}^d. \tag{3}$$

Barycentric Coordinates. The coordinate functions defined by Eq. (3) provide a proper generalization of Möbius' simplicial barycentric coordinates [Möbius 1827] to arbitrary cage primitives when two fundamental properties hold. First, the cage coordinates must be interpolant, i.e., $\alpha_v(\mathsf{y}) \equiv \phi_v(\mathsf{y})\; \forall v$ for any point $\mathsf{y} \in C$, so that the extended function f reproduces the input function g when restricted to the cage. Second, the coordinate functions must reproduce linear functions, i.e., any cage function of the form $g(\mathsf{y}) = \mathbf{a}^t \mathbf{y}$ with constant vector $\mathbf{a} \in \mathbb{R}^{d+1}$ must generate $f(\mathsf{x}) = \mathbf{a}^t \mathbf{x}$. The latter property, often denoted as *linear precision* or *linear reproduction*, implies that the coordinates must both form a partition of unity and reproduce the identity map, which can be concisely written using homogeneous coordinates as the linear constraint

$$\sum\nolimits_v \alpha_v(\mathsf{x})\, \mathbf{x}_v = \mathbf{x} \quad \forall \mathsf{x} \in \mathbb{R}^d. \tag{4}$$

This constraint limits the kernel functions suitable to define barycentric coordinates, referred to as *transfinite barycentric kernels* [Belyaev 2006]. In particular, this constraint suggests that numerical approximations of the kernel integral in Eq. (3) are likely to violate linear precision and lead to flawed barycentric coordinates. To overcome these undesirable artifacts, we introduce next a new construction of coordinate functions that guarantees linear precision.

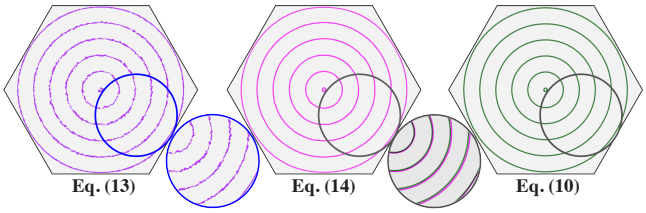

Fig. 2. **Reproduction of linear functions:** To demonstrate the improvements that our stochastic harmonic coordinates bring compared to a raw Monte Carlo interpolation [Sawhney and Crane 2020], we visualize isocontours of the distance function from the cage center for the identity map reconstructed using 10k samples per query via α_v^{MC} from Eq. (13) (left), its denoised version $\alpha_v^{\mathrm{MC},\sigma}$ from Eq. (14) (center), and our coordinates $\widehat{\alpha}_v$ from Eq. (10) (right). The high-frequency noise introduced by the Monte Carlo method ruins linear precision, leading to visual defects (left inset) with a L^2 error of 0.45% and a L^∞ error of 1.15%, both measured relative to the cage's diameter. Denoising the raw Monte-Carlo evaluations reduces the noise but shifts isocontours away from their correct locations (right inset) due to the lack of linear precision, with a L^2 error of 0.37% and a L^∞ error of 0.56%, measured once again relative to the cage's diameter. In contrast, our projection to linear-precise coordinates reproduces the identity map exactly, thus retaining evenly-spaced isocontours.

4 REFORMULATING BARYCENTRIC COORDINATES

In this section, we describe how to systematically convert possibly-flawed estimates of coordinate functions into barycentric coordinates. This step will be at the heart of our stochastic construction of barycentric coordinates, as it will ensure that numerical defects in the computation of kernel integrals are properly compensated for.

4.1 Rationale

As expressed by Eq. (3), the coordinate function $\alpha_v(\mathsf{x})$ is defined by a boundary integral that pairs a kernel function $\kappa(\mathsf{x},\mathsf{y})$ to the basis function $\phi_v(\mathsf{y})$ of a cage vertex v. However, computing this boundary integral analytically is unfeasible in most cases depending on the choice of the kernel function and the specific type of cage being used. Numerical integration offers a more practical approach to evaluate Eq. (3), but it induces oft-inevitable inaccuracies, resulting in deficient coordinates that are not linear precise (Figure 2 left). Our approach for constructing barycentric coordinates borrows from the *Reproducible Kernel Particle Method* (RKPM) [Liu et al. 1995], which is commonly used by mesh-free techniques in order to evaluate functions through a numerical approximation of a kernel integral

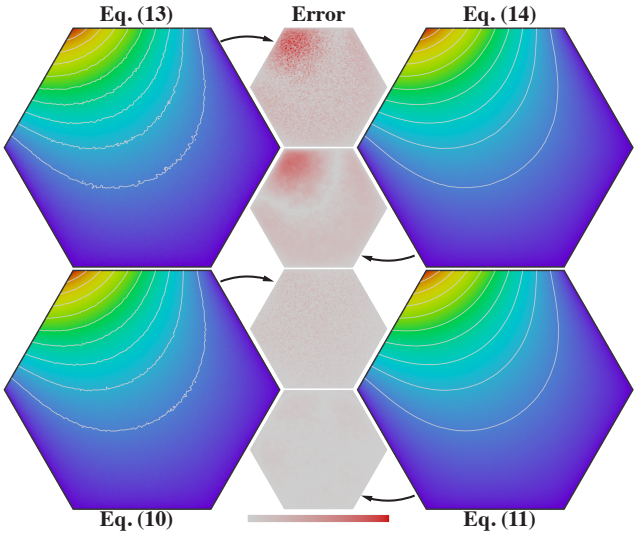

Fig. 3. **Accuracy:** For a more quantitative comparison between a raw Monte Carlo interpolation [Sawhney and Crane 2020] and our stochastic harmonic coordinates, we reuse the setup of Figure 2 and compute the difference of the resulting barycentric coordinates versus a reference solution. In the top row, we plot the coordinate function after the raw Monte Carlo evaluations from Eq. (13) (with L^2 error of 0.18% and L^∞ error of 1.12%) and after they are denoised by Eq. (14) (with L^2 error of 0.15% and L^∞ error of 0.65%). In the bottom row, we plot the Monte Carlo estimates after projection to linear-precise coordinates via Eq. (10) (with L^2 error of 0.06% and L^∞ error of 0.40%) and finally after they are denoised by Eq. (11) (with L^2 error of 0.02% and L^∞ error of 0.10%). Additionally, the middle column displays the point-wise error for each coordinate approximation, with all the errors calculated as a percentage of the cage's diameter.

while reproducing a desired sub-class of functions (typically low-order polynomials) exactly. This formalism is particularly well suited to our task at hand of computing barycentric coordinates based on sampling approximations without sacrificing linear precision.

4.2 Barycentric Coordinates through RKPM

To adapt RKPM to the construction of barycentric coordinates, we associate each control vertex v of the cage C with an auxiliary function $\Lambda_v(\mathsf{x},\mathsf{y}) = \mathbf{u}_v(\mathsf{x})^t\mathsf{y}$ parametrized by the $(d+1)$-sized vector $\mathbf{u}_v(\mathsf{x})$. Note that $\Lambda_v(\mathsf{x},\mathsf{y})$ is a separable function, defining a linear function relative to its second argument. For a given point $\mathsf{x}\in\mathbb{R}^d$, we seek a vector $\mathbf{u}_v(\mathsf{x})$ so that its auxiliary function $\Lambda_v(\mathsf{x},\mathsf{y})$ best approximates the vertex-based basis function $\phi_v(\mathsf{y})$ when evaluated on any cage point $\mathsf{y}\in C$. Following Liu et al. [1997], we compute $\mathbf{u}_v(\mathsf{x})$ through a kernel-based least-squares minimization

$$\mathbf{u}_v(\mathsf{x}) = \arg\min_{\mathbf{u}} \int_C \kappa(\mathsf{x},\mathsf{y})\,\|\mathbf{u}^t\mathsf{y} - \phi_v(\mathsf{y})\|^2 d\mathsf{y}. \tag{5}$$

Once the vector $\mathbf{u}_v(\mathsf{x})$ is found, we can then evaluate the coordinate function for the cage vertex v at the query point x as

$$\widehat{\alpha}_v(\mathsf{x}) = \Lambda_v(\mathsf{x},\mathsf{x}) = \mathbf{u}_v(\mathsf{x})^t\mathsf{x}. \tag{6}$$

Notice that we use the symbol $\widehat{\alpha}_v$ in Eq. (6) to differentiate from the kernel integral α_v defined by Eq. (3), as we will see that even if the latter is not linear precise, the former will be. It is also worth mentioning that other fitting schemes could be employed to calculate the vector $\mathbf{u}_v(\mathsf{x})$ [Fries and Matthies 2004], however, we favored the integral-based functional in Eq. (5) as it inherits the properties of moving least-squares approximation [Levin 1998] such as bounded fitting error and easy access to derivative estimates, while also being amenable to stochastic sampling as we describe next.

4.3 Numerical Evaluation

In what follows, we will consider that a series of m sample points $\{\mathsf{y}_k\}$ on the cage C has been selected for a given query point $\mathsf{x}\in\mathbb{R}^d$, where each sample y_k is accompanied by the values of its respective vertex-based basis functions $\{\phi_v(\mathsf{y}_k)\}$. Additionally, we will assume that each cage sample y_k is assigned a scalar weight w_k that estimates $\kappa(\mathsf{x},\mathsf{y})d\mathsf{y}$ measured at y_k, thus indicating the sample importance to the cage integral in Eq. (5). We will discuss how to generate cage samples and set their weights later in §5 when we go through specific examples of barycentric coordinates, since the sampling strategy depends directly on the choice of kernel function.

Equipped with these weighted samples, we can trivially discretize the cage integral in Eq. (5) and then compute the vector $\mathbf{u}_v(\mathsf{x})$ for the control vertex v at a specific query point x via

$$\mathbf{u}_v(\mathsf{x}) = \arg\min_{\mathbf{u}} \sum\nolimits_k w_k \|\mathbf{u}^t\mathsf{y}_k - \phi_v(\mathsf{y}_k)\|^2. \tag{7}$$

The unique solution to Eq. (7) is of the form $\mathbf{u}_v(\mathsf{x}) = \mathbf{M}(\mathsf{x})^{-1}\mathbf{m}_v(\mathsf{x})$, where the $(d+1)$-sized vector $\mathbf{m}_v(\mathsf{x})$ is given as

$$\mathbf{m}_v(\mathsf{x}) = \sum\nolimits_k w_k\,\phi_v(\mathsf{y}_k)\,\mathsf{y}_k, \tag{8}$$

while the $(d+1)\times(d+1)$ symmetric matrix $\mathbf{M}(\mathsf{x})$ represents the second-order moment of the weighted cage samples defined as

$$\mathbf{M}(\mathsf{x}) = \sum\nolimits_k w_k\,\mathsf{y}_k\,\mathsf{y}_k^t. \tag{9}$$

Therefore, we can expand Eq. (6) and express our barycentric coordinate $\widehat{\alpha}_v(\mathsf{x})$ for the cage vertex v at the query point x as

$$\boxed{\widehat{\alpha}_v(\mathsf{x}) = \mathbf{x}^t\mathbf{M}(\mathsf{x})^{-1}\mathbf{m}_v(\mathsf{x}).} \tag{10}$$

Importantly, by utilizing the same set of samples to compute $\mathbf{u}_v(\mathsf{x})$ for every cage vertex v at a fixed query point x, we can easily show that $\sum_v \mathbf{m}_v(\mathsf{x})\mathbf{x}_v^t = \mathbf{M}(\mathsf{x})$ and, consequently, $\sum_v \mathbf{u}_v(\mathsf{x})\mathbf{x}_v^t = \mathbf{I}$. Using these identities, we can then verify that our construction of barycentric coordinates given by Eq. (10) satisfies Eq. (4), thus resulting in linear-precise coordinates despite the use of a numerical evaluation.

4.4 Denoising

So far we have described how to generate linear-precise coordinates numerically by aggregating sample estimates from a single query point. However, sample-based integration such as Monte Carlo methods tends to introduce high-frequency noise to the resulting coordinates, thus conflicting with the spatial smoothness expected from barycentric coordinates. Conventional sampling schemes alleviate these artifacts by averaging point-wise evaluations around nearby query points. Unfortunately, denoising each vertex-based coordinate individually is inappropriate due to the linear-precision condition in Eq. (4) that couples the coordinates together (Figure 2).

To address this issue, we retrofit the construction of denoising schemes tailored to barycentric coordinates. To this end, we consider a smoothing operator $\sigma_{\mathsf{x}} : \mathbb{R}^d \to \mathbb{R}$ is given for any query point $\mathsf{x} \in \mathbb{R}^d$. We also make the mild assumption that the smoothing operator is normalized, i.e., $\int_{\mathbb{R}^d} \sigma_{\mathsf{x}}(\mathsf{y}) d\mathsf{y} = 1 \ \forall \mathsf{x} \in \mathbb{R}^d$. We then generalize our barycentric coordinates defined in Eq. (6) by convolving σ_x against the RKPM auxiliary function Λ_v, yielding

$$\widehat{\alpha}_v(\mathsf{x}) = \int_{\mathbb{R}^d} \sigma_{\mathsf{x}}(\mathsf{y}) \Lambda_v(\mathsf{y}, \mathsf{x}) d\mathsf{y} = \left(\int_{\mathbb{R}^d} \sigma_{\mathsf{x}}(\mathsf{y}) \mathbf{u}_v(\mathsf{y}) d\mathsf{y} \right)^t \mathbf{x}, \qquad (11)$$

where the last equality uses the fact that Λ_v is a separable function. Note that, if σ_{x} is simply a Dirac function centered at x, then Eq. (11) simplifies to Eq. (6). Importantly, by reusing the identity $\sum_v \mathbf{u}_v(\mathsf{x}) \mathbf{x}_v^t = \mathbf{I}$ discussed in §4.3, we can easily verify that our filtered coordinates satisfy Eq. (4), thus defining linear-precise coordinates.

In addition to linear precision, we also need to ensure that the denoising process keeps the coordinates interpolatory. To achieve this goal, we detect any query point x that lies on the cage C and then constraint the smoothing convolution so that $\widehat{\alpha}_v(\mathsf{x}) = \phi_v(\mathsf{x})$ for every control vertex v. Since the values for the vertex-based basis functions are known for any cage query, we can impose this constraint by a simple change of variables: we first introduce a new variable $\mathsf{g}_v(\mathsf{x})$ that defines a d-sized vector and then express the vector $\mathbf{u}_v(\mathsf{x})$ as $[\mathsf{g}_v(\mathsf{x}) \, ; \, \phi_v(\mathsf{x}) - \mathsf{g}_v(\mathsf{x})^t \mathsf{x}]$. With this split encoding, we can filter each vector $\mathbf{u}_v(\mathsf{x})$ at any cage query x by smoothing only its sub-vector $\mathsf{g}_v(\mathsf{x})$, while correcting its last coefficient in order to restore the interpolation property. As a result, our denoising method ensures interpolatory coordinates even if the input kernel function is not interpolatory. Based on this construction, we can now compute denoising of the sample-based evaluation of barycentric coordinates by simply applying any existing smoothing operator σ_{x} to each vertex-based vector $\mathbf{u}_v$ individually and the results are guaranteed to produce valid barycentric coordinates.

In graphics applications, barycentric coordinates are often evaluated on query points that come from specific geometric primitives such as polygonal meshes, point clouds, or grid nodes. Based on this observation, we selected the heat kernel associated with the query primitives as the smoothing operator σ_{x}, since the heat kernel has discretizations available for various types of geometric primitives. For instance, when the queries correspond to a uniform grid, the smoothing operator is simply a Gaussian convolution. In the case of queries forming a polygonal surface mesh, we compute the smoothing operator as the symmetric matrix $(\mathbf{A} + t\mathbf{L})^{-1}\mathbf{A}$, which amounts to an implicit Laplacian smoothing [Desbrun et al. 1999] with time-step t, where the diagonal mass matrix $\mathbf{A}$ indicates the local area of each query point on the surface mesh and $\mathbf{L}$ is the sparse Laplacian matrix with zero Neumann boundary condition, for which we implemented the polygonal discretization proposed by de Goes et al. [2020]. In the general case, we can discretize the heat kernel by handling the entire set of queries seen as a point cloud following the work of Sharp and Crane [2020]. Finally, we set the heat kernel time-step t to the mean squared distance between neighboring queries scaled by a function s of the number of samples m so that fewer samples imply more smoothing. This scaling function was experimentally chosen as $s(m) = \max(1, 5 - \log_{10}(m))$.

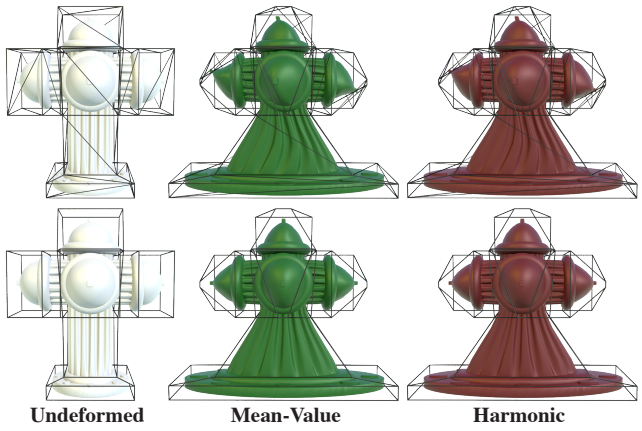

Fig. 4. **Triangulated vs. Quadrangulated Cage:** Our stochastic approach can seamlessly handle cage meshes made of triangles and/or quadrilaterals, thus extending the work of Thiery and Boubekeur [2022] to a broader class of barycentric coordinates. Symmetry artifacts (top), caused by a triangulated cage generated by adding quad diagonals to a quad mesh, are evident with mean-value coordinates [Ju et al. 2005; Thiery et al. 2018] (middle) or harmonic coordinates [Joshi et al. 2007] (right), but all disappear when the same coordinates are evaluated directly on the quad cage (bottom). These results were generated with a sample count $m = 50$ and one denoising step.

4.5 Discussion

Our method presented above has many unique properties besides guaranteeing linear-precise and interpolatory coordinates. First, it is general in the sense that it accommodates a number of cage representations, including meshes with boundaries and imperfect geometries, for which there are only rare examples of known barycentric coordinates [Yan and Schaefer 2019]. Second, it can handle any kernel function, be it transfinite or not, relying only on sample estimates. As we will see in §5, our formulation also offers a new means to evaluate many of the state-of-the-art barycentric coordinates, including harmonic coordinates [Joshi et al. 2007] and positive mean-value coordinates [Lipman et al. 2007]. Moreover, our sampling-based generation of barycentric coordinates directly at the points of interest is easy to parallelize, free of any volumetric discretization. Even our denoising scheme based on linear smoothing operators can be parallelized, making the computation of barycentric coordinates per query point efficient. At last, our approach provides a unified construction of barycentric coordinates for query points both inside and outside a cage independent of the choice of kernel function.

Projection Operator. While we derived barycentric coordinates through a sampling-based evaluation, it is worth noticing that this approach amounts to a discrete version of a continuous counterpart, where sums of sample terms are replaced by integrals. This continuous formulation, sketched in Appendix A, allows any kernel function to be projected to a transfinite barycentric kernel. We can even deduce from this continuous derivation that our barycentric coordinates in Eq. (10) can be expanded into Eq. (15), thus revealing that our construction offers a *correction term* that compensates for any lack of linear precision, hence forming a proper projection of coordinate functions to linear-precise coordinates. Compared to

prior projection techniques [Hormann and Sukumar 2008; Chang et al. 2023], our scheme does not impose restrictions on the type of kernel or prior function.

Gradient of Barycentric Coordinates. Furthermore, our variational definition of the $(d+1)$-sized vector $\mathbf{u}_v(\mathsf{x})$ in Eq. (5) as a linear regression of the vertex-based basis function ϕ_v over the cage C leads to an interesting side effect: the resulting vector $\mathbf{u}_v(\mathsf{x})$ contains a weighted least-squares approximation of the gradient vector $\nabla\alpha_v(\mathsf{x})$. Indeed, the solution of Eq. (5) can be re-expressed as $\mathbf{u}_v(\mathsf{x}) = \left[\nabla\widehat{\alpha}_v(\mathsf{x})\,;\, \widehat{\alpha}_v(\mathsf{x}) - \nabla\widehat{\alpha}_v(\mathsf{x})^t\mathsf{x}\right]$, i.e., $\mathbf{u}_v(\mathsf{x})$ provides in its first d coefficients an estimate $\nabla\widehat{\alpha}_v(\mathsf{x})$ of the gradient vector $\nabla\alpha_v(\mathsf{x})$, while its last coefficient quantifies the residual between the barycentric coordinate $\widehat{\alpha}_v(\mathsf{x})$ and its first-order approximation. Figure 6 shows an example of the gradient estimate produced by our method both inside and outside the cage domain.

Queries at the Cage. In general, barycentric coordinates are only C_0 at the cage, with gradient vector $\nabla\alpha_v$ matching the gradient of the vertex-based basis function ϕ_v tangent to the cage, but ill-defined along the cage normal. By leveraging our interpretation of the vector $\mathbf{u}_v(\mathsf{x})$, we can now calculate smooth gradient vectors for barycentric coordinates even when the query point x lies on the cage geometry C. To this end, we initialize the gradient vector at a cage point x by perturbing x away from the cage and computing the vector $\mathbf{u}_v(\mathsf{x})$ on this shifted location. We then filter the approximated $\mathbf{u}_v(\mathsf{x})$ using our denoising scheme, which smooths the gradient estimates spatially while enforcing that $\widehat{\alpha}_v(\mathsf{x}) = \phi_v(\mathsf{x})$ for every control vertex v at any cage query x, as previously detailed in §4.4.

Limitations. Finally, we point out theoretical limitations to our formulation. First, the matrix $\mathbf{M}(\mathsf{x})$ must be invertible to be able to evaluate our barycentric coordinates. Yet, this may not be the case if the input (continuous) kernel function has poles, i.e., it integrates to zero at any point in space. Too few samples in our numerical integration may also cause $\mathbf{M}(\mathsf{x})$ to be singular, but all our experiments suggest that 20 samples or more are sufficient to produce

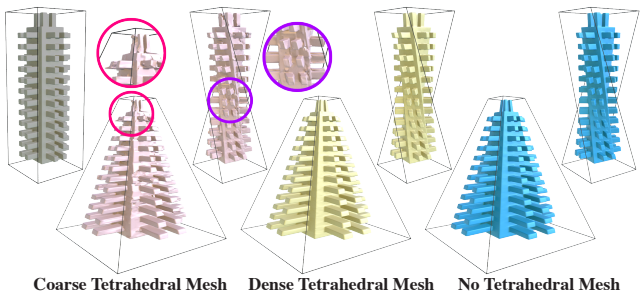

Fig. 5. **Volume Discretization:** The method proposed by Joshi et al. [2007] exhibits artifacts when the tetrahedral discretization of the cage interior used to evaluate harmonic coordinates is too coarse compared to the details of the binding geometry (left, for a tet-mesh with 369 vertices). These artifacts are resolved through finer discretizations (center, for a tet-mesh with 28k vertices) at the price of greater memory footprint and computational times. Instead, our approach (with $m=50$ and one denoising step) produces similar results free of any volumetric meshing requirements (right).

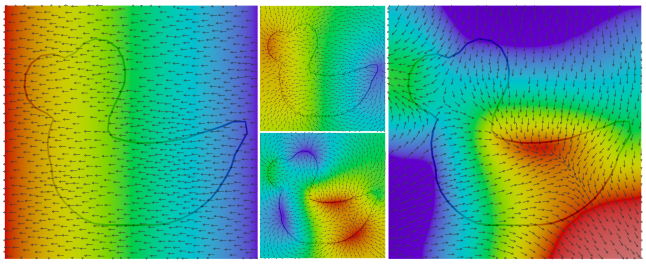

Fig. 6. **Gradient Estimation:** Our method provides accurate estimations of barycentric coordinates and their gradients at arbitrary query points, be they inside or outside the cage. In this example, we use stochastic harmonic coordinates to interpolate linear (left) and sinusoidal (right) color attributes from a cage polyline to an enclosing background grid of 12k points, using 1k samples per query and one denoising step. For comparison, results obtained via a direct application of Sawhney and Crane [2020] for these same inputs exhibit inaccurate color interpolations and noisy gradients (center column).

valid results. Second, our method can lead to negative coordinates even if the kernel function is supposed to be positive, due to either the linear-precision projection or the denoising step. For example, our stochastic approximation of harmonic coordinates [Joshi et al. 2007] and positive mean-value coordinates [Lipman et al. 2007] may present negative weights for interior queries in the vicinity of the cage boundary, depending on the number m of samples used. A possible workaround is to add $\mathbf{u}_v(\mathsf{x})^t\mathbf{x} \geq 0$ as a linear inequality constraint to the denoising so that the coordinates are always forced to be positive, but a different denoising solver must be used then. At last, we note that our denoising step introduces smoothness for both the coordinates and their gradients, however, we can not claim any specific order of continuity for the resulting coordinates since our approach evaluates them at discrete locations.

5 EXAMPLES

We now present how to reproduce various types of barycentric coordinates using our sampling-based construction. For each example of barycentric coordinates, we will describe a sampling strategy that produces a list of m sample points $\{\mathsf{y}_k\}$ on the cage C, with each sample y_k containing the values of its vertex-based basis function $\{\phi_v(\mathsf{y}_k)\}$. Following the setup outlined in §4.3, we will also detail the process of computing a scalar weight w_k for each sample point y_k relative to its originating query point x.

5.1 Harmonic Coordinates

Since the Laplacian kernel is not known in closed-form for arbitrary domains, we adopt the Walk-on-Spheres (WoS) algorithm [Muller 1956; Sawhney and Crane 2020] which consists in generating samples through a Monte Carlo method with uniform weights to approximate the Dirichlet problem for the Laplacian equation. Given a query point x, we run the WoS algorithm that simulates a random walk from x to the cage C in order to generate a boundary sample point y_k which we associate with a weight $w_k = 1$. This Monte-Carlo-based estimate is performed for a cage thickness (indicating the tolerance to decide convergence) set to 0.1% of the length of the diagonal of the cage's bounding box, and for a maximum number of

steps per random walk set to 20. If a call to WoS does not converge to a cage location, we just discard this random walk. This simple procedure provides an approximation of harmonic coordinates [Joshi et al. 2007] for all query points, with an accuracy in $O(1/\sqrt{m})$ for m calls to WoS. Note that even for thousands of calls, the spatial smoothness of these estimates is limited, but our denoising removes the high-frequency noise present in the raw results (Figure 3). It is also worth pointing out that the WoS-based results are known to converge to harmonic functions *in expectation* [Binder and Braverman 2012], thus zeroing out the contribution of our RKPM-based projection gradually as the sample count m increases.

5.2 Mean-Value Coordinates

Our generation of mean-value coordinates builds on the Gordon-Wixon interpolation scheme [Gordon and Wixom 1974], which was generalized to non-convex domains by Belyaev [2006]. We complement this interpolation scheme by proposing a Monte Carlo evaluation that traces rays stochastically from a query point x against the cage C. More precisely, for the k-th ray test, we choose a random direction to shoot from x and compute every intersection $\{\mathsf{y}_{k_i}\}$ of this ray against C, which we then sort in ascending distance $d_{k_i} = \|\mathsf{y}_{k_i} - \mathsf{x}\|$ from x. We finally create a cage sample for each y_{k_i} with an associated weight $w_{k_i} = (-1)^{i+1}/d_{k_i}$ so that intersections with odd indices have positive weights, while even indices receive negative weights. When a ray does not intersect the cage, we simply ignore it. We repeat tracing rays until m samples are collected per query point, or until m random ray tests are performed, whichever is reached first. Just like the harmonic coordinates case, our estimations of the mean-value coordinates based on Belyaev [2006] are then denoised to provide our final barycentric coordinates.

5.3 Positive Mean-Value Coordinates

Positive mean-value coordinates were originally computed by extracting the visible part of the cage C w.r.t. the query point x and using only this subdomain to evaluate mean-value coordinates [Lipman et al. 2007]. However, generating the visible part of a cage for every query point in 3D is computationally expensive. Here again, our stochastic construction of coordinates finds another practical application by, instead, considering these positive coordinates as a special case of weighted mean-value coordinates via a generalized Gordon-Wixon interpolation scheme [Belyaev 2006].

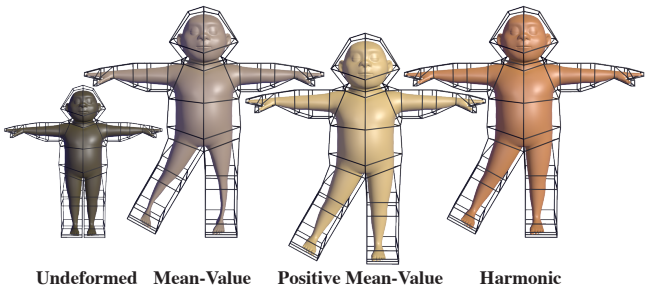

Fig. 7. **Comparing Coordinates:** Our formulation is general enough to provide stochastic evaluations of (quadrangulated) mean-value coordinates [Ju et al. 2005; Thiery et al. 2018] (left-center), positive mean-value coordinates [Lipman et al. 2007] (right-center), and harmonic coordinates [Joshi et al. 2007] (right). This example is purposely replicating the test originally shown in Joshi et al. [2007] illustrating the artifacts caused by negative mean-value coordinates (here, thinning of the legs), which are resolved by using either harmonic coordinates or positive mean-value coordinates, both computed using $m = 50$ and a single denoising step. ©Pixar

The weighted mean-value coordinates are based on the mean-value kernel multiplied by a weighting function ω satisfying the following condition:

$$\int_C \omega(\mathsf{x},\mathsf{y})\mathsf{v}(\mathsf{y})\, d\mathsf{y} = 0 \quad \forall \mathsf{x} \in \mathbb{R}^d, \tag{12}$$

where $\mathsf{v}(\mathsf{y})$ is the direction of a ray emanating from the query x and passing through the cage point y, i.e., $\mathsf{v}(\mathsf{y}) = (\mathsf{y} - \mathsf{x})/\|\mathsf{y} - \mathsf{x}\|$. For example, the weighting for mean-value coordinates used above is simply $\omega(\mathsf{x}, \mathsf{y}_{k_i}) = (-1)^{i+1}$, which verifies Eq. (12) due to symmetry. We can easily adapt our stochastic construction to weighted mean-value coordinates by still following the procedure described in §5.2, but now every intersection of a random ray from x towards C is associated with a sample weight $w_{k_i} = \omega(\mathsf{x}, \mathsf{y}_{k_i})/d_{k_i}$.

We can thus evaluate positive mean-value coordinates by defining a weighting function ω that accounts for the cage visibility from a query x while holding Eq. (12). To design this custom weighting function, we exploit two simple observations about ray intersections with a closed cage. First, an odd number of intersections along a ray indicates that the starting point x is inside the cage C, while an even number signifies that x is outside. Second, the first ray intersection with a cage is visible from x. Therefore, if the number of intersections for a random ray is odd, we create a *single* sample located at the first hit with an associated weight $w_k = 1/d_k$, disregarding all further hits along the ray. Conversely, if the number of intersections is even, we only create samples for the *first two hits* using the same weights as in the mean-value case, that is, the first sample y_{k_1} is given weight $w_{k_1} = 1/d_{k_1}$, while the second sample y_{k_2} is given weight $w_{k_2} = -1/d_{k_2}$. Note that our choice of weights ensures Eq. (12) for closed cages. Finally, our coordinate denoising allows us not only to suppress high-frequency noise, but to remove the lack of smoothness of the original positive mean-value coordinates near reflex vertices without having to refine the cage (Figure 11).

5.4 Other Coordinates

As discussed by Belyaev [2006], the weighted version of mean-value coordinates can be used to compute other known types of barycentric coordinates, such as the Wachspress-Warren coordinates [Warren et al. 2007], and our method can reproduce these cases by simply adjusting the sample weights. The moving least-squares coordinates [Manson and Schaefer 2010] is also simple to reproduce with our scheme, since their formulation corresponds to Eq. (5) with a non-transfinite kernel of the form $\kappa(\mathsf{x},\mathsf{y}) = 1/\|\mathsf{x} - \mathsf{y}\|^{2a}$, where $a > 0$ is a user-specific parameter. When the cage is a 2D polygon, closed-form expressions are available for the moving least-squares coordinates, but quadrature rules are necessary to compute the 3D version of these coordinates from triangular cages [Manson and Schaefer 2010, §7]. Therefore, we can interpret their numerical approximation as a sampling strategy that generates samples

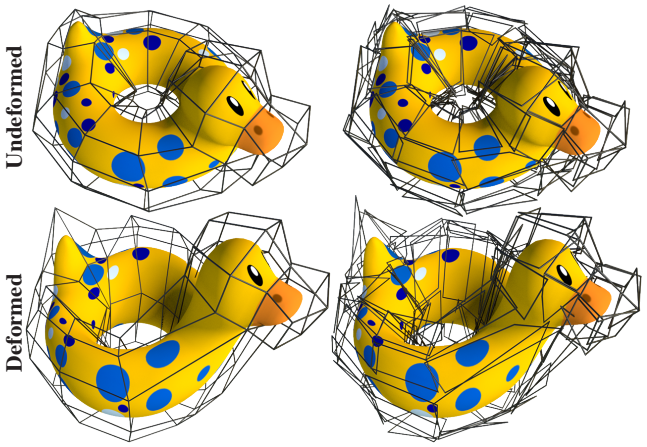

Fig. 8. **Cage Mesh vs. Cage Soup:** Our stochastic approach is well suited for computing barycentric coordinates (here, positive mean-value coordinates) from a cage made of a soup of disconnected polygons (right), producing results qualitatively similar to those generated from a manifold cage mesh (left), both using $m=50$ and one denoising step. Also, notice that our method can handle non-simply connected cages, as on this torus-like shape.

only once over the entire cage and reuses this sample set for every input query point. It is also worth noting that the mean-value coordinates as presented by Ju et al. [2005] could be computed using query-independent samples. To do so, the sample weights should be set based on the kernel function $\kappa(\mathsf{x},\mathsf{y})=\mathsf{n}(\mathsf{y})^t(\mathsf{y}-\mathsf{x})/\|\mathsf{y}-\mathsf{x}\|^{d+1}$, where $\mathsf{n}(\mathsf{y})$ is the outward normal at the cage point y. However, requiring sample normals prohibits some cage discretizations such as 3D curves, unoriented point clouds, and non-orientable surfaces.

6 RESULTS

We now go over a few implementation details and present a series of experiments showcasing the accuracy, efficiency, and generality of our stochastic approach to barycentric coordinates. We also point the reader to our supplemental video, which provides additional comparisons and visualizations of our results.

Cage Representation. As discussed in §5, a core component of our construction is the computation of closest-point projections and ray intersections against the cage primitive C. In our implementation, we handle various forms of cage discretization by encoding them into an axis-aligned bounding volume hierarchy and changing just the proximity calculation to its atomic elements. For instance, when using a triangular cage, we evaluate the proximity queries to individual triangles analytically. For cages made of (possibly non-planar) quads, we define the shape of each quad as a bilinear patch and compute ray intersections using the algorithm described by Reshetov [2019], while closest points are found through a custom-made solver presented in Appendix B. In the case of parametric curves, we convert each curve into a dense polyline and compute the proximity queries to each line segment. Finally, proximity queries on point clouds can be performed following Madan and Levin [2022].

Selection of Random Rays. To reduce the number of ray tests with zero intersections, we restrict the random selection of rays to be within the visibility cone of the query point x against the bounding box of the cage C. This is particularly valuable for the stochastic evaluation of mean-value coordinates and their variants when the query points are far outside the cage. In the case of cages made of a sparse network of 3D curves, we adopt a different strategy that consists in picking first a random point y on the curve network, then setting the ray direction from the query point x to y, ensuring that the ray test will produce at least one hit on the cage.

Accuracy. To analyze the accuracy of our method, we compare our results to a Monte Carlo integration of Eq. (3) that reuses our stochastically generated weighted samples via

$$\alpha_v^{\mathrm{MC}}(\mathsf{x}) = \left(\sum\nolimits_k w_k \phi_v(\mathsf{y}_k)\right) / \left(\sum\nolimits_k w_k\right), \tag{13}$$

where the denominator is a normalization term enforcing partition of unity. For completeness, we also include comparisons to a smoothed version of Eq. (13) computed by applying our denoising operator described in §4.4 to each coordinate approximation $\alpha_v^{\mathrm{MC}}(\mathsf{x})$ separately (thus losing linear precision), which can be expressed as

$$\alpha_v^{\mathrm{MC},\sigma}(\mathsf{x}) = \int_{\mathbb{R}^d} \sigma_{\mathsf{x}}(\mathsf{y})\alpha_v^{\mathrm{MC}}(\mathsf{y})d\mathsf{y}. \tag{14}$$

In Figure 3, we use pseudo-colors to display the harmonic coordinates associated with a single cage vertex over a 2D mesh produced by our method versus Eq. (13) and Eq. (14), followed by error plots that quantify the difference of these results to a baseline solution obtained by implementing Joshi et al. [2007]. In this example, our approach reduced both the maximum and the mean point-wise error

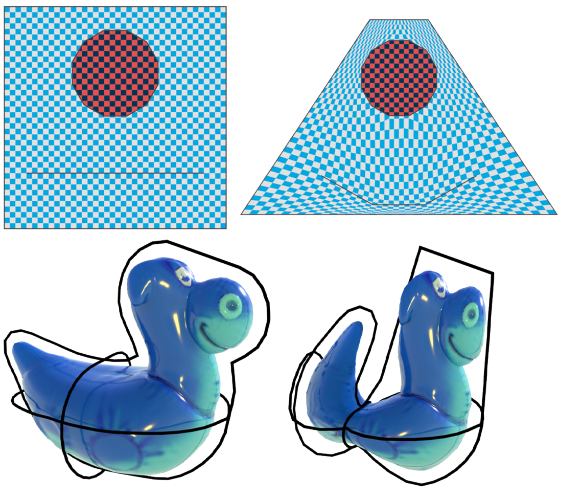

Fig. 9. **Different Cages:** Our stochastic construction applies even to cages with multiple components or cages defined by curve networks. When we compute 2D harmonic coordinates using $m=1\mathrm{k}$ and one denoising step for a cage made of an outer square, an inner circle, and a line, deforming the outer square and the line keeps the region inside the inner circle unaffected (top). For a 3D curve-network cage, manipulating the few curves is enough to induce an intuitive spatial deformation via our stochastic harmonic coordinates using $m=50$ and one denoising step (bottom).

by a factor of 3 before denoising, and by a factor of 10 after denoising despite the use of a high sample count of 10k per query. Note that approximating harmonic coordinates through Eq. (13) boils down to the method of Sawhney and Crane [2020] without further alterations, thus explaining the noise artifacts found in the top-left image. Figure 11 shows the results for three types of barycentric coordinates computed on a 2D non-convex cage with two sample counts, confirming that our results are qualitatively similar to reference solutions (left column) with error decaying as the number of samples increases. We also point out that our denoising step smooths the kinks of the positive mean-value coordinates around cage concavities, instead of relying on a post-processing subdivision as proposed by Lipman et al. [2007]. In our supplemental video, we provide qualitative and quantitative comparisons for our 3D experiments, including statistics for the deformation differences between our results and baseline solutions. In particular, we observed empirically throughout these 3D tests that $m=50$ is a reasonable default in practice, offering efficiency without sacrificing visual quality.

2D Experiments. In Figure 2, we illustrate that the lack of linear precision caused by naive numerical integration of barycentric coordinates leads to visual artifacts even when reconstructing the identity map. Figure 6 showcases that our method improves the estimate of gradient vectors for functions generated through cage interpolation. In particular, our construction offers a linear extrapolation of the cage values in contrast to the fading behavior produced by Sawhney and Crane [2020]. We show in Figure 10 the 2D harmonic coordinates computed on a closed versus open cage, demonstrating the generality of our approach. At last, Figure 9 (left) exemplifies how our method handles cages with multiple parts, generating deformations bounded within each cage element (see the supplemental video for a short animated sequence).

3D Experiments. We discuss next the results produced by our method on various 3D scenarios, all of them computed using 50 samples per query and a single denoising step. Figure 1 showcases several poses of a 3D character produced by our stochastic version of harmonic coordinates using a quadrangulated cage mesh. In Figure 7, we reproduce the test originally presented by Joshi et al. [2007] exemplifying the impact of negative mean-value coordinates within concave regions. In addition to harmonic coordinates, our approach confirms that positive mean-value coordinates can also resolve these deformation artifacts. Similarly, we repeat in Figure 4 the test originally performed by Thiery et al. [2018] exhibiting the artifacts caused by an asymmetric triangular cage versus the symmetric results obtained from a quadrangulated cage. Observe that our technique enables the usage of quad cages for both harmonic and mean-value coordinates. In Figure 5, we show that the original implementation of harmonic coordinates [Joshi et al. 2007] can lead to visual defects when the resolution of the volumetric discretization is not sufficient to capture the details of the query geometry. This is in sharp contrast to our approach which evaluates harmonic coordinates directly at the points of interest, without the need for a volumetric discretization. Figure 8 demonstrates that our method produces similar deformations for a non-simply connected surface driven either by a manifold mesh or by a polygon soup. Finally, Figure 9 (right) illustrates a harmonic deformation generated by a sparse network of 3D curves.

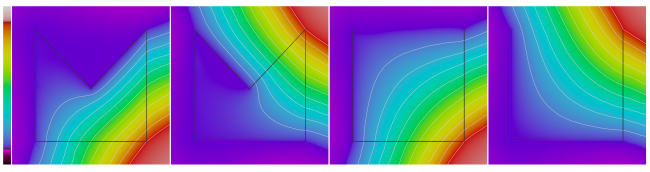

Fig. 10. **Open Cages:** Our stochastic evaluation of barycentric coordinates applies as is to cages with boundaries. The harmonic coordinates produced by our method on a closed cage in 2D (left) change dramatically when the cage is open through the removal of two cage edges (right). Here, we visualize the before and after coordinates for two control vertices computed on a mesh with 2.5k points using 1k samples per query.

Table 1. **Timing:** Wall clock time in seconds spent by our method to evaluate and denoise 3D barycentric coordinates using 50 samples per query, all measured on 3.49 GHz Apple M2 with 8 cores. We also report the coordinate type, the number of queries, and the number of cage vertices.

Example	Coordinates	# Queries	# Verts	Evaluation	Denoising
Fig. 1	harmonic	30.8k	88	2.5	0.4
Fig. 4	harmonic	39k	48	3.2	0.5
Fig. 4	mean-value	39k	48	0.3	0.5
Fig. 5	harmonic	60.8k	8	2.2	0.9
Fig. 7	harmonic	11k	138	1.1	0.4
Fig. 7	mean-value	11k	138	0.1	0.4
Fig. 7	pos. mean-value	11k	138	0.1	0.4
Fig. 8	pos. mean-value	5k	98	0.1	0.2
Fig. 9	harmonic	8k	145	0.8	0.2

Performance. We implemented our technique as a C++ plugin to Houdini [Side Effects 2024] with query-based evaluations multithreaded via Intel TBB [Reinders 2007]. Table 1 presents timings of our prototype for all 3D experiments clocked on a 3.49 GHz Apple M2 with 8 cores. In our supplemental video, we also report the performance of our method using $m=50$, 500, and 5000 followed by one denoising step for the meshes from Figure 1, which reveals a linear growth of the computational cost as the number of samples increases. In general, we observed that computing harmonic coordinates is more costly than any of the variants of the mean-value coordinates, since the former generates one sample through multiple closest-point projections, versus a single ray test per sample for the latter. This cost discrepancy can be reduced by tweaking the parameters of WoS algorithm that control the termination of each random walk, thus invoking fewer closest-point calls. Still, computing harmonic coordinates with our method is more efficient than Joshi et al. [2007], producing barycentric coordinates for queries both inside and outside the cage without spending time and memory on calculating and storing the harmonic coordinates on a volumetric mesh. For instance, our own implementation of Joshi et al. [2007] is over ten times slower than our stochastic evaluation of harmonic coordinates for the example shown in Figure 1, while producing visually similar deformations (see supplemental video). Profiling the

execution of our implementation of the original harmonic coordinates shows that 90% of the cost is due to the coordinate solve on the volumetric mesh, which we generated using Houdini's tetrahedral meshing tool for a resolution set to three times the mean edge length of the deforming meshes. We also measured the performance of our technique for every model provided by Thiery et al. [2018], which took less than 1 ms per query. While our timings are comparable to those reported by Thiery et al. [2018] and slower than evaluating the analytical expressions presented by Ju et al. [2005] and Floater et al. [2005], only our approach is able to support non-watertight cages, incorporate positive mean-value coordinates, and estimate gradient vectors at no extra cost. Therefore, our construction can be seen as a more practical approach that unifies the generation of different types of barycentric coordinates to various cage discretizations and evaluated at arbitrary locations, at the expense of approximated yet visually plausible results. Since all the pieces of our method are parallelizable, we believe further performance improvements can be achieved by implementing our algorithm directly on the GPU. Finally, we point out that our approach is most advantageous in 3D, since 2D coordinates are often analytical or just involve a linear solve with a computational cost similar to our denoising step.

7 CONCLUSION

We have introduced a stochastic approach for the construction of barycentric coordinates, where the usual kernel integral formulation of barycentric coordinates is turned into a weighted least-squares minimization, thus enabling numerical integration that preserves linear precision. Our contribution offers a unified evaluation method to barycentric coordinates for cages as varied as polygon soups, open meshes, or curve networks. Repeated ray intersections against the cage or Walk-on-Sphere runs over the cage are all that is needed to provide the resulting linear-precise coordinates. To resolve the noise caused by stochastic sampling, we also devised a denoising strategy that can filter each vertex-based coordinate separately while retaining their core properties. Moreover, our method supports the computation of barycentric coordinates and their gradients for both inside and outside query points, and for any kernel function. We demonstrated the efficiency and flexibility of our method by approximating key known coordinates, namely, harmonic coordinates, mean-value coordinates, and positive mean-value coordinates.

As future work, it would be interesting to extend our RKPM-based formulation to non-interpolatory methods, such as Green coordinates [Lipman et al. 2008] or Somigliana coordinates [Chen et al. 2023], but this may need new sampling strategies. Since our approach provides gradient estimates, we could also exploit our construction to evaluate deformation gradients and compute shape deformations based on distortion minimization, similar to [Ben-Chen et al. 2009]. Another interesting direction would be to design new types of barycentric coordinates by either projecting custom kernels onto the space of transfinite barycentric kernels, or as a way to incorporate additional application-specific features. Adapting variance reduction techniques (see, e.g., [Sawhney and Crane 2020, §4.1]) to barycentric coordinates is also desirable in order to control the coordinate accuracy as a function of the number of samples. Finally, our method is well suited to compute free-form deformations driven by implicit functions using recent advances on proximity queries for implicit representations [Sharp and Jacobson 2022].

ACKNOWLEDGMENTS

We are grateful to William Sheffler and Michael Nieves for modeling the cage meshes used in Figures 1 and 7 respectively, and to Andrew Butts and Mark Meyer for feedback. We also thank Thiery et al. [2018] for sharing their code and the 3D models used in Figures 4 and 5 (FireHydrant and SpikyBox). The 3D model used in Figure 8 is courtesy of Keenan Crane. Finally, MD acknowledges the support of Ansys, Adobe Research, and the MediTwin consortium, as well as a Choose France Inria chair.

REFERENCES

Alexander Belyaev. 2006. On Transfinite Barycentric Coordinates. In *Symposium on Geometry Processing*. https://doi.org/10.2312/SGP/SGP06/089-099

Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. 2009. Variational Harmonic Maps for Space Deformation. *ACM Trans. Graph.* 28, 3, Article 34 (2009).

Ilia Binder and Mark Braverman. 2012. The rate of convergence of the Walk on Spheres Algorithm. *Geometric and Functional Analysis* 22 (aug 2012), 558–587. https://doi.org/10.1007/s00039-012-0161-z

Max Budninskiy, Beibei Liu, Yiying Tong, and Mathieu Desbrun. 2016. Power Coordinates: A Geometric Construction of Barycentric Coordinates on Convex Polytopes. *ACM Trans. Graph.* 35, 6, Article 241 (2016).

Qingjun Chang, Chongyang Deng, and Kai Hormann. 2023. Maximum likelihood coordinates. *Computer Graphics Forum* 42, 5 (Aug. 2023), Article e14908, 13 pages. Proceedings of SGP.

Jiong Chen, Fernando De Goes, and Mathieu Desbrun. 2023. Somigliana Coordinates: An Elasticity-Derived Approach for Cage Deformation. In *ACM SIGGRAPH Conference Proceedings*. Article 52, 8 pages. https://doi.org/10.1145/3588432.3591519

Xiao-Song Chen, Chen-Feng Li, Geng-Chen Cao, Yun-Tao Jiang, and Shi-Min Hu. 2020. A Moving Least Square Reproducing Kernel Particle Method for Unified Multiphase Continuum Simulation. *ACM Trans. Graph.* 39, 6, Article 176 (nov 2020). https://doi.org/10.1145/3414685.3417809

Fernando de Goes, Andrew Butts, and Mathieu Desbrun. 2020. Discrete Differential Operators on Polygonal Meshes. *ACM Trans. Graph.* 39, 4, Article 110 (aug 2020).

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. 1999. Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. 317–324. https://doi.org/10.1145/311535.311576

Ana Dodik, Oded Stein, Vincent Sitzmann, and Justin Solomon. 2023. Variational Barycentric Coordinates. *ACM Trans. Graph.* 42, 6, Article 255 (2023), 16 pages. https://doi.org/10.1145/3618403

Michael S. Floater. 2003. Mean Value Coordinates. *Comput. Aided Geom. Design* 20, 1 (2003), 19–27.

Michael S. Floater. 2015. Generalized barycentric coordinates and applications. *Acta Numerica* 24 (2015), 161–214.

Michael S. Floater, Kai Hormann, and Géza Kós. 2006. A general construction of barycentric coordinates over convex polygons. *Advances in Computational Mathematics* 24 (2006), 311–331. https://doi.org/10.1007/s10444-004-7611-6

Michael S. Floater, Géza Kós, and Martin Reimers. 2005. Mean value coordinates in 3D. *Computer Aided Geometric Design* 22, 7 (2005), 623–631. https://doi.org/10.1016/j.cagd.2005.06.004

Thomas-Peter Fries and Hermann G. Matthies. 2004. Classification and Overview of Meshfree Methods. *Informatik-Berichte der Technischen Universität Braunschweig* 2003-03 (2004). https://doi.org/10.24355/dbbs.084-200511080100-465

William J. Gordon and James A. Wixom. 1974. Pseudo-Harmonic Interpolation on Convex Domains. *SIAM J. Numer. Anal.* 11, 5 (1974), 909–933. https://doi.org/10.1137/0711072

Kai Hormann and Michael S. Floater. 2006. Mean Value Coordinates for Arbitrary Planar Polygons. *ACM Trans. Graph.* 25, 4 (oct 2006), 1424–1441. https://doi.org/10.1145/1183287.1183295

Kai Hormann and N. Sukumar. 2008. Maximum Entropy Coordinates for Arbitrary Polytopes. In *Symp. Geo. Proc.* 1513–1520.

Kai Hormann and N. Sukumar. 2017. *Generalized barycentric coordinates in computer graphics and computational mechanics.* CRC press.

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic Coordinates for Character Articulation. *ACM Trans. Graph.* 26, 3, Article 71 (2007). https://doi.org/10.1145/1276377.1276466

Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean Value Coordinates for Closed Triangular Meshes. *ACM Trans. Graph.* 24, 3 (jul 2005), 561–566. https://doi.org/10.

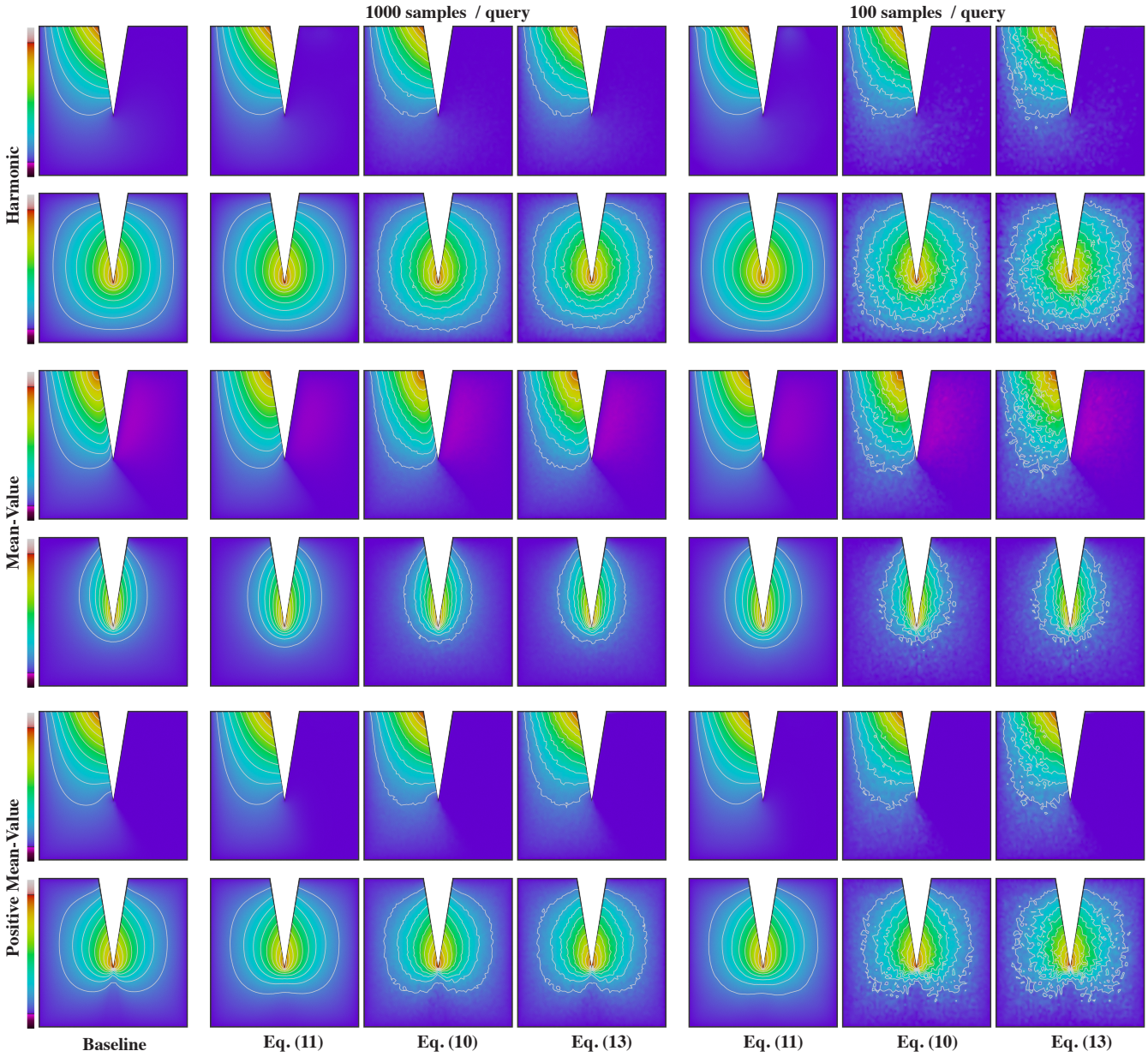

Fig. 11. **Gallery:** We use pseudo-colors to showcase the barycentric coordinates for two control vertices generated by our stochastic method on 3k queries inside a non-convex cage in 2D. The two top rows display our results for harmonic coordinates, while the center and bottom rows present results for mean-value coordinates and positive mean-value coordinates, respectively. For each row, we compare a reference solution (left) versus our results computed using 1k samples per query (middle) and 100 samples per query (right). Additionally, for each number of samples per query, we include our denoised coordinates produced by Eq. (11), our linear-precise evaluation defined by Eq. (10), and a raw Monte Carlo approximation computed via Eq.(13). Note that our denoised coordinates are visually similar to the baselines, even when computed with a small sample count per query, while the largest point-wise error is concentrated near the reflex control vertex, especially in the case of positive mean-value coordinates.

1145/1073204.1073229

David Levin. 1998. The approximation power of moving least-squares. *Math. Comp.* 67, 224 (oct 1998), 1517–1531. https://doi.org/10.1090/S0025-5718-98-00974-0

Yaron Lipman, Johannes Kopf, Daniel Cohen-Or, and David Levin. 2007. GPU-assisted Positive Mean Value Coordinates for Mesh Deformations. In *Symposium on Geometry Processing*.

Yaron Lipman, David Levin, and Daniel Cohen-Or. 2008. Green Coordinates. *ACM Trans. Graph.* 27, 3 (2008), 1–10.

Wing-Kam Liu, Sukky Jun, and Yifei Zhang. 1995. Reproducing kernel particle methods. *Int. J. Num. Meth. Fluids* 20, 8-9 (1995), 1081–1106. https://doi.org/10.1002/fld.1650200824

Wing-Kam Liu, Shaofan Li, and Ted Belytschko. 1997. Moving least-square reproducing kernel methods (I) Methodology and convergence. *Computer Methods in Applied Mechanics and Engineering* 143, 1 (1997), 113–154.
Abhishek Madan and David I. W. Levin. 2022. Fast Evaluation of Smooth Distance Constraints on Co-Dimensional Geometry. *ACM Trans. Graph.* 41, 4, Article 68 (jul 2022), 17 pages. https://doi.org/10.1145/3528223.3530093
Josiah Manson and Scott Schaefer. 2010. Moving Least Squares Coordinates. *Computer Graphics Forum* 29, 5 (2010), 1517–1524. https://doi.org/10.1111/j.1467-8659.2010.01760.x
Sebastian Martin, Peter Kaufmann, Mario Botsch, Martin Wicke, and Markus Gross. 2008. Polyhedral Finite Elements Using Harmonic Basis Functions. *Computer Graphics Forum* 27, 5 (2008), 1521–1529.
Mark Meyer, Alan Barr, Haeyoung Lee, and Mathieu Desbrun. 2002. Generalized Barycentric Coordinates on Irregular Polygons. *J. Graph. Tools* 7, 1 (2002), 13–22.
Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2023. Boundary Value Caching for Walk on Spheres. *ACM Trans. Graph.* 42, 4 (2023).
August Ferdinand Möbius. 1827. *Der Barycentrische Calcul : ein neues Hülfsmittel zur analytischen Behandlung der Geometrie.* Leipzig, Verlag von Johann Ambrosius Barth.
Mervin E. Muller. 1956. Some Continuous Monte Carlo Methods for the Dirichlet Problem. *The Annals of Mathematical Statistics* 27, 3 (1956), 569–589. https://doi.org/10.1214/aoms/1177728169
James Reinders. 2007. *Intel Threading Building Blocks.* O'Reilly & Associates, Inc.
Alexander Reshetov. 2019. *Cool Patches: A Geometric Approach to Ray/Bilinear Patch Intersections.* 95–109 pages.
Rohan Sawhney and Keenan Crane. 2020. Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains. *ACM Trans. Graph.* 39, 4 (2020).
Nicholas Sharp and Keenan Crane. 2020. A Laplacian for Nonmanifold Triangle Meshes. *Computer Graphics Forum (SGP)* 39, 5 (2020).
Nicholas Sharp and Alec Jacobson. 2022. Spelunking the Deep: Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis. *ACM Trans. Graph.* 41, 4, Article 107 (jul 2022), 16 pages.
Side Effects. 2024. Houdini Engine. http://www.sidefx.com.
Ryusuke Sugimoto, Terry Chen, Yiti Jiang, Christopher Batty, and Toshiya Hachisuka. 2023. A Practical Walk-on-Boundary Method for Boundary Value Problems. *ACM Trans. Graph.* 42, 4, Article 81 (jul 2023), 16 pages. https://doi.org/10.1145/3592109
Jean-Marc Thiery and Tamy Boubekeur. 2022. Green Coordinates for Triquad Cages in 3D. In *SIGGRAPH Asia Conference Papers.* Article 38.
Jean-Marc Thiery, Pooran Memari, and Tamy Boubekeur. 2018. Mean Value Coordinates for Quad Cages in 3D. *ACM Trans. Graph.* 37, 6, Article 229 (2018).
Joe Warren, Scott Schaefer, Anil Hirani, and Mathieu Desbrun. 2007. Barycentric Coordinates for Convex Sets. *Adv. Comput. Math.* 27, 3 (2007), 319–338.
Ofir Weber. 2017. Planar Shape Deformation. In *Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics*, Kai Hormann and N. Sukumar (Eds.). CRC Press, Chapter 7, 109–133.
Ofir Weber, Mirela Ben-Chen, and Craig Gotsman. 2009. Complex Barycentric Coordinates with Applications to Planar Shape Deformation. *Comput. Graph. Forum* 28, 2 (2009).
Ofir Weber and Craig Gotsman. 2010. Controllable Conformal Maps for Shape Deformation and Interpolation. In *ACM SIGGRAPH Proceedings.* Article 78.
Lukas Westhofen, Stefan Jeske, and Jan Bender. 2023. A Comparison of Linear Consistent Correction Methods for First-Order SPH Derivatives. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 3, Article 48 (2023), 20 pages.
Zhipei Yan and Scott Schaefer. 2019. A Family of Barycentric Coordinates for Co-Dimension 1 Manifolds with Simplicial Facets. *Computer Graphics Forum* 38, 5 (2019), 75–83. https://doi.org/10.1111/cgf.13790
Juyong Zhang, Bailin Deng, Zishun Liu, Giuseppe Patanè, Sofien Bouaziz, Kai Hormann, and Ligang Liu. 2014. Local Barycentric Coordinates. *ACM Trans. Graph.* 33, 6, Article 188 (nov 2014), 12 pages. https://doi.org/10.1145/2661229.2661255

A RKPM-CORRECTED COORDINATES

In this section, we provide additional insights on our approach described in §4 by starting from the minimization in Eq. (5) and without ever discretizing the integrals involving the kernel function. This continuous counterpart of our derivation offers a better understanding of why our novel means to evaluate barycentric coordinates is in fact a projection onto linear-precise coordinates. Moreover, we present a closed-form expression for the resulting coordinates without using homogeneous coordinates in order to reveal how the projection ensuring linear precision is achieved.

Continuous derivation. Since the variation of the functional in Eq. (5) w.r.t. $\mathbf{u}$ is $\int_C \kappa(\mathrm{x},\mathrm{y})\,\mathbf{y}(\mathbf{u}^t\mathbf{y}-\phi_v(\mathrm{y}))\,d\mathrm{y}$, the vector $\mathbf{u}_v(\mathrm{x})$ that minimizes Eq. (5) also takes the form $\mathbf{u}_v(\mathrm{x}) = \mathbf{M}(\mathrm{x})^{-1}\mathbf{m}_v(\mathrm{x})$, but now with the vertex-based vector $\mathbf{m}_v(\mathrm{x})$ written as

$$\mathbf{m}_v(\mathrm{x}) = \int_C \kappa(\mathrm{x},\mathrm{y})\,\phi_v(\mathrm{y})\,\mathbf{y}\,d\mathrm{y}$$

and the the second-order moment matrix $\mathbf{M}(\mathrm{x})$ defined as

$$\mathbf{M}(\mathrm{x}) = \int_C \kappa(\mathrm{x},\mathrm{y})\,\mathbf{y}\,\mathbf{y}^t\,d\mathrm{y}.$$

Observe that these expressions are clearly continuous versions of Eqs. (8) and (9), respectively.

Linear precision. Just like in the discretized version presented in §4.3, one can easily prove the following property for the vertex-based vectors $\{\mathbf{u}_v(\mathrm{x})\}$ for any query point x not lying on C:

$$\begin{aligned}\sum\nolimits_v \mathbf{u}_v(\mathrm{x})\mathbf{x}_v^t &= \mathbf{M}(\mathrm{x})^{-1}\left(\sum\nolimits_v \int_C \kappa(\mathrm{x},\mathrm{y})\phi_v(\mathrm{y})\,\mathbf{y}\,\mathbf{x}_v^t\,d\mathrm{y}\right)\\ &= \mathbf{M}(\mathrm{x})^{-1}\int_C \kappa(\mathrm{x},\mathrm{y})\mathbf{y}\left(\sum\nolimits_v \phi_v(\mathrm{y})\mathbf{x}_v\right)^t d\mathrm{y}\\ &= \mathbf{M}(\mathrm{x})^{-1}\int_C \kappa(\mathrm{x},\mathrm{y})\mathbf{y}\mathbf{y}^t d\mathrm{y} = \mathbf{I}.\end{aligned}$$

By expressing our coordinate functions in terms of the vector $\mathbf{u}_v$ as defined by Eq. (6), the final proof of linear precision for the resulting coordinates is then straightforward:

$$\sum\nolimits_v \widehat{\alpha}_v(\mathrm{x})\mathbf{x}_v = \sum\nolimits_v \mathbf{x}_v\big(\mathbf{u}_v(\mathrm{x})^t\mathbf{x}\big) = \left(\sum\nolimits_v \mathbf{u}_v(\mathrm{x})\,\mathbf{x}_v^t\right)^t\mathbf{x} = \mathbf{x}.$$

RKPM-based correction. To further detail how our approach corrects coordinate functions that are not linear-precise, we introduce the definition of the first three kernel-based moment terms:

$$\begin{cases}\gamma(\mathrm{x}) = \displaystyle\int_C \kappa(\mathrm{x},\mathrm{y})d\mathrm{y},\\ \mathrm{c}(\mathrm{x}) = \dfrac{1}{\gamma(\mathrm{x})}\displaystyle\int_C \kappa(\mathrm{x},\mathrm{y})\,\mathrm{y}\,d\mathrm{y},\\ \mathrm{C}(\mathrm{x}) = \dfrac{1}{\gamma(\mathrm{x})}\displaystyle\int_C \kappa(\mathrm{x},\mathrm{y})\,[\mathrm{y}-\mathrm{c}(\mathrm{x})]\,[\mathrm{y}-\mathrm{c}(\mathrm{x})]^t\,d\mathrm{y}.\end{cases}$$

With these terms, we can expand the matrix $\mathbf{M}(\mathrm{x})$ into

$$\mathbf{M}(\mathrm{x}) = \gamma(\mathrm{x})\begin{bmatrix}\mathrm{C}(\mathrm{x})+\mathrm{c}(\mathrm{x})\mathrm{c}(\mathrm{x})^t & \mathrm{c}(\mathrm{x})\\ \mathrm{c}(\mathrm{x})^t & 1\end{bmatrix}.$$

Its inverse matrix is thus:

$$\mathbf{M}(\mathrm{x})^{-1} = \frac{1}{\gamma(\mathrm{x})}\begin{bmatrix}\mathrm{C}(\mathrm{x})^{-1} & -\mathrm{C}(\mathrm{x})^{-1}\mathrm{c}(\mathrm{x})\\ -\mathrm{c}(\mathrm{x})^t\mathrm{C}(\mathrm{x})^{-1} & 1+\mathrm{c}(\mathrm{x})^t\mathrm{C}(\mathrm{x})^{-1}\mathrm{c}(\mathrm{x})\end{bmatrix}.$$

We can also rewrite the vector $\mathbf{m}_v(\mathrm{x})$ as $\alpha_v(\mathrm{x})[\mathrm{a}_v(\mathrm{x});1]$, which uses $\alpha_v(\mathrm{x})$ given by Eq. (3) and defines the d-sized vertex-based vector

$$\mathrm{a}_v(\mathrm{x}) = \frac{1}{\alpha_v(\mathrm{x})}\int_C \kappa(\mathrm{x},\mathrm{y})\phi_v(\mathrm{x})\,\mathrm{y}\,d\mathrm{y}.$$

Finally, the continuous version of our barycentric coordinate $\widehat{\alpha}_v(\mathrm{x})$ for the cage vertex v defined by Eq. (6) can be expanded from the

possibly-flawed coordinate function $\alpha_v(\mathrm{x})$ into

$$\widehat{\alpha}_v(\mathrm{x}) = \frac{\alpha_v(\mathrm{x})}{\gamma(\mathrm{x})}\left(1 + [\mathrm{a}_v(\mathrm{x}) - \mathrm{c}(\mathrm{x})]^t\, \mathrm{C}(\mathrm{x})^{-1}\, [\mathrm{x} - \mathrm{c}(\mathrm{x})]\right). \quad (15)$$

Given that the term $\alpha_v(\mathrm{x})/\gamma(\mathrm{x})$ simply corresponds to normalized coordinates derived from the kernel function, we thus confirm that our approach adds a correction to the kernel integral that enforces linear precision. Note that, if the kernel is a transfinite function, then $\gamma(\mathrm{x}) = 1$ and $\mathrm{c}(\mathrm{x}) = \mathrm{x}$, implying that $\widehat{\alpha}_v(\mathrm{x}) \equiv \alpha_v(\mathrm{x})$. Our correction thus represents a *projection* to linear-precise coordinates. Consequently, our formulation can be used to return linear-precise coordinates from coordinate functions of an arbitrary kernel that may not be linear-precise, or to resolve numerical artifacts introduced by a numerical approximation of the kernel integration.

B CLOSEST POINT TO A BILINEAR PATCH

We denote the corners of a (possibly non-planar) quadrilateral element by the 3D points $\{\mathrm{q}_0, \mathrm{q}_1, \mathrm{q}_2, \mathrm{q}_3\}$ sorted counter-clockwise. We then define the bilinear interpolation within the quad parametrized by the values $0 \leq u, v \leq 1$ as

$$\mathrm{q}(u, v) = (1 - v)\left[(1 - u)\mathrm{q}_0 + u\mathrm{q}_1\right] + v\left[(1 - u)\mathrm{q}_3 + u\mathrm{q}_2\right].$$

Given a query point x, our goal is to compute the parameters $(\overline{u}, \overline{v})$ that minimizes $\|\mathrm{q}(u, v) - \mathrm{x}\|^2$. The optimal parameters can be computed analytically in the special case of planar quads, as described by Floater [2015, Theorem 1]. For non-planar quads, one approach to find these parameters is through the smooth projection presented by Thiery et al. [2018, Algorithm 1]. We instead propose an alternative algorithm that uses just projections to line segments. We initialize $(\overline{u}, \overline{v})$ by finding the closest point to x along the boundary of the quad, which amounts to computing the closest point to four line segments. We then refine this initial guess by alternating the update of each parameter $\overline{u}$ and $\overline{v}$. Consider w.l.o.g. that the last iteration updated the parameter $\overline{u}$, we then optimize the parameter $\overline{v}$ by computing the closest point to the line segment $\mathrm{q}(\overline{u}, v)$ defined by fixing $\overline{u}$. We repeat these alternating steps until no more progress is made. In our tests, this procedure converges after 5 to 10 iterations.