Accurate Detection of Symmetries in 3D Shapes

Aurélien Martinet, Cyril Soler, Nicolas Holzschuch, François X. Sillion

Accurate Detection of Symmetries in 3D Shapes

AURÉLIEN MARTINET, CYRIL SOLER, NICOLAS HOLZSCHUCH and FRANÇOIS X. SILLION
ARTIS, INRIA Rhône-Alpes

We propose an automatic method for finding symmetries of 3D shapes, that is, isometric transforms which leave a shape globally unchanged. These symmetries are deterministically found through the use of an intermediate quantity: the generalized moments. By examining the extrema and spherical harmonic coefficients of these moments, we recover the parameters of the symmetries of the shape. The computation for large composite models is made efficient by using this information in an incremental algorithm capable of recovering the symmetries of a whole shape using the symmetries of its subparts. Applications of this work range from coherent remeshing of geometry with respect to the symmetries of a shape to geometric compression, intelligent mesh editing, and automatic instantiation.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid and object representations*

General Terms: Algorithms

1. INTRODUCTION

Many shapes and geometrical models exhibit *symmetries*: isometric transforms that leave the shape globally unchanged. Using symmetries, one can manipulate models more efficiently through coherent remeshing or intelligent mesh editing programs. Other potential applications include model compression, consistent texture-mapping, model completion, and automatic instantiation.

The symmetries of a model are sometimes made available by the creator of the model and represented explicitly in the file format the model is expressed in. Usually, however, this is not the case, and automatic translations between file formats commonly result in the loss of this information. For scanned models, symmetry information is also missing by nature.

In this article, we present an algorithm that automatically retrieves symmetries in a geometrical model. Our algorithm is independent of the tesselation of the model; in particular, it does not assume that the model has been tesselated in a manner consistent with the symmetries we attempt to identify, and it works well on noisy objects such as scanned models. Our algorithm uses a new tool, the *generalized moment* functions. Rather than computing these functions explicitly, we directly compute their spherical harmonic coefficients, using a fast and accurate technique. The extrema of these functions and their spherical harmonic coefficients enable us to deterministically recover the symmetries of a shape.

For composite shapes, that is, shapes built by assembling simpler structures, we optimize the computation by applying the first algorithm to the subparts, then iteratively building the set of symmetries of

Authors' addresses: A. Martinet, C. Soler, N. Holzschuch, F. X. Sillion, ARTIS, INRIA Rhône-Alpes, Saint Ismier, France; email: Aurelien. Martinet@imag.fr.

the composite shape, taking into account both the relative positions of the subparts and their relative orientations.

We envision many applications for our work, including geometric compression, consistent mesh editing, and automatic instantiation.

This article is organized as follows. In the following section, we review previous work on identifying geometric symmetries on 2D and 3D shapes. Then in Section 3, we present an overview of the symmetry-detection problem and the quantities used in our algorithms. In Section 4, we introduce the generalized moments and our method to compute them efficiently; in Section 5, we present our algorithm for identifying symmetries of a shape. The extension of this algorithm to composite shapes is then presented in Section 6., Finally, in Section 7, we show various applications of our algorithm.

2. RELATED WORK

Early approaches to symmetry detection focused on the 2D problem. Attalah [1985], Wolter et al. [1985] and Highnam [1985] present methods to reduce the 2D-symmetry detection problem to a 1D pattern matching problem for which efficient solution are known [Knuth et al. 1977]. Their algorithms efficiently detect all possible symmetries in a point set but are highly sensitive to noise.

Identifying symmetries for 3D models is much more complex, and little research on this subject has been published. Jiang and Bunke [1991] present a symmetry-detection method, restricted to rotational symmetry, based on a scheme called generate and test, first finding hypothetical symmetry axes, then verifying these assumptions. This method is based on a graph representation of a solid model and uses graph theory. The dependency between this graph representation and the mapping between points makes their method highly dependent on the topology of the mesh and sensitive to small modifications of the object geometry. Brass and Knauer [2004] provide a model for general 3D objects and give an algorithm to test congruence or symmetry for these objects. Their approach is capable of retrieving symmetry groups of an arbitrary shape but is also topology-dependent since it relies on a mapping between points of the model. Starting from an octree representation, Minovic et al. [1993] describe an algorithm based on octree traversal to identify symmetries of a 3D object. Their algorithm relies on PCA to find the candidate axis; PCA, however, fails to identify axes for a large class of objects, including highly symmetric objects such as regular solids.

All these methods try to find strict symmetries for 3D models. As a consequence, they are sensitive to noise and data imperfections. Zabrodsky et al. [1995] define a measure of symmetry for nonperfect models, defined as the minimum amount of work required to transform a shape into a symmetric shape. This method relies on the ability to first establish correspondence between points, a very restrictive precondition.

Sun and Sherrah [1997] use the *Extended Gaussian Image* to identify symmetries by looking at correlations in the Gaussian image. As in Minovic et al. [1993], they rely on PCA to identify potential axes of symmetry, thus possibly failing on highly symmetric objects. More recently, Kazhdan et al. [2004] introduced the *symmetry descriptors*, a collection of spherical functions that describe the measure of a model's rotational and reflective symmetry with respect to every axis passing through the center of mass. Their method provides good results in the shape identification but involves a surface integration for each sampled direction; this surface integration is carried on a voxel grid. Using the symmetry descriptors to identify symmetries requires an accurate sampling in all directions, making their algorithm very costly for an accurate set of results. In contrast, our algorithm only computes a deterministic small number of surface integrals, which are performed on the shape itself, and still provides very accurate results. Effective complexity comparisons will be given in Section 8.

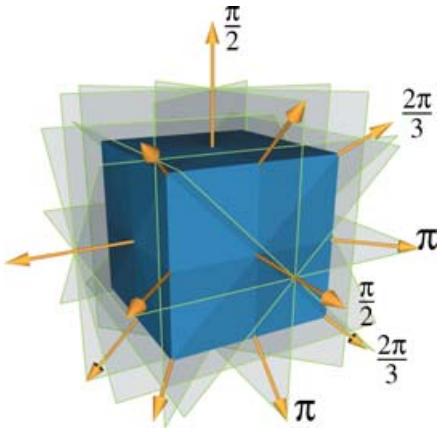


Fig. 1. Mirror symmetries and rotational symmetries found by our algorithm for a cube (for clarity, not all elements are represented).

3. OVERVIEW

Considering a surface $\mathcal{S}$, the *symmetries* of $\mathcal{S}$ are the isometric transforms which map $\mathcal{S}$ onto itself, in any coordinate system centered on its center of gravity. Symmetries of a shape form a group for the law of function composition with identity as its neutral element. For a given shape, the study of such a group relates to the domain of mathematical crystallography [Prince 2004].

The group of the cube, for instance, contains 48 elements (see Figure 1): the identity, eight 3−fold rotations around 4 possible axes, nine 4−fold rotations around 3 possible axes, six 2−fold rotations around 6 possible axes, nine mirror-symmetries, and fifteen other elements obtained by composing rotations and mirror symmetries.

Studying the group of isometries in $\mathbb{R}^3$ shows that, for a given isometry I, there always exists an orthonormal basis $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ into which the matrix of I takes the following form:

$$I(\lambda, \alpha) = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \quad \text{with} \quad \begin{cases} \alpha \in [0, 2\pi[\\ \lambda = \pm 1 \end{cases}$$

As suggested by the example of the cube, this corresponds to 3 different classes of isometries: rotations, mirror symmetries, and their composition, depending whether λ is positive and/or $\alpha = 0 (\mathrm{mod}\ \pi)$. Finding a symmetry of a shape thus resolves into finding a vector $\boldsymbol{X}$ — which we call the *axis* of the isometry — and an angle α — which we call the *angle* of the isometry — such that $I(\lambda, \alpha)$ maps this shape onto itself.

However, finding all symmetries of a shape is much more difficult than simply checking whether a given transform actually is a symmetry. In particular, the naive approach that would consist of checking as many sampled values of $(\boldsymbol{X}, \lambda, \alpha)$ as possible to find a symmetry is far too costly. We thus need a deterministic method for finding good candidates.

Our approach to finding symmetries is to use intermediate functions, which set of symmetries is a superset of the set of symmetries of the shape itself, but for which computing the symmetries is much easier. By examining these functions, we will derive in Section 5 a deterministic algorithm which finds a finite number of possible candidates for $\boldsymbol{X}$, λ, and α. Because some unwanted triplets of values will appear during the process, these candidates are then checked back on the original shape. Choosing a family of functions which fulfill these requirements is easy. More difficult is the

task of finding such functions for which computing the symmetries can be done both accurately and efficiently.

Inspired by the work on principal component analysis [Minovic et al. 1993], we introduce the *generalized moment* functions of the shape for this purpose. These functions will be the topic of Section 4. These functions, indeed, have the same symmetries as the shape itself plus a small number of extra candidates. Furthermore, we propose an elegant framework based on spherical harmonics to accurately and efficiently find their symmetries.

A second contribution of this article is to extend the proposed algorithm into a constructive algorithm which separately computes the symmetries of subcomponents of an object—using the first method—, and then associates this information to compute symmetries of the whole composite shape. This constructive algorithm proves to be more accurate in some situations and more efficient when it is possible to decompose an object according to its symmetries. It is presented in Section 6.

4. GENERALIZED MOMENTS

In this section, we introduce a new class of functions: the generalized moments of a shape. We then show that these functions have at least the same symmetries as the shape itself and that their own symmetries can be computed in a very efficient way.

4.1 Definition

For a surface $\mathcal{S}$ in a 3-dimensional domain, we define its *generalized moment* of order $2p$ in direction $\boldsymbol{\omega}$ by

$$\mathcal{M}^{2p}(\boldsymbol{\omega}) = \int_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{s} \times \boldsymbol{\omega}\|^{2p} \, \mathrm{d}\boldsymbol{s}. \tag{1}$$

In this definition, $\boldsymbol{s}$ is a vector which links the center of gravity of the shape (placed at the origin) to a point on the surface, and $\mathrm{d}\boldsymbol{s}$ is thus an infinitesimal surface element. $\mathcal{M}^{2p}$ itself is a directional function.

It should be noted that, considering $\mathcal{S}$ to have some thickness dt, the expression $\mathcal{M}^2(\boldsymbol{\omega})dt$ (i.e., the generalized moment of order 2) corresponds to the moment of inertia of the thin shell $\mathcal{S}$ along $\boldsymbol{\omega}$, hence the name of these functions. Furthermore, the choice of an even exponent and a cross-product will lead to very interesting properties.

4.2 Shape Symmetries and Moments

Symmetry properties of a shape translate into symmetry properties of its moment functions. We now introduce a theorem that we will be rely on (see proof in Appendix):

THEOREM 1. *Any symmetry I of a shape $\mathcal{S}$ also is a symmetry of all its $\mathcal{M}^{2p}$ moment functions:*

$$I(\mathcal{S}) = \mathcal{S} \quad \Rightarrow \quad \forall\boldsymbol{\omega} \;\; \mathcal{M}^{2p}(I(\boldsymbol{\omega})) = \mathcal{M}^{2p}(\boldsymbol{\omega}).$$

Furthemore, if $\mathcal{M}^{2p}$ has a symmetry I with axis $\boldsymbol{\omega}$, then the gradient of $\mathcal{M}^{2p}$ is null at $\boldsymbol{\omega}$:

$$\forall\boldsymbol{\omega} \quad \mathcal{M}^{2p}(I(\boldsymbol{\omega})) = \mathcal{M}^{2p}(\boldsymbol{\omega}) \quad \Rightarrow \quad (\boldsymbol{\nabla}\mathcal{M}^{2p})(\boldsymbol{\omega}) = 0.$$

This theorem implies that the axes of the symmetries of a shape are to be found in the intersection of the sets of directions which zero the gradients of each of its moment functions. The properties are not reciprocal, however. Once the directions of the zeros of the gradients of the moment functions have been found, they must be checked on the shape itself to eliminate false positives.

4.3 Efficient Computation

At first sight, looking for the zeros of the gradient of the moment functions requires precise and dense sampling of these functions which would be very costly using their integral form of Equation (1). We thus present an efficient method to compute the generalized even moment functions of a shape, using spherical harmonics. In particular, we can accurately compute the spherical harmonic coefficients of the moment functions without sampling these functions. The search for zeros in the gradient will then be performed efficiently on the spherical harmonic decomposition itself.

Spherical Harmonics. We use real-valued spherical harmonics [Hobson 1931] to represent directional functions. Real spherical harmonics are defined, for integers $l \geq 0$ and $-l \leq m \leq l$, by:

$$Y_l^m(\theta,\varphi) = \begin{cases} \sqrt{2}\, N_l^m P_l^m(cos\theta)\cos(m\varphi) & \text{for} \quad 0 < m \leq l \\ N_l^m P_l^0 \ (cos\theta) & \text{for} \qquad m = 0 \\ \sqrt{2}\, N_l^m P_l^{-m}(cos\theta)\sin(m\varphi) & \text{for} -l \leq m < 0 \end{cases}$$

where P_l^m are the associated Legendre polynomials; the normalization constants N_l^m are such that the spherical harmonics form an orthonormal set of functions for the scalar product:

$$< f, g >= \int_{\|\omega\|=1} f(\omega) g(\omega)\, \mathrm{d}\omega.$$

This corresponds to choosing:

$$N_l^m = \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}}.$$

We will use the following very powerful property of spherical harmonics. Any spherical harmonic of degree l can be expressed in a rotated coordinate system using harmonics of same degree and coefficients depending on the rotation R:

$$Y_l^m \circ R = \sum_{-l \leq m' \leq l} D_l^{m,m'}(R) Y_l^{m'}. \tag{2}$$

Any combination of spherical harmonics of degree less than l can therefore be expressed in a rotated coordinate system, using spherical harmonics of degree less than l, without loss of information. Coefficients $D_l^{m,m'}(R)$ can efficiently be obtained using recurrence formula [Ivanic and Ruedenberg 1996] or directly computed [Ramamoorthi and Hanrahan 2004].

Computation of Moment Functions. As defined by Equation (1), the $2p-$moment function of a shape $\mathcal{S}$ is expressed as:

$$\begin{aligned} \mathcal{M}^{2p}(\boldsymbol{\omega}) &= \int_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{s}\times\boldsymbol{\omega}\|^{2p}\, \mathrm{d}\boldsymbol{s} \\ &= \int_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{s}\|^{2p} \sin^{2p}\beta\, \mathrm{d}\boldsymbol{s} \end{aligned}$$

In this expression, β is the angle between s and ω.

Function $\beta \mapsto \sin^k \beta$ has angular dependence on β only and therefore decomposes into zonal harmonics (i.e., harmonics Y_l^m for which $m = 0$). Performing the calculation shows that, when k is even, the decomposition is finite. Setting $k = 2p$, we obtain :

$$\sin^{2p}\beta = \sum_{l=0}^{p} S_p^l Y_{2l}^0(\beta, .)$$

with:

$$S_p^l = \frac{\sqrt{(4l+1)\pi}}{2^{2l}} \sum_{k=l}^{2l} (-1)^k \frac{2^{2p+1} p!(2k)!(p+k-l)!}{(2(p+k-l)+1)!(k-l)!k!(2l-k)!} \ . \tag{3}$$

For the sake of completeness, we provide the corresponding derivation and the proof of the finite decomposition in the appendix section of this article.

Let R_s be a rotation which maps z, unit vector along $z-$axis, to s. Using Equation (2) for rotating the Y_{2l}^0 zonal harmonics, we have :

$$\sin^{2p}\beta = \sum_{l=0}^{p} S_p^l \sum_{m=-2l}^{2l} D_{2l}^{0,m}(R_s) Y_{2l}^m(\omega).$$

And finally:

$$\mathcal{M}^{2p}(\omega) = \sum_{l=0}^{p} \sum_{m=-2l}^{2l} C_{2l,m}^{2p} Y_{2l}^m(\omega), \tag{4}$$

using

$$C_{2l,m}^{2p} = S_p^l \int_{s\in\mathcal{S}} \|s\|^{2p} D_{2l}^{0,m}(R_s)\, \mathrm{d}s. \tag{5}$$

Equation (4) says that $\mathcal{M}^{2p}$ decomposes into a finite number of spherical harmonics, and Equation (5) allows us to directly compute the coefficients. The cost of computing $\mathcal{M}^{2p}$ is therefore $(p+1)(2p+1)$ surface integrals (one integral per even order of harmonic, up to order $2p$). This is much cheaper than the alternative method of computing the scalar product of $\mathcal{M}^{2p}$ as defined by Equation (1) with each spherical harmonic basis function: this would indeed require many evaluations of $\mathcal{M}^{2p}$, which itself is defined as a surface integral. Furthermore, numerical accuracy is only a concern when computing the $C_{2k,p}^m$ coefficients, and we can now compute both $\mathcal{M}^{2p}$ and its gradient analytically from Equation (4).

5. FINDING SYMMETRIES OF A SINGLE SHAPE

In this section, we present our algorithm for identifying symmetries of a shape seen as a single entity as opposed to the algorithm presented in the next section where the shape is considered as an aggregation of multiple subparts. For a given shape, we want to determine the axis $\boldsymbol{X}$ and the (λ, α) parameters of the potential isometries, using the generalized moment functions, and check the isometries found against the actual shape.

Central symmetries ($\lambda = -1$ and $\alpha = \pi$) form a specific case since, by construction, $\mathcal{M}^{2p}$ always has a central symmetry. Because central symmetries also do not require an axis, we treat this case directly while checking the other candidate symmetries on the shape itself in Section 5.3.

5.1 Determination of the Axis

As we saw in Section 4.2, the axis of isometries which let a shape globally unchanged also zero the gradient of the generalized even moments of this shape. We thus obtain a superset of them by solving for:

$$\nabla(\mathcal{M}^{2p})(\omega) = \mathbf{0}.$$

In a first step, we estimate a number of vectors which are close to the actual solutions by refining the sphere of directions starting from an icosahedron. In each face, the value of $\|\nabla(\mathcal{M}^{2p})(\omega)\|^2$ is examined

in several directions, and faces are sorted by order of the minimal value found. Only faces with small minimum values are refined recursively. The number of points to look at in each face, as well as the number of faces to keep at each depth level, are constant parameters of the algorithm.

In a second step, we perform a steepest descent minimization on $\|\nabla(\mathcal{M}^{2p})(\omega)\|^2$, starting from each of the candidates found during the first step. For this we need to evaluate the derivatives of $\|\nabla(\mathcal{M}^{2p})\|$ which we do using analytically computed second-order derivatives of the spherical harmonics along with Equation (4). The minimization converges in a few steps because starting positions are by nature very close to actual minima. This method has the double advantage that (1) the derivatives are very efficiently computed and (2) no approximation is contained into the calculation of the direction of the axis beyond the precision of the calculation of the $C^{2p}_{2l,m}$ coefficients.

During this process, multiple instances of the same direction can be found. We filter them out by estimating their relative distance. While nothing in theory prevents the first step from missing the area of attraction of a minimum, it works very well in the present context. Indeed, moment functions are very smooth, and shapes having two isometries with very close—yet different—axis are not common.

Finally, because all moment functions, whatever their order, must have an extremum in the direction of the axis of the symmetries of the shape, we compute such sets of directions for multiple moment functions (e.g., $\mathcal{M}^4$, $\mathcal{M}^6$ and $\mathcal{M}^8$) but keep only those which simultaneously zero the gradient of all these functions, which in practice leaves none or very few false positives to check for.

5.2 Determination of Rotation Parameters

After finding the zero directions for the gradient of the moment functions, we still need to find the parameters of the corresponding isometric transforms. This is done deterministically by studying the spherical harmonic coefficients of the moment functions themselves. We use the following properties.

PROPERTY 1. *A function has a mirror symmetry S_z around the $z = 0$ plane if and only if all its spherical harmonic coefficients for which $l + m$ is even are zero (i.e., it decomposes onto $z-$symmetric harmonics only). In the specific case of the moment functions:*

$$\forall \omega \quad \mathcal{M}^{2p}(\omega) = \mathcal{M}^{2p}(S_z\omega) \quad \Leftrightarrow \quad m \equiv 0(mod\ 2) \Rightarrow C^{2p}_{2l,m} = 0.$$

PROPERTY 2. *A function has a revolution symmetry around the z axis if and only if it decomposes onto zonal harmonics only, that is,*

$$\forall l \quad \forall m \quad m \neq 0 \quad \Rightarrow \quad C^m_l = 0.$$

PROPERTY 3. *A function is self-similar through a rotation R_α of angle α around z if and only if all its spherical harmonic coefficients C^m_l verify:*

$$\forall l \quad \forall m \quad C^m_l = \cos(m\alpha)C^m_l - \sin(m\alpha)C^{-m}_l. \tag{6}$$

Property 3 can be adapted to check if the function is self-similar through the composition of a rotation and a symmetry with the same axis (i.e., the case $\lambda = -1$ as defined in Section 3). In this case, the equation to be checked for is:

$$\forall l \quad \forall m \quad (-1)^{l+m}C^m_l = \cos(m\alpha)C^m_l - \sin(m\alpha)C^{-m}_l. \tag{7}$$

These properties are easily derived from the very expression of the spherical harmonic functions [Hobson 1931].

Before using these properties, the moment function must be expressed in a coordinate system where the z axis coincides with the previously found candidate axis. This is performed using the rotation formula in Equation (2). Then checking for Properties 1 and 2 is trivial provided that some tolerance is accepted on the equalities. Using Property 3 is more subtle; coefficients of the function are first examined by order of decreasing m. For $\lambda = 1$, for instance, when the first nonzero value of C_l^m is found, Equation (6) is solved by:

$$\tan\frac{m\alpha}{2} = \frac{C_l^{-m}}{C_l^m}, \quad \text{that is,} \quad \alpha = \frac{2}{m}\arctan\left(\frac{C_l^{-m}}{C_l^m}\right) + \frac{k\pi}{m},$$

then all the remaining coefficients are checked with the obtained values of α. If the test passes, then α is the angle of an existing rotation symmetry for the moment function. A very similar process is used to search for α when $\lambda = -1$.

The error tolerance used when checking for Properties 1, 2, and 3 can be considered as a way of detecting approximate symmetries on objects. We will show in the results section that symmetries can indeed be detected on noisy data such as scanned models.

5.3 Filtering Results

The condition extracted from Theroem 1 is a necessary condition only. To avoid false positives, the directions and rotation angles obtained from the moment functions must therefore be verified on the shape itself. We do this using a *symmetry measure* inspired by the work of Zabrodsky et al. [1995]. Let $\mathcal{S}$ and $\mathcal{R}$ be two tessellated shapes. Let $V_\mathcal{S}$ and $V_\mathcal{R}$ be the mesh vertices of $\mathcal{S}$ and $\mathcal{R}$. We define the *measure* d_M between $\mathcal{S}$ and $\mathcal{R}$ by:

$$d_M(\mathcal{S},\mathcal{R}) = \max_{p\in V_\mathcal{S}}(\min_{q\in\mathcal{R}} \|p-q\|). \tag{8}$$

The *symmetric measure* $d_A(\mathcal{S})$ of a shape $\mathcal{S}$ with respect to a symmetry A is then defined by:

$$d_A(\mathcal{S}) = \max(d_M(\mathcal{S}, A\mathcal{S}), d_M(A\mathcal{S}, \mathcal{S})).$$

It should be noted that this definition is different from that of the *Hausdorff distance* since, in Equation (8), not all points of $\mathcal{S}$ are considered but only the mesh vertices, whereas all points of $\mathcal{R}$ are used. However, because $\mathcal{S}$ is polyhedral, $d_A(\mathcal{S}) = 0$ still implies that $A\mathcal{S} = \mathcal{S}$.

Computing d_A is costly, but fortunately we only compute it for a few choices of A which are the candidates we found at the previous step of the algorithm. This computation is much cheaper than computing a full symmetry descriptor [Kazhdan et al. 2004] for a sufficient number of directions to reach the precision of our symmetry detection algorithm.

5.4 Results

Complete Example. The whole process is illustrated in Figure 2. Starting from the original object (a), the moment functions of orders 4, 6, and 8 are computed (see, e.g., $\mathcal{M}^8$ in (b)). The gradients of these moments are then computed analytically (c) and used for finding the directions of the minima. The unfiltered set of directions contains 7 directions among which only 3 are common extrema of $\mathcal{M}^4$, $\mathcal{M}^6$, and $\mathcal{M}^8$. This set of 3 directions ($\boldsymbol{D}_1$,$\boldsymbol{D}_2$, and $\boldsymbol{D}_3$) must contain the axes of the symmetries of the shape. After checking the symmetry axis and parameters on the actual shape, $\boldsymbol{D}_1$ is revealed as the axis of a 2-fold symmetry which is the composition of the two remaining mirror symmetries of axes $\boldsymbol{D}_2$ and $\boldsymbol{D}_3$.

The example of the cube, shown in Figure 1, illustrates the extraction of rotations and mirror symmetries. Experiments have shown that our method finds all 48 symmetries whatever the coordinate system the cube is expressed in originaly.

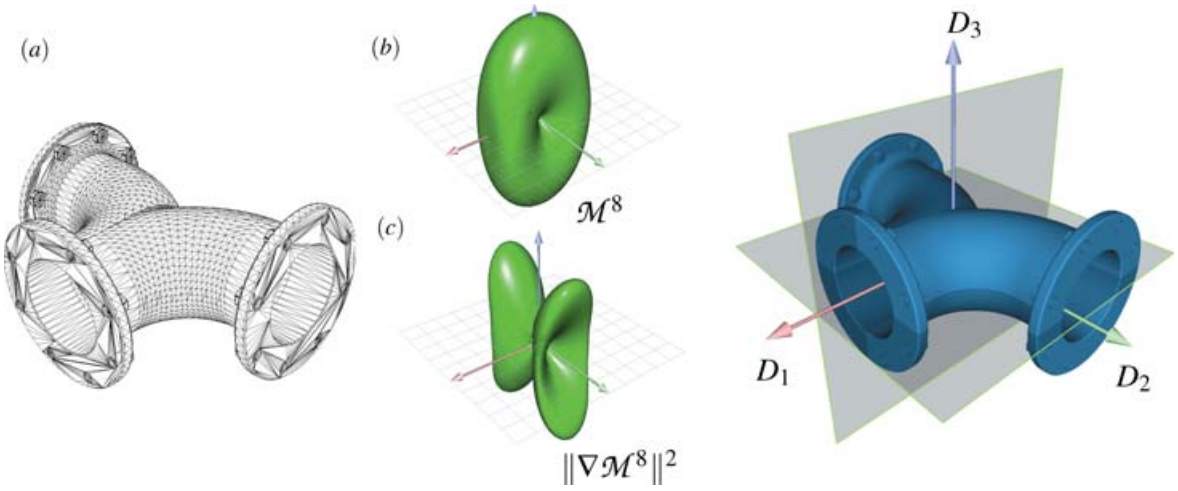


Fig. 2. Extraction of symmetries for a single shape. Starting from the original shape (*a*), generalized moments (*b*) and their gradients (*c*) are computed. The set of their common extrema directions contains the axes of the symmetries of the shape, depicted at right. Here, both mirror symmetries have been found as well as the 2-fold rotational symmetry. Note that the original shape is neither convex nor star-shaped and that the mesh is not consistent with the symmetries of the geometry.

Fig. 3. View of the three 3D models used in the robustness tests presented in Figure 4 shown with their symmetries. For the sake of clarity, we chose models with only one symmetry each.

Robustness Tests. We now study the sensitivity of our method to small perturbations of the 3D model in two different ways.

(1) *Noise.* We randomly perturb each vertex of each polygon independently in the original model by a fraction of the longest length of the model's bounding box.

(2) *Delete.* We randomly delete a small number of polygons in the model.

We use a set of three models to test the robustness of our method. These model as well as their symmetry are shown in Figure 3. For the sake of clarity, we use objects with only one symmetry.

In order to test the robustness of the method, we progressively increase the magnitude of the noise and let the algorithm automatically detect the symmetry. In our robustness tests, we consider shapes as single entities and use the first algorithm presented in Section 5 to detect these symmetries. To evaluate

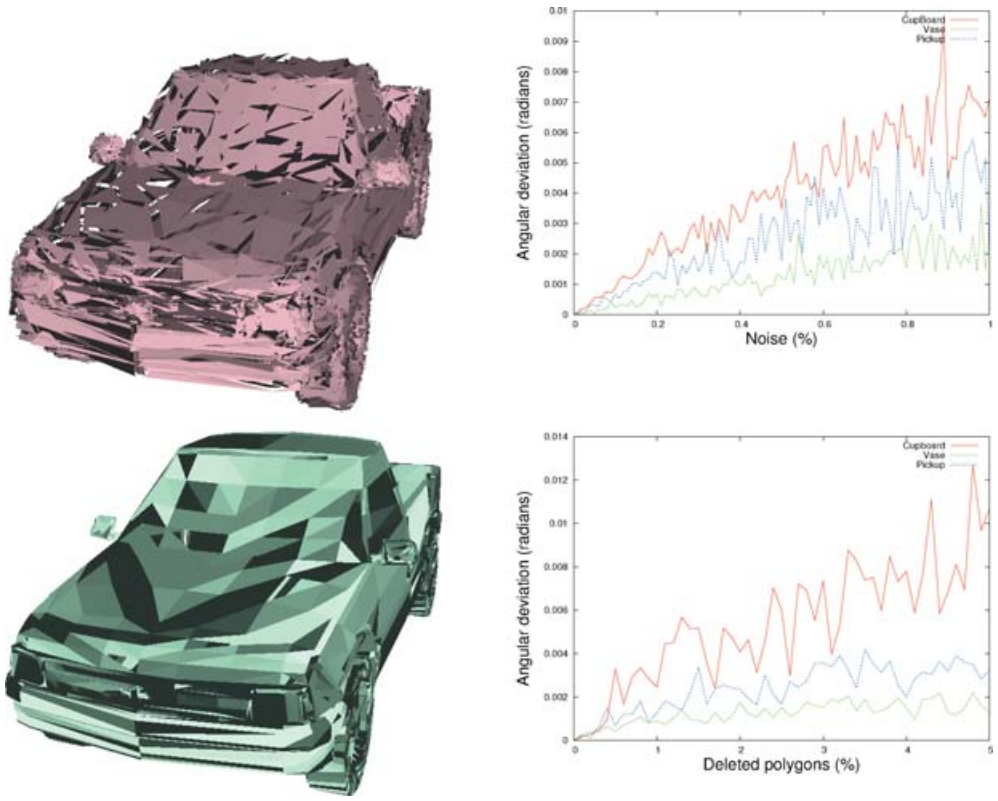


Fig. 4. We test the sensitivity of the method to noise by progressively increasing noise magnitude and letting the algorithm detect the symmetry for each of our three test models. We evaluate the accuracy of the results by computing the angular deviation between the axis found and the axis of the symmetry of the original model. *Top row:* We perturb each vertex of each polygon independently by a fraction of the longest length of the bounding box on each of the three test models. The left figure shows a noisy pick-up model with a noise magnitude of 1% and the right figure shows angular deviation evolution for the three models for a magnitude ranging from 0% to 1%. *Bottom row:* We randomly delete polygons of the models. The left figure shows a noisy pick-up obtained by deleting 5% of the polygons and the right figure shows angular deviation evolution by deleting 0% to 5% of the polygons of the three models. As can be seen from the curve, for small variations of the models, our method has approximatively linear dependency regarding noise and delivers high-quality results even for nonperfect symmetries.

the reliability of the results, we compute the angular deviation between the found axis of symmety and the real one, that is, computed with no noise. In our experiments, noise magnitude varies from 0 to 1% of the longest length of the model's bounding box, and the number of deleted polygons ranges from 0 to 5% of the total number of polygons in the model (see Figure 4).

The results of these experiments show that, for small variations, our method has approximatively linear dependency regarding noise and delivers high-quality results even for nonperfect symmetries. These statistical results can also be used to derive an upper bound on the mean angular error obtained as a function of the noise in the model.

5.4.1 *Application to Scanned Models.* We present in Figure 5 examples of applying the single-shape algorithm to scanned models, retreived from a Web database and used as is (see `http://shapes.aim-at-shape.net`). Our algorithm perfectly detects all the parameters of candidate symmetries for all these

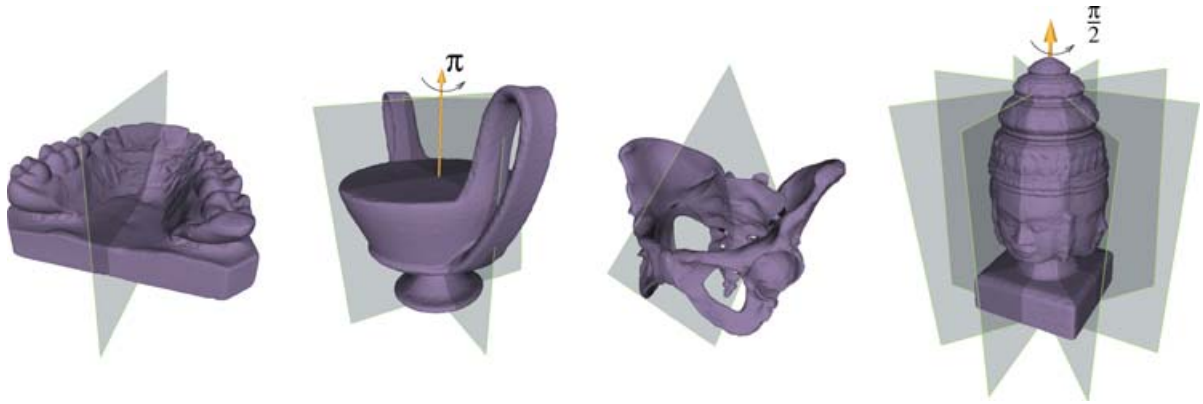


Fig. 5. Our algorithm perfectly detects approximate symmetries of scanned models. Detecting these symmetries requires relaxing the constraints when checking candidate symmetries on the model. Please note that these scanned models are by nature neither axis-aligned nor tesselated according to their symmetries. This illustrates the fact that our algorithm does not depend on the coordinate system nor on the mesh of the objects.

Table I. Computation times (in seconds) for the Four Scanned Models Presented in Figure 5

Model	Teeth	Vase	Pelvis	Angkor statue
# polygons	233, 204	76, 334	50, 000	163, 054
Computing moments*	33.7	11.8	7.26	23.26
Finding parameters	0.4	0.6	0.4	0.7
Checking candidates	9.4	11.1	5	12.2
Total	43.5	23.5	12.66	36.16

*Global computation times for moments of order 2 to 8

shapes. When testing these symmetries, one should allow a large enough symmetry distance error (as defined in Section 5.3) because these models are by nature not perfectly symmetric.

5.5 Discussion

Because the $\mathcal{M}^{2p}$ functions are trigonometric polynomials on the sphere, they have a maximum number of strict extrema depending on p: the larger p is, the more $\mathcal{M}^{2p}$ is able to capture the information of a symmetry, that is, to have an extremum in the direction of its axis. But because all moment functions *must* have a null gradient in this direction (according to Theorem 1), these extrema are bound to become nonstrict extrema for small values of p, and $\mathcal{M}^{2p}$ is forced to be constant on a subdomain of nonnull dimension. Using the cube as an example in which case $\mathcal{M}^2$ is a constant function a trigonometric polynomial of order 2 can simply not have enough strict extrema to represent all 12 distinct directions of the symmetries of the cube.

In all the tests we conducted, however, using moments up to order 10 has never skipped any symmetry on any model. But it would still be interesting to know the exact maximum number of directions permitted by moments of a given order.

6. FINDING SYMMETRIES OF GROUPS OF OBJECTS

In Section 5, we have presented an algorithm for finding the symmetries of single shapes. In this section, we present a *constructive* algorithm which recovers the symmetries of a group of objects—which we call *tiles* to indicate that together they form a larger object—from the symmetries and positions of each separate tile.

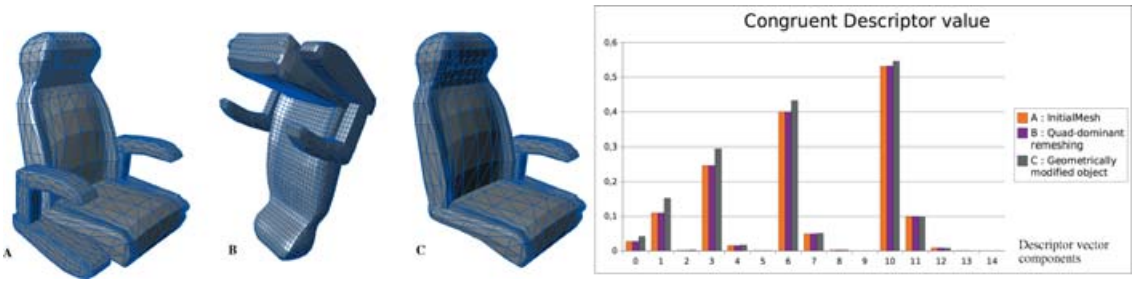


Fig. 6. This figure illustrates the reliability of our congruency descriptor (as defined by Equation (9)). Two identical objects meshed differently and expressed in two different coordinate systems (A and B) have extremely close descriptor vectors, but a slightly different object (C) has a different descriptor. The graphics on the right shows each component of the three descriptors.

The constructive algorithm first computes (if necessary) the symmetries of all separate tiles using the single shape algorithm. Then it detects which tiles are similar up to an isometric transform and finds the transformations between similar tiles. Then it explores all one-to-one mappings between tiles, discarding mappings which do not correspond to a symmetry of the group of tiles as a whole.

Section 6.2 explains how we detect similar tiles and Section 6.3 details the algorithm which both explores tile-to-tile mappings and finds the associated symmetry for the whole set of tiles.

Because it is always possible to apply the algorithm presented in Section 5 to the group of tiles, considering it as a single complex shape, questioning the usefulness of the constructive method is legitimate. For this reason, we will explain in Section 6.5 in which situations the constructive method is preferable to the algorithm for single shapes; but let us first explain the method itself.

6.1 Computing the Symmetries of Each Tile

If not available, the symmetries of each tile are computed using the algorithm presented in Section 5. When assembling known objects together, the economy of this computation can, of course, be performed by simply computing the symmetries of one instance for each class of different tiles.

6.2 Detecting Tiles Congruency

In this subsection, we introduce a shape descriptor suitable for detecting whether two shapes are identical up to an—unknown—isometry. We will use this tool for classifying tiles before trying to find a mapping of a composite object onto itself.

Let $\mathcal{S}$ be a shape and $C^{2p}_{2l,m}$ the spherical harmonic coefficients of its generalized even moment functions $\mathcal{M}^{2p}$ up to an order p. Our shape descriptor is defined as the $p(p+1)/2$-vector obtained by packing together the frequency energy of the spherical harmonic decomposition of all moments of $\mathcal{S}$ up to order p:

$$D_{2p} = \left[d^0_0, d^2_0, d^2_2, \ldots, d^{2p}_0, d^{2p}_2 \ldots d^{2p}_{2p}\right] \tag{9}$$

with

$$d^{2k}_{2l} = \sum_{-2l \le m \le 2l} \left(C^{2k}_{2l,m}\right)^2 \tag{10}$$

(See Figure 6). It has been shown by Kazhdan et al. [2003] that d^k_l, as defined in Equation (10), does not depend on the coordinate system the spherical harmonic decomposition is expressed in. This means that each d^{2p}_{2l}, and therefore D_{2p} itself, is not modified by isometric transforms of the shape. Mirror

Table II. Percentage of Tiles Matched by our Shape Descriptor That Are Effectively Identical For Our Test Scenes

Max order	39,557 Polygons 851 Tiles	182,224 Polygons 480 Tiles	515,977 Polygons 5,700 Tiles
2	92.1%	43.9%	92.3%
4	100%	78.0%	100%
6	100%	92.2%	100%
8	100%	100%	100%

Fig. 7. Scenes used for testing the object congruency descriptor. In each scene, the descriptor has been used to detect objects with similar geometry (but possibly different meshes) up to a rigid transform. Objects found to be congruent are displayed with the same color.

symmetries do not affect d_{2l}^{2p} either since they only change the sign of the coefficient for some harmonics in a coordinate system aligned with the axis.

Two tiles A and B are considered to be similar up to an isometric transform, at a precision ε, when:

$$\|D_{2p}(A) - D_{2p}(B)\| < \varepsilon.$$

Theoretically, this shape descriptor can produce false positives, that is, tiles that are not congruent but have the same descriptor, but it can not produce false negatives because of its deterministic nature. Our experiments have shown that using moments up to order 6 produces a sufficiently discriminant shape descriptor on all test scenes. This is illustrated in Table II where we present the average precision value, that is, the percentage of matched tiles that are actually identical up to an isometric transform, for a set of architectural scenes (Figure 7).

By definition, congruent tiles should have the same set of symmetries, possibly expressed in different coordinate systems. Since we know the symmetries of each of the tiles, we introduce this constraint, thereby increasing the discriminating power of our shape descriptor as shown in Table III.

6.3 Algorithm for Assembled Objects

6.3.1 *Overview.* Once we have determined all classes of congruent tiles, the algorithm examines all the one-to-one mappings of the set of all tiles onto itself which map each tile onto a similar tile. For each one-to-one mapping found, it determines the isometric transforms which are simultaneously compatible with each tile and its symmetries.

The algorithm works recursively: at the beginning of each recursion step, we have extracted two subsets of tiles, $\mathcal{H}_1$ and $\mathcal{H}_2$, of the composite shape $\mathcal{S}$, and we have computed the set of all possible isometric transforms that globally transform $\mathcal{H}_1$ into $\mathcal{H}_2$. Then, taking two new similar tiles, $\mathcal{S}_1 \in \mathcal{S} \setminus \mathcal{H}_1$

Table III. Percentage of Tiles Matched By Our Shape Descriptor That Are Effectively Identical Using the Added Constraint That Identical Tiles Must Have the Same Set of Symmetries Up to a Rigid Transform

Max order	39,557 Polygons 851 Tiles	182,224 Polygons 480 Tiles	515,977 Polygons 5,700 Tiles
2	95.6%	73.4%	97%
4	100%	96.0%	100%
6	100%	100%	100%
8	100%	100%	100%

and $\mathcal{S}_2 \in \mathcal{S} \setminus \mathcal{H}_2$, we restrict the set of isometric transforms to the isometric transforms that also map $\mathcal{S}_1$ onto $\mathcal{S}_2$ (but not necessarily $\mathcal{S}_2$ onto $\mathcal{S}_1$). Because these tiles have symmetries, this usually leaves multiple possibilities.

Note that the global symmetries found must always be applied with respect to the center of mass $\boldsymbol{g}$ of $\mathcal{S}$, according to the definition of a symmetry of $\mathcal{S}$.

At the end of the recursion step, we have the set of isometric transforms that map $\mathcal{H}_1 \cup \{\mathcal{S}_1\}$ onto $\mathcal{H}_2 \cup \{\mathcal{S}_2\}$.

Each recursion step narrows the choice of symmetries for $\mathcal{S}$. The recursion stops when either this set is reduced to identity transform or when we have used all the component tiles in the model. In the latter case, the isometric transforms found are the symmetries of the composite shape. The recursion is initiated by taking for $\mathcal{H}_1$ and $\mathcal{H}_2$ two similar tiles, that is, two tiles of the same class.

In the following paragraphs, we review the individual steps of the algorithm: finding all the isometric transforms which map tile $\mathcal{S}_1$ onto similar tile $\mathcal{S}_2$ and reducing the set of compatible symmetries of $\mathcal{S}$. We then illustrate the algorithm in a step-by-step example.

6.3.2 *Finding All the Isometries Which Transform a Tile onto a Similar Tile.* At each step of our algorithm, we examine pairs of similar tiles, $\mathcal{S}_1$ and $\mathcal{S}_2$, and we have to find all the isometries which map $\mathcal{S}_1$ onto $\mathcal{S}_2$.

If $\boldsymbol{g}_i$ is the center of mass of tile $\mathcal{S}_i$ and $\boldsymbol{g}$ is the center of mass of the composite shape $\mathcal{S}$, this condition implies that the isometries we are looking for transform vector $\boldsymbol{g}_1 - \boldsymbol{g}$ into $\boldsymbol{g}_2 - \boldsymbol{g}$. In order to generate the set of all isometric transforms that map $\mathcal{S}_1$ onto $\mathcal{S}_2$, we use the following property.

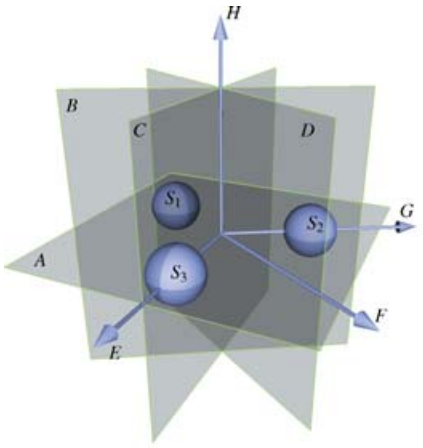
PROPERTY 4. *If J is an isometry that maps $\mathcal{S}_1$ onto a similar tile $\mathcal{S}_2$, then all the isometries K which map $\mathcal{S}_1$ onto $\mathcal{S}_2$ are of the following form:*

$$K = JT^{-1}AT \quad \textit{with} \quad A \in G_{\mathcal{S}_1} \quad \textit{such that} \quad A(\boldsymbol{g}_1 - \boldsymbol{g}) = \boldsymbol{g}_1 - \boldsymbol{g}, \tag{11}$$

where $G_{\mathcal{S}_1}$ is the group of symmetries of $\mathcal{S}_1$, and T is the translation of vector $\boldsymbol{g} - \boldsymbol{g}_1$ (refer to the Appendix for proof of this property).

This property states that, once we know a single *seed* isometric transform which maps $\mathcal{S}_1$ onto $\mathcal{S}_2$, we can generate all such transforms by using the elements of $G_{\mathcal{S}_1}$ in Equation (11).

6.3.3 *Finding a Seed Transform.* We need to find a seed transform J that maps $\mathcal{S}_1$ onto $\mathcal{S}_2$. For each tile, we extract a minimum set of independent vectors that correspond to extremas of their generalized even moment functions. The number of vectors needed depends on the symmetries of the tile. J is then defined as any isometric transform that maps the first set of vectors onto the second as well as vector $\boldsymbol{g}_1 - \boldsymbol{g}$ onto $\boldsymbol{g}_2 - \boldsymbol{g}$. Most of the time, a single isometric transform is possible at most. When multiple choices exist, the candidate transforms are checked onto the shapes using the distance presented in Section 5.3. This ensures that we find at least one seed transform.



Mapping	Symmetry
$S_1 \to S_1$ $S_2 \to S_2$ $S_3 \to S_3$	mirror-symmetry A
$S_1 \to S_3$ $S_3 \to S_1$ $S_2 \to S_2$	mirror-symmetry B 2-fold symmetry G
$S_1 \to S_2$ $S_2 \to S_1$ $S_3 \to S_3$	mirror-symmetry C 2-fold symmetry E
$S_2 \to S_3$ $S_3 \to S_2$ $S_1 \to S_1$	mirror-symmetry D 2-fold symmetry F
$S_1 \to S_2$ $S_2 \to S_3$ $S_3 \to S_1$	3-fold symmetry H

Fig. 8. Three spheres uniformly distributed on a circle in the z-plane. Etablishing all one-to-one mappings of the set of all tiles onto itself, which map each tile onto a similar tile, are used to detect all the symmetries of the shape. Note that the 3−fold symmetry H is detected and is associated to a circular permutation mapping.

6.3.4 *Ensuring Compatibility with Previous Isometries.* During the recursion, we need to store the current set of compatible isometries we have found. We do this by storing a minimal set of linearly independent vectors along with their expected images by these isometries. For example, if we have to store a symmetry of revolution, we store only one vector, the axis of the symmetry, and its image (itself). For mirror symmetries, rotations, and central symmetries, we store three independent vectors, along with their images by this isometric transform. For instance, in the case of a rotation of angle π around axis $\boldsymbol{X}$, we have:

$$\boldsymbol{X} \mapsto \boldsymbol{X} \quad \boldsymbol{Y} \mapsto -\boldsymbol{Y} \quad \boldsymbol{Z} \mapsto -\boldsymbol{Z}. \tag{12}$$

By examining all the one-to-one mappings of the set of all tiles onto itself, which map each tile onto a similar tile, we are able to detect all symmetries of the set of tiles (see Figure 8). Note in this example that the 3−fold symmetry H is detected and is associated to a circular permutation mapping.

6.4 Step-By-Step Example

Figure 9 presents a very simple example of a shape (a pair of pliers) composed of 3 tiles, $\mathcal{S}_1, \mathcal{S}_2$ (the handles), and $\mathcal{R}$ (the head). Two of the tiles are similar up to an isometric transform, $\mathcal{S}_1$ and $\mathcal{S}_2$. Figure 9 also displays the centers of mass, $\boldsymbol{g}_1$, and $\boldsymbol{g}_2$ of tiles $\mathcal{S}_1$ and $\mathcal{S}_2$ (which are not in the plane $z = 0$), and the center of mass $\boldsymbol{g}$ of the whole shape. In the coordinate systems centered on their respective centers of mass, $\mathcal{S}_1$ and $\mathcal{S}_2$ have a mirror symmetry of axis $\boldsymbol{Z}$, and $\mathcal{R}$ has a rotation symmetry around axis $\boldsymbol{X}$ of angle π.

Our constructive algorithm starts by selecting tile $\mathcal{R}$ and a similar tile (here, the only possible choice is $\mathcal{R}$).

Step 1. The algorithm explores the possibilities to transform $\mathcal{R}$ into itself. Two possibilities exist (a) the identity transform, and (b) the rotation around $\boldsymbol{X}$ of angle π, deduced from (a) by Property 4.
At this point, the algorithm branches, and either tries to map $\mathcal{S}_1$ to itself (branch 1) or to $\mathcal{S}_2$ (branch 2).
Branch 1, Step 1. The algorithm tries to match $\mathcal{S}_1$ to itself. The only compatible transform is the identity transform.

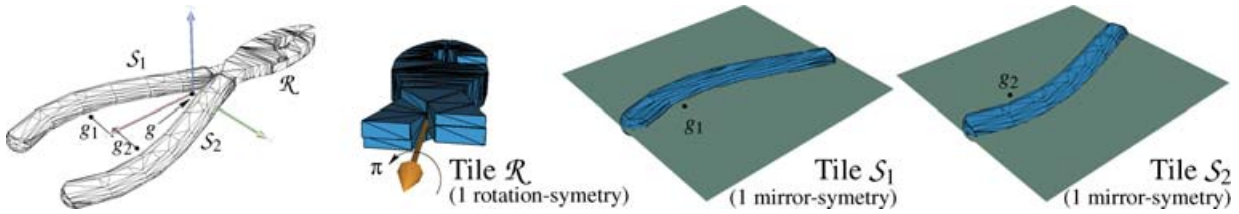


Fig. 9. Illustration of the constructive algorithm on a very simple example: from the symmetries of each of the 3 parts of the object, the symmetries of the whole object are recovered. Please note that no symmetry was ommitted in this Figure. In particular, tile $\mathcal{R}$ has only a rotational symmetry but no mirror symmetry. See text of Section 6.4 for a detailed explanation.

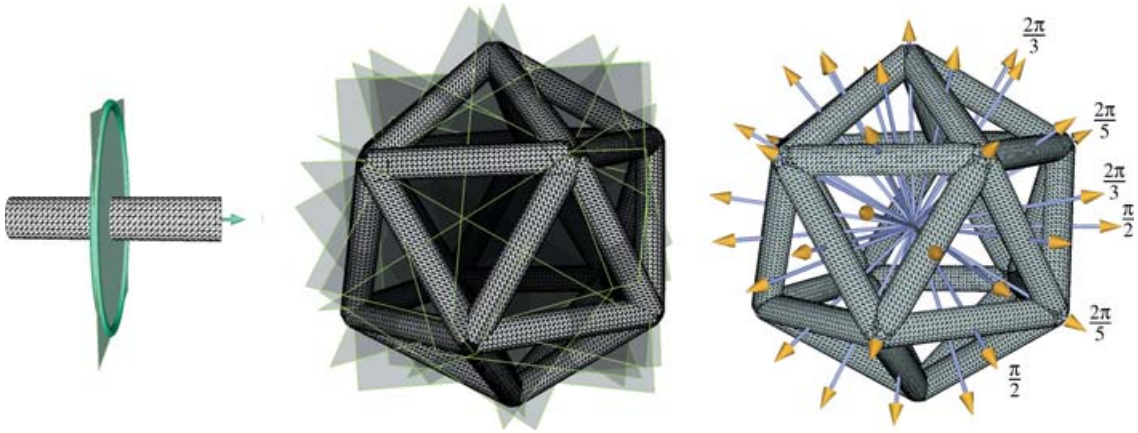


Fig. 10. A complex model which has the same group of symmetries as the icosahedron. The constructive algorithm successfully retrieves all 15 planes of mirror symmetries (*center*) and all 31 distinct axes of rotational symmetries (*right*) using the rotational and mirror symmetry of each tile (at *left*). The presence of 3−fold and 5−fold symmetries proves that our algorithm also detects symmetries which map a set of similar tiles onto itself through a complex permutation.

Branch 1, Step 2. The algorithm then tries to map $\mathcal{S}_2$ to itself. Once again, the only possible transform is the identity transform, and the recursion stops because all the tiles in the model have been used.
Branch 2, Step 1. The algorithm tries to match $\mathcal{S}_1$ to $\mathcal{S}_2$. The only compatible transform is the rotation around $\boldsymbol{X}$ of angle π.
Branch 2, Step 2. The algorithm then tries to match $\mathcal{S}_2$ to $\mathcal{S}_1$. Once again, the only compatible transform is the rotation around $\boldsymbol{X}$ of angle π, and the recursion stops because all the tiles in the model have been used.

Two symmetries have been found that map the shape onto itself, the identity transform and the rotation around $\boldsymbol{X}$ of angle π. Note that, although our algorithm can potentially create lots of branching, we prune branches that result in empty sets of transforms and, in practice, we only explore a small number of branches.

6.5 Application Scenarios

In order to illustrate the efficiency of the constructive algorithm, we show in this section various situations where this method is a valuable alternative to the single-shape algorithm.

6.5.1 *Application to an Agregation of Many Objects.* Figure 10 presents a complex model which has the same group of symmetries as an icosahedron. The constructive algorithm retrieves all the 31 distinct axis of rotational symmetries (Figure 10, *right*) as well as the 15 axis of plannar symmetries (Figure 10,

Table IV. Comparison of the costs of the single-shape algorithm presented in Section 5 to the cost of the constructive algorithm to find all 46 symmetries of the icosahedron shape displayed on Figure 10 at equivalent precision. Because the object is close to a sphere and because it has many symmetries, the constructive algorithm performs much better

Method	Single shape (order 10)	Constructive (order 4)
Moments calculation	500 sec	30×0.5 sec
Symmetry verification	46×55 sec	$30 \times 2 \times 1.5$ sec
Tile congruency	N/A	2 sec
Tile mappings	N/A	10 sec
Total	**50mn 30 sec**	**1mn 57 sec**

middle) of the shape, using the symmetries of each tile (Figure 10, *left*), which are 1 revolution symmetry and 1 mirror symmetry.

Conversely, directly applying the first algorithm on such a shape shows that $\mathcal{M}^2$ to $\mathcal{M}^8$ are extremely close to constant functions, making the extraction of directions an inaccurate process. The single-shape algorithm still correctly finds all the axis if using moments up to order 10, but this has some impact on computation times. Furthermore, the single-shape algorithm requires checking all of the symmetries found on the model which is a significant part of its computation time. This is not the case for the constructive algorithm because it relies on its knowledge of the symmetries of the tiles only. Because many symmetries exist for this model, the total computation time of the single-shape algorithm is therefore much higher. This is summarized in Table IV where we compare the computation times for both methods at equivalent precision (i.e., 10^{-4} radians).

6.5.2 *Finding Symmetries Inside Noncoherent Geometry.* There exist common situations where 3D scenes do not come as a set of closed separate objects but as an incoherent list of polygons. This happens, for instance, when retrieving geometric data from a Web site, mostly because a list of polygons constitutes a practical common denominator to all possible formats.

In such a case, applying the single-shape algorithm would certainly give the symmetries of the whole scene but if we are able to partition the set of polygons into adequate groups, that is, tiles to which we apply the constructive algorithm, we may be able to extract symmetric objects from the scene as well as the set of symmetries for the whole scene more rapidely as illustrated in Figure 10.

The gain in using the constructive algorithm to recover symmetries in the scene resides in the fact that, once tile symmetries have been computed, grouping them together and testing for symmetries in composed objects only adds a negligible cost which is not the case when we try to apply the single-shape algorithm to many possible groups of polygons or even to the entire scene itself.

The various issues in the decomposition of a raw list of polygons into intelligent tiles are beyond the scope of this article. In our case, tiles only need to be consistent with the symmetries. We propose the following heuristic to achieve this correctly for most scenes:

We define tiles as maximal sets of edge-connected polygons. To obtain them, we insert all vertices of the model into a KDTree and use this KDTree to efficiently recover which polygons share vertices up to a given precision and share an edge. By propagating connectivity information between neighboring polygons, we then build classes of edge-connected polygons, which we define to be our tiles. Figure 11 gives examples of such tiles for objects collected from the Web as a raw list of polygons.

Our simple heuristic approach of making tiles produced very good results on all scenes we tested and suffices for a proof of concept of the constructive algorithm. This is illustrated in Figure 11 where a lamp object and a chess game are shown along with their global symmetries. These symmetries were

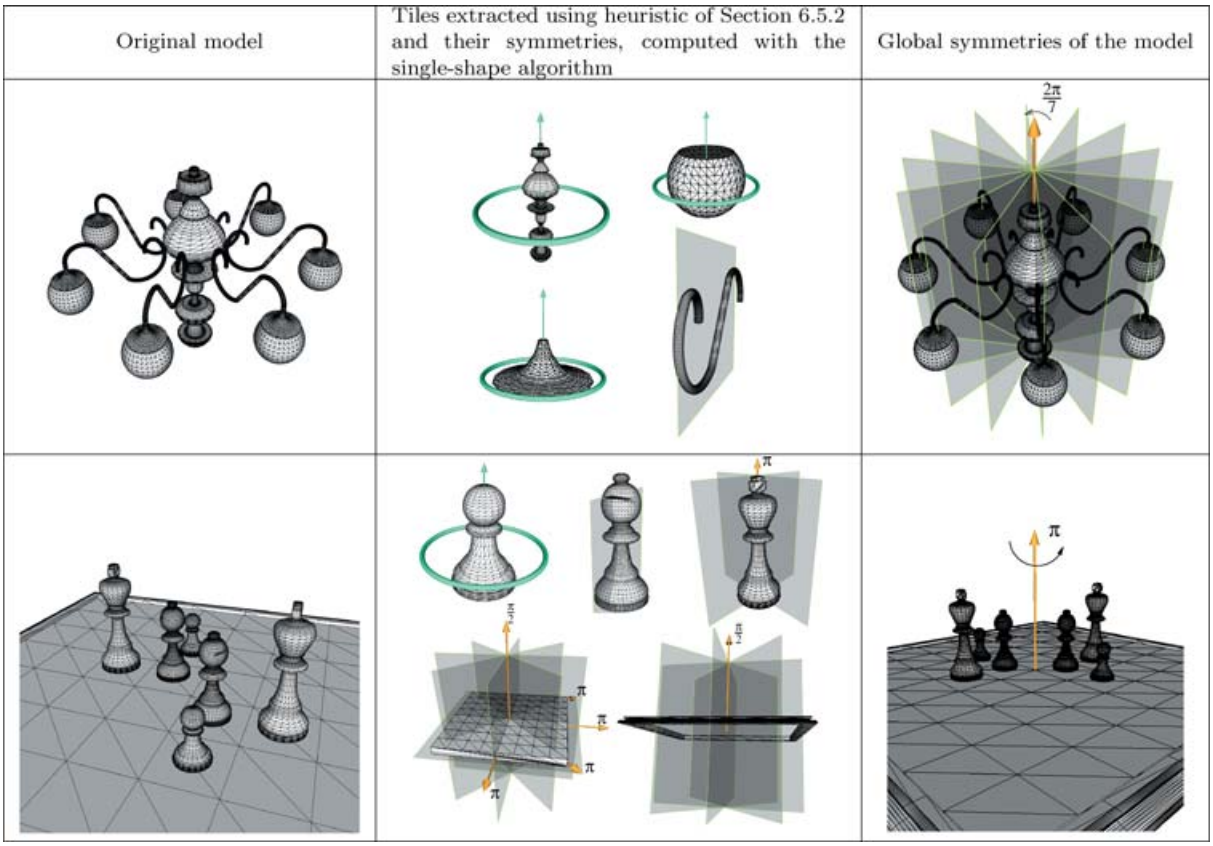


Fig. 11. Two models taken from the Web. From the raw list of polygons (left) our heuristic for scene partitionning extracts tiles before the single-shape algorithm computes the symmetries for each of them (center). Using this information, the constructive algorithm computes the symmetries of the whole model (right). *Top row*: A lamp object which has seven mirror symmetries and a 7−fold rotational symmetry. *Bottom row*: a chess board which is composed of pieces with very different symmetries but reveals to only have a single 2−fold symmetry around a vertical axis (Note: in this last model, once tiles have been identified, chess pieces were moved so as to obtain a model with at least one global symmetry).

computed from the symmetries of each of the subparts. These, in turn, were separately computed using the algorithm presented in Section 5.

Obviously, this application requires that constructed tiles be consistent with symmetries, that is, that it is possible to partition the scene into tiles which will map onto each other through the symmetries of the scene. This may not be easy with scanned models, for instance, nor in perturbated data. In such a case, our simple heuristic should be modified so as to base polygon neighborood relationships on proximity distances between polygons rather than vertex positions only. Doing so, cutting one tile into two parts and remeshing them independently, would have a high probability of producing the same original tile after reconstruction. If not, then the existance of a symmetry inside the model may become questionnable. Suppose, for instance, that the wrench in the step-by-step example (Section 6.4) gets split into tiles that are not exact symmetrical copies of one another, and that these two tiles are too far away to be merged into a single tile. Then the model is by nature not symmetric anymore which will also be the output of the constructive algorithm.

Table V. Computation Times (in seconds) for the Different Steps of our Algorithm, for the Models Shown in this Article

Model	Plier	Lamp	Chessboard
# polygons	1,940	39,550	24,942
# tiles	3	22	8
Computing moments*	0.9	18.2	15
Finding parameters	0.4	1.2	2.0
Checking candidates	2.3	7.4	7.9
Constructive algo.	0.001	0.05	0.01
Total	3.601	26.85	24.91

*Global computation time for moments of order 2 to 8.

6.6 Computation Cost

Computation times (in seconds) for the models shown in this article are given in Table V as well as the complexity of the models. They were measured on a machine equipped with a 2.4GHz processor with 512MB of memory. As expected, the cost of the computation of the moment functions and the cost of the verification of the candidates required by the first algorithm occupy the most important part of the total cost and depend on the model complexity. Conversely, finding the parameters of the symmetries (Section 5.2) as well as applying the constructive algorithm only depends on the number of these symmetries.

Regarding accuracy, both algorithms computed the axes of the symmetries with a maximum error of 10^{-4} radians, independently of shape complexity, in our tests.

7. APPLICATIONS

7.1 Geometry Compression and Instantiation

Our framework can be used for model compression at two different levels. (1) If a model exhibits symmetries, then it can be compressed by storing only the significant part of the model and using the symmetries to recreate the full model. (2) If a model contains multiple instances of the same part, then these parts can be instantiated. (see Figure 12).

Although complex models often do not present symmetries, symmetry-based compression can usually be used on some subparts of the model. The ability to express a model by explicitely storing the significant parts only while instancing the rest of the scene is provided by some recent 3D file formats such as X3D (see Table VI). We thus measure our compression ratios as the size of the X3D files before and after our two compression operations which we detail now.

The scene is first loaded as a raw collection of polygons, before being decomposed into tiles, using the heuristic presented in Section 6.5.2. We then compute symmetries and congruent descriptors for each tile. Computation times shown in Table VI present the average time needed to compute symmetries and congruent descriptors for a single tile. As the process of computing tile properties does not depend on the other tiles, it is an easily parallelizable process. The scene is then first compressed by instancing the tiles. Secondly, when storing each tile, we only store the minimum significant part of its geometry according to its symmetries. This part is extracted using the same algorithm we will present for remeshing a tile according to its symmetries in the next section. Note that compression rates shown on this table are computed using geometry informations only, that is, neither texturing nor material information are taken into account. Compression times shown in Table VI are the times needed to detect all classes of tile congruency.

Fig. 12. Detecting symmetries and similarities between tiles created from a raw list of polygons allows us to compress geometric models in two ways: (1) by instancing similar tiles and (2) inside each symmetric tile, by instancing the part of the geometry which permits to reconstruct the whole tile. In such a big model as the powerplant (13 millions triangles), we achieve a compression ratio (ratio of geometry file size in X3D format) of 1:4.5. We show in this figure two subparts of the complete model. For each, we show the tiles computed by our heuristic (see Section 6.5) as well as the obtained compression ratio. The PowerPlant model is a courtesy of The Walkthru Project.

Table VI. Examples of Compression Rates Obtained Using our Symmetry Detection Method Coupled with the Congruency Descriptor. (See text in Section 7.1 for a detailed explanation.)

Model	Room	Plane	Studio	Powerplant
# polygons	39, 557	182, 224	515, 977	12, 748, 510
# tiles	851	480	5, 700	525, 154
av. computing tile properties (secs)	1.45	1.3	1.9	1.1
Compression time (secs)	7.2	9	14.6	311
Compression rate	1 : 2.7	1 : 8.3	1 : 3.5	1 : 4.5

7.2 Mesh Editing

It may be interesting, when an object contains symmetries, to remesh the object with respect to these symmetries. In order to do this, we proceed by first extracting the minimum part of the shape that can be reconstructed through each symmetry independently, then we apply the corresponding symmetry to each of them in order to get as many meshes of the shape which are consistent with each symmetry independently. The final step is to compute the union of all these meshes, merging identical vertices and adding new vertices at edge crossings. While not necessarily optimal, the obtained mesh is consistent with all symmetries of the shape.

Since a coherent remeshing allows for the establishment of a correspondence between model vertices, we have developed a proof-of-concept mesh editing system which allows the user to modify a 3D object under the constraints given by the symmetries of the original object. It appears that, under the constraint of too many symmetries, no vertices can be moved independently of the others, and the geometry is sometimes bound to scale about its center of gravity. Images collected from this program are displayed in Figure 13.

8. DISCUSSION

We discuss here a number of features of our technique as well as differences with existing approaches.

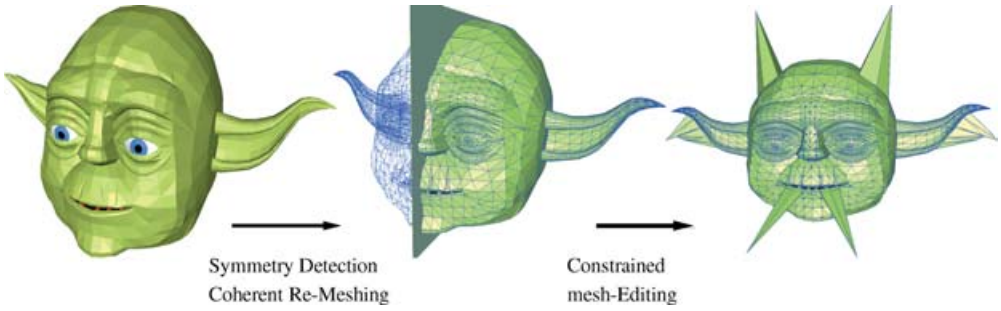


Fig. 13. Starting from an object in arbitrary orientation, we detect symmetries of the shape (in the figure, a planar symmetry) and use it to remesh the objects with respect to these symmetries. Then, a user can easily edit the mesh and modify it while keeping the symmetries of the initial shape.

Using Spherical Harmonics

Generalized moments are a central component of our system. As stated before, we do not compute these functions explicitly but we rather compute their coefficients in a spherical harmonics basis. As for the decomposition itself, any basis could be used. In particular, a well chosen basis of 3D monomials restricted to the unit sphere may also lead to a finite decomposition. Still, using spherical harmonics has many advantages, in particular, because we use the same coefficients computed once for different tasks throughout this article. (1) The expression of moment function as a sum of spherical harmonics provides an accurate detection of the potential axes of symmetries. This detection is made deterministic by finding the zero directions for the gradient of the moment functions. Such a computation is performed analytically from the 2nd order derivatives of the spherical harmonics, and thus does not introduce further approximation. (2) Computing symmetry parameters for the moment functions is made very easy by working on the spherical harmonic coefficients themselves. Since spherical harmonics are orthogonal and easily rotated, finding symmetries on the moment functions translates into simple relationships between the coefficients. (3) The spherical harmonic coefficients provide an effective shape congruency descriptor which we use to detect which tiles are identical up to an unknown isometric transform.

In summary, the use of spherical harmonics provides us a consistent framework throughout the whole process of our symmetry-finding algorithm.

Non Star-Shaped Objects

Whether the direct algorithm presented in Section 5 works for non star-shaped objects is a legitimate question. Our approach never relies on a spherical projection. Indeed, the moment functions, as expressed in Equations (1) and (5) are computed through an integration over the surface itself, possibly covering the same directions multiple times but with different values. Parts of a shape which correspond to a same direction during integration will not contribute the same into the various moment functions because of the varying exponent. By using various orders of moment functions in our symmetry detection process and in the computation of our shape congruency descriptor, we thus capture the geometry of non star-shaped objects as well. Some previous approaches [Kazhdan et al. 2004] achieved this by decomposing the shape into concentric spherical regions before doing a spherical integration which can be assimilated to convoluting the shape with 0-degree functions with concentric spherical support; Our technique is similar, but with another, kind of functions expressed into the form of the even moments. In summary, detecting symmetries on non star-shaped objects has no particular reason to fail which is illustrated by the result in Figure 2.

The second algorithm (for assembled objects) naturally works just as well for non star-shaped objects as illustrated by the examples in Figure 11.

Avoiding Dense Sampling

Previous methods that defined a continuous measure of symmetry([Zabrodsky et al. 1995; Kazhdan et al. 2004]) can theoretically compute both perfect and approximate symmetries. However, detecting symmetries using such methods involves a sampling step of the directions on the sphere, whose density must be adapted to the desired angular precision for the axis of the symmetry.

The work of Kazhdan et al. [2004] leads to impressive results concerning the improvment on the shape matching process. However, relying on this technique to obtain accurate symmetries with high angular precision requires a time-consuming step for the construction of the symmetry descriptors. According to the presented results, the time needed to compute reflective, 2-fold, 3-fold, 4-fold, 5-fold, and axial symmetry information for a spherical function of bandwidth $b = 16$ is 0.59 seconds. As stated in the article [Kazhdan et al. 2004], the number of samples taken on the sphere is $O(b^2)$ (i.e., approximately 10^3 sample directions) which is insufficent to reach a high angular precision equivalent to the one obtained with our method: reaching a precision of 10^{-4} radians would require approximately 10^9 sample directions. This would theoretically increase the computation time to approximately $0.59 \times 10^9/10^3 = 5.9\ 10^5$ seconds, making the method inefficient for this task.

In contrast, our method does not rely on a dense sampling of directions to find symmetries but on the computation of a fixed number of surface integrals which—thanks to the Gauss integration used—provides an extremely accurate approximation of the spherical harmonic coefficients of the moment functions. From there on, no further approximation is introduced in the computation of the directions of the candidate symmetries which lets us achieve an excellent angular precision at a much lower cost.

Furthermore, the cost of our algorithm does not rely on assumptions about the expected results. The method of Kazhdan et al. [2004] indeed computes symmetry descriptors for each kind of searched symmetry. Our method in turn computes all directions of possible symmetries and then checks back on the shape of the obtained candidates.

9. CONCLUSIONS

We have presented an algorithm to automatically retrieve symmetries for geometric shapes and models. Our algorithm efficiently and accurately retrieves all symmetries from a given model, independently of its tesselation.

We use a new tool, the generalized moment functions, to identify candidates for symmetries. The validity of each candidate is checked against the original shape using a geometric measure. Generalized moments are not computed directly: instead, we compute their spherical harmonic coefficients using an integral expression. Having an analytical expression for the generalized moment functions and their gradients, our algorithm finds potential symmetry axes quickly and with good accuracy.

For composite shapes assembled from simpler elements, we have presented an extension of this algorithm that works by first identifying the symmetries of each element, then sets of congruent elements. We then use this information to iteratively build the symmetries of the composite shape. This extension is able to handle complex shapes with better accuracy since it pushes the accuracy issues down to the scale of the tiles.

Future Work

The constructive algorithm presented in Section 6 automatically detects instantiation relationships between tiles into a composite shape.

We are currently developing a constructive instantiation algorithm which iteratively collates similar tiles into instances, checking at each step that the relative orientation of each tile with respect to each already constructed instance is preserved.

This algorithm requires the symmetries of the tiles, and maintaining the symmetries of the instances found so far. For this, we use our shape congruency metric, our algorithm for finding symmetries of single shapes, and our algorithm for finding symmetries on composite shapes.

APPENDIX (PROOFS)

PROOF OF THEOREM 1. Let A be an isometric transform which lets a shape $\mathcal{S}$ be globally unchanged. We have:

$$\begin{aligned}
\forall \boldsymbol{\omega} \quad \mathcal{M}^{2p}(A\boldsymbol{\omega}) &= \int_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{s}\times A\boldsymbol{\omega}\|^{2p}\,\mathrm{d}\boldsymbol{s} \\
&= \int_{\boldsymbol{t}\in A^{-1}\mathcal{S}} \|A\boldsymbol{t}\times A\boldsymbol{\omega}\|^{2p}|\det A|\,\mathrm{d}\boldsymbol{t} \\
&= \int_{\boldsymbol{t}\in A^{-1}\mathcal{S}} \|\boldsymbol{t}\times \boldsymbol{\omega}\|^{2p}\,\mathrm{d}\boldsymbol{t} \\
&= \mathcal{M}^{2p}(\boldsymbol{\omega})
\end{aligned}$$

At line 2, we change variables and integrate over the surface transformed by A^{-1}. At line 3, an isometric transform is a unit transform and so, its determinant is ± 1 and thus vanishes. The cross product is also left unchanged by applying an isometric transform to each of its terms. Line 4: because $A\mathcal{S} = \mathcal{S}$, we also have $\mathcal{S} = A^{-1}S$. The isometric transform A is thus also a symmetry of the $\mathcal{M}^{2p}$ moment functions.

Let A be an isometric transform with axis $\boldsymbol{v}$, and suppose that A is a symmetry of $\mathcal{M}^{2p}$. Let $\boldsymbol{d_v}$ be the direction of steepest descent of function $\mathcal{M}^{2p}$ around direction $\boldsymbol{v}$. Because A is a symmetry of $\mathcal{M}^{2p}$, we have:

$$\boldsymbol{d}_{A\boldsymbol{v}} = A\boldsymbol{d_v} = \boldsymbol{d_v}. \tag{13}$$

If A is a rotation, this is impossible because $\boldsymbol{d_v} \perp \boldsymbol{v}$. Moreover, for all directions ω, we have $\mathcal{M}^{2p}(-\omega) = \mathcal{M}^{2p}(\omega)$ and thus:

$$\boldsymbol{d}_{-\boldsymbol{v}} = -\boldsymbol{d_v}. \tag{14}$$

So, if A is a symmetry, we have $A\boldsymbol{v} = -\boldsymbol{v}$. From Equations (13) and (14), we get $\boldsymbol{d_v} = -\boldsymbol{d_v}$ which is impossible.

In both cases, $\mathcal{M}^{2p}$ can not have a direction of steepest descent in direction $\boldsymbol{v}$. Because $\mathcal{M}^{2p}$ is infinitely derivable, this implies that $\boldsymbol{\nabla}\mathcal{M}^{2p}(\boldsymbol{v}) = 0$ □

PROOF OF PROPERTY 4. Let $\mathcal{S}$ and $\mathcal{R}$ be two shapes, identical up to an isometric transform. Let J be an isometric transform such that $J\mathcal{S} = \mathcal{R}$. Let T be the translation of vector $-u_{\mathcal{S}}$ with $\boldsymbol{u}_{\mathcal{S}} = \boldsymbol{g}_{\mathcal{S}} - \boldsymbol{g}$ with $\boldsymbol{g}_{\mathcal{S}}$ as the center of mass of $\mathcal{S}$, and $\boldsymbol{g}$ the origin of the coordinate system into which J is applied.

— Let $A \in G_S$ be a symmetry of $\mathcal{S}$ such that $A\boldsymbol{u}_{\mathcal{S}} = \boldsymbol{u}_{\mathcal{S}}$. We have $AT\mathcal{S} = T\mathcal{S}$ (the symmetry A operates in the coordinate system centered on $\boldsymbol{g}_{\mathcal{S}}$). Let $K = JT^{-1}AT$. Then

$$\begin{aligned}
K\mathcal{S} &= JT^{-1}AT\mathcal{S} \\
&= JT^{-1}T\mathcal{S} \\
&= J\mathcal{S} \\
&= \mathcal{R}
\end{aligned}
\qquad \text{and} \qquad
\begin{aligned}
K\mathbf{0} &= JT^{-1}AT\mathbf{0} \\
&= JT^{-1}A(-\boldsymbol{u}_{\mathcal{S}}) \\
&= JT^{-1}(-\boldsymbol{u}_{\mathcal{S}}) \\
&= J\mathbf{0} = 0
\end{aligned}$$

By construction K is a rigid transform and conserves distances. It maps the origin onto itself. K is thus an isometric transform. Furthermore, K maps $\mathcal{S}$ to $\mathcal{R}$.

— Let K be an isometric transform such that $K\mathcal{S} = \mathcal{R}$. Let us choose $A = TJ^{-1}KT^{-1}$. This choice leads to $K = JT^{-1}AT$. Moreover:

$$\begin{aligned} AT\mathcal{S} &= TJ^{-1}KT^{-1}T\mathcal{S} \\ &= TJ^{-1}KS \\ &= T\mathcal{S} \end{aligned} \quad \text{and} \quad \begin{aligned} A\boldsymbol{u}_{\mathcal{S}} &= TJ^{-1}KT^{-1}\boldsymbol{u}_{\mathcal{S}} \\ &= TJ^{-1}K2\boldsymbol{u}_{\mathcal{S}} \\ &= T2\boldsymbol{u}_{\mathcal{S}} = \boldsymbol{u}_{\mathcal{S}} \end{aligned}$$

and

$$\begin{aligned} A\mathbf{0} &= TJ^{-1}KT^{-1}\mathbf{0} \\ &= TJ^{-1}K\boldsymbol{u}_{\mathcal{S}} \\ &= TJ^{-1}(g_{\mathcal{R}} - \boldsymbol{g}) \\ &= T(-\boldsymbol{u}_{\mathcal{S}}) \\ &= \mathbf{0} \end{aligned}$$

By construction A is affine and conserves distances. It maps $\mathbf{0}$ onto $\mathbf{0}$. A is thus an isometric transform. A is also a symmetry of $\mathcal{S}$ which verifies $A\boldsymbol{u}_{\mathcal{S}} = \boldsymbol{u}_{\mathcal{S}}$.

— The set of isometries which map $\mathcal{S}$ to $\mathcal{R}$ is therefore the set of functions K of the form $K = JT^{-1}AT$, where $A \in G_{\mathcal{S}}$ is a symmetry of $\mathcal{S}$ such that $A(\boldsymbol{g} - \boldsymbol{g}_{\mathcal{S}}) = (\boldsymbol{g} - \boldsymbol{g}_{\mathcal{S}})$.

□

PROOF OF EQUATION 3. We compute the decomposition of function $\theta \longmapsto \sin^{2p}\theta$ into zonal spherical harmonics. We prove that this decomposition is finite, and give the values of the coeficients.

By definition [Hobson 1931], we have:

$$\begin{aligned} Y_L^0(\theta,\varphi) &= \sqrt{\frac{2L+1}{4\pi}} P_L(\cos\theta) \\ &= \sqrt{\frac{2L+1}{4\pi}} \frac{(-1)^L}{2^L L!} \frac{d^L}{dx^L}\left[(1-x^2)^L\right](\cos\theta) \end{aligned}$$

where P_k is the Legendre polynomial of order k. Because the set of Legendre polynomials $P_0, P_1, ..., P_n$ is a basis for polynomials of order not greater than n, function $\theta \longmapsto \sin^{2p}\theta = (1-\cos^2\theta)^p$ can be uniquely expressed in terms of $P_L(\cos\theta)$. The decomposition of $\theta \longmapsto \sin^{2p}\theta$ is thus finite and has terms up to Y_{2p}^0 at most.

Let's compute them explicitely:

$$\begin{aligned} \frac{d^L}{dx^L}\left[(1-x^2)^L\right] &= \frac{d^L}{dx^L}\sum_{k=0}^{L}(-1)^{L-k}x^{2L-2k}C_L^k \\ &= (-1)^L\frac{d^L}{dx^L}\sum_{k=0}^{L}(-1)^k x^{2k}C_L^k \\ &= \sum_{L\le 2k\le 2L}(-1)^{L+k}C_L^k 2k(2k-1)...(2k-L+1)x^{2k-L} \\ &= \sum_{L\le 2k\le 2L}(-1)^{L+k}C_L^k\frac{(2k)!}{(2k-L)!}x^{2k-L} \end{aligned}$$

So:

$$Y_L^0(\theta,\varphi) = \sqrt{\frac{2L+1}{4\pi}} \sum_{L\leq 2k\leq 2L} \frac{(-1)^k}{2^L L!} C_L^k \frac{(2k)!}{(2k-L)!} \cos^{2k-L}\theta$$

The coeficients of the decomposition we are interested in are thus:

$$\int_{\theta=0}^{\pi}\int_{\varphi=0}^{2\pi} Y_L^0(\theta,\varphi)\sin^{2p}\theta\sin\theta d\theta d\varphi = 2\pi\sqrt{\frac{2L+1}{4\pi}} \sum_{L\leq 2k\leq 2L} \frac{(-1)^k}{2^L L!} C_L^k \frac{(2k)!}{(2k-L)!} I_{2k-L}^p \tag{15}$$

where integrals I_m^p are defined by:

$$I_m^p = \int_{\theta=0}^{\pi} \sin^{2p+1}\theta\cos^m\theta d\theta$$

First, $I_m^p = 0$ for all odd m because the integrand in antisymetric around $x = \pi/2$. Then, if m is even:

$$\begin{aligned} I_m^p &= \underbrace{\left[\frac{1}{2p+2}\sin^{2p+2}\theta\cos^{m-1}\theta\right]_0^{\pi}}_{0} + \frac{m-1}{2p+2}\int_0^{\pi}\sin^{2p+3}\theta\cos^{m-2}\theta d\theta \\ &= \frac{m-1}{2p+2} I_{m-2}^{2p+3} \\ &= \frac{(m-1)(m-3)\ldots 1}{(2p+2)(2p+4)\ldots(2p+m)}\int_o^{\pi}\sin^{2p+m+1}\theta d\theta \end{aligned}$$

Let J_q be the integral defined by

$$J_q = \int_0^{\pi}\sin^{2q+1}\theta d\theta.$$

We have

$$\begin{aligned} J_q &= \underbrace{[-\cos\theta\sin^{2q}\theta]_0^{\pi}}_{0} + 2q\int_0^{\pi}\cos^2\theta\sin^{2q-1}\theta d\theta \\ &= 2qJ_{q-1} - 2qJ_q \end{aligned}$$

Therefore

$$\begin{aligned} J_q &= \frac{2q}{2q+1}J_{q-1} \\ &= \frac{2q(2q-2)\ldots 2}{(2q+1)(2q-1)\ldots 3}J_0 \\ &= \frac{2^{2q+1}(q!)^2}{(2q+1)!} \end{aligned}$$

For m even, we can take $m = 2r$ and $q = p + r$; we get:

$$\begin{aligned} I_{2r}^p &= \frac{(2r)!p!}{2^r r! 2^r (p+r)!}\frac{2^{2p+2r+1}(p+r)!^2}{(2p+2r+1)!} \\ &= \frac{(2r)!p!2^{2p+1}(p+r)!}{r!(2p+2r+1)!} \end{aligned} \tag{16}$$

From Equation (15), we deduce that, for L odd,

$$\int\int Y_L^0(\theta,\varphi)\sin^{2p}\theta\sin\theta d\theta d\varphi = 0.$$

For L even, we set $L = 2l$. Using $r = k - l$ to match Equation (16) in Equation (15), we get:

$$\begin{aligned}
S_p^l &= \int\int Y_{2l}^0(\theta,\varphi)\sin^{2p}\theta\sin\theta d\theta d\varphi \\
&= 2\pi\sqrt{\frac{4l+1}{4\pi}}\sum_{2l\le 2k\le 4l}\frac{(-1)^k}{2^{2l}(2l)!}C_{2l}^k\frac{(2k)!}{(2k-2l)!}\frac{(2k-2l)!p!2^{2p+1}(p+k-l)!}{(k-l)!(2p+2k-2l+1)!} \\
&= \frac{\sqrt{(4l+1)\pi}}{2^{2l}(2l)!}\sum_{l\le k\le 2l}(-1)^k C_{2l}^k\frac{(2k)!p!2^{2p+1}(p+k-l)!}{(k-l)!(2p+2k-2l+1)!} \\
&= \frac{\sqrt{(4l+1)\pi}}{2^{2l}}\sum_{l\le k\le 2l}(-1)^k\frac{(2k)!p!2^{2p+1}(p+k-l)!}{k!(2l-k)!(k-l)!(2p+2k-2l+1)!}
\end{aligned}$$

□

REFERENCES

ATTALAH, M. J. 1985. On symmetry detection. *IEEE Trans. Comput. 34*, 663–666.

BRASS, P. AND KNAUER, C. 2004. Testing congruence and symmetry for general 3-dimensional objects. *Comput. Geom. Theory Appl. 27*, 1, 3–11.

HIGHNAM, P. T. 1985. Optimal algorithms for finding the symmetries of a planar point set. Tech. Rep. CMU-RI-TR-85-13 (Aug). Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

HOBSON, E. W. 1931. *The Theory of Spherical and Ellipsoidal Harmonics*. Cambridge University Press, Cambridge, UK.

IVANIC, J. AND RUEDENBERG, K. 1996. Rotation matrices for real spherical harmonics, direct determination by recursion. *J. Phys. Chem. A. 100*, 6342–6347. (See also *Additions and corrections* in vol. 102, No. 45, 9099-9100).

JIANG, X.-Y. AND BUNKE, H. 1991. Determination of the symmetries of polyhedra and an application to object recognition. In *Proceedings of the International Workshop on Computational Geometry—Methods, Algorithms and Applications (CG '91)*. Lecture Notes in Computer Science, vol. 553. Springer, London, UK, 113–121.

KAZHDAN, M. M., FUNKHOUSER, T. A., AND RUSINKIEWICZ, S. 2003. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Proceedings of the 2003 Eurographics/ACM Siggraph Symposium on Geometry Processing (SGP '03)*. Eurographics Association, Aire-la-Ville, Switzerland, 167–175.

KAZHDAN, M. M., FUNKHOUSER, T. A., AND RUSINKIEWICZ, S. 2004. Symmetry descriptors and 3D shape matching. In *Proceedings of the 2004 Eurographics/ACM Siggraph Symposium on Geometry Processing (SGP '04)*. Eurographics Association, Aire-la-Ville, Switzerland.

KNUTH, D. E., MORRIS, JR., J. H., AND PRATT, V. R. 1977. Fast pattern matching in strings. *SIAM J. Comput. 6*, 2, 323–350.

MINOVIC, P., ISHIKAWA, S., AND KATO, K. 1993. Symmetry identification of a 3-D object represented by octree. *IEEE Trans. Patt. Analy. Mach. Intell. 15*, 5, 507–514.

PRINCE, E. 2004. *Mathematical Techniques in Crystallography and Materials Science*, 3rd Ed. Springer, Berlin, Germany.

RAMAMOORTHI, R. AND HANRAHAN, P. 2004. A signal-processing framework for reflection. *ACM Trans. Graph. 23*, 4, 1004–1042.

SUN, C. AND SHERRAH, J. 1997. 3D symmetry detection using extended Gaussian image. *IEEE Trans. Patt. Analy. Mach. Intell. 19*, 2 (Feb.), 164–168.

WOLTER, J. D., WOO, T. C., AND VOLZ, R. A. 1985. Optimal algorithms for symmetry detection in two and three dimensions. *Visual Comput. 1*, 37–48.

ZABRODSKY, H., PELEG, S., AND AVNIR, D. 1995. Symmetry as a continuous feature. *IEEE Trans. Patt. Analy. Mach. Intell. 17*, 12, 1154–1166.

Received August 2005; revised December 2005; accepted January 2006