INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Accurately Detecting Symmetries of 3D Shapes

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

ARTIS - GRAVIR/IMAG - INRIA

N° 5692

Septembre 2005

Thème COG

Rapport de recherche

ISRN INRIA/RR--5693--FR+ENG

N 0249-6399

Accurately Detecting Symmetries of 3D Shapes

Aurélien Martinet, Cyril Soler, Nicolas Holzschuch and François X. Sillion
ARTIS - GRAVIR/IMAG - INRIA

Thème COG — Systèmes cognitifs
Projet ARTIS

Rapport de recherche n° 5692 — Septembre 2005 — 27 pages

Abstract: 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 sub-parts. 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.

Key-words: Massive Datasets, Symmetries, Geometry Instancing

Détection précise des symétries de formes 3D

Résumé : Nous proposons une méthode automatique pour identifier les symétries de formes 3D, i.e. les isométries qui laisse une forme globalement invariante. Ces symétries sont déterminées de manière précise à l'aide de l'utilisation d'une quantité intermédiaire : Les moments généralisés. En examinant les extrema et les coefficients d'harmoniques sphériques de ces moments, nous retrouvons les paramètres des symétries de la forme. La recherche de symétries pour les modèles complexes est rendu efficace grâce à l'utilisation de ces informations dans un algorithme incrémental, capable de retrouver les symétries d'une forme en utilisant les symétries de ses sous-parties. Les applications de ce travail concernent le remaillage cohérent de la géométrie en respectant la symétrie des modèles, la compression géométrique, l'édition intelligente de maillages et l'instanciation automatique.

Mots-clés : Masse de Données, Symétries, Instanciation de Géométrie

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 paper, 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, i.e. shapes built by assembling simpler structures, we optimize the computation by applying the first algorithm to the sub-parts, then iteratively building the set of symmetries of the composite shape taking into account both the relative positions of the sub-parts and their relative orientations.

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

This paper 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. The extension of this algorithm to composite shapes is presented in section 6. Finally, in section 7 we show some results and 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 non-perfect 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.

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,

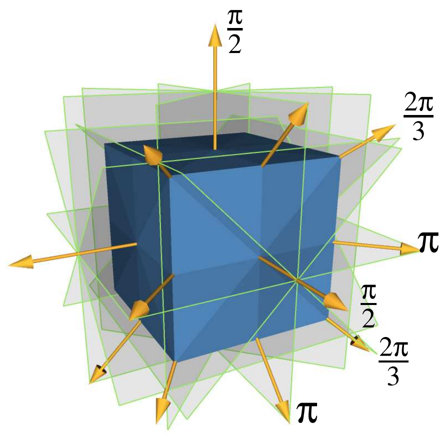


Figure 1: Mirror-symmetries and rotational-symmetries found by our algorithm for a cube (for clarity, not all elements are represented).

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:

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

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 to find good candidates.

Our approach to finding symmetries uses an intermediate quantity, called the *generalized moment* functions of the shape. These functions will be the topic of section 4. By examining these quantities 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 may appear during the process, these candidates are then checked back on the original shape. In the specific case of a central symmetry ($\lambda = -1$ and $\alpha = \pi$), it should be noted that any direction can be the *axis* of the symmetry according to the above definition. However, this specific case does not require finding any parameters and will simply be added to the candidate symmetries to check for.

Some shapes however may be *almost* symmetric, and computing symmetries for these shapes using the generalized moment functions of the whole shape may prove inaccurate, producing for instance false positives. A second contribution of this paper is to show that symmetries can also be found by a constructive algorithm, by separately computing the symmetries of sub-components of an object using the first method, determining their similarity up to isometric transformations, and associating this information to compute symmetries of the whole shape. This constructive algorithm prooves to be more accurate. 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 interesting relationships with the symmetries of the shape, and that they provide an effective method to retrieve such symmetries.

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 a 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 leads to very interesting properties.

4.2 Shape symmetries and moments

Symmetry properties of a shape translate into symmetry properties of its moment functions. We will make use of the following two properties (see proof in Appendix):

Property 1 *: All symmetries I of a shape $\mathcal{S}$ are also symmetries of all its $\mathcal{M}^{2p}$ moment functions:*

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

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

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

These two properties together imply 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

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.

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:

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

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:

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

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 formulae [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:

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

The corresponding derivation is too long to be exposed here, but does not present any difficulty.

Let $R_{\boldsymbol{s}}$ 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 :

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

And finally:

$$\mathcal{M}^{2p}(\boldsymbol{\omega}) = \sum_{l=0}^{p}\sum_{m=-2l}^{2l} C_{2l,m}^{2p} Y_{2l}^m(\boldsymbol{\omega}) \tag{3}$$

using

$$C_{2l,m}^{2p} = S_p^l \int_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{s}\|^{2p} D_{2l}^{0,m}(R_{\boldsymbol{s}})\,\mathrm{d}\boldsymbol{s} \tag{4}$$

Equation 3 proves that $\mathcal{M}^{2p}$ decomposes into a *finite* number of spherical harmonics, and Equation 4 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 define 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 concerned when computing the $C_{2k,p}^m$ coefficients, and we can now compute both $\mathcal{M}^{2p}$ and its gradient analytically from Equation 3.

5 Finding symmetries of a 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 first decomposed into sub-parts. 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 at the time of 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:

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

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 $\|\boldsymbol{\nabla}(\mathcal{M}^{2p})(\boldsymbol{\omega})\|^2$ 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 $\|\boldsymbol{\nabla}(\mathcal{M}^{2p})(\boldsymbol{\omega})\|^2$, starting from each of the candidates found during the first step. For this we need to evaluate the derivatives of $\|\boldsymbol{\nabla}(\mathcal{M}^{2p})\|$, which we do using analytically computed second order derivatives of the spherical harmonics along with Equation 3. The minimization converges in a few steps because starting positions are 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 $C_{2l,m}^{2p}$ 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.

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 have the following properties:

Property 3 *: 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:*

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

Property 4 *: A function has a revolution-symmetry around the z axis if and only if it decomposes onto zonal harmonics only, i.e.*

$$\forall l \;\; \forall m \qquad m \neq 0 \quad \Rightarrow \quad C_l^m = 0$$

Property 5 *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_l^m verify:*

$$\forall l \;\; \forall m \qquad C_l^m = \cos(m\alpha) C_l^m - \sin(m\alpha) C_l^{-m} \tag{5}$$

Property 5 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:

$$\forall l \;\; \forall m \qquad (-1)^{l+m} C_l^m = \cos(m\alpha) C_l^m - \sin(m\alpha) C_l^{-m} \tag{6}$$

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 3 and 4 is trivial provided that some tolerance is accepted on the equalities. Using property 5 is more subtle: coefficients of the function are first examined by order of decreasing m. For $\lambda = 1$ for instance, when the first non zero value of C_l^m is found, Equation 5 is solved by:

$$\tan \frac{m\alpha}{2} = \frac{C_l^{-m}}{C_l^m} \qquad \text{i.e.} \qquad \alpha = \frac{2}{m} \arctan\left(\frac{C_l^{-m}}{C_l^m}\right) + \frac{k\pi}{m}$$

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 3, 4 and 5 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 pertubated data, such as scanned models.

5.3 Filtering results

The conditions extracted from Properties 1 and 2 are necessary conditions 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:

$$d_M(\mathcal{S}, \mathcal{R}) = \max_{p \in V_{\mathcal{S}}} (\min_{q \in \mathcal{R}} \|p - q\|) \tag{7}$$

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 7: 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 perform 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, in particular, 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 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 (e.g. $\mathcal{M}^8$ on (b)). The gradient of these moments is 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 $\boldsymbol{D}_3$) contains the axes of the symmetries of the shape. $\boldsymbol{D}_1$ is 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 the 48 symmetries whatever the coordinate system the cube is originaly expressed in.

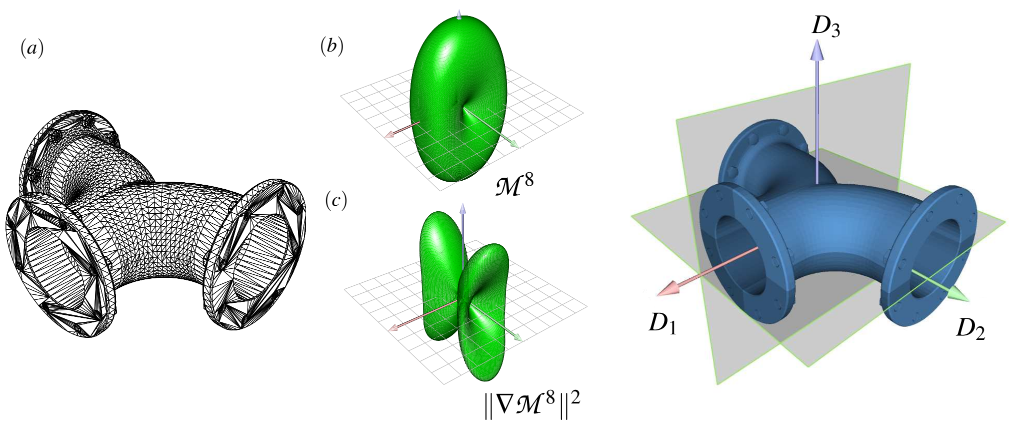


Figure 2: Extraction of symmetries for a single shape. Starting from the original shape (a), generalized moments (b) and their gradients (c) are computed. The set of their extrema direction contains the axes of the symmetries of the shape, depicted at right. Here, both mirror-symmetries have been found as well as the 2-fold rotational symmetry. Note that the original shape is neither convex nor star-shaped.

5.5 Discussion

Although it works fairly well, the algorithm presented in this section has the disadvantage that moment functions are limited in frequency, and thus may not capture enough information about the shape. This is specially true for nearly-symmetric complex shapes such as the one presented on Figure 3, where a vertical mirror-symmetry will most probably be detected in all the moments, and may not be filtered out by the symmetry measure because differences occur at a very small scale between the wheels. This may be considered an interesting feature of the method when approximate symmetries are searched for. Otherwise, this is a limitation of the method.

Moreover, the $\mathcal{M}^{2p}$ functions being 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, i.e. 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 Property 2), these extrema become non-strict extrema for small values of p, and $\mathcal{M}^{2p}$ is forced to be constant on a sub-domain of non null dimension. This is what happens e.g. for the cube, in which case $\mathcal{M}^2$ is a constant function.

6 Finding symmetries of assembled objects

In this section, we present a *constructive* algorithm which recovers the symmetries of composite objects, built from simpler parts which we call *tiles*. This algorithm starts from the symmetries of each tile to recover the symmetries of the whole shape. Applying the constructive algorithm on a complex shape is much more accurate than using the algorithm presented in Section 5 on the whole shape at once, because it pushes the accuracy issues down to a smaller scale in the object. In addition, the constructive algorithm constitutes a preliminary tool for automatic instantiation inside a scene, as demonstrated in Section 7.

After computing the tiles (Section 6.1), the constructive algorithm looks for symmetries of the whole shape by first detecting which tiles are similar up to a rigid transform (Section 6.2). Then it progressively identifies symmetries of the composite shape, by recursively refining the set of transforms which map a part of the composite shape onto another part, at each step increasing these parts until it has built the set of isometries which map the complete shape onto itself (Section 6.3).

6.1 Cutting object into tiles

The task of dividing a complex object into simpler sub-parts is a fundamental problem in various disciplines. In the computer graphics community, several approaches have been proposed for automatic mesh segmentation (e.g. Katz and Tal [2003]) as well as interactive methods (e.g. [Zöckler et al. 2000; Lee and Lee 2002]). In this paper, we rely on a simple automatic decomposition into *tiles*, which are defined as maximal sets of edge-connected polygons (see Figure 3). An octree is first filled with the vertex positions of all triangles in the model. This octree is then used to recover all triangles which share each edge. Neighbor triangles are then assigned to the same tile. Although very simple, this approach gives very good result, since the process of constructing those tiles is relevant with the way 3D-information are usually stored.

The overall computational complexity of this decomposition, starting from a polygon soup is $O(Nlog(N))$, where N stands for the number of triangles in the model.

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 $C_{2l,m}^{2p}$ 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:

$$D_{2p} = \left[d_0^0, d_0^2, d_2^2, \dots, d_0^{2p}, d_2^{2p} \dots d_{2p}^{2p}\right] \quad (8)$$

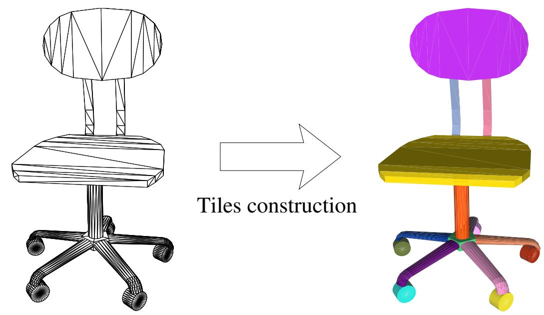


Figure 3: For the constructive algorithm, a preliminary scene analysis determines maximal sets of edge-connected polygons, called *tiles*. The generalized even moment functions and the symmetries are computed for each tile. This information is combined into the symmetries of the whole shape.

Table I. 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 %

with

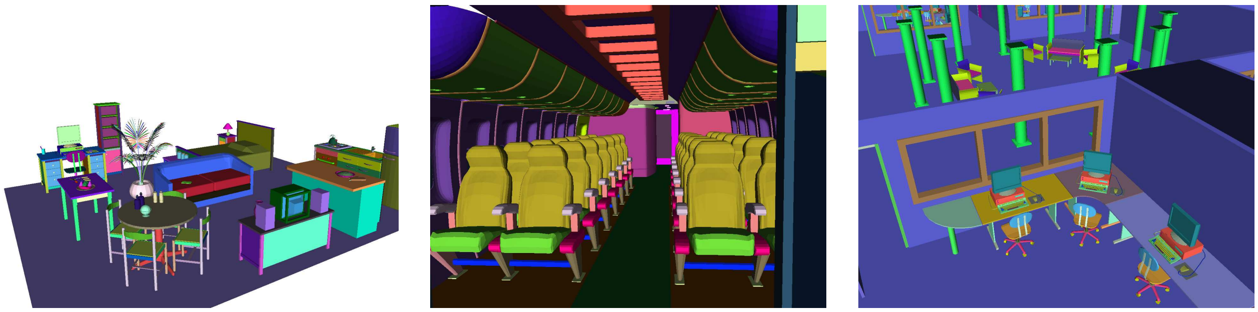
$$d_{2l}^{2k} = \sum_{-2l \leq m \leq 2l} \left(C_{2l,m}^{2k}\right)^2 \tag{9}$$

It has been shown by Kazhdan et al. [2003] that d_l^k, as defined in Equation 9, does not depend on the coordinate system the spherical harmonic decomposition is expressed in. This means that each d_{2l}^{2p}, and therefore D_{2p} itself, is not modified by isometric transforms of the shape. Mirror-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:

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

This shape descriptor can produce false positives, i.e. tiles that are not congruent but have the same descriptor, but no 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. Table I presents the average “precision” value, that is the percentage of matched tiles that are identical up to an isometry for a set of architectural scenes we used in our tests (Figure 4).



Room
39,557 Polygons

Plane
182,224 Polygons

Studio
515,977 Polygons

Figure 4: Scenes used for testing the tile congruency descriptor. In each scene the descriptor has been used to detect tiles with similar geometry (but possibly different meshes) up to a rigid transform. Congruent tiles are shown with the same color.

Table II. 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 %

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 II.

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 $\mathcal{S}_1 \in \mathcal{S} \setminus \mathcal{H}_1$ 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 on 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 6 *: 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:*

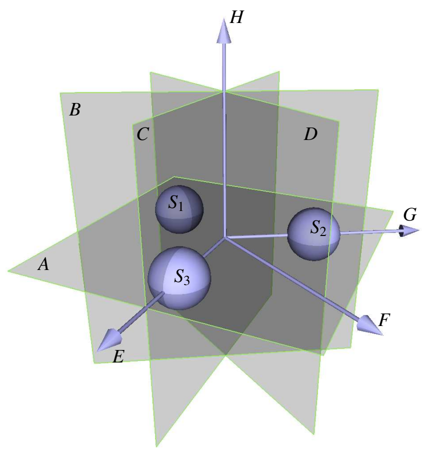
$$K = JT^{-1}AT \qquad \textit{with} \qquad A \in G_{\mathcal{S}_1} \quad \textit{such that} \quad A(\boldsymbol{g}_1 - \boldsymbol{g}) = \boldsymbol{g}_1 - \boldsymbol{g} \tag{10}$$

where $G_{\mathcal{S}_1}$ is the group of symmetries of $\mathcal{S}_1$ and T is the translation of vector $\boldsymbol{g} - \boldsymbol{g}_1$ (please 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 10.

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

Figure 5: 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. It must be noticed here 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:

$$\boldsymbol{X} \mapsto \boldsymbol{X} \qquad \boldsymbol{Y} \mapsto -\boldsymbol{Y} \qquad \boldsymbol{Z} \mapsto -\boldsymbol{Z} \tag{11}$$

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 5). It must be noticed on 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 6 presents a very simple example of a shape (a plier) 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: $\mathcal{S}_1$

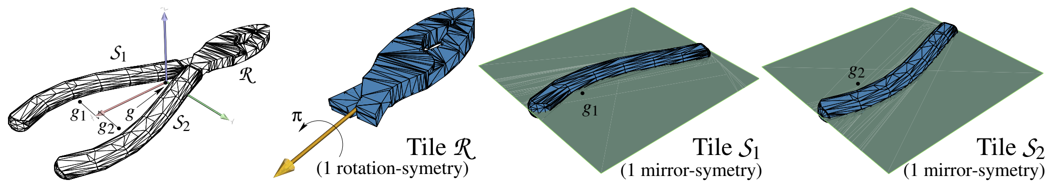


Figure 6: Illustration of the constructive algorithm on a very simple example: from the symmetries of the 3 parts of the object, the symmetries of the whole object are recovered. See text of Section 6.4 for a detailed explanation.

and $\mathcal{S}_2$. Figure 6 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 6.

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.

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.

7 Results and Applications

Figure 7 shows applications of the constructive algorithm presented in Section 6. At left, a composite lamp object (40,000 polygons) is shown along with its global symmetries (four mirror-symmetries and a 4-fold rotational-symmetry). These symmetries were computed

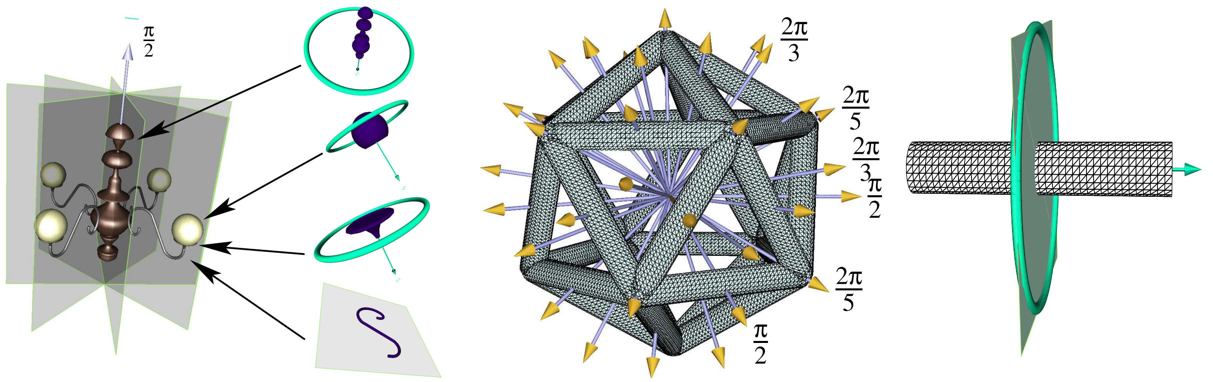


Figure 7: *Left:* a composite lamp object (40,000 polygons) shown along with its global symmetries (four mirror-symmetries and a 4-fold rotational-symmetry) computed using the constructive algorithm described in Section 6. *Middle*: a complex model which has the same group of symmetries than the icosahedron. The constructive algorithm retrieved all 46 distinct axes of rotational-symmetries using the rotational and mirror symmetries of each tile (at *right*). 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.

from the symmetries of each of the 13 sub-parts. These in turn were separately computed using the algorithm presented in Section 5.

The center image presents a complex model which has the same group of symmetries as an icosahedron. The constructive algorithm retrieved the 46 distinct axis of rotational-symmetries using the symmetries of each tile displayed at right (i.e 1 revolution-symmetry and 1 mirror-symmetry). Directly applying the first algorithm on the whole shape shows that $\mathcal{M}^2$ and $\mathcal{M}^4$ are constant functions, but using $\mathcal{M}^6$ and $\mathcal{M}^8$ still produces a correct result. Conversely, applying the first algorithm to the chair of Figure 3 in one piece produces a mirror-symmetry around the vertical plane, which is a false positive because the wheels are not exactly symmetric. The constructive algorithm in turn, does not find such a symmetry, which is correct.

Computation times (in seconds) for the models shown in this paper are given in Table III, as well as the models complexity. They were measured on a machine equipped with a 2.4 GHz processor with 512 MB 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 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.

Table III. Computation times in seconds for the different steps of our algorithm, for the models shown in this paper.
*Global computation time for moments of order 2 to 8

Model	Plier	Chair	Lamp	Icos.
# polygons	1940	1122	39550	46800
# tiles	3	17	13	30
Computing moments*	0.9	0.8	18.2	22.7
Finding parameters	0.4	1.0	1.2	2.0
Checking candidates	2.3	1.7	7.4	7.9
Constructive algo.	0.001	0.001	0.05	1.2
Total	3.601	3.501	26.85	33.8

Figure 8: View of the three 3D models used in our robustness experiments, shown with their symmetries. For the sake of clarity, we chose models with only one symmetry each.

7.1 Robustness Results

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 models are shown in Figure 8 as well as their symmetry. For 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 considere shapes as a single entities and use the first algorithm presented in Section 5 to detect these symmetries. To evaluate the fiability of the results, we compute the angular deviation between the found axis of symmety and the real one i.e. compute 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 9).

The results of these experiments show that for small variations, our method has approximatively linear dependency regarding noise and deliver high-quality results even for non-perfect 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.

7.2 Applications

Geometry compression

Our algorithm can be used for model compression. 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. Although complex models often do not present a symmetry, compression can usually be used on some sub-parts of the model.

The ability to compress a model by storing only the significant parts is provided by some recent 3D file formats such as X3D.

Mesh Editing

It might be interesting, when an object presents symmetries to reflect these properties on the object topology i.e. remesh the object with respect to its symmetries. Since a coherent remeshing allows to establish a correspondence between model vertices, it is then straightforward to create a Mesh Editing system that allow the user to modify a 3D object under constraint given by the symmetries of the original object, as depicted in Figure 10.

Instantiation

The constructive algorithm presented in section 6 automatically detects instantiation relationships between tiles into a composite shape. In fact it appears that our algorithm can be used for automatic instantiation.

Indeed, we have developed a constructive instancing 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 to know the symmetries of the tiles, and to maintain 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.

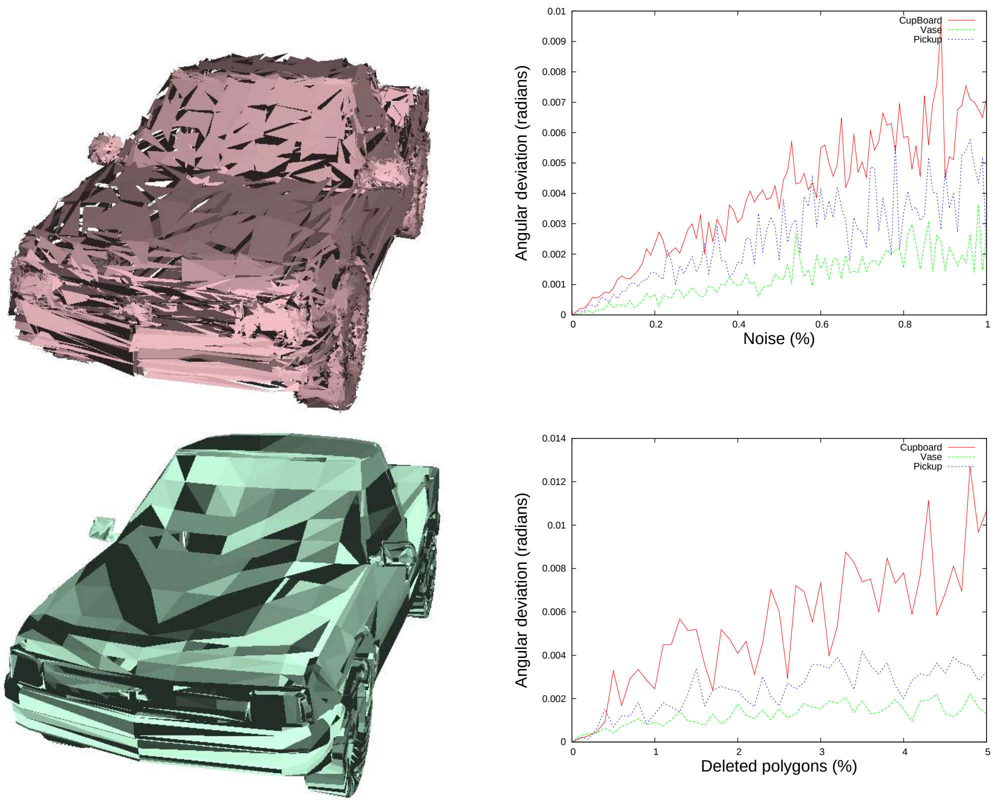


Figure 9: We test the sensitivity of the method to noise by progressively increasing noise magnitude and letting the algorithm detect the symmetry for each of our three test models. We evaluate the accuracy of the results by computing the angular deviation between the axis found and the axis of the symmetry 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 tests models. Left figure shows a noisy "pickup" model with a noise magnitude of 1% and right figure show angular deviation evolution for the three models, for magnitude ranging from 0% to 1%.*Bottom row:* We randomly delete polygons of the models. Left figure shows noisy pickup obtained by deleting 5% of the polygons and right figure show angular deviation evolution by deleting 0% to 5% of the polygons of the three models. As can be seen on the curve, for small variation of the models, our method has approximatively linear dependency regarding noise and delivers high-quality results even for non-perfect symmetries.

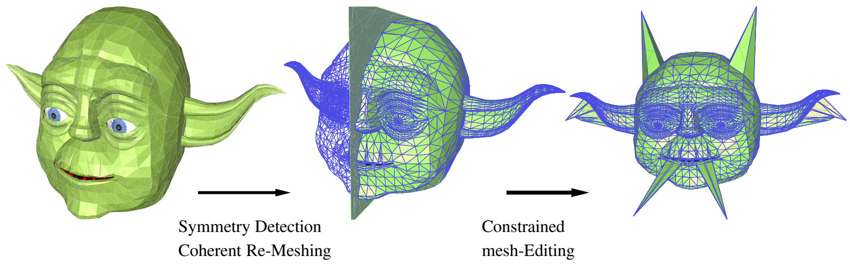


Figure 10: Starting from an object in arbitrary orientation, we detect symmetries of the shape (in the figure, a planar symmetry) and use it to remesh the objects with respect to these symmetries. Then, a user can easily edit the mesh and modify it while keeping the symmetries of the initial shape.

8 Discussion

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

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 paper: (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. Spherical harmonics being orthogonal and easily rotated, finding symmetries on the moment functions translates into simple relationships between the coeficients. (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 4 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 capture 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 an other 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 7.

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 axe of the symmetry.

The work of Kazhdan et al. [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 paper [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 approximately require 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.

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 from 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.

Appendix: Proofs

PROOF OF PROPERTY 1. let A be an isometry which lets a shape $\mathcal{S}$ 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 isometry being a unit transform, its determinant is ± 1 and thus vanishes. The cross product is also left unchanged by applying an isometry to each of its terms. Line 4: because $A\mathcal{S} = \mathcal{S}$ we also have $\mathcal{S} = A^{-1}S$.

The isometry A is thus also a symmetry of the $\mathcal{M}^{2p}$ moment functions. $\square$

PROOF OF PROPERTY 2. Let A be an isometry 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:

$$\boldsymbol{d}_{A\boldsymbol{v}} = A\boldsymbol{d}_{\boldsymbol{v}} = \boldsymbol{d}_{\boldsymbol{v}} \tag{12}$$

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

$$\boldsymbol{d}_{-\boldsymbol{v}} = -\boldsymbol{d}_{\boldsymbol{v}} \tag{13}$$

So, if A is a symmetry, we have $A\boldsymbol{v} = -\boldsymbol{v}$. From Equations 12 and 13 it now comes that $\boldsymbol{d}_{\boldsymbol{v}} = -\boldsymbol{d}_{\boldsymbol{v}}$, 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 6. Let $\mathcal{S}$ and $\mathcal{R}$ be two shapes, identical up to an isometric transform. Let J be an isometry such that $J\mathcal{S} = \mathcal{R}$. Let T be the translation of vector $-u_{\mathcal{S}}$ with $\boldsymbol{u}_{\mathcal{S}} = \boldsymbol{g}_{\mathcal{S}} - \boldsymbol{g}$, $\boldsymbol{g}_{\mathcal{S}}$ being the center of mass of $\mathcal{S}$, and $\boldsymbol{g}$ the origin of the coordinate system into which J is applied.

— Let $A \in G_{\mathcal{S}}$ 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

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

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

— Let K be an isometry 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 isometry. 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 $A(\boldsymbol{g} - \boldsymbol{g}_{\mathcal{S}}) = (\boldsymbol{g} - \boldsymbol{g}_{\mathcal{S}})$.

□

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, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. Aug.

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 *CG '91: Proceedings of the International Workshop on Computational Geometry - Methods, Algorithms and Applications*. Lecture Notes in Computer Science, vol. 553. Springer, London, UK, 113–121.

Katz, S. and Tal, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graphics 22,* 3 (July), 954–961.

Kazhdan, M. M., Funkhouser, T. A., and Rusinkiewicz, S. 2003. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *SGP '03: Proceedings of the 2003 Eurographics/ACM Siggraph Symposium on Geometry Processing*. 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 *SGP '04: Proceedings of the 2004 Eurographics/ACM Siggraph Symposium on Geometry Processing*. 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.

Lee, Y. and Lee, S. 2002. Geometric snakes for triangular meshes. *Computer Graphics Forum 21,* 3, 229–238.

Minovic, P., Ishikawa, S., and Kato, K. 1993. Symmetry identification of a 3-D object represented by octree. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15,* 5, 507–514.

Prince, E. 2004. *Mathematical Techniques in Crystallography and Materials Science*, 3rd ed. Springer, Berlin, Germany, London, UK.

Ramamoorthi, R. and Hanrahan, P. 2004. A signal-processing framework for reflection. *ACM Trans. Graphics 23,* 4, 1004–1042.

Sun, C. and Sherrah, J. 1997. 3D symmetry detection using extended Gaussian image. *IEEE Transactions on Pattern Analysis and Machine Intelligence 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. *The Visual Computer 1*, 37–48.

Zabrodsky, H., Peleg, S., and Avnir, D. 1995. Symmetry as a continuous feature. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17,* 12, 1154–1166.

Zöckler, M., Stalling, D., and Hege, H.-C. 2000. Fast and intuitive generation of geometric shape transitions. *The Visual Computer 16,* 5, 241–253.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399