Mean value coordinates for quad cages in 3D

JEAN-MARC THIERY, LTCI, Télécom ParisTech, Paris-Saclay University
POORAN MEMARI, LIX, Ecole Polytechnique, CNRS, Paris-Saclay University
TAMY BOUBEKEUR, LTCI, Télécom ParisTech, Paris-Saclay University

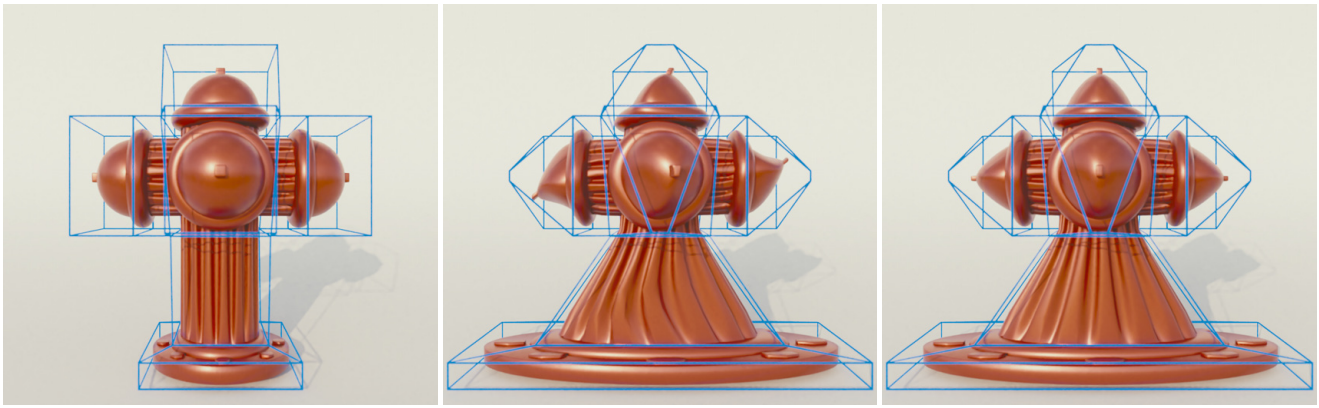


Fig. 1. We propose a generalized coordinate system for tri-quad cages. Left: embedding a high-resolution mesh (red) within the volume of a coarse cage (blue). Middle: propagating the cage deformation using Mean Value Coordinates (MVC) after cage triangulation. Right: using our new Quad MVC with the quad cage. Suppressing cage triangulation and directly using quads to construct a coordinate system provides higher quality cage edits propagation, less unpredictable distortions and reproduces the natural symmetries present in the coarse quad layout – the standard polygon mesh format in the modeling industry.

Space coordinates offer an elegant, scalable and versatile framework to propagate (multi-)scalar functions from the boundary vertices of a 3-manifold, often called a *cage*, within its volume. These generalizations of the barycentric coordinate system have progressively expanded the range of eligible cages to triangle and planar polygon surface meshes with arbitrary topology, concave regions and a spatially-varying sampling ratio, while preserving a smooth diffusion of the prescribed on-surface functions. In spite of their potential for major computer graphics applications such as freeform deformation or volume texturing, current space coordinate systems have only found a moderate impact in applications. This follows from the constraint of having only triangles in the cage most of the time, while many application scenarios favor arbitrary (non-planar) quad meshes for their ability to align the surface structure with features and to naturally cope with anisotropic sampling. In order to use space coordinates with arbitrary quad cages currently, one must triangulate them, which results in large propagation distortion. Instead, we propose a generalization of a popular coordinate system – Mean Value Coordinates – to quad and tri-quad cages, bridging the gap between high-quality coarse meshing and volume diffusion through space coordinates. Our method can process non-planar quads, comes with a closed-form solution free from global optimization and reproduces the expected behavior of Mean Value Coordinates, namely smoothness within the cage volume and continuity everywhere. As a result, we show how these coordinates compare favorably to classical space coordinates on triangulated quad cages, in particular for freeform deformation.

CCS Concepts: • **Computing methodologies → Mesh geometry models**; *Volumetric models*;

Additional Key Words and Phrases: Space coordinates, mean value coordinates, tri-quad cages, quad cages, free-form modeling

ACM Reference Format:
Jean-Marc Thiery, Pooran Memari, and Tamy Boubekeur. 2018. Mean value coordinates for quad cages in 3D. *ACM Trans. Graph.* 37, 6, Article 229 (November 2018), 14 pages. https://doi.org/10.1145/3272127.3275063

1 INTRODUCTION

A number of computer graphics problems are addressed by extrapolating in the 3D space a solution prescribed on a lower dimensional one, such as skeletons (1D) or cages (2D). This allows for simple control with few degrees of freedom, and finds applications in freeform deformation, texturing and animation for instance. In particular, space coordinates [Joshi et al. 2007; Ju et al. 2005a; Lipman et al. 2008] express each point of the inner volume of the closed 2-manifold triangle mesh – referred to as a *cage* – as a linear combination of the cage vertices, which allows to smoothly propagate functions modeled on the cage to the inner volume. Depending on the application, the particular nature of the function may range from color (texturing) to scalar weights (rigging), through 3D displacement (freeform deformation) functions. This last modeling

This work is partially supported by the French National Research Agengy (ANR) under grant ANR 16-LCV2-0009-01 ALLEGORI and by BPI France, under grant PAPAYA.
Authors' addresses: Jean-Marc Thiery, LTCI, Télécom ParisTech, Paris-Saclay University, Paris, France, jthiery@telecom-paristech.fr; Pooran Memari, LIX, Ecole Polytechnique, CNRS, Paris-Saclay University, Palaiseau, France, memari@lix.polytechnique.fr; Tamy Boubekeur, LTCI, Télécom ParisTech, Paris-Saclay University, Paris, France, tamy.boubekeur@telecom-paristech.fr.

0730-0301/2018/11-ART229 $15.00
https://doi.org/10.1145/3272127.3275063

application is often available in 3D modeling packages: in this scenario, a low-resolution cage is used as a coarse, high-level control structure for interactive deformation of complex objects. Compared to other control structures such as surface handles or skeletons, cages have strengths and weaknesses. On one hand they offer a global shape control, where each vertex of the cage has an impact on the whole volume and the cage resolution shall remain low to preserve a reasonable amount of coordinates to manipulate. On the other hand, cages offer a smooth volumetric control over the entire shape, allowing for quick, high-quality large deformations with no constraints on the target shape topology, geometry nor manifoldness – cages can help deforming dense point clouds and polygon soups. This makes cages complementary to deformation skeletons, variational surface deformation and displacement painting.

When modeling with cages, the user's prime focus is typically on creating an initial cage that properly reflects the intended editing, contains the high-resolution target shape, is tight, captures the main topological and geometrical structures of this shape and remains as coarse as possible [Jacobson et al. 2014]. To that end, the user can use either semi-interactive specialized modeling systems [Cadleron and Boubekeur 2017; Le and Deng 2017] or state-of-the-art polygonal (re)meshing tools. Usually, at such coarse levels, the ideal cage takes the form of a quad mesh, for the ability of such polygons to align with anistropic features and reflect symmetries. As in numerous computer graphics scenarios, e.g., animation or subdivision modeling, coarse quad layouts are the de facto standard in the industry.

Unfortunately, space coordinate systems are defined for triangle meshes. This implies that the cage must be triangulated before embedding the target shape and starting editing, which introduces significant artifacts as can be seen in Fig. 1. In particular, the resulting cage-to-volume function propagation is highly biased by the artificial edge introduced to split each quad, breaking up symmetries and introducing unpredictable distortions in the deformation or volume texture mapping for example.

We address this problem with a new kind of cage coordinates, designed for quad cages instead of triangle ones, and help to bridge the gap between industry standards, where quads heavily dominate, and space coordinates, which have a potentially high impact for any surface-based volumetric control. We base our work on the *Mean Value coordinates* (MVC) introduced by Ju. et al. [2005a], as they offer a closed-form solution which, to date, remains the most convenient generalization of barycentric coordinateto 3D surface meshes (Sec. 2). At the core of our method (Sec. 3), we propose (i) a new geometric characterization of **valid quads** to assess the quality of the input cage automatically, (ii) a new **smooth projection operator** that robustly project a 3D point onto quad (u, v)−domains, even for highly non-planar ones, together with (iii) a new **adaptive integration scheme** which takes the form of a spatially-varying Riemann summation where samples of the (u, v)−domain are denser next to the evaluation point. Our new coordinate system provides smooth cage-to-surface diffusion within the cage volume, is continuous everywhere, interpolates the functions prescribed on the cage and comes with a closed-form solution which is easy to implement (Sec. 4). As a result, one can freely use quad and tri-quad cages – folding back to traditional MVC in the presence of triangles – to smoothly propagate high-level cage edits toward higher resolution surfaces (Sec. 5). Compared to space coordinates for triangle meshes such as classical MVC and Green Coordinates (GC) [Lipman et al. 2008], our experiments on deformations show that the distortion introduced by the mandatory cage triangulation completely vanishes when using our Quad MVC (QMVC). To our knowledge, QMVC is the first space coordinate system for (non-planar) quad cages in 3D, which opens a collection of interesting research directions (Sec. 6).

Related work

Classical coordinates. Barycentric coordinates, introduced by Mobius in 1827 [Möbius 1827], are still widely used for data interpolation in any d-dimensional simplices. Since then, motivated by numerous applications, they have been generalized in different ways to cover arbitrary polygons or polytopes [Hormann and Sukumar 2017]. In computer graphics, 3D generalized barycentric coordinates provide a simple and efficient way to extrapolate functions defined on a 2-manifold surface (the cage), within its volume. The extrapolated value at an arbitrary point $\boldsymbol{\eta}$ is then defined as a weighted combination of values associated with the cage vertices, the weights being referred to as coordinates of $\boldsymbol{\eta}$.

Generalized barycentric coordinates for 3D shape deformation being too large to be covered in detail here, we briefly review closely related previous approaches only. Interested readers are referred to recent surveys [Floater 2015; Hormann and Sukumar 2017] for more details on generalized coordinates.

3D extensions of classical coordinates, such as Wachspress [Ju et al. 2005b; Warren et al. 2007] and harmonic coordinates [Ju et al. 2007], are well-defined within a convex polyhedron with triangular faces. Recently, Budninskiy et al. [Budninskiy et al. 2016] presented a full geometric parametrization of generalized barycentric coordinates on convex polytopes through a power diagram i.e., a generalized Voronoi diagram. For the case of non-convex polytopes, unfortunately, most of classical generalized barycentric coordinates often lose smoothness or are ill-defined.

Mean value coordinates. Represented as simple closed forms in arbitrary convex or non-convex polyhedra with triangular faces [Floater 2003; Ju et al. 2005a], 3D mean value coordinates are particularly popular, and have the practical advantage of being defined even outside the cage – property that makes them an ideal solution for, e.g., low-resolution to high-resolution solution propagation in optimization frameworks [Borosán et al. 2010; Huang et al. 2006]. They also come with closed-form expressions for their derivatives [Thiery et al. 2014] and, as such, are a viable candidate system for e.g., variational deformation or spatial function design. While they fail to guarantee positivity, they are still widely used for applications, such as cage-based freeform deformation.

Green coordinates. 3D Green coordinates [Lipman et al. 2008] use Green's third identity to express everywhere inside the cage a harmonic deformation from combined Dirichlet (function) and Neumann (normal derivative) boundary conditions, defining coordinates with respect to the cage vertex positions and triangle normals. By setting the normal derivative on each cage triangle as a function of its stretch, they enforce as-similar-as-possible Neumann conditions, and obtain experimentally quasi-conformal shape deformations.

Their derivatives have been used in variational frameworks [Ben-Chen et al. 2009] to provide as-rigid-as-possible deformations.

Polyhedra with arbitrary faces. We refer to [Floater 2015] for a generalization of MVC to polyhedra with arbitrary faces, in which *Inverse bilinear coordinates* within planar quads are introduced. Langen et al. [2006] proposed *Spherical Mean Value Coordinates* (SMVC), which derive MVC on the Gauss sphere. They also show how to extend their formulation to compute MVC for planar polygons, and illustrate the benefits of using, e.g., quads instead of triangles in cages. Our contribution is complementary to theirs, in that they consider arbitrary n-gons (and not just quads) while we focus on arbitrary quads including non-planar ones, which is highly relevant in practical shape modeling, as cages designed by artists contain mostly non-planar quads. We note that the 2D case of a curved cage network, whose boundaries are piecewise cubic Bezier curves, has been addressed by Li et al. [2013], in order to offer, like us, smoother boundary control. Last, Schaefer et al. [2017] used multi-sided patches as a generalization of Bezier basis functions and exploited them for cage-based deformations.

Non-analytical coordinates. Several coordinate systems have been introduced to address specific scenarios, such as Positive Mean Value Coordinates (PMVC) [Lipman et al. 2007] on non-convex polyhedra or Harmonic Coordinates (HC) on multigrids [Joshi et al. 2007], which both ensure positivity and favor locality of the cage's influence, as well as Maximum Entropy Coordinates (MEC) [Hormann and Sukumar 2008] which project input vertex masses to the set of valid coordinates, i.e., verifying linear reproduction and partition of unity (see Sec. 2). Although HC and PMVC are of interest for quad-based deformation in terms of computation efficiency, they do not admit a closed-form expression and require the numerical approximation of complicated integrals, without any guarantees on linear reproduction. In practice, their computation involves the rasterization of the vertex basis functions, which make them compatible with quad cages but also lead to discretization artifacts (see Fig. 2), that can be resolved only by manually fine-tuning both models and cages before editing. Last, HC are computed on an explicit volumetric structure (voxel grid, octree, tetrahedral mesh), *one-by-one*, due to heavy memory consumption; adding new evaluation points or changing their binding positions requires each time to recompute the whole grid from scratch, unless a weight sparsification procedure is done in a post process (see [Joshi et al. 2007], section 4), which results in additional approximation in the computations. These various shortcomings motivate the pursue of discretization-free coordinates such as mean-value or Green coordinates.

Cage generation. Before using space coordinates, one must create a cage around the target model. While this can be done using any polygon modeler, it turns out to be a tedious task, in particular when seeking for both coarse and tight cages. Several automatic and semi-automatic methods have been proposed, ranging from volume primitive stitching [Xian et al. 2012] to constrained simplified mesh optimization [Sacht et al. 2015]. A recent survey on such methods can be found in [Cadleron and Boubekeur 2017]. Often, the user may eventually fine tune the output of such systems on a per-primitive basis, which favors quad layouts over triangle ones.

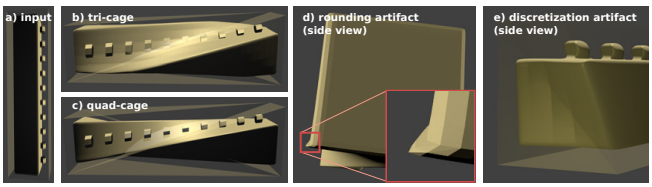


Fig. 2. Results were generated using the implementation of the harmonic coordinates (HC) provided in Blender, which is widely used in the CG modeling community. **a)** Input mesh and cage for deformations with HC. **b)** Deformation based on HC using the triangulated cage. **c)** Similar deformation based on HC using the quad cage. **d)** Grid rounding artifact (convergence issues). **e)** Grid discretization artifact for a triquad cage (approximate rasterization of the basis functions on the grid, also visible in b).

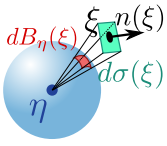
2 BACKGROUND AND NOTATIONS

Thoughout the paper , we will note C the cage, $\mathcal{V} = \{v_i\}$ its vertices, $\mathcal{F}$ its faces which include its triangles $\mathcal{T}$ and quads $\mathcal{Q}$. We will denote vectorial quantities with **bold** fonts. In particular, we will often note $\boldsymbol{v}$ the 3D position of a vertex and v its index. Furthemore, vectorial quantities will be column vectors, and $|\boldsymbol{a};\boldsymbol{b};\boldsymbol{c}|$ will denote the determinant of the 3D vectors $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$. We note $\{\Phi_i\}$ a partition of unity on C associated with vertices $\{i\}$ in C, verifying

- $\Phi_i(\boldsymbol{v_j}) = \delta_i^j$ (*interpolation*),
- $1 = \sum_i \Phi_i(\boldsymbol{\xi})\ \forall \boldsymbol{\xi}$ (*partition of unity*),
- $\boldsymbol{\xi} = \sum_i \Phi_i(\boldsymbol{\xi})\boldsymbol{v_i}$ (*linear reproduction* of the geometry), and
- $\boldsymbol{\xi} \notin F_1(i) \Rightarrow \Phi_i(\boldsymbol{\xi}) = 0$ (*local support on the cage*: the support of a vertex' basis function is confined to its adjacent faces F_1).

On a triangle, the so-called "hat" function, $\Phi_i(\boldsymbol{\xi})$, is the piecewise-linear function that takes value 1 at vertex i and 0 at the others. For a quad, various geometric representations and associated basis functions exist. In particular, for a *planar* quad, an infinity of basis functions exist, which reproduce the exact same geometry: consider for example the two sets of basis functions corresponding to cutting the quad into two triangles along both diagonals, and any linear combination of these. In our work, we will consider $\{\Phi_i(\boldsymbol{\xi})\}$ to be the bilinear interpolation coordinates on the quads. These basis functions are widely used in Computer Graphics, and inverse bilinear coordinates for 2D points within a 2D quad were for example presented by Floater et al. in [Floater 2015].

Further, $d\sigma(\boldsymbol{\xi})$ denotes the surfacic element on C around point $\boldsymbol{\xi}$, $B_{\boldsymbol{\eta}}(S)$ denotes the projection of a surface S onto the unit sphere centered in $\boldsymbol{\eta} \in \mathbb{R}^3$, and $dB_{\boldsymbol{\eta}}(\boldsymbol{\xi})$ the surfacic element on this sphere, around the projection of point $\boldsymbol{\xi} \in S$.



Note that these two quantities are related by

$$dB_{\boldsymbol{\eta}}(\boldsymbol{\xi}) = \frac{(\boldsymbol{\xi} - \boldsymbol{\eta}) \cdot \boldsymbol{n}(\boldsymbol{\xi})}{\|\boldsymbol{\xi} - \boldsymbol{\eta}\|^3} d\sigma(\boldsymbol{\xi}) \tag{1}$$

2.1 Bilinear interpolation on quads

We refer to the C^∞ 3D surface interpolating the four 3D positions $(\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3})$ as the bilinear sheet of the quad, defined as

$$\boldsymbol{\xi}(u,v) := \sum_{i=0}^{3} \Phi_i(u,v)\boldsymbol{q_i}\ ,\ (u,v) \in \mathbb{R}^2\ , \text{with} \tag{2}$$

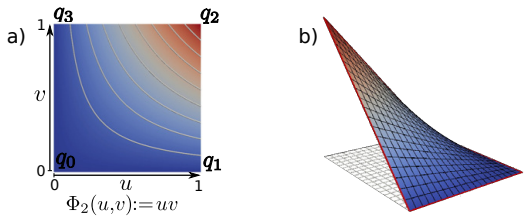


Fig. 3. Bilinear quad parameterization in $[0,1]^2$ (a) and 3D geometry (b).

$$(\Phi_0, \Phi_1, \Phi_2, \Phi_3)(u,v) := ((1-u)(1-v), u(1-v), uv, (1-u)v) \quad (3)$$

We refer to its restriction to $[0,1]^2$ as the bilinear quad (see Fig. 3). For convenience, we note $b_{uv} := (\Phi_0(u,v), \cdots, \Phi_3(u,v))^{\mathrm{T}}$ the 4D vector stacking the bilinear coordinates at (u,v).

It is worth mentioning that the normal and the surfacic elements on the bilinear quads can be derived from simple formulae as:

$$\begin{cases} \boldsymbol{Q_i} := (\boldsymbol{q_{i+1}} - \boldsymbol{q_i}) \times (\boldsymbol{q_{i-1}} - \boldsymbol{q_i}) \\ \boldsymbol{N}(u,v) = \sum_i \Phi_i(u,v) \boldsymbol{Q_i} \\ \boldsymbol{n}(u,v) = \boldsymbol{N}(u,v) / \|\boldsymbol{N}(u,v)\| \\ d\sigma(u,v) = \|\boldsymbol{N}(u,v)\| du dv \\ dB_{\boldsymbol{\eta}}(u,v) = \frac{(\boldsymbol{\xi}(u,v) - \boldsymbol{\eta}) \cdot \boldsymbol{N}(u,v)}{\|\boldsymbol{\xi}(u,v) - \boldsymbol{\eta}\|^3} du dv \,. \end{cases} \quad (4)$$

2.2 Mean value coordinates in 3D

Given a vector function f_C defined on the cage, one can extend it to $\mathbb{R}^3$ through the mean value interpolant, as:

$$f_{\mathbf{MV}}(\boldsymbol{\eta}) = \int_{B_{\boldsymbol{\eta}}(C)} \frac{f_C(\boldsymbol{\xi}) dB_{\boldsymbol{\eta}}(\boldsymbol{\xi})}{\|\boldsymbol{\xi} - \boldsymbol{\eta}\|} \Big/ \int_{B_{\boldsymbol{\eta}}(C)} \frac{dB_{\boldsymbol{\eta}}(\boldsymbol{\xi})}{\|\boldsymbol{\xi} - \boldsymbol{\eta}\|} \quad (5)$$

Please forgive the slight abuse of notations in this integral, and note that $\int_{B_{\boldsymbol{\eta}}(S)} \cdot \, dB_{\boldsymbol{\eta}}(\boldsymbol{\xi})$ denotes the integral *on the unit sphere centered in* $\boldsymbol{\eta}$, when going over the surface S ($\boldsymbol{\xi} \in S$). Formally, one can interpret this type of integral as

$$\int_{B_{\boldsymbol{\eta}}(C)} f_C(\boldsymbol{\xi}) dB_{\boldsymbol{\eta}}(\boldsymbol{\xi}) = \int_C f_C(\boldsymbol{\xi}) \frac{(\boldsymbol{\xi} - \boldsymbol{\eta}) \cdot \boldsymbol{n}(\boldsymbol{\xi})}{\|\boldsymbol{\xi} - \boldsymbol{\eta}\|^3} d\sigma(\boldsymbol{\xi}) \quad (6)$$

Linear precision: Eq. 5 implies directly linear precision, i.e., an input point is not moved by a cage that has not moved:

$$f_C = \mathbf{Id} \quad \Rightarrow \quad f_{\mathbf{MV}} = \mathbf{Id} \quad (7)$$

Indeed, the integral of the unit normal on any closed surface (as is $B_{\boldsymbol{\eta}}(C)$) is null, and therefore:

$$\mathbf{0} = \int_{B_{\boldsymbol{\eta}}(C)} \frac{\boldsymbol{\xi} - \boldsymbol{\eta}}{\|\boldsymbol{\xi} - \boldsymbol{\eta}\|} dB_{\boldsymbol{\eta}}(\boldsymbol{\xi}) \quad \Leftrightarrow \quad \boldsymbol{\eta} = \int_{B_{\boldsymbol{\eta}}(C)} \frac{\boldsymbol{\xi} dB_{\boldsymbol{\eta}}(\boldsymbol{\xi})}{\|\boldsymbol{\xi} - \boldsymbol{\eta}\|} \Big/ \int_{B_{\boldsymbol{\eta}}(C)} \frac{dB_{\boldsymbol{\eta}}(\boldsymbol{\xi})}{\|\boldsymbol{\xi} - \boldsymbol{\eta}\|}$$

Interpolation: Although the integrals in Eq. 5 are ill-defined for $\boldsymbol{\eta} \in C$, one can check by taking the limit that $f_{\mathbf{MV}}$ is indeed an interpolant on the cage, as it is defined through the averaging of a singular kernel ($1/\|\boldsymbol{\xi} - \boldsymbol{\eta}\| = \infty$ for $\boldsymbol{\xi} = \boldsymbol{\eta}$), i. e.,

$$\boldsymbol{\eta} \in C \quad \Rightarrow \quad f_{\mathbf{MV}}(\boldsymbol{\eta}) = f_C(\boldsymbol{\eta}) \quad (8)$$

MVC for polygonal cages: On polygonal cages, we consider that both the geometry of the cage and the functions defined on the cage are linearly interpolated on the cage faces with the same basis functions. Noting f_i the value prescribed at vertex $\boldsymbol{v_i}$, this reads:

$$\begin{cases} f_C(\boldsymbol{\xi}) = \sum_{i \in \mathcal{V}} \Phi_i(\boldsymbol{\xi}) f_i \\ \boldsymbol{\xi} = \sum_{i \in \mathcal{V}} \Phi_i(\boldsymbol{\xi}) \boldsymbol{v_i} \end{cases}$$

It follows that the MVC interpolant $f_{\mathbf{MV}}$ takes the following form:

$$f_{\mathbf{MV}}(\boldsymbol{\eta}) = \sum_{i \in \mathcal{V}} \phi_{\mathbf{MV}\,i}(\boldsymbol{\eta}) f_i \,, \text{ with} \quad (9)$$

$$\begin{cases} w_i(\boldsymbol{\eta}) := \int_{B_{\boldsymbol{\eta}}(C)} \frac{\Phi_i(\boldsymbol{\xi})}{\|\boldsymbol{\xi} - \boldsymbol{\eta}\|} dB_{\boldsymbol{\eta}}(\boldsymbol{\xi}) \\ \phi_{\mathbf{MV}\,i}(\boldsymbol{\eta}) := w_i(\boldsymbol{\eta}) / \sum_j w_j(\boldsymbol{\eta}) \end{cases}$$

Noting that the support of Φ_i is restricted to the faces $F_1(i)$, which are adjacent to vertex i, we obtain further that

$$w_i(\boldsymbol{\eta}) = \sum_{f \in F_1(i)} w_i^f(\boldsymbol{\eta}) \,, \text{ with} \quad (10)$$

$$w_i^f(\boldsymbol{\eta}) := \int_{B_{\boldsymbol{\eta}}(f)} \frac{\Phi_i(\boldsymbol{\xi})}{\|\boldsymbol{\xi} - \boldsymbol{\eta}\|} dB_{\boldsymbol{\eta}}(\boldsymbol{\xi}) \quad (11)$$

Smoothness: For any point $\boldsymbol{\eta}$ inside the cage ($\boldsymbol{\eta} \notin C$), the MVC (and $f_{\mathbf{MV}}$) are C^∞, since $1/\|\boldsymbol{\xi} - \boldsymbol{\eta}\|$ is C^∞ and bounded for any $\boldsymbol{\xi} \in C$ (similarly for its derivatives w.r.t. $\boldsymbol{\eta}$), and C is compact.

For any point on the edges of the cage, the $f_{\mathbf{MV}}$ is C^0 only, since it is an interpolant of f_C, which is continuous only across the edges of the cage C.

On positivity: MVC can be negative for non-convex cages, when the signed solid angle $dB_{\boldsymbol{\eta}}(\boldsymbol{\xi})$ is negative for triangles that are facing the evaluation point $\boldsymbol{\eta}$. This does not necessarily translate into negative coordinates w.r.t. these faces: only the unnormalized coordinates $w^f(\boldsymbol{\eta})$ are negative, but they might dominate in magnitude the other unnormalized coordinates in the final set of coordinates, and the MVC w.r.t. the corresponding vertices might end up being positive after normalization.

The non-positivity of the MVC is a well-documented problem and can result in non intuitive deformations e.g., a translation of a cage vertex in one direction results in translations in the opposite direction in the mesh. We show this behavior in the results section, as well as the behavior of our coordinates, once rendered positive using the MEC approach.

2.3 Computation of mean value coordinates

At this point, assuming we can compute $\{w_i^f(\boldsymbol{\eta})\}$, the mean value coordinates of $\boldsymbol{\eta}$ can be computed using the following recipe:

(1) If $\boldsymbol{\eta}$ lies on the cage, then $\phi_{\mathbf{MV}\,i}(\boldsymbol{\eta}) = \Phi_i(\boldsymbol{\eta}), \forall i$.
(2) Otherwise:
- compute $w_i^f(\boldsymbol{\eta}) \;\; \forall i, \forall f \in F_1(i)$
- compute $w_i(\boldsymbol{\eta}) = \sum_{f \in F_1(i)} w_i^f(\boldsymbol{\eta}) \;\; \forall i$
- compute $\phi_{\mathbf{MV}\,i}(\boldsymbol{\eta}) = w_i(\boldsymbol{\eta}) / \sum_j w_j(\boldsymbol{\eta}) \;\; \forall i$

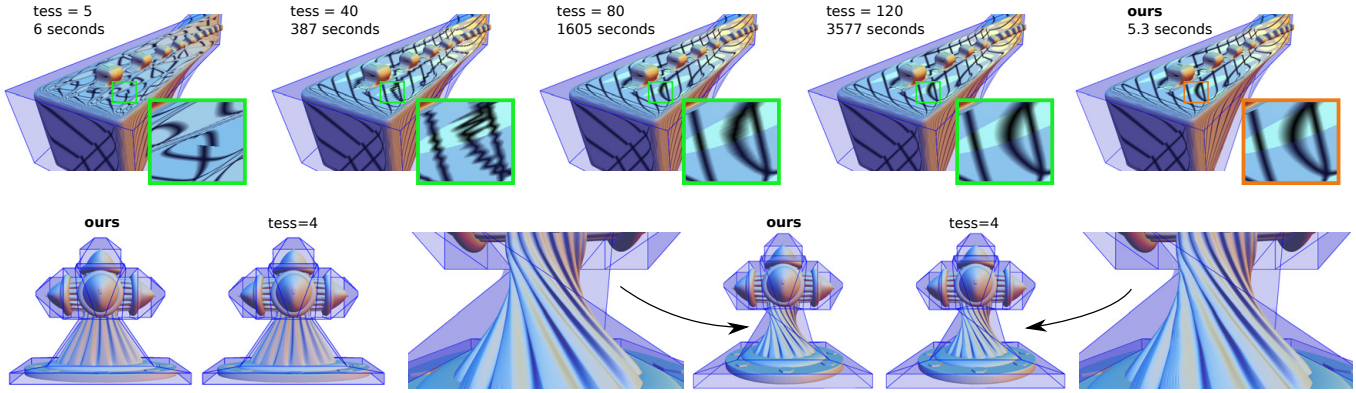
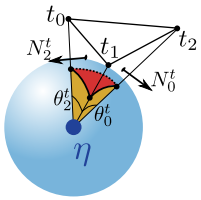


Fig. 4. **Top:** Approximating the integrals with a tessellation approach might require extreme tessellation levels before rendering artifacts (here, reflection lines obtained with a cube-map) become invisible. Ultimately, it might even be necessary to tessellate the quads enough to match the local mesh vertex density. In the figure, $tess$ indicates that each quad of the cage has been subdivided into $tess \times tess$ quads further subdivided into two triangles each. At equal timings ($tess = 5$, left), discretization artifacts are visible on the position function itself (C^0 continuity). Only for an extreme tessellation level ($tess = 120$, 4^{th} image), which required 1h of computations, visible artifacts in the higher continuity orders disappear and match the quality obtained with our scheme (right). **Bottom:** The tessellation artifacts are less visible when the model is further away from the cage model, as the basis functions are then further blended. Nevertheless, twisting the cage model reveals these artifacts, even on the geometry itself (C^0 continuity), taking the form of undesired wavy deformations. This illustrates once again how difficult it is to obtain satisfactory weights with a fixed approximation pattern, since artifacts might become visible depending on (i) the input cage geometry, (ii) the input mesh geometry i.e., location of the mesh w.r.t. the cage and mesh density, and (iii) the performed cage deformation.

Computation of $w_i^f(\boldsymbol{\eta})$ on a triangle. We present here the derivation of these coordinates for triangle meshes, as introduced in [Ju et al. 2005a]. For a triangle $t = (t_0, t_1, t_2)$, we note



$$\theta_i^t := \angle(\boldsymbol{t_{i+1}\eta t_{i+2}})$$
$$\boldsymbol{N_i^t} := (\boldsymbol{t_{i+2}}-\boldsymbol{\eta}) \times (\boldsymbol{t_{i+1}}-\boldsymbol{\eta}),$$

all indices being modulo 3.

Ju et al. [Ju et al. 2005a] remarked that

$$\sum_{i=0}^{2} w_{t_i}^t(\boldsymbol{\eta})(\boldsymbol{t_i}-\boldsymbol{\eta}) = \sum_{i=0}^{2} \int_{B_{\boldsymbol{\eta}}(t)} \frac{\Phi_i(\xi)}{\|\xi-\boldsymbol{\eta}\|} dB_{\boldsymbol{\eta}}(\xi)(\boldsymbol{t_i}-\boldsymbol{\eta})$$
$$= \int_{B_{\boldsymbol{\eta}}(t)} \frac{\xi-\boldsymbol{\eta}}{\|\xi-\boldsymbol{\eta}\|} dB_{\boldsymbol{\eta}}(\xi).$$

This last integral is the integral of the unit normal over the spherical triangle $B_{\boldsymbol{\eta}}(t)$ (see inset, in red), and is commonly referred to as the mean vector of t at $\boldsymbol{\eta}$, which we denote here $\boldsymbol{m}_t$. Noting, once again, that the integral of the unit normal over a closed surface is null, one can compute $\boldsymbol{m}_t$ as (minus) the integral of the unit normal over the three spherical triangles (see inset, in orange), and derive

$$\sum_{i=0}^{2} w_{t_i}^t(\boldsymbol{\eta})(\boldsymbol{t_i}-\boldsymbol{\eta}) = \boldsymbol{m}_t, \text{ with} \tag{12}$$

$$\boldsymbol{m}_t := -\sum_{i=0}^{2} \frac{\theta_i^t \boldsymbol{N_i^t}}{2\|\boldsymbol{N_i^t}\|}. \tag{13}$$

Inverting this system leads to

$$w_{t_i}^t(\boldsymbol{\eta}) = \frac{\boldsymbol{m}_t \cdot \boldsymbol{N_i^t}}{(\boldsymbol{t_i}-\boldsymbol{\eta}) \cdot \boldsymbol{N_i^t}}. \tag{14}$$

Note that, when $\boldsymbol{\eta}$ lies on the support plane of t, the system cannot be inverted ($(\boldsymbol{t_i}-\boldsymbol{\eta}) \cdot \boldsymbol{N_i^t} = 0$). However, in this situation, when $\boldsymbol{\eta}$ is inside the triangle t, the MVC are simply set to the basis functions at $\boldsymbol{\eta}$ (as MVC are an interpolant on the cage), whereas when $\boldsymbol{\eta}$ is outside t the three coordinates $w_{t_i}^t(\boldsymbol{\eta})$ are simply null, since the projection of t on the sphere centered in $\boldsymbol{\eta}$ has a null area.

Approximate computation of $w_i^f(\boldsymbol{\eta})$ on a quad. Integrals over bilinear quads are difficult to compute, as the integration domain is curved. One simple strategy to approximate $w_i^q(\boldsymbol{\eta})$ on a quad q is to tessellate q into triangles $\{t_k = (t_{k0}t_{k1}t_{k2})\}$, compute q's bilinear basis functions on $\{t_k\}$'s corners, and rely on linear interpolation over $\{t_k\}$ so that usual triangle-based MVC can be used:

$$\Phi_i^q(\xi) \simeq \sum_{t_k}\sum_{c=0}^{2} \Phi_i^q(\boldsymbol{t_{kc}})\Phi_{t_{kc}}^{t_k}(\xi) \Rightarrow w_i^q(\boldsymbol{\eta}) \simeq \sum_{t_k}\sum_{c=0}^{2} \Phi_i^q(\boldsymbol{t_{kc}}) w_{t_{kc}}^{t_k}(\boldsymbol{\eta})$$

As the tessellation level increases, the MVC for the bilinear quad can be obtained at the limit. Unfortunately, with this simple approach, deformation artifacts quickly appear in the results due to the continuous-only approximation of the quad (combined with the interpolation property of the MVC). When the local mesh geometry is close to the cage quads, it might even be necessary to tessellate the quads enough to match the local mesh vertex density (see Fig. 4, top) before visual artifacts disappear, which reflects unfavorably on the computation timings. We address this problem by making the domain sampling (and therefore the integrals) *dependent on the evaluation point* (see Section 3.4).

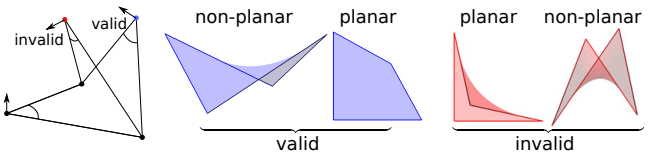


Fig. 5. A quad is valid (in blue) if its projection is convex in every four planes supporting the triangles at its corners. This condition extends naturally the notion of convexity in 2D (see invalid planar quad in red), and is a natural condition on the faces of cages used for interactive modeling.

3 MEAN-VALUE COORDINATES FOR QUADS

3.1 Overview

In order to design a generalized coordinate system in the spirit of the original MVC but for quad cages, we first characterize the set of valid quads, for which our QMVC can be defined (Sec. 3.2). As we show, this validity condition for each quad face of a candidate cage model is quite natural and prevents for example fold-overs for planar faces. Moreover the computation of QMVC for points which lie on such a quad is straightforward and we provide a closed form formula (Sec. 3.3) for points on planar and non-planar quads. However, the computation of QMVC coordinates for points which do not lie on the quad is harder, involving a numerical approximation of integrals on quads (Sec. 3.4). More precisely, we first explain how to derive the weights with respect to a quad, up to a translation in a 4D space, as they are solution of a linear equation of rank 3 in $\mathbb{R}^4$. In order to obtain this translation, we introduce a robust estimation of mean value integrals over the quad via an adaptive integration scheme which takes the form of a discrete Riemann summation, and observe the component of this estimate along the corresponding axis. Given any bilinear quad q, we ensure that this sum tends to a Dirac distribution around q (thus ensuring *interpolation*), by introducing a smooth projection operator on q that quickly tends to the orthogonal projection around the quad.

As a result, our coordinates approximate well the ground truth QMVC, retaining the three properties that are most important for shape deformations, namely **(i)** smoothness, **(ii)** interpolation and **(iii)** natural deformation behavior.

In the following, we give a detailed description of the mathematical motivation and derivation that leads to QMVC, while the reader may jump to Sec. 4 for a compact implementation recipe.

3.2 Valid quads for modeling

Non-convex planar polygons are degenerate configurations for interpolation; in particular the interpolation given by the bilinear interpolant introduces fold-overs. In this type of configuration, several points of the bilinear quad i.e., with different (u, v) coordinates, are located at the same 3D position (see Fig. 5, invalid planar quads). These quads are therefore not suitable for spatial interpolation.

We provide a geometric characterization of quads which are compatible with our approach (see Fig. 5, blue), as follows:

Definition 1. *A quad with successive corners* $(\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3})$ *is* **valid** *if its orthogonal projection on the four supporting planes of the triangles* $(\boldsymbol{q_{i-1}}, \boldsymbol{q_i}, \boldsymbol{q_{i+1}})$*, (indices mod 4) for* $0 \leq i \leq 3$*, is convex.*

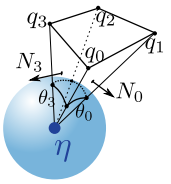
This characterization will be essential to establish the validity of our coordinates, and extends naturally the notion of convexity in two dimensions: a planar quad will be convex if and only if it is valid, according to Def. 1.

Note that this condition is *sufficient*, yet not necessary, and that the characterization of the complete family of quads compatible with our approach (in particular, with the projection operator we design in the next sections) is a challenging avenue for future work (see limitations in section 6).

In the following, we will assume cages made of valid quads only.

Algebraic characterization of MVC. For a quad $q = (\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3})$, we note

$$\boldsymbol{N_i} := (\boldsymbol{q_{i+1}} - \boldsymbol{\eta}) \times (\boldsymbol{q_i} - \boldsymbol{\eta})$$
$$\theta_i := \angle(\boldsymbol{q_i \eta q_{i+1}})$$



As noticed by Ju et al. [2005a], we remark that

$$\sum_{i=0}^{3} w^q_{q_i}(\boldsymbol{\eta})(\boldsymbol{q_i} - \boldsymbol{\eta}) = \boldsymbol{m}_q \text{ , with} \tag{15}$$

$$\boldsymbol{m}_q := -\sum_{i=0}^{3} \frac{\theta_i \boldsymbol{N_i}}{2\|\boldsymbol{N_i}\|} \tag{16}$$

which we write using the following matrix expression in $\mathbb{R}^4$:

$$A_q \cdot w^q = \boldsymbol{m}_q \text{ ,} \tag{17}$$

with $\text{col}_j(A_q) := (\boldsymbol{q_j} - \boldsymbol{\eta})$ and $w^q := (w^q_{q_0}, w^q_{q_1}, w^q_{q_2}, w^q_{q_3})^{\mathrm{T}}$.

For the sake of simplicity, we do not index A_q over $\boldsymbol{\eta}$, but please keep in mind that it depends on both the quad geometry and $\boldsymbol{\eta}$, similarly to $\boldsymbol{m}_q$. Unfortunately, Eq. 17 is underdetermined, and does not allow us to compute the unnormalized weights w^q.

Note on the rank of A_q. The rank of A_q is directly given by the dimension of the space that is spanned by the vectors $\{\boldsymbol{q_i} - \boldsymbol{\eta}\}_i$:

- For a *planar* quad q, A_q is of rank 3 everywhere in space but on the support plane of q where it is of rank 2.
- For a *non-planar* quad q, A_q is of rank 3 everywhere in space.

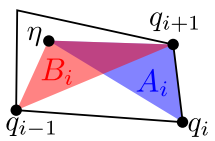
When A_q is of rank 3, we note $\kappa_A = (k_0, k_1, k_2, k_3) \in \mathbb{R}^4$ the unique (up to scaling) vector spanning its kernel i.e., $A \cdot \kappa_A = 0$. This vector can be obtained with the singular value decomposition (or SVD) of $A_q := U \cdot \Sigma \cdot V^{\mathrm{T}}$, as the 4^{th} column of V, assuming the singular values are ordered by decreasing magnitude:

$$\kappa_A := col_3(V), \text{ with} \tag{18}$$
$$A_q := U \cdot \Sigma \cdot V^{\mathrm{T}} \text{ (SVD)}$$

Interestingly, κ_A can be alternatively expressed in simple geometric terms (see Appendix A).

3.3 When η lies on the bilinear quad: Inverse bilinear weights computation

As recalled in Sec. 2.3, if $\boldsymbol{\eta}$ lies on the cage, the mean values coordinates at $\boldsymbol{\eta}$ are simply given by the vertex basis functions $\{\Phi_i(\boldsymbol{\eta})\}$, since mean-value coordinates interpolate these on the cage.



n_q : quad's normal
$A_i := |q_i - \eta; q_{i+1} - \eta; n_q|/2$
$B_i := |q_{i-1} - \eta; q_{i+1} - \eta; n_q|/2$
$D := B_0^2 + B_1^2 + 2A_0A_2 + 2A1A3$
$E_i := 2A_i - B_i + B_{i+1} + \sqrt{D}$

Fig. 6. Notations introduced by Floater et al. [2015] for their computation of inverse bilinear coordinates in planar quads.

If q is planar, and $\boldsymbol{\eta}$ lies on its support plane: Checking then if $\boldsymbol{\eta}$ is located *inside* or *outside* q is straightforward. If η is outside the quad, the coordinates $w^q_{q_i}(\boldsymbol{\eta})$ are null since q projects on a null area on the sphere centered in $\boldsymbol{\eta}$. Otherwise we need to detect the bilinear coordinates b_{uv}, which reproduce $\boldsymbol{\eta}$, and set the mean-value coordinates to those. We use the derivation proposed by Floater et al. in [2015] for that purpose, which we recall here for completeness. Using the notations of Fig. 6, they derived

$$\begin{cases} u = & 2A_3/E_3 \\ v = & 2A_0/E_0 \end{cases} \tag{19}$$

If q is non-planar: In this case, η might still be lying on the bilinear (non-planar) quad. To detect if this is the case, we look for $(u, v) \in [0, 1]^2$, whose associated bilinear coordinates $b_{uv} \in \mathbb{R}^4$ reproduce $\boldsymbol{\eta}$, or equivalently: $(\boldsymbol{q_0}-\boldsymbol{\eta}, \cdots, \boldsymbol{q_3}-\boldsymbol{\eta}) \cdot b_{uv} = \mathbf{0}$, i.e., b_{uv} belongs to the null space of A_q. Since κ_A spans alone A_q's kernel, we only need to check whether κ_A is of the form $\kappa_A := \alpha b_{uv}$, with $(u, v) \in [0, 1]^2$. This can be tested by computing candidate parameters (α, u, v) as $\alpha = \sum_i k_i$, $u = (k_1 + k_2)/\alpha$ and $v = (k_2 + k_3)/\alpha$, and checking that the corresponding b_{uv} verifies $\kappa_A = \alpha b_{uv}$, with $(u, v) \in [0, 1]^2$. If so, the mean-value coordinates equal b_{uv}. Otherwise, $\boldsymbol{\eta}$ does not belong to the quad.

3.4 General case: when η does not lie on the quad

When η does not belong to the quad, the rank of A_q equals 3. The "minimal-norm solution" is then given by

$$\bar{w}^q := {A_q}^\dagger \cdot \boldsymbol{m}_q, \tag{20}$$

where ${A_q}^\dagger$ denotes the pseudo-inverse of A_q computed using its SVD. Note that one can express $\bar{w}^q$ in simple geometric terms, as explained in appendix A.

The full solution is given by $w^q = \bar{w}^q + \lambda\kappa_A$, $\lambda : \mathbb{R}^3 \to \mathbb{R}$ a scalar field depending on η that remains to be found.

Valid λ fields. We make the following key observations:

(1) Eq.17 is independent from the choice of basis functions defined on the quad, and a particular choice of basis functions leads to a specific corresponding scalar field λ.
(2) $\lambda : \mathbb{R}^3 \to \mathbb{R}$ should be C^∞ everywhere but on the quad, to match the expected regularity of the MVC.
(3) *Any* choice of λ field leads to linear reproduction: the weights verify Eq. 16 regardless of the value of λ, which, once summed over the faces of the cage provide linear reproduction.
(4) Valid λ fields should lead to interpolation on the cage.

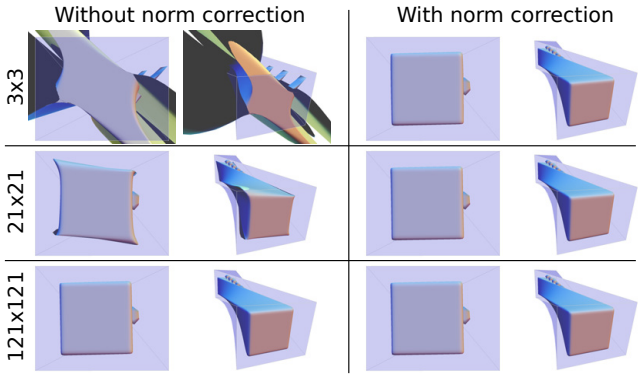


Fig. 7. The encoding cage is a straight bar, which we twist by 90 degrees to illustrate the impact of our norm correction: It provides results close to the ground truth and qualitatively similar, even on extremely coarse sampling grids (compare the norm-corrected 3×3 and 121×121 grids).

3.4.1 Robust estimate of λ. Unfortunately, the integrals over the bilinear quads, which are required for the closed-form computations of the MVC, do not admit closed-form expressions.

We provide in this section a geometric construction of λ, which retains the three properties that are most important for shape deformations, which are **i)** smoothness, **ii)** interpolation and **iii)** natural deformation behavior. Additionally, our construction is independent of a particular choice of basis functions and could be used for others than bilinear interpolants.

We approximate these coordinates by the means of a spatially-varying Riemann summation. We consider *samples* $\{u_k, v_l\}_{kl}$ equipped with areas $\{a_{kl}\}_{kl}$ and distributed on a $(2n + 1)\times(2n + 1)$-grid $\{u_k\}_k \times \{v_l\}_l$, whose values vary smoothly with $\boldsymbol{\eta}$. This sampling is key to fulfilling our desiderata, and we will detail its construction in the next paragraph.

The exact (ground truth) value of the λ coordinate is given by

$$\lambda_{\text{gd}} := \frac{w^q \cdot \kappa_A}{\|\kappa_A\|^2}, \tag{21}$$

which, unfortunately, requires the exact value of the coordinates $w^q_{q_i} := \int_{B_{\boldsymbol{\eta}}(q)} \frac{\Phi_{q_i}(\xi)}{\|\xi - \boldsymbol{\eta}\|} dB_{\boldsymbol{\eta}}(\xi)$. We compute $\{\Omega^q_{q_i}\} \in \mathbb{R}^4$, which ***estimate*** $\{w^q_{q_i}\}$ using a spatially-varying Riemann summation as

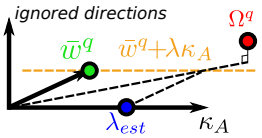
$$\Omega^q_{q_i} := \sum_{k,l} \frac{\Phi_{q_i}(u_k, v_l)}{\|\boldsymbol{\xi}(u_k, v_l) - \boldsymbol{\eta}\|} dB_{kl}, \text{ with} \tag{22}$$

$$dB_{kl} := \frac{\mathbf{N}(u_k, v_l) \cdot (\boldsymbol{\xi}(u_k, v_l) - \boldsymbol{\eta})}{\|\boldsymbol{\xi}(u_k, v_l) - \boldsymbol{\eta}\|^3} a_{kl} \tag{23}$$

Finally, our λ estimate is given by

$$\lambda_{\text{est}} := \frac{\Omega^q \cdot \kappa_A}{\|\kappa_A\|^2} \frac{\|\bar{w}^q\|^2}{\Omega^q \cdot \bar{w}^q}. \tag{24}$$

We introduced $\|\bar{w}^q\|^2/(\Omega^q \cdot \bar{w}^q)$ in Eq.24 (compare with Eq. 21) to account for a *norm correction*. Indeed, while we cannot bound *a priori* the error that is made in the λ coordinate, we can safely



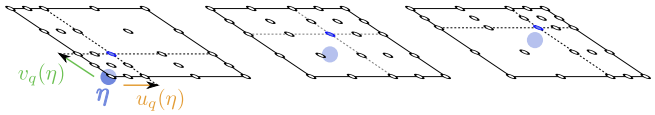


Fig. 8. **Smooth (u, v) sampling.** A 5×5-grid sampling (black circles) of the (u, v)-domain of a quad built to contain a sample (blue circle) at position $\boldsymbol{\eta}$ when $\boldsymbol{\eta}$ (blue point) lies on the quad.

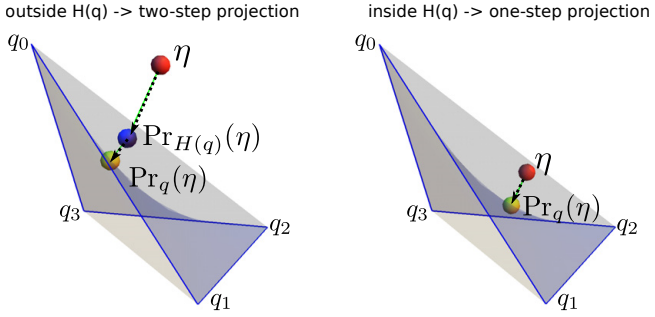


Fig. 9. **Projection operator.** We project $\boldsymbol{\eta}$ on q in two steps: first we project on the convex hull $H(q)$ of q if $\boldsymbol{\eta}$ is outside $H(q)$, then project on q.

assume that the scaling error is distributed approximately similarly in every direction (all singular vectors of A^q), and we can *measure* this distortion on the axes that are orthogonal to κ_A, as we know exactly the value of $w^q \cdot \bar{w}^q$, which is given by $\|\bar{w}^q\|^2$. Note that it is not equivalent to simply rescaling Ω^q, as two directions orthogonal to κ_A and $\bar{w}^q$ are ignored in this process: we cannot simply rescale Ω^q because it does not respect linear reproduction, i.e., $A_q \cdot \Omega^q$ is not colinear with $\boldsymbol{m}_q$. We show in Fig. 7 the effect of this norm correction, which illustrates the validity of our assumption.

3.4.2 Smooth projection operator on quads. We make the simple following observation: in order to obtain interpolation ($\boldsymbol{\eta} \to \boldsymbol{\xi} \in C \Rightarrow \{\phi_{\mathbf{MV}\,i}(\boldsymbol{\eta})\} \to \{\Phi_i(\boldsymbol{\xi})\}$), we must be sure to have $\boldsymbol{\xi}$ as a sample when $\boldsymbol{\eta}$ converges towards $\boldsymbol{\xi}$, so that the Riemman summation tends to a Dirac distribution (as the normalization of a singular kernel). Because we need this property for all points on the quad, and we can have in practice a finite number of samples only, *the only remaining option is to make the sampling dependent to the evaluation point* $\boldsymbol{\eta}$. As we further need the result to be smooth, we also need to make this sampling smooth. Therefore, our strategy is to **(i)** design a smooth projection for point $\boldsymbol{\eta} \in \mathbb{R}^3$ on the (u, v)–domain of q, and further **(ii)** center the sampling distribution around this point (see Fig. 8).

One simple option could be to take the point on q that is closest to $\boldsymbol{\eta}$, which is the simple orthogonal projection operator on convex sets. However, since q is in general not a convex geometry (as soon as q is non-planar), there are points $\boldsymbol{\eta}$ for which several points on q are closest and for which this operator is not well defined, and there are also points at which this operator is discontinuous. Another option would be to integrate the (u, v)-parameterization of the quad against a singular kernel ($(\bar{u}, \bar{v})(\boldsymbol{\eta}) = \int_{[0,1]^2} (u, v) k(\boldsymbol{\xi}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{\eta}) d\sigma(u, v) \,/ \int_{[0,1]^2} k(\boldsymbol{\xi}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{\eta}) d\sigma(u, v)$, with $k(\boldsymbol{\xi}, \boldsymbol{\xi}) = \infty$ to ensure interpolation), but once again the integrals over the curved bilinear quad are difficult to compute. It turns out that obtaining a smooth projection operator on an arbitrary bilinear quad is quite challenging, and we detail our two-step construction in the following (see Fig. 9).

Computation: Noting $H(q)$ the convex hull of q, with oriented boundary $\partial H(q) := \{(\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_3}); (\boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3}); (\boldsymbol{q_0}, \boldsymbol{q_2}, \boldsymbol{q_1}); (\boldsymbol{q_0}, \boldsymbol{q_3}, \boldsymbol{q_2})\}$, we construct a *projection operator* $\mathcal{P}_q : \boldsymbol{\eta} \in \mathbb{R}^3 \to (u_q, v_q)(\boldsymbol{\eta}) \in [0, 1]^2$ mapping the 3D space to the (u, v)–domain of q in the following manner:

If $\boldsymbol{\eta}$ is located outside $H(q)$, we first project it on it. This projection, denoted by $\mathcal{P}_{H(q)}(\boldsymbol{\eta})$, is obtained by averaging the position function using a positive singular kernel (thus ensuring interpolation on the convex hull's boundary), in a spirit similar to MVC:

$$\mathcal{P}_{H(q)}(\boldsymbol{\eta}) = \int_{\boldsymbol{\xi} \in \partial H(q)} \boldsymbol{\xi} k(\boldsymbol{\xi}, \boldsymbol{\eta}) d\sigma(\boldsymbol{\xi}) \,/ \int_{\boldsymbol{\xi} \in \partial H(q)} k(\boldsymbol{\xi}, \boldsymbol{\eta}) d\sigma(\boldsymbol{\xi}) \,, \text{ with} \quad (25)$$

$k(\boldsymbol{\xi}, \boldsymbol{\eta}) := \frac{|(\boldsymbol{\xi} - \boldsymbol{\eta}) \cdot \boldsymbol{n}(\boldsymbol{\xi})|}{\|\boldsymbol{\xi} - \boldsymbol{\eta}\|^3}$. This amounts to computing the standard unnormalized MVC w.r.t. $\partial H(q)$, and taking their absolute value as unnormalized weights for the averaging of the 4 corners. As a result, this operator projects any point inside $H(q)$, and is an interpolant on $\partial H(q)$, which ensures that this operator is continuous everywhere.

We follow by computing the mean vector $\boldsymbol{m}_q(\boldsymbol{\eta'})$ at this location $\boldsymbol{\eta'} = \mathcal{P}_{H(q)}(\boldsymbol{\eta})$, and intersecting the line $\mathcal{D}(\boldsymbol{\eta'}, \boldsymbol{m}_q(\boldsymbol{\eta'}))$ with q (see appendix B), which results in the projection of $\boldsymbol{\eta}$ on q $\mathcal{P}_q(\boldsymbol{\eta})$ through its associated (u, v) coordinates.

We report in Alg. 1 the pseudo-code of our projection operator.

Algorithm 1 Calculate $(u_q, v_q)(\boldsymbol{\eta}) = \mathcal{P}_q(\boldsymbol{\eta})$ // projection of $\boldsymbol{\eta}$ on the bilinear quad q

```
if η is inside the tetrahedron (q0, q1, q2, q3) then
    η' = η
else
    T(q) = {(q0, q1, q3); (q1, q2, q3); (q0, q2, q1); (q0, q3, q2)}
    η' = 0
    for all t ∈ T(q) do
        Compute unnormalized MVC weights w^t_{t_i}(η)  ∀i < 3.
        η' += Σ_{i=0}^{2} |w^t_{t_i}(η)| t_i
    end for
end if
Compute mean vector m_q(η') at η'.
Compute η'' as the intersection of q and the line D(η', m_q(η'))
    in the (u, v)-domain (see appendix B).
Set (u_q, v_q) to the parameters associated with η''.
```

Analysis: We derive here the properties of the projection operator, which inherits smoothness from the MVC used for its computation.

Lemma 3.1. *$\mathcal{P}_q$ is a mapping of the Euclidean space to the bilinear quad q (i.e., $\mathcal{P}_q(\eta) \in [0, 1]^2 \; \forall \eta \in \mathbb{R}^3$) if q is valid, according to Def. 1.*

Proof. The proof of this lemma is given in appendix C. □

Lemma 3.2. *$\mathcal{P}_q$ is a projection operator (i.e., $\mathcal{P}_q \circ \mathcal{P}_q = \mathcal{P}_q$).*

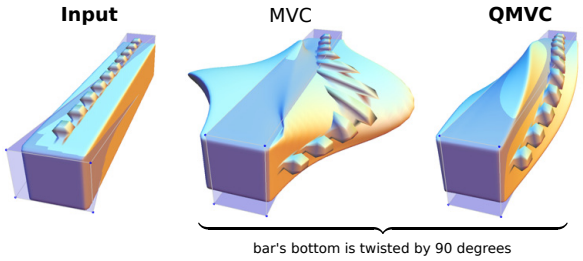


Fig. 10. The input cage is a twisted box, composed of highly distorted quads. We compare our quad-based QMVC with standard MVC, which required a triangulation of the cage. Our formulation exhibits smooth transformations, even for highly distorted quads, which furthermore intersect the input geometry (a large part of it is therefore located *outside* the cage model).

Proof. When $\boldsymbol{\eta}$ lies on q (i.e., $\boldsymbol{\eta} = \boldsymbol{\xi}(u,v)$ with $(u,v) \in [0,1]^2$), $\boldsymbol{m}_q(\boldsymbol{\eta}) \cdot \boldsymbol{n}_q(\boldsymbol{\eta}) \neq 0$ as $dB_{\boldsymbol{\eta}}(\boldsymbol{\xi})$ is singular and thus the intersection between q and $\mathcal{D}(\boldsymbol{\eta}, \boldsymbol{m}_q(\boldsymbol{\eta}))$ is well defined and equals $\boldsymbol{\eta}$. □

Lemma 3.3. *For any planar quad q, $\mathcal{P}_q$ is C^∞ everywhere except on the support plane of q.*

Proof. In this configuration, the first step is sufficient as it maps any point that is not on the support plane on the quad directly, and the second step is then the identity. The regularity of the projection operator is then directly implied by the properties of the MVC. □

Note that we do not need smoothness on the support plane of q if it is planar, as this special case is handled differently.

Lemma 3.4. *For any non-planar quad q, $\mathcal{P}_q$ is C^∞ everywhere except on the tetrahedron $\partial H(q)$, where it is only continuous.*

Proof. The first step of the projection relies on the MVC, and is therefore C^∞ everywhere but on the tetrahedron's boundary $\partial H(q)$, where it is only continuous. The second step relies on the unique intersection between two C^∞ geometries, as the quad's mean vector is C^∞ inside the convex hull, and thus this step is C^∞. □

On $\partial H(q)$, the first step of our projection transitions continuously only from the C^∞ averaging we defined in Eq. 25 to the identity operator. Note however, that we report experiments on highly-challenging quad geometries and models intersecting those, which tend to empirically illustrate that no non-smooth signal results from this in practice (see Fig. 10 and Fig. 14 for real-life models). An explanation of this phenomenon is that the non-smooth signal on $\partial H(q)$ resulting from the first projection is mostly orthogonal to the quad and not tangential: it is therefore almost aligned with $\boldsymbol{m}_q(\boldsymbol{\eta}')$ and thus factored out after the projection along that direction in the second step.

3.4.3 Smooth (u,v) sampling. With the projection operator $\mathcal{P}_q : \boldsymbol{\eta} \in \mathbb{R}^3 \rightarrow (u_q, v_q)(\boldsymbol{\eta}) \in [0,1]^2$ in hand, we can finally perform an adaptive integration by constructing the sample distribution on a $(2n+1) \times (2n+1)$-grid $\{u_i\}_i \times \{v_j\}_j$ as follows (see Fig. 8):

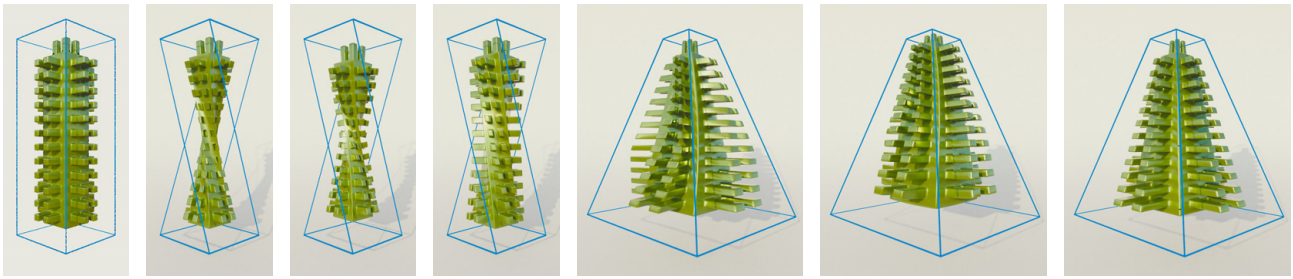
$$\begin{cases} u[n] = u_q(\boldsymbol{\eta}) \\ u[k] = \frac{k}{n} u_q(\boldsymbol{\eta}) \quad \forall 0 \leq k < n \\ u[k] = u_q(\boldsymbol{\eta}) + \frac{k-n}{n}(1 - u_q(\boldsymbol{\eta})) \quad \forall n < k \leq 2n \\ v[n] = v_q(\boldsymbol{\eta}) \\ v[k] = \frac{k}{n} v_q(\boldsymbol{\eta}) \quad \forall 0 \leq k < n \\ v[k] = v_q(\boldsymbol{\eta}) + \frac{k-n}{n}(1 - v_q(\boldsymbol{\eta})) \quad \forall n < k \leq 2n\,, \end{cases}$$

which inherits its smoothness from the projection operator $\mathcal{P}_q : \boldsymbol{\eta} \rightarrow (u_q, v_q)(\boldsymbol{\eta})$, and we equip each sample at position (i,j) with its (u,v)-Voronoï area $a_{ij} := \delta u_i . \delta v_j$ (used in Eq. 23), with

$$\begin{cases} \delta u_i = (u[\min(i+1, 2n)] - u[\max(i-1, 0)])/2 \\ \delta v_j = (v[\min(j+1, 2n)] - v[\max(j-1, 0)])/2 \end{cases}$$

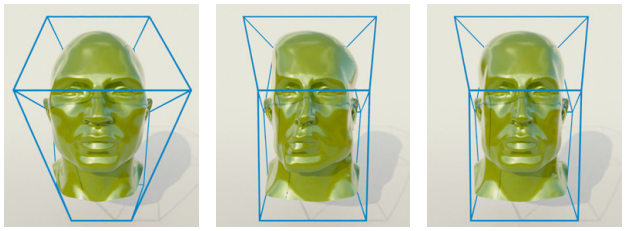
Algorithm 2 Calculate $\{\phi_{\mathbf{MV}\,i}(\boldsymbol{\eta})\}_i$ // QMVC/MVC of $\boldsymbol{\eta}$

for all $i \in \mathcal{V}$ **do**
  $w_i(\boldsymbol{\eta}) = 0$ //initialization
end for
for all $t \in \mathcal{T}$ **do**
  if $\boldsymbol{\eta}$ on the support plane of t **then**
    if $\boldsymbol{\eta}$ is inside t **then**
      $\{\phi_{\mathbf{MV}\,i}(\boldsymbol{\eta})\}_i = \{\Phi_i(\boldsymbol{\eta})\}_i$ **return**
    end if
  else
    compute $w^t_{t_i}(\boldsymbol{\eta}) \ \forall i < 3$ //Eq. 14
    $w_{t_i}(\boldsymbol{\eta}) += w^t_{t_i}(\boldsymbol{\eta}) \ \forall i < 3$
  end if
end for
for all $q \in \mathcal{Q}$ **do**
  if q planar and $\boldsymbol{\eta}$ on its support plane **then**
    if $\boldsymbol{\eta}$ is inside q **then** //Sec. 3.3
      compute b_{uv} s.t. $(\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3}) b_{uv} == \boldsymbol{\eta}$ //Eq. 19
      basis functions $\{\Phi_{q_k}(\boldsymbol{\eta})\}_k$ at $\boldsymbol{\eta}$ equal b_{uv}
      $\{\phi_{\mathbf{MV}\,i}(\boldsymbol{\eta})\}_i = \{\Phi_i(\boldsymbol{\eta})\}_i$ **return**
    end if
  else//then $rank(A_q) == 3$
    compute $\kappa_A = (k_0, k_1, k_2, k_3)$ //kernel of A_q, Eq. 18 or 26
    if $\kappa_A = \alpha b_{uv}$ with $(u,v) \in [0,1]^2$ //Sec. 3.3 **then**
      basis functions $\{\Phi_{q_k}(\boldsymbol{\eta})\}_k$ at $\boldsymbol{\eta}$ equal b_{uv}
      $\{\phi_{\mathbf{MV}\,i}(\boldsymbol{\eta})\}_i = \{\Phi_i(\boldsymbol{\eta})\}_i$ **return**
    else //see Sec. 3.4
      compute m_q //mean vector, Eq. 16
      compute minimal-norm solution $\bar{w}^q(\boldsymbol{\eta})$ //Eq. 20 or 28
      compute $\Omega^q(\boldsymbol{\eta})$ //*estimate*, Eq. 22
      compute λ //*estimate*, Eq. 24
      $w_{q_i}(\boldsymbol{\eta}) += \bar{w}^q_{q_i}(\boldsymbol{\eta}) + \lambda k_i \ \forall i < 4$ //full solution
    end if
  end if
end for
for all $i \in \mathcal{V}$ **do**
  $\phi_{\mathbf{MV}\,i}(\boldsymbol{\eta}) = w_i(\boldsymbol{\eta}) / \sum_j w_j(\boldsymbol{\eta})$
end for



(a) Input (b) MVC (c) GC (d) **QMVC** (e) MVC (f) GC (g) **QMVC**

Fig. 11. Cage-based deformation comparison between Mean Value Coordinate (MVC), Green Coordinates (GC) and our quad scheme (QMVC).



(a) Input (b) MVC (c) **QMVC**

Fig. 12. **Symmetry preservation**: on the contrary to triangle coordinates (middle), symmetric editing of the cage quads reflect in the deformation without distorsion with our approach (right).

4 ALGORITHM AND IMPLEMENTATION

We report in Alg. 2 the pseudo-code of our algorithm. Given a evaluation point $\boldsymbol{\eta}$, this algorithm computes our generalized coordinates $\{\phi_{\mathbf{MV}_i}(\boldsymbol{\eta})\}_i$ for triangle, quad and tri-quad cages as follow. First we initialize a null weight for each cage vertices. Second, we iterate over the cage triangle and accumulate per-triangle weights similar to standard MVC for each so-indexed vertices. Third, we iterate over each quad and detect whether we are in a special case ($\boldsymbol{\eta}$ belonging to the quad) or in a general one, where in this case we accumulate the per-quad weight on each indexed vertex as described in Sec. 3.4). Last, we compute the coordinate set through the normalization of the weights by their total sum. Note that the complexity of the projection of a point in the quad cage coordinate system is O(N) with N the number of cage vertices.

While Ju et al. [2005a] used a hard threshold to numerically detect whether $\boldsymbol{\eta}$ lies on a triangle, we use a similar test for detecting if a quad is planar (line 15 of Alg. 2), and if $\boldsymbol{\eta}$ lies on a (planar and non-planar, lines 15 and 23 of Alg. 2) quad [1].

5 RESULTS

We evaluate our coordinates on a collection of shapes and (tri-)quad cages used to propagate deformation from the cage to the inner volume, hence the embedded high resolution target shape. We use simple cages to better underline the behavior of our scheme compared to triangle ones. In particular, we compare to MVC and Green Coordinates (GC). Figure 11 shows two major benefits of our method when using quad cages: first, the induced deformation is more regular and avoids the large artifacts introduced by the artificial triangulation of the cage for triangle schemes. Second, the inherent symmetry of the cage deformation affects the high resolution target shape similarly, on the contrary to triangle schemes, for which achieving a symmetric morphing is almost impossible in these cases.

Figure 12 illustrates how symmetries in the cage deformation are reproduced on the target shape when using QMVC for freeform deformation, while the artificial edges (added by triangulating the cages to make them compatible with triangle MVC) introduce a significant amount of distortion. Figure 13 shows additional examples.

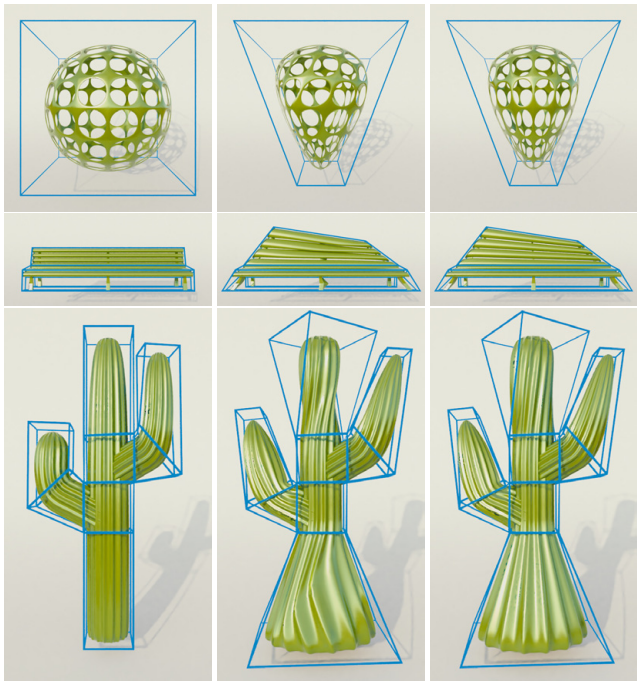


Fig. 13. **Additional comparisons**. From left to right: input, MVC and QMVC deformation.

[1] To further facilitate the use of our new coordinates system, we join a C++ reference implementation as a supplemental material to this paper, under an open source license.

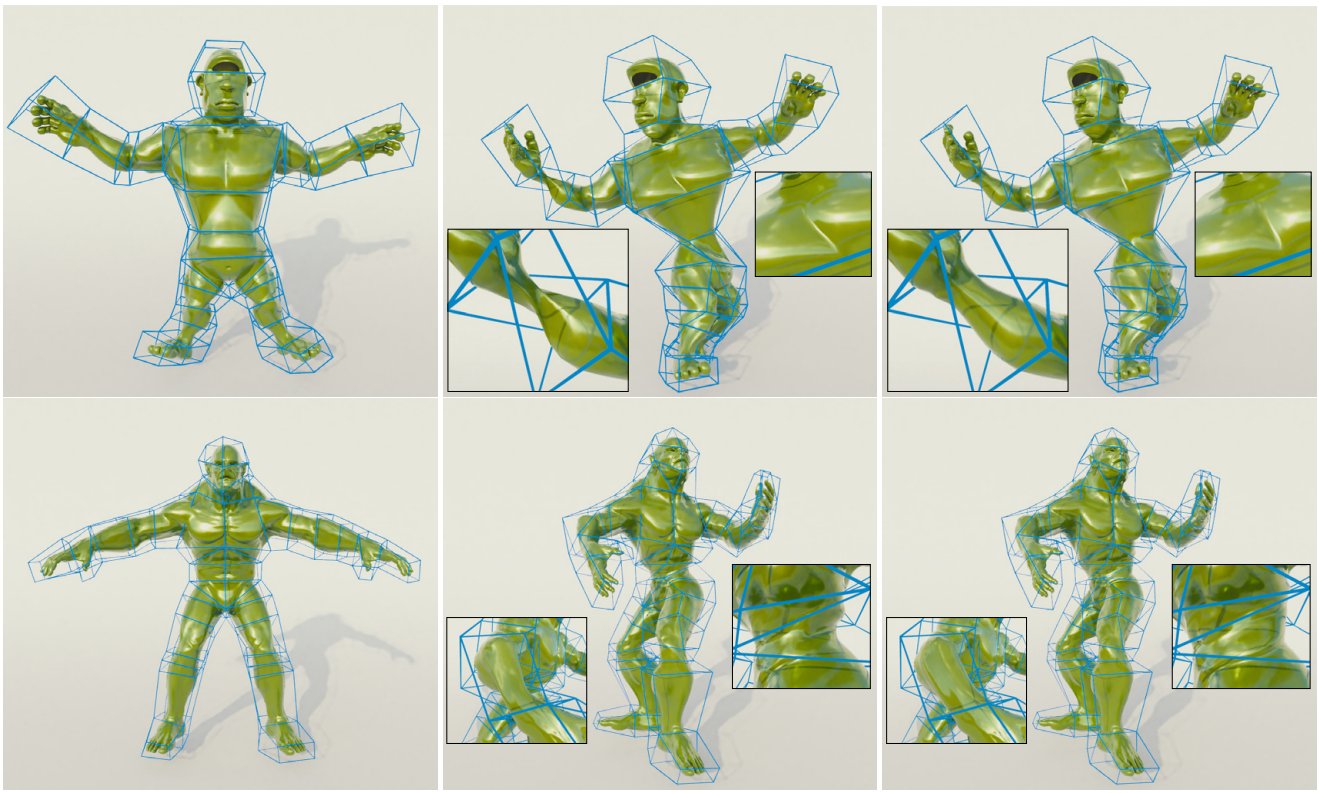


Fig. 14. **Triquad cages** on larger models. From left to right: input, MVC (with triangulated cages) and QMVC deformation (with triquad cages).

Figure 14 shows cage-based freeform deformations on larger models using tri-quad cages. Beyond the reduced distortion compared to MVC, we can observe that the induced space deformation accounts seamlessly for both triangles and quads with no artifacts. This makes it possible to use the right polygon type for the right shape component: quads for null curvature/tubular regions and triangles around isotropic high curvature/conic regions.

Comparison with Spherical Mean Value Coordinates (SMVC). Fig. 15 shows a comparison with SMVC [Langer et al. 2006]. While the method handles gracefully planar polygons with an arbitrary number of vertices, using the SMVC on non-planar quad cages leads to artifacts, as the method was not intended to be used with such input in the first place. We can observe in regions that are free of artifacts a similar behavior with QMVC. While our method uses a smooth quad geometry defined through 2D bilinear coordinates, SMVC lead to 2D mean value coordinates interpolation for points lying on the polygons and thus result in a slightly different but similar deformation behavior around these. Regions exhibitting artifacts correspond to evaluation points η where the mean vector of a quad has small (or zero) magnitude. For planar quads, a zero-norm mean vector corresponds to an evaluation point η lying on the quad's support plane and a zero area of its projection on the sphere centered in η, leading – as it should, to null unnormalized weights with respect to the quad's corners. For a non-planar quad however, there are regions in space where the mean vector is null but the unnormalized

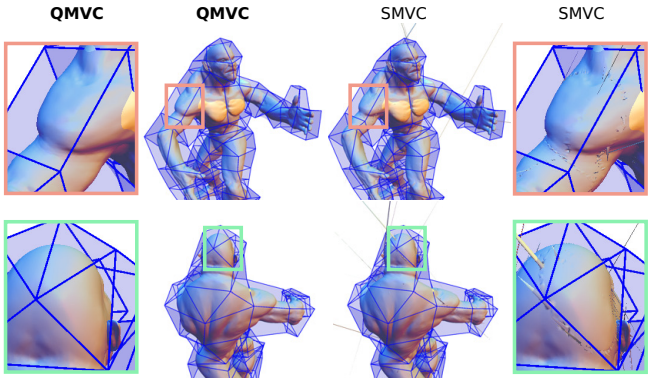


Fig. 15. Comparison with Spherical Mean Value Coordinates [Langer et al. 2006]. The input model is the *Beast* model used in Fig. 14 (bottom) along with its cage. The model was subdivided 2 times to emphasize the regions where non-planar quads result in unstable coordinates.

weights should not be null: consider an evaluation point from which the quad exhibits an "8" shape (see Fig. 5, second left for example). Around these regions, the mean vector – and all angle quantities that are derived from it in [Langer et al. 2006], are unstable and subject to strong bias. This explains why the artifacts are distributed so unevenly and localized around these regions (see Fig. 15, insets).

Note that the input shape and cage are displayed in Fig. 14 (bottom), and that the input cage features quads that are moderately non-planar for the most part. Note also that the instability of the mean vector around these regions is the main reason why our projection procedure was designed in a two-step manner, and we had to resort for the first step to projecting on the *convex hull of the quad*.

Comparison with Harmonic Coordinates. Contrary to HC [Joshi et al. 2007], our coordinates may be not positive for non-convex cages, and can result in counter-intuitive deformations for badly-posed input configurations e.g., where limbs are too close to each other. This problem is well documented and inherent to MVC. Being positive everywhere, HC naturally avoid this problem. We provide in Fig. 16 a side-by-side comparison with HC on several models. While HC are strictly positive everywhere, they can only be evaluated on points that are strictly inside the cage only, which can produce large artifacts if the cage is not adequately constructed.

This non-positivity issue can be tackled using various methods, such as Maximum Entropy Coordinates (MEC) [Hormann and Sukumar 2008], that can make an existing set of barycentric coordinates positive. In Fig. 17, we illustrate the impact of using MEC as a post-process to our coordinates, after having clamped the negative QMVC to 0 (no other result than the ones showcased in this figure were processed with MEC). MEC are obtained through the numerical optimization of a functional, but contrary to HC which require a global optimization solver (one cage vertex at a time, for all mesh vertices altogether), the MEC optimization is performed for each mesh vertex independently (one mesh vertex at a time, for all cage vertices altogether) and constructing these coordinates remains linear in both cage vertices and mesh vertices. Smoothness of the output MEC is ensured if the input masses prescribed to the solver are smooth, because the MEC function is regular, as demonstrated by Hormann and Sukumar [2008]. Note that we cannot claim C^∞ regularity everywhere for the resulting coordinates (QMVC+MEC) due to the clamping to 0, which is non-differentiable. In that sense, these coordinates are similar to PMVC [Lipman et al. 2007], which are computed using a non-differentiable visibility function. Note that the same visibility prior could be used as well, instead of the much simpler prior we used to create the results in Fig. 17. Note also that this strategy works only for points that are located *inside the cage*, similarly to HC.

Finally, a practical advantage of HC compared to our coordinates is that cages can feature inside non-manifold faces and wire-edges (see [Joshi et al. 2007], Section 3), as long as the cage is closed, in order to be able to define without ambiguity the *inside* and *outside* voxels. MVC cannot be generalized to support such control structures, because MVC rely on the fact that the cage is a closed manifold, and that the integral of its geometry after projection on the sphere centered in the evaluation point is always 0 in order to obtain *linear precision*.

Timings. We report in Tab. 1 the timings for the computations of the QMVC (for several values of the grid sampling parameter n), as well as timings for the computations of MVC and MEC. As expected, our method scales quadratically with n, and results for $n = 4$ (resp. $n = 10$) in computation timings that are roughly 10 (resp. 40) times higher than for computing the triangle-based MVC

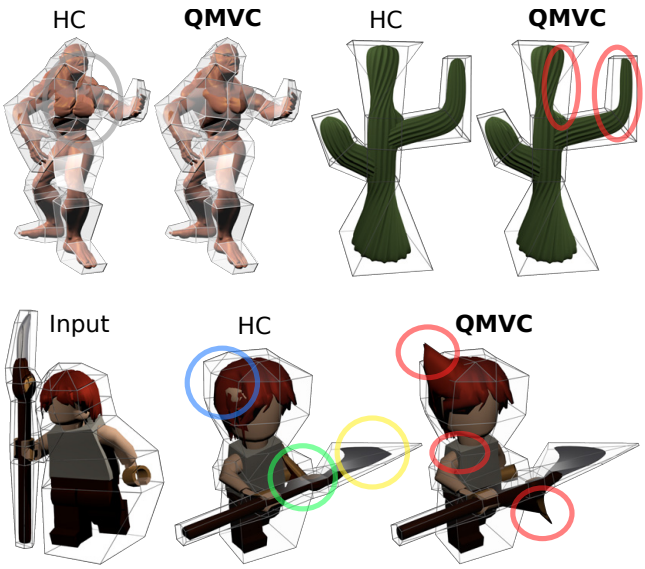


Fig. 16. Side-by-side comparison with Harmonic Coordinates [Joshi et al. 2007]. Red circles (Cactus and Lego): artifacts resulting from QMVC weight negativity. Grey circle (Beast): HC artifacts for parts located outside the cage. Blue circle (Lego): HC grid rounding artifact (probably early termination of computation). Green circle (Lego): HC grid bleeding artifact (the cage is glued because the limbs are too close to each other). Yellow circle (Lego): HC grid discretization artifact.

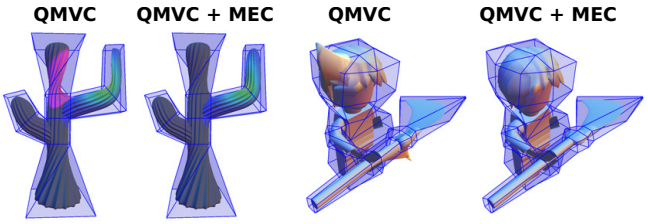


Fig. 17. Rendering positive our coordinates with MEC [Hormann and Sukumar 2008]. We display one basis function in the first example (Cactus), with purple colors indicating negative values. The negative values of this basis function explain why the cactus is crushed on the left, as a result of a translation of the branch to the right. The coordinates, once rendered positive with MEC, retain the overall behavior of the QMVC but do not exhibit negativity artifacts anymore.

with a cage, where each quad has been subdivided into two triangles. All results showcased in this paper were obtained with $n = 4$ which, based on our experiments, is sufficient in practice. All measures were performed on a laptop computer equipped with an Intel Core i7-8750H at 2.2 GHz.

6 DISCUSSION

Limitations & Future Work. The first limitation of our work is that our final coordinates are only an approximation of the ground truth QMVC, as integrals over curved bilinear quads rely on a sampling scheme.

Table 1. Detailed timings in ms. #M/#C: number of mesh/cage vertices. For completeness, we report timings for MVC (for which the quads have been subdivided into two triangles) and our implementation of Maximum Entropy Coordinates (MEC) [Hormann and Sukumar 2008], which we use as a post-process to render our coordinates positive. We used n=4 for the results illustrated in the paper.

Model (#M / #C)	MVC	QMVC **n = 4**	QMVC n = 7	QMVC n = 10	MEC
Hydrant (39k / 48)	86	1045	2133	4052	291
Bar (406k / 8)	185	2247	4738	8370	1666
SpikyBox (60.8k / 8)	44	337	705	1241	8
Head (24.6k / 12)	26	246	480	842	4
WireSphere (48.9k / 8)	29	273	556	1006	5
Bench (65.4k / 24)	101	1321	2780	4884	245
Cactus (98.8k / 36)	146	1960	3897	6988	607
Ogre (26.2k / 118)	120	1565	3270	5882	424
Beast (28.4k / 152)	164	2261	4574	8260	577
Lego (8.9k / 85)	40	395	800	1483	130

Then, our approach comes with guarantees only for quads which pass our validity predicate (see Def. 3.2). However, considering the strategy we promote in this work, our predicate is *conservative* and we suspect that it is sufficient but not necessary. It would be interesting to find the exact (larger) family of valid quads compatible with our approach, in particular since we believe that the design of a projection operator on any curved non-convex quad would find additional applications in modeling.

Moreover, our approach requires significantly more computations than usual triangle-based MVC (using a 9×9-grid, computing our QMVC for a quad takes roughly 10 times longer than computing the MVC over the two triangles obtained by cutting the quad in two).

Enabling quad meshes to act as cages opens several research directions. Our projection operator provides the coefficients of the basis functions of the projected point, interpolates the quad's geometry and is used as a sampling for a Riemann summation leading to surface interpolation. All these properties could make it instrumental in some fields of geometric modeling, in particular surface reconstruction and shape refinement.

Last, taking inspiration from other coordinate systems defined for triangle cages, new quad coordinate schemes could be defined using a similar strategy to ours; in particular, locally quasi-conformal deformations for quad cages could be designed taking Green coordinates as a basis. Our approach could also be extended to other kinds of polygons, or n-gons, with the benefit of employing coarser cages and avoiding superficial edges.

Conclusion. To the best of our knowledge, we have proposed the first space coordinate system for 3D surface cages made of arbitrary i.e., non-planar, quads. Our solution retains the simplicity of traditional space coordinate systems, with no global optimization, parallel scalability and linear precision. Our quad validity predicate allows automatically checking whether a cage can be used, with guarantees, for coordinate projection or not. Most importantly, the combination of our new quad projection operator with our new adaptive integration scheme provides high-quality surface-to-volume propagation even on challenging cases. Avoiding triangulating quad cages provides a significant boost in the propagation regularity, which translates immediately to smoother deformations and better preservation of symmetries. Additionally, our approach remains compatible with triangles and makes possible using tri-quad cages seamlessly.

Cage design is a critical step when using space coordinates for freeform deformation; with the user often ending up hand-crafting the cage vertices, edges and faces to obtain the ideal control rig. As such, our method widens the tool set to create such cages and, in particular, opens cage design to the dominant polygonal modelers: quad mesh ones.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments and remarks that helped improving our submission, in particular regarding MEC and SMVC. We also thank hjmediastudios (Blendswap.com) for sharing his Lego model.

REFERENCES

Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. 2009. Variational harmonic maps for space deformation. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 34.

Péter Borosán, Reid Howard, Shaoting Zhang, and Andrew Nealen. 2010. Hybrid Mesh Editing.. In *Eurographics (short papers)*. 41–44.

Max Budninskiy, Beibei Liu, Yiying Tong, and Mathieu Desbrun. 2016. Power coordinates: a geometric construction of barycentric coordinates on convex polytopes. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 241.

Stéphane Cadleron and Tamy Boubekeur. 2017. Bounding Proxies for Shape Approximation. *ACM Trans. Graph.* 36, 5, Article 57 (2017), 57:1–57:13 pages.

Michael S Floater. 2003. Mean value coordinates. *Computer aided geometric design* 20, 1 (2003), 19–27.

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

Kai Hormann and Natarajan Sukumar. 2008. Maximum entropy coordinates for arbitrary polytopes. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1513–1520.

Kai Hormann and N. Sukumar (Eds.). 2017. *Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics.* Taylor & Francis, CRC Press, Boca Raton.

Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, Li-Yi Wei, Shang-Hua Teng, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2006. Subspace Gradient Domain Mesh Deformation. *ACM Trans. Graph.* 25, 3 (2006), 1126–1134.

Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. 2014. Skinning: Real-time Shape Deformation. In *ACM SIGGRAPH Courses.*

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

Tao Ju, Peter Liepa, and Joe Warren. 2007. A general geometric construction of coordinates in a convex simplicial polytope. *Computer Aided Geometric Design* 24, 3 (2007), 161–178.

Tao Ju, Scott Schaefer, and Joe Warren. 2005a. Mean value coordinates for closed triangular meshes. In *ACM SIGGRAPH 2005 Papers.* ACM, 561–566.

Tao Ju, Scott Schaefer, Joe Warren, and Mathieu Desbrun. 2005b. A Geometric Construction of Coordinates for Convex Polyhedra Using Polar Duals. In *Proceedings of the Third Eurographics Symposium on Geometry Processing (SGP '05).* Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, Article 181. http://dl.acm.org/citation.cfm?id=1281920.1281950

Torsten Langer, Alexander Belyaev, and Hans-Peter Seidel. 2006. Spherical barycentric coordinates. In *Symposium on Geometry Processing.* 81–88.

Binh Huy Le and Zhigang Deng. 2017. Interactive Cage Generation for Mesh Deformation. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (SI3D).* 3:1–3:9.

Xian-Ying Li, Tao Ju, and Shi-Min Hu. 2013. Cubic Mean Value Coordinates. *ACM Transactions on Graphics* 32, 4 (2013), 126:1–10.

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 Transactions on Graphics (TOG)* 27, 3 (2008), 78:1–78:10.

August Ferdinand Möbius. 1827. *Der barycentrische calcul.*

Leonardo Sacht, Etienne Vouga, and Alec Jacobson. 2015. Nested Cages. *ACM Trans. Graph.* 34, 6, Article 170 (2015), 170:1–170:14 pages.

Scott Schaefer. 2017. Multi-Sided Patches via Barycentric Coordinates. In *Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics*, Kai Hormann and N. Sukumar (Eds.). Taylor & Francis, CRC Press, Boca Raton, Chapter 8, 135–146.

Jean-Marc Thiery, Julien Tierny, and Tamy Boubekeur. 2014. Jacobians and Hessians of mean value coordinates for closed triangular meshes. *The Visual Computer* 30, 9 (2014), 981–995.

Joe Warren, Scott Schaefer, Anil N Hirani, and Mathieu Desbrun. 2007. Barycentric coordinates for convex sets. *Advances in computational mathematics* 27, 3 (2007), 319–338.

Chuhua Xian, Hongwei Lin, and Shuming Gao. 2012. Automatic cage generation by improved OBBs for mesh deformation. *The Visual Computer* 28, 1 (2012), 21–33.

A CLOSED-FORM GEOMETRIC EXPRESSION OF THE KERNEL AND MINIMAL-NORM SOLUTION

When A_q is of rank 3, we recall that $\kappa_A = (k_0, k_1, k_2, k_3) \in \mathbb{R}^4$ is the unique (up to scaling) vector spanning its kernel (i.e., $A \cdot \kappa_A = 0$). Since $\sum_i k_i(\boldsymbol{q_i} - \boldsymbol{\eta}) = 0$, $\{k_i\}_i$ are simply the barycentric coordinates of $\boldsymbol{\eta}$ in the tetrahedron $(\boldsymbol{q_0 q_1 q_2 q_3})$, and therefore:

$$\begin{aligned} \kappa_A &:= (k_0, k_1, k2, k3), \\ k_i &:= (-1)^i |\boldsymbol{q_{i+1}} - \boldsymbol{\eta}; \boldsymbol{q_{i+2}} - \boldsymbol{\eta}; \boldsymbol{q_{i+3}} - \boldsymbol{\eta}| \end{aligned} \tag{26}$$

One can check that this formula is true, even if q is planar, as $\sum_i k_i \boldsymbol{q_i} = 0$ and $\sum_i k_i = 0$ in this case.

Noting $\bar{w}^q$ the "minimal-norm solution" to Eq. 17, $\bar{w}^q$ is the unique solution to the linear system

$$\begin{pmatrix} A_q \\ \kappa_A^T \end{pmatrix} \cdot \bar{w}^q = \begin{pmatrix} \boldsymbol{m}_q \\ 0 \end{pmatrix} , \tag{27}$$

which, noting $\boldsymbol{p_i} := \boldsymbol{q_i} - \boldsymbol{\eta}$, leads to the following closed-form expression (by using Cramer's rule, and developing all determinants along the last row using Laplace's formula):

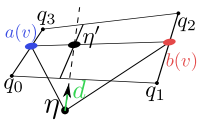
$$\begin{aligned} \bar{w}^q &= \begin{pmatrix} k_1|\boldsymbol{p_2};\boldsymbol{p_3};\boldsymbol{m}_q| + k_2|\boldsymbol{p_3};\boldsymbol{p_1};\boldsymbol{m}_q| + k_3|\boldsymbol{p_1};\boldsymbol{p_2};\boldsymbol{m}_q| \\ k_0|\boldsymbol{p_3};\boldsymbol{p_2};\boldsymbol{m}_q| + k_2|\boldsymbol{p_0};\boldsymbol{p_3};\boldsymbol{m}_q| + k_3|\boldsymbol{p_2};\boldsymbol{p_0};\boldsymbol{m}_q| \\ k_0|\boldsymbol{p_1};\boldsymbol{p_3};\boldsymbol{m}_q| + k_1|\boldsymbol{p_3};\boldsymbol{p_0};\boldsymbol{m}_q| + k_3|\boldsymbol{p_0};\boldsymbol{p_1};\boldsymbol{m}_q| \\ k_0|\boldsymbol{p_2};\boldsymbol{p_1};\boldsymbol{m}_q| + k_1|\boldsymbol{p_0};\boldsymbol{p_2};\boldsymbol{m}_q| + k_2|\boldsymbol{p_1};\boldsymbol{p_0};\boldsymbol{m}_q| \end{pmatrix} /D \\ D &:= k_0^2 + k_1^2 + k_2^2 + k_3^2 = \|\kappa_A\|^2. \end{aligned} \tag{28}$$

B INTERSECTION BETWEEN A LINE AND A QUAD

To intersect a line with q, we look for $(u, v) \in [0, 1]^2$, such that

$$\begin{aligned} \boldsymbol{\eta'} &= (\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3}) \cdot b_{uv} \\ &= (1-u) \underbrace{[(1-v)\boldsymbol{q_0} + v\boldsymbol{q_3}]}_{:=\boldsymbol{a}(v)} + u \underbrace{[(1-v)\boldsymbol{q_1} + v\boldsymbol{q_2}]}_{:=\boldsymbol{b}(v)} , \end{aligned}$$

and that $\boldsymbol{\eta'}$ belongs to the line $\mathcal{D}(\boldsymbol{\eta}, \boldsymbol{d})$. In this case, d should belong to the plane supported by the three points $\boldsymbol{\eta}$, $a(v)$ and $b(v)$, which is equivalent to solving for $[(\boldsymbol{a}(v) - \boldsymbol{\eta}) \times (\boldsymbol{b}(v) - \boldsymbol{\eta})] \cdot \boldsymbol{d} = 0$, which is a simple second order polynomial in v.



Once v is found, u can be computed similarly, or by finding $\boldsymbol{\eta'}$ as the intersection of the two lines $\mathcal{D}(\boldsymbol{\eta}, \boldsymbol{d})$ and $\mathcal{D}(\boldsymbol{a}(v), \boldsymbol{b}(v) - \boldsymbol{a}(v))$, and computing u as

$$u = (\boldsymbol{\eta'} - \boldsymbol{a}(v)) \cdot (\boldsymbol{b}(v) - \boldsymbol{a}(v)) \tag{29}$$

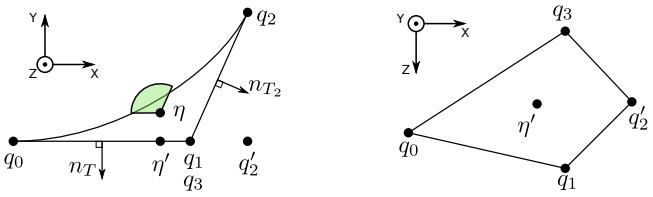


Fig. 18. For the proof of Lemma 3.1. Left: side view. Right: top view.

C PROOF OF LEMMA 3.1

Lemma C.1. *$\mathcal{P}_q$ is a mapping of the Euclidean space to the bilinear quad q (i.e., $\mathcal{P}_q(\boldsymbol{\eta}) \in [0,1]^2 \ \forall \boldsymbol{\eta} \in \mathbb{R}^3$) if q is valid, according to Def. 1.*

Proof. The first projection maps trivially any point to a point on the convex hull of q. What remains to prove is that the line $\mathcal{D}(\boldsymbol{\eta}, \boldsymbol{m}_q(\boldsymbol{\eta}))$ intersects the bilinear quad, for any point $\boldsymbol{\eta}$ in the convex hull.

For clarity, we align the quad's geometry in a coordinate system $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ such that $\boldsymbol{Z}$ is aligned with $\overrightarrow{\boldsymbol{q_1 q_3}}$, and that $(\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_3})$ is contained in the $\boldsymbol{XZ}$ plane (see Fig. 18, left). For a valid quad, according to the convexity of the projected quad, $\boldsymbol{q_2}$ is guaranteed to lie on the right of $\boldsymbol{q_1}$ and $\boldsymbol{q_3}$.

Any line going through $\boldsymbol{\eta}$ (which is located inside the convex hull) will intersect the convex hull exactly twice, on two different triangles. To ensure that $\mathcal{D}(\boldsymbol{\eta}, \boldsymbol{m}_q(\boldsymbol{\eta}))$ intersects the bilinear quad, it is enough to check that the line cannot intersect both triangles $T = (\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_3})$ and $T_2 = (\boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3})$, which is true if $sign(\boldsymbol{m_q} \cdot \boldsymbol{n_T}) = sign(\boldsymbol{m_q} \cdot \boldsymbol{n_{T_2}})$ (see Fig. 18, left: the green zone describes acceptable directions for the mean vector). If that is the case, then the line will have intersected two triangles of the convex hull, which are separated by the bilinear quad. It will have thus intersected the bilinear quad.

We note $\boldsymbol{\eta'}$ and $\boldsymbol{q'_2}$ the projections of $\boldsymbol{\eta}$ and $\boldsymbol{q_2}$ in the $\boldsymbol{XZ}$ plane. Using once again that the integral of the unit normal is null on any closed surface, we write the mean vector $\boldsymbol{m_q}$ as

$$\begin{aligned} \boldsymbol{m_q} &= (\theta_0 \boldsymbol{n_0} + \theta_1 \boldsymbol{n_1} + \theta_2 \boldsymbol{n_2} + \theta_3 \boldsymbol{n_3})/2 \text{ , with} \\ \boldsymbol{N_i} &:= (\boldsymbol{q_{i+1}} - \boldsymbol{\eta}) \times (\boldsymbol{q_i} - \boldsymbol{\eta}) \\ \theta_i &:= \angle(\boldsymbol{q_i \eta q_{i+1}}) \\ \boldsymbol{n_i} &= \boldsymbol{N_i}/\|\boldsymbol{N_i}\| \ \ \forall i \end{aligned}$$

We need to prove that $\boldsymbol{m_q} \cdot \boldsymbol{n_T} \leq 0$ necessarily (by a symmetry argument we will have to prove that $\boldsymbol{m_q} \cdot \boldsymbol{n_{T_2}} \leq 0$). This is true since, according to the validity condition, the projection of the quad on the $\boldsymbol{XZ}$ plane is convex. Therefore, we have (see Fig. 18, right)

$$\boldsymbol{N_i} \cdot \boldsymbol{n_T} = -\boldsymbol{N_i} \cdot Y = -2area(\boldsymbol{q_i q_{i+1} \eta'}) \leq 0 \ \forall i \ .$$

□