

- 2002 International Computer Symposium (ICS2002) Workshop on Multimedia Technology
- Paper title: A 3D Model Alignment and Retrieval System
- Abstract: Techniques for 3D model alignment and retrieval are proposed in this paper. Since the techniques of 3D modeling and digitizing tools are in great demand, the expectations of 3D models alignment and retrieval are increasingly. We propose an algorithm for 3D model alignment, which gets the affine transformation between two 3D models. The main idea of our 3D alignment algorithm in rotation is to search the similarity of projected 2D shapes from each viewing aspect of two models. Then, we apply the technique to match two 3D models after recovering the affine transformation.
- Author 1: Ding-Yun Chen
Communication and Multimedia Lab, Department of Computer Science and Information Engineering National Taiwan University, Taiwan
Room 505, Dept. of CSIE, NTU, 1 Roosevelt Rd. Sec. 4, Taipei, 106 Taiwan
Phone: 886-2-23625336 Ext.505
Fax: 886-2-23625336 Ext.507
Email: dynamic@cmlab.csie.ntu.edu.tw
- Author 2: Ming Ouhyoung
Communication and Multimedia Lab, Department of Computer Science and Information Engineering National Taiwan University, Taiwan
Room 421, Dept. of CSIE, NTU, 1 Roosevelt Rd. Sec. 4, Taipei, 106 Taiwan
Phone: 886-2-23625336 Ext.421
Fax: 886-2-23628167
Email: ming@csie.ntu.edu.tw
- Contact author: Ming Ouhyoung
- Keywords: 3D model alignment, 3D model retrieval, content-based retrieval

A 3D Model Alignment and Retrieval System

Ding-Yun Chen and Ming Ouhyoung

Department of Computer Science and Information Engineering,

National Taiwan University, Taipei, Taiwan

dynamic@cmlab.csie.ntu.edu.tw, ming@csie.ntu.edu.tw

Abstract

Techniques for 3D model alignment and retrieval are proposed in this paper. Since the techniques of 3D modeling and digitizing tools are in great demand, the expectations of 3D models alignment and retrieval are increasingly. We propose an algorithm for 3D model alignment, which gets the affine transformation between two 3D models. The main idea of our 3D alignment algorithm in rotation is to search the similarity of projected 2D shapes from each viewing aspect of two models. Then, we apply the technique to match two 3D models after recovering the affine transformation.

1. Introduction

The problem of 3D objects recognition, retrieval, clustering and classification is a traditional research topic during previous decades in computer vision, mechanical engineering, medical imaging and molecular biology. The research topic is important in computer graphic because the techniques of 3D modeling and digitizing tools are in great demand. Many tools of digitized and constructed 3D objects are getting more and more popular, for example, 3D model acquisition systems [14], 3D model capturing systems [17], 3D freeform design

systems [12] and sculpting systems [13]. Therefore, 3D objects can be digitized and modeled easier, faster and less expensive. A large number of free 3D models can be accessed all over the world via the Internet, such as in [15, 16]. Although text-based search engines are ubiquitous today, multimedia data such as 3D models lack meaningful and semantic description for automatic matching. The MPEG group aims to create an MPEG-7 international standard, also known as “Multimedia Content Description Interface”, for the description of the multimedia data, including image, video, audio, 2D shapes and 3D objects [11]. However, there is currently only one descriptor for 3D model. This has highlighted the need for developing efficient techniques of content-based retrieval for 3D model.

The problem of 3D model retrieval can be stated as follows: given a 3D model, the retrieval system compares it with all other 3D models from the database, and shows ranked similar models. In short, the problem is to determine the similarity between two given 3D models. The most important issue is to extract suitable features for matching. The feature should represent the characteristics of different 3D models, and should be invariant to translation, rotation and scaling, and robust against re-meshing,

subdivision as well as simplification, noise and deformation. The second important issue is to define a meaningful distance metric, which should be efficient.

Most previous works of 3D model retrieval focus on finding a good feature for matching [1~10]. Most of those are based on either statistical properties, such as global shape histograms, or the skeletal structure of 3D model. Zhang and Chen [2] propose an algorithm to efficiently calculate features, such as volume, moments, and Fourier transformation coefficients. In many applications, there is a high demand to calculate these important features for a 3D model. Volume-surface ratio, aspect ratio, moment invariants and Fourier transformation coefficients [3] are often used in 3D model retrieval. In their current system, they simply normalize the features and use Euclidean to measure the similarity. The total number in their database is around 2,000 models, which are in VRML format.

Osada et al. [5, 6] propose and analyze a method for computing shape signatures for arbitrary (possibly degenerate) 3D polygonal models. The key idea is to represent the signature of an object as a shape distribution sampled from a shape function measuring global geometric properties of an object. The primary motivation for this approach is to reduce the shape matching problem to the comparison of probability distributions, which is simpler than traditional shape matching methods that require pose registration, feature correspondence, or model fitting. More

specifically, they have experimented with five different shape functions, and the D2 shape function can classify 3D objects better than the other shape functions. The D2 shape function is defined as follows: measures the distance between two random points on a surface. In addition, the entire shape distribution is scaled based on the mean in order to deal with the scaling problem. Finally, they examine that the PDF L1 norm performed the best for comparing shape distributions. In their experimental results, they achieve 60% accuracy with a diverse database of degenerate 3D models. They also compare D2 shape distribution method against surface moments, and find the D2 shape distributions outperform moments for classification of 3D models. The approach is simple and fast, and robust to scaling, rotation, mirror, noise, re-mesh, simplification, deleting and inserting polygon. They test the algorithm using 133 models now, and they will test for larger database in the future.

Hilaga et al. [1] propose a technique in which similarity between polyhedral models is quickly, accurately, and automatically calculated by comparing the skeletal and topological structure. Therefore, their algorithm can handle the global and local properties simultaneously. The skeletal and topological structure decomposes to a one-dimensional graph structure. The graph is invariant to translation, rotation and scaling, robust against connectivity changes caused by simplification, subdivision and re-meshing, and resistant against noise, certain changes due to deformation, such as

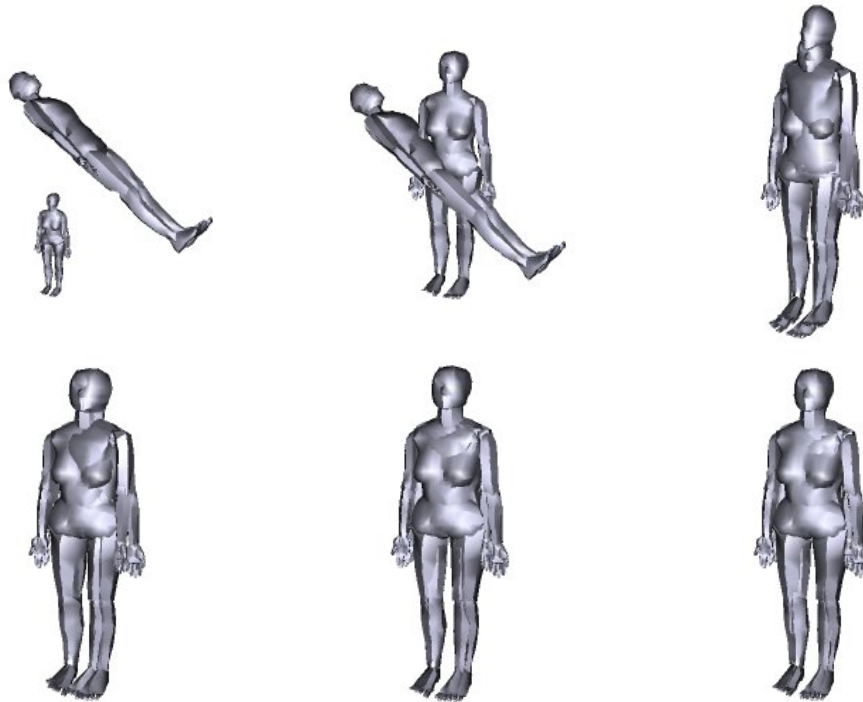


Fig. 1 The order of 3D models alignment.

an articulated object change its posture. Their search key is a multi-resolutional structure of the graph, so that the comparison can simply and fast. Their experiments made use of 230 different polyhedral meshes. In their experimental results, the search key for a mesh of 10,000 vertices can be calculated in approximately 15 seconds with a Pentium II 400MHz processor. The average search time is about 12 seconds, that is, it took only 0.05 second in average to calculate one similarity.

In general, features of 3D models should be invariant to affine transformations, since each 3D model has its own coordinate axis for different use. In contrast, we propose an algorithm to recover translation, scaling and rotation between two 3D models, and then extend the technique to measure the similarity. Furthermore, the function of 3D model alignment can not only be used in

3D model retrieval, but also in many other applications, such as mesh watermarking, 3D model morphing, 3D animation, and so on.

The main idea of our 3D alignment algorithm in rotation is to render 2D silhouettes from each viewing aspect of two models, and get rotation which has minimum error summing from all viewing aspect using 2D shape matching algorithm. Our approach of 3D model retrieval takes the minimum error as the similarity between two 3D models. The remaindered part of this paper is organized as follows. In Chapter 2, we propose an algorithm to do 3D model alignment. We detail rotation alignment in Chapter 3. The experimental results of 3D model alignment are represented in Chapter 4. 3D model retrieval, one application of 3D model alignment, is proposed in Chapter 5. Finally, the paper is

summarized and concluded in Chapter 6.

2. Flow of 3D Model Alignment

The order of 3D model alignment is as follows: $TS \rightarrow R_c \rightarrow TS \rightarrow R_r \rightarrow TS$. Where TS denotes the translation and scaling alignment of two models; R_c denotes the coarser rotation alignment and R_r refine the rotation. All TS apply the same operator, that is, translate to the same origin and scale to the same size between two models. The purpose of first two TS is to let two models be roughly in similar position and of the same size, which will make it easier to get the correct rotation R_c and R_r . Once the correct rotation is recovered, the last TS will be easier to get the correct translation and scaling. Fig. 1 shows an example of the five steps.

The approach of translation and scaling is very simple. The translation $T=(T_x, T_y, T_z)$ assigns the middle point of the whole model to be the new origin. The scaling is isotropic, and normalizes according to the maximum distance from x, y and z axes of the whole model. That is,

$$T_i = \frac{MaxCoor_i + MinCoor_i}{2}, i = x, y, z \quad (1)$$

$$S = \frac{1}{\max_{i=x,y,z}(MaxCoor_i - MinCoor_i)} \quad (2)$$

where the $MaxCoori$ and $MinCoori$ are the maximum and minimum coordinate value of i axis, respectively.

An intuitional thought of recovering the rotation from two models is to rotate

model to all possible viewing angles, and get the rotation that has minimum error from all viewing angles. We take Fig. 2 as a typical example to explain the idea. There are two models, *pig* and *cow*, with different rotations, and both models have been applied coarser translating and scaling (TS) alignment. To recover the rotation from model *cow* to model *pig*, a set of cameras surrounding a model to render 2D shapes from each viewing angle. Those cameras are put on the surface of a sphere and scatter viewing angles all over the sphere. Fig. 2 (a) shows a set of cameras surrounding the model *pig*, where each intersection point indicates a camera position. Then, apply those camera set to the model *cow*, as shown in Fig. 2 (b), and calculate the difference of 2D shape for each camera pair. We define the error of the two models in a specific rotation as summing the difference of 2D shapes for all camera pairs. Therefore, the goal is to find a rotation that has the minimum error from all rotation angles of the camera set. That is,

$$\min_i \sum_j ShapeDiff(j), \quad (3)$$

i : rotation angle of the camera set

j : camera pair between two models

where $ShapeDiff$ denotes the difference of two 2D shapes. Fig. 2 (b)~(f) show various rotation angles of a camera set, and we suppose that Fig. 2 (e) will get the minimum error, since the rotation matrix of the two models can be calculated from the rotation of the two camera sets between Fig. 2 (a) and (e).

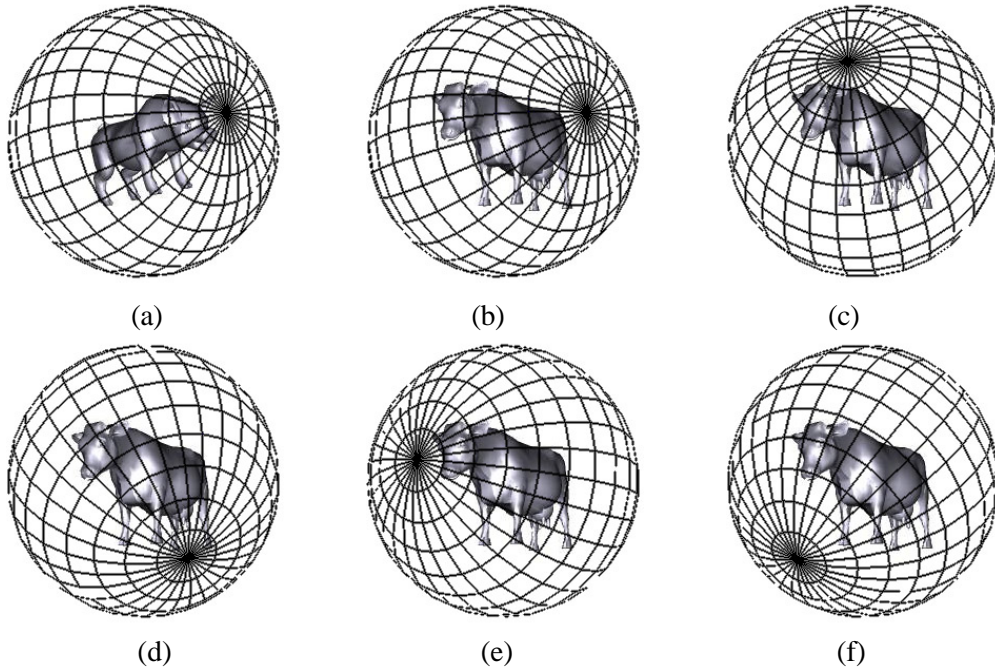


Fig. 2 A typical example to show our algorithm.

When rotating the camera set to a new orientation, all 2D shapes should be rendered from all new cameras position, and the similarity of 2D shapes between each camera pair have to be calculated. This will cause large amount of calculation, and is time consuming. Therefore, we put the camera set in the vertices of a regular convex polyhedron, so that the number of rendering 2D shapes and calculating the similarity between them will be greatly reduced.

There are only five regular convex polyhedrons, which are named as Platonic bodies, and was known to the ancient Greeks. The fact can also be proved using Euler's theorem. The five regular convex polyhedrons are tetrahedron, hexahedron or cube, octahedron, dodecahedron, and icosahedron. We take vertices of dodecahedron, which has the maximum

vertices from five regular convex polyhedrons, as the position of the camera set. There are 20 scattering viewing aspects for each 3D model. The set of 20 2D shapes, rendered from the 20 cameras, is a basis for each 3D model to align between two 3D models, and contains knowledge from various viewing aspects for a 3D model. Fig. 3 explains the reason that we can reduce the number of calculation from rendering and 2D shapes matching by using the dodecahedron. Fig. 3 (a) shows a camera set of model *pig*, and the same camera set applying to model *cow* shows in Fig. 3 (b). That is, the indices of camera set are all the same between Fig. 3 (a) and (b). In addition, we can rotate the dodecahedron resulting in the camera set is at the same position. For instance, rotate edge (1,2) from Fig. 3 (b) to position of edge (1,3) and (1,4), which show in Fig. 3 (c) and (d), respectively. Since a dodecahedron has 20 vertices and each

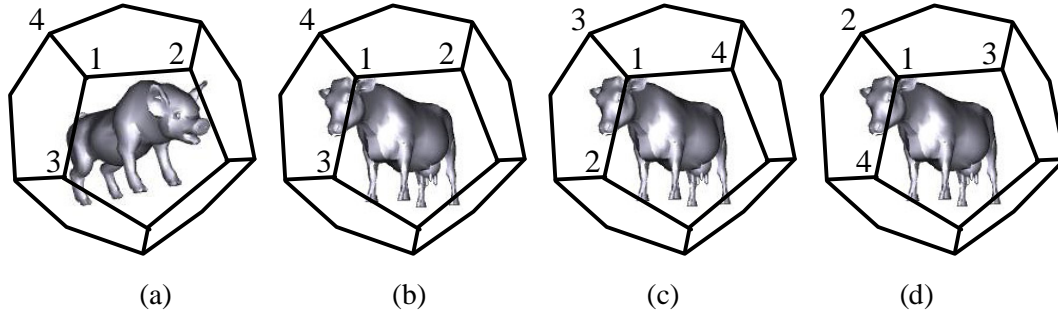


Fig. 3 We can reduce the number of calculation from rendering and 2D shapes matching by using the dodecahedron.

vertex connects 3 edges, there are 60 kinds of different rotation, which share the same 20 camera positions. Table 1 shows the number of rendering and 2D shapes matching for 60 different rotations with and without using dodecahedron. Without using dodecahedron, one model should render 20 times for a camera set, and another model should render 60 camera sets, that is, 1200 times. The number of rendering can reduce to 40 times by using dodecahedron, 20 times for each model. On the other hand, calculation of 2D shapes matching requires 1200 times without using dodecahedron, 20 times for each rotation. The number of 2D shapes matching is 400 by using dodecahedron, because there are 20 shapes for each model. The table shows the reason why we use dodecahedron.

For testing 60 kinds of different rotation	Number of rendering	Number of 2D shapes matching
Without using dodecahedron	$20 + 20 \times 60 = 1220$	$20 \times 60 = 1200$
Using dodecahedron	$20 + 20 \times 1 = 40$	$20 \times 20 = 400$
Ratio	30.5	3

Table 1 Number of rendering and 2D shapes matching for 60 different rotations with and without using dodecahedron.

There are 60 different rotations to test by using one camera set of dodecahedron for both models. However, it's usually not enough to recover rotation from the best solution of the 60 candidates for the coarser rotation alignment, R_c . The coarser rotation alignment should provide a good initial, so that the refined rotation alignment, R_r , can easily get the best result from the local estimation. Therefore, we can use more camera sets from different dodecahedrons. There will increase 60 different rotations when adding one camera set of dodecahedron. If apply one dodecahedron to first model and apply ten dodecahedrons to another, there will be 600 different rotations. Furthermore, we can also apply more than one camera set of dodecahedron. That is, when applying m dodecahedron to first model and n dodecahedron to another, there will be $60 \times m \times n$ kinds of different rotation. However, the more dodecahedrons are used, the more computation is. Table 2 shows the number of rendering and 2D shape matching by using different dodecahedrons. In our current implementation, we use $m=10$ and $n=10$, that is, we take the best one from 6000 different rotations as the coarser

rotation alignment, R_c . The algorithm of rotation alignment, R_c and R_r , will detail in next chapter.

Since we use more than one dodecahedron, the way to scatter the dodecahedrons is also consideration. The purpose is to scatter the camera sets to whole rotation space, so that any possible rotation will close to a candidate. If n dodecahedrons should be scattered, we use iterative approach to get the best one by:

$$\max_n \sum_j MinDist(n, j) \quad (4)$$

where j is index of dodecahedron vertex, and $MinDist(n, j)$ return the minimum distance from j vertex of n dodecahedron to all vertices of other dodecahedrons. That is, iteratively rotate each dodecahedron from larger to smaller rotation angle, so that all vertices of all dodecahedrons are as scattering as possible. The approach is not effective, however, the pre-processing stage only need to be run once.

In the end of this chapter, we detail the operative flow of both models. There are

two 3D models A and B , and the operations, which translate, scale and rotate 3D model B to align 3D model A , show in the follows:

$$A' = A \cdot T_A \cdot S_A$$

$$B' = B \cdot T_B \cdot S_B$$

$$R_c = RotateCoarse(B', A')$$

$$B'' = B' \cdot R_c \cdot T_B \cdot S_B$$

$$R_r = RotateRefine(B'', A')$$

$$A' \sim B'' \cdot R_r \cdot T_B \cdot S_B$$

$$A \cdot T_A \cdot S_A \sim B' \cdot R_c \cdot T_B \cdot S_B \cdot R_r \cdot T_B \cdot S_B$$

$$A \sim B \cdot T_B \cdot S_B \cdot R_c \cdot T_B \cdot S_B \cdot R_r \cdot T_B \cdot S_B \cdot S_A^{-1} \cdot T_A^{-1}$$

where *RotateCoarse* and *RotateRefine* recover the coarser and refined rotation, respectively, and detail in next chapter.

3. 3D model Alignment in Rotation

This chapter details our approach to align rotation from two models. The rotation alignment divides into two parts: coarse and refined alignment. The coarse alignment gets the approximate rotation from all possible orientation between two models.


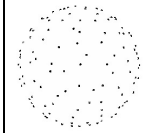
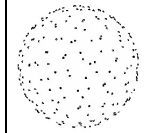
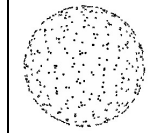
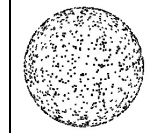
$(m-n)$	1-1	1-10	1-20	1-40	10-10
Number of rendering ($20 \times m + 20 \times n$)	40	220	420	820	400
Number of 2D shape matching ($20 \times m \times 20 \times n$)	400	4000	8000	16000	40000
Number of different rotations ($60 \times m \times n$)	60	600	1200	2400	6000
Vertices of scattering dodecahedrons					

Table 2 Number of rendering and matching 2D shapes are calculated by mapping m to n different dodecahedrons.

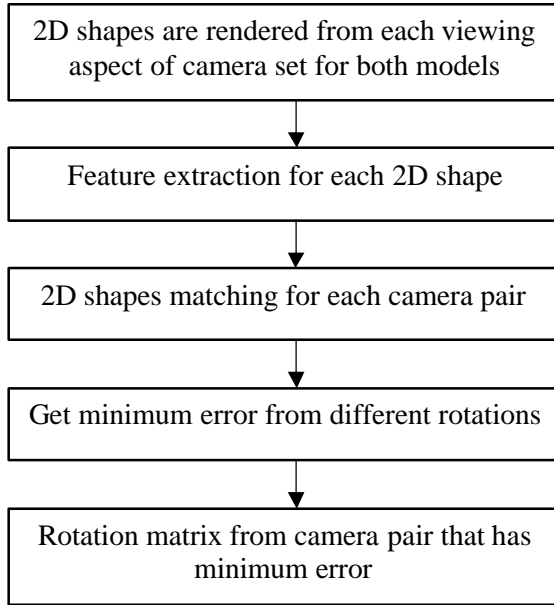


Fig. 4 The flow of rotation alignment

The refined alignment adjusts approximate to accurate rotation from neighbor orientation. Since the position and size of both models are approximate, not exactly the same, the approach should be invariant to translation and scaling.

We align rotation between two models by matching 2D shapes from camera set of dodecahedron. Fig. 4 shows the flow of rotation alignment. First, 2D shapes should be rendered from camera set of dodecahedron for both models. For each 2D shape, feature can be extracted for matching later. The operation of rendering and feature extraction do 40 times respectively, if using 1-1 dodecahedron, that is, one camera set of dodecahedron for both models. Then 2D shapes of each camera pair are matched, and the operation does 400 times if using 1-1 dodecahedron. Next, get minimum error form different rotations, as defined in Eqn. (3). Finally, once two camera sets of dodecahedron with minimum error are

determined, the rotation matrix of two models can be obtained by the rotation of the two dodecahedrons. The main flows of coarser and refined rotation alignment are the same.

The character of 2D shape depends on which matching algorithm is used. We use OpenGL to render 2D silhouette by putting camera to vertex of dodecahedron and facing to origin. The size of 2D silhouette is 256 by 256 pixels. Since 3D models should be translated, T , and Scaled, S , before rotation alignment, it's easier to make sure that whole 3D models will be rendered into 2D silhouette, that is, no clipping happen. Fig. 5 shows a typical example of 2D silhouettes from a camera set of dodecahedron. In our implementation, we render to screen by perspective projection, and then use *glReadPixels()* to copy 2D silhouettes to memory.

To measure the similarity between two shapes, we use region-based shape descriptor of the MPEG-7 [11] to match. The matching algorithm can be invariant to translation, scaling and rotation in 2D shapes, and allowable of minor non-rigid deformations. The region-based shape descriptor makes use of all pixels constituting the shape, so that it can describe complex shape including holes and several disjoint regions. The descriptor utilizes a set of ART (Angular Radial Transform) coefficients to describe the shape. The ART is a 2D complex transform defined on a unit disk in polar coordinates. Twelve angular and three radial functions are used, and 35



Fig. 5 A typical example of 2D silhouettes from a camera set of dodecahedron.

ART coefficients of 2D shapes are used for matching.

There are several notices for using the 2D shape matching to our approach. In general, in order to invariant to translation in pure 2D case, the center of mass in 2D shape should be aligned to the center of the unit disk. However, our final alignment is in 3D case, so it's no reason to align the center for each 2D shape. Since translating 3D model to origin has applied before rotation alignment, we use center of rendered 2D shape as the center of the unit disk. Furthermore, in order to invariant to scaling, linear interpolation is applied to align between rendered 2D shapes from each viewing aspect and the unit disk. The same as translation, each 2D shapes in a camera set should have the same scaling. Finally,

quantization is applied to the ART coefficients for 2D shapes matching. However, we didn't quantize the ART coefficients for more accurate.

After feature extraction for each 2D shape, shape matching for each shape from two models is calculated. Number of feature extraction is the same as rendering, but the number of shape matching is much more. In general, the computation of matching is much less than that of feature extraction in order to speedy retrieval from a large database, since feature can be previously calculated and saved to database. The region-based shape matching algorithm use simple L1 distance to measure similarity:

$$ShapeDiff((A, B)) = \sum_i |ArtM_A[i] - ArtM_B[i]| \quad (5)$$

where $ArtM$ is the ART coefficients, $ShapeDiff$ is the same in the Eqn. (3); A and B are two 2D shapes for matching; i is index of ART coefficients. Therefore, the 2D shape matching is speedy.

Next, get minimum error from different rotations of dodecahedron, as defined in Eqn. (3). The error between different rotations is defined as summing distances from all 2D shapes pair of dodecahedron. For each camera set of dodecahedron pair, there has 60 different rotations. However, there is a little difference in this stage between coarser and refined rotation alignment. In coarser rotation alignment, we use 10-10 dodecahedrons, that is, there are 100 kinds of dodecahedron pair, so that 6000 different rotations are tested. In refined rotation alignment, we use iterative approach to close the best solution. We start from 10° and step half for each iterative until less than 1° . In each iterative, we adjust rotation of one axis and fix that of another two axes in the order of X, Y and Z axis, respectively. When adjusting rotation of one axis, we rotate the dodecahedron to the direction, which has less error, until no improvement. Therefore, we can align the rotation with error less than 1° .

Finally, rotation matrix between camera set of dodecahedron pair, that has minimum error, should be calculated. The rotation matrix is then applied to one model in order to align rotation to another. The problem of solving the rotation matrix can be considered as aligning an edge between two dodecahedrons, because all edges are

aligned if one edge is aligned. We utilize the function of coordinate conversion between the Cartesian and an arbitrary coordinate system to obtain the rotation matrix. We use Fig. 6 to explain our approach. Fig. 6 (a) and (c) are the dodecahedron pair of model A and B , respectively. The rotation matrix aligns edge (1,2) in Fig. 6 (c) to that in (a). The vector $\vec{o1}$ and $\vec{o2}$ can form a unique coordinate frame, defined as follows:

$$\begin{cases} \vec{x} = \vec{o1} \\ \vec{y} = \vec{z} \times \vec{x} \\ \vec{z} = \vec{o1} \times \vec{o2} \end{cases} \quad (6)$$

where “ \times ” denotes cross produce. The notation F_A and F_B denote the coordinate system of model A and B , respectively, and F_C denote the Cartesian coordinate system. Therefore, the rotation matrix is the coordinate conversion from F_B to F_A , that is, F_{BA} . However, 3D models are in Cartesian coordinate system, F_C , so we cannot apply F_{BA} to model B directly. Model B should be converted to F_B coordinate system first and back to Cartesian coordinate system after applying F_{BA} . The rotation matrix is defined as:

$$\begin{aligned} & F_{CB} \cdot F_{BA} \cdot F_{BC} \\ = & F_{CB} \cdot F_{AC} \cdot F_{CB} \cdot F_{BC} \\ = & F_{CB} \cdot F_{AC} \end{aligned}$$

The F_{BA} can be obtained by $F_{AC} \cdot F_{CB}$, and $F_{CB} \cdot F_{BC}$ can be eliminated, so the rotation matrix from model B to model A is $F_{CB} \cdot F_{AC}$.

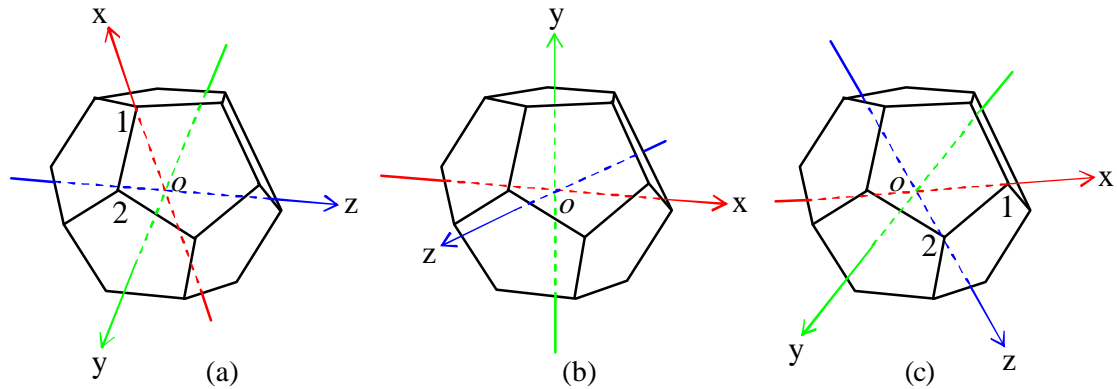


Fig. 6 Rotation matrix is calculated between two camera sets.

4. Experimental Results of 3D Alignment

In order to experiment with the 3D alignment algorithm, we use 445 models, downloaded from [15] and [16], for initial testing. The alignment algorithm should work well at least using the same models. So those models are randomly rotated, translated and scaled by another program, and then using our 3D alignment algorithm to test. Fig. 7 ~ Fig. 10 show the results, and most of them work well. Each model has six pictures. Picture 1 is the original model, and picture 2 is the destination model, which randomly translate, scale and rotate from original model. Picture 3 is the result of rotating the destination model to align the original model. To clearly look the relation of the two models, picture 4 put original and destination model together. So we can see the difference of translation, rotation and scaling between two models. Picture 5 and 6 are the coarsely and refined alignment results between two models, respectively. Fig. 10 demonstrates our algorithm can also work well for many separated models.

In the 445 models, there are 5274.4 vertices and 10233.8 triangles in average. The average execution time for coarser and refined alignments are 25.7 and 39.2 seconds, respectively, in a PC with Pentium III 800MHz CPU, 128MByte RAM and WinFast S680 VGA (S3 ViRGE GX2 chip).

Next, we also test our algorithm by using different models. Those models are also randomly rotated, translated and scaled by another program first, and then using our 3D alignment algorithm to test. Fig. 11 ~ Fig. 14 show the experiment results. All experiment results are available in the web pages: <http://3dsite.dhs.org/~dynamic/3dAlign.html>.

5. 3D Model Retrieval

We apply the technique of 3D model alignment to perform 3D model retrieval. In order to reduce the retrieval time, we move 3D model alignment up to coarser rotation alignment stage. That is, the search key of 3D models is the ART coefficients from 2D shapes of each camera set. We take the minimum error between two models as the similar measurement. All models are

randomly rotated, translated and scaled by another program first, and then using our 3D model retrieval to test. The retrieval time is about 11 seconds in a PC with Pentium III 800MHz CPU. Fig. 15 shows several experimental results of 3D model retrieval. The demo can be found in <http://3dsite.dhs.org/~dynamic/cgi-bin/art/list.php>.

6. Conclusion and Future Works

The paper presents an algorithm of 3D model alignment based on a set of 2D shapes, which are projected from a 3D position, and then applies the technique to 3D model retrieval. The goal of 3D model retrieval is to recover coarser affine transformations first, and is robust against re-meshing, simplification, sub-division, and noise. In the future, other 2D shape matching algorithms can be applied to improve the 3D model alignment algorithm. In addition, other attributes, such as color and texture, can be introduced for 3D model retrieval.

References

- [1] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura and Toshiyasu L. Kunii, "Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes", *Proceedings of ACM SIGGRAPH*, pp. 203-212, Los Angeles, USA, Aug. 2001.
- [2] Cha Zhang and Tsuhan Chen, "Efficient Feature Extraction for 2D/3D Objects in Mesh Representation", *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Thessaloniki, pp. 935-938, Greece, Oct. 2001.
- [3] Cha Zhang and Tsuhan Chen, "Indexing and retrieval of 3D models aided by active learning", *Proceedings of ACM International Conference on Multimedia*, pp. 615-616, Ottawa, Canada, Oct. 2001.
- [4] Ilias Kolonias, Dimitrios Tzovaras, Stratos Malassiotis and Michael G. Strintzis, "Fast Content-Based Search of VRML Models Based on Shape Descriptors", *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pp. 133-136, Thessaloniki, Thessaloniki, Greece, Oct. 2001.
- [5] Robert Osada, Thomas Funkhouser, Bernard Chazelle and David Dobkin "Matching 3D Models with Shape Distributions", *Shape Modeling International*, pp. 154-166, Genova, Italy, May 2001.
- [6] Robert Osada, Thomas Funkhouser, Bernard Chazelle and David Dobkin "Shape Distributions", to appear in *ACM Transactions on Graphics*, Oct. 2002.
- [7] Christopher M. Cyr and Benjamin B. Kimia, "3D Object Recognition Using Shape Similarity-Based Aspect Graph", *Proceedings of International Conference on Computer Vision (ICCV)*, pp. 254-261, 2001.
- [8] Michael Elad, Ayellet Tal and Sigal Ar, "Content Based Retrieval of VRML Objects – A Iterative and Interactive Approach", *Proceedings of 6th*

- Eurographics Workshop on Multimedia*, Manchester UK, Sept. 2001
- [9] Eric Paquet and Marc Rioux, "Content-Based Access of VRML Libraries", *Lecture Notes in Computer Sciences*, Vol. 1464, pp. 20-32, 1998.
- [10] Eric Paquet, Marc Rioux, Anil Murching, Thumpudi Naveen and Ali Tabatabai. "Description of shape information for 2-D and 3-D objects", *Signal Processing: Image Communication*, Vol. 16, pp. 103-122, Sept. 2000.
- [11] Sylvie Jeannin, Leszek Cieplinski, Jens Rainer Ohm and Munchurl Kim, *MPEG-7 Visual part of eXperimentation Model Version 7.0*, ISO/IEC JTC1/SC29/WG11/N3521, Beijing, China, July 2000.
- [12] Takeo Igarashi, Satoshi Matsuoka and Hidehiko Tanaka, "Teddy: A Sketching Interface for 3D Freeform Design", *proceedings of ACM SIGGRAPH*, pp. 409-416, Los Angeles, USA, Aug. 1999.
- [13] Guo-Luen Perng, Wei-Teh Wang and Ming Ouhyoung, "A Real-time 3D Virtual Sculpting Tool Based on Marching Cubes", *Proceedings of International Conference on Artificial Reality and Tele-existence (ICAT)*, pp. 64-72, Tokyo, Japan, Dec. 2001.
- [14] Szymon Rusinkiewicz, Olaf Hall-Holt and Marc Levoy, "Real-Time 3D Model Acquisition", *proceedings of ACM SIGGRAPH*, pp. 438-446, San Antonio, USA, July 2002.
- [15] <http://www.3dcafe.com>
- [16] <http://deep.sitenest.net>
- [17] <http://www.3dfamily.com/products/digibox/main.htm>

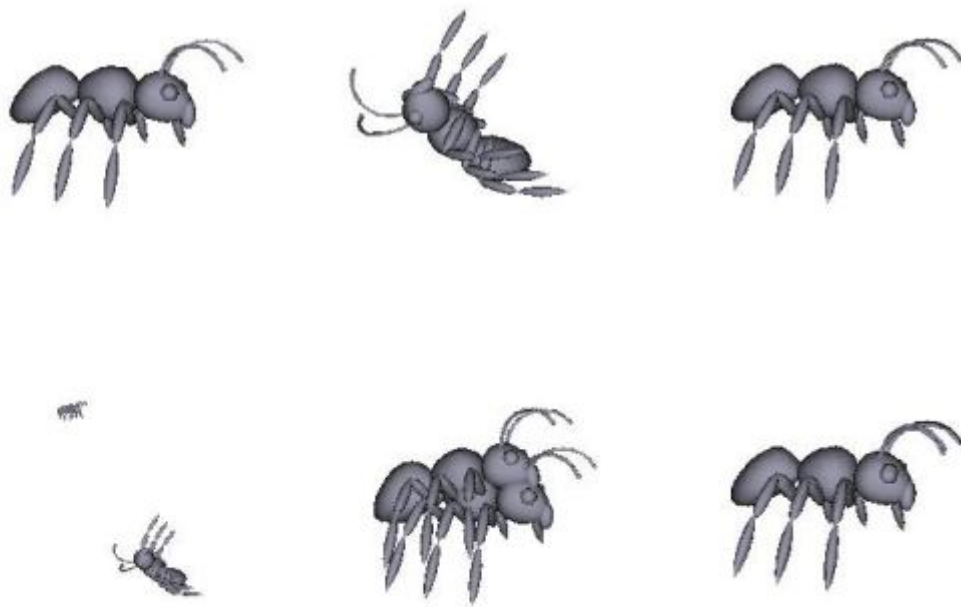


Fig. 7 Results of an experiment by using the same model "3dcafe_ant" among different affine transformations.

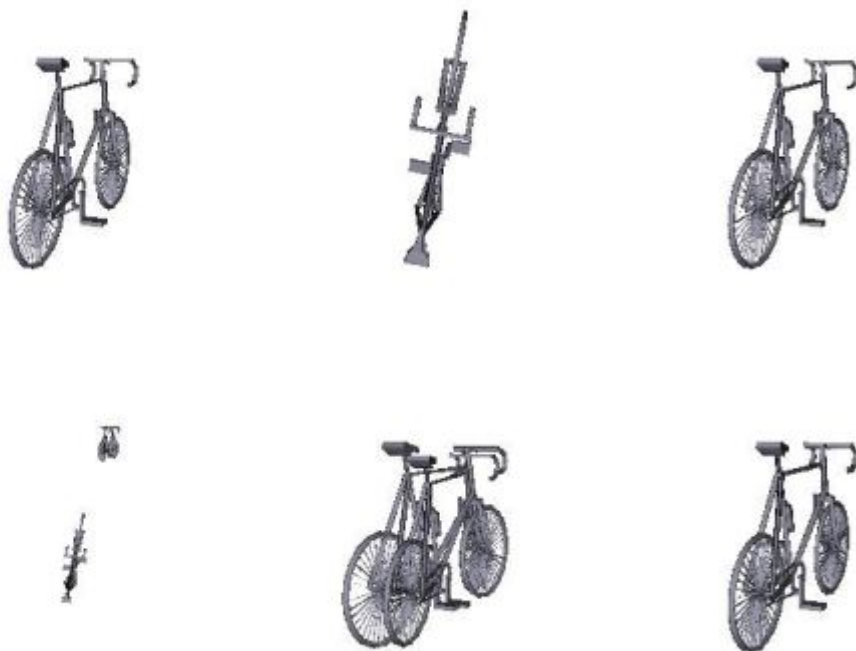


Fig. 8 Results of an experiment by using the same model "3dcafe_bicycle" among different affine transformations.

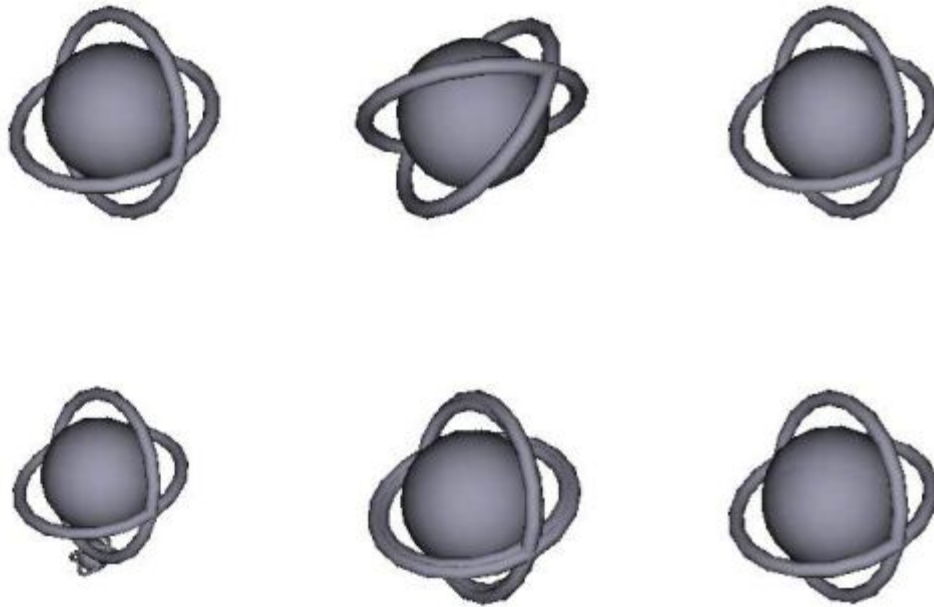


Fig. 9 Results of an experiment by using the same model “3dcafe_orbit” among different affine transformations.



Fig. 10 Results of an experiment by using the same model “3dcafe_fishbird” among different affine transformations. Our algorithm can work well for separated models.

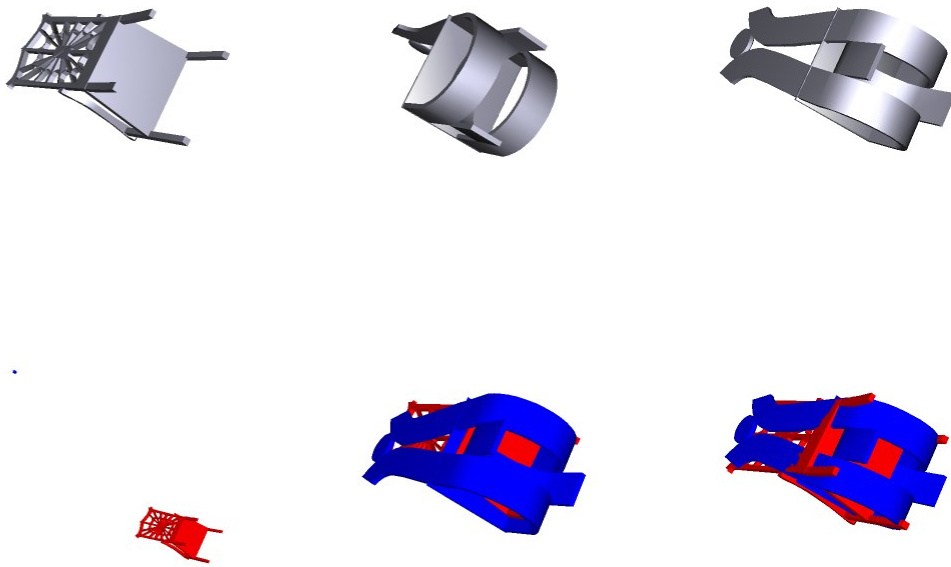


Fig. 11 Results of an experiment in aligning model “3dcafe_chair01” to model “3dcafe_chair”.

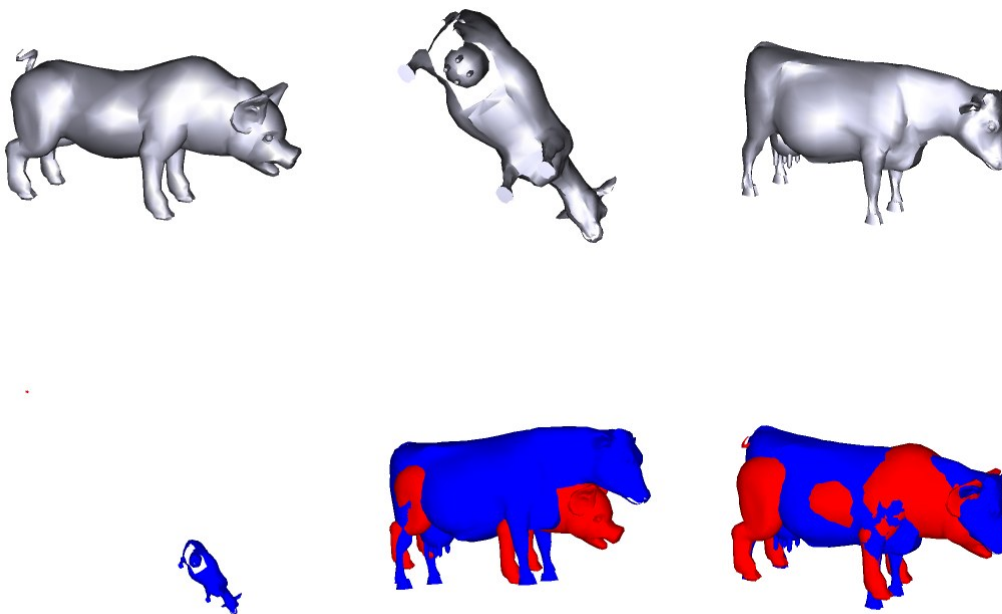


Fig. 12 Results of an experiment in aligning model “3dcafe_cow” to model “3dcafe_pig”.

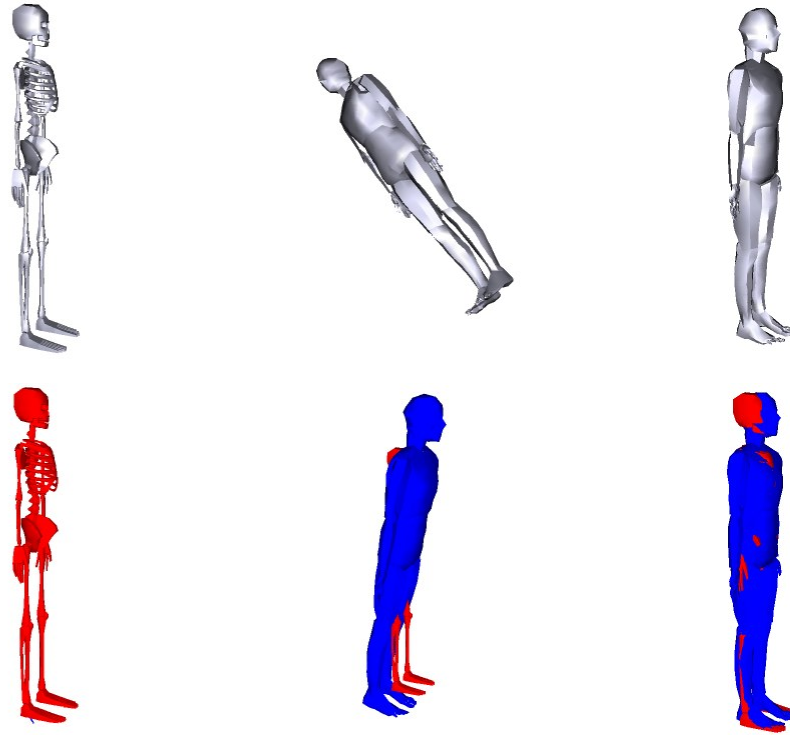


Fig. 13 Results of an experiment in aligning model “3dcafe_man1” to model “3dm-mc_slim”.

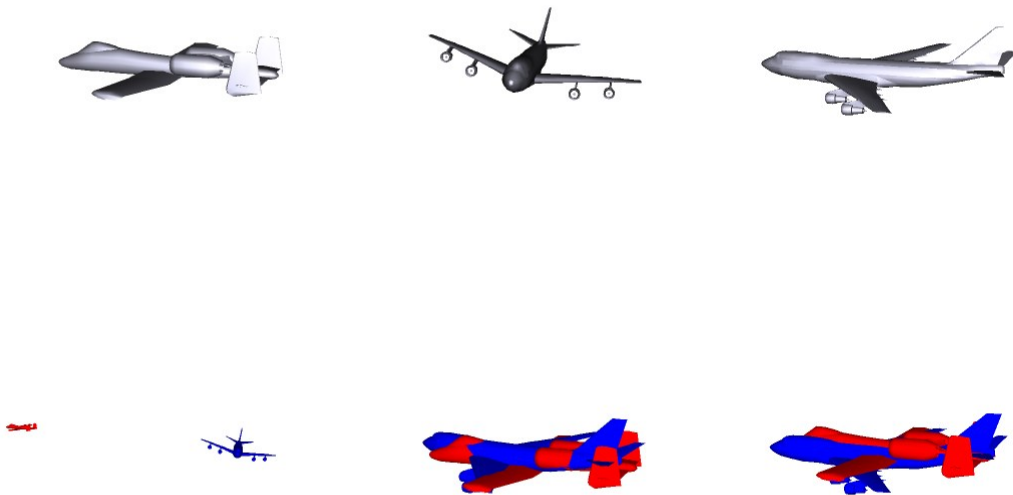


Fig. 14 Results of an experiment in aligning model “3dcafe_dc10” to model “3dcafe_a-10”.



Fig. 15 Experimental results of 3D model retrieval. The first one of each row is the target to be queried. The top 7 similar models are ranked from left to right