Jacobians and Hessians of Mean Value Coordinates for Closed Triangular Meshes

Jean-Marc Thiery, Julien Tierny, Tamy Boubekeur

The Visual Computer manuscript No.
(will be inserted by the editor)

Jacobians and Hessians of Mean Value Coordinates for Closed Triangular Meshes

Jean-Marc Thiery · Julien Tierny · Tamy Boubekeur

Received: date / Accepted: date

Abstract Mean Value Coordinates provide an efficient mechanism for the interpolation of scalar functions defined on orientable domains with non-convex boundary. They present several interesting features, including the simplicity and speed that yield from their closed-form expression. In several applications though, it is desirable to enforce additional constraints involving the partial derivatives of the interpolated function, as done in the case of the Green Coordinates approximation scheme [2] for interactive 3D model deformation.

In this paper, we introduce the analytic expressions of the Jacobian and the Hessian of functions interpolated through Mean Value Coordinates. We provide these expressions both for the 2D and 3D case. We also provide a thorough analysis of their degenerate configurations along with accurate approximations of the partial derivatives in these configurations. Extensive numerical experiments show the accuracy of our derivation. In particular, we illustrate the improvements of our formulae over a variety of Finite Difference schemes in terms of precision and usability. We demonstrate the utility of this derivation in several applications, including cage-based implicit 3D model deformations (i.e. *Variational MVC deformations*). This technique allows for easy and interactive model deformations with sparse positional, rotational and smoothness constraints. Moreover, the cages produced by the algorithm can be directly re-used for further manipulations, which makes our framework directly compatible with existing software supporting Mean Value Coordinates based deformations.

Keywords Cage coordinates · Mean Value Coordinates · Constrained interpolation · Implicit cage deformation

J.-M. Thiery
Telecom-ParisTech – CNRS / LTCI, 43 rue Barrault, 75013 Paris, France
E-mail: thiery@telecom-paristech.fr

J. Tierny
Telecom-ParisTech – CNRS / LTCI

T. Boubekeur
Telecom-ParisTech – CNRS / LTCI

1 Introduction

Boundary value interpolation is a common problem in computer-aided design, simulation, visualization, computer graphics and geometry processing. Given a polygonal domain with prescribed scalar values on its boundary vertices, barycentric coordinate schemes enable to compute a smooth interpolation of the boundary values at any point of the Euclidean space located in the interior (and possibly the exterior) of the domain. Such interpolations are efficiently obtained through a linear combination involving a weight (or *coordinate*) for each boundary vertex. These weights can be obtained by solving a system of linear equations [16] or, more efficiently, through a closed-form expression [12, 17].

However, in several applications, it may be desirable to enforce additional constraints on the interpolation, in particular constraints involving the partial derivatives of the interpolated function. Derivative constraints have been shown to provide additional flexibility to the interpolation problem and many optimization tasks can benefit from them. In the context of *approximation* schemes based on Green Coordinates [20], constraints on the Jacobian and the Hessian have been used for implicit cage-based deformations [2]. In such a setting, given some sparse user constraints, an optimization process automatically retrieves an embedding of the polygonal

Fig. 1 "Kicking Demon". Red points indicate positional constraints; blue points indicate unknown rotational constraints. This pose was obtained by specifying only 14 positional constraints.
The derivation of the Jacobian and Hessian of the MVC coordinates makes it possible to induce *variational* MVC deformations (implicit cage deformation based on sparse user constraints, with rotation and smoothness enforcement).

domain (the *cage*) such that the transformation of the interior space is locally close to a rotation.

In order to achieve acceptable precision and time efficiency, such an optimization process requires a closed-form expression of the Jacobian and the Hessian of the interpolated function. While such closed-form expressions are known for *approximation* schemes (in particular for Green Coordinates [2, 26]), this is not the case for *interpolation* schemes. Moreover, these formulations are specific to the context of space deformation and it is not clear how to extend them to arbitrary functions.

In this paper, we bridge this gap by deriving the closed-form expressions of the Jacobians and the Hessians of functions interpolated with Mean Value Coordinates (MVC) [12, 17], both for the 2D and 3D case. We also provide a complete analysis of their degenerate configurations (on the support of the simplices of the cage [17]) along with accurate approximations of the derivatives for these configurations. We show the accuracy of this derivation with extensive numerical experiments.

We demonstrate the utility of this derivation for several applications, including cage-based implicit 3D model deformations (i.e. *Variational MVC deformations*). This technique allows for easy and interactive model deformations with sparse positional, rotational and smoothness constraints.

The cages produced by our algorithm can be directly re-used for further manipulations, which makes our framework directly compatible with existing software supporting MVC-based deformations. In addition, we provide as supplemental material a lightweight C++ implementation of our derivation, enabling its usage in further optimization problems involving constrained interpolation.

1.1 Related work

Boundary value interpolation through barycentric coordinates has been widely studied in the case of convex domains, first in 2D [4, 5, 22], and more recently in higher dimensions [27, 28]. Several techniques have been proposed to extend these interpolants to non-convex 2D domains [9, 10, 11, 14, 21]. In particular, Floater introduced a coordinate system motivated by the Mean Value Theorem, that smoothly interpolates function values defined on concave 2D polygons [11].

The generalization of these techniques to non-convex 3D domains has been motivated mostly by computer graphics applications, in particular interactive shape deformation. In this setting, a 3D shape (in the form of a triangle soup, point cloud or volumetric mesh) is enclosed by a closed triangle surface called a *cage*, from which barycentric coordinates are computed. A deformation of the cage yields a transformation of its embedding functions f_x, f_y and f_z. These functions can be interpolated efficiently and smoothly in the interior space enclosed by the cage, providing a mean to interactively deform the interior 3D shape. Mean Value Coordinates (MVC) have been generalized independently to non-convex 3D domains by Floater [12], Ju et al. [17] and further by Langer et al. [18], with applications to boundary value interpolation, volumetric texturing, shape deformation and speed-up of deformations based on non-linear systems [15]. To overcome artifacts occurring in highly concave portions of the cage, Joshi et al. [16] introduced the Harmonic Coordinates (HC). However, these coordinates do not admit a closed form expression and require a numerical solver for their computation. Lipman et al. [20] introduced the Green Coordinates (GC), which induce near-conformal (detail

preserving) transformations and which admit a closed form expression. However, these coordinates define an *approximation* scheme, not an *interpolation* one.

Among the existing barycentric coordinate systems which support non-convex 3D domains and which admit closed-form expressions (MVC and GC), the expressions of the Jacobian and the Hessian are only known for the Green Coordinates [2, 26]. In particular, Ben Chen et al. [2] proposed an implicit cage-based deformation technique called *Variational Harmonic Maps* (VHM), where the target embedding of the cage is automatically optimized from sparse user-imposed positional constraints with rotational and smoothness constraints respectively involving the Jacobian and the Hessian of the deformation. As the 3D values attributed to the normals of the cage triangles are de-correlated from the 3D values attributed to its vertex positions in their optimization process, the cage cannot be manipulated in a post process for further detail editing. In this work, we introduce these expressions for functions interpolated with Mean Value Coordinates. We also develop an application to implicit cage-based transformations similar to VHM [2]. In contrast to VHM, the solution of the optimization process does not involve the normals of the cage triangles, hence the cages generated by this technique cannot be exploited in a consistent manner for post-processing tasks. On the contrary, the cages produced by our algorithm can be directly re-used for further manipulations, which makes our framework directly compatible with existing software supporting MVC-based deformations.

1.2 Contributions

This paper makes the following contributions:

1. the closed-form expressions of the Jacobian and the Hessian of Mean Value Coordinates, both in 2D and 3D – these expressions are more accurate than a variety of experimented Finite Difference schemes and they are not prone to numerical instability;
2. a thorough analysis of the degenerate configurations of these expressions, along with accurate alternate approximations for these configurations;
3. an implicit cage-based transformation technique using Mean Value Coordinates, called *Variational MVC Deformation*, which interactively optimizes the target cage embedding given sparse user positional constraints, while respecting smoothness and rotational constraints;

1.3 Overview

We first review Mean Value Coordinates in Sec. 2. The core contribution of our work, the derivation of the Jacobian and Hessian, is presented in Sec. 3 through 6. In particular, Sec. 4 and 5 provide the specific results for the 2D and 3D cases respectively.

As the derivation of the Jacobian and the Hessian is relatively involved, for the reader's convenience, we highlighted the final expressions with rectangular boxes, whereas the final expressions for degenerate cases are highlighted with a ellipsoidal box. Note, that the derivation details are given in Appendix, provided as **supplemental material**.

Experimental evidence of the accuracy of our derivation is presented in Sec. 7.

Finally we present applications demonstrating the utility of our contributions in Sec. 8.

2 Background

In this section, we review the formulation of Mean Value Coordinates in 2D and 3D [17].

2.1 Mean Value Coordinates

Similar to [17], we note $p[x]$ a parameterization of a closed $(d-1)$-manifold mesh (the cage) M embedded in $\mathbb{R}^d$, where x is a $(d-1)$-dimensional parameter, and n_x the unit outward normal at x. Let η be a point in $\mathbb{R}^d$ expressed as a linear combination of the positions p_i of the vertices of the cage M:

$$\eta = \frac{\sum_i w_i p_i}{\sum_i w_i} = \sum_i \lambda_i p_i$$

where λ_i is the *barycentric coordinate* of η with respect to the vertex i.

Let $\phi_i[x]$ be the linear function on M which maps the vertex i to 1 and all other vertices to 0.

The definition of the coordinates λ_i should guarantee *linear precision* (i.e. $\eta = \sum_i \lambda_i p_i$).

Similar to [17], we note $B_\eta(M)$ the projection of the manifold M onto the unit sphere centered around η, and $dS_\eta(x)$ the infinitesimal element of surface on this sphere at the projected point ($dS_\eta(x) = \frac{(p[x]-\eta)^t \cdot n_x}{|p[x]-\eta|^3} dx$ in 3D)

Since $\int_{B_\eta(M)} \frac{p[x]-\eta}{|p[x]-\eta|} dS_\eta(x) = 0$ (the integral of the unit outward normal onto the unit sphere is 0 in any dimension $d \geq 2$), the following equation holds:

$$\eta = \frac{\int_{B_\eta(M)} \frac{p[x]}{|p[x]-\eta|} dS_\eta(x)}{\int_{B_\eta(M)} \frac{1}{|p[x]-\eta|} dS_\eta(x)}$$

By writing $p[x] = \sum_i \phi_i[x] p_i \ \forall x$, with $\sum_i \phi_i[x] = 1$, we have:

$$\eta = \frac{\sum_i \int_{B_\eta(M)} \frac{\phi_i[x]}{|p[x]-\eta|} dS_\eta(x) p_i}{\int_{B_\eta(M)} \frac{1}{|p[x]-\eta|} dS_\eta(x)}$$

The coordinates λ_i are then given by:

$$\lambda_i = \frac{\int_{B_\eta(M)} \frac{\phi_i[x]}{|p[x]-\eta|} dS_\eta(x)}{\int_{B_\eta(M)} \frac{1}{|p[x]-\eta|} dS_\eta(x)}$$

and the weights w_i such that $\lambda_i = \frac{w_i}{\sum_i w_i}$ are given by:

$$w_i = \int_{B_\eta(M)} \frac{\phi_i[x]}{|p[x]-\eta|} dS_\eta(x) \tag{1}$$

This definition guarantees linear precision [17]. It also provides a linear interpolation of the function prescribed at the vertices of the cage onto its simplices and it smoothly extends it to the entire space. This construction of Mean Value Coordinates is valid in any dimension $d \geq 2$. In the following, we present their computation in 2D and in 3D, as they were introduced in [17].

2.2 3D Mean Value Coordinates computation

The support of the function $\phi_i[x]$ is only composed of the adjacent triangles to the vertex i (noted $N1(i)$). Eq. 1 can be re-written as $w_i = \sum_{T \in N1(i)} w_i^T$, with

$$w_i^T = \int_{B_\eta(T)} \frac{\phi_i[x]}{|p[x]-\eta|} dS_\eta(x) \tag{2}$$

Given a triangle T with vertices t_1, t_2, t_3, the following equation holds:

$$\begin{aligned} \sum_j w_{t_j}^T (p_{t_j} - \eta) &= \int_{B_\eta(T)} \frac{\sum_j \phi_{t_j}[x](p_{t_j}-\eta)}{|p[x]-\eta|} dS_\eta(x) \\ &= \int_{B_\eta(T)} \frac{p[x]-\eta}{|p[x]-\eta|} dS_\eta(x) \triangleq m^T \end{aligned} \tag{3}$$

The latter integral is the integral of the unit outward normal on the spherical triangle $\overline{T} = B_\eta(T)$ (see Fig. 2). By noting the unit normal as $n_i^T = \frac{N_i^T}{|N_i^T|}$, with $N_i^T \triangleq (p_{t_{i+1}} - \eta) \wedge (p_{t_{i+2}} - \eta)$ (see Fig.2), m^T is given by (since the integral of the unit normal on a closed surface is always 0):

$$m^T = \sum_i \frac{1}{2} \theta_i^T n_i^T \tag{4}$$

As suggested in [17], the weights $w_{t_j}^T$ can be obtained by noting A^T the 3x3 matrix $\{p_{t_1} - \eta, p_{t_2} - \eta, p_{t_3} - \eta\}$ (where ^{t} denotes the transpose):

$$\{w_{t_1}^T, w_{t_2}^T, w_{t_3}^T\}^t = A^{T^{-1}} \cdot m^T$$

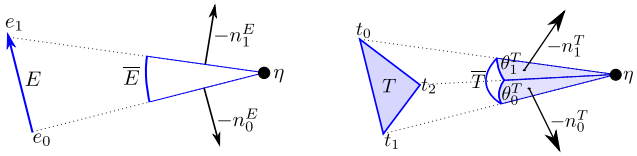


Fig. 2 Spherical edge $\bar{\mathrm{E}}$ (left) and triangle $\bar{\mathrm{T}}$(right).

Since $N_i^{T^t} \cdot (p_{t_j} - \eta) = 0 \quad \forall i \neq j$ and $N_i^{T^t} \cdot (p_{t_i} - \eta) = \det(A^T) \ \ \forall i$, the final expression for the weights is given by:

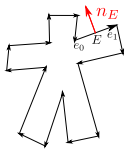
$$w_{t_i}^T = \frac{N_i^{T^t} \cdot m^T}{N_i^{T^t} \cdot (p_{t_i} - \eta)} = \frac{N_i^{T^t} \cdot m^T}{\det(A^T)} \quad \forall \eta \notin Support(T) \tag{5}$$

where $Support(T)$ denotes the support plane of T, i. e. $Support(T) = \{\eta \in \mathbb{R}^3 | \det(A^T)(\eta) = 0\}$.

2.3 2D Mean Value Coordinates computation

Let I_2 be the 2×2 identity matrix and $R_{\frac{\pi}{2}}$ the rotation matrix $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

In 2D, the orientation of an edge $E = e_0 e_1$ of a closed polygon is defined by the normal n_E:



$$n_E = \frac{R_{\frac{\pi}{2}}(p_{e_1} - p_{e_0})}{|p_{e_1} - p_{e_0}|}$$

It defines consistently the *interior* and the *exterior* of the closed polygon. Then, similarly to the 3D case:

$$\sum_j w_{e_j}^E \cdot (p_{e_j} - \eta) = m^E = \sum_j n_j^E \tag{6}$$

with:

$$n_j^E = \frac{N_j^E}{|N_j^E|}, \ N_0^E = R_{\frac{\pi}{2}}(\eta - p_{e_0}), \ N_1^E = -R_{\frac{\pi}{2}}(\eta - p_{e_1})$$

Therefore:

$$m^E = R_{\frac{\pi}{2}}\left(\frac{\eta - p_{e_0}}{|\eta - p_{e_0}|} - \frac{\eta - p_{e_1}}{|\eta - p_{e_1}|}\right) \tag{7}$$

Since $(p_{e_j} - \eta)^t \cdot N_j^E = 0$ (Fig.2), we obtain w_i^E with:

$$w_{e_i}^E = \frac{m^{E^t} \cdot N_{i+1}^E}{(p_{e_i} - \eta)^t \cdot N_{i+1}^E} \tag{8}$$

3 Derivation Overview

In the following, we present the main contribution of the paper: the derivation of the Jacobians and the Hessians of Mean Value Coordinates. In this section, we briefly give an overview of the derivation.

Let $f : M \to \mathbb{R}^d$ be a piecewise linear field defined on M (in 2D, M is a closed polygon, in 3D, M is a closed triangular mesh). As reviewed in the previous section, f can be smoothly interpolated with Mean Value Coordinates for any point η of the Euclidean space:

$$f(\eta) = \sum_i \lambda_i \cdot f(p_i)$$

Then, the Jacobian and the Hessian of f, respectively noted Jf and Hf, are expressed as the linear tensor product of the values $f(p_i)$ with the gradient $\overrightarrow{\bigtriangledown}\lambda_i$ and the Hessian $H\lambda_i$ of the coordinates respectively:

$$\begin{cases} Jf &= \sum_i f(p_i) \cdot \overrightarrow{\bigtriangledown}\lambda_i{}^t \\ Hf &= \sum_i f(p_i) \cdot H\lambda_i \end{cases}$$

Since $\lambda_i = \frac{w_i}{\sum_j w_j}$,

$$\overrightarrow{\bigtriangledown}\lambda_i = \frac{\overrightarrow{\bigtriangledown} w_i}{\sum_j w_j} - \frac{w_i \cdot \sum_j \overrightarrow{\bigtriangledown} w_j}{(\sum_j w_j)^2} \tag{9}$$

Then, the Hessian can be obtained with the following equations

$$\begin{aligned} H\lambda_i =& \frac{Hw_i}{\sum_j w_j} - \frac{w_i \sum_j Hw_j}{(\sum_j w_j)^2} \\ & - \frac{\overrightarrow{\bigtriangledown} w_i \cdot \sum_j \overrightarrow{\bigtriangledown}(w_j)^t + \sum_j \overrightarrow{\bigtriangledown}(w_j) \cdot \overrightarrow{\bigtriangledown} w_i{}^t}{(\sum_j w_j)^2} \\ & + \frac{2w_i(\sum_j \overrightarrow{\bigtriangledown} w_j) \cdot (\sum_j \overrightarrow{\bigtriangledown} w_j)^t}{(\sum_j w_j)^3} \end{aligned} \tag{10}$$

The above expressions are general and valid for the 2D and 3D cases. Thus, in order to derive a closed-form expression of the gradient and the Hessian of the Mean Value Coordinates λ_i, one needs to derive the expressions of $\overrightarrow{\bigtriangledown} w_i$ (Eq. 9) and Hw_i (Eq. 10). The expressions of these terms are derived in Sec. 4.1 and Sec. 4.2 respectively for the 2D case and in Sec. 5.1 and Sec. 5.2 for the 3D case.

3.1 Properties

Functions interpolated by means of Mean Value Coordinates as previously described have the following properties:

1. they are interpolant on M
2. they are defined everywhere in $\mathbb{R}^d$
3. they are C^∞ everywhere not on M
4. they are C^0 on the edges (resp. vertices) of M in 3D (resp. in 2D)

Since these are interpolant of piecewise linear functions defined on a piecewise linear domain, they cannot be differentiable on the edges of the triangles (resp. the vertices of the edges) of the cage in 3D (resp. in 2D). Although, as they are continuous everywhere, they may admit in these cases **directional derivatives** like almost all continuous functions do. Recall that the directional derivative of the function f in the direction u is the value $\partial f_u(\eta) = \lim_{\epsilon \to 0^+} \frac{f(\eta+\epsilon \cdot u) - f(\eta)}{\epsilon}$, with $u \in \mathbb{R}^3, ||u|| = 1, \epsilon \in \mathbb{R}$, which strongly depends on the orientation of the vector u where the limit is considered. These derivatives cannot be used to evaluate nor constrain the function around the point in general with a single gradient (or Jacobian if the function is multidimensional).

In this paper, we provide formulae for the 1^{st} and 2^{nd} order derivatives of the Mean Value Coordinates everywhere in space but on the cage.

4 MV-Gradients and Hessians in 2D

For conciseness, the details of this derivation are given in Appendix (additional material) and only the final expressions are given here.

In the following, we note (pq) the line going through the points p and q, and $[pq]$ the line segment between them.

4.1 Expression of the MV-gradients

Given an edge $E = e_0e_1$, in the general case where $(p_{e_i} - \eta)^t \cdot N^E_{i+1} \neq 0 (\eta \notin (p_{e_0}p_{e_1}))$, the gradient of the weights is given by the following expression:

$$\boxed{\overrightarrow{\bigtriangledown} w^E_{e_i} = \frac{Jm^{E\,t} \cdot N^E_{i+1}}{(p_{e_i} - \eta)^t \cdot N^E_{i+1}} + \frac{\sum_j w^E_{e_j} N^E_{i+1}}{(p_{e_i} - \eta)^t \cdot N^E_{i+1}} \quad \forall \eta \notin (p_{e_0}p_{e_1})} \tag{11}$$

with

$$\begin{aligned} Jm^E = R_{\frac{\pi}{2}} (& \frac{I_2}{|\eta - p_{e_0}|} - \frac{I_2}{|\eta - p_{e_1}|} \\ & - \frac{(\eta - p_{e_0}) \cdot (\eta - p_{e_0})^t}{|\eta - p_{e_0}|^3} \\ & + \frac{(\eta - p_{e_1}) \cdot (\eta - p_{e_1})^t}{|\eta - p_{e_1}|^3}) \end{aligned} \tag{12}$$

Special case: $\eta \in (p_{e_0}p_{e_1}), \notin [p_{e_0}p_{e_1}]$

$$\boxed{\overrightarrow{\bigtriangledown} w_{e_i}^E = (\sum_j \frac{{N_j^E}^t \cdot N_{i+1}^E}{2|E||N_j^E|^3} + \frac{(-1)^{i+j}}{|E||N_j^E|})n_E \qquad (13)}$$
$$\forall \eta \in (p_{e_0}p_{e_1}), \notin [p_{e_0}p_{e_1}]$$

4.2 Expression of the MV-Hessians

We define $\delta_x = \begin{pmatrix}1\\0\end{pmatrix}$ and $\delta_y = \begin{pmatrix}0\\1\end{pmatrix}$.

$$\boxed{Hw_{e_i}^E = \begin{pmatrix}\partial_x(\overrightarrow{\bigtriangledown} w_{e_i}^E)^t\\ \partial_y(\overrightarrow{\bigtriangledown} w_{e_i}^E)^t\end{pmatrix} \qquad (14)}$$
$$\forall \eta \notin (p_{e_0}p_{e_1})$$

with

$$\left\{\begin{array}{l}\partial_x(\overrightarrow{\bigtriangledown} w_{e_i}^E) = \frac{{C_x^E}^t \cdot N_{i+1}^E}{(p_{e_i}-\eta)^t \cdot N_{i+1}^E}\\ \partial_y(\overrightarrow{\bigtriangledown} w_{e_i}^E) = \frac{{C_y^E}^t \cdot N_{i+1}^E}{(p_{e_i}-\eta)^t \cdot N_{i+1}^E}\end{array}\right\} \quad \forall \eta \notin (e_0e_1).$$

and

$$\left\{\begin{array}{l}C_x^E = \sum_i \delta_x \cdot \overrightarrow{\bigtriangledown} {w_{e_i}^E}^t + \partial_x(Jm^E) + \sum_i \partial_x(w_{e_i}^E) \cdot I_2\\ C_y^E = \sum_i \delta_y \cdot \overrightarrow{\bigtriangledown} {w_{e_i}^E}^t + \partial_y(Jm^E) + \sum_i \partial_y(w_{e_i}^E) \cdot I_2\end{array}\right.$$

and

$$\begin{aligned}\partial_c(Jm^E) =& R_{\frac{\pi}{2}} \cdot (\frac{(\eta-p_{e_1})_{(c)}I_2}{|\eta-p_{e_1}|^3} - \frac{(\eta-p_{e_0})_{(c)}I_2}{|\eta-p_{e_0}|^3}\\ &- \frac{\delta_c \cdot (\eta-p_{e_0})^t + (\eta-p_{e_0}) \cdot {\delta_c}^t}{|\eta-p_{e_0}|^3}\\ &+ \frac{3(\eta-p_{e_0})_{(c)}(\eta-p_{e_0}) \cdot (\eta-p_{e_0})^t}{|\eta-p_{e_0}|^5}\\ &+ \frac{\delta_c \cdot (\eta-p_{e_1})^t + (\eta-p_{e_1}) \cdot {\delta_c}^t}{|\eta-p_{e_1}|^3}\\ &- \frac{3(\eta-p_{e_1})_{(c)}(\eta-p_{e_1}) \cdot (\eta-p_{e_1})^t}{|\eta-p_{e_1}|^5})\end{aligned}$$

Special case: $\eta \in (p_{e_0}p_{e_1}), \notin [p_{e_0}p_{e_1}]$

$$\boxed{\begin{aligned}Hw_{e_i}^E = \overrightarrow{\bigtriangledown}(dw_{e_i}^E) \cdot n_E^t + n_E \cdot \overrightarrow{\bigtriangledown}(dw_{e_i}^E)^t\\ \forall \eta \in (p_{e_0}p_{e_1}), \notin [p_{e_0}p_{e_1}] \qquad (15)\end{aligned}}$$

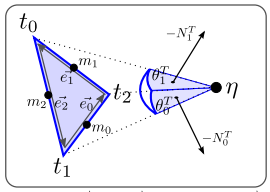
with

$$\begin{aligned}\overrightarrow{\bigtriangledown}(dw_{e_i}^E) =& R_{\frac{\pi}{2}} \cdot \sum_j \frac{3(-1)^{i+1}N_j^E + (-1)^j N_{i+1}^E}{2|E||N_j^E|^3}\\ &+ R_{\frac{\pi}{2}} \cdot \sum_j \frac{3(-1)^{j+1}({N_j^E}^t \cdot N_{i+1}^E)N_j^E}{2|E||N_j^E|^5}\end{aligned}$$

5 MV-Gradients and Hessians in 3D

For conciseness, the details of this derivation are given in Appendix (additional material) and only the final expressions are given here.

In the following, we note $e_i(x)$ a set of functions that are well-defined functions on $]0, \pi[$ and admit well-controlled Taylor expansions around 0. These Taylor expansions are given in Appendix (additional material). Note, that the functions $e_i(x)$ are not defined in 0 and that we make use of the Taylor expansions to estimate their values near 0 as well as in 0. The reason these terms appear in the final expressions is, that we organized the terms in order to provide formulae whose evaluation converges everywhere, avoiding the typical 0/0 and $+\infty + -\infty$ cases for example. To simplify the expressions, we translate all 3D quantities to the origin (i. e. $\hat{p} := p - \eta$). We also note $u_{ij} = \overrightarrow{e_i}^t \cdot \overrightarrow{e_j}$ the dot product between edges i and j, $m_i = (p_{t_{i+2}} + p_{t_{i+1}})/2$ the midpoint of the edge i of the triangle, and $J_{ij} = JN_i^T \cdot N_j^T$ the vectorial product between edge i of the triangle and the normal of the triangle supported by the edge j (see inset). From the expression of N_j^T, we obtain that its jacobian equals



$$JN_j^T = \overrightarrow{e_j}_{[\wedge]} \qquad (16)$$

, where $k_{[\wedge]}$ is the skew 3×3 matrix (i.e. ${k_{[\wedge]}}^t = -k_{[\wedge]}$) such that $k_{[\wedge]} \cdot u = k \wedge u \quad \forall k, u \in \mathbb{R}^3$.

5.1 Expression of the MV-Gradients

$$\boxed{\overrightarrow{\bigtriangledown} w_{t_i}^T = \frac{{Jm^T}^t \cdot N_i^T}{\det(A^T)} + \frac{\sum_j w_{t_j}^T N_i^T}{\det(A^T)} \qquad (17)}$$
$$\forall \eta \notin Support(T)$$

with

$$Jm^T = -\sum_j \frac{e_1(\theta_j^T)N_j^T \cdot {J_{jj}}^t}{2(|\hat{p_{t_{j+2}}}||\hat{p_{t_{j+1}}}|)^3} \qquad (18)$$
$$+\sum_j \frac{N_j^T \cdot \hat{m_j}^t}{(|\hat{p_{t_{j+2}}}||\hat{p_{t_{j+1}}}|)^2} + \sum_j \frac{e_2(\theta_j^T)JN_j^T}{2|\hat{p_{t_{j+2}}}||\hat{p_{t_{j+1}}}|}$$

where $e_1(x) = \frac{\cos(x)\sin(x)-x}{\sin(x)^3}$ and $e_2(x) = \frac{x}{\sin(x)}$.

Special case: $\eta \in Support(T), \notin T$

$$\begin{aligned}\overrightarrow{\bigtriangledown} w_{t_i}^T = &-\sum_j \frac{e_2(\theta_j^T)u_{ij}}{4|T||p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|} n_T \\ &-\sum_j \frac{e_1(\theta_j^T)u_{jj}{N_i^T}^t \cdot N_j^T}{8|T|(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^3} n_T \\ &+\sum_j \frac{{N_i^T}^t \cdot N_j^T}{4|T|(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^2} n_T \\ &\forall \eta \in Support(T), \notin T\end{aligned}$$

5.2 Expression of the MV-Hessians

We define $\delta^x = \begin{pmatrix}1\\0\\0\end{pmatrix}$, $\delta^y = \begin{pmatrix}0\\1\\0\end{pmatrix}$, and $\delta^z = \begin{pmatrix}0\\0\\1\end{pmatrix}$.

$$\begin{aligned}Hw_{t_i}^T = &\frac{1}{\det(A^T)} \begin{pmatrix} {N_i^T}^t \cdot \partial_x(Jm^T) \\ {N_i^T}^t \cdot \partial_y(Jm^T) \\ {N_i^T}^t \cdot \partial_z(Jm^T) \end{pmatrix} \\ &+\frac{1}{\det(A^T)}(N_i^T \cdot (\sum_j \overrightarrow{\bigtriangledown} w_{t_j}^T)^t + \sum_j \overrightarrow{\bigtriangledown} w_{t_j}^T \cdot {N_i^T}^t)\end{aligned} \quad (19)$$

with

$$\begin{aligned}\partial_c(Jm^T) = &\sum_j \frac{e_3(\theta_j^T)(J_{jj})_{(c)}N_j^T \cdot {J_{jj}}^t}{2(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^5} \\ &-\sum_j \frac{e_4(\theta_j^T)(\hat{m_j})_{(c)}N_j^T \cdot {J_{jj}}^t}{(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^4} \\ &-\sum_j \frac{e_1(\theta_j^T)(\partial_c(N_j^T) \cdot {N_j^T}^t + N_j^T \cdot \partial_c(N_j^T)^t) \cdot JN_j^T}{2(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^3} \\ &+\sum_j \frac{\partial_c(N_j^T) \cdot (\hat{m_j})^t}{(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^2} - \sum_j \frac{e_5(\theta_j^T)(J_{jj})_{(c)}JN_j^T}{2(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^3} \\ &+\sum_j \frac{e_6(\theta_j^T)(\hat{m_j})_{(c)}JN_j^T}{(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^2} - \sum_j \frac{N_j^T \cdot \delta^{ct}}{(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^2} \\ &-\sum_j \frac{3e_1(\theta_j^T)N_j^T \cdot {J_{jj}}^t}{2(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^3} \left(\frac{(p_{\hat{t_{j+1}}})_{(c)}}{|p_{\hat{t_{j+1}}}|^2} + \frac{(p_{\hat{t_{j+2}}})_{(c)}}{|p_{\hat{t_{j+2}}}|^2} \right) \\ &+\sum_j \frac{2N_j^T \cdot {\hat{m_j}}^t}{(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^2} \left(\frac{(p_{\hat{t_{j+1}}})_{(c)}}{|p_{\hat{t_{j+1}}}|^2} + \frac{(p_{\hat{t_{j+2}}})_{(c)}}{|p_{\hat{t_{j+2}}}|^2} \right) \\ &+\sum_j \frac{e_2(\theta_j^T)JN_j^T}{2|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|} \left(\frac{(p_{\hat{t_{j+1}}})_{(c)}}{|p_{\hat{t_{j+1}}}|^2} + \frac{(p_{\hat{t_{j+2}}})_{(c)}}{|p_{\hat{t_{j+2}}}|^2} \right) \\ &\forall c \in \{x, y, z\}\end{aligned}$$

where (noting $c := cos(x)$ and $s := sin(x)$)
$e_3(x) = (3c(cx - s) + cs^3)/s^5$, $e_4(x) = (3(cx - s) + s^3)/s^3$, $e_5(x) = (s - xc)c/s^3$, and $e_6(x) = (s - xc)/s$.

Special case: $\eta \in Support(T), \notin T$

$$\begin{aligned}Hw_{t_i}^T = &\overrightarrow{\bigtriangledown} dw_i^T \cdot n_T^t + n_T \cdot \overrightarrow{\bigtriangledown} {dw_i^T}^t \\ &\forall \eta \in Support(T), \notin T\end{aligned} \quad (20)$$

with

$$\begin{aligned}-2|T|\overrightarrow{\bigtriangledown} dw_i^T = &-\sum_j \frac{e_1(\theta_j^T)u_{ij}J_{jj}}{2(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^3} \\ &+\sum_j \frac{u_{ij}\hat{m}_j}{(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^2} + \sum_j \frac{e_7(\theta_j^T)u_{jj}(N_i^T \cdot N_j^T)J_{jj}}{4(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^5} \\ &-\sum_j \frac{u_{jj}(N_i^T \cdot N_j^T)\hat{m}_j}{(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^4} - \sum_j \frac{e_1(\theta_j^T)u_{jj}(J_{ji} + J_{ij})}{4(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^3} \\ &-\sum_j \frac{(N_i^T \cdot N_j^T)J_{jj}}{(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^4} - \sum_j \frac{2\cos(\theta_j^T)(N_i^T \cdot N_j^T)\hat{m}_j}{(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^3} \\ &+\sum_j \frac{(J_{ji} + J_{ij})}{2(|p_{\hat{t_{j+2}}}||p_{\hat{t_{j+1}}}|)^2}\end{aligned}$$

where $e_7(x) = \frac{2\cos(x)\sin(x)^3 + 3(\sin(x)\cos(x) - x)}{\sin(x)^5}$.

6 Continuity between the general case and the special case

We obtained the formulae for the gradient and the Hessian of $w_i^T(\eta)$ in the general case, when the point of interest η does not lie on the triangle T, and in the special case when η lies on it.

As MVC are C^∞ everywhere not on M, these formulae are guaranteed to converge, since in particular, the gradient and the Hessian are continuous functions everywhere not on M.

The same holds in 2D where the distinction is made for the computation of $w_i^E(\eta)$ whether η lies on the line supported by the edge E or not.

7 Experimental Analysis

In this section, we present experimental evidence of the numerical accuracy of our derivation and provide computation timings.

7.1 Complexity

For each point η, computing the MVC, the MVC gradients and the MVC Hessians is linear in the number of vertices and edges (faces in 3D) of the cage.

7.2 C++ Implementation

We implemented the computation of the Mean Value Coordinates derivatives in C++, following the layout described in Sec. 4 and 5. **This C++ implementation is provided as additional material [25].** As it only requires simple matrix-vector products, no

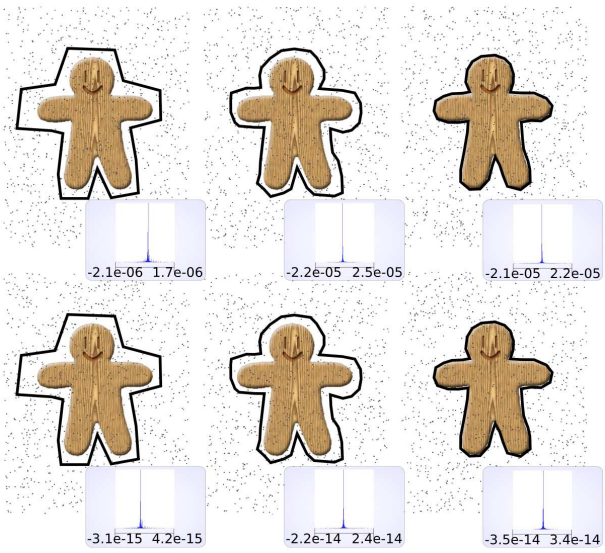


Fig. 3 Validation based on a manufactured solution (group of rigid transformations) for the 2D case. The histograms show the violation of the correctness conditions associated with the manufactured solution (95% most relevant samples). Top: simple floating point precision (output precision: 10^{-5}). Bottom row: double precision (output precision: 10^{-14}). Size of the diagonal of the domain: $\sim$ 1000.

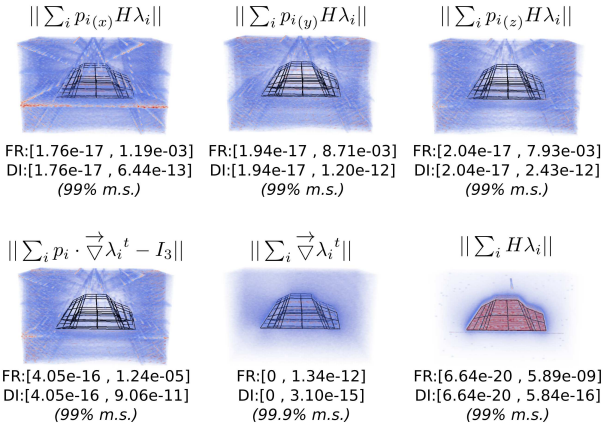


Fig. 4 Validation based on a manufactured solution (group of rigid transformations) for the 3D case. The violation of the correctness conditions (from transparent blue, low values, to opaque red, high value) is measured on each vertex of a 100^3 voxel grid. The full value range (FR) is given below each image while only the 99% most significant samples are displayed (DI). Size of the diagonal of the domain: $\sim$600.

third-party library is needed. To implement the applications discussed later in the paper, matrix Singular Value Decomposition needs to be performed, for instance to project Jacobian matrices onto the space of 2D/3D rotations. We used the GNU Scientific Library for that purpose.

7.3 Global validation with a manufactured solution

We first inspect the numerical accuracy of our derivation using the Method of Manufactured Solution (MMS), a popular technique in code verification [1, 6, 7]. Such a verification approach consists in designing an input configuration such that the resulting solution is known a priori. Then the actual verification procedure aims at assessing that the solution provided by the program conforms to the manufactured solution. In other words, MMS verification consists in designing *exact ground-truths* for accuracy measurement. However, note that this verification is not general, as it only assesses correctness for the set of manufactured solutions.

Manufactured solution: As Mean Value Coordinates provide smooth interpolations, a global rigid transformation of the cage $\overline{p_i} = T + R \cdot p_i$ should infer a global rigid transformation of the entire Euclidean space. In particular, the Jacobians of the embedding function f of the transformed cage ($f : \mathbb{R}^3 \to \mathbb{R}^3$, $f(p_i) = \overline{p_i}$) should be equal to R everywhere, and its Hessian should be exactly 0. Then our manufactured configuration is the space of global rigid transformations and our manufactured solution is defined by $Jf = R$ and $Hf = 0$.

Given this manufactured solution, we can derive correctness conditions for the Jacobian evaluation from the following expression:

$$Jf = \sum_i f(p_i) \cdot \overrightarrow{\nabla}\lambda_i{}^t = R \cdot \sum_i p_i \cdot \overrightarrow{\nabla}\lambda_i{}^t + T \cdot \sum_i \overrightarrow{\nabla}\lambda_i{}^t$$

Thus, to conform to the manufactured solution $Jf = R$, the following equations should be satisfied:

$$\begin{cases} \sum_i \overrightarrow{\nabla}\lambda_i{}^t = (0,0,0) \\ \sum_i p_i \cdot \overrightarrow{\nabla}\lambda_i{}^t = I_3 \end{cases} \tag{21}$$

As for the Hessian evaluation, we can derive similar correctness conditions:

$$\begin{aligned} Hf_c &= \sum_i f_c(p_i) H\lambda_i \\ &= \sum_{\forall d \in \{x,y,z\}} R_{cd} \sum_i p_{i(d)} H\lambda_i + T_c \sum_i H\lambda_i \ \forall c = \{x,y,z\} \end{aligned}$$

where T_x, T_y and T_z are the first, second and third coordinate of the vector T respectively (similarly for R_{cd}).

Thus, to conform to the manufactured solution $Hf = 0$, the following equations should be satisfied:

$$\begin{cases} \sum_i H\lambda_i = 0_3 \\ \sum_i p_{i(x)} H\lambda_i = 0_3 \\ \sum_i p_{i(y)} H\lambda_i = 0_3 \\ \sum_i p_{i(z)} H\lambda_i = 0_3 \end{cases} \tag{22}$$

Note that both the Jacobian and Hessian correctness conditions (Eq. 21 and 22) are not functions of the rigid transformation parameters; they also correspond to the constant and linear precision properties of the MVC.

These properties remain valid for arbitrary translations and rotations and thus cover the entire group of rigid transformations.

Fig. 3 shows numerical evaluations of these correctness conditions for different cages, at random points (in grey) of a 2D domain. In particular, the histograms plot the entries of the left-hand term (a vector or a matrix) of each of these equations, which should all be zero (for the second Jacobian condition, the entries of the matrix $\sum_i p_i \cdot \overrightarrow{\nabla}\lambda_i{}^t - I_2$ are shown). As shown in this experiment, the error induced by the violation of the correctness conditions is close to the actual precision of the data structure employed for real numbers (*float* or *double*). Also, the error slightly increases when the cage is denser. Indeed, with dense cages, it is more likely that the randomly selected samples lie in the vicinity of the support of the cage edges. These configurations correspond to the special cases discussed earlier and for which Taylor expansions are employed.

Fig. 4 shows a similar experiment in 3D, with a coarse cage (black lines). Similarly, most of the errors are located on the tangent planes of the triangles (special cases). Note, that an important part of the error yields from the samples which are located in the vicinity of the cage triangle, a configuration for which we do not provide a closed-form expression, as discussed in Sec. 3. Interestingly, the errors on the correctness conditions for the Jacobian and the Hessian are comparable to the errors induced by the actual computation of the Mean Value Coordinates λ_i. In the example shown in Fig. 4, the error range of the positional reconstruction on the grid (i. e. the violation of the linear precision property of the MVC) is $[0, 1.08\ 10^{-5}]$, which is larger than the error ranges observed in two of the six correctness condition evaluations of the derivatives.

7.4 Taylor approximations behavior

Validation based on manufactured solutions enables assessing the accuracy of a numerical computation on a sub-set of pre-defined configurations. However, in our setting, designing manufactured solutions corresponding to other configurations than rigid transformations is highly involved.

Thus, to extend our analysis to arbitrary configurations, we present in this paragraph an analysis of the Taylor approximations of MVC-functions based on our derivation.

In contrast to manufactured solutions, this analysis is not meant to *validate* our results, but simply analyse how functions expressed by MVC behave in the local neighborhood of a point.

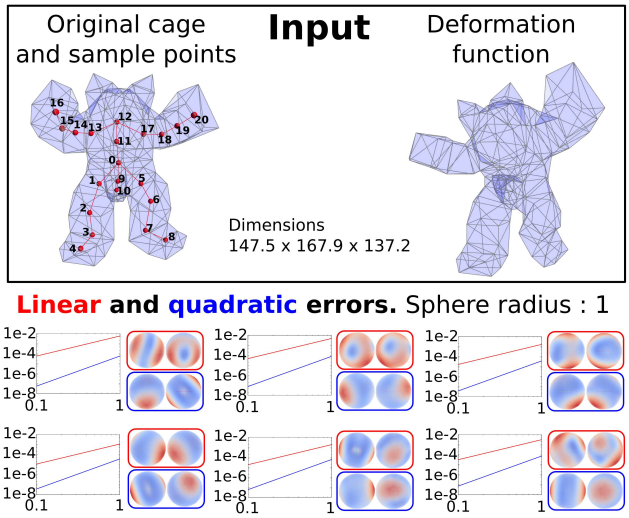


Fig. 5 Red curve: Linear approximation. Blue curve: Quadratic approximation. Plots show **on a logarithmic scale** the radial function defined as the average of the absolute error on the sphere of radius r ($r \in [0, 1]$). The evaluation points η were taken on the skeleton shown in the original cage (here are displayed the evaluations for the points $\#0, \#1, \#2, \#3, \#4, \#5$, but these are representative of the curves we have for all locations). For each plot, the spheres on the top show the linear approximation error from two points of view (red box), the others show the quadratic approximation error from the same two points of view (blue box). From the tangent, we can assess the quadratic convergence of the linear approximation scheme and the cubic convergence of the quadratic approximation scheme.

For regular functions, function values in the neighborhood of a point can be approximated up to several orders of precision, using Taylor approximations:

$$f(\eta + d\eta) = f(\eta) + \overrightarrow{\nabla} f_\eta^t \cdot d\eta + o(||d\eta||)$$

$$f(\eta + d\eta) = f(\eta) + \overrightarrow{\nabla} f_\eta^t \cdot d\eta + \frac{1}{2} d\eta^t \cdot Hf_\eta \cdot d\eta + o(||d\eta||^2)$$

In the following, we use these approximations to analyze the behavior of our derivation for arbitrary configurations. In particular, we evaluate the following errors:

$$E^1 = ||f(\eta + d\eta) - f(\eta) - \overrightarrow{\nabla} f_\eta^t \cdot d\eta||$$

$$E^2 = ||f(\eta + d\eta) - f(\eta) - \overrightarrow{\nabla} f_\eta^t \cdot d\eta - \frac{1}{2} d\eta^t \cdot Hf_\eta \cdot d\eta||$$

As the evaluation neighborhood shrinks to a point, these errors should tend to zero, with a horizontal tangent. Fig. 5 shows plots of these errors (logarithmic scale) on an arbitrary deformation function defined by user interactions:

– On regular functions, the derivatives of the MVC characterize the interpolated function correctly: the maximum error is 7. 10^{-3} (for a bounding diagonal of 316). Note, that the radius $r \in [0, 1]$ of the evaluation neighborhood where they can be used to

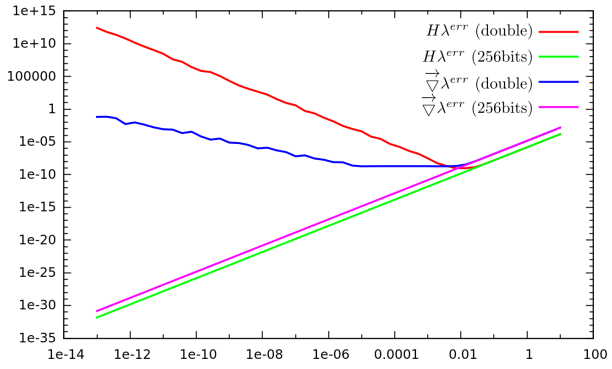


Fig. 6 Comparison with Finite Differences: the domain are the same as described previously in Fig. 5, and the evaluations are performed on point 0. x axis: size of the stencil for Finite Differences. y axis: difference between Finite Differences approximations of the derivatives and our formulae. Axes of the plots are in logarithmic scale. The functions that are plotted are $\overrightarrow{\nabla}\lambda^{err}(r) = \sqrt{\sum_i ||\overrightarrow{\nabla}\lambda_i - \overrightarrow{\nabla}{\lambda_i}^{FD(r)}||^2}$ and $H\lambda^{err}(r) = \sqrt{\sum_i ||H\lambda_i - H{\lambda_i}^{FD(r)}||^2}$.

approximate the function is not too small. Therefore, our derivative formulation provides enough accuracy to enforce *sparse* derivative constraints on local neighborhoods such as those expressed in the applications described in Sec. 8.

– The linear approximation can sometimes produce more accurate approximations in average than the quadratic approximation on a large neighborhood, while the quadratic approximation provides results which are less direction-dependant.
– The induced errors indeed tend to zero when the neighborhood shrinks to a point and the tangent of the curves in logarithmic scale illustrates that the error conforms to the expected form ($O(||d\eta||^2)$ for the linear approximation,$O(||d\eta||^3)$ for the quadratic approximation). Indeed, remember, that if $y = \lambda \cdot x^n$, then $\log(y) = \log(\lambda) + n \cdot \log(x)$.

7.5 Comparison with Finite Difference schemes

In this section we use Finite Differences schemes to derive the gradient and the Hessian of the MVC, to compare with the expressions we obtained.

A conventional scheme for approximations of first and second order derivatives at point (x, y, z) is the following:

$f_x \simeq \frac{f(x+h,y,z)-f(x-h,y,z)}{2h}$
$f_{xx} \simeq \frac{f(x+h,y,z)-2f(x,y,z)+f(x-h,y,z)}{h^2}$
$f_{xy} \simeq \frac{f(x+h,y+h,z)-f(x+h,y-h,z)-f(x-h,y+h,z)+f(x-h,y-h,z)}{4h^2}$
...

This scheme requires 19 evaluations of the function in total. Results of convergence of Finite Differences (FD) of the Mean Value Coordinates derivatives using this scheme are presented on an example in Fig. 6, using double precision and 256 bits precision (using **mpfrc++**, which is a c++ wrapper of the *GNU multiple precision floating point library (mpfr)*). The domain is the same as described previously in Fig. 5, and the plots correspond here to the evaluation made in *point* 0. The error functions that are plotted are $\overrightarrow{\nabla}\lambda^{err}(r) = \sqrt{\sum_i ||\overrightarrow{\nabla}\lambda_i - \overrightarrow{\nabla}{\lambda_i}^{FD(r)}||^2}$ and $H\lambda^{err}(r) = \sqrt{\sum_i ||H\lambda_i - H{\lambda_i}^{FD(r)}||^2}$. Note, that these plots are representative of all the experiments we made (i. e. with other cages, at other locations, etc.).

These results validate empirically our formulae, as the Finite Differences scheme converges to our formulae when the size of the stencil tends to 0 (Fig. 6, using 256 bits precision). It also indicates that Finite Differences schemes are not suited to evaluate MVC derivatives in real life applications (see Fig. 11 for example), as these schemes diverge near 0 when using double precision only (Fig. 6 blue and red curves). Note, that this behavior is not typical of Mean Value Coordinates, but rather of finite differences schemes. The choice of the size of the stencil is a typical difficulty in finite difference schemes. Choosing a size which is too small may introduce large rounding errors [8, 24]. Finding the smallest size which minimizes rounding error is both machine dependent and application dependent (in our case $\simeq 0.01$ on the example of Fig. 6). Moreover, its has been shown that all finite difference formulae are ill-conditioned [13] and suffer from this drawback. We used different schemes to approximate the derivatives using Finite Differences methods (9 points evaluation + linear system inversion, 19 points evaluation on a $3\times3\times3$-stencil, tricubic interpolation on a $4\times4\times4$-stencil), and that they all diverge in the same manner when using double precision.

The error curves are also similar when looking at the deviation of the gradients and Hessians of the **function itself** that is interpolated (e. g. the deformation function in Fig. 5), instead of the gradients and Hessians of the **weights** themselves.

7.6 Comparison with Automatic Differentiation

Along with Finite Differences, Automatic Differentiation (AD) is also a popular class of techniques for the numerical evaluation of derivatives of functions expressed by a computer program. In this sub-section, we compare for the 3D case our formulae to an evaluation provided by an AD software, the C++ library ADOL-C. As expected, in practice, the computation of

Input Cage Model (#V / #T)	Coord. Only **(ms)**	Coord. + Deriv. [Analytic] **(ms)**	Coord. + Deriv. [FD] **(ms)**	Coord. + Deriv. [TriC] **(ms)**
Beetle (32 / 60)	0.060	0.781	1.157	4.196
Beetle (130 / 256)	0.256	3.261	4.881	17.625
Beetle (514 / 1024)	1.027	13.171	19.620	70.768
Armadillo (164 / 324)	0.324	4.130	6.174	22.428
Monster (128 / 252)	0.252	3.212	4.809	17.503
Monster (506 / 1008)	1.013	12.860	19.389	69.900

Table 1 Performance of the computation of the 3D Mean Value Coordinates and their gradients and Hessians at a single point. Tests were performed on 1000 points and average timings are presented. For the sake of completeness, we present timings of classical Finite Differences (FD) methods that require 19 evaluations in total, and tricubic approximations (TriC) that require 65 evaluations in total.

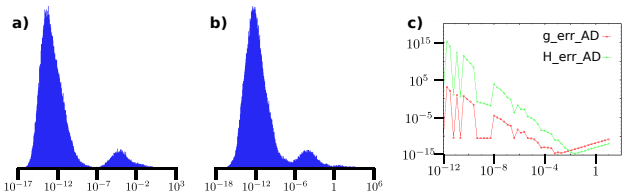


Fig. 7 Comparison with Automatic Differentiation tools. a) Histogram of errors (x axis is logarithmic) of the computation of the gradients of the weights by ADOL-C (on 1000 points randomly distributed in the model's bounding box). **b)** Histogram of errors (x axis is logarithmic) of the computation of the Hessians of the weights by ADOL-C on the same set. **c)** Error of the computation of the derivatives by ADOL-C on the special case of the points lying on the support of the cage's triangles (x axis represents the distance to the support plane).

the Jacobian and Hessian with AD is slower than the evaluation with our formulae (20 times slower in average). A more problematic drawback of AD is its numerical stability, especially in regions nearby the support of the cage triangles, where the MVC are forced to zero (they are only defined by continuity and Eq. 5 is not defined in these positions). To the best of our knowledge, this subtlety cannot be captured efficiently by AD tools. Fig. 7 (a,b) shows histograms of errors (ie. absolute difference between our formulae and the AD evaluation, $|\overrightarrow{\nabla}\lambda_i(\eta) - \overrightarrow{\nabla}_{AD}\lambda_i(\eta)|$ and $|H\lambda_i(\eta) - H_{AD}\lambda_i(\eta)|$), obtained on a set of 1000 points randomly distributed in the bounding box of the cage model (the cage is the Armadillo cage of Fig. 5). Fig. 7 (c) shows the convergence of the values computed by ADOL-C in the vicinity of the support of the cage triangles (i. e. $|\overrightarrow{\nabla} w_{t_i}^T(\eta) - \overrightarrow{\nabla}_{AD} w_{t_i}^T(\eta + \epsilon n_T)|$ and $|Hw_{t_i}^T(\eta) - H_{AD} w_{t_i}^T(\eta + \epsilon n_T)|$). This plot shows that, as the evaluation point gets closer to the support plane, the AD evaluation diverges. Moreover, in practice, when it lies exactly on the support plane, the value returned by ADOL-C is undefined (*NaN*, Not a Number).

7.7 Timings

Table 1 shows average computation times of the evaluation of the Mean Value Coordinates and their derivatives for several input cages. As the cost of the evaluation depends on the occurence of the special cases (point lying on the support plane of the triangles of the cage), we performed the computation on a set of 1000 points that were randomly distributed inside the bounding box of the model, and the average time is presented. As shown in this table, these computations take only a few milliseconds, which allows their usage in interactive contexts. Also, note that in the applications discussed in the following section, the derivatives are only evaluated on a very small set of points for constraint enforcement.

8 Applications

In this section, we review the applications presented in the original paper [17], and illustrate the utility of our contribution for all of them.

8.1 MVC derivatives visualization

In the context of function design/editing/visualization, the derivatives of the function can be of use to the user, as they have very often an intuitive meaning.

Fig. 8 Visualization of rotations on the shape skeleton.

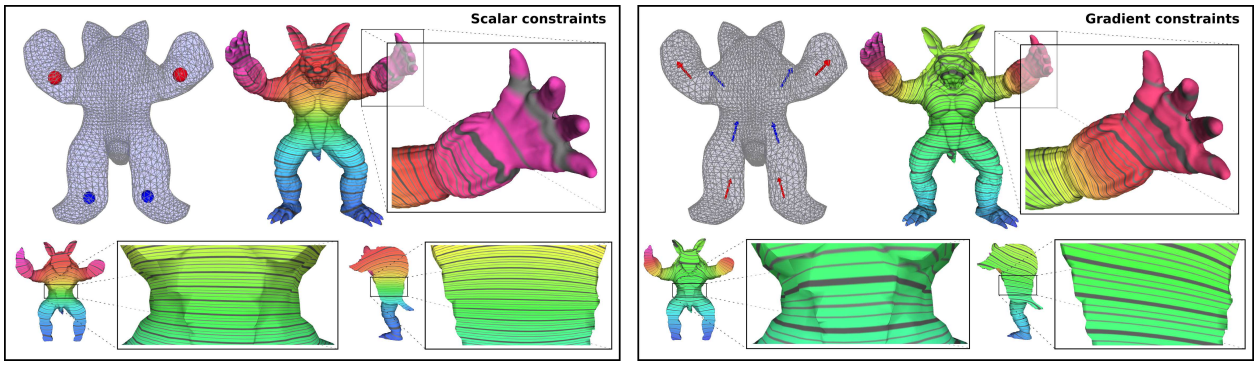


Fig. 9 Flexible volumetric scalar field design with MVC gradient constraints. Left: Scalar constraints (spheres in the cage). Right: Gradient constraints (arrows in the cage). Blue and red colors respectively correspond to low and high value/gradient. In contrast to simple scalar constraints (left), gradient constraints (right) enable to intuitively interact with the shape and the velocity of the level lines.

For example, in the context of 2D or 3D deformation, the Jacobian of the transformation $J(\eta)$ can be put in the form $J(\eta) = R(\eta) \cdot B(\eta) \cdot \Sigma(\eta) \cdot B(\eta)^t$ using Singular Value Decomposition. These different matrices represent the scales of the transformation (Σ) in the basis given by B, and the rotation that is applied afterwards (R) – which can be represented easily as a vector and an angle (see Fig. 8). In the context of color interpolation, the gradients of the different channels can be displayed. In general, the norm of the Hessian provides the information of how locally rigid the function is around the point of interest. Using our formulation, one can obtain these informations at any scale with the same precision, to the contrary of what *finite differences schemes* would provide.

8.2 Flexible volumetric scalar field design

As shown in [17], Mean Value Coordinates can be used to solve the boundary value interpolation problem for the definition of volumetric scalar fields, given an input field prescribed on a closed surface. Our derivation of the MVC gradients and Hessians enables to extend this application to more flexible volumetric scalar field designs, in particular by enforcing the gradient of the interpolated function. Such flexible scalar fields contribute to volumetric texturing [17] and meshing [23]. Fig. 9 illustrates this application where the user sketched gradient constraints inside the volume. To compute a function which satisfies these constraints, a linear system is solved, where the unknowns are the scalar field values on the cage. In particular the following energy is minimized:

$$E = \sum_{v_i \in \overline{V}} w_i^P || \sum_j \lambda_j(v_i) f_j - \overline{f_i}||^2$$
$$+ \sum_{v_i \in M} || \triangle f_i||^2 + \sum_{v_i \in \overline{G}} w_i^G || \sum_j \overrightarrow{\nabla} \lambda_j(v_i) \cdot f_j - \overline{g_i}||^2$$

where $\overline{V}$ is a set of points where hard constraints are applied on function values, $\triangle f_i$ denotes the cotangent Laplacian of the function at the vertex c_i of the cage, and $\overline{G}$ is the set of points where the gradient constraints are specified. Such an optimization procedure generates a smooth function on the cage (by minimizing its Laplacian) as well as in the interior volume (thanks to the MVC) with enforced gradient constraints. As shown in Fig. 9, the gradient constraints enable interacting with the shape and the *velocity* of the level sets of the designed function.

8.3 Implicit Cage Manipulation with Variational MVC

As shown in [2], intuitive, low distortion, volumetric deformations can be obtained through a variational framework. In this context, the space of allowed transformations is explicitly described and the cage deformation is automatically optimized to satisfy positional constraints, while respecting the allowed transformations.

Since our derivation enables to express the Jacobian and Hessian of the transformation at any point in space as a linear combination of the cage vertices, the user can specify rigidity constraints (by minimizing the norm of the Hessian) or rotational constraints (by setting the Jacobian to the corresponding value).

The solution to this optimization problem is a 3D field $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, defined everywhere in space using MVC, which interpolates the transformation of the cage vertices.

Let $\overline{P}$ be the set of positional constraints of the transformation ($\forall v_i \in \overline{P}, f(v_i) = \overline{v_i}$), $\overline{J}$ the set of Jacobian constraints ($\forall v_i \in \overline{J}, Jf(v_i) = \overline{J_i}$), and $\overline{H}$ the set of Hessian constraints ($\forall v_i \in \overline{H}, Hf_x(v_i) = Hf_y(v_i) = Hf_z(v_i) = 0_3$). The solution is given by minimizing the

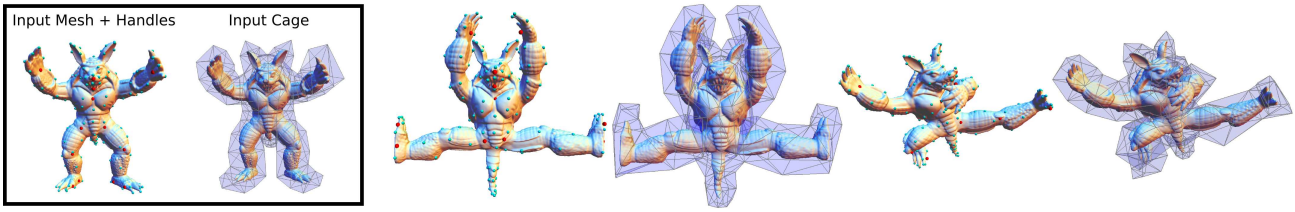


Fig. 10 Implicit Cage Manipulation with *Variational MVC*. Red points indicate positional constraints; blue points indicate unknown rotational constraints. The quality of the produced cages allows the user to edit small details manually by switching from an *implicit* manipulation to an *explicit* manipulation. The same cannot be done with a Green Coordinates solver.

following energy:

$$
\begin{aligned}
E = & \sum_{v_i \in \overline{P}} (w_i^P || \sum_j \lambda_j(v_i) c_j - \overline{v_i}||^2) \\
+ & \sum_{v_i \in \overline{J}} (w_i^J || \sum_j c_j \cdot \overrightarrow{\nabla} {\lambda_j}^t(v_i) - \overline{J_i}||^2) \\
+ & \sum_{v_i \in \overline{H}} (w_i^H || \sum_j H\lambda_j(v_i) \cdot c_{j\,(x)}||^2) \\
+ & \sum_{v_i \in \overline{H}} (w_i^H || \sum_j H\lambda_j(v_i) \cdot c_{j\,(y)}||^2) \\
+ & \sum_{v_i \in \overline{H}} (w_i^H || \sum_j H\lambda_j(v_i) \cdot c_{j\,(z)}||^2)
\end{aligned}
$$

where w^P, w^J, and w^H are weights for the positional, Jacobian, and Hessian constraints respectively. Similarly to [2], the transformation can be constrained locally to be a pure rotation. Then, in prescribed locations, the following property should hold:

$$Jf(v_i)^t \cdot Jf(v_i) = I_3 \quad \forall v_i \in \overline{J}$$

Note, that the actual values of these Jacobian constraints are now unknowns which can be obtained through an iterative optimization, as described in [2]. Due to the non-local nature of Mean Value Coordinates (in comparison to Green Coordinates), we constrain pure rotations to a subset of the enclosed surface vertices (blue spheres in Fig. 1 and 10) instead of constraining them to the medial axis.

Fig. 1 and 10 illustrate the algorithm, where the input surface is shown on the left in its enclosing cage. In these examples, rotational constraints have been distributed evenly on the surface (blue points) and only 14 positional constraints have been specified and edited by the user. As showcased in the accompanying video, the interactions required by our system are limited and intuitive and our resolution of this optimization process is fast enough to provide interactive feedback despite a CPU-only implementation.

Fig. 11 shows a comparison with the results one can obtain using a Finite Differences scheme in the context of shape deformation. Using Finite Differences requires to tune the size of the stencil used for computation of the MVC derivatives case by case, and it can be difficult to set it up correctly, resulting in poor reconstructions.

Discussion

Other techniques have been proposed in the past for as-rigid-as-possible (ARAP) cage-driven shape deformations. For instance, Borosán et al. [3] presented a technique which solves for ARAP transformations on the cage, while interpolating the results in the interior with MVC. However, as discussed earlier, MVC coordinates exhibit a very global behavior. Thus, ARAP transformations on the cage do not necessarily imply ARAP transformations in the interior. Reciprocally, it is often necessary to generate non-ARAP transformations of the cage in order to yield ARAP transformations in the interior. Instead, our technique enforces ARAP constraints directly on the enclosed shape.

Variational Harmonic Maps (VHM) [2] use Green Coordinates as the underlying machinery for deforming shapes in an implicit fashion. Note however, that the triangle normals are unknowns of the system in VHM, they can therefore take values arbitrarily far from the actual normals dictated by the normalization of the Euclidean cross product of face edges. Hence, the cages generated by this technique cannot be exploited in a consistent manner for post-processing tasks. For instance, loading these cages in a modeling software supporting Green Coordinates would fail to correctly reconstruct the enclosed shape if the traditional Euclidean normals were used. Even if the normal solutions provided by the VHM system were used for this

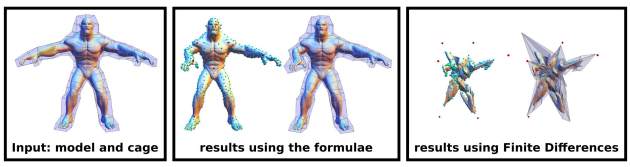


Fig. 11 Comparison with Finite Differences schemes (step: 10^{-3}) in the context of Variational MVC Deformations. The parameters for the linear system are strictly the same.

initial reconstruction, it would be not clear how to update them consistently given some explicit user deformation of the cage. The same remark goes for other post-processing tasks, such as cage-driven shape interpolation, for animation generation based on key-frames provided by implicit cage manipulation.

On the contrary, our technique (*variational MVC*) does not suffer from this drawback as only cage vertex positions are unknowns. Thus, the cages produced by our algorithm can be manipulated and re-used directly and consistently with existing software supporting MVC based deformations in various post-processing tasks. Also, as demonstrated in the accompanying video, our technique allows the user to switch at any time from *implicit* to *explicit* cage manipulation for small detail tuning, which cannot be done with a solver based on Green Coordinates. Note however, that it is not clear how to go from an *explicit* manipulation to an *implicit* manipulation, as the constraints enforced by our system are not respected when moving each cage's vertex independently from the others. Thus, *implicit* manipulation of the cage using our system can only be done before an *explicit* manipulation, or this editing phase will be discarded by the system.

9 Conclusion and Future Work

In this paper we have presented the closed form expressions of the derivatives of Mean Value Coordinates for piecewise linear cages, both in 2D and 3D. To our knowledge, this is the first work that provides derivatives of *interpolant* barycentric coordinates, which can be used for interpolation of arbitrary functions prescribed on cage vertices. Similar formulae have been proposed for Green Coordinates already, but they are limited to the context of shape deformation, and they are not an interpolant. A full numerical analysis of the derivation has been carried out and both its accuracy and reliability has been demonstrated experimentally. Furthermore, applications involving optimization problems benefiting from MVC derivative constraint enforcement have been presented and the utility of our contribution has been demonstrated.

In future work, we would like to investigate the possibility of expressing derivatives for Positive Mean Value Coordinates (PMVC) [19]. The possible negativity of MVC coordinates has often been discussed as a drawback in certain contexts. Note however, that this particular property makes them the only barycentric coordinates which allow the definition of coordinates outside of the cage in a straightforward manner. Nevertheless, PMVC can overcome this possible drawback, by only taking into consideration the cage vertices which are *visible* from the point under evaluation, which can be done very efficiently on the GPU, using the rasterization hardware machinery. However, these coordinates are not smooth since the visibility function is not smooth either. It would be interesting to study in practice how reliable the MVC derivatives can be when restricted to visibility dependant sub-cages, in order to mimic the behavior observed with PMVC.

References

1. Babuska I, Oden J (2004) Verification and validation in computational engineering and science: basic concepts. Computer Methods in Applied Mechanics and Engineering pp 4057–4066
2. Ben-Chen M, Weber O, Gotsman C (2009) Variational harmonic maps for space deformation. ACM Trans Graph (ACM SIGGRAPH) pp 1–11
3. Borosán P, Howard R, Zhang S, Nealen A (2010) Hybrid mesh editing. In: Proc. of Eurographics
4. C L, T D (1989) A mulisided generalization of Bézier surfaces. ACM Trans Graph 8:204–234
5. E W (1975) A rational finite element basis. Academic Press, New York
6. Etiene T, Scheiddeger C, Nonato L, Kirby R, Silva C (2009) Verifiable visualization for isosurface extraction. IEEE Trans on Vis and Comp Graph (IEEE VIS) 15:1227–1234
7. Etiene T, Nonato L, Scheiddeger C, Tierny J, Peters TJ, Pascucci V, , Kirby R, Silva C (2011) Topology verification for isosurface extraction. IEEE Trans on Vis and Comp Graph
8. Flannery BP, Press WH, Teukolsky SA, Vetterling W (1992) Numerical recipes in c. Press Syndicate of the University of Cambridge, New York
9. Floater M (1997) Parameterization and smooth approximation of surface triangulations. Comp Aided Geom Design 14:231–250
10. Floater M (1998) Parametric tilings and scattered data approximation. International Journal of Shape Modeling 4:165–182
11. Floater M (2003) Mean value coordinates. Comp Aided Geom Design 20:19–27
12. Floater MS, Kos G, Reimers M (2005) Mean value coordinates in 3D. Comp Aided Geom Design 22:623–631
13. Fornberg B (1981) Numerical differentiation of analytic functions. ACM Transactions on Mathematical Software (TOMS) 7(4):512–526
14. Hormann K, Floater M (2006) Mean value coordinates for arbitrary planar polygons. ACM Trans Graph 25:1424–1441

15. Huang J, Shi X, Liu X, Zhou K, Wei LY, Teng SH, Bao H, Guo B, Shum HY (2006) Subspace gradient domain mesh deformation. ACM Trans Graph (ACM SIGGRAPH) 25:1126–1134
16. Joshi P, Meyer M, DeRose T, Green B, Sanocki T (2007) Harmonic coordinates for character articulation. ACM Trans Graph (ACM SIGGRAPH) 26
17. Ju T, Schaefer S, Warren J (2005) Mean value coordinates for closed triangular meshes. ACM Trans Graph (ACM SIGGRAPH) 24(3):561–566
18. Langer T, Belyaev A, Seidel HP (2006) Spherical barycentric coordinates. In: Proc. of Symposium on Geometry Processing, pp 81–88
19. Lipman Y, Kopf J, Cohen-Or D, Levin D (2007) Gpu-assisted positive mean value coordinates for mesh deformation. In: Symposium on Geometry Processing, pp 117–123
20. Lipman Y, Levin D, Cohen-Or D (2008) Green coordinates. ACM Trans Graph (ACM SIGGRAPH) 27(3):1–10
21. Malsch E, Dasgupta G (2003) Algebraic construction of smooth interpolants on polygonal domains. In: Proc. of International Mathematica Symposium
22. Meyer M, Lee H, Barr A, Desbrun M (2002) Generalized barycentric coordinates for irregular polygons. Graphics Tools 7:13–22
23. Nieser M, Reitebuch U, Polthier K (2011) CubeCover - Parameterization of 3D volumes. Comp Graph Forum (SGP) 30:1397–1406
24. Squire W, Trapp G (1998) Using complex variables to estimate derivatives of real functions. Siam Review 40(1):110–112
25. Thiery JM (2013) MVC derivatives C++ implementation. URL http://sourceforge.net/projects/meanvaluecoordinatesderivs/files/latest/download?source=files
26. Urago M (2000) Analytical integrals of fundamental solution of three-dimensional laplace equation and their gradients. Trans of the Japan Soc of Mech Eng 66:254–261
27. Warren J (1996) Barycentric coordinates for convex polytopes. Advances in Computational Mathematics 6:97–108
28. Warren J, Schaefer S, Hirani A, Desbrun M (2007) Barycentric coordinates for convex sets. Advances in Computational Mathematics 27:319–338