Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Reconstruction of symmetric models composed of analytic curves and surfaces from point cloud*

Qing WANG†, Wei-dong ZHU†‡, Ying-lin KE

(*State Key Lab of Fluid Power Transmission and Control, Zhejiang University, Hangzhou 310027, China*)

†E-mail: uphover@163.com; turbozhu@hotmail.com

Received Apr. 27, 2008; revision accepted July 24, 2008

Abstract: This paper presents a method to reconstruct symmetric geometric models from point cloud with inherent symmetric structure. Symmetry types commonly found in engineering parts, i.e., translational, reflectional and rotational symmetries are considered. The reconstruction problem is formulated as a constrained optimization, where the objective function is the sum of squared distances of points to the model, and constraints are enforced to keep geometric relationships in the model. First, the explicit representations of symmetric models are presented. Then, by using the concept of parameterized points (where the coordinate components are represented as functions rather than constants), the distances of points to symmetric models are deduced. With these distance functions, symmetry information, for both 2D and 3D models, is uniformly represented in the process of reconstruction. The constrained optimization problem is solved by a standard nonlinear optimization method. Owing to the explicit representation of symmetry information, the computational complexity of our method is reduced greatly. Finally, examples are given to demonstrate the application of the proposed method.

Key words: Reverse engineering, Model reconstruction, Constrained optimization, Symmetry
doi:10.1631/jzus.A0820324 **Document code:** A **CLC number:** TP391.7

INTRODUCTION

Geometric model reconstruction from point cloud is an important step in reverse engineering (Várady *et al.*, 1997). To ensure the usefulness of a reconstructed model in engineering applications, properties such as parallelism, perpendicularity, tangency, symmetry, etc., should be preserved. Ensuring the preservation of these properties in model reconstruction is called capturing design intention in state-of-the-art reverse engineering methods. Thompson *et al.*(1999) used manufacturing features as geometric primitives in the reconstruction of mechanical parts. Since geometric relationships are inherently represented in manufacturing features, they are retained in the reconstructed model. Werghi *et al.*(1999) presented a framework for the integration of geometric relationships in the reconstruction of models composed of quadric surfaces. The geometric relationships are formulated as a set of constraint equations linking the parameters that describe the geometric model. A constrained optimization problem was set up based on systematically represented objective functions and constraint equations, and was solved by a standard constrained nonlinear optimization method in which the Levenberg-Marquardt Method method was applied. Benkő *et al.*(2002) used different geometric representations and numerical methods to solve the constrained fitting problem of analytic curves and surfaces. The distance functions are written in a form that separates terms describing parameters from terms regarding data points, so that the objective functions can be evaluated efficiently and the time cost of the optimization process can be reduced. Furthermore, Benkő's method is capable of rejecting conflict constraints in the reconstruction

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 50575098) and China Postdoctoral Science Foundation (No. 20070421176)

process. In contrast to Benkő's approach, Langbein (2003) studied the detection of design intentions from initial B-rep models. First a set of geometric constraints are derived from the initial B-rep model, then a subset of geometric constraints which likely represent the design intention of the original model are selected based on a solvability test. The model is adjusted according to this subset of constraints, without further referring to the point cloud.

Symmetry, as a geometric relationship, often involves many geometric primitives, appears frequently in man-made objects and should be preserved in reconstructed models. Symmetry is the harmony of proportion of objects, and includes various types, such as reflectional symmetry, translational symmetry, and rotational symmetry. Weyl (1952) discussed the application of symmetries in the arts in the non-organic and organic world; he used the theory of symmetry groups to describe these symmetries and the relationships among them. Symmetry is also important in engineering applications. For example, to meet the requirement of dynamic equilibrium, strict rotational symmetry must be kept in a mechanical system. For various reasons (functional, esthetic, etc.), symmetries exist widely in automobile, airplane, ship and household appliances. In fact, nearly all commercial CAD software provides methods for the design of symmetric models. For example, in Unigraphics, 'Rotate About a Point', 'Mirror Through a Line', 'Rectangular Array', 'Circular Array', 'Rotate About a Line', 'Mirror Through a Plane', etc., are available for creating symmetric 2D and 3D models. In industry, nearly all manufactured products rely on CAD models. Thus, symmetric models are prerequisite for the manufacturing of symmetric parts. Furthermore, symmetry properties in CAD models can be used in computer aided engineering software (such as Ansys) to simplify finite element analysis.

In the literature about model reconstruction (Thompson *et al.*, 1999; Werghi *et al.*, 1999; Benkő *et al.*, 2002), symmetries, like other geometric relationships, are represented as constraints. However, if symmetry information is indeed represented as constraints, the models will be 'over-parameterized'. When we use the standard 10-coefficient quadratic equation to represent a cylinder in space, to ensure the 10 parameters represent a cylinder, we have to impose three more constraint equations (Werghi *et al.*, 1999). We use a simple example to demonstrate a similar situation when a symmetric model is based on constraints. Suppose n translational symmetric circles are to be described (Fig.1). Each circle is represented by three parameters, i.e., the center (x_i, y_i) and the radius r_i. However, to keep these circles to be translationally symmetric, constraints are required to restrict the degrees of freedom (DoFs). More precisely, to force the model to be symmetric, we need $n-1$ equal radius constraints $\{EqualRad(r_i, r_{i+1})=0, i=1, 2, ..., n-1\}$, $n-2$ collinear center constraints $\{Collinear(x_i, y_i, x_{i+1}, y_{i+1}, x_{i+2}, y_{i+2})=0, i=1, 2, ..., n-2\}$, and $n-2$ equal distance constraints $\{EqualDist(x_i, y_i, x_{i+1}, y_{i+1}, x_{i+2}, y_{i+2})=0, i=1, 2, ..., n-2\}$. As a result, $3n-(n-1)-(n-2)-(n-2)=5$ DoFs are used to describe the system. Another method to represent these circles is called '1D array', which is frequently used in conventional CAD design (Fig.2). First, a base circle (center (x, y), radius r) is created, then other circles are created by translations with a series of translational vectors, which are $i=1, 2, ..., n-1$ times a base vector. The DoFs used are again 5, 3 for the base circle and 2 for the translational vector. Both methods can represent a set of translational symmetric circles, but the second method is more efficient owing to the explicit representation of symmetry information. By 'efficiency', here we mean that the model can be described with fewer parameters and constraints. For a constrained optimization problem, the computational complexity is about $O(n^2)$, where n is the number of parameters. Furthermore, the evaluation of constraint equations costs a lot of computation. Thus, explicit representation is more appropriate for reconstruction problems but this kind of representation has not been used in previous constrained reconstruction methods. In this paper, we investigate explicit representations for various types of symmetric models, and use these representations in model reconstruction. Specifically, for 2D models, translational symmetry, reflectional symmetry and rotational symmetry are considered; for 3D models, translational symmetry and reflectional symmetry are considered.

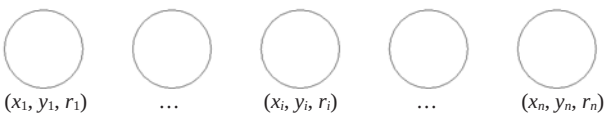


Fig.1 Constraint-based representation of translational symmetry

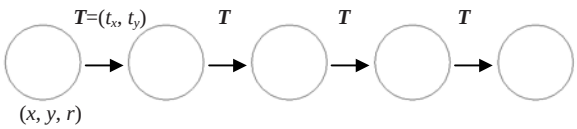
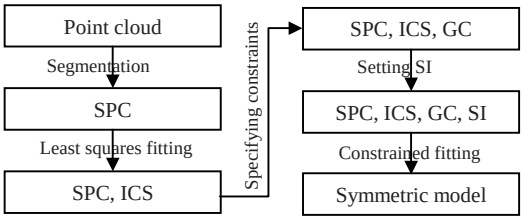


Fig.2 Explicit representation of translational symmetry

In our previous research (Ke *et al.*, 2006), we approached the reconstruction of extruded surfaces, rotational surfaces and lofted surfaces via a 2D constrained fitting method. In this paper, we focused our attention on the problem of reconstructing symmetric models in the framework of constrained fitting. The geometric primitives used in our method are restricted to some analytic curves and surfaces, i.e., lines and arc curves in 2D, planes, spheres, and cylinders in 3D. Since a high percentage of mechanical parts are composed solely of analytic curves and surfaces, our method would be useful in practical applications. We assume that the point data have been segmented into sub-point sets, each corresponding to one geometric primitive. The initial curves and surfaces are created by fitting each set of points independently with a standard least squares method, or by adoption from a nominal model. The geometric constraints (coincidence, tangency, etc.) between the primitives of the base component (in cases where the base component is composed of several primitives) have been specified by user interaction. Symmetry detection is an important problem for some applications in computer vision (Sun and Sherrah, 1997; Shen *et al.*, 1999; Mitra *et al.*, 2006). Symmetry information of geometric models is a prerequisite for the reconstruction of symmetric models from point clouds. So, symmetry detection is also important for reverse engineering applications. Considerable work has been carried out in detecting symmetries from discrete point subsets (Mills *et al.*, 2001; Langbein *et al.*, 2004; Li *et al.*, 2007; 2008). However, to alleviate the requirements of complex algorithms for the detection of symmetry and to focus our attention on the reconstruction problem, in our study the symmetry information is designated by users interactively. Users can detect the symmetries by hand and obtain the symmetry information from point cloud using the methods mentioned above. The symmetry information includes the type of symmetry, the grouping information and the ordering information for primitives in each component. The symmetry structure can be described as (*SymmetryType*, *Component*$_i$) for a 1D array (i=0, 1, ..., n), rotational symmetry (i=0, 1, ..., n), and reflectional symmetry (i=0, 1); (*SymmetryType*, *Component*$_{ij}$) for a 2D array (i=0, 1, ..., m; j=0, 1, ..., n). Each component contains k primitives $\{O_i|i=1, 2, ..., k\}$, which are ordered in the same manner for all components. Fig.3 shows the steps in the reconstruction of symmetric models. Figs.4~10 illustrate these symmetry structures when k=1 in the next section.



SPC: Sub-point cloud; ICS: Initial curves or surfaces;
GC: Geometric constraints; SI: Symmetry information

Fig.3 Operation sequence for the reconstruction of symmetric models

Here, we list the structure of this paper. In Section 2, the explicit representations of symmetric models are presented. In Section 3, we describe all related aspects about constrained fitting of symmetric models, i.e., representation of geometric primitives, objective functions for explicitly represented symmetric models, constraints, initial values, and numerical optimization methods. Several examples are given in Section 4 to show the behavior of our algorithm and its usefulness in applications. Conclusions are drawn in Section 5. Distance functions of relevance to various symmetric models are listed in the Appendix.

EXPLICIT REPRESENTATION OF SYMMETRIC MODELS

The symmetric models involved in this paper reveal some important cases that may occur in engineering applications. We do not try to cover all types of symmetry in these models, and the meaning of symmetry in this paper is not strictly equivalent to the mathematical definitions (Weyl, 1952). A model is symmetric if one component is congruent to another component when a symmetry transformation (translation, reflection, or rotation) is applied. In other words, a symmetric model can be created by symmetry transformations of a base component (one or

several geometric primitives). As a result, for an explicitly represented symmetric model, two kinds of parameters are used: those for the base geometric primitive(s) and those for objects describing symmetry transformations, i.e., translational vectors for translational symmetry, symmetry axes or planes for reflectional symmetry, and rotational centers for rotational symmetry.

1D and 2D translational symmetry in 2D

1. 1D array

A 1D array is described by a base component (in Fig.4, it is a circle) and a translational vector $\boldsymbol{T}=[u_x, u_y]^{\mathrm{T}}$.

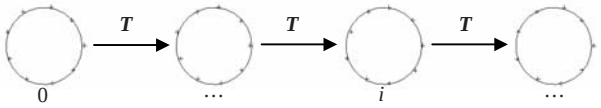


Fig.4 1D array in 2D

2. 2D array

A 2D array is described by a base component (in Fig.5, it is a circle) and two translational vectors $\boldsymbol{T}_1=[u_x, u_y]^{\mathrm{T}}$ and $\boldsymbol{T}_2=[v_x, v_y]^{\mathrm{T}}$.

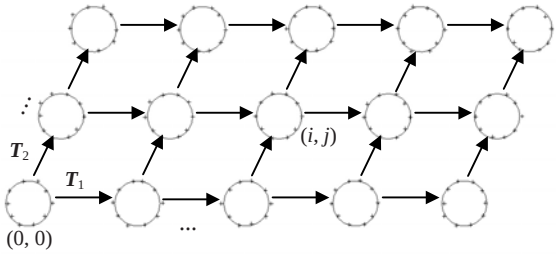


Fig.5 2D array in 2D

Reflectional symmetry in 2D

Reflectional symmetry, also known as 'bilateral symmetry' or 'mirror symmetry', is indicated by a symmetry axis (Fig.6). A symmetry axis is actually an infinite line and there are various methods to represent a line. A suitable representation should be free from singularity and have the smallest number of parameters. Like (Kós *et al.*, 1999), we define an infinite line by ($\boldsymbol{N}$, d), with a normalization constraint $|\boldsymbol{N}|=1$, where $\boldsymbol{N}$ is the normal vector to the line and d is the signed distance of the origin to the line. More explicitly, the line is represented as a triple (n_x, n_y, d) with $n_x^2+n_y^2=1$.

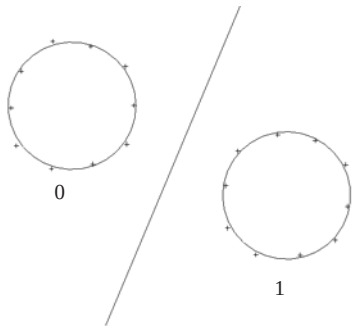


Fig.6 Reflectional symmetry in 2D

Rotational symmetry in 2D

Rotational symmetry can be represented as a base component, a rotation center (ξ, η), and a rotation angle φ between two adjacent components (Fig.7). In a rotational symmetry, $\varphi=2\pi/N$, where N is an integer. Since N is always detectable from initial curves, φ is a constant in the constrained fitting stage.

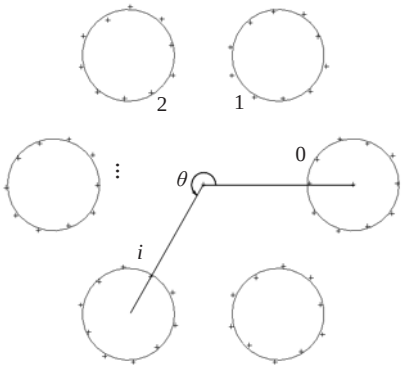


Fig.7 Rotational symmetry in 2D

Translational symmetry in 3D

1. 1D array

A 1D array is described by a base component (in Fig.8, it is a cylinder) and a translational vector $\boldsymbol{T}=[u_x, u_y, u_z]^{\mathrm{T}}$.

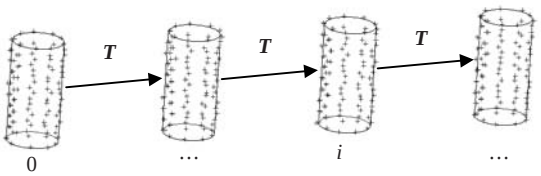


Fig.8 1D array in 3D

2. 2D array

A 2D array is described by a base component (in Fig.9, it is a cylinder) and two translational vectors $\boldsymbol{T}_1=[u_x, u_y, u_z]^{\mathrm{T}}$ and $\boldsymbol{T}_2=[v_x, v_y, v_z]^{\mathrm{T}}$.

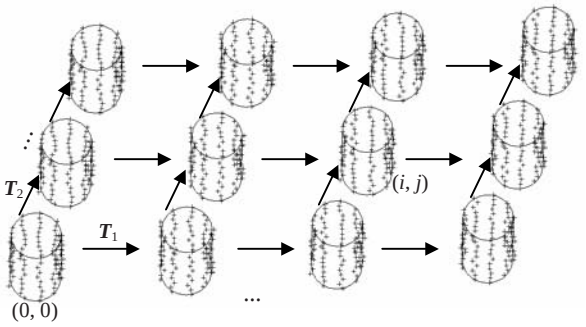


Fig.9 2D array in 3D

Reflectional symmetry in 3D

Reflectional symmetry in 3D is characterized by a plane (Fig.10). The symmetry plane is defined by ($\boldsymbol{N}$, d), with a normalization constraint $|\boldsymbol{N}|=1$, where $\boldsymbol{N}$ is the normal vector to the plane and d is the signed distance of the origin to the plane. More explicitly, the plane is represented as a triple (n_x, n_y, n_z, d) with $n_x^2+n_y^2+n_z^2=1$.

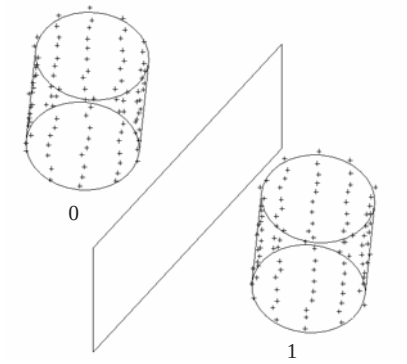


Fig.10 Reflectional symmetry in 3D

RECONSTRUCTION OF SYMMETRIC MODELS BY CONSTRAINED FITTING

Suppose a geometric model is described by a set of parameters $\boldsymbol{X}$. To reconstruct the geometric model from point clouds, a fitting method is usually used. With the fitting method, a set of parameters that best fits the point cloud can be estimated. Least squared distances of points to the geometric primitives (curves or surfaces) are commonly adopted in the formulation of objective functions for fitting. An appropriate distance function should satisfy two conditions: it should be close to the Euclidean distance and it can be computed efficiently. The distance functions rely heavily on the representation of geometric primitives. Since polynomials are easy to compute and close to the Euclidean distances under appropriate quadratic normalization conditions, we adopt the representations for geometric primitives from the literature (Pratt, 1987; Benkő *et al.*, 2002).

Representation of geometric primitives and distance functions

Although geometric fitting with Euclidean distance is more accurate and robust (Faber and Fisher, 2001), the computational load is much heavier than using algebraic distance in geometric fitting. In this paper, carefully chosen algebraic distances are used to describe the distances between a point and geometric primitives.

1. Line

A line is represented by $l_0x+l_1y+l_2=0$ and the distance function is $d=l_0x+l_1y+l_2$, with normalization constraint $l_1^2+l_2^2-1=0$ (Pratt, 1987).

2. Circle

The equation of a circle is $c_0(x^2+y^2)+c_1x+c_2y+c_3=0$ and the distance function is $d=c_0(x^2+y^2)+c_1x+c_2y+c_3$ with normalization constraint $c_1^2+c_2^2-4c_0c_3-1=0$. Its radius is $1/|2c_0|$, and the center is $(-c_1/(2c_0), -c_2/(2c_0))$ (Pratt, 1987).

3. Plane

A plane is represented by its usually implicit equation $q_0x+q_1y+q_2z+q_3=0$. The distance function is $d=q_0x+q_1y+q_2z+q_3$, under normalization condition $q_0^2+q_1^2+q_2^2-1=0$ (Pratt, 1987).

4. Sphere

A sphere is represented by equation $s_0(x_2+y_2+z_2)+s_1x+s_2y+s_3z+s_4=0$. The distance function is $d=s_0(x_2+y_2+z_2)+s_1x+s_2y+s_3z+s_4$ under normalization constraint $s_1^2+s_2^2+s_3^2-4s_0s_4-1=0$. Its radius is $1/|2s_0|$, and the center is $(-s_1/(2s_0), -s_2/(2s_0), -s_3/(2s_0))$ (Pratt, 1987).

5. Cylinder

A cylinder is represented by $c_3[x^2+y^2+z^2-(c_0x+c_1y+c_2z)^2]+c_4x+c_5y+c_6z+c_7=0$, the distance function is $d=c_3[x^2+y^2+z^2-(c_0x+c_1y+c_2z)^2]+c_4x+c_5y+c_6z+c_7$, under normalization constraints, $c_0^2+c_1^2+c_2^2-1=0$, $c_0c_4+c_1c_5+c_2c_6=0$, $c_4^2+c_5^2+c_6^2-4c_3c_7-1=0$. Its radius is $1/|2c_3|$, the nearest point of its axis to the origin is $-(c_4, c_5, c_6)/(2c_3)$ and the direction of its axis is (c_0, c_1, c_2) (Benkő *et al.*, 2002).

To further reduce the amount of computation, we write the distance function between an object (described by parameters $\boldsymbol{X}$) and a point in a separated form $d=\boldsymbol{S}^{\mathrm{T}}\boldsymbol{P}=\boldsymbol{P}^{\mathrm{T}}\boldsymbol{S}$, where $\boldsymbol{S}$ is a vector defined by the

parameters of geometric primitives and $\boldsymbol{P}$ is a vector defined by the coordinates of a data point. For example, in the case of a line, $\boldsymbol{S}=[l_0, l_1, l_2]^{\mathrm{T}}$, $\boldsymbol{P}=[x, y, 1]^{\mathrm{T}}$, and the objective function to be minimized is $J(\boldsymbol{X})=\sum d_i=\sum(\boldsymbol{S}^{\mathrm{T}}\boldsymbol{P}_i)(\boldsymbol{P}_i^{\mathrm{T}}\boldsymbol{S})=\boldsymbol{S}^{\mathrm{T}}\sum(\boldsymbol{P}_i\boldsymbol{P}_i^{\mathrm{T}})\boldsymbol{S}$. The matrix $\boldsymbol{M}=\sum\boldsymbol{P}_i\boldsymbol{P}_i^{\mathrm{T}}$ depends only on the coordinates of the data points and needs to be computed only once before iteration starts. In iteration, the objective function and its derivatives are $J=\boldsymbol{S}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{S}$, $J'=\boldsymbol{S}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{S}'$ and $J''=2((\boldsymbol{S}')^{\mathrm{T}}\boldsymbol{M}\boldsymbol{S}'+\boldsymbol{S}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{S}'')$, respectively (Benkő *et al.*, 2002). For the $\boldsymbol{S}$ and $\boldsymbol{P}$ of each distance function, refer to Table A1 in the Appendix.

Distance functions for explicitly represented symmetric models

We use a 1D translational symmetric model to illustrate the deduction of distance functions for symmetric models (Fig.11).

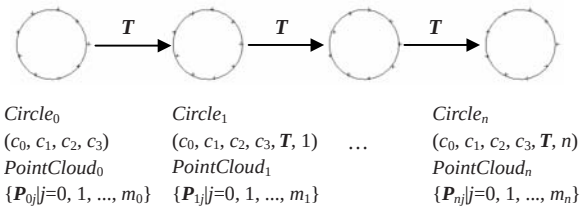


Fig.11 Translational symmetric circles and their corresponding point clouds

For the base component $Circle_0$ and its associated points $PointCloud_0$, the objective function is $J_0=\sum_{j=0}^{m_0} dist^2(Circle_0, \boldsymbol{P}_{0j})$. The objective functions of transformed components and their related points are $J_i=\sum_{j=0}^{m_j} dist^2(Circle_i, \boldsymbol{P}_{ij})$, i=1, 2, ..., n.

Since translation is isometric, J_i will not change if the same translation is applied to both $Circle_i$ and $PointCloud_i$. Applying a translation $-i\boldsymbol{T}$ to both $Circle_i$ and $PointCloud_i$, we get $Circle_0$ and a new point cloud $TransformedPointCloud_i$, which depends on the parameters of the transformation. We call the new point cloud the 'parameterized point cloud' and its points the 'parameterized points'. Then the objective function J_i can be defined as the squared distance between the parameterized points and the base component $Circle_0$. The parameterized points for other types of symmetric models are deduced and listed below. By substituting these parameterized points into the distance functions in the previous subsection, the distance functions of points to symmetric models are acquired. For the separated form of these distance functions, refer to the Appendix.

1. 1D array in 2D

The symmetry structure is ($OneDimArray$, $Component_i$), i=0, 1, ..., n (Fig.4). For a point (x, y) in $Component_i$, its corresponding parameterized point is $(x^*, y^*)=(x-iu_x, y-iu_y)$, where $\boldsymbol{T}=[u_x, u_y]^{\mathrm{T}}$ is the basic translational vector.

2. 2D array in 2D

The symmetry structure is ($TwoDimArray$, $Component_{ij}$), i=0, 1, ..., n_u; j=0, 1, ..., n_v (Fig.5). For a point (x, y) in $Component_{ij}$, its corresponding parameterized point is $(x^*, y^*)=(x-iu_x-jv_x, y-iu_y-jv_y)$, where $\boldsymbol{T}_1=[u_x, u_y]^{\mathrm{T}}$ and $\boldsymbol{T}_2=[v_x, v_y]^{\mathrm{T}}$ are the basic translational vectors.

3. Reflectional symmetry in 2D

The symmetry structure is ($ReflectionalSymmetry$, $Component_i$), i=0, 1 (Fig.6). For a point (x, y) in $Component_1$, the parameterized point is deduced as follows (Fig.12a). Suppose $\boldsymbol{P}=(x, y)$ is a conventional point, $\boldsymbol{Q}=(x^*, y^*)$ is the symmetric point of $\boldsymbol{P}$ with respect to symmetry axis ($\boldsymbol{N}$, d). Then $\boldsymbol{Q}$ can be represented in parameters of $\boldsymbol{P}$ and ($\boldsymbol{N}$, d) as $\boldsymbol{Q}=\boldsymbol{P}-2(\boldsymbol{P}\cdot\boldsymbol{N}-d)\boldsymbol{N}$, yielding $(x^*, y^*)=(x-2(xn_x+yn_y-d)n_x, y-2(xn_x+yn_y-d)n_y)$.

4. Rotational symmetry in 2D

The symmetry structure is ($RotationalSymmetry$, $Component_i$), i=0, 1, ..., n (Fig.7). For a point (x, y) in $Component_i$, the parameterized point is deduced (Fig.12b). Suppose $\boldsymbol{P}=(x, y)$ is a conventional point, $\boldsymbol{Q}=(x^*, y^*)$ is the rotated point, which can be represented as $(x^*, y^*)=(x\cos\theta-y\sin\theta+(1-\cos\theta)\xi+\eta\sin\theta, x\sin\theta-y\cos\theta+(1-\cos\theta)\eta-\xi\sin\theta)$, where (ξ, η) are parameters defining the rotation center, $\theta=-i\varphi$ (negative sign for a clockwise angle) is the rotation angle, and $\varphi=2\pi/N$ is the basic rotation angle.

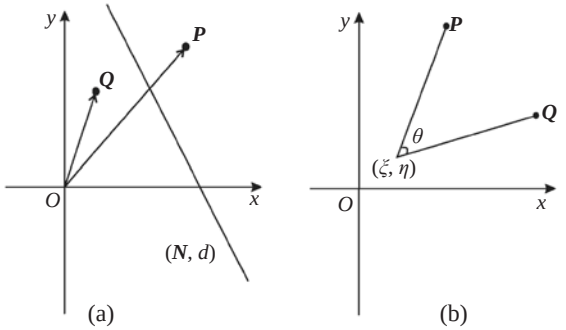


Fig.12 Parameterized point in 2D reflectional symmetry (a) and 2D rotational symmetry (b)

5. 1D array in 3D

Similar to 1D array in 2D, the symmetry structure is (*OneDimArray*, *Component*$_i$), i=0, 1, ..., n (Fig.8). For a point (x, y) in *Component*$_i$, its corresponding parameterized point is (x^*, y^*, z^*)=($x-iu_x$, $y-iu_y$, $z-iu_z$), where $\boldsymbol{T}=[u_x, u_y, u_z]^{\mathrm{T}}$ is the basic translational vector.

6. 2D array in 3D

The symmetry structure is (*TwoDimArray*, *Component*$_{ij}$), i=0, 1, ..., n_u; j=0, 1, ..., n_v (Fig.9). For a point (x, y) in *Component*$_{ij}$, its corresponding parameterized point is (x^*, y^*, z^*)=($x-iu_x-jv_x$, $y-iu_y-jv_y$, $z-iu_z-jv_z$), where $\boldsymbol{T}_1=[u_x, u_y, u_z]^{\mathrm{T}}$ and $\boldsymbol{T}_2=[v_x, v_y, v_z]^{\mathrm{T}}$ are the basic translational vectors.

7. Reflectional symmetry in 3D

As in 2D, the symmetry structure is (*ReflectionalSymmetry*, *Component*$_i$), i=0, 1 (Fig.10). For a point (x, y) in *Component*$_1$, the parameterized point is (x^*, y^*, z^*)=($x-2(xn_x+yn_y+zn_z-d)n_x$, $y-2(xn_x+yn_y+zn_z-d)n_y$, $z-2(xn_x+yn_y+zn_z-d)n_z$).

Constraints

To preserve local geometric regularities, geometric constraints must be specified on related objects. These geometric constraints are represented as algebraic equations in the fitting process. Furthermore, since polynomial distance functions are used, normalization constraints must be applied to keep the calculated distances close to the true Euclidean distances. Three types of constraints are listed as follows:

(1) Constraints between geometric primitives. When the base component is composed of more than one geometric primitive, geometric constraints such as parallel, perpendicular or tangent primitives are often imposed to ensure certain relationships among these geometric primitives. For example, the tangency constraints in Fig.13 belong to this type. The representation of these constraints can be found in (Benkő *et al.*, 2002).

(2) Constraints between geometric primitives and the symmetry parameters. For example, the constant angle constraint in Fig.13 belongs to this type. The representations of these constraints are similar to the constraints associated with geometric primitives alone.

(3) Normalization constraints. These constraints are described in the subsection of "Representation of geometric primitives and distance functions".

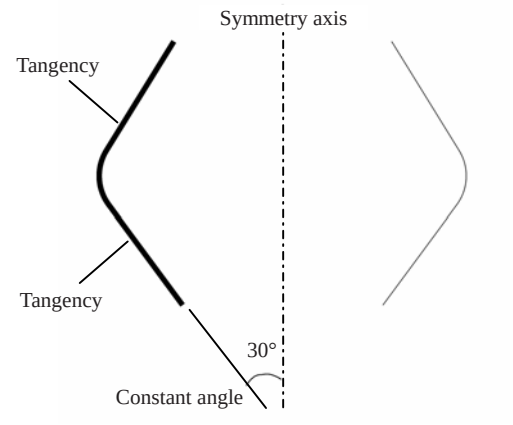


Fig.13 Constraints in symmetric model reconstruction

Initial values

Initial values of geometric primitives can be acquired by least squares fittings, or imported from nominal models. However, the initial values regarding parameters of symmetry objects need more explanation. For convenience, we name these points to be characteristic points: end points, middle points of curves or surface edges.

1. Calculation of the translational vector for a 1D array

(1) Calculate the bounding box of the first and last components, the centers of which are denoted as $\boldsymbol{C}_0$ and $\boldsymbol{C}_n$, respectively.

(2) The initial translational vector is taken as

$$\boldsymbol{T}=\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{C}_i-\boldsymbol{C}_{i-1})=\frac{1}{n}(\boldsymbol{C}_n-\boldsymbol{C}_0).$$

2. Calculation of the translational vectors for a 2D array

(1) Calculate the bounding box of boundary components, the centers of which are denoted as $\boldsymbol{C}_{ij}$, i=0, n_u; j=0, 1, ..., n_v and $\boldsymbol{C}_{ij}$, i=0, 1, ..., n_u; j=0, n_v.

(2) Calculate u- and v-directional vectors

$$\boldsymbol{T}_1=\frac{1}{n_u n_v}\sum_{j=0}^{n_v}(\boldsymbol{C}_{n_u,j}-\boldsymbol{C}_{0,j}),$$

$$\boldsymbol{T}_2=\frac{1}{n_u n_v}\sum_{i=0}^{n_u}(\boldsymbol{C}_{i,n_v}-\boldsymbol{C}_{i,0}).$$

3. Calculation of symmetry axis

(1) Calculate two sets of characteristic points corresponding to primitives from two symmetric

components $\boldsymbol{P}_i$ and $\boldsymbol{Q}_i$, i=1, 2, ..., m.

(2) The initial symmetry axis is taken as the line best fitting points $(\boldsymbol{P}_i+\boldsymbol{Q}_i)/2$, i=1, 2, ..., m.

4. Calculation of symmetry center and rotational angle

We assume that at least three components are involved in the rotational symmetry structure.

(1) Calculate a set of characteristic points $\boldsymbol{P}_i$, i=1, 2, ..., m.

(2) The initial value is taken as the center of the circle best fitting $\boldsymbol{P}_i$, i=1, 2, ..., m.

(3) With the approximate rotation center, the approximate basic rotation angle φ' can be calculated. Then the number of cycles in the whole rotation system can be determined by N=int($2\pi/\varphi'$), where the int operator calculates the integer number closest to $2\pi/\varphi'$. Finally, the base rotation angle can be determined by $\varphi=2\pi/N$.

5. Calculation of symmetry plane

(1) Calculate two sets of characteristic points $\boldsymbol{P}_i$ and $\boldsymbol{Q}_i$, i=1, 2, ..., m.

(2) The symmetry plane is taken as the plane best fitting $(\boldsymbol{P}_i+\boldsymbol{Q}_i)/2$, i=1, 2, ..., m.

Numerical optimization problem

Suppose $\boldsymbol{X}$ is a vector collecting all the parameters describing the symmetric model, then the constrained fitting problem can be formulated as

$$\min J=\sum J_i=\sum_i \boldsymbol{S}_i^{\mathrm{T}}(\boldsymbol{X})\left(\sum\nolimits_j \boldsymbol{P}_{ij}\boldsymbol{P}_{ij}^{\mathrm{T}}\right)\boldsymbol{S}_i(\boldsymbol{X}),$$

$$\text{s.t.}\quad c_r(\boldsymbol{X})=0,\ r=0,\ 1,\ ...,\ l,$$

where $\boldsymbol{S}$'s and $\boldsymbol{P}$'s are expressions listed in Table A2 in the Appendix and $c_r(\boldsymbol{X})$=0 are constraint equations. According to optimization theory (Fiacco and McCormick, 1968), the original constrained optimization problem can be converted into a series of unconstrained optimization problems. The objective function to be minimized in each iteration is

$$J=\sum_i^n \boldsymbol{S}_i^{\mathrm{T}}(\boldsymbol{X})\left(\sum_{j=1}^{n_m}\boldsymbol{P}_{ij}\boldsymbol{P}_{ij}^{\mathrm{T}}\right)\boldsymbol{S}_i(\boldsymbol{X})+\sum_{r=0}^{l}\lambda_r^k c_r^2(\boldsymbol{X}).$$

The steps are as follows:

Step 1: $k\leftarrow 0$, $\lambda_j^0\leftarrow\lambda_0$, j=0, 1, …, l. $\boldsymbol{X}^0$ is initialized by parameters from initial curves and initial symmetry transformation parameters.

Step 2: Take $\boldsymbol{X}^k$ as the initial value to find $\boldsymbol{X}^{k+1}$ which minimizes J (using the Levenberg-Marquardt algorithm).

Step 3: If $|c_j(\boldsymbol{X}^k)|<\tau$, j=0, 1, …, l or $k>N$, stop.

Step 4: $\lambda_j^k\leftarrow f\lambda_j^k$, j=0, 1, …, l.

Step 5: $k\leftarrow k+1$, go to Step 2.

In our implementation, λ_0=1, f=10, $\tau=10^{-10}$, N=40, for the standard Levenberg-Marquardt algorithm (Werghi *et al.*, 1999).

EXAMPLES

All the examples were tested on a PC with a 2 GHz Pentium CPU and 256M RAM. Some examples use synthetic data while others use point data captured from real engineering parts.

2D translational symmetry

This example demonstrates the reconstruction of a 3×4 array (Fig.14). The synthetic data were generated as follows. First a unit square was created, then four corners were rounded by a fillet operation with a radius of 0.25 units. The array was created with two translational vectors, $\boldsymbol{T}_1=[2,\ 0]^{\mathrm{T}}$ and $\boldsymbol{T}_2=[0,\ 2]^{\mathrm{T}}$. Ten points from each line segment and eight points from each arc segment were uniformly sampled. Gaussian noise (μ=0, σ=0.01 unit) was added to the points to create the synthetic data. The initial curves were created by least squares fitting (Pratt, 1987) and constraints (8 tangent constraints, 2 parallel constraints, 1 perpendicular constraint) were specified between the initial curves of the base component. The average distance of the synthetic points to the initial model (geometric constraints and symmetry properties were not satisfied) was 0.0063 units. After constrained fitting, this average distance increased to 0.0101 units. However, geometric constraints were satisfied to a high level (the errors caused by constraint equations were less than 10^{-10}), and the symmetry properties were perfectly preserved owing to the explicit representation of translational symmetry. The constrained optimization of this model cost 15 s. When the number of the sampling points was doubled, the time cost remained nearly the same. This phenomenon resulted from the separated form of distance functions.

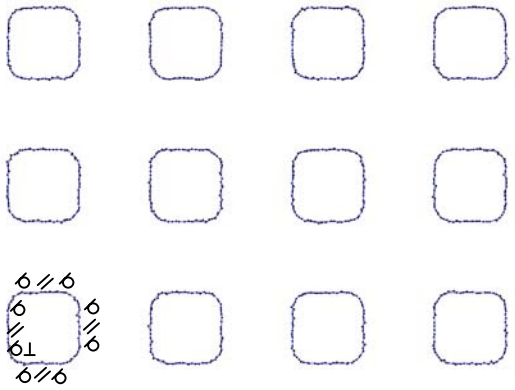
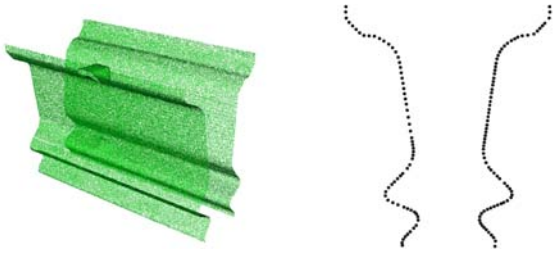
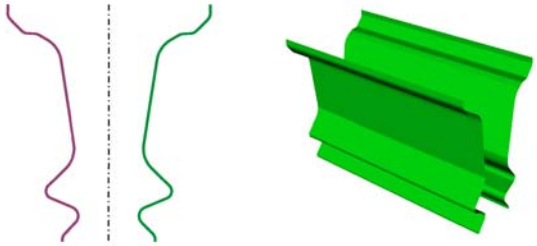


Fig.14 2D array of rounded squares and associated points



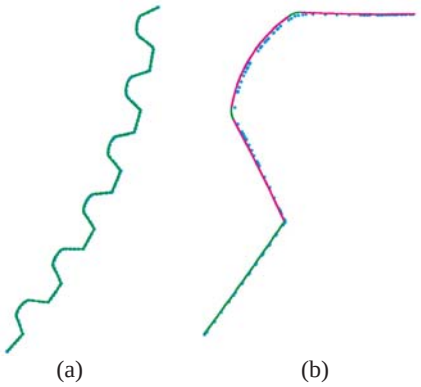
(a) (b)

Fig.15 Point data of the dovetail of a turbine blade. (a) Point cloud; (b) Points from a cross section



(a) (b)

Fig.16 Reconstructed curves and surfaces of the dovetail of a turbine blade. (a) Optimized curves of a cross section; (b) Symmetric surface created from optimized curves



(a) (b)

Fig.17 Point data and nominal curves of the spline part. (a) Six cycles of initial curves and points; (b) Magnification of one cycle

2D reflectional symmetry—turbine blade model

Fig.15a is a point cloud (about 132.08×63.50×76.20 mm^3, 80 000 points) from the dovetail of a turbine blade, acquired by industrial CT. First, we extracted the extruded direction of the surface by a method that used the principal directions of points, which were estimated from the point cloud. Then we sliced the point cloud by a plane that was perpendicular to the extruded direction. Based on the sequenced points (250 points) acquired by slicing (Fig.15b), the initial curves (a total of 16 curve segments on each side, all segments were lines or circular arcs) were created and the constraints (tangency) on these curves were specified. Symmetric curves were reconstructed after constrained optimization (Fig.16a). The average distance of section points to initial curves and symmetric curves was 0.010 and 0.023 mm, respectively. The optimization process cost 6 s. The reconstructed model reached a geometric constraint tolerance of 10^{-10} and preserved a perfect reflectional symmetry structure. The symmetric dovetail surface (minor geometric feature ignored) was created from the reconstructed symmetric curves and the extruded direction (Fig.16b).

2D rotational symmetry—spline part model

This example relates to the reconstruction of section profiles of a gear wheel. Six cycles of points (338 points) were measured points and the initial curves were from a nominal model. Geometric constraints between curve segments were specified (Fig.17). The nominal model satisfied the required geometric constraints and preserved symmetry structure; however, it was not the best fitting model to the measured points. The average distance of measured points to the profile curve before and after optimization was 0.0277 and 0.0034 mm, respectively. The time cost in the optimization stage of this example was 8 s. Fig.18 is the whole spline (56 cycles) created based on the optimized base cycle and optimized rotation center.

3D translational symmetry—internal cooling holes of a turbine blade

The point data were gathered from the internal structure of a turbine blade by an industrial CT

(7.112×23.368×43.180 mm^3, 13 000 points). Five rows of cooling holes were to be reconstructed (Fig.19). Each row formed a 1D array structure (Note that the radii of holes from different rows were different and that this model was not a 2D array structure). Points belonging to the holes were separated out and segmented into sub-point clouds, each of which corresponded to a single cylinder surface. Initial cylinders were fitted by a robust method (Marshall *et al.*, 2001). The average distance of points to cylinders before and after constrained optimization was 0.0483 and 0.0635 mm, respectively. Although the average distance of points to cylinders increased by about 30%, the symmetry structure was preserved after optimization, which was not the case for the initial cylinders. The time cost for the five 1D arrays ranged from 12 to 20 s, the differences being caused by the different qualities of the initial values. For the reconstructed 1D arrays of cylinders, refer to Fig.20. Note that the direction of cylinders in the lower rows cannot be recovered precisely owing to the lack of sufficient points.

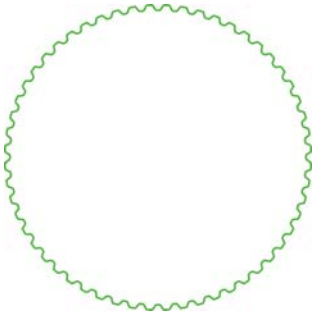


Fig.18 The whole spline recovered

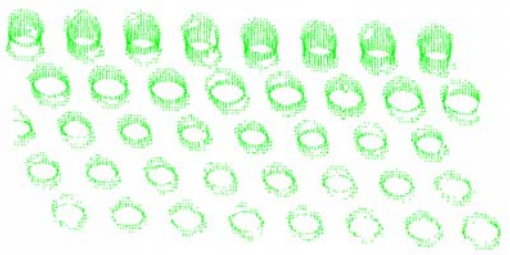


Fig.19 Point data from the pin-fin structure of a turbine blade

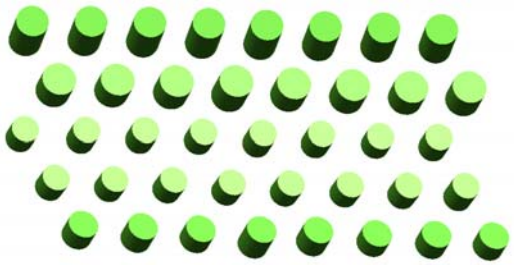


Fig.20 Reconstructed 1D array of cylinders

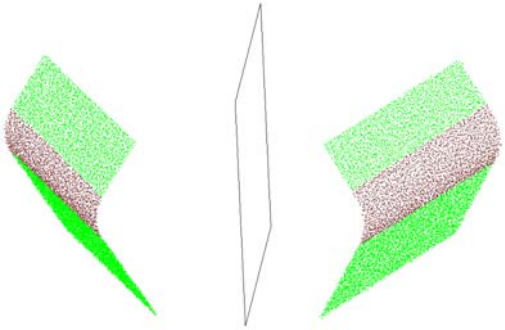


Fig.21 Synthetic point data from a smoothly connected plane and cylinder surfaces

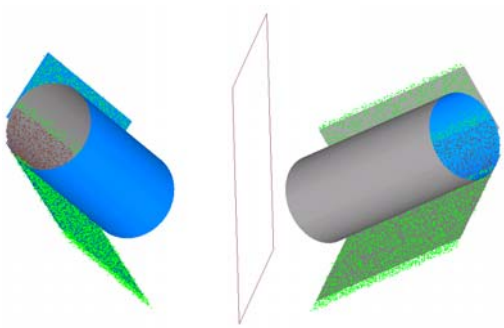


Fig.22 Reconstructed plane and cylinder surfaces (not trimmed)

3D reflectional symmetry

In engineering parts, often the reconstruction of reflectional symmetric surfaces can be converted into the reconstruction of reflectional symmetric curves (e.g., the turbine blade dovetail). However, there may be situations where such a conversion is inapplicable. Here, we use a synthetic model to test our algorithm. In the synthetic model, two planar surfaces were smoothly connected by a cylindrical patch. 19 200 data points were uniformly sampled, and Gaussian noise (μ=0, σ=0.01 unit) was added. The size of the point cloud was 3.0 units×1.7 units×3.1 units (Fig.21). Initial surfaces were fitted (Marshall *et al.*, 2001) and tangency constraints were manually specified between the planes and the cylinder. The average distance of points to the initial surfaces was 0.007 62 units. After constrained fitting, the average distance was 0.009 37 units. The values of constraint equations were less than 10^{-10} and symmetry structure was preserved. The optimization process for this model cost 33 s. Fig.22 shows the reconstructed symmetric model (surfaces not trimmed).

CONCLUSION AND FUTURE WORK

In this paper, we discussed the reconstruction of symmetric geometric models in the framework of constrained fitting. Explicit representation of symmetric models are presented and used in the representation of constrained fitting problems. Synthetic and industrial examples demonstrate that the proposed method is applicable to the reconstruction of engineering parts. Furthermore, the algorithm is robust and the time cost is reasonable. However, further research work is needed to reconstruct real life workpieces with very complicated symmetry structure. For example, symmetry information is currently specified by user interaction, in which the user needs to be very careful, otherwise errors may occur. Thus, automatic or semi-automatic solutions may be desirable in real life. Our current objectives are to develop algorithms to detect symmetry structures from initial curves or surfaces to avoid the need for user interaction and to develop algorithms to reconstruct symmetric freeform surfaces, such as the roof surface of a car, the hull surface of a ship, etc.

Reference

Benkő, P., Kós, G., Várady, T., Andor, L., Martin, R.R., 2002. Constrained fitting in reverse engineering. *Computer Aided Geometric Design*, **19**(3):173-205. [doi:10.1016/S0167-8396(01)00085-1]

Faber, P., Fisher, R.B., 2001. Pros and cons of Euclidean fitting. *LNCS*, **2191**:414-420.

Fiacco, A., McCormick, G., 1968. Nonlinear Programming: Sequential Unconstrained Minimization Techniques. Wiley, New York, p.593-594.

Ke, Y., Zhu, W., Liu, F., Shi, X., 2006. Constrained fitting for 2D profile-based reverse engineering. *Computer-Aided Design*, **38**(2):101-114. [doi:10.1016/j.cad.2005.07.004]

Kós, G., Martin, R.R.,Várady, T., 1999. Methods to recover constant radius rolling ball blends in reverse engineering. *Computer Aided Geometric Design*, **17**(2):127-160. [doi:10.1016/S0167-8396(99)00043-6]

Langbein, F.C., 2003. Beautification of Reverse Engineered Geometric Models. PhD Thesis, Department of Computer Science, Cardiff University, Cardiff, Wales, UK.

Langbein, F.C., Marshall, A.D., Martin, R.R., 2004. Choosing consistent constraints for beautification of reverse engineered geometric models. *Computer-Aided Design*, **36**(3):261-278. [doi:10.1016/S0010-4485(03)00108-8]

Li, M., Langbein, F.C., Martin, R.R., 2007. Detecting Approximate Incomplete Symmetries in Discrete Point Sets. Proc. ACM Symp. on Solid and Physical Modeling. ACM Press, p.335-340. [doi:10.1145/1236246.1236294]

Li, M., Langbein, F.C., Martin, R.R., 2008. Detecting approximate symmetries of discrete point subsets. *Computer-Aided Design*, **40**(1):76-93. [doi:10.1016/j.cad.2007.06.007]

Marshall, D., Lukacs, G., Matin, R.R., 2001. Robust segmentation of primitives from range data in the presence of geometric degeneracy. *IEEE Trans. on Pattern Anal. Machine Intell.*, **23**(3):304-314. [doi:10.1109/34.910883]

Mills, B.I., Langbein, F.C., Marshall, A.D., Martin, R.R., 2001. Approximate Symmetry Detection for Reverse Engineering. Proc. 6th ACM Symp. on Solid Modelling and Applications. ACM Press, p.241-248. [doi:10.1145/376957.376985]

Mitra, N.J., Guibas, L.J., Pauly, M., 2006. Partial and approximate symmetry detection for 3D geometry. *ACM Trans. on Graph.*, **25**(3):560-568. [doi:10.1145/1141911.1141924]

Pratt, V., 1987. Direct least-squares fitting of algebraic surfaces. *Comput. Graph.*, **21**(4):145-152. [doi:10.1145/37402.37420]

Shen, D., Ip, H.H.S., Cheung, K.K.T., Teoh, E.K., 1999. Symmetry detection by generalized complex (GC) moments: a close-form solution. *IEEE Trans. on Pattern Anal. Machine Intell.*, **21**(5):466-476. [doi:10.1109/34.765657]

Sun, C., Sherrah, J., 1997. 3D symmetry detection using the extended Gaussian image. *IEEE Trans. on Pattern Anal. Machine Intell.*, **19**(2):164-168. [doi:10.1109/34.574800]

Thompson, W.B., Owen, J.C., James, H., Stark, S.R., Henderson, T.C., 1999. Feature-based reverse engineering of mechanical parts. *IEEE Trans. on Rob. Autom.*, **15**(1):57-66. [doi:10.1109/70.744602]

Várady, T., Martin, R.R., Coxt, J., 1997. Reverse engineering of geometric models—an introduction. *Computer-Aided Design*, **29**(4):255-268. [doi:10.1016/S0010-4485(96)00054-1]

Werghi, N., Fisher, R.B., Robertson, C., Ashbrook, A., 1999. Object reconstruction by incorporating geometric constraints in reverse engineering. *Computer-Aided Design*, **31**(6):363-399. [doi:10.1016/S0010-4485(99)00038-X]

Weyl, H., 1952. Symmetry. Princeton University Press, Princeton, New Jersey.

APPENDIX: DISTANCE OF POINT TO CURVE OR SURFACE IN SYMMETRIC MODEL

Table A1 Points from base component

	S	***P***
Line	$[l_0, l_1, l_2]$,	$[x, y, 1]$
Circle	$[c_0, c_1, c_2, c_3]$	$[x^2+y^2, x, y, 1]$
Plane	$[q_0, q_1, q_2, q_3]$	$[x, y, z, 1]$
Sphere	$[s_0, s_1, s_2, s_3, s_4]$	$[x^2+y^2+z^2, x, y, z, 1]$
Cylinder	$[c_3(1-c_0^2), c_3(1-c_1^2), c_3(1-c_2^2), -c_0c_1c_3, -c_0c_2c_3, -c_1c_2c_3, c_4, c_5, c_6, c_7]$	$[x^2, y^2, z^2, 2xy, 2xz, 2yz, x, y, z, 1]$

Table A2 Points from symmetric component

	S	***P***
1D array in 2D		
Line	$[l_0, l_1, -i(l_0u_x+l_1u_y)+l_2]$	$[x, y, 1]$
Circle	$[c_0, -2ic_0u_x+c_1, -2ic_0u_y+c_2, i^2c_0(u_x^2+u_y^2)-i(c_1u_x+c_2u_y)+c_3]$	$[x^2+y^2, x, y, 1]$
2D array in 2D		
Line	$[l_0, l_1, -l_0(iu_x+jv_x)-l_1(iu_y+jv_y)+l_2]$	$[x, y, 1]$
Circle	$[c_0, -2c_0(iu_x+jv_x)+c_1, -2c_0(iu_y+jv_y)+c_2, i^2c_0(u_x^2+u_y^2)+j^2c_0(v_x^2+v_y^2)+2ijc_0(u_xv_x+u_yv_y)$ $-c_1(iu_x+jv_x)-c_2(iu_y+jv_y)+c_3]$	$[x^2+y^2, x, y, 1]$
Reflectional symmetry in 2D		
Line	$[l_0-2n_x(l_0n_x+l_1n_y), l_1-2n_y(l_0n_x+l_1n_y), l_2+2d(l_0n_x+l_1n_y)]$	$[x, y, 1]$
Circle	$[c_0+4c_0n_x^2(n_x^2+n_y^2-1), 8c_0n_xn_y(n_x^2+n_y^2-1), 4c_0n_y^2(n_x^2+n_y^2-1)+c_0, c_1+4c_0n_xd(1-2n_x^2-2n_y^2)$ $-2n_x(c_1n_x+c_2n_y), c_2+4c_0n_yd(1-2n_x^2-2n_y^2)-2n_y(c_1n_x+c_2n_y), c_3+4c_0d^2(n_x^2+n_y^2)+2d(c_1n_x+c_2n_y)]$	$[x^2, xy, y^2, x, y, 1]$
Rotational symmetry in 2D		
Line	$[l_0\cos\theta+l_1\sin\theta, l_1\cos\theta-l_0\sin\theta, l_0(\eta\sin\theta+\xi-\xi\cos\theta)-l_1(\xi\sin\theta+\eta\cos\theta-\eta)+l_2]$	$[x, y, 1]$
Circle	$[c_0, 2c_0(\xi\cos\theta+\eta\sin\theta-\xi)+c_1\cos\theta+c_2\sin\theta, 2c_0(\eta\cos\theta-\xi\sin\theta-\eta)+c_2\cos\theta-c_1\sin\theta,$ $2c_0(\eta^2+\xi^2-\eta^2\cos\theta-\xi^2\cos\theta)+c_1(\eta\sin\theta+\xi-\xi\cos\theta)-c_2(\xi\sin\theta-\eta+\eta\cos\theta)+c_3]$	$[x^2+y^2, x, y, 1]$
1D array in 3D		
Plane	$[q_0, q_1, q_2, -i(q_0u_x+q_1u_y+q_2u_z)+q_3]$	$[x, y, z, 1]$
Sphere	$[s_0, -2is_0u_x+s_1, -2is_0u_y+s_2, -2is_0u_z+s_3, i^2s_0(u_x^2+u_y^2+u_z^2)-i(s_1u_x+s_2u_y+s_3u_z)+s_4]$	$[x^2+y^2+z^2, x, y, z, 1]$
Cylinder	$[c_3(1-c_0^2), c_3(1-c_1^2), c_3(1-c_2^2), -c_3c_0c_1, -c_3c_1c_2, -c_3c_0c_2, c_4+2ic_3(c_0(c_0u_x+c_1u_y+c_2u_z)-u_x),$ $c_5+2ic_3(c_0(c_0u_x+c_1u_y+c_2u_z)-u_y), c_6+2ic_3(c_0(c_0u_x+c_1u_y+c_2u_z)-u_z),$ $i^2c_3(u_x^2+u_y^2+u_z^2-(c_0u_x+c_1u_y+c_2u_z)^2)-i(c_4u_x+c_5u_y+c_6u_z)+c_7]$	$[x^2, y^2, z^2, 2xy, 2xz, 2yz, x, y, z, 1]$
2D array in 3D		
Plane	$[q_0, q_1, q_2, -q_0(iu_x+jv_x)-q_1(iu_y+jv_y)-q_2(iu_z+jv_z)+q_3]$	$[x, y, z, 1]$
Sphere	$[s_0, s_1-2s_0(iu_x+jv_x), s_2-2s_0(iu_y+jv_y), s_3-2s_0(iu_z+jv_z),$ $s_0((iu_x+jv_x)^2+(iu_y+jv_y)^2+(iu_z+jv_z)^2)-s_1(iu_x+jv_x)-s_2(iu_y+jv_y)-s_3(iu_z+jv_z)+s_4]$	$[x^2+y^2+z^2, x, y, z, 1]$
Cylinder	$[c_3(1-c_0^2), c_3(1-c_1^2), c_3(1-c_2^2), -c_3c_0c_1, -c_3c_1c_2, -c_3c_0c_2, c_4+2c_3(c_0(c_0(iu_x+jv_x)+c_1(iu_y+jv_y)$ $+c_2(iu_z+jv_z))-iu_x-jv_x), c_5+2c_3(c_0(c_0(iu_x+jv_x)+c_1(iu_y+jv_y)+c_2(iu_z+jv_z))-iu_y-jv_y),$ $c_6+2c_3(c_0(c_0(iu_x+jv_x)+c_1(iu_y+jv_y)+c_2(iu_z+jv_z))-iu_z-jv_z), c_3((iu_x+jv_x)^2+(iu_y+jv_y)^2+(iu_z+jv_z)^2$ $-(c_0(iu_x+jv_x)+c_1(iu_y+jv_y)+c_2(iu_z+jv_z))^2-c_4(iu_x+jv_x)-c_5(iu_y+jv_y)-c_6(iu_z+jv_z)+c_7)]$	$[x^2, y^2, z^2, 2xy, 2xz, 2yz, x, y, z, 1]$
Reflectional symmetry in 3D		
Plane	$[q_0(1-2n_x^2)-2q_1n_xn_y-2q_2n_xn_z, q_1(1-2n_y^2)-2q_0n_xn_y-2q_2n_yn_z, q_2(1-2n_z^2)-2q_0n_xn_z-2q_1n_yn_z,$ $2d(q_0n_x+q_1n_y+q_2n_z)+q_3]$	$[x, y, z, 1]$
Sphere	$[s_0(4n_x^2(n_y^2+n_z^2)+(1-2n_x^2)^2), s_0(4n_y^2(n_x^2+n_z^2)+(1-2n_y^2)^2), s_0(4n_z^2(n_x^2+n_y^2)+(1-2n_z^2)^2),$ $s_0n_xn_y(n_x^2+n_y^2+n_z^2-1), s_0n_yn_z(n_x^2+n_y^2+n_z^2-1), s_0n_zn_x(n_x^2+n_y^2+n_z^2-1), 4s_0dn_x(1-2(n_x^2+n_y^2+n_z^2))$ $+s_1(1-2n_x^2)-2s_2n_xn_y-2s_3n_xn_z, 4s_0dn_y(1-2(n_x^2+n_y^2+n_z^2))+s_2(1-2n_y^2)-2s_1n_xn_y-2s_3n_yn_z,$ $4s_0dn_z(1-2(n_x^2+n_y^2+n_z^2))+s_3(1-2n_z^2)-2s_1n_xn_z-2s_2n_yn_z, 4s_0d^2(n_x^2+n_y^2+n_z^2)+2d(s_1n_x+s_2n_y+s_3n_z)+s_4]$	$[x^2, y^2, z^2, 8xy, 8xz, 8yz, x, y, z, 1]$
Cylinder	$[c_3(4n_x^2(n_y^2+n_z^2)+(1-2n_x^2)^2-(c_0(1-2n_x^2)-2c_1n_xn_y-2c_2n_xn_z)^2),$ $c_3(4n_y^2(n_x^2+n_z^2)+(1-2n_y^2)^2-(c_1(1-2n_y^2)-2c_0n_xn_y-2c_2n_yn_z)^2), c_3(4n_z^2(n_x^2+n_y^2)+(1-2n_z^2)^2$ $-(c_2(1-2n_z^2)-2c_0n_xn_z-2c_1n_yn_z)^2), c_3(4n_xn_y(n_x^2+n_y^2+n_z^2-1)-(c_1-2n_y(c_0n_x+c_1n_y+c_2n_z))$ $\cdot(c_0-2n_x(c_0n_x+c_1n_y+c_2n_z))), c_3(4n_yn_z(n_x^2+n_y^2+n_z^2-1)-(c_2-2n_z(c_0n_x+c_1n_y+c_2n_z))(c_1-2n_y(c_0n_x+c_1n_y$ $+c_2n_z))), c_3(4n_zn_x(n_x^2+n_y^2+n_z^2-1)-(c_0-2n_x(c_0n_x+c_1n_y+c_2n_z))(c_2-2n_z(c_0n_x+c_1n_y+c_2n_z))),$ $c_4-2n_x(c_4n_x+c_5n_y+c_6n_z)+c_3(4dn_x(1-2n_x^2-2n_y^2-2n_z^2)-4d(c_0n_x+c_1n_y+c_2n_z)(c_0-2n_x(c_0n_x+c_1n_y+c_2n_z))),$ $c_5-2n_y(c_4n_x+c_5n_y+c_6n_z)+c_3(4dn_y(1-2n_x^2-2n_y^2-2n_z^2)-4d(c_0n_x+c_1n_y+c_2n_z)(c_1-2n_y(c_0n_x+c_1n_y+c_2n_z))),$ $c_6-2n_z(c_4n_x+c_5n_y+c_6n_z)+c_3(4dn_z(1-2n_x^2-2n_y^2-2n_z^2)-4d(c_0n_x+c_1n_y+c_2n_z)(c_2-2n_z(c_0n_x+c_1n_y+c_2n_z))),$ $4c_3d^2(n_x^2+n_z^2+n_y^2-(c_0n_x+c_1n_y+c_2n_z)^2)+2d(c_4n_x+c_5n_y+c_6n_z)+c_7]$	$[x^2, y^2, z^2, 2xy, 2xz, 2yz, x, y, z, 1]$