

# 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 angle between two models. Then, we apply the technique to match two 3D models after recovering the affine transformation. Our database has more than 10,000 3D models, and a query interface of drawing is easy to use to retrieval 3D models from 2D shapes.*

## 1. Introduction

The problem of 3D objects recognition, retrieval, clustering and classification is a traditional research topic during previous decades in computer vision, computer graphic, mechanical engineering, medical imaging and molecular biology. The research topic is more and more important since 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 [19], 3D model capturing systems [22], 3D freeform design systems [17] and sculpting systems [18]. 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]. 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 [16]. 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 affine transformation (translation, rotation and scaling), and robust against most 3D model processing, such as re-meshing, subdivision, 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 [1~14]. 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] proposed an algorithm to efficiently calculate features, such as volume, moments and Fourier transformation coefficients. They also applied active learning algorithm in 3D model retrieval using volume-surface ratio, aspect ratio, moment invariants and Fourier transformation coefficients [3, 4]. The features are normalized to be within range (-1, 1).

Osada et al. [6, 7] propose and analyze a method for computing shape signatures for arbitrary 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 best one of their experimental results

is using the D2 shape function, which measures the distance between two random points on a surface. Then, 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, the top matches are mostly from the same class as the query object and that shape distributions provide a useful signature for shape-based retrieval of 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. Their approach is simple, fast, and is invariant under affine transformation, and is not sensitive to most 3D model processing and degeneracies.

Thomas Funkhouser et al. [8] describe a web-based search engine system that supports queries based on 3D sketches, 2D sketches, 3D models, and/or text keywords. For the shape-based queries, they have developed a new matching algorithm that uses spherical harmonics to compute discriminating similarity measures without requiring repair of model degeneracies or alignment of orientations. Their matching methods are fast enough to match a single query to a repository of 20,000 models in under a second.

Hilaga et al. [1] propose a technique in which similarity between polyhedral models is 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 affine transformation and robust against most 3D model processing and deformation caused by changing posture of an articulated object. Their search key is a multi-resolution 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 average search time is about 12 seconds in a PC with a Pentium II 400MHz processor. A 3D model retrieval system [13] in our previous work is extended the work of Hilaga et al.

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

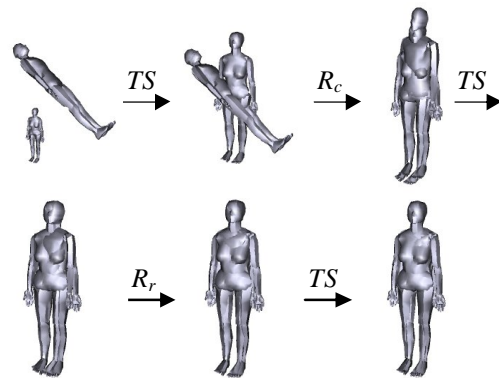


Figure 1 The order of 3D models alignment.

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 basic concept of our approach is that when two 3D models are similar, they will also look similar from their corresponding viewing angles. Therefore, the main idea of our 3D alignment algorithm in rotation is to render 2D silhouettes from each viewing angle of two models, and get rotation which has minimum error summing from all viewing angles using 2D shape matching algorithm. Our approach of 3D model retrieval takes the minimum error as the similarity between two 3D models. The remainder 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 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 in our approach is as follows:  $TS \rightarrow R_c \rightarrow TS \rightarrow R_r \rightarrow TS$ . Where  $TS$  denotes the translation and scaling alignment from one 3D model to another;  $R_c$  and  $R_r$  denote the coarser and refined rotation alignment, respectively (detailed in next chapter). 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 the two models be roughly in similar position and close to 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. Figure 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

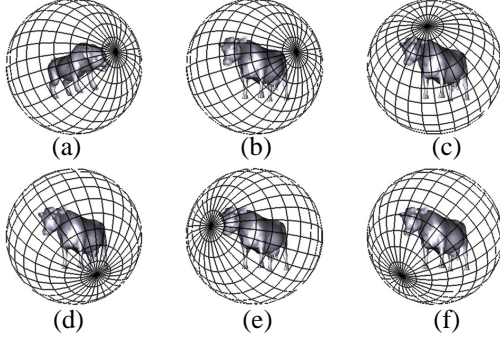


Figure 2 A typical example to show our algorithm.

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  $MaxCoor_i$  and  $MinCoor_i$  are the coordinate of vertex, which has maximum and minimum value in  $i$ -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 Figure 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. Figure 2 (a) shows a set of cameras surrounding the model *pig*, where each intersection point indicates a camera position. Then, apply the camera set to the model *cow*, as shown in Figure 2 (b), and then get the difference between two 2D shapes for each camera pair (A camera pair is the corresponding viewing angle between the two models). Therefore, we define the error between two 3D models, when applying some rotated camera set, as summing the difference from all camera pairs. The rotation of a camera set, which has minimum error, also indicates the rotation between the two 3D models. The minimum error can also be used for judging the similarity of the two 3D models, and is defined as:

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

where  $ShapeDiff$  denotes the difference between two 2D shapes;  $i$  denotes different rotation of the

camera set, and  $j$ -th camera pair between the two models. Figure 2 (b)~(f) show several examples of different rotation of the camera set, and we can suppose that the camera set in Figure 2 (e) will get the minimum error.

When rotating the camera set to a new orientation, all 2D shapes should be re-rendered, and the similarity matching between each camera pair also has to be calculated again. 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 and matching will be greatly reduced.

However, there are only five regular convex polyhedrons, which are named as Platonic bodies, and were 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 amount of vertices, as the position of the camera set. There are 20 scattering viewing angles for each 3D model. Huber and Hubert [15] proposed an automatic registration algorithm, which is able to reconstruct real world objects from 15 to 20 various viewpoints of laser scanner. Therefore, the 20 viewing angles can roughly represent the shape of a 3D model. The 20 2D shapes are the bases of each 3D model for alignment, and contain enough information from various viewing angles for a 3D model. Figure 3 show that we can reduce the number of rendering and matching by using the dodecahedron. Figure 3 (a) shows a camera set for model *pig*, and then the same camera set is applied to model *cow*, as shown in Figure 3 (b). That is, the indices of camera set are all the same between Figure 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 Figure 3 (b) to edge (1,3) and (1,4), which show in Figure 3 (c) and (d), respectively. Since a dodecahedron has 20 vertices and each 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

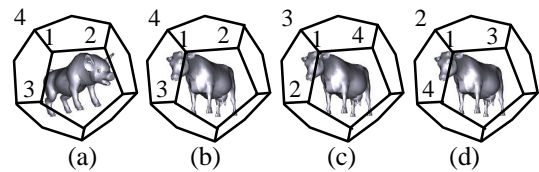







Figure 3 We can reduce the number of calculation from rendering and 2D shapes matching by using the dodecahedron.

Table 1 Number of rendering and 2D shapes matching for 60 different rotations with and without using 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 2 Number of rendering and matching 2D shapes are calculated by mapping m to n different dodecahedrons.

(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					

matching for 60 different rotations with and without using dodecahedron. Consequently, amount of rendering and matching can be reduced.

There are 60 different rotations for searching by using one camera set of both models. However, it's usually not enough to recover rotation from the best one 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 for each 3D model. When adding one camera set, 60 different rotations will be searched. Furthermore, we can also apply more camera sets to both 3D models. That is, when applying  $m$  and  $n$  camera set to both models, respectively, there will be  $60 \times m \times n$  kinds of different rotation for searching. Table 2 shows the number of rendering and 2D shape matching by using different number of camera sets. In our implementation, we set  $m=10$  and  $n=10$  (10-10), 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. In addition, those camera sets are pre-processed for scattering over the whole rotation space, so that any possible rotation can close to one of them.

In the end of this chapter, we detail the flow of aligning one model to another. The operations of aligning model  $B$  to model  $A$  is shown as follows:

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

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

$$R_c = \text{RotateCoarse}(B', A')$$

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

$$R_r = \text{RotateRefine}(B'', A')$$

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

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

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

$$S_A^{-1} \cdot T_A^{-1}$$

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 of aligning rotation between two models. The rotation alignment has two levels: coarser ( $R_c$ ) and refined ( $R_r$ ) alignment. The coarser alignment gets the approximate rotation from all possible orientation between two models. The refined alignment is to adjust the rotation from the above result to more accurate one. After doing first *TS* (described in previous chapter), the position and size of both models are approximate, but not exactly the same, so our approach of 2D shape matching should be invariant to translation and scaling.

The rotation between two models is aligned by matching 2D shapes from camera sets of both models. Figure 4 shows the flow of rotation alignment. First, 2D shapes should be rendered from camera sets for both models. For each 2D shape, its feature can be extracted and saved for matching later. The operation of rendering and feature extraction do 40 times, if using 1-1 camera set. Then, 2D shapes of each camera pair have to be matched, and the operation does 400 times if using 1-1 camera set. Next, get minimum error from different rotations, as defined in Eqn. (3). Finally, once two camera sets of both models are determined, the rotation matrix of two models can be obtained by the rotation of the two camera sets. The main flows of coarser and refined rotation alignment are the

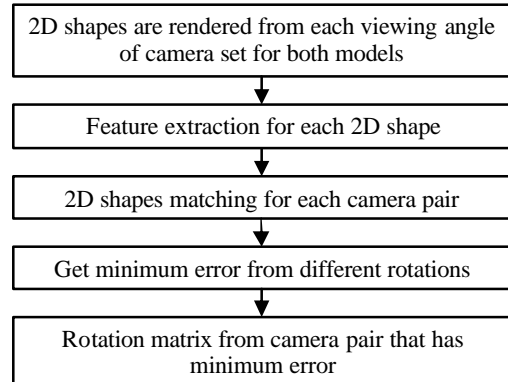


Figure 4 The flow of rotation alignment



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

same.

The character of 2D shape matching depends on which algorithm is used. We use OpenGL to render 2D silhouettes by putting camera to vertex of dodecahedron and facing to origin. The size of 2D silhouettes is 256 by 256 pixels in our implementation. Since 3D models should be translated and scaled, *TS*, before rotation alignment, it's easier to make sure that whole 3D models will be rendered into a 2D silhouette, that is, no clipping happen. Figure 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 get 2D silhouettes.

To measure the similarity between two shapes, we use region-based shape descriptor of the MPEG-7 [16] 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 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.

After feature extraction for each 2D shape, shape matching for each shape from two models is calculated. Amount of feature extraction is the same as rendering, but the amount 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 the feature can be previously calculated and saved to database. The region-based shape matching algorithm use simple L1 distance to measure similarity:

$$ShapeDiff((A, B)) = \sum_i |ArtM_A[i] - ArtM_B[i]| \quad (4)$$

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 differences from all corresponding 2D shapes pair. However, there is a little difference in this stage between coarser and refined rotation alignment. In coarser rotation alignment, we use 10-10 camera sets, that is, searching the best one from 6000 different rotations. 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 camera set to the direction that has less error, until no improvement. Therefore, we can align the rotation with error less than 1°.

Finally, rotation matrix between dodecahedron pair that has minimum error, should be calculated. The rotation matrix (*R<sub>c</sub>* or *R<sub>r</sub>*) is then applied to one model 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 Figure 6 to explain our approach. Suppose that Figure 6 (a) and (c) are the dodecahedron pair of model A and B, respectively. The rotation matrix aligns edge (1,2) in Figure 6 (c) to that in (a). The vector  $\vec{o_1}$  and  $\vec{o_2}$  can form a unique coordinate frame, defined as follows:

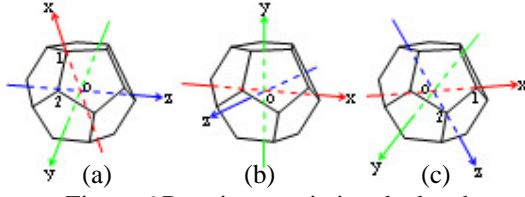


Figure 6 Rotation matrix is calculated between two camera sets.

$$\begin{cases} \bar{x} = \bar{o}1 \\ \bar{y} = \bar{z} \times \bar{x} \\ \bar{z} = \bar{o}1 \times \bar{o}2 \end{cases} \quad (5)$$

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  $F_{CB} \cdot F_{AC}$ .

#### 4. Experimental Results of 3D Alignment

In order to experiment with the 3D alignment algorithm, we use 445 models, downloaded from [20] and [21], 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. Figure 7 shows the results, and almost all of them work well. Each model has five pictures. Picture 1 is the original model, and picture 2 is the destination model, which randomly translate, scale and rotate from original model. To clearly look the relation of the two models, picture 3 put original and destination model together. So we can see the difference of translation, rotation and scaling between two models. Picture 4 and 5 are the results of coarser and refined alignment between two models, respectively.

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 14.5 and 23.5 seconds, respectively, in a PC with Pentium III 800MHz

CPU.

In addition, 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. Figure 8 shows the experiment results. All experiment results are available in the web pages: <http://www.cmlab.csie.ntu.edu.tw/~dynamic/3DRetrieval/index.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 (defined in Eqn. (3)). 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 4 seconds in a PC with Pentium III 800MHz CPU. Figure 9 shows several experimental results of 3D model retrieval. In addition, we also test a database which has more than 10,000 3D models, and the results are shown in Figure 10 and 11. Furthermore, a query interface can be used to retrieval 3D models from 2D shapes, as shown in Figure 12. The retrieval from user drawing is very fast, about 1~2 seconds for searching from more than 10,000 3D models. The demo can also be found in <http://www.cmlab.csie.ntu.edu.tw/~dynamic/3DRetrieval/index.html>.

#### 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 affine transformation and most 3D model processing. 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.

#### Acknowledgement

We would like to thank Jeng-Sheng Yeh for providing the web server in our experiments. This project is partially funded by the National Science Council (NSC, Taiwan) under the grant number NSC90-2622-E-002-008.



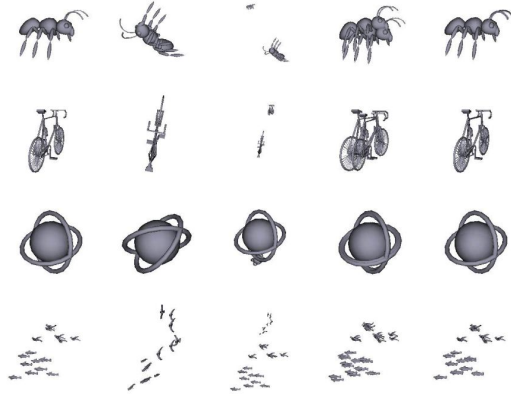


Figure 7 Results of experiments by using the same model among different affine transformations.

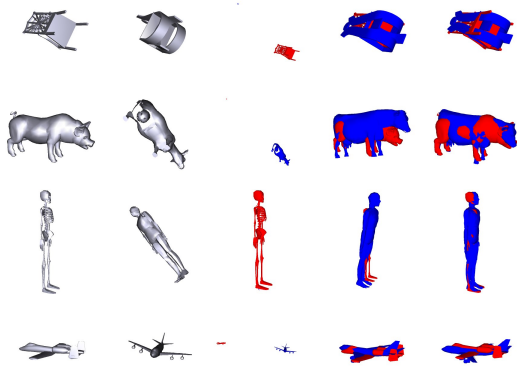


Figure 8 Results of experiments in aligning model before and after randomly rotating, translating and scaling.

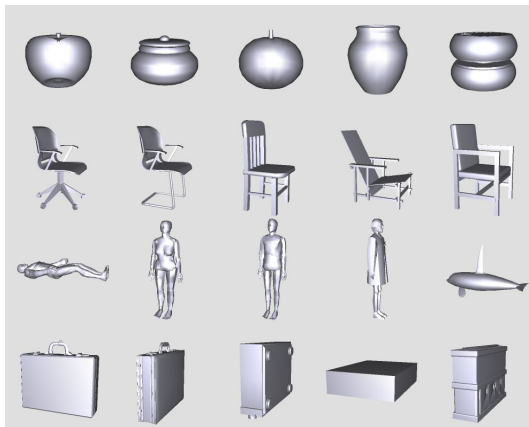


Figure 9 Experimental results of 3D model retrieval. The first one of each row is the target to be queried. The top 5 similar models are ranked from left to right

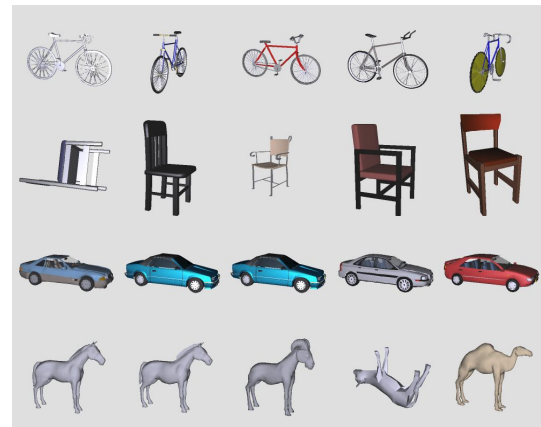


Figure 10 Experimental results of 3D model retrieval using a database which has more than 10,000 3D models. The first one of each row is the target to be queried. The top 5 similar models are ranked from left to right

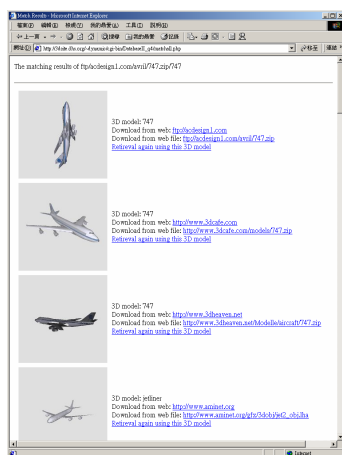


Figure 11 A retrieval result using a database which has more than 10,000 3D models. Top one is the input 3D models, and the similar models are ranked from top to down.

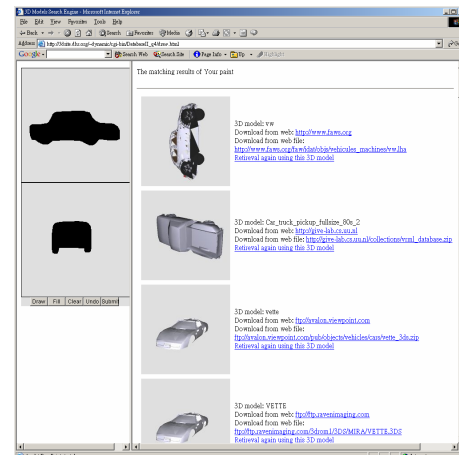


Figure 12 A query interface of user drawing can be used to retrieval 3D models from 2D shapes.

## References

- [1] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura and Tosiya 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] Cha Zhang and Tsuhan Chen, "An Active Learning Framework for Content-Based Information Retrieval", *IEEE Transactions on Multimedia*, Vol. 4, No. 2, June 2002.
- [5] 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.
- [6] 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.
- [7] Robert Osada, Thomas Funkhouser, Bernard Chazelle and David Dobkin "Shape Distributions", *ACM Transactions on Graphics*, Vol. 21, No. 4, pp. 807-832, Oct. 2002.
- [8] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs, "A Search Engine for 3D Models" to appear in *ACM Transactions on Graphics*, Jan. 2003.
- [9] 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.
- [10] 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
- [11] Eric Paquet and Marc Rioux, "Content-Based Access of VRML Libraries", *Lecture Notes in Computer Sciences*, Vol. 1464, pp. 20-32, 1998.
- [12] 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.
- [13] Ding-Yun Chen and Ming Ouhyoung, "A 3D Object Retrieval System Based on Multi-Resolution Reeb Graph", *Proceedings of Computer Graphics Workshop*, pp. 16, Tainan, Taiwan, June 2002.
- [14] Ryutarou Ohbuchi, Tomo Otagiri, Masatoshi Ibato and Tsuyoshi Takei, "Shape-Similarity Search of Three-Dimensional Models Using Parameterized Statistics", *Proceedings of 10th Pacific Graphics*, pp. 265-273, Beijing China, Oct. 2002.
- [15] Daniel F. Huber and Martial Hebert, "Fully Automatic Registration of Multiple 3D Data Sets", *Proceedings of IEEE Workshop on Computer Vision Beyond the Visible Spectrum: Methods and Applications (CVBVS)*, Kauai, Hawaii, USA, Dec. 2001.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] <http://www.3dcafe.com>
- [21] <http://deep.sitenest.net>
- [22] <http://www.3dfamily.com/products/digibox/main.htm>