This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Efficient 3D Geometric and Zernike moments computation from unstructured surface meshes

J. M. Pozo, M. C. Villa-Uriol, A. F. Frangi, *Senior Member, IEEE*

Abstract—This paper introduces and evaluates a fast exact algorithm and a series of faster approximate algorithms for the computation of 3D geometric moments from an unstructured surface mesh of triangles. Being based on the object surface reduces the computational complexity of these algorithms with respect to volumetric grid-based algorithms. In contrast, it can only be applied for the computation of geometric moments of homogeneous objects. This advantage and restriction is shared with other proposed algorithms based on the object boundary. The proposed exact algorithm reduces the computational complexity for computing geometric moments up to order N, with respect to previously proposed exact algorithms, from N^9 to N^6. The approximate series algorithm appears as a power series on the rate between triangle size and object size, which can be truncated at any desired degree. The higher the number and quality of the triangles, the better the approximation. This approximate algorithm reduces the computational complexity to N^3.
In addition, the paper introduces a fast algorithm for the computation of 3D Zernike moments from the computed geometric moments, with a computational complexity N^4, while the previously proposed algorithm is of order N^6.
The error introduced by the proposed approximate algorithms is evaluated in different shapes and the cost-benefit ratio in terms of error and computational time is analyzed for different moment orders.

Index Terms—Image analysis, geometric moments, 3D Zernike moments, shape characterization, object characterization.

1 Introduction

A STANDARD tool used in computer vision and image analysis for very general purposes, like classification, search and recognition of objects or features, is the computation of moments from an image [1], [2]. Classically, this technique has been applied to 2D images. But with the current broad availability of 3D images and models, these tools have been rapidly extended to 3D. One of the most basic moments are *geometric moments*, which are trivially extendible to any dimension. These moments are simple to compute and are often used as a basis for the computation of other, more elaborate, sets of moments.

Many different algorithms have been proposed for the computation of geometric moments, most of them are specific to 2D, but still there are several valid for 3D or even general to any dimension. In computational imaging, the most used representation of the target object is a volumetric image grid. But a convenient alternative for homogeneous objects is a surface mesh defining its boundary. For general inhomogeneous objects, the computational complexity is L^3 for a grid with L voxels per dimension. For homogeneous objects, the use of the boundary allows reducing the computational complexity to L^2.

In this direction, Yang *et al.*[3] presented a fast algorithm based on a discrete version of the divergence (Gauss's) theorem, applicable to voxel-like object boundaries in a binary grid. However, a characteristic property of this method, common to most of the grid-based methods, is that the volume integral defining the moments is approximated by a summation over the voxels, where each voxel contributes only as a point-like object located at its center. This approximation introduces a discretization error, which escalates as the order of the moments increases [4], [5], and makes the scale and rotation invariants obtained from the moments only approximately invariant [6]. Moreover, in the case of moments with non-square domains, like Zernike moments, the discretization of the domain definition also limits their precision [5].

A more convenient representation of 3D object boundaries is a polygonal mesh. An exact formula of moments from a general 3D polyhedron with arbitrary shape and topology was first obtained by Lien and Kajiya [7], where the integral in the polyhedron is decomposed into integrals in the *oriented* tetrahedra defined by the origin and each triangle of the surface. Cattani and Paoluzzi [8] reformulated it and extended the idea to the boundary integration for a thin polyhedral surface object. Bernardini [9] extended it to general dimensional polyhedra. Sheynin and Tuzikov [10], [11] presented a

- *This work was partially supported by Philips Healthcare (Best, The Netherlands), the @neurIST Integrated Project (co-financed by the European Commission through the contract no. IST-027703), the CDTI CENIT-CDTEAM grant funded by the Spanish Ministry of Industry and the research project STIMATH (TIN2009-14536-C02-01) funded by the Spanish Ministry of Science and Innovation (MICINN) and the European Regional Development Fund (ERDF).*
- *J. M. Pozo, M. C. Villa-Uriol and A. F. Frangi are with the Research Group for Computational Imaging Simulation Technologies in Biomedicine (CISTIB), Information & Communication Technologies Department, Universitat Pompeu Fabra, c/ Tanger 122–140, E08018, Barcelona, Spain, and with the Networking Research Center on Bioengineering, Biomaterials and Nanomedicine (CIBER-BBN).*
- *A. F. Frangi is also with the Institució Catalana de Recerca i Estudis Avançats (ICREA), c/ Tanger 122–140, E08018, Barcelona, Spain.*

Digital Object Indentifier 10.1109/TPAMI.2010.139

similar algorithm also extended to general dimensions. However, the exact formulas for the integration on a single tetrahedron, on which all these works are based, can be traced back to [12]. All of them involve an affine transformation of each tetrahedron into a standard tetrahedron, becoming the most expensive operation in their computation. This transformation was formalized using matrix products by DiCarlo and Paoluzzi [13] for up to second order moments. An alternative approach is based on the application of the divergence theorem, already suggested in [14] and linked to the oriented tetrahedron integration in [7]. Several algorithms have been proposed following this approach [15], [16], [17], [18], [19]. Most of these works develop symbolic formulas for any order but specify them only for low orders (up to 2nd or 3rd). Only a few introduce explicit general algorithms. Since they are based on a boundary representation, their computational complexity is linear in the number of facets. With respect to the order N up to which the moments are computed, all the proposed exact algorithms have computational complexity N^9.

Some approximate integration formulas using quadratures have been also proposed for the moments of a single tetrahedron [20], [21], [22], [23], [24]. However an evaluation of their accuracy for high orders in general polyhedra is not available. In addition, moment computation from smoother boundary representations have been also proposed as the scheme using polynomial patches described by Gonzalez-Ochoa *et al.*[25].

2D *Zernike moments* are classically used because they have shown robustness against image noise and good discriminatory power for object detection and recognition [26], [27]. In contrast to geometric moments, they constitute an orthonormal basis, enabling the easy reconstruction of the original object from its Zernike moments. Since they form an infinite series, the reconstruction obtained from Zernike moments up to order N provides an approximation of the original object. The higher the order, the more accurate the reconstruction. An important advantage of Zernike moments is that their domain of definition is the unit circle and their natural definition appears in terms of polar coordinates in the complex plane. This facilitates obtaining rotational invariants, which are essential for the characterization and recognition of objects independently of their pose. Non-circular objects can be fit within this circular domain by centering, scaling and zero-padding.

Motivated by the good properties of Zernike moments in 2D, Canterakis [28], [29] generalized them to 3D. The criterion was the substitution of the complex exponential in polar coordinates by the spherical harmonics in spherical coordinates. Then, the series of radial factors were fixed by imposing the condition of orthonormality to the moments, and a predetermined increasing order to their polynomial degree. This generalization, especially the computation of the radial factors, is far from straightforward. This 3D version of the Zernike moments and their rotational invariants have been tested for their retrieval capabilities of different classes of 3D objects [30], [31] and have been applied, for instance, to the morphological characterization of cerebral aneurysms [19]. Although in these papers the name used has been *3D Zernike moments*, the authors of the present paper believe that it would be more accurate to call them *Zernike–Canterakis moments* (*Z-C moments* for short)

In [30] an algorithm was introduced for the computation of Zernike–Canterakis moments from previously computed geometric moments. A formula for the coefficients relating each one of the Z-C moments as a linear combination of geometric moments is given. These coefficients can be computed off-line since they are independent of the object. However, their storage requires a large amount of memory for high orders. The computational complexity of the object-dependent part of the algorithm is N^6, with N being the order up to which the moments are computed.

In this paper, we propose two new algorithms for the computation of geometric moments of homogeneous objects from their boundary when represented as a triangular mesh. The starting point of both algorithms follows the same approach as in [7], [10], [11] obtaining the exact formula for moments of tetrahedra. As in their case, our algorithms are valid for objects with arbitrary shape and topology. However, in all three cases, the general formula was presented without focusing on reducing its high computational complexity with respect to the moments order N. Their interest was in low order moments and only presented explicit formulas up to order $N = 3$. Here we obtain more efficient general formulas, which greatly reduce the computational complexity. In addition, following also [11], with a slight modification, the algorithms can be adapted for the computation of the geometric moments of a surface as an object in itself, which we call *surface-like* moments, in contrast to *volume-like* moments.

The first presented algorithm (Section 3) is exact, in the sense that, given an object defined *exactly* by a particular triangle mesh, the algorithm provides the exact geometric moments of this object. However, in general the object of interest is never the triangle mesh itself, but this mesh models the desired object with some precision or resolution, which depends on the number of triangles and on how the mesh was generated and processed, as well as its provenance. This algorithm reduces the computational complexity from N^9 in [11] to N^6.

The second algorithm (Section 4) allows one to compute an approximation of geometric moments. Indeed, the algorithm provides a series of approximations which can be truncated at any desired degree. It is a power series in a parameter, λ, dependent on the ratio of triangle size to object size. Thus, the larger the number of triangles, the smaller the error introduced by the approximation. If the series is computed up to power λ^N, with N being the maximum desired order of the geometric moments, then the algorithm becomes exact. The advantage of the approximate algorithm is that the

computational complexity is notably reduced from N^6 in the exact algorithm to N^3.

If the error introduced by the chosen approximation is smaller than the precision of the mesh in modeling the object, then this error can be ignored and the use of the approximate algorithm will be practically indistinguishable from the exact algorithm. In Section 6 some experiments and analytical computations estimating and comparing the different errors are presented, together with a cost-benefit analysis of the computational time.

In addition, the paper introduces (Section 5) an efficient algorithm for the computation of Z-C moments from geometric moments, decreasing the computational complexity of the previous algorithm presented in [30] from N^6 to N^4. This reduces the computational time of Z-C moments from geometric moments to the order of milliseconds, making it negligible with respect to the computation of geometric moments.

2 Moments from surface meshes

An object in computational imaging can be represented by a scalar field, $f(\boldsymbol{p})$, defined in a finite region, where the intensity value corresponds to some property of the represented object. This can be applied to both 2D and 3D objects, but we will focus on 3D. Given a collection of functions $M_I(\boldsymbol{p})$, together with their proper domain D, the moments of the scalar field with respect to this collection are defined by

$$\mathcal{M}(f)_I \equiv \int_D f(\boldsymbol{p})\, M_I(\boldsymbol{p})\, \mathrm{d}V \tag{1}$$

where $\mathrm{d}V$ is the volume element. One of the most well known moments are *geometric moments*, whose defining collection of functions is the set of monomials of any order on the Cartesian coordinates:

$$G_{ijk}(\boldsymbol{p}) = x^i y^j z^k$$

and their domain is, in principle, the whole of $\mathbb{R}^3$. Observe that the index I is split into 3 integer indices $i, j, k \geq 0$. The other moments considered in this work are the 3D *Zernike–Canterakis moments*. Their defining collection of functions are the *Zernike–Canterakis polynomials* [29]:

$$Z_{n\ell}^m(\boldsymbol{p}) = R_{n\ell}(r) Y_\ell^m(\theta, \varphi)$$

where $\{r, \theta, \varphi\}$ are the spherical coordinates and Y_ℓ^m are the spherical harmonics. Written in Cartesian coordinates they are polynomials of order n. Note that since Z-C polynomials are derived from spherical harmonics, they are complex. The range of the indices are $0 \leq \ell \leq n$, with $n - \ell$ even, and $-\ell \leq m \leq \ell$.

As opposed to geometric moments, Z-C moments constitute a complete collection of orthonormal functions. This property allows a direct reconstruction of the original scalar field from the Z-C moments. In addition, the natural domain of Z-C moments is the interior of the unit sphere. This domain is by construction invariant to rotations, enabling the natural calculation of rotational invariants from the moments.

The simplest model to represent an object without texture or substructures is a binary field with value 1 inside the object and 0 outside. However, the actual realization of this field does not need to be a volumetric image. A homogeneous volumetric object is completely determined by its boundary. Thus a convenient representation is a surface. Typically, this boundary representation is less expensive in terms of memory than the image representation, since for a given object resolution the number of points in the surface are, in general, significantly less than in the volume.

Although computed using a boundary mesh, the moments computed by integrating over the interior of the solid will be called *volume-like* moments. A different possibility is to treat the surface as a thin object of which to compute the moments. This second type will be called *surface-like* moments. Observe that for volume-like moments the surface must be closed, separating the space into an interior region and an exterior region, while for surface-like moments this is not necessary.

The mathematical definition of the two types of moments is, respectively

$$\mathcal{M}_I^{(\mathsf{V})} = \int_V \mathrm{d}V\, M_I(\boldsymbol{p}) \qquad \text{and} \qquad \mathcal{M}_I^{(\mathsf{S})} = \int_S \mathrm{d}S\, M_I(\boldsymbol{p})$$

where $\mathrm{d}V$ is the volume element in the volume V and $\mathrm{d}S$ the area element on the surface S. For their implementation we shall start from these two formulas. However, they can be formalized, consistently with (1), as moments of the density functions, respectively:

$$f^{(\mathsf{V})}(\boldsymbol{p}) = \begin{cases} 1 & \text{if } \boldsymbol{p} \in V \\ 0 & \text{if } \boldsymbol{p} \notin V \end{cases} \quad \text{and} \quad f^{(\mathsf{S})}(\boldsymbol{p}) = \int_S \mathrm{d}S|_{\boldsymbol{q}}\, \delta(\boldsymbol{p}-\boldsymbol{q})$$

where δ is the 3D Dirac-delta function and $\mathrm{d}S|_{\boldsymbol{q}}$ the area element evaluated at point $\boldsymbol{q}$. In the volume-like case, $f^{(\mathsf{V})}$ is the indicator function of the object interior.

3 Exact algorithm for geometric moments

3.1 Volume-like geometric moments

Using the notation introduced above, the geometric moments of the homogeneous object defined by the volume V are

$$\mathcal{G}_{ijk}^{(\mathsf{V})} = \int_V x^i\, y^j\, z^k\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z$$

The object is modeled by a closed triangle-mesh defining its boundary. Each triangle is characterized by three vertices: $\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}$, oriented consistently counter-clockwise when seen from the exterior. Using the origin, each of these triangles defines an oriented tetrahedron. As these tetrahedra are all oriented, an integral over the whole object can be expressed as a sum of integrals over them, irrespectively of the object topology and the point chosen as origin [7], [11].

The oriented volume is given by the determinant

$$\mathrm{Vol} = \frac{1}{6}\begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{vmatrix}$$

obtained from the Cartesian coordinates of the three ordered vertices of the triangle. The sign of this determinant gives the orientation of the tetrahedron. Thus, the integral over the whole object can be decomposed into a summation of the integrals over each tetrahedron T_c, defined by each triangular facet c, with its corresponding sign:

$$\mathcal{G}^{(\mathsf{V})}_{ijk} = \sum_{c\in\text{Facets}} \mathrm{sign}(\mathrm{Vol}_c) \int_{\mathcal{T}_c} x^i\, y^j\, z^k \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z$$

The integral over each tetrahedron can be obtained analytically. The first step is to perform a change of basis, parameterizing the tetrahedron in terms of the barycentric coordinates of the triangle and the fractional distance to the origin:

$$\boldsymbol{p} = \rho\,\boldsymbol{p}', \qquad \boldsymbol{p}' = \alpha\boldsymbol{p_1} + \beta\boldsymbol{p_2} + \gamma\boldsymbol{p_3} \tag{2}$$

with $\alpha,\beta,\gamma \geq 0$, $\alpha+\beta+\gamma = 1$ and $0 \leq \rho \leq 1$. The volume element in the coordinates α,β,ρ is given by the Jacobian of the transformation from Cartesian coordinates, which is proportional to the volume of the tetrahedron: $J = 6\mathrm{Vol}\,\rho^2$. Thus, the geometric moments can be expressed as

$$\mathcal{G}^{(\mathsf{V})}_{ijk} = \sum_{c\in\text{Facets}} 6\mathrm{Vol}_c \int_0^1 \mathrm{d}\rho\,\rho^{n+2} \int_0^1 \mathrm{d}\alpha \int_0^{1-\alpha} \mathrm{d}\beta$$
$$(\alpha x_1+\beta x_2+\gamma x_3)^i\,(\alpha y_1+\beta y_2+\gamma y_3)^j\,(\alpha z_1+\beta z_2+\gamma z_3)^k$$

with $n = i+j+k$ and $\gamma = 1-\alpha-\beta$, and where Vol_c includes the corresponding sign. Observe that the integral over ρ is independent of the others, resulting in a simple factor $\int_0^1 \mathrm{d}\rho\ \rho^{n+2} = \frac{1}{n+3}$. Hence,

$$\mathcal{G}^{(\mathsf{V})}_{ijk} = \frac{1}{n+3} \sum_{c\in\text{Facets}} 6\mathrm{Vol}_c\,\mathcal{S}_{ijk} \tag{3}$$

where

$$\mathcal{S}_{ijk} \equiv \int_0^1 \mathrm{d}\alpha \int_0^{1-\alpha} \mathrm{d}\beta\ \ (\alpha x_1 + \beta x_2 + \gamma x_3)^i$$
$$\times\,(\alpha y_1 + \beta y_2 + \gamma y_3)^j\,(\alpha z_1 + \beta z_2 + \gamma z_3)^k \tag{4}$$

Before advancing more in the expansion, we introduce the surface-like geometric moments, since the subsequent steps will be in common.

3.2 Surface-like geometric moments

Surface-like geometric moments of a surface S (not necessarily closed) are defined by

$$\mathcal{G}^{(\mathsf{S})}_{ijk} = \int_S \mathrm{d}S\ x^i\, y^j\, z^k$$

The object is still modeled by a triangular mesh. But, in this case, the object is the surface itself. The integral can be decomposed into the sum for all the facets of the integral performed for each triangle:

$$\mathcal{G}^{(\mathsf{S})}_{ijk} = \sum_{c\in\text{Facets}} \int_c \mathrm{d}S\ x^i\, y^j\, z^k$$

As for tetrahedra in the volumetric case, the integral over each triangle can be obtained analytically. Any point on the triangle can be expressed in barycentric coordinates in terms of the three vertices with the same expression as $\boldsymbol{p}'$ in (2). Taking $\gamma = 1-\alpha-\beta$, we can express the area element in each triangle as

$$\mathrm{d}S = 2\mathrm{Area}\ \mathrm{d}\alpha\,\mathrm{d}\beta$$

where

$$\mathrm{Area} = \frac{1}{2}\left\|(\boldsymbol{p}_1-\boldsymbol{p}_3)\wedge(\boldsymbol{p}_2-\boldsymbol{p}_3)\right\|$$

is the area of the triangle.

Thus, the surface-like geometric moments can be expressed as

$$\mathcal{G}^{(\mathsf{S})}_{ijk} = \sum_{c\in\text{Facets}} 2\mathrm{Area}_c\ \mathcal{S}_{ijk} \tag{5}$$

where $\mathcal{S}_{ijk}$ is defined by (4).

3.3 Common factor for both types of geometric moments

Observe that the formulas for volume-like (3) and surface-like (5) geometric moments are very similar. The only difference is the use of the volume or the area respectively, and the appearance of the factor $\frac{1}{n+3}$ in the volume-like ones. The following steps correspond to the computation of the common factor $\mathcal{S}_{ijk}$.

When expanding some of the powers in the following formulas, there will appear binomial coefficients:

$$\binom{p}{a} \equiv \frac{p!}{a!\,(p-a)!}$$

and trinomial coefficients:

$$(a \mid b \mid c) \equiv \frac{(a+b+c)!}{a!\,b!\,c!}$$

Expanding the powers in (4) and taking the Cartesian coordinates of the three vertices out of the integral, we get

$$\mathcal{S}_{ijk} = \sum_{\substack{i_1+i_2+i_3=i\\ 0\leq i_1,i_2,i_3}} \sum_{\substack{j_1+j_2+j_3=j\\ 0\leq j_1,j_2,j_3}} \sum_{\substack{k_1+k_2+k_3=k\\ 0\leq k_1,k_2,k_3}}$$
$$(i_1 \mid i_2 \mid i_3)\,{x_1}^{i_1}\,{x_2}^{i_2}\,{x_3}^{i_3}$$
$$\times\,(j_1 \mid j_2 \mid j_3)\,{y_1}^{j_1}\,{y_2}^{j_2}\,{y_3}^{j_3}$$
$$\times\,(k_1 \mid k_2 \mid k_3)\,{z_1}^{k_1}\,{z_2}^{k_2}\,{z_3}^{k_3}$$
$$\times \int_0^1 \mathrm{d}\alpha \int_0^{1-\alpha} \mathrm{d}\beta\ \alpha^{i_1+j_1+k_1}\,\beta^{i_2+j_2+k_2}\,\gamma^{i_3+j_3+k_3} \tag{6}$$

Then we need to solve the series of integrals

$$I_{abc} = \int_0^1 \mathrm{d}\alpha \int_0^{1-\alpha} \mathrm{d}\beta\ \ \alpha^a\,\beta^b\,(1-\alpha-\beta)^c$$

for all a, b, c. They are coefficients depending only on the numbers a, b, c, and independent of the triangle vertices:

$$I_{abc} = \frac{a!\, b!\, c!}{(a+b+c+2)!}$$

which is a particular case of a general formula proved in [12]. Using this expression and expanding the trinomial coefficients in (6), a collection of factorial factors appear. These can be rearranged to obtain a more convenient set of trinomials, enabling us to take some factorials out of the summations and to group the point-dependent factors by vertices instead of by coordinate axes:

$$\begin{aligned}\mathcal{S}_{ijk} = \frac{i!\, j!\, k!}{(n+2)!} \sum_{\substack{i_1+i_2+i_3=i \\ 0\le i_1,i_2,i_3}} \sum_{\substack{j_1+j_2+j_3=j \\ 0\le j_1,j_2,j_3}} \sum_{\substack{k_1+k_2+k_3=k \\ 0\le k_1,k_2,k_3}} \\ (i_1 \mid j_1 \mid k_1)\ {x_1}^{i_1}\, {y_1}^{j_1}\, {z_1}^{k_1} \\ \times (i_2 \mid j_2 \mid k_2)\ {x_2}^{i_2}\, {y_2}^{j_2}\, {z_2}^{k_2} \\ \times (i_3 \mid j_3 \mid k_3)\ {x_3}^{i_3}\, {y_3}^{j_3}\, {z_3}^{k_3}\end{aligned}$$

This rearrangement is crucial in order to reduce the computational time. Using the notation

$$C^{(a)}_{ijk} = (i \mid j \mid k)\ {x_a}^{i}\ {y_a}^{j}\ {z_a}^{k} \tag{7}$$

and

$$D_{abc} = \sum_{i_2=0}^{a} \sum_{j_2=0}^{b} \sum_{k_2=0}^{c} C^{(2)}_{i_2 j_2 k_2}\ C^{(3)}_{a-i_2,b-j_2,c-k_2}, \tag{8}$$

we arrive to the expression

$$\mathcal{S}_{ijk} = \frac{i!\, j!\, k!}{(n+2)!} \sum_{i_1=0}^{i} \sum_{j_1=0}^{j} \sum_{k_1=0}^{k} C^{(1)}_{i_1 j_1 k_1}\ D_{i-i_1,j-j_1,k-k_1}. \tag{9}$$

Observe that the intermediate quantities D_{abc} are independent of the indices i_1, j_1 and k_1. Hence, they can be computed independently and collected in an array.

3.3.1 Implementation

An array collecting the trinomial factors is precomputed for $n \le$ order. Then the algorithm runs through each facet sequentially, with the computation in each facet being independent from the others.

For each facet, the three vertices of the triangle, $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3$, are taken. Then, for each vertex independently, the series of monomial combinations of its components (7) is computed and stored. These three point-like combinations are then combined in two steps. First, monomials of $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ are combined (8), and second, the result is combined with $\boldsymbol{p}_1$ (9). The result is multiplied by the facet area (3) or volume (5), according to the moment type, and is added to the corresponding geometric moment.

This structure allows to compute the contribution of each facet to all the geometric moments with two loops of depth 6, instead of one loop of depth 9. In addition, the symmetry of the formulas simplifies the implementation, and the independence of the contribution from each facet allows an easy parallelization.

4 Approximate series algorithm for geometric moments

The algorithm described in the previous section is exact in the sense that, given an object defined *exactly* by a particular triangle mesh, the algorithm gives the exact geometric moments of this object. In this section, an algorithm to compute an approximation of geometric moments is described. Indeed, the algorithm provides a series of approximations which can be truncated at any desired degree, thus controlling the approximation error. It appears as a power series in a parameter, λ, dependent on the ratio of the average triangle size to the model size. Thus, the error introduced by the approximation at any degree falls off with the number of triangles modeling the object. If the series is computed up to the power λ^N, with N the maximum desired order of the geometric moments, then the algorithm becomes exact.

In the definition of $\mathcal{S}_{ijk}$ (4) the points on the triangle are written (2) in terms of the three vertices of the triangle:

$$\boldsymbol{p}' = \alpha \boldsymbol{p_1} + \beta \boldsymbol{p_2} + (1-\alpha-\beta)\boldsymbol{p_3}.$$

with $\alpha, \beta \ge 0$ and $\alpha + \beta \le 1$. Applying the change of variables, $\alpha = \alpha' + \frac{1}{3}$ and $\beta = \beta' + \frac{1}{3}$, and using the triangle centroid $\overline{\boldsymbol{p}} = (\boldsymbol{p_1} + \boldsymbol{p_2} + \boldsymbol{p_3})/3$ and the vectors $\boldsymbol{u} = \boldsymbol{p_1} - \boldsymbol{p_3}$ and $\boldsymbol{v} = \boldsymbol{p_2} - \boldsymbol{p_3}$, the point can be expressed as

$$\boldsymbol{p}' = \overline{\boldsymbol{p}} + \alpha' \boldsymbol{u} + \beta' \boldsymbol{v}.$$

with $\alpha', \beta' \ge -\frac{1}{3}$ and $\alpha' + \beta' \le \frac{1}{3}$.

Considering this change of variables we can rewrite (4) as

$$\begin{aligned}\mathcal{S}_{ijk} = \int_{-\frac{1}{3}}^{\frac{2}{3}} \mathrm{d}\alpha' \int_{-\frac{1}{3}}^{\frac{1}{3}-\alpha'} \mathrm{d}\beta' \\ (\overline{x} + \alpha' u_x + \beta' v_x)^i\, (\overline{y} + \alpha' u_y + \beta' v_y)^j\, (\overline{z} + \alpha' u_z + \beta' v_z)^k.\end{aligned}$$

Expanding the powers using the trinomial coefficients we get

$$\begin{aligned}\mathcal{S}_{ijk} = \sum_{\substack{i_1+i_2+i_3=i \\ 0\le i_1,i_2,i_3}} \sum_{\substack{j_1+j_2+j_3=j \\ 0\le j_1,j_2,j_3}} \sum_{\substack{k_1+k_2+k_3=k \\ 0\le k_1,k_2,k_3}} J_{i_2+j_2+k_2,i_3+j_3+k_3} \\ (i_1 \mid j_1 \mid k_1)\ (i_2 \mid j_2 \mid k_2)\ (i_3 \mid j_3 \mid k_3) \\ \overline{x}^{i_1}\, \overline{y}^{j_1}\, \overline{z}^{k_1}\ {u_x}^{i_2}\, {u_y}^{j_2}\, {u_z}^{k_2}\ {v_x}^{i_3}\, {v_y}^{j_3}\, {v_z}^{k_3}\end{aligned} \tag{10}$$

defining J_{ab} as the series of integrals

$$J_{ab} \equiv \int_{-\frac{1}{3}}^{\frac{2}{3}} \mathrm{d}\alpha \int_{-\frac{1}{3}}^{\frac{1}{3}-\alpha} \mathrm{d}\beta\ \alpha^a\, \beta^b$$

which is solved in appendix A, yielding an expression not as compact as the integrals I_{abc}.

Vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ have the length of the triangle edges. Let us assume that the object is centered at the origin and the triangle mesh is uniform so that all the triangles are almost equilateral and of similar size. This size can

be quantified with respect to the object radius with the parameter

$$\lambda \equiv \frac{\sqrt{\frac{4}{\sqrt{3}}\langle \text{Facet area}\rangle}}{\langle\|\overline{\boldsymbol{p}}\|\rangle} = \frac{\sqrt{\frac{4}{T\sqrt{3}}\left(\sum_{c\in\text{Facets}}\text{Area}_c\right)^3}}{\sum_{c\in\text{Facets}}\|\overline{\boldsymbol{p}}\|\,\text{Area}_c},$$

where $\langle\ \rangle$ denotes the mean on the mesh, T is the number of triangles and $\sqrt{3}/4$ is the area of an equilateral triangle with unit edges. This parameter is defined for the sake of a correct formalization of the aproximate series. However, it will not be explicitly used in the final algorithm formula and implementation. Using this parameter we can *normalize* the two vectors:

$$\hat{\boldsymbol{u}} \equiv \boldsymbol{u}/\lambda \qquad \text{and} \qquad \hat{\boldsymbol{v}} \equiv \boldsymbol{v}/\lambda.$$

Note that they are not unit vectors but their modulus will be of the same order of the object mean radius:

$$\|\hat{\boldsymbol{u}}\| \simeq \|\boldsymbol{u}\|\,\frac{\langle\|\overline{\boldsymbol{p}}\|\rangle}{\sqrt{\langle\|\boldsymbol{u}\|\|\boldsymbol{v}\|\rangle}} \simeq \langle\|\overline{\boldsymbol{p}}\|\rangle.$$

Including these definitions, (10) becomes

$$\begin{aligned}\mathcal{S}_{ijk} = \sum_{\substack{i_1+i_2+i_3=i\\ 0\le i_1,i_2,i_3}}\ \sum_{\substack{j_1+j_2+j_3=j\\ 0\le j_1,j_2,j_3}}\ \sum_{\substack{k_1+k_2+k_3=k\\ 0\le k_1,k_2,k_3}} & (i_1 \mid j_1 \mid k_1)\ (i_2 \mid j_2 \mid k_2)\ (i_3 \mid j_3 \mid k_3)\\ & \overline{x}^{i_1}\,\overline{y}^{j_1}\,\overline{z}^{k_1}\ \hat{u}_x{}^{i_2}\,\hat{u}_y{}^{j_2}\,\hat{u}_z{}^{k_2}\ \hat{v}_x{}^{i_3}\,\hat{v}_y{}^{j_3}\,\hat{v}_z{}^{k_3}\\ & J_{i_2+j_2+k_2,i_3+j_3+k_3}\ \lambda^{i_2+i_3+j_2+j_3+k_2+k_3}.\end{aligned}$$

This expression can be rearranged as a power series in the parameter λ:

$$\mathcal{S}_{ijk} = \sum_{r=0}^{n} \lambda^r\, S_{ijk}^{(r)}.$$

The contribution to each degree is

$$\begin{aligned}S_{ijk}^{(r)} \equiv \sum_{i_+=\max(0,r-j-k)}^{\min(i,r)} \binom{i}{i_+} \sum_{j_+=\max(0,r-k-i_+)}^{\min(j,r-i_+)} \binom{j}{j_+}\binom{k}{k_+}\\ \sum_{i_2=0}^{i_+}\binom{i_+}{i_2}\sum_{j_2=0}^{j_+}\binom{j_+}{j_2}\sum_{k_2=0}^{k_+}\binom{k_+}{k_2}\ J_{i_2+j_2+k_2,r-i_2-j_2-k_2}\\ \overline{x}^{i_1}\,\overline{y}^{j_1}\,\overline{z}^{k_1}\ \hat{u}_x{}^{i_2}\,\hat{u}_y{}^{j_2}\,\hat{u}_z{}^{k_2}\ \hat{v}_x{}^{i_3}\,\hat{v}_y{}^{j_3}\,\hat{v}_z{}^{k_3},\end{aligned} \tag{11}$$

where we have used $i_1 = i - i_+$, $j_1 = j - j_+$, $k_1 = k - k_+$, $i_3 = i_+ - i_2$, $j_3 = j_+ - j_2$, $k_3 = k_+ - k_2$ and $k_+ = r - i_+ - j_+$.

4.1 Degree 0

Implementing the degree 0 of this approximate series algorithm is extremely simple. The coefficients are reduced to

$$S_{ijk}^{(0)} = J_{00}\ \overline{x}^i\,\overline{y}^j\,\overline{z}^k.$$

Using that $J_{00} = 1/2$ we obtain

$$\mathcal{S}_{ijk} = \tfrac{1}{2}\overline{x}^i\,\overline{y}^j\,\overline{z}^k + \mathcal{O}(\lambda).$$

4.2 Degree 1

For degree 1 the coefficients are

$$\begin{aligned}S_{ijk}^{(1)} = {} & i\,\overline{x}^{i-1}\,\overline{y}^j\,\overline{z}^k(J_{10}\hat{u}_x + J_{01}\hat{v}_x)\\ & + j\,\overline{x}^i\,\overline{y}^{j-1}\,\overline{z}^k(J_{10}\hat{u}_y + J_{01}\hat{v}_y)\\ & + k\,\overline{x}^i\,\overline{y}^j\,\overline{z}^{k-1}(J_{10}\hat{u}_z + J_{01}\hat{v}_z).\end{aligned}$$

But $J_{10} = J_{01} = 0$. Hence, the contribution of the degree 1 identically vanishes:

$$\mathcal{S}_{ijk} = \tfrac{1}{2}\overline{x}^i\,\overline{y}^j\,\overline{z}^k + \mathcal{O}(\lambda^2).$$

This is an interesting property, which makes the approximation at degree 0 more accurate than expected.

4.3 Degree 2

For degree 2, the coefficients $S_{ijk}^{(2)}$ use the values $J_{11} = -1/72$ and $J_{20} = J_{02} = 1/36$, which lead to the approximation of degree 2:

$$\begin{aligned}\mathcal{S}_{ijk} = {} & \tfrac{1}{2}\overline{x}^i\,\overline{y}^j\,\overline{z}^k + \tfrac{1}{36}\Big(\tbinom{i}{2}\,\overline{x}^{i-2}\,\overline{y}^j\,\overline{z}^k(u_x{}^2 - u_xv_x + v_x{}^2)\\ & + \tbinom{j}{2}\,\overline{x}^i\,\overline{y}^{j-2}\,\overline{z}^k(u_y{}^2 - u_yv_y + v_y{}^2)\\ & + \tbinom{i}{2}\,\overline{x}^i\,\overline{y}^j\,\overline{z}^{k-2}(u_z{}^2 - u_zv_z + v_z{}^2)\\ & + ij\,\overline{x}^{i-1}\,\overline{y}^{j-1}\,\overline{z}^k(u_xu_y - \tfrac{1}{2}(u_xv_y + v_xu_y) + v_xv_y)\\ & + ik\,\overline{x}^{i-1}\,\overline{y}^j\,\overline{z}^{k-1}(u_xu_z - \tfrac{1}{2}(u_xv_z + v_xu_z) + v_xv_z)\\ & + jk\,\overline{x}^i\,\overline{y}^{j-1}\,\overline{z}^{k-1}(u_yu_z - \tfrac{1}{2}(u_yv_z + v_yu_z) + v_yv_z)\Big)\\ & + \mathcal{O}(\lambda^3).\end{aligned}$$

Observe that the factor λ^2 has been reintegrated into the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. Thus, as announced, the parameter λ does not appear explicitly in the final formulas, but it plays an essential role in their conceptualization and formalization.

5 Zernike–Canterakis moments algorithm

The definition of Z-C moments is given in [28], [29]. Their defining series of functions are the Z-C polynomials:

$$Z_{n\ell}^m(\boldsymbol{p}) = R_{n\ell}(r)Y_\ell^m(\theta,\varphi),$$

where $\{r,\theta,\varphi\}$ are the spherical coordinates. These polynomials are based on the spherical harmonics:

$$Y_\ell^m(\theta,\varphi) = \sum_{j=0}^{(\ell-m)/2} Y_{\ell j}^m\,(\cos\theta)^{\ell-m-2j}(\sin\theta)^m\,e^{im\varphi},$$

where

$$Y_{\ell j}^m = (-1)^j\,\frac{\sqrt{2\ell+1}}{2^\ell}\ \frac{(m \mid j \mid \ell-m-2j)\binom{2(\ell-j)}{\ell-j}}{\sqrt{(m \mid m \mid \ell-m)}}. \tag{12}$$

And they include a factor which is polynomial in the radial coordinate:

$$R_{n\ell}(r) = \sum_{\nu=0}^{k} Q_{k\ell\nu}r^{2\nu+\ell}$$

with $k = (n-\ell)/2$. The coefficients are defined by

$$Q_{k\ell\nu} = \frac{(-1)^{k+\nu}}{4^k}\sqrt{\frac{2\ell+4k+3}{3}} \times \frac{(\nu \mid k-\nu \mid \ell+\nu+1)\binom{2(\ell+\nu+1+k)}{\ell+\nu+1+k}}{\binom{2(\ell+\nu+1)}{\ell+\nu+1}}.$$

Although their definition is given in spherical coordinates, Z-C polynomials are actually polynomial functions on Cartesian coordinates:

$$Z_{n\ell}^m(\boldsymbol{p}) = \sum_{\nu=0}^{k} Q_{k\ell\nu} \sum_{j=0}^{(\ell-m)/2} Y_{\ell j}^m \, r^{2(\nu+j)} \, z^{\ell-m-2j} \, (x+iy)^m.$$

Observe that the definition of spherical harmonics used (12) follows the 4π-normalization convention (see appendix B for more details and comparison with other formulations). Following this convention, and in analogy with the exponential Fourier series, the definition of Z-C moments [28] is given by

$$\mathcal{Z}_{n\ell}^m = \frac{3}{4\pi}\int_V \mathrm{d}V \, f(\boldsymbol{p}) \, \overline{Z_{n\ell}^m(\boldsymbol{p})},$$

where the over-line denotes the complex conjugate.

We propose an efficient algorithm for computing Z-C moments from geometric moments. The strategy is based on the definition of a chain of four auxiliary series of polynomials:

$$\begin{aligned} V_{abc} &= x^a z^b (x+iy)^{a+c} \\ W_{abc} &= (x^2+y^2)^a z^b (x+iy)^c \\ X_{abc} &= r^{2a} z^b (x+iy)^c \\ \hat{Y}_{\ell\nu}^m &= r^{2\nu+\ell} Y_\ell^m = \sum_{j=0}^{(\ell-m)/2} Y_{\ell j}^m \, r^{2(\nu+j)} z^{\ell-m-2j} (x+iy)^m. \end{aligned}$$

The associated moments for each of these series of polynomials will be denoted by the corresponding calligraphic character. They can be computed recursively as

$$\mathcal{V}_{abc} = \sum_{\alpha=0}^{a+c} i^\alpha \binom{a+c}{\alpha} \mathcal{G}_{2a+c-\alpha,\alpha,b} \tag{13a}$$

$$\mathcal{W}_{abc} = \sum_{\alpha=0}^{a} (-1)^\alpha \, 2^{a-\alpha} \binom{a}{\alpha} \mathcal{V}_{a-\alpha,b,c+2\alpha} \tag{13b}$$

$$\mathcal{X}_{abc} = \sum_{\alpha=0}^{a} \binom{a}{\alpha} \mathcal{W}_{a-\alpha,b+2\alpha,c} \tag{13c}$$

$$\hat{\mathcal{Y}}_{\ell\nu}^m = \sum_{j=0}^{(\ell-m)/2} Y_{\ell j}^m \, \mathcal{X}_{\nu+j,\ell-m-2j,m} \tag{13d}$$

$$\mathcal{Z}_{n\ell}^m = \frac{3}{4\pi} \sum_{\nu=0}^{(n-\ell)/2} Q_{k\ell\nu} \, \overline{\hat{\mathcal{Y}}_{\ell\nu}^m} \, . \tag{13e}$$

In order to compute Z-C moments up to order $n = N$, $\mathcal{V}_{abc}$, $\mathcal{W}_{abc}$ and $\mathcal{X}_{abc}$ need to be computed for $a, b, c \geq 0$ and $2a+b+c \leq N$, $\hat{\mathcal{Y}}_{\ell\nu}^m$ for $0 \leq m \leq \ell \leq N$ and $0 \leq \nu \leq (N-\ell)/2$, and $\mathcal{Z}_{n\ell}^m$ for $0 \leq m \leq \ell \leq n \leq N$ with $n-\ell$ even.

This formulation of the Z-C moments of an object in terms of its geometric moments passing through the chain of auxiliary moments gives a very efficient algorithm for their computation, involving 5 loops of depth 4. In contrast, the algorithm proposed in [30] involves one loop of depth 6.

6 Algorithm Evaluation

6.1 Computational complexity analysis

6.1.1 Geometric moments

Both algorithms introduced for the computation of geometric moments, exact and approximate series, are linear in the number of triangles of the mesh. Compared with a volumetric image with L^3 grid dimension, the number of triangles, and thus the complexity of the presented algorithm, can be considered of the order of L^2.

Regarding the maximum order N of the computed geometric moments, the exact algorithm is of order N^6. This complexity coincides with the direct extension to 3D of the correct (in the simplest case considering constant function value in each voxel) algorithm for 2D images [4]. However, in computer vision and image analysis it is common to approximate the moments applying a summation over the image grid instead of an integral. The straightforward algorithm from this discrete approximation is of order N^3.

The 0th-degree algorithm proposed in this paper is a kind of discrete version for surface-based computation of moments. Thus, it shares the low computational complexity of both approaches: $L^2 \times N^3$.

The introduction of the approximate series algorithm is meant to improve the accuracy of the approximation of the discrete approach. For any degree r, the rth-degree algorithm is of the same order with respect to N. However, their computational complexity depends on the degree: $L^2 \times N^3 \times r^5$. For this reason, using high degrees ($r \simeq N$) can be counterproductive. But for low degrees ($r \ll N$), the accuracy is significantly improved while the computational complexity is kept low.

6.1.2 Zernike–Canterakis moments

As mentioned above, the proposed algorithm for the computation of Z-C moments involves 5 loops of depth 4. Hence, its computational complexity is N^4. Observe that it is completely independent of the image or mesh resolution since it is applied to the computed geometric moments. In addition, it can be pipelined to whatever algorithm used for computing geometric moments. If the approximate rth-degree algorithm is used, then the total computational complexity is $L^2 \times N^3 \times r^5 + N^4$. As in general $L \gg N$, the computational time involved in computing Z-C moments from geometric moments is negligible.

6.2 Estimation of the power series behavior

In general, an accurate and smooth object representation requires the use of sufficiently small triangles. We can then assume that the parameter satisfies $\lambda \ll 1$. As an example, a rough estimation for a uniform mesh of 20K triangles gives $\lambda^2 \simeq 10^{-3}$.

The expression of $S_{ijk}^{(r)}$ (11) includes the triangle-dependent factors

$$\overline{x}^{i_1}\,\overline{y}^{j_1}\,\overline{z}^{k_1}\;\hat{u}_x{}^{i_2}\,\hat{u}_y{}^{j_2}\,\hat{u}_z{}^{k_2}\;\hat{v}_x{}^{i_3}\,\hat{v}_y{}^{j_3}\,\hat{v}_z{}^{k_3}.$$

The points and vectors $\overline{\boldsymbol{p}}$, $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{v}}$ are in general all of the same order ($\|\overline{\boldsymbol{p}}\|, \|\hat{\boldsymbol{u}}\|, \|\hat{\boldsymbol{v}}\| \simeq \langle\|\overline{\boldsymbol{p}}\|\rangle$). There can be always some proportion of triangles with some of the coordinates close to zero, so that they will be of a lower order of magnitude. Thus, these factors will satisfy:

$$\overline{x}^{i_1}\,\overline{y}^{j_1}\,\overline{z}^{k_1}\;\hat{u}_x{}^{i_2}\,\hat{u}_y{}^{j_2}\,\hat{u}_z{}^{k_2}\;\hat{v}_x{}^{i_3}\,\hat{v}_y{}^{j_3}\,\hat{v}_z{}^{k_3} \lesssim \langle\|\overline{\boldsymbol{p}}\|\rangle^n.$$

However, it is expectable that their summation over all the facets will be dominated by the order of magnitude $\langle\|\overline{\boldsymbol{p}}\|\rangle^n$, which is independent of the degree r. Taking this value as an approximation for estimating the coefficients $S_{ijk}^{(r)}$, we can recompute the series from the integral. After some algebra we obtain:

$$S_{ijk} \simeq \langle\|\overline{\boldsymbol{p}}\|\rangle^n \frac{(3n\lambda+5\lambda-3)(3+\lambda)^{n+1}+(3-2\lambda)^{n+2}}{3^{n+2}(n+1)(n+2)\lambda^2}$$

and

$$S_{ijk}^{(r)} \simeq \langle\|\overline{\boldsymbol{p}}\|\rangle^n \frac{3r+5+(-2)^{r+2}}{3^{r+2}(r+1)(r+2)}\binom{n}{r}.$$

For $r=0$ and $r=2$ it gives

$$S_{ijk}^{(0)} \simeq \langle\|\overline{\boldsymbol{p}}\|\rangle^n \frac{1}{2} \quad \text{and} \quad S_{ijk}^{(2)} \simeq \langle\|\overline{\boldsymbol{p}}\|\rangle^n \frac{1}{36}\binom{n}{2}.$$

Fig. 1 shows the behavior of this estimation of S_{ijk} and its 0th- and 2nd-degree approximations as a function of λ for different values of n. The error introduced by the approximations increases with the order n and with the value of λ, and the use of 2nd-degree instead of 0th-degree approximation reduces the relative error by almost 2 orders of magnitude.

6.3 Experimental results

Four types of experiments have been performed to evaluate the reconstruction correctness, accuracy, computational time and numerical stability of the different algorithms. The algorithms have been implemented in C++ using double float arithmetic and their behavior has been tested on different publicly available shapes and from some of our own databases. All the evaluations were run on a Pentium® IV 3GHz with 2Gb of RAM.

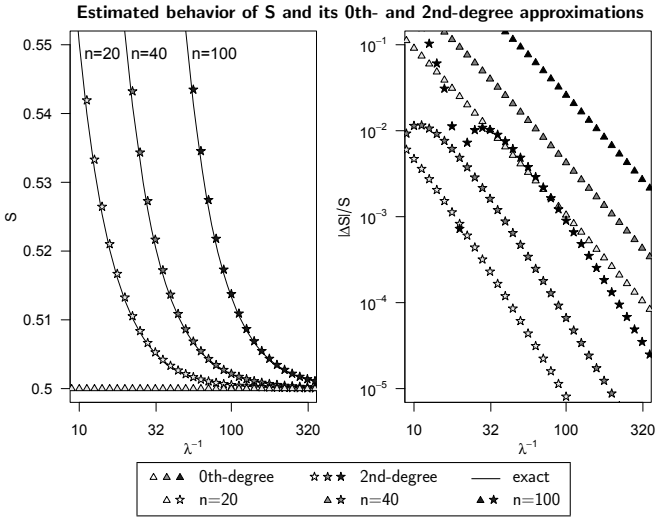

Fig. 1. *Left:* Estimated behavior of S_{ijk} as a function of λ, compared with its 0th-degree $S_{ijk}^{(0)}$ and 2nd-degree $S_{ijk}^{(0)}+S_{ijk}^{(2)}\lambda^2$ approximations. *Right:* Relative error ($|\Delta S|/S$) of using 0th- and 2nd-degree approximations with respect to the exact S_{ijk}.

6.3.1 Reconstruction correctness

Two series of meshes with different number of triangles modeling a sphere and a torus have been synthetically produced with *Paraview 2.6* (Kitware, Inc., Illinois, USA). For all the meshes, the number of triangles has been kept small, and the triangle size uniformity and quality rather poor. The goal of this experiment is to qualitatively test the limits of the approximate algorithms of 0th- and 2nd-degree, when the number of triangles is reduced and when the moment order is increased. Z-C moments have been computed for every mesh up to different orders with 0th-degree, 2nd-degree and exact algorithms. Then, the object indicator function has been reconstructed from the computed Z-C moments using (15), and its isosurface at 0.5 has been extracted using marching cubes.

Fig. 2 compares the reconstruction for a sphere from a very sparse mesh (64 triangles) with 0th-degree and exact algorithms. For 0th-degree, when the order increases, the effect of approximating each triangle by its centroid becomes more evident. Whereas, for the exact algorithm, there appears an excessive adaptation to the particular mesh, which is in general an undesired effect when modeling a smooth object.

Fig. 3 shows the reconstructions up to different orders with each of the three algorithms and from two different meshes (400 and 1K triangles) representing a torus. It can be seen that as the order increases, the reconstruction from the approximate moments becomes more sensitive to the triangles size, but that 2nd-degree algorithm is less sensitive than 0th-degree algorithm.

6.3.2 Accuracy

As commented above, in general when an object is modeled by a triangle mesh some error is introduced. This error depends on all the combined imaging and

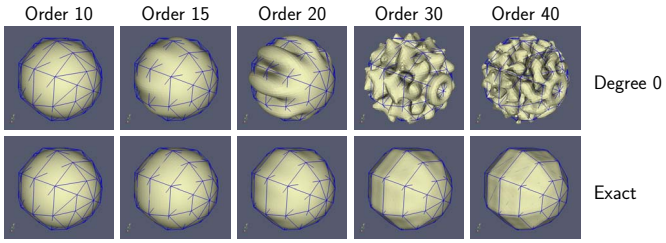

Fig. 2. Reconstruction obtained for a 64 triangle mesh representing a sphere, from the Z-C moments computed with 0th-degree and exact algorithms up to different orders.

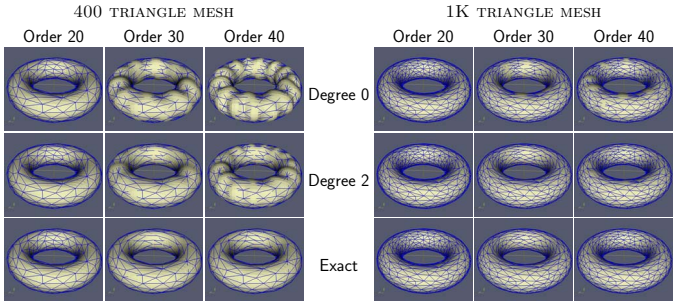

Fig. 3. Reconstruction obtained for a 400 triangle mesh and a 1K triangle mesh representing a torus, from the Z-C moments computed with 0th-degree, 2nd-degree and exact algorithms up to different orders.

modeling process. The error introduced by this modeling results in the corresponding error in the geometric and Z-C moments computed from the mesh. The order of magnitude of this error in the moments will be taken as a reference error level. Thus, an approximate computation introducing an error smaller than this error level can be considered negligible.

One of the relevant factors in the modeling, which is likely independent of the application and completely controllable, is the quality and number of triangles used in the final mesh. For 4 different objects (see Fig. 4), a smooth, uniform and high quality triangle mesh with 1 million facets has been considered as the reference model. From the reference model, the Z-C moments have been computed with the exact algorithm to obtain the reference Z-C moments. Then a series of meshes has been created by sequentially reducing the number of triangles down to 4K using the software Remesh [33]. Two different processes have been followed for this facet number reduction. In the first one, all the meshes are constrained to be uniform with nearly equilateral triangles. In the second one, the triangles size and shape are allowed to adapt according to the object curvature (see Fig. 5). Then the error in the Z-C moments computed from the reduced meshes compared to the reference moments has been computed for each mesh size, for uniform and adaptive meshes, using the exact algorithm and the approximate algorithms of 0th and 2nd-degree, and for orders 10, 20 and 30. The error has been measured using the Euclidean distance of the Z-C moments divided by the Euclidean modulus of the reference:

$$\text{Error} = \frac{\sqrt{\sum |\mathcal{Z}_{n\ell}^{m} - \mathcal{Z}_{n\ell}^{m\,(\text{ref})}|^2}}{\sqrt{\sum |\mathcal{Z}_{n\ell}^{m\,(\text{ref})}|^2}}$$

Fig. 6 shows the results obtained with the first shape model, for both surface-like and volume-like moments. The results obtained with the other 3 shapes are very similar.

For uniform meshes the error introduced by the 2nd-degree algorithm is indistinguishable from the one obtained with the exact algorithm, while for the 0th-degree algorithm the error is only slightly larger. As expected, for the exact algorithm, adaptive meshes introduce less error than uniform meshes with the same number of triangles. Despite this, it is interesting to observe that, for the 0th-degree algorithm and $N \geq 20$, the error on adaptive meshes is larger than on the corresponding uniform meshes. This is reasonable since adaptive meshes have more elongated triangles, causing the approximation to be less accurate. However, the 2nd-degree algorithm with adaptive meshes produces errors substantially smaller than the exact algorithm with uniform meshes. For example, for surface-like moments with $N = 20$, the error for an adaptive mesh of 30K triangles with the 2nd-degree algorithm is 3.6×10^{-4}, while for

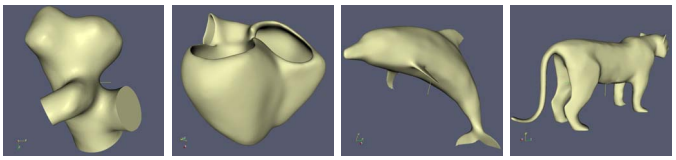

Fig. 4. Smooth 1M-triangle models of the 4 shapes used for the accuracy experiments. The two models on the left are a cerebral aneurysm and a heart from our own medical image databases. The two models on the right have been generated from the m73 and the m95 shapes of the Princeton Shape Benchmark [32].

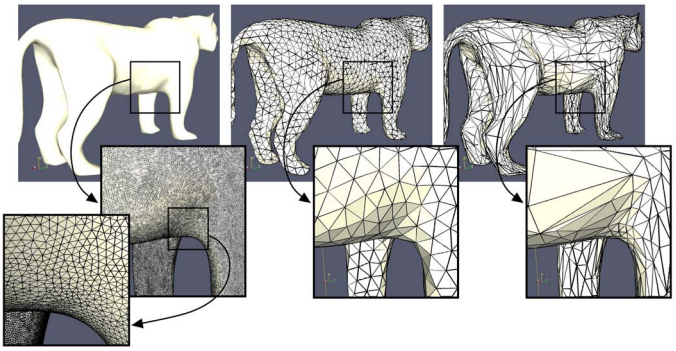

Fig. 5. Detail of the 1M-triangle reference mesh (left) for the fourth shape and the corresponding 4K triangle meshes of both types: uniform (middle) and adaptive (right). For uniform meshes, the triangles are optimized for being as equilateral and equally sized as possible. For adaptive meshes, the triangles adapt their shape and size to the curvature of the represented object boundary.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

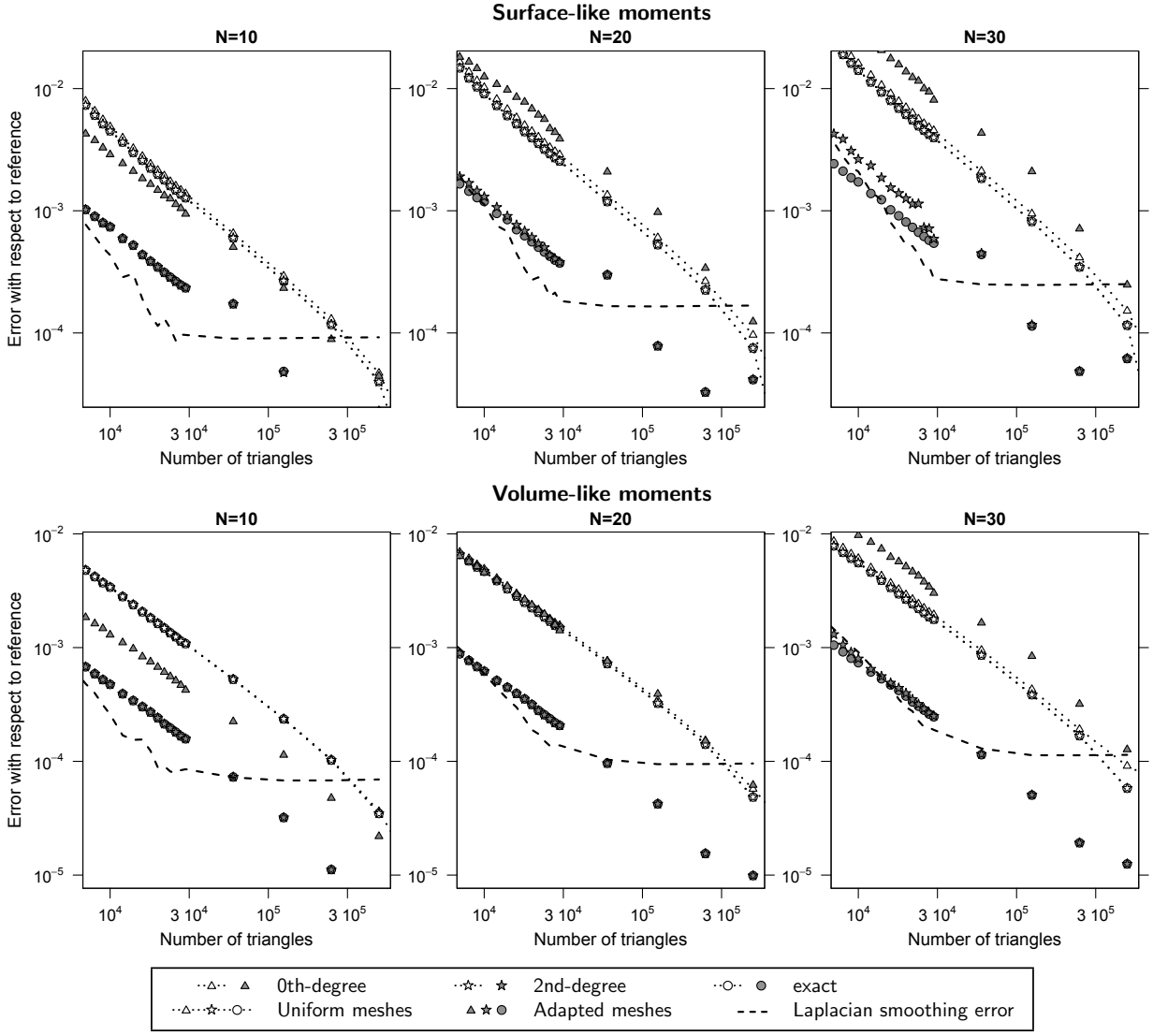

Fig. 6. Error in the surface-like and volume-like Z-C moments of the first shape computed up to orders 10, 20 and 30 with exact, 0th-degree and 2nd-degree algorithms, from meshes of different size and quality. For uniform meshes, the circles corresponding to the error of the exact algorithm are completely superposed with the stars displaying the error of the 2nd-degree algorithm. It is visible that there is also some superposition for adaptive meshes with more than 30K triangles between exact and 2nd-degree algorithms. The dashed line displays the error introduced in the mesh by one step of Laplacian smoothing in the reference model.

the equally sized uniform mesh the error with the exact algorithm is 2.2×10^{-3} and it would be necessary to use a uniform mesh with 150K triangles to obtain a similar error.

We can also observe that the higher the order, the larger the error for all the algorithms. But the errors for the approximate algorithms increase faster than the ones for the exact algorithm. This is especially evident for the 0th-degree algorithm with adaptive meshes, but can also be seen for the 2nd-degree algorithm.

It is also interesting to notice that surface-like moments are more sensitive than volume-like moments to both mesh accuracy and approximate algorithm errors. Observe that the errors of volume-like moments for orders $N = 20$ and $N = 30$ are similar to the errors of surface-like moments for order $N = 10$ and $N = 20$ respectively.

In order to complement the estimation of the *acceptable* error level, a series of pairs of uniform meshes representing the same object with the same number of triangles have been created. A Laplacian smoothing of one step with 100% neighbor weighting has been applied to the reference model with 1 million facets used above. This produces a second model where the only difference with respect to the original is a minor remeshing operation considered almost trivial in many applications. Then, the same process as described above to reduce the facet number keeping uniform meshes is applied to the smoothed copy to produce another series of differently sized meshes. The union of both series is a paired series of uniform meshes ranging from 1 million to 4K facets, where each mesh is differentiated from its pair by this

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

smoothing step. The Z-C moments of each pair have been computed with the exact algorithm. Then, the Z-C moments error of each mesh with respect to its pair has been computed as an estimate of the error introduced by this processing step. This error is displayed in Fig. 6 by the solid dashed lines. Observe that it is very stable between 10^{-3} and 10^{-4} for all the cases. Thus, it seems that for most applications a computational error below 10^{-4} can be considered negligible.

The accuracy of the approximate algorithms, isolated from the mesh representation error, can be measured with the error of the approximate Z-C moments with respect to the exact Z-C moments of the same mesh, instead of with respect to the reference moments from the 1 million triangle mesh. Fig. 7 (left) displays this error for volume-like moments up to order $N = 20$ against the number of triangles. In Fig. 7 (right) the same error is plotted against the corresponding parameter λ and superimposing the results for the 4 different shapes. We can observe that, for uniform meshes, the values are in agreement with the estimation of the relative error displayed in Fig. 1. The error for adaptive meshes is larger than for uniform meshes with the same λ, since the hypothesis of nearly equilateral and similarly sized triangles made for the accuracy estimation is not satisfied. In this sense, it can be seen that the difference between the errors for adaptive meshes and uniform meshes increases as λ grows since, in the series used for the experiments, adaptive meshes with fewer triangles are less regular. It is also worth observing that the error of the 2nd-degree algorithm on both, uniform and adaptive meshes, is markedly smaller than the error introduced by a Laplacian smoothing on the 1-million-triangle mesh.

These results allow to compare the accuracy obtained by using the 2nd-degree algorithm on a uniform mesh of T triangles with the accuracy obtained using the 0th-degree algorithm but after subdividing the triangles. For comparable computational times, the ratio of the relative errors between the 2nd- and 0th-degree algorithms for $N = 20$ will be around $5/\sqrt{T}$, which decreases in favor of the precision of the 2nd-degree algorithm as the number of triangles increases.

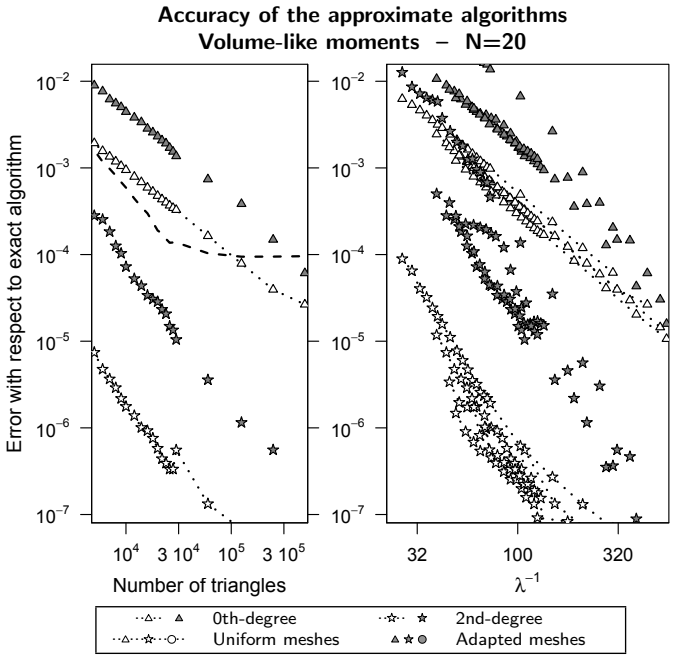

Fig. 7. Error in the volume-like Z-C moments computed up to order 20 with the 0th- and 2nd-degree algorithms, with respect to the Z-C moments computed with the exact algorithm from the same mesh. It is plotted against the number of triangles for the first shape (left) and against the corresponding parameter λ for the four shapes (right). The dashed line is the error introduced by one step of Laplacian smoothing in the reference model.

6.3.3 Computational time

The same meshes of different sizes used for the accuracy experiments have been used for obtaining the computational times involved in computing Z-C moments up to orders 10, 20 and 30 using the exact and the 0th- and 2nd-degree algorithms. Fig. 8 shows the error for each algorithm and mesh size and quality against its computational time. It can be seen that, concerning these two factors, the optimal choice is the 2nd-degree algorithm applied to adaptive meshes. This fact becomes more evident when increasing the order. However, if the available mesh is uniform the best option is the 0th-degree algorithm. By far, the worst option is the exact algorithm applied to uniform meshes, which for the same error level introduces a factor in the computational time with respect to the best option of around 60, 300 and 1000 for orders $N = 10$, 20 and 30, respectively. For order $N = 20$ and error 10^{-3}, for instance, the computational time is $0.6\,\mathrm{s}$ for 2nd-degree on adaptive meshes, $1.2\,\mathrm{s}$ for 0th-degree either on adaptive meshes or uniform meshes, $4\,\mathrm{s}$ for 2nd-degree on uniform meshes, $30\,\mathrm{s}$ for exact algorithm on adaptive meshes, and $170\,\mathrm{s}$ for exact algorithm on uniform meshes.

As commented in Subsection 6.1.2, all exact algorithms previously presented have higher computational complexity (N^9). For comparison, the algorithm in [19] has been applied to some of the meshes used in this experiment. The computational times obtained increase the time needed for our exact algorithm with a multiplicative factor of 30, 110 and 310, for orders $N = 10$, 20 and 30, respectively. For order $N = 20$ and error 10^{-3}, the computational time increases to $3300\,\mathrm{s}$ for adaptive meshes and $19000\,\mathrm{s}$ for uniform meshes.

Let us remark that these computational times are mainly due to the geometric moment computation. In agreement with the computational complexity analysis done in Subsection 6.1.2, the time involved in the computation of Z-C moments from geometric moments is negligible: $0.1\,\mathrm{ms}$, $0.9\,\mathrm{ms}$ and $3.9\,\mathrm{ms}$ for orders $N = 10$, 20 and 30 respectively, regardless of the mesh size.

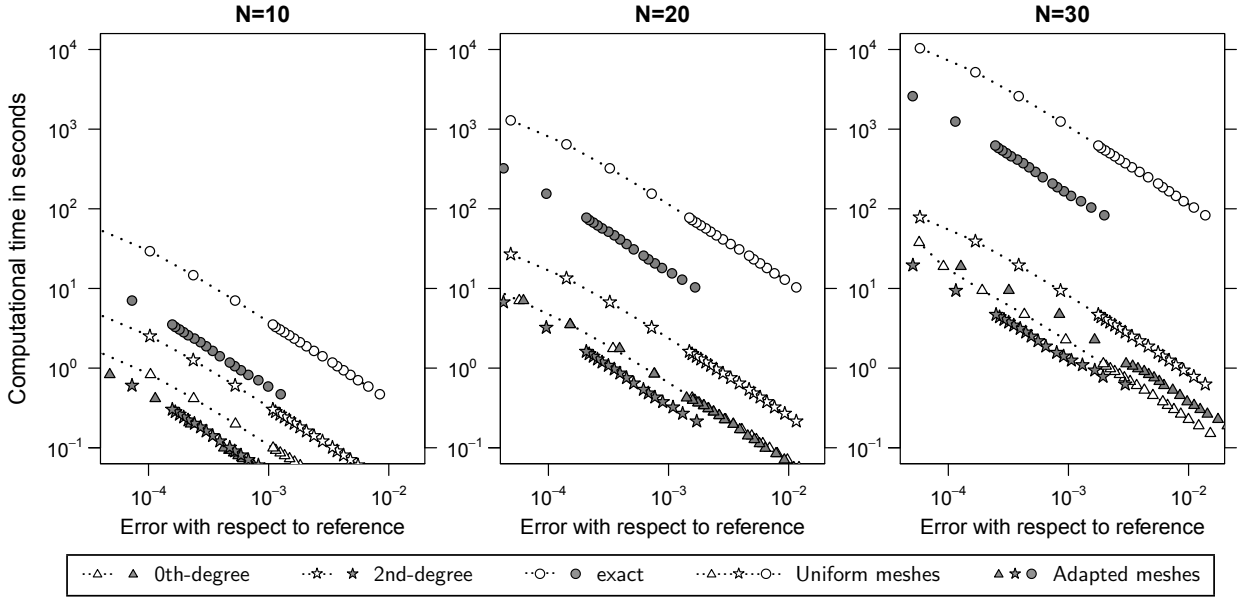

Fig. 8. Computational time required for volume-like Z-C moments of the first shape computed up to orders 10, 20 and 30 with exact, 0th-degree and 2nd-degree algorithm from meshes of different size and quality against their error with respect to reference.

6.3.4 Numerical stability

An important issue in any algorithm is its numerical stability when, say, double float computer precision is used instead of infinitely precise real numbers. To test this aspect, we have considered a solid cube of edge length 0.5 aligned with the positive Cartesian axes and with a vertex in the origin, because its geometric moments can be analytically computed:

$$\mathcal{G}_{ijk} = \frac{2^{-(i+j+k+3)}}{(i+1)(j+1)(k+1)}, \tag{14}$$

and can be exactly represented by a finite mesh. 0th and 2nd-degree algorithms for geometric moments up to order $N = 150$ have been applied to 2K, 125K and 2M-triangle meshes representing this cube. Exact algorithm has been applied only to the 2K-triangle mesh due to its computational cost. Figure 9 (top) shows the maximum relative errors with respect to analytically computed geometric moments for each order. The relative errors for the exact algorithm ($\sim 10^{-14}$) correspond to the double float precision. For 0th and 2nd-degree algorithms up to order 90 and 110 for 2M and 125K triangles respectively, the error seems to be attributable to the approximate algorithm accuracy. From these orders the numerical error is still small, at least up to order 125, where a rapid divergence appears for 2M-triangle mesh.

Z-C moments up to order 60 have been computed from the previously computed geometric moments and also from their analytical expression (14), using both the new algorithm and the previous one [30]. From the orthonormality of Z-C polynomials and the homogeneity of the object from which the moments are computed, it follows the identity

$$\sum_{n=0}^{\infty} \sum_{\ell} \sum_{m} |\mathcal{Z}_{n\ell}^{m}|^2 = \mathcal{Z}_{00}^{0}$$

This implies that the Euclidean norm of Z-C moments up to order N converges to $\sqrt{\mathcal{Z}_{00}^{0}}$ when N increases, which can be used as a test for the numerical stability of the algorithm. Figure 9 (bottom) shows this convergence for the computed Z-C moments, which breaks down at $N = 50$ for the new algorithm and at slightly larger order $N = 53$ for the previously published method. Very likely, this divergence is caused by the addition and subtraction of terms including binomial factors (13ab), which exceed double float precision for $N \simeq 50$.

7 Conclusion

In this paper two types of algorithms have been presented for the computation of geometric moments of an homogeneous volumetric or a thin surface-like object, from a triangle mesh representing the object boundary or the object itself respectively. The first algorithm is exact for the particular triangular mesh, while the second one is an approximation, expressed as a power series in a parameter λ proportional to the mesh triangle size and can be truncated at any desired degree r. The exact algorithm already represents a reduction in the computational complexity with respect to the moment order, N, from N^9, for the previously proposed algorithms [7], [8], [16], [18], [11], [19], to N^6. The approximate algorithm with $r \ll N$ further reduces the computational complexity to $N^3 \times r^5$.

In addition, an algorithm for the computation of 3D Zernike–Canterakis moments from geometric moments

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

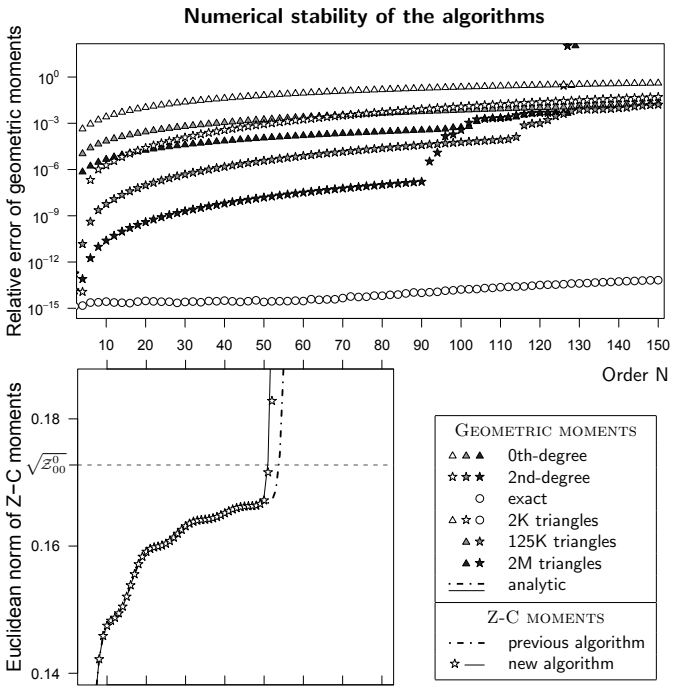

Fig. 9. Numerical stability tested on the moments of a cube. Left: Maximum relative error of order N geometric moments computed with exact, 0th-degree and 2nd-degree algorithm from meshes of 2K, 125K and 2M triangles with respect to the exact analytical values. Right: Euclidean norm of Z-C moments up to order N computed with the previously existing algorithm and the new algorithm, from geometric moments computed analytically and with 2nd-degree algorithm.

has been introduced. It reduces the computational complexity with respect to the only previously existing algorithm [30] from N^6 to N^4.

The accuracy of the approximation increases with the number of triangles of the mesh and with the approximation degree r. From the experiments performed, even for low degrees as 0 and 2, the error introduced can be acceptable or even negligible. In particular, this error has been compared with the error introduced by modeling a smooth object with a finite-number-of-triangle mesh. While both errors are of the same level for the 0th-degree algorithm applied to uniform meshes, the error introduced by the 2nd-degree algorithm is substantially smaller for both, uniform meshes and adaptive meshes. The error of the 2nd-degree algorithm is also smaller than the error derived from applying an additional one-step Laplacian smoothing during the mesh generation.

Regarding the computational time needed for each algorithm in order to obtain a predetermined error level, the most efficient algorithm is the 2nd-degree algorithm applied to adaptive meshes, followed by the 0th-degree algorithm applied to uniform meshes. By far the least efficient is the exact algorithm applied to uniform meshes. Besides, the time involved in the computation of Zernike–Canterakis moments from geometric moments with the proposed algorithm is negligible.

Both types of algorithms for geometric moments have a good numerical stability at high orders. In the worst case, for dense meshes, approximate algorithms are stable up to order 125. And no instability has been found for the exact algorithm in our tests up to order 150. The stability of the presented algorithm for Zernike–Canterakis moments is much lower (up to order 50), but is only slightly lower than the only previously existing algorithm. Hence, new algorithms for Z-C moments (or a higher precision or exact arithmetics implementation) should be developed if orders higher than 50 are required.

These results enable us to conclude that the proposed 2nd-degree algorithm for computing 3D geometric moments, together with the algorithm for computing Zernike–Canterakis moments from them, is an efficient and accurate method for obtaining moments from an object modeled by a triangular mesh, surpassing by 2 to 3 orders of magnitude the computational time of the exact algorithm, and by 2 additional orders of magnitude the previously proposed algorithms.

APPENDIX A
COMPUTATION OF THE INTEGRALS J_{ab}

Section 4 presented the series of integrals

$$J_{ab} = \int_{-\frac{1}{3}}^{\frac{2}{3}} \mathrm{d}\alpha \int_{-\frac{1}{3}}^{\frac{1}{3}-\alpha} \mathrm{d}\beta \;\; \alpha^a \, \beta^b.$$

To solve them, first we apply the change of variables $\alpha' = 3\alpha$ and $\beta' = 3\beta$:

$$J_{ab} = \frac{1}{3^{a+b+2}} \int_{-1}^{2} \mathrm{d}\alpha' \int_{-1}^{1-\alpha'} \mathrm{d}\beta' \; \alpha'^a \, \beta'^b.$$

Solving the integral for β':

$$J_{ab} = \frac{1}{3^{a+b+2}} \frac{1}{b+1} \int_{-1}^{2} \mathrm{d}\alpha' \; \alpha'^a \left[(1-\alpha')^{b+1} - (-1)^{b+1} \right].$$

We then define the quantities

$$K_{ab} \equiv \int_{-1}^{2} \mathrm{d}\alpha' \; \alpha'^a (1-\alpha')^b,$$

giving

$$J_{ab} = \frac{1}{3^{a+b+2}} \frac{1}{b+1} (K_{a,b+1} - (-1)^{b+1} K_{a0}).$$

In its turn, K_{ab} can be computed recursively:

$$K_{a,b+1} = K_{ab} - K_{a+1,b}$$

with

$$K_{a0} = \frac{1}{a+1} (2^{a+1} - (-1)^{a+1}).$$

These formulas allow us to compute the integrals J_{ab} for any value of a and b. Their expression as a closed formula could be written using hypergeometric functions, but it would be hardly useful. Observe also that, although they are by construction symmetric: $J_{ab} = J_{ba}$

and $K_{ab} = K_{ba}$, the obtained formulas do not manifest these symmetries.

For the lower values, we get:

$$K_{00} = 3, \quad K_{10} = K_{01} = \tfrac{3}{2}, \quad K_{20} = K_{02} = 3,$$
$$K_{11} = -\tfrac{3}{2}, \quad K_{30} = K_{03} = \tfrac{15}{4}, \quad K_{21} = K_{12} = -\tfrac{3}{4}$$

and

$$J_{00} = \tfrac{1}{2}, \quad J_{01} = J_{10} = 0, \quad J_{11} = -\tfrac{1}{72}, \quad J_{02} = J_{20} = \tfrac{1}{36}.$$

APPENDIX B
COMPARISON WITH OTHER FORMULATIONS OF ZERNIKE–CANTERAKIS MOMENTS

In Section 5 the definition of Zernike–Canterakis moments is explicitly written. A component of this definition is the set of spherical harmonics. The coefficients defining these functions (12) correspond to the 4π-normalization without Condon-Shortley phase. This is the normalization used for its definition in [28], although the phase convention is not explicitly stated. However, a non-standard i^m phase is used in [30]. There, a further confusion could follow, since first a definition with complete normalization is given, but later the 4π-normalization is used.

Coherently with this 4π-normalization, Z-C polynomials are defined to satisfy the normalization

$$\frac{3}{4\pi} \int_V \mathrm{d}V \, Z_{n\ell}^m(\boldsymbol{p}) \, \overline{Z_{n'\ell'}^{m'}(\boldsymbol{p})} = \delta_{nn'} \delta_{\ell\ell'} \delta^{mm'}.$$

The original definition of Z-C moments in [28] includes this factor:

$$\mathcal{Z}_{n\ell}^m = \frac{3}{4\pi} \int_V \mathrm{d}V \, f(\boldsymbol{p}) \, \overline{Z_{n\ell}^m(\boldsymbol{p})}.$$

Thus, the *reconstruction* of the function (or inverse transformation) does not include any factor:

$$f(\boldsymbol{p}) = \sum_{n=0}^{\infty} \sum_{\substack{\ell=0 \\ \ell+n \text{ even}}}^{n} \sum_{m=-\ell}^{\ell} \mathcal{Z}_{n\ell}^m \, Z_{n\ell}^m(\boldsymbol{p}) \tag{15}$$

In [30], [19] and [31] the used Z-C moments follow this convention. And it is the one followed in this paper. However, in [29] Canterakis himself presents the Z-C moments with a different convention: the factor $3/4\pi$ was moved from the moments definition to the reconstruction formula.

ACKNOWLEDGMENTS

The authors would like to thank the valuable and constructive comments from the anonymous reviewers.

REFERENCES

[1] R. Mukundan and K. R. Ramakrishnan, *Moment Functions in Image Analysis - Theory and Applications*. World Scientific Pub Co., 1998.
[2] J. Flusser, "Moment invariants in image analysis," *Proceedings of World Academy of Science, Engineering and Technology*, vol. 11, pp. 196–201, February 2006.
[3] L. Yang, F. Albregtsen, and T. Taxt, "Fast computation of three-dimensional geometric moments using a discrete divergence theorem and a generalization to higher dimensions," *Graphical Models and Image Processing*, vol. 59, no. 2, pp. 97–108, 1997.
[4] S. X. Liao and M. Pawlak, "On image analysis by moments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 3, p. 254266, 1996.
[5] ——, "On the accuracy of Zernike moments for image analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, p. 13581364, 1998.
[6] S. Rodtook and S. S. Makhanov, "Numerical experiments on the accuracy of rotation moments invariants," *Image and Vision Computing*, vol. 23, p. 577586, 2005.
[7] S. Lien and J. T. Kajiya, "Symbolic method for calculating the integral properties of arbitrary nonconvex polyhedra," *IEEE Computer Graphics and Applications*, vol. 4, no. 10, pp. 35–41, Oct. 1984.
[8] C. Cattani and A. Paoluzzi, "Boundary integration over linear polyhedra," *Comput. Aided Des.*, vol. 22, no. 2, pp. 130–135, 1990.
[9] F. Bernardini, "Integration of polynomials over n-dimensional polyhedra," *Comput. Aided Des.*, vol. 23, no. 1, pp. 51–58, 1991.
[10] S. A. Sheynin and A. V. Tuzikov, "Explicit formulae for polyhedra moments," *Pattern Recognition Letters*, vol. 22, pp. 1103–1109, 2001.
[11] A. V. Tuzikov, S. A. Sheynin, and P. V. Vasiliev, "Computation of volume and surface body moments," *Pattern Recognition*, vol. 36, pp. 2521–2529, 2003.
[12] G. C. Best, "Helpful formulas for integrating polynomials in three dimensions," *Mathematics of Computation*, vol. 18, no. 86, pp. 310–312, 1964. [Online]. Available: http://www.jstor.org/stable/2003308
[13] A. DiCarlo and A. Paoluzzi, "Fast computation of inertia through affinely extended euler tensor." *Computer-Aided Design*, vol. 38, no. 11, pp. 1145–1153, 2006.
[14] H. G. Timmer and J. M. Stern, "Computation of global geometric properties of solid objects," *Computer-Aided Design*, vol. 12, no. 6, pp. 301–304, 1980.
[15] J. Liggett, "Exact formulae for areas, volumes and moments of polygons and polyhedra," *Commun. Appl. Numer. Methods*, vol. 4, no. 6, p. 815820, 1988.
[16] B. Li, "The moment calculation of polyhedra," *Pattern Recognition*, vol. 26, no. 8, pp. 1229–1233, 1993.
[17] B. Mirtich, "Fast and accurate computation of polyhedral mass properties," *journal of graphics, gpu, and game tools*, vol. 1, no. 2, pp. 31–50, 1996.
[18] H. T. Rathod and H. S. G. Rao, "Integration of polynomials over linear polyhedra in euclidean three-dimensional space," *Computer Methods in Applied Mechanics and Engineering*, vol. 126, no. 3-4, pp. 373 – 392, 1995.
[19] R. Millan, L. Dempere-Marco, J. Pozo, J. Cebral, and A. Frangi, "Morphological characterization of intracranial aneurysms using 3-D moment invariants," *Medical Imaging, IEEE Transactions on*, vol. 26, no. 9, pp. 1270–1282, Sept. 2007.
[20] P. C. Hammer, O. P. Marlowe, and A. H. Stroud, "Numerical integration over simplexes and cones," *Math. Tables Aids Comp.*, vol. 10, no. 55, pp. 130–137, 1956.
[21] A. H. Stroud, "Approximate integration formulas of degree 3 for simplexes," *Mathematics of Computation*, vol. 18, no. 88, pp. 590–597, 1964.
[22] P. Hillion, "Numerical integration on a tetrahedron," *Calcolo*, vol. 18, no. 2, pp. 117–130, 1981.
[23] M. Gellert and R. Harbord, "Moderate degree cubature formulas for 3-D tetrahedral finite-element approximations," *Communications in Applied Numerical Methods*, vol. 7, no. 6, pp. 487–495, 1991.
[24] H. Rathod, K. Nagaraja, and B. Venkatesudu, "Numerical integration of some functions over an arbitrary linear tetrahedra in euclidean three-dimensional space," *Applied Mathematics and Computation*, vol. 191, no. 2, pp. 397 – 409, 2007.
[25] C. Gonzalez-Ochoa, S. McCammon, and J. Peters, "Computing moments of objects enclosed by piecewise polynomial surfaces," *ACM Trans. Graph.*, vol. 17, no. 3, pp. 143–157, 1998.

[26] C.-H. Teh and R. T. Chin, "On digital approximation of moment invariants," *Computer Vision, Graphics, and Image Processing*, vol. 33, pp. 318–326, 1986.
[27] A. Khotanzad and Y. H. Hong, "Invariant image recognition by Zernike moments," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 5, pp. 489–497, 1990.
[28] N. Canterakis, "Fast 3D Zernike moments and invariants," Albert-Ludwigs-Universität Freiburg, Institut für Informatik, Tech. Rep. 5/97, 1997. [Online]. Available: citeseer.ist.psu.edu/canterakis97fast.html
[29] ——, "3D Zernike moments and Zernike affine invariants for 3D image analysis and recognition," in *11th Scandinavian Conf. on Image Analysis*, 1999, pp. 85–93.
[30] M. Novotni and R. Klein, "Shape retrieval using 3D Zernike descriptors," *Computed-Aided Design*, vol. 36, pp. 1047–1062, Jan. 2004.
[31] W. Qiuting and Y. Bing, "3D terrain matching algorithm and performance analysis based on 3D Zernike moments," *Computer Science and Software Engineering, 2008 International Conference on*, vol. 6, pp. 73–76, Dec. 2008.
[32] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Shape Modeling International*, 2004, pp. 167–178.
[33] M. Attene and B. Falcidieno, "ReMESH: An interactive environment to edit and repair triangle meshes," in *Proceedings of Shape Modeling Internationl (SMI'06)*. IEEE Computer Society Press, 2006, pp. 271–276.

José María Pozo graduated in Theoretical Physics (1996) at the Universitat de Barcelona (UB) and obtained his PhD in Physics (2002) under the supervision of Josep Manel Parra at the Departament de Física Fonamental of the UB. His thesis focused on the applications of the geometric Clifford algebra to general relativity. In June 2004 he joined the Department Systèmes de Référence Temps Espace (SYRTE) at the Observatoire de Paris with a postdoctoral fellowship from the spanish Ministerio de Educación y Ciencia. He worked under the supervision of Bartolomé Coll on the development of the theory of relativistic positioning systems. In July 2006, he joined the Center for Computational Imaging and Simulation Technologies in Biomedicine (CISTIB) at Universitat Pompeu Fabra as a postdoctoral reseacher, where he works on the three-dimensional morphological characterization of cerebral aneurysms and flow characterization in vasculatures. He is a visiting professor since September 2007.

Maria-Cruz Villa-Uriol graduated in Telecommunications (1994) and Electronics Engineering (1997) at La Salle School of Engineering in Barcelona, Spain. In July 2000 she was awarded the Balsells Graduate fellowship and in 2005 received her PhD in Computer Engineering from the University of California, Irvine, on video-based human motion capture. She worked as a postdoctoral researcher in the Calit2 Center of GRAVITY at the University of California, Irvine. Since March 2006 she is a postdoctoral researcher at the Center for Computational Imaging & Simulation Technologies in Biomedicine (CISTIB) at the Universitat Pompeu Fabra (Barcelona, Spain). She has actively contributed to several Spanish and EC FP6 projects, among them @neurIST and CDTEAM. Her research interests are biomedical computational imaging and modeling, computer graphics, and motion kinematic analysis and synthesis. Currently, she works on the geometrical analysis and three-dimensional dynamic reconstruction of cerebral vasculature, with an emphasis in their translation into clinical practice. She is a member of the IEEE.

Alejandro Frangi obtained his BSc/MSc degree in Telecommunications Engineering from the Technical University of Catalonia (Barcelona) in 1996. He subsequently carried out research on electrical impedance tomography for image reconstruction and noise characterization at the same institution under a CIRIT grant. In 1997 he obtained a grant from the Dutch Ministry of Economic Affairs to pursue his PhD at the Image Sciences Institute of the University Medical Center Utrecht on model-based cardiovascular image analysis. During this period he was visiting researcher at the Imperial College in London, UK, and in Philips Medical Systems BV, The Netherlands. Dr. Frangi is Associate Professor at Universitat Pompeu Fabra (UPF) and ICREA-Academia Researcher. He currently leads the Center for Computational Imaging & Simulation Technologies in Biomedicine at the UPF. His main research interests are in medical image computing, medical imaging and image-based computational physiology. Dr. Frangi has edited a book, published 3 editorial articles and over 55 journal papers in key international journals and more than 100 book chapters and international conference papers. Dr. Frangi received the IEEE Engineering in Medicine and Biology Early Career Award in 2006 and the Prizes for Knowledge Transfer (2008) in the Information and Communication Technologies domain and for Teaching Excellence (2008) by the Social Council of the Universitat Pompeu Fabra.