List of Authors

Lilit Yenokyan
lilit@umich.edu
Department of Industrial and Manufacturing Systems Engineering
University of Michigan – Dearborn, Dearborn, MI 48128

Lilit Yenokyan is the M.S. degree candidate in Computer and Information Systems Department at the University of Michigan-Dearborn (UM-D). She received the B.S. degree in applied mathematics from Yerevan State University and currently holds a Research Assistant position in Industrial and Manufacturing Systems Engineering Department at UM-D. Her research interests include CAD/CAM systems, symmetric reconstruction and 3D part reconstruction, machine learning and intelligent systems.

Armen Zakarian
zakarian@umich.edu
Department of Industrial and Manufacturing Systems Engineering
University of Michigan – Dearborn, Dearborn, MI 48128

Armen Zakarian received the B.S. degree in mechanical engineering from Yerevan Polytechnic University, Yerevan, Armenia, the M.S. degree in industrial and systems engineering from University of Southern California, Los Angeles, California and the Ph.D. degree in industrial engineering from The University of Iowa, Iowa City, Iowa, in 1997.
He is an Associate Professor of Industrial and Manufacturing Systems Engineering at the University of Michigan-Dearborn. He published papers in journals sponsored by ASME, IEEE and IIE societies. His research interests include development of integrated products and systems, system engineering and architecting, and intelligent design and manufacturing His research work is supported through grants from AT&T, FORD Motor Company, General Motors, Visteon Corporation, TRW, US Army, and US Navy. He is a member of IEEE and IIE.

Pravansu Mohanty
pmohanty@umich.edu
Department of Mechanical Engineering
University of Michigan – Dearborn, Dearborn, MI 48128

Pravansu Mohanty is an Associate Professor of Mechanical Engineering at the University of Michigan-Dearborn. His research interests include laser materials processing, spray forming, solidification processing, metal matrix composites materials. His research work is supported through grants from NSF, US Army, US Navy.

Vahram Avagyan
avagyanv@ncbi.nlm.nih.gov
Lockheed Martin Corporation, Bethesda, MD 20817

Vahram Avagyan received the M.S. degree in Computer and Information Systems from the University of Michigan-Dearborn. His research interests include CAD/CAM systems, symmetric reconstruction and 3D part matching. He is currently with the Lockheed Martin Coorporation.

Symmetric Reconstruction Algorithms for Incomplete 3D Models

Abstract

Reconstruction of incomplete 3D models of real-life objects and use of these objects in a variety of applications, such as medical, engineering, computer vision, and manufacturing, are of increased interest. In this paper, a new approach and algorithms for 3D model reconstruction, based on the model symmetry, are presented. Every phase of the proposed approach slices the 3D model into layers along one of the axes, then the layers are analyzed individually, and the results of the analysis are combined to produce an overall assessment of the model's positioning and geometric properties. After the incomplete 3D model is introduced to the system, symmetry type and model parameters are established, and an algorithm is developed for model alignment. Symmetry analysis algorithms that use steps from the slicing procedure are proposed for collecting information on the asymmetric/missing sections of the aligned model on a layer-by-layer basis. The outputs of these algorithms are arrays of points that complete symmetric properties of the layers, which are input to the mesh generation algorithm that generates a triangular mesh based on those points. The methodology developed in this paper is illustrated with the 3D model reconstruction examples. A computation study on the algorithms performance is also presented.

Keywords: Symmetric reconstruction, CAD/CAM, symmetry analysis, 3D geometry, model alignment, 3D mesh generation, 3D model slicing, planar symmetry, axis symmetry

1. Introduction

Reconstruction of incomplete 3D models of real-life objects and use of these objects in a variety of applications, such as medical, engineering, computer vision, and manufacturing are of increased interest. Many of the traditional 3D model reconstruction techniques are based on finding a match for an incomplete model from the database (Shilane and Funkhouser 2006, Osada et al. 2001, Avagyan et al. 2007). However, often the original instance of an incomplete model is unavailable in the database and the missing section of the object needs to be reconstructed. For such applications, content-based 3D model reconstruction techniques are necessary. Many 3D model reconstruction techniques are based on the detection of the symmetric features of an object presented in the polygonal mesh (see Kazdhan et al. 2004(a), Podolak et al. 2006, Martinet et al. 2006, Del Bimbo and Pala 2006, Gal and Cohen-or 2003, Elad et al. 2001). There are several broadly studied categories of 3D model feature-based reconstruction, including skewed models (Vincent et al. 2003(a), Vincent el al. 2003(b)), polygonal soup or defective objects (Ohbuchi et al. 2003, Barequet and Kumar 1997, Bischoff and Kobbelt 2005), Gaussian image (Sun and Sherrah 1997), and spherical harmonics (Burel and Henocq 1995, Erturk and Dennis 1997). Del Bimbo and Pala (2006) provide a comprehensive overview of these approaches and highlight their advantages and disadvantages.

Most content-based reconstruction techniques assume that incomplete models do not have significant loss of the original volume. Incomplete 3D models considered in this paper are somewhat different from those studied in the literature. In this paper, the model is assumed to have up to 25% loss of original volume, which affects the characteristics (i.e., mass center, geometric moments, etc.) used as a basis in most recent object reconstruction research approaches. The reconstruction approach proposed in this paper requires partial user intervention to specify the expected symmetry type for the object. The reconstruction accuracy of the approach depends on the profile of the missing volume. Meaning, if the missing section of the object does not significantly impact its symmetric features, then the approach accurately reconstructs missing sections of an object even when more than 25% of the original volume is missing. However, if the missing section significantly impacts initial symmetry properties of the object, then the reconstruction accuracy may be compromised.

The main advantages of the proposed approach are its efficiency, high level of flexibility, user control, and extendibility. These are some of the most important features required by industrial applications. Complete mathematical accuracy, on the other hand, is not pursued by this method. It reconstructs the symmetry accurately enough for the expected type of input, i.e., for mechanical parts with partial deformations and defects, while much more complex 3D models, e.g., whole engineering devices or machines, fall out of the applicability scope of the method. The drawbacks of the proposed algorithms are the requirement of partial user intervention and that the accuracy of reconstruction may depend on user selection.

The rest of the paper is organized as follows. In Section 2, the related work is briefly reviewed. Section 3 provides a general outline of the approach and the expected results. In Section 4 we present the algorithm for slicing 3D objects, considered a critical step in the proposed solution. In Section 5, the heuristic algorithm for model alignment is described. Section 6 presents symmetry analysis, reconstruction, and 3D mesh generation algorithms. Section 7 presents reconstruction examples and the computation study.

2. Literature Review

One of the commonly used methods for feature-based reconstruction of 3D objects is to detect the symmetric features of an object and reconstruct the object using those features. Initial approaches reduce symmetry detection of planar 2D models (Attalah 1985, Wolter et al. 1985) into a 1D string matching problem for which an efficient solution has been developed (Knuth et al. 1977). These algorithms are able to detect only the symmetries of perfect models and are highly sensitive to the minimum noise.

Zabrodsky et al. (1995) used measure of symmetry for estimating the minimum amount of work required to transform a given shape into a symmetric one. The work amount is measured by the minimum mean squared distance required to move points of the original shape to obtain a symmetric shape. This approach helps to determine symmetries in noisy 2D objects, but since the method relies on the ability to establish point correspondence, it is mostly impractical for 3D reconstruction applications.

Kazhdan et al. (2003) introduced a reflective symmetry descriptor measure for obtaining a 3D model's center of mass. Kazhdan et al. (2004(b)) used prior work on reflection symmetries (Kazhdan et al. 2004(a), Kazhdan et al. 2003) and generalized the meaning of symmetry descriptors, defining them as symmetries of an object with respect to all the planes and rotations through its center of mass. Podolak et al. (2006) used planar reflective symmetry transform by capturing the measure of reflectional symmetry of a shape with respect to all possible planes. Though these methods are sufficient for reconstructing the voxel (3D pixel) models with small noise, they are sensitive to higher noise amounts and provide no guarantee for the accurate reconstruction of the polygonal mesh models. Martinet et al. (2006) proposed automatic method for finding symmetries of 3D shapes. Symmetries are found using transitional condition and the generalized moments. By examining both extema and spherical harmonic coefficients of these moments, they recover symmetry parameters of shapes. For large and fused shapes, the approach first obtains symmetries of parts separately and then recovers symmetries for the whole shape using the incremental algorithm. This method does not depend on the tessellation of a model and works well with noisy objects.

Gal and Cohen-or (2006) introduced new local surface descriptors for the surfaces represented by the triangular meshes, which efficiently represent the geometry of local regions of the surface independently on the underlying mesh. Del Bimbo and Pala (2006) performed comparative analysis of 3D model retrieval approaches, classifying them into (a) statistics-based solution approaches (Osada et al. 2001, Osada et al. 2002, Ohbuchi et al. 2003), (b) geometry-based approaches (Elad and Tal 2001, Cybenco et al. 1997), and (c) view-based solution approaches (Assfalg et al. 2003, Chen et al. 2003).

3. Symmetric Reconstruction of 3D Models

In this section, an analytical approach for symmetry reconstruction is presented. The approach includes four main phases, which are outlined in Figure 1. The proposed method is semi-automatic, meaning it requires user intervention (participation).

The initial user-guided step establishes the symmetry type (axis or planar) and the parameters for the symmetry analysis. The system parameters and user options are

discussed in the next several sections of the paper, depending on the step of the method where they first come into play. After the initial settings are established, the remaining steps, such as searching for a particular axis of symmetry, alignment of the input 3D model, analysis of the symmetric properties of the model, determination of possible defects, and precise reconstruction of missing portions in the form of 3D meshes, are automatic and require no user intervention.

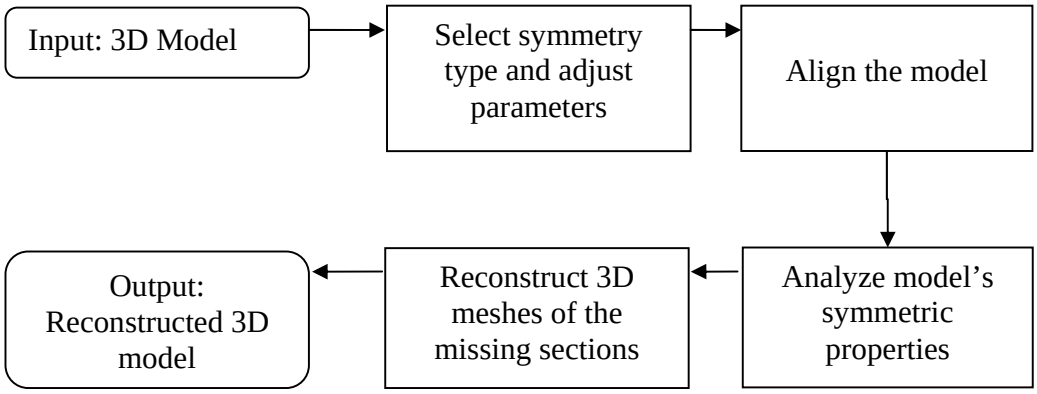


Figure 1: Main phases of the symmetric reconstruction approach

Every phase of the proposed solution slices the 3D model into layers along one of the axes, then the layers are analyzed individually, and the results of the analysis are combined to produce an overall assessment of the model's positioning and its geometric properties. Because this slicing procedure (Algorithms 1-3), in its many variations, is used iteratively throughout the symmetry reconstruction, it is considered a key part of the proposed solution and must be designed to be efficient and accurate to ensure the overall efficiency of the approach.

Once the 3D model is introduced and symmetry type and model parameters are established, an algorithm is developed for model alignment. The alignment algorithm rotates the model along the *z*-axis by the predefined angle and slices it into layers. The number of symmetric layers is determined for each rotation, and the rotation that produces the maximum number of symmetric layers is used to align the model. Once the model is aligned in the third phase, analysis of model symmetric features is performed. Both symmetry analysis algorithms use steps from the slicing procedure. These steps are used to analyze the symmetry and collect information on the asymmetric/missing sections

of the model on a layer-by-layer basis. The information on missing sections is stored in polyline pairs, which are used to generate 3D meshes. The outputs of both symmetry analysis algorithms are arrays of points that complete the symmetric properties of the layers, which are input to the mesh generation algorithm that generates a triangular mesh based on those points.

Before proceeding to the main phases of the reconstruction method, a template algorithm for slicing 3D models is presented on the next section.

4. Slicing 3D Models

In this section, two algorithms for slicing 3D models into 2D layers are presented. The slicing procedure accepts the 3D model as an input and divides the model into 2D polylines against a particular axis (Algorithms 1 and 2) and allows further work with the 2D models. Both algorithms derive an outline of a layer in the form of an array of $N_{sectors}$ points and can be used alternatively. The first algorithm is faster but less precise, while the other is much more accurate but requires more execution time. Neither of these algorithms physically generate the 2D layer; both of them approximate it “on the fly”, operating directly on the 3D model.

The slicing procedure presented in this section is a template that provides the basis for several algorithms used in this approach. Its main idea is to derive meaningful, structured, and easy-to-process information about the particular positioning of the 3D model in question. Processing of this information varies depending on the algorithm, but the overall structure/skeleton of the slicing procedure remains the same.

Next, we introduce several parameters of the system that deal with the slicing of 3D models. These parameters are defined on the very first steps of the reconstruction procedure, when the input 3D model is provided. All the user-defined (independent) parameters are given default values for the general case. Yet those can be modified by the user, while the others depend on the independent parameters and the model itself and therefore are calculated. Notations used in layer approximation algorithms are presented next.

Notation:

N_{layers} – the number of layers the model will be divided into

$N_{sectors}$ - the number of sectors each layer will be divided into

H_{axis} - the maximum "height" of the model at any possible rotation

Z_{init} – the location of the first possible layer of the model

Δ_z –the distance between layers, $\Delta_z = H_{axis}/N_{layers}$

Algorithm 1: Fast Approximation of a Layer

Input: 3D model *M* and *L* (height of the layer)
Output: *A* (array of $N_{sectors}$ points), *D* (array of $N_{sectors}$ real distance values)

Step 1: Initialize: Array *A* with *null* values, *D* with 0's.
Step 2: Calculate

$Z_{lower} = L - \Delta_z / 2$, $Z_{upper} = L + \Delta_z / 2$, $\Delta_{angle} = 2\pi / N_{sectors}$

Step 3: For each endpoint $P=(x, y, z)$ of a triangle in the model's mesh

If $z \notin \{Z_{lower}; Z_{upper}\}$ then consider next point

Else Goto 4 *(given point belongs to the current layer)*

Step 4: Set $r = \sqrt{(x^2 + y^2)}$, $\alpha = arccos(x/r)$

If $y > 0$ Then $k = [\alpha / \Delta_{angle}]$

Else $k = N_{sectors} - 1 - [\alpha / \Delta_{angle}]$

endif

Step 6: If $(r > D_k)$ then $D_k = r$, $A_k = (x, y, z)$

(Assign D_k the greatest distance from Z-axis)

Endfor

Independent system parameters (N_{layers}, $N_{sectors}$) are used to divide the model into N_{layers}, and each layer into $N_{sectors}$ number of sections. The algorithm traverses all the points of the mesh that correspond to the triangle endpoints (which must be arranged in a highly accessible way for best performance) and filters the points that do not belong to the given layer (Step 3) (see Figure 2a). With the remaining points, it calculates the approximate sector they belong to and their distances from the *z*-axis (Steps 4-5) (see Figure 2b) and leaves only the outermost points in the final results, each sector containing one point at most (Step 6) (see Figure 2c). Note that the accuracy of the approximation generated by

this algorithm greatly depends on the detail level of the model's mesh (number of triangle endpoints), except for the independent system parameters (N_{layers}, $N_{sectors}$). For this reason, the Algorithm 1 is used only as heuristic step, where high accuracy of the results is not required. Steps 4 and 5 of the algorithm are used for calculation of the sector index *k* for the given point *(x, y, z)*. These steps are common in all of the future algorithms and will be omitted in future by referring to their calculation in Algorithm 1 instead. Note that the actual number of sectors ($N_{sectors}$) used in the application is much larger, depending on the user choice of context. This kind of layer approximation, where points are evenly arranged at predefined angles, helps in detecting symmetric properties and possible deviations from the symmetry.

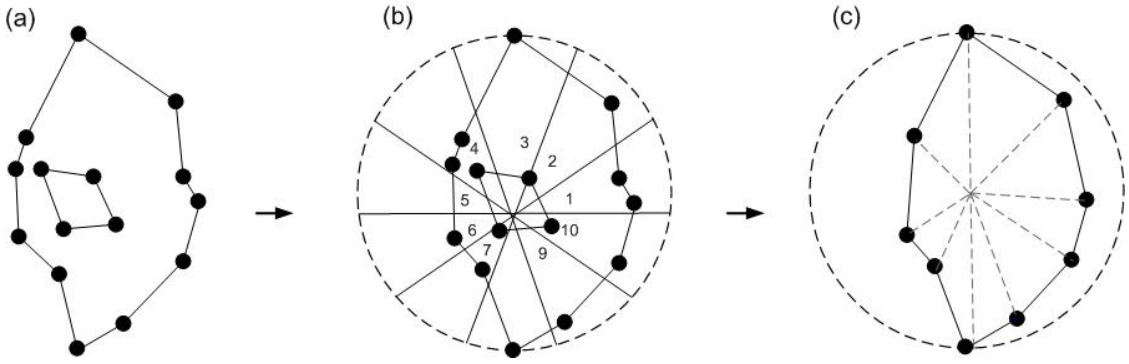


Figure 2: Steps of the slicing procedure: (a) layer of a 3D model with the corresponding triangle endpoints that belong to the layer, (b) layer in (a) divided into $N_{sectors}$=10 sectors, and (c) layer with the outermost points in the final results

The alternative slicing procedure (Algorithm 2) for layer approximation is presented next. The algorithm uses the same input, however it is more precise as it generates and uses far more points for layer approximation. The Accurate Approximation algorithm is used during the symmetry analysis phase (Algorithms 4-5), where the accuracy is most critical.

Algorithm 2: Accurate Approximation of a Layer
Input: 3D model *M and L* (height of the layer)
Output: *A* (array of $N_{sectors}$ points), *D* (array of $N_{sectors}$ real distance values)

Step 1: Initialize: Array *A* with *null* values, *D* with 0's.

Step 2: For each triangle T in the model mesh

If (T intersects with the plane $z=L$) Then Go to Step 3

Step 3: Determine the intersection of T and $z=L$. The result is a line segment (P_1, P_2)

Step 4: Calculate sector indices k_1 and k_2 for the points P_1 and P_2 (see Algorithm 1, Steps 4, 5)

Step 5: Calculate $Count = |k_1 - k_2| + 1$ (Count of angles to consider)

Step 6: For every integer $i = 0, \ldots, Count$

Calculate $P=(x, y, z)$ as follows:

$P = P_1 + (i /Count) (P_2 - P_1)$, $\quad r = \sqrt{(x^2 + y^2)}$

Step 7: If ($r > D_k$) then Set $D_k = r$, $A_k = (x, y, z)$

(Assign D_k the greatest distance from Z-axis)

EndFor

EndFor(Step 2)

The accuracy of Algorithm 2 mainly depends on the independent parameter $N_{sectors}$, as the number of points used in layer approximation depends on this parameter. Points that it generates do not necessarily coincide with the triangle endpoints on the mesh. Instead, an intersection of all triangles with a given layer is detected, and the intersecting segment is calculated. After the intersecting segment is found, the algorithm determines the number of sectors between the segment endpoints (indicated by *Count*) and generates those many new points (Steps 6). It determines the sector indexes for each generated point and leaves the outermost point in each sector in the final results (Steps 7). Therefore, this algorithm generates far more points compared to the Algorithm 1. The benefits are evident, especially for less detailed meshes, as it still produces layer outlines with the predefined level of accuracy. On the other hand, due to the complexity of the point generation procedure and the larger number of points considered, this algorithm requires longer execution time. The computational complexity of both algorithms has linear dependency to the number of triangles in the mesh, and therefore, both algorithms guarantee high efficiency for even highly detailed meshes.

5. Aligning the model

In this section, we discuss the user-controlled step for the alignment of the input model along one of the model axes. Before any symmetry analysis takes place, the user must specify the type of symmetry that the input model is expected to possess and optionally adjust the numeric parameters. In the current approach, two user choices (i.e., axial or planar) for model symmetry are considered. Furthermore, for planar symmetry, two subclasses, mirror and glide reflection symmetry, are considered.

For the *axial* symmetry, the model is aligned so that its axis of symmetry nearly coincides with the model *z*-axis, and all future symmetry analysis and reconstruction of the missing sections are based on the model slicing along the *z*-axis. This is largely done automatically, although the alignment procedure is only a heuristic one. In the case of the *planar* symmetry, in addition to the model alignment along *z*-axis, the model alignment along its orthogonal axis is desirable for the best results. Both these axes serve as defining lines for the plane of symmetry. The remaining alignment steps are similar to the axial symmetry example. It needs to be emphasized that the determination of the symmetry depends on the use of the centroid for each layer of the model. Additional notations used in an alignment procedure are presented next.

Notation:

$r_j\,(x, y, z)$- set of rotations (angle triplets) to be tested for model alignment, where $j = 1,\ldots, R$ (user-defined)

δ_D – the threshold for difference between the distance of outermost point from *z*-axis and the average distance of the points from *z*-axis on the layer (user-defined)

N_{sym} - the number of symmetric layers at each rotation

Algorithm 3: Heuristic Alignment of a Model in Respect to Z-Axis

Input: 3D model *M*

Output: Aligned model M

Step 1: Initialize: $N_{max} = 0$, $r_{best}(x, y, z) = null$, $N_{sym} = 0$

Step 2: For every rotation $r_j(x, y, z)$, $j = 1,\ldots, R$

Step 3: For every layer index i, $i = 1,\ldots, N_{layers}$

Step 4: Calculate arrays A and D using Alg. 1, with the input $L = Z_{init} + i\Delta z$

Calculate D_{avg} as the average of nonzero distances stored in D

Calculate relative difference of distances stored in D from the average

$RelativeDifference = \mathrm{Max}_{k=1..Nsectors} (|D_k - D_{avg}| / D_{avg})$

Step 5: If ($RelativeDifference < \delta_D$) then Set $N_{sym} = N_{sym} + 1$

EndFor (Step 3)

Step 6: If ($N_{\mathrm{sym}} > N_{\mathrm{max}}$) then Set $N_{\mathrm{max}} = N_{\mathrm{sym}}$, $r_{best}(x, y, z) = r_j(x, y, z)$

Endfor(Step 2)

Step 7: Rotate model M according to the rotation $r_{best}(x, y, z)$.

The model in any rotation along the z-axis is an input to the heuristic alignment algorithm (see Figure 3a). Once the input model is obtained, it is rotated by $r_j(x, y, z)$ and sliced using Algorithm 1 for each rotation (Steps 2-4) (Figure 3b). The number of symmetric layers is calculated for each rotation. The layer is considered symmetric when the difference between the distance of the outermost point from the z-axis and the average distance of the points from the z-axis on the layer is less than the user-defined threshold δ_D (Steps 5). The number of symmetric layers is determined for each rotation $r_j(x, y, z)$ (Step 6) and the rotation $r_{best}(x, y, z)$ that produces the most symmetric layers is used to align the model (Step 7) (Figure 3c).

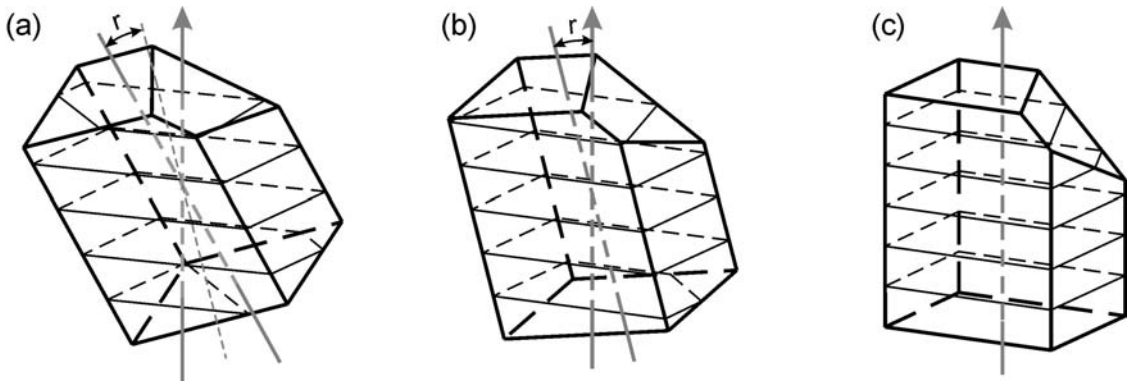


Figure 3: Alignment example of a model with six layers: (a) initial rotation, (b) intermediate rotation, and (c) final rotation of the model

6. Symmetry Analysis, Reconstruction, and 3D Mesh Generation Algorithms

The above algorithms allow one to input a 3D model and heuristically align it along the *z*-axis so that it is close to the actual axis or symmetry (or passes through the plane of symmetry). Next, we present methods that detect symmetric features of the aligned model and reconstruct the missing sections of the model to make it more symmetric in respect to the alignment.

Both symmetry analysis algorithms presented in this section use steps from the slicing procedure. These steps are used to analyze the symmetry and collect information on the asymmetric/missing sections of the model on a layer-by-layer basis. The missing sections are stored in polyline pairs, which are used to generate 3D meshes. Depending on the selected type of symmetry, Algorithm 4 (for axis symmetry) or 5 (for planar symmetry) is considered.

The numeric parameters used in the symmetry analysis procedure through this section are presented next and are listed in the order of first appearance.

Notation:

γ_{axis} - the threshold for the distance differences of points on the two sides of the axis to be still considered a symmetric pair.

γ_{plane} - the threshold for the distance differences of points on the two sides of the plane to be still considered a symmetric pair.

Algorithm 4: Axis Symmetry Analysis

Input: 3D model M

Output: N_{layers} of 2D arrays of polylines $Polylines^{(l)}_{(i,\,j)}$ with dimensions $2 \times N_{layers}$

Step 1: For every layer index l; $l = 1, \ldots, N_{layers}$

Calculate arrays $A^{(l)}$ and $D^{(l)}$ using Alg. 2, with the input $L^{(l)} = Z_{init} + l\Delta_z$

Create an array of points *PointSaved*, with elements initialized as follows:

If $(A^{(l)}_k \neq null)$ Then $PointSaved_k = A^{(l)}_k$

Else $PointSaved_k = (0, 0, L^{(l)})$

Step 2: Initialize: $Polylines^{(l)}_{(i,j)}$ with *null* values

Step 3: Initialize: $D^{(l)}_{max}$ with 0

Step 4: For each k; $k = 1,\ldots, N_{sectors}$

If ($D^{(l)}_i > D^{(l)}_{max}$) Then

Set $D^{(l)}_{max} = D^{(l)}_i$

Step 5: For each j; $j = 1,\ldots, N_{sectors}$

Set $Polylines^{(l)}_{(0,j)} = PointSaved_i$

Set $\theta = j \times 2\pi / N_{sectors}$

If ($D^{(l)}_i < \gamma_{axis} \times D^{(l)}_{max}$) Then

$Polylines^{(l)}_{(1,j)} = (D^{(l)}_{max} \times \cos\theta, D^{(l)}_{max} \times \sin\theta, L^{(l)})$

Else $Polylines^{(l)}_{(1,j)} = (D^{(l)}_j \times \cos\theta, D^{(l)}_j \times \sin\theta, L^{(l)})$

Step 6: Return arrays $Polylines^{(l)}_{(i,j)}$

During the execution of Algorithm 4, the model is sliced, and for each layer, the point with the largest distance from *z*-axis is recorded in $D^{(l)}_{max}$ (Step 4). The initial distribution of points on a single layer is shown on Figure 4a. For each point j on the layer, its coordinates and distance ($D^{(l)}_j$) coordinates at angle θ are recoded in the array $Polylines^{(l)}_{(i,j)}$. When the point j distance ($D^{(l)}_j$) from z-axis is not within γ_{axis} of the maximum distance ($D^{(l)}_{max}$), then it is replaced by the $D^{(l)}_{max}$ in $Polylines^{(l)}_{(i,j)}$ (Step 5), (Figure 4b). The output of the algorithm is the arrays $Polylines^{(l)}_{(i,j)}$, one for each layer (Step 6) that is later used for mesh generation.

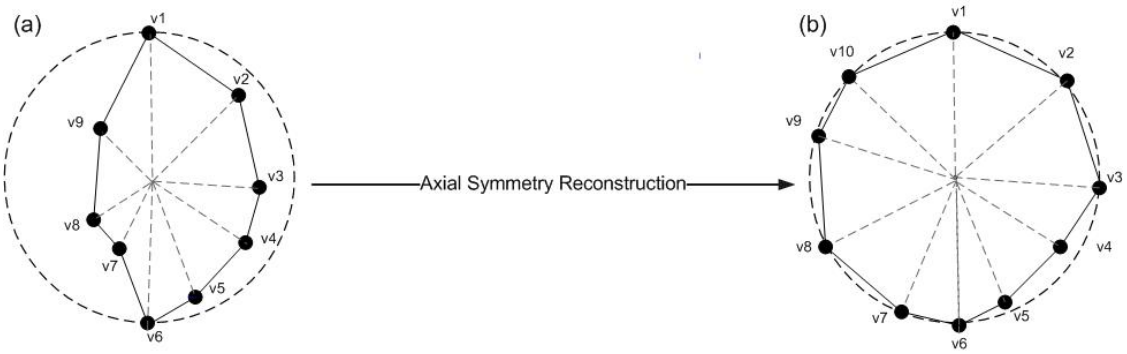


Figure 4: Symmetry analysis for axis reconstruction

In the case of a *planar* symmetry, the initial symmetry plane that divides the object into two sections containing equal number of sectors is determined. Additional analyses are performed for the layer to determine the symmetry plane that minimizes the distance differences between the point pairs in two sections. This symmetry plane is fixed, and the point pairs in the opposite sides of the symmetry plane for each layer are examined. For each pair, a point that has the smaller distance from the *z*-axis is replaced with the projection of a point with the greater distance at the opposite side of the plane.

Algorithm 5: Planar Symmetry Analysis

Input: 3D model *M*

Output: N_{layers} of 2D arrays of polylines $Polylines^{(l)}_{(i,j)}$ with dimensions $2 \times N_{layers}$

Step 1: For every layer index *l;* $l = 1, \ldots, N_{layers}$

Calculate arrays $A^{(l)}$ and $D^{(l)}$ using Alg. 2, with the input $L^{(l)} = Z_{init} + l\Delta_z$

Create an array of points *PointSaved*, with elements initialized as follows:

If $(A^{(l)}_k \neq null)$ Then $PointSaved_k = A^{(l)}_k$

Else $PointSaved_k = (0, 0, L^{(l)})$

Step 2: Initialize: $Polylines^{(l)}_{(i,j)}$ with *null* values

Step 3: Initialize two auxiliary real-number arrays *DProj* and *DOffset* with 0's.

Set $Diff_{min} = 0$, Set $Diff = 0$, Set $I_{offset} = 0$, Set $\theta_{offset} = I_{offset} + 2\pi / N_{sectors}$

Step 4: For each $i = 0, \ldots, N_{sectors} / 2$

Set $j = i \bmod N_{sectors}$

If *mirror symmetry* then Set $k = (I_{offset} + N_{sectors} - i) \bmod N_{sectors}$

If *glide reflection* then

Set $k =$ *first non-zero value of* $(I_{offset} + N_{sectors}/2 - i) \bmod N_{sectors}$

Set $\theta_1 = j \times 2\pi / N_{sectors}$, $\theta_2 = k \times 2\pi / N_{sectors}$

Set the projection and offset factors for $A^{(l)}_j$ and $A^{(l)}_k$ points from the plane

Set *Diff* value descriptor as weighted sum of projection and offset factors

Endfor

Step 5: If $Diff < Diff_{min}$ then Set $Diff_{min} = Diff$ and $I_{best} = I_{offset}$

Step 6: For each i between 0 and $N_{sectors} / 2$

Calculate j, k, θ_1, θ_2 as in Step 4 using $I_{offset} = I_{best}$

If ($Min(D^{(l)}_j, D^{(l)}_k) / Max(D^{(l)}_j, D^{(l)}_k) > \gamma_{plane}$) then

Set $Polylines^{(l)}_{(0,\ \min(D(l)j,\ D(l)j))} = PointSaved_{\min(D(l)j,\ D(l)j)}$

Set $Polylines^{(l)}_{(1,\ \min(D(l)j,\ D(l)j))} = (D^{(l)}_{\min(D(l)j,\ D(l)j)} \times \cos\theta_1,\ D^{(l)}_{\min(D(l)j,\ D(l)j)} \times \sin\theta_1, L^{(l)})$

Endfor

Step 7: Return arrays $Polylines^{(l)}_{(i,j)}$

Note, initialization Steps 1-2 are identical for both Algorithms 4 and 5. For a given layer, the algorithm traverses all the point pairs *(j, k)*, one on each side of predefined plane of symmetry, and calculates the distance differences between the point pairs in two sections (Step 4) (Figure 5a). Two different symmetry types are considered in point pair selection. When a *glide reflection* symmetry is considered, point pairs from the opposite sectors are evaluated (Figure 5b). When a *mirror* symmetry is considered, point pairs with the same offset from the symmetry plane are evaluated (Figure 5c). Analyses are performed for a layer to determine the symmetry plane that minimizes the distance differences between the point pairs in two sides of the symmetry plane (Step 5). Once this symmetry plane is found and fixed, for each point pair *(j, k)*, the point with the largest distance from the *z*-axis is found and its coordinates and distance coordinates at angle θ are recoded in the array $Polylines^{(l)}_{(i,j)}$ (Step 6). The output of the algorithm is the arrays $Polylines^{(l)}_{(i,j)}$, one for each layer that is later used for mesh generation.

The final algorithm presented in this section builds 3D meshes based on the polyline pairs generated by Algorithms 4 or 5 and is presented next.

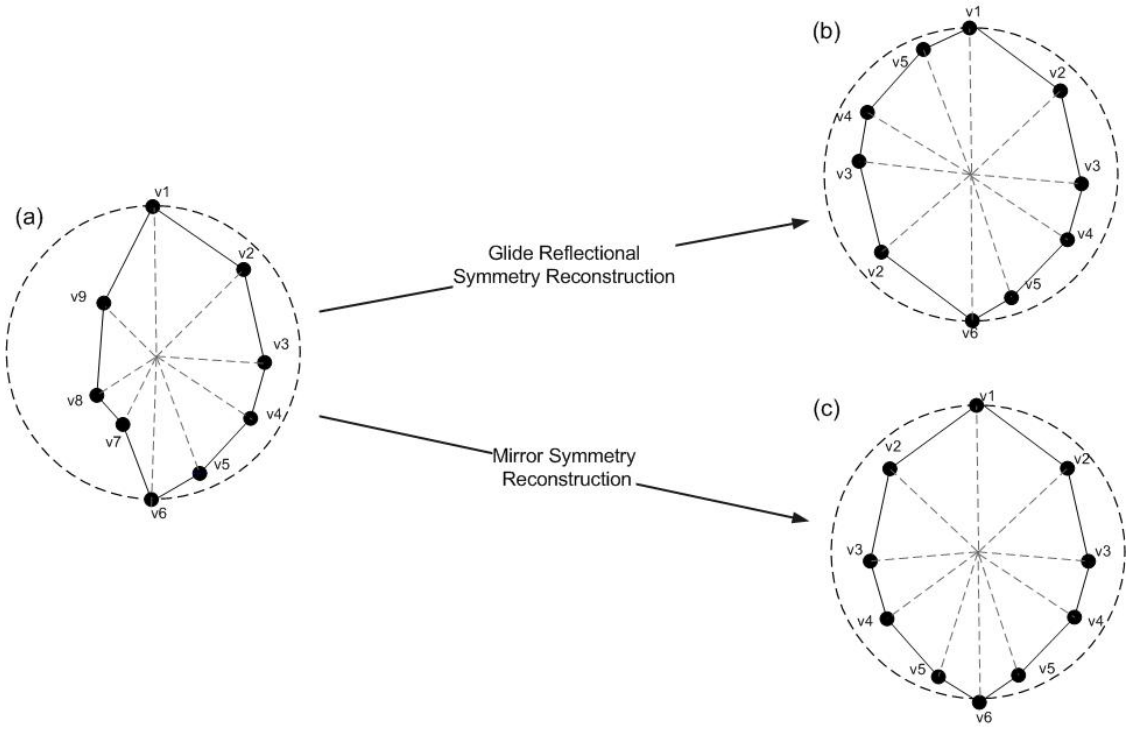


Figure 5: Symmetry analysis for planar reconstruction

Algorithm 6: 3D Mesh Generation for Missing Sections

Input: N_{layers} array of polylines $Polylines_{(i,\,j)}$

Output: 3D mesh (composed of one or more closed surfaces) *TriangleArray(max)*

Step 1: Initialize: *TriangleArray(max)* with *nulls*

Step 2: Partition the arrays $Polylines^{(l)}_{(i,\,j)}$ into intervals divided by empty polylines, i.e., find the disjoint missing sections and treat them separately.

Step 3: For every layer $l = 1,\ldots, N_{layers}$ connect pair of indices j and $j+1$ that correspond to two non-empty non-trivial polyline pairs:

(a) Connect edges and of middle sections of $Polylines^{(l)}_{(i,\,j)}$ and $Polylines^{(l)}_{(i,\,j+1)}$ and add resulting triangles to *TriangleArray.*

(b) Perform 3a step for $Polylines^{(l)}_{(0,\,j)}$ and $Polylines^{(l)}_{(1\,,j)}$.

(d) For the first and the last non-empty polyline pairs, join the edges of each pair of the polylines and triangulate the resulting polygon. Add triangles to *TriangleArray.*

Step 4: For 2 consecutive layers *l* and *l+1*, connect pairs of indexes *j* and *j+1* on both layers, add resulting triangles to *TriangleArray*.

Step 5: Generate triangular mesh by drawing each triangle in *TriangleArray*.

Algorithm 6 traverses all the points of $Polylines^{(l)}_{(i,\ j)}$ for each layer, first dividing the $Polylines^{(l)}_{(i,\ j)}$ into disconnected sections that correspond to multiple reconstructed sections of the 3D object and then treating each of those sections separately (Step 2). For each layer *l*, $Polylines^{(l)}_{(i,\ j)}$ generates six edges and forms four triangles from these edges (Step 3). Two consecutive layers *l* and *l+1* are connected and formed triangles added to *TriangleArray* (Step 4). Finally, the mesh is drawn by visualizing all triangles in *TriangleArray* (Step 5).

7. Reconstruction Examples and Computation Results

The methodology developed in this paper is illustrated with the 3D model reconstruction examples. The software that executes the slicing, model alignment, symmetry analyses, and mesh generation algorithms is used to illustrate the application. The software is written in Visual C++, supported by the Oracle database, and it is part of larger expert system software developed to assist rapid manufacturing and repair of damaged mechanical parts. The software database includes more than one hundred models that were developed by various techniques. The damaged (incomplete) instance of a 3D model is an input to the model reconstruction system (see Figure 7). The objective is to perform symmetry analysis and reconstruct missing sections of the original object.

The reconstruction results of three different mechanical parts are summarized in Table 1. For each of the three parts in Table 1, independent parameters N_{layers}=80 and $N_{sectors}$= 60 are used. The animation snapshots are shown when all the points of reconstructed mesh are generated. The layer-based structure used throughout the reconstruction process can also be observed in Table 1. After the triangular mesh is generated, the missing sections presented in the polygonal mesh are filled and presented in the final results. The highlighted (light blue) sections on the models show missing sections of the input parts. This information is translated to a higher feature-level description of missing sections and

is used to generate expert advice for the repair and rebuilding process of damaged objects with the spray forming technology process. For this purpose, the object is sliced into several layers (see Figure 8), a tool path is generated for each layer, and fabrication is performed by building damaged sections layer by layer.

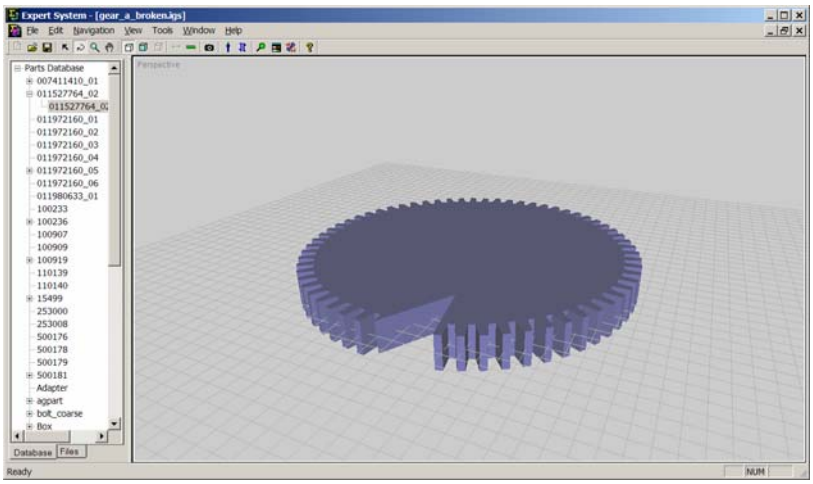


Figure 7: Input broken part selected from the database

Computation analysis of the algorithm execution time and the total time (includes visualization) for three different combinations of independent parameters ($N_{sectors}$, N_{layers}) are provided in Table 2 and performed on an Intel Core™ 2, 1.66GHz machine. For 3D models of various complexities, the execution time is in the range of one to eighteen seconds, averaging about five seconds for most typical models. Computation results show that the execution time has linear dependency on the independent parameters ($N_{sectors}$, N_{layers}). For example, from Table 1 one may see that for all three models considered in this study, the increase in independent parameters (N_{sector},,N_{layers}) produces similar rate of increase in the execution time. Also, analysis did not reveal clear dependencies between independent parameters and the total time, as the latter mostly depends on machine characteristics. Finally, analyses show that the execution time not only depends on the triangles count, but also on the size of the missing section, as the latter determines the number of points needed for the layer approximation. The analysis results show that even for complex models, such as the Model 2 in Table 1, which has largest number of triangles and missing volume, it takes less than ten seconds to reconstruct the part even for large values of N_{sector}=240 and N_{layers}=320 (see Table 2).

Table 1: Symmetric reconstruction examples for three different models

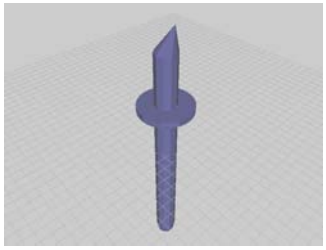
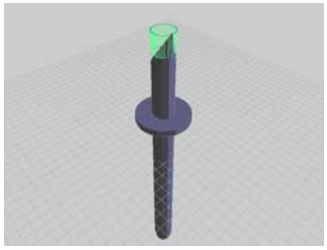
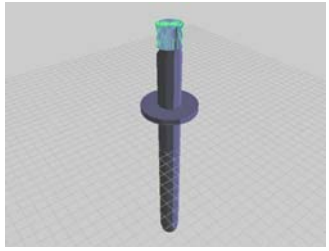
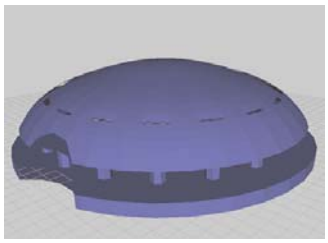
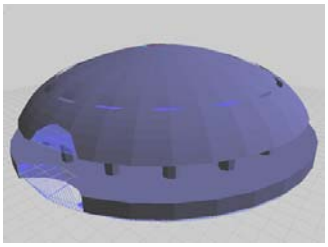
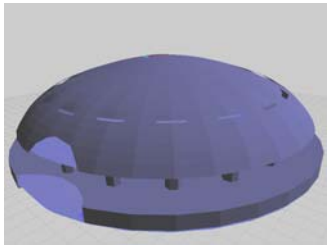
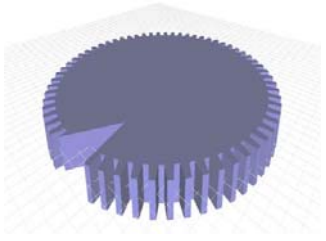
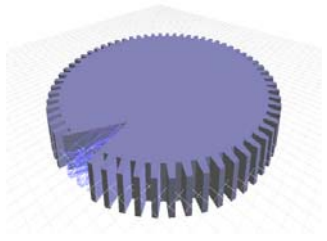
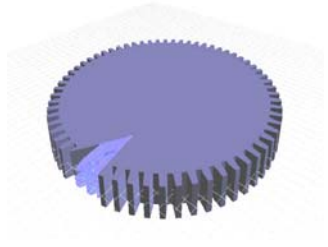
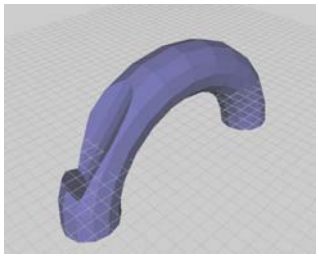
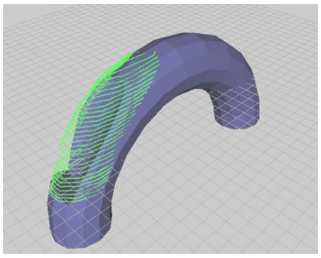
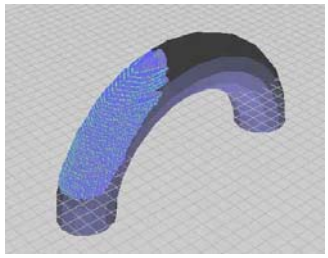
	Input model	Reconstructed triangular mesh	Reconstructed solid based on the mesh
MODEL 1 - AXIAL RECONSTRUCTION			
MODEL 2 - MIRROR RECONSTRUCTION			
MODEL 3 - PLANAR RECONSTRUCTION			
MODEL 4 - PLANAR RECONSTRUCTION			

Table 2: Computation time for the symmetric reconstruction examples in Table 1

	MODEL 1	MODEL 2	MODEL 3	Model 4
Triangle count of the model	462	1435	1057	230
$N_{sectors}$/ N_{layers}	Execution time (ms)			
60/80	3797	6015	3453	1863
120/160	5969	12016	4735	2270
240/320	9313	17922	7469	5860

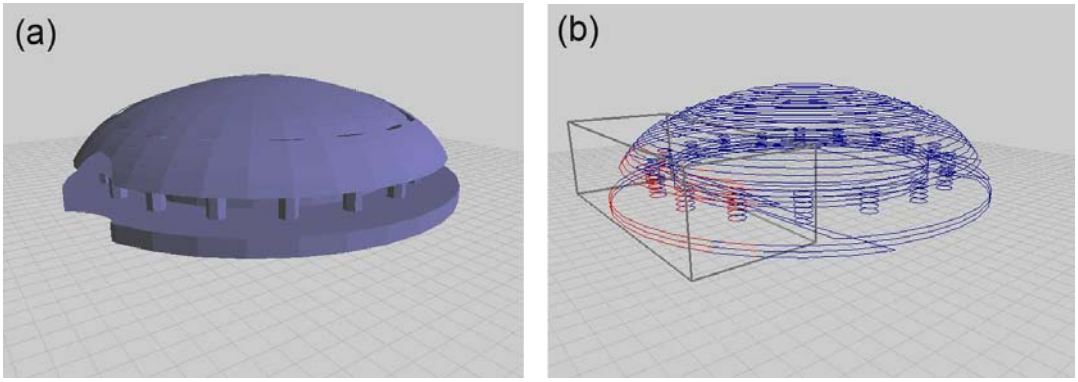


Figure 8: Layered representation of reconstructed model for rapid repair: (a) damaged 3D model, and (b) layered representation of the reconstructed part

Although the symmetric reconstruction examples presented in this section are from manufacturing arena, the approach presented in this paper has a wider applicability. For example, the approach can be used in 3D shape matching systems or in the systems designed for the shape-based search of 3D models to improve their search and shape matching accuracy (Osada et al. 2001, Avagyan et al. 2007). In these systems 3D model is provided as an input to the shape matching system to search the database for related or identical models with the purpose of extracting useful information. When incomplete models are provided to the system, matching search results may produce inaccurate outcomes. Therefore, first the approach presented in this paper can be used to reconstruct incomplete 3D model before search and shape matching activities are performed. The approach can also be used in new and emerging field of 3D shape reconstruction of bones from CT scan images for medical applications (see Zheng 2006, Shuxian et al. 2005). Finally, approach presented in this paper should be used for reconstruction of low complexity and symmetric models for which the complete mathematical accuracy is not highly important. More complex 3D models, e.g., whole engineering devices or machines, complex assemblies fall out of the applicability scope of the method.

8. Conclusion

In this paper, a new approach and algorithms for 3D model reconstruction based on the model symmetric features was presented. The approach has continuously used a model slicing procedure to produce an overall assessment of the model's positioning and

geometric properties. Once the 3D model was introduced to the system, its symmetry type and model parameters were established, and an algorithm was developed for aligning the model at the rotation that produced the maximum number of symmetric layers. Symmetry analysis algorithms that used steps from the slicing procedure were developed to collect information on the asymmetric/missing sections of the aligned model on a layer-by-layer basis. The information on missing sections was stored in polyline pairs, which were used to generate 3D meshes. The methodology developed in this paper was illustrated with the 3D model reconstruction examples. A computation study on the algorithms performance was also presented.

References:

Assfalg J., Del Bimbo A., and Pala P., Curvature maps for 3D CBR. Proc. of the Int. Conference on Multimedia and Expo (ICME'03), 2003, July, Baltimore, MD.

Attalah M. J., On symmetry detection. *IEEE Transactions on Computers*, 1985, vol. C-34, no. 7, pp. 663-666.

Avagyan V., Zakarian A., and Mohanty P., Scanned 3D model matching and comparison algorithms for manufacturing applications. *ASME Journal of Manufacturing Science and Engineering*, vol. 129, no. 1, 2007, pp. 190-201.

Barequet G. and Kumar S., Repairing CAD models, *Proceedings of the 8th Conference on Visualization '97***,** pp. 363 - 371.

Bischoff S. and Kobbelt L., Structure preserving CAD model repair, *Computer Graphics Forum*, 2005, vol. 24, no. 3, pp. 527 -- 536.

Burel G. and Henocq H., Determination of the orientation of 3D objects using spherical harmonics. *Graphical Models and Image Processing*, 1995, vol. 57, no. 5, pp. 400--408.

Chen D. Y., Tian X. P., Shen Y. T. and Ouhyoung M., On visual similarity based 3D model retrieval. *Proceedings of Eurographics Computer Graphics Forum*, 2003, vol. 22, no. 3, pp. 224-232.

Cybenco G., Bhasin A., and Conhen K. D., Pattern recognition of 3D CAD objects. *Smart Engineering Systems Design*, 1997, vol. 1, pp. 1–13.

Del Bimbo A. and Pala P., Content-based retrieval of 3D models. *ACM Transactions on Multimedia Computing, Communications and Applications,* 2006, vol. 2, no. 1, pp. 20-43.

Elad M., Tal A., and Ar S., Content based retrieval of VRML objects: An iterative and interactive approach. *Proceedings of the 6th Eurographics Workshop on Multimedia,* 2001, pp. 107–118

Erturk S. and Dennis T., J., 3D model representation using spherical harmonics. *Electronics Letters*, 1997, vol. 33, no. 11, pp. 951-952.

Gal R. and Cohen-Or D., Salient geometric features for pPartial matching and similarity. *ACM Transactions of Graphics,* 2006, vol. 25, no. 1, pp. 130-150.

Kazhdan M., Funkhouser T., and Rusinkiewicz S., Rotation invariant spherical harmonic representation of 3D shape descriptors. *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 2003, vol. 43, pp. 156-164.

Kazhdan M., Chazelle B., Dobkin D., Funkhouser, T. and Rusinkiewicz S., A reflective symmetry descriptor for 3D models. *Algorithmica,*, 2004(a), vol.38, no.2, pp. 201-225.

Kazhdan M., Funkhouser T., and Rusinkiewicz S., Symmetry descriptors and 3D shape matching. *Symposium on Geometry Processing*, 2004(b), pp.116-125.

Knuth D. E., Morris Jr., J. H., and Pratt V. R., Fast pattern matching in strings. *SIAM Journal on Computing*, 1977, no. 6, pp.323-350.

Martinet A., Soler C., Holzschuch N., and Sillion Artis F., Accurate detection of symmetries in 3D shapes. *ACM Transactions of Graphics*, 2006, vol. 25, no. 1, pp. 439-464.

Ohbuchi R., Nakazawa M., and Takei, T., Retrieving 3D shapes based on their appearance. *Proceedings of MIR*, 2003, pp. 39–46.

Osada R., Funkhouser T., Chazelle B., and Dobkin D., Matching 3D models with shape distributions. *Proc of the Int. Conference on Shape Modeling & Applications*. IEEE Computer Society Press, 2001, p. 154

Osada R., Funkhouser T., Chazelle B., and Dobkin D., Shape distributions. *ACM Transactions on Graphs*, 2002, vol. 21, no. 4, pp. 807–832.

Podolak J., Shilane P., Golovinskiy A., Rusinkiewicz S., and Funkhouser T., A Planar-reflective symmetry transform for 3D shapes. *Int. Conference on Computer Graphics and Interactive Techniques* (*ACMSIGGRAPH)*, 2006, vol. 25, no.3, pp. 549 – 559.

Shilane P. and Funkhouser T., Selecting distinctive 3D shape descriptors for similarity retrieval, *Proceedings of the IEEE International Conference on Shape Modeling and Applications*, 2006, pp. 108-117.

Shuxian Z., Wanhua Z., and Bingheng L., 3D reconstruction of the structure of a residual limb for customizing the design of a prosthetic socket. *Medical Engineering & Physics*, 2005, vol. 27, no. 1, pp. 67 – 74.

Sun Ch. and Sherrah J., 3-D Symmetry detection using the extended gaussian image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997, vol. 19, no. 2, pp. 164-168.

Vicent A. P., Martin R.R., and Company P., Skewed mirror symmetry for depth estimation in 3D line-drawings. *GREC*, 2003(a), pp.142-153.

Vicent A. P., Company Calleja P., and Martin R.R., Skewed mirror symmetry in the 3D reconstruction of polyhedral models. *Winter School on Computer Graphics*, 2003(b), vol. 11, no. 3, pp. 504-511.

Wolter J. D., Woo T. C., and Volz R. A., Optimal algorithms for symmetry detection in two and three dimensions. *The Visual Computer*, 1985, no. 1, pp. 37-48.

Zabrodsky H., Peleg S., and Avnir D., Symmetry as a continuous feature. *IEEE Transactions on Pattern. Analysis Machine Intelligence*, 1995, vol. 17, no. 12, pp. 1154–1166.

Zheng G., Reconstruction of patient-specific 3D bone model from Biplanar X-ray images and point distribution models. *IEEE International Conference on Image Processing*, October, 2006, pp. 1197-1200, Atlanta, GA.