On 3D Shape Similarity

Heung-yeung Shum   Martial Hebert Katsushi Ikeuchi

November 1995

CMU-CS-95-212

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This research is partially sponsored by the Advanced Research Projects Agency under the Department of the Army, Army Research Office under grant number DAAH04-94-G-0006, partially supported by ONR under grant number N00014-95-1-0591, and partially supported by NSF under Contract IRI-9224521. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the Department of the Army, or the U.S. government.

Keywords: Object recognition, object representation, 3D shape similarity, shape metric, curvature.

Abstract

We study the ***3D shape similarity*** between closed surfaces. We represent a curved or polyhedral 3D object of genus zero using a mesh representation that has nearly uniform distribution with known connectivity among mesh nodes. We define a shape similarity metric based on the L_2 distance between the local curvature distributions over the mesh representations of the two objects. For both convex and concave objects, the shape metric can be computed in time $O(n^2)$, where n is the number of tessellation of sphere or the number of meshes which approximate the surface. Experiments show that our method produces good shape similarity measurements.

Table of Content

List of Figures

1 Introduction

Being able to compare object shapes is essential for many computer vision tasks such as object model categorization and hypothesis verification in model-based object recognition [6]. Previous work has focused on comparing 2D scene images with 2D object models. For example, as shown in Figure 1, a gradual shape change of a 2D closed curve, from a square to a concaved triangle, can be captured by previous shape similarity measures (e.g., [1] [19]).

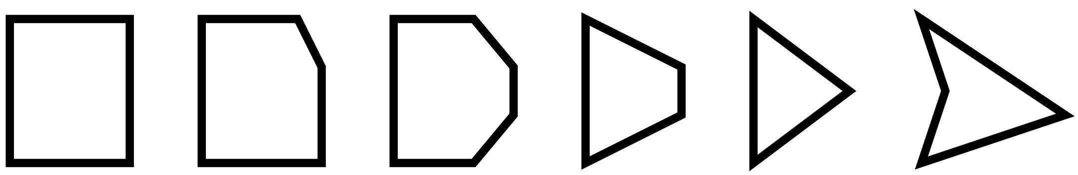


Figure 1 **An example of 2D shape similarity: how to measure the gradual shape change from left to right?**

Recent progress in 3D sensors such as laser range finders and real-time stereo machines has led us to the problem of comparing 3D objects with 3D or 2D scene image. In this paper, we address the following question: to what extent is 3D shape A similar (or dissimilar) to 3D shape B?

The desirable properties of such a shape similarity measure are as follows. First, such a measure between two geometrical shapes should be a metric. In particular, the triangle inequality is necessary since it is desirable in pattern matching and object recognition applications. In addition, the distance function between two shapes should be invariant under rigid transformation and scaling, easy to compute, and intuitive with human shape perception [1].

The problem of shape similarity has been studied extensively in both machine vision and biological vision. Readers interested in human perception of similarity, such as contextual and asymmetrical properties, are referred to Tversky [23] and Mumford [16]. For human perception, other features such as color or functional information are also used to compare objects. In this paper, we focus on geometrical shape similarity because geometry is the basis for other features such as color and reflectance, etc. As a first step toward 3D shape similarity with arbitrary topology, we restrict ourselves only to objects of genus zero (objects without holes). We want to compare polyhedral shapes as well as smooth surfaces.

To compare different shapes, one must first understand how to represent them. Most work assumed an object shape to be a two dimensional closed contour. Many previous methods can evaluate the shape similarity among the set of 2D closed polygons shown in Figure 1. For instance, Schwartz and Sharir [19] proposed to approximate a closed 2D curve, after proper smoothing if necessary, by a simple polygon with equal-length edge segments. The polygon was then represented by the turning angle (a measure of local curvature) at each vertex. Arkin et al. [1] also represented local curvature at each vertex of a polygon using a turning angle and, in addition, proposed an efficient algorithm to directly compare polygons. Mumford [16] also suggested the use of moments as an alternative to curvature because moments are also invariant to rigid transformation and scaling. Other 2D shape similarity methods include 2D planar graph and graph matching by Kupeev and Wolfson [14], and shape deformation by Basri et. al. [3].

Unlike using a closed 2D curve which can be simply parameterized by its arc-length, however, it is much more difficult to find an appropriate "data structure" in which to store a 3D surface. How to compute and store the curvature information on the surface depends on the choice of coordinate system. Without a proper representation, it is unclear how to compare polyhedral shapes because curvature is zero everywhere except on vertices and edges. It is no trivial task to compare simple 3D shapes as shown in Figure 2. In practice, local curvature on each sample point of surface is difficult to estimate robustly from noisy range data. This problem is even more severe when only a single view depth map is available because of surface discontinuity and occlusion.

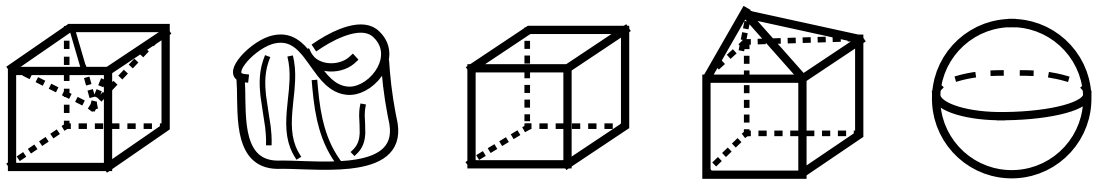


Figure 2 **An example of 3D shape similarity: how similar are these shapes?**

Because a closed surface is topologically equivalent to a sphere, many spherical representations have been proposed to represent closed surfaces. The Gauss map characterizes the surface normal at each point on a unit sphere, called a Gaussian sphere. Horn [9] proposed to represent objects using an extended Gaussian image (EGI) which uses a distribution of mass over the Gaussian sphere. Little [15] showed that an EGI could be used for pose determination. A Complex EGI was proposed by Kang and Ikeuchi [12] to store both surface area and distance information which can be very useful for recovering translation. It has been proven that two convex objects are congruent if they have the same EGIs. Nalwa [18] augmented Gaussian images by some support function which was the signed distance of the oriented tangent plane from a predefined origin. Hebert, Ikeuchi and Delingette [8] proposed a simplex attribute image (SAI) to characterize the convex/concave surfaces, both as a coordinate system and as a representation. For a summary of different spherical representations, the reader is referred to [11] by Ikeuchi and Hebert. Brechbuhler, Gerig and Kubler [5] also defined a one-to-one mapping from a simply-connected surface to a unit sphere, using extended 3D elliptical Fourier descriptors.

The lack of a proper coordinate system (or data structure) for geometrical entities has driven many researchers to compare 3D shapes in domains other than geometrical space. For example, Sclaroff and Pentland [20] used many modes to represent shapes and to compare shapes based on the coefficients of the modes. Murase and Nayar [17] represented objects in eigenspace, and compared objects depending on the proximity of two eigenvalues to one another. Unfortunately, these quantities used for measuring similarity do not provide us with geometrical intuition.

Even with the appropriate data structure, choosing a good metric for comparing shapes can be confusing. For example, Arkin et. al. [1] used L_2 norm to compare polygons. Huttenlocher and Kedem [10] used Hausdorff distance to compare the distance between two point sets under translation. Kupeev and Wolfson [14] used graph matching to compare 2D shapes. Basri et. al. [3] emphasized that the distance function has to be continuous and should matter less as curvature becomes greater. Comparison among different metrics can be found in [16].

Using a special spherical coordinate system we represent a closed curved or polyhedral 3D surface without holes. A semi-regularly tessellated sphere is deformed so that the meshes sit on the original data points while the connectivity among the mesh nodes is preserved. After the deformation process, we obtain a spherical representation with local curvature at each mesh node. The problem of comparing two shapes becomes that of comparing the corresponding curvature distributions on spherical coordinates. This approach is illustrated in Figure 3. The local curvature at each node is calculated by its relative position to its neighbors. We then present an efficient shape metric between two objects: the metric is a distance function between two corresponding curvature distributions on spherical coordinates.

The paper is organized as follows. In Section 2 we introduce a spherical representation of a 3D surface. Then we define local curvature and show how to compute it. In Section 3 we present a distance metric between two objects, or between two spherical approximations of these two objects obtained from surface deformation. We also construct two algorithms to compute the metric. We show experimental results in Section 4 and give final comments in Section 5.

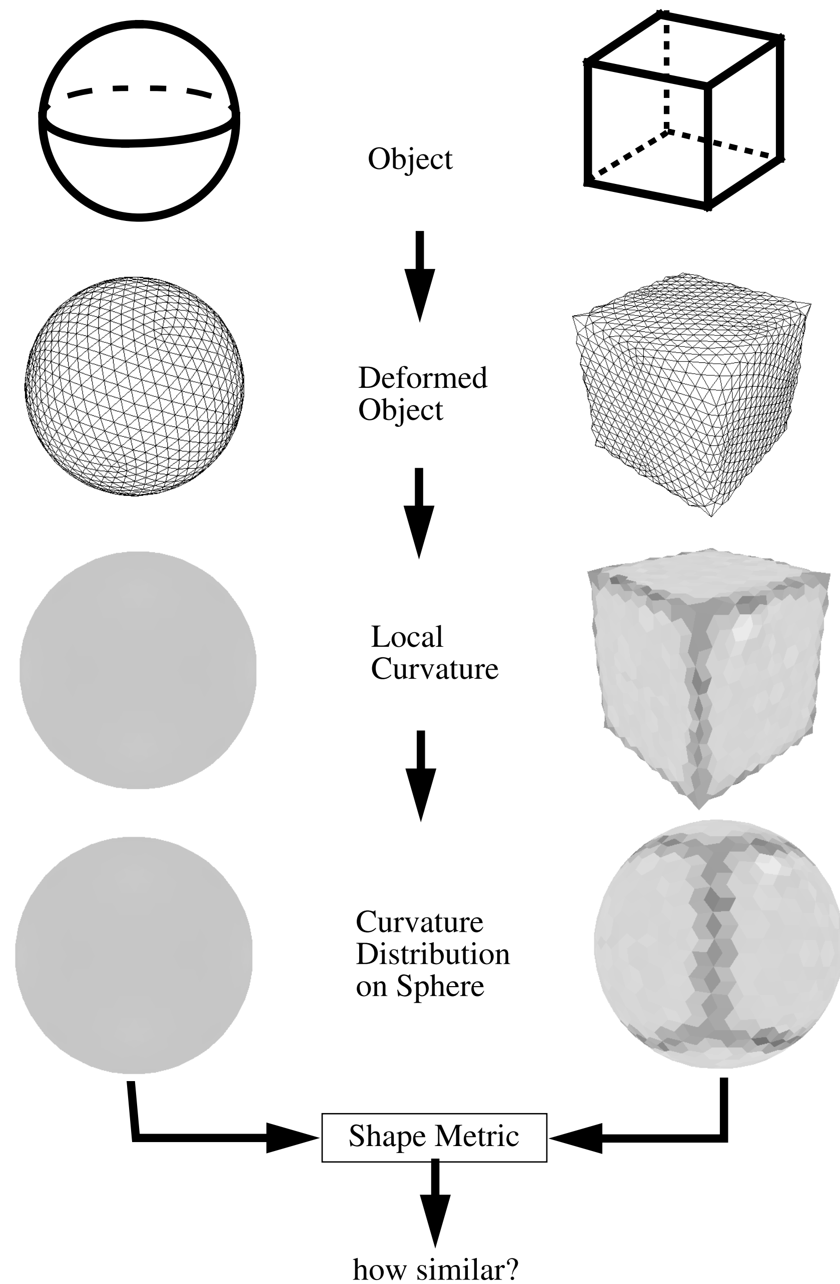


Figure 3 **Comparing shapes from curvature distribution: an example of a sphere and a hexahedron. The curvature has been color-coded so that the darker the bigger positive curvature and the lighter the bigger negative curvature.**

2 Representation of a Closed Surface

2.1 Discrete Representation of a Curve

To compare object shapes, one first has to find appropriate representations of those shapes [2]. A standard way of representing a simple polygon is to describe its boundary by a circular list of vertices with known coordinates. To represent a simple closed 2D curve (not self intersecting), one can parameterize the curve by a number of points. For example, one can approximate the curve by equal length line segments. The similarity between two curves can be measured by comparing the distribution of curvature measurement at the vertices of the approximating polygons.

The curvature of a discrete curve at each node of the polygonal approximation can be approximated by the turning angle between adjacent line segments. The turning angle can be viewed as a discrete average measure of local curvature at the vertex. Like curvature, the turning angle is independent of rigid transformation and scaling. To avoid possible unstable representation under certain kinds of noise, dense equal length line segments have been adopted in [19] and [8]. For noise-free polygons with few vertices, Arkin et. al. [1] showed a very efficient algorithm which directly compares turning angles on vertices. Unfortunately Arkin's approach can not be extended to 3D polyhedra because of the lack of a proper coordinate system.

2.2 Spherical Representation of a 3D Surface

A natural discrete representation of a surface is a graph of nodes, or tessellation, such that each node is connected to each of its closest neighbors by an arc of the graph. We use a special mesh each node of which has exactly three neighbors. Such a mesh can be constructed as the dual of a triangulation of the surface [7]. To tessellate a unit sphere, we use a standard semi-regular triangulation of the unit sphere constructed by subdividing each triangular face of a 20-face icosahedron into N^2 smaller triangles. The final tessellation is built by taking the dual of the 20 N^2-face triangulation, yielding a tessellation with the same number of nodes.

In order to obtain a mesh representation for an arbitrary surface, we use a deformable surface algorithm in which we deform a tessellated surface until it is as close as possible to the object surface. This algorithm drives the spherical mesh to converge to the correct object shape by combining forces between the data set and the mesh. Our algorithm originates from the idea of a 2D deformable surface [22] and is described in detail in [8]. The deformed surface can accurately represent concave as well as convex surfaces. Our deformable algorithm is not sensitive to deformation parameters such as initial center and radius of the sphere. An example of a free-form object model created using the deformable surface and multiple view merging techniques [21] is show in Figure 4. The deformation process is robust against data noise and moderate change of parameters such as initial sphere center and radius [21].

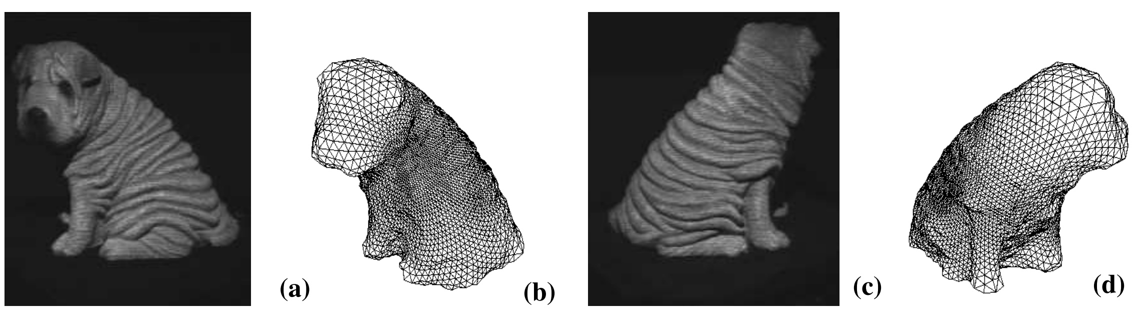


Figure 4 **An example of free-form object modeled from deformable surface: (a) (c) Images of a sharpei; (b) (d) Deformable models of a shapei**

The key idea of our spherical representation of surface is to produce meshes in which the density of nodes on the object's surface is nearly uniform[1]. Although perfectly uniform dis-

1. Koenderink warned that one has to be very careful of any method that uses surface area of a polyhedral model (p.597 of [13]). Surface area depends on the way in which triangulations are done. In our previous work, we have shown how areas of different shapes are adjusted before comparison, in particular for partial views[8].

tribution is impossible, a simple local regularity constraint can enforce a very high degree of uniformity across the mesh. First of all, we start with a semi-regularly tessellated sphere. Then we implement the local regularity constraint in the deformable surface algorithm such that each mesh is similar to the others in area [8].

This local regularity constraint is a generalization to three dimensions of the regularity condition on two dimensional discrete curves; this condition simply states that all segments are of equal lengths. The difference between 2D and 3D cases is that it is always possible to create a uniform discrete curve in 2-D, while only nearly uniform discrete surfaces can be generated in 3-D. In practice, the variation of mesh nodes on the surface is on the order of 2% [8].

2.3 3D Local Curvature: An Approximation

After we obtain a nearly uniform surface mesh representation, the next step is to define a measure of curvature that can be computed from the surface representation. Conventional ways of estimating surface curvature, either by locally fitting a surface or by estimating first and second derivatives [4], are very sensitive to noise. This sensitivity is mainly due to the discrete sampling and, possibly, to the noisy data. We introduced in [8] a robust measure of curvature computed at every node from the relative positions of its three neighbors. Our method is robust because all the nodes are at relatively stable position after the deformation process. The deformable surface process serves as a smoothing operation over the possibly noisy original data. We called this measure of curvature the simplex angle.

The simplex angle varies between $-\pi$ and π, and is negative if the surface is locally concave, positive if it is convex[1]. Given a configuration of four points, the angle is invariant by rotation, translation, and scaling because it depends only on the relative positions of the points, not on their absolute positions.

1. It is interesting to note that the simplex angle is related to mean curvature at the vertex [Delingette].

The spherical representation can approximate not only free-form objects, but also polyhedral objects. For example, Figure 5 shows an example of a spherical polyhedral approximation of an octahedron with one concave face. Because of the regularity constraint, corners and edges are not represented perfectly. All plane surfaces, however, are well approximated even though the local regularity is enforced on all meshes.

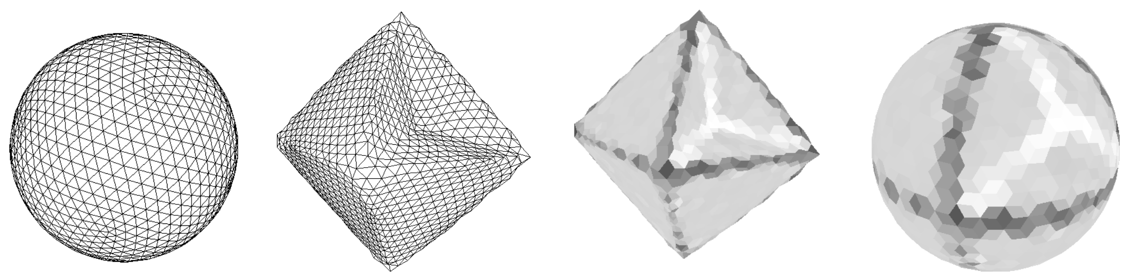


Figure 5 **(a) A spherical tessellation; (b) Deformable surface of an octahedron with a concave dent; (c) Local curvature on each mesh node; (d) Curvature distribution on spherical representation (The curvature on (c) and (d) is negative if it is light, positive if dark, zero if grey).**

3 The 3D Shape Similarity Metric

In section 2 we have explained how we obtain mesh representation and curvature distribution of a 3D surface over the sphere. Let S_A and S_B be the mesh representations of shape *A* and shape *B*, and $k_R(S_A)$ and $k_R(S_B)$ be the curvature distribution functions under a spherical rotation *R*. We then formally define the distance function between two 3D surfaces *A* and *B* as the L_p distance between their local curvature functions $k_I(S_A)$ and $k_R(S_B)$, minimized with respect to the rotation matrix *R* over the sphere. The function $k_I(S_A)$ denotes the curvature distribution of S_A under no rotation where *I* is the identity matrix. Hausdorff distance [10] can be an alternative to L_p distance, but the computation is formidable.

3.1 A Distance Function on Sphere

We define the L_p distance $d_p(S_A, S_B, R)$ between *A* and *B* at a certain spherical rotation *R* as

$$d_p(S_A, S_B, R) = \left(\int\left|k_I(S_A) - k_R(S_B)\right|^p dS\right)^{\frac{1}{p}}$$

which is the sum of curvature differences over the sphere. Then the distance function between *A* and *B*, $D_p(A, B)$ becomes

$$D_p(A, B) = min_R \quad d_p(S_A, S_B, R)$$

which is minimized d_p over all possible rotations *R*.

Property 1: $D_p(A, B)$ **is a metric for all** $p>0$**.**

Proof:

Because d_p is a L_p norm, we have

- D_P is positive. $D_p(A, B) \geq 0$;
- D_P is identity. $D_p(A, A) = 0$;
- D_P is symmetric. $D_p(A, B) = D_p(B, A)$.

The only thing left to prove is the triangle inequality $D_p(A, B) + D_p(B, C) \geq D_p(A, C)$.

Let R_1 and R_2 be the rotation matrices which minimize the $D_p(A, B)$ and $D_p(B, C)$, respectively,

$$D_p(A, B) + D_p(B, C)$$

$$= min_{R_1}\ d_p(S_A, S_B) + min_{R_2}\ d_p(S_B, S_C)$$

$$= min_{R_1}\ \left(\int\left|k_I(S_A) - k_{R_1}(S_B)\right|^p dS\right)^{\frac{1}{p}} + min_{R_2}\ \left(\int\left|k_I(S_B) - k_{R_2}(S_C)\right|^p dS\right)^{\frac{1}{p}}$$

$$\geq min_{R_1, R2}\left(\left(\int\left|k_I(S_A) - k_{R_1}(S_B)\right|^p dS\right)^{\frac{1}{p}} + \left(\int\left|k_{R_1}(S_B) - k_{R_1^{-1}R_2}(S_C)\right|^p dS\right)^{\frac{1}{p}}\right)$$

$$= min_{R_1, R_2}\ (d_p(S_A, S_B, R_1) + d_p(S_B, S_C, R_2))$$

$$\geq min_{R_3}\ d_p(S_A, S_C, R_3)$$

$$= D_p(S_A, S_C)$$

where $R_3 = R_1^{-1}R_2$.

3.2 Algorithm One: A Global Search

The above proof showed that we can search over the spherical rotation space to compute the distance between two curvature distributions. A naive algorithm can then be easily constructed. Because this is an exhaustive search, global minimum is always found provided that the search step is small enough (this is, the number of searches is sufficiently large). This leads to the following property:

Property 2: The distance between two shapes A **and** B, $D_2(A, B)$**, can be computed in time** $O(m^3)$ **where** m **is the number of searches in each rotational space.**

3.3 Algorithm Two: An Efficient Algorithm

The above time bound can be improved by employing a property of the semi-regularly tessellated sphere: each node has exactly three neighbors. We have observed [8] that the only rotations for which $d(S_A, S_B)$ should be evaluated are the ones that correspond to a valid list of correspondences $\{P_i, P_j\}$ between the nodes P_i of S_A and the nodes P_j' of S_B. In Fig. ??, node P_1 of the S_A corresponds P_1' of S_B, and the two neighbors of P1 (P_2, and P_3) are put in correspondence with two of three neighbors (P_2', P_3' and P_4') of P_1', respectively. Fig. ?? shows only 3 valid neighborhood matchings since each node has exactly three neighbors and the connectivity among them is always preserved. Given correspondence of three nodes, a spherical rotation can be calculated. This rotation defines a unique assignment for the other nodes. In other words, there is a unique correspondence between a node P_j' of S_B and a node P_i of S_A, given the initial correspondences between $\{P_1, P_2, P_3\}$ and $\{P_1', P_2', P_3'\}$. Moreover, the number of such correspondences is $3n$ where n is the number of nodes of spherical tessellation [8]. Figure 6 shows that there are three possible matchings at each node. Equivalently, there are $3n$ distinct valid rotations of the unit sphere. This analysis leads us to an efficient algorithm for comparing two shapes.

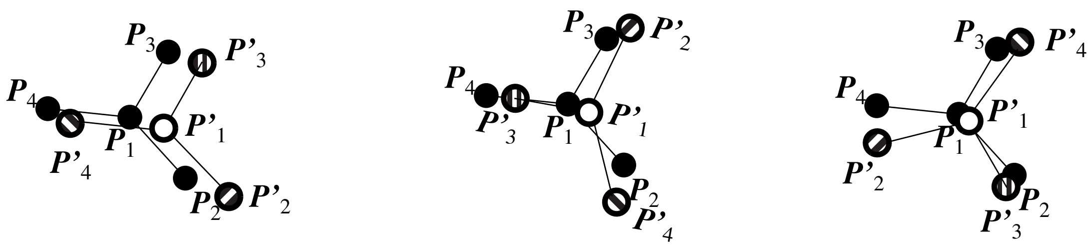


Figure 6 **Matching of neighbors from (P_2, P_3, P_4) to: (a) (P_2', P_3', P_4'); (b) (P_3', P_4', P_2'); (c) (P_4', P_2', P_3') when P_1 of shape *SA* is matched to P_1' of shape *SB*.**

Property 3: The distance between two shapes A and B, $D_2(A, B)$, can be computed in time $O(n^2)$ where n is the number of nodes, with preprocessing storage $O(n^2)$.

Proof:

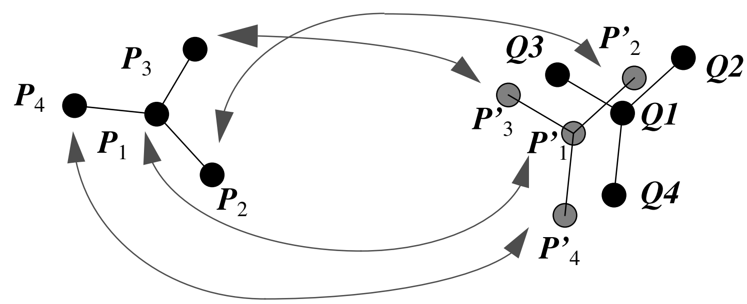
Because the total match for each node is $3n$, we can find the global minimum in $3n^2$. To speed up the search for correspondence, we can make the lookup table for each node on S_A matching each node on S_B. Since each match gives 3n correspondences, this lookup table requires $3n^2$ for storage.

Up to now we do not have exact one-to-one correspondence between S_A and S_B because of the non-uniform nature of the semi-tessellated mesh structure. But the one-to-one correspondence can be established by resampling each mesh using the regularity constraint [Shum]. Once a set of spherical mesh nodes (along with its local curvature) is obtained, it is possible to interpolate any point on the spherical coordinate from this set. For example, in Figure 7, although there exist many-to-one mappings between P and Q (both P_1, and P_2 can be matched to Q_1), mapping between P' and P is unambiguous because P' results from the rotation of P. The new mesh node at P'_1 and its SAI value can be interpolated from its nearest point Q_1 on set Q and three neighbors of Q_1. Let $g(P')$ and $g(Q_1)$ be the values of the simplex angles at node P' and its nearest node Q_1, respectively. We have the following local interpolation:

$$g(P') = \sum_{i=1}^{4} w_i Q_i \qquad \text{(EQ 1)}$$

where Q_2, Q_3, and Q_4 are three neighbors of Q_1, and w_i are the weights depending on the distance between P' and Q_i $(i=1,2,3,4)$. The coordinates of the mesh node at P' can be interpolated in the same way.

(a)



(b)

node number →

1	2	3	4	– – –	K
P_1	P_2	P_3	P_4	– – –	P_K
P'_1	P'_2	P'_3	P'_4	– – –	P'_K
Q_1	Q_1	Q_3	Q_4	– – –	Q_K

Figure 7 **One-to-one matching: (a) Valid correspondence between nodes; (b) Table of correspondences.**

4 Experiments

In this section, we present the results of applying our shape similarity metric to synthetic data and to real objects. We have used L_p distance in the metric function defined in Section 3. For all experiments below, we will use L_2 distance for the ease of computation.

Our data set consists of several polyhedra such as icosahedron and dodecahedron whose shapes are known in advance. To make deformable surfaces, we generate uniformly random-sampled data points over each object surface. We also use the free-form object model generated from real range images. Unless specified, the frequency of spherical tessellation is set to 7, which means that the total number of meshes is 980.

Figure 8 shows the approximation of a sphere by a set of regular polyhedra: a tetrahedron, a hexahedron, a dodecahedron, and an icosahedron. Since curvature is constant everywhere on a sphere, computing the minimum distance between a polyhedron and a sphere can be greatly simplified. Figure 9 shows the distance between the polyhedron (convex) and the sphere. Figure 10 shows a sequence of concave objects which are generated by making concave dents on an octahedron. We show shape similarity between this sequence of concave objects and an octahedron in Figure 11. The distance between the object (1) and the octahedron is big because it is more concave and no longer star-shaped.

Figure 12 shows a comparison of shape similarity among a set of free-form objects. The distance functions among all objects are plotted in Figure 13a where all the distances on the diagonal are zero. Figure 13b and Figure 13c show the distance between the object dog and others, and the distance between the object sha and others, respectively. Figure 14 shows the shape change from others to dog. Figure 15 shows the shape change from others to sha.

One possible drawback of our approach is that the quality of approximation of a polyhedral or free-form surface depends on the number of patches chosen. For example, with frequency

7 semi-regular spherical tessellation, we have 980 surface patches. We have 3380 patches when the frequency is 13. Obviously the more surface patches we use, the better the approximation is. Figure 16 presents the curvature distribution of an approximated hexahedron when different tessellation frequencies are used. When the higher frequency is used, the higher curvature distribution is narrower because of the better approximation. Figure 17 shows comparison of shape similarity measure when different tessellation frequencies are used. The results demonstrate that the shape similarity measure is robust provided that a sufficient number of tessellations is adopted.

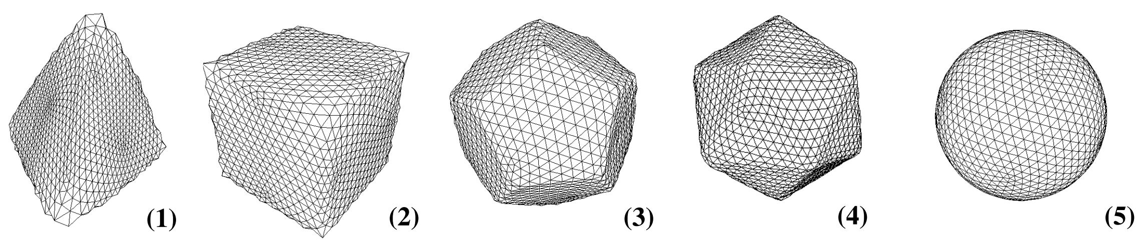


Figure 8 **Polyhedral approximation of a sphere (1) Tetrahedron; (2) Hexahedron; (3) Dodecahedron; (4) Icosahedron; (5) Sphere.**

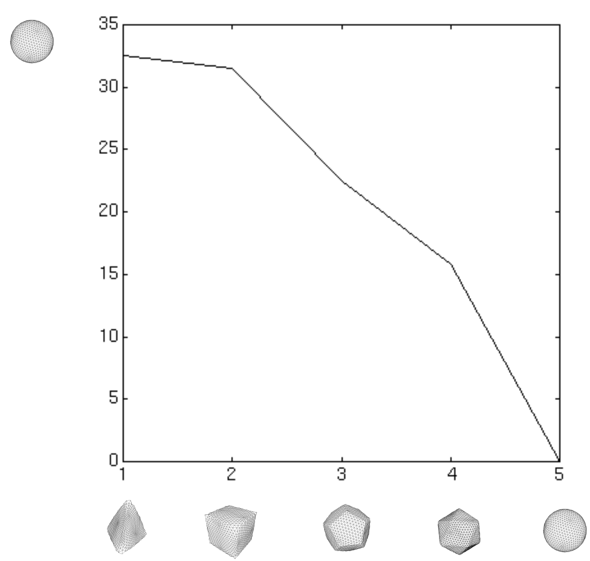


Figure 9 **Distance between a sphere and its polyhedral approximations.**

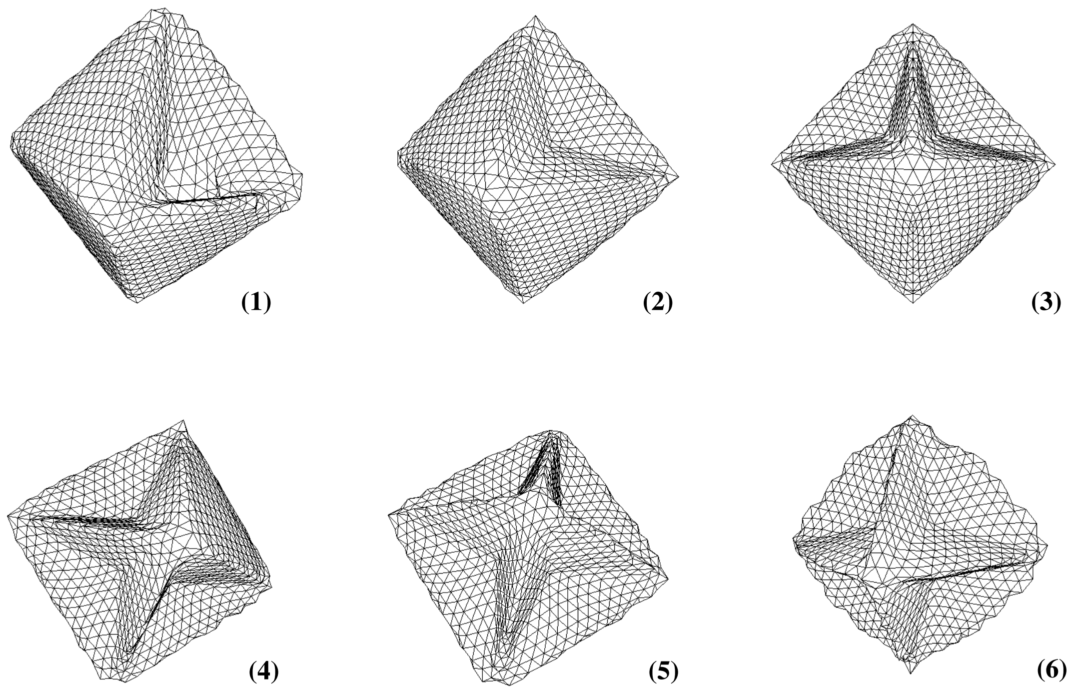


Figure 10 **Concaved octahedron (1) with one deep concave dent; (2) with one concave dent; (3) with two dents; (4) with three; (5) with four; (6) with eight.**

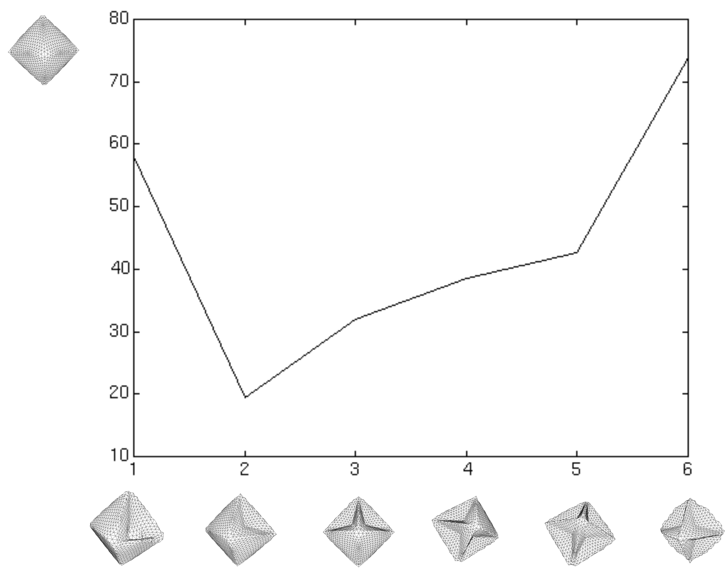


Figure 11 **Distance between an octahedron and several concaved octahedron objects.**

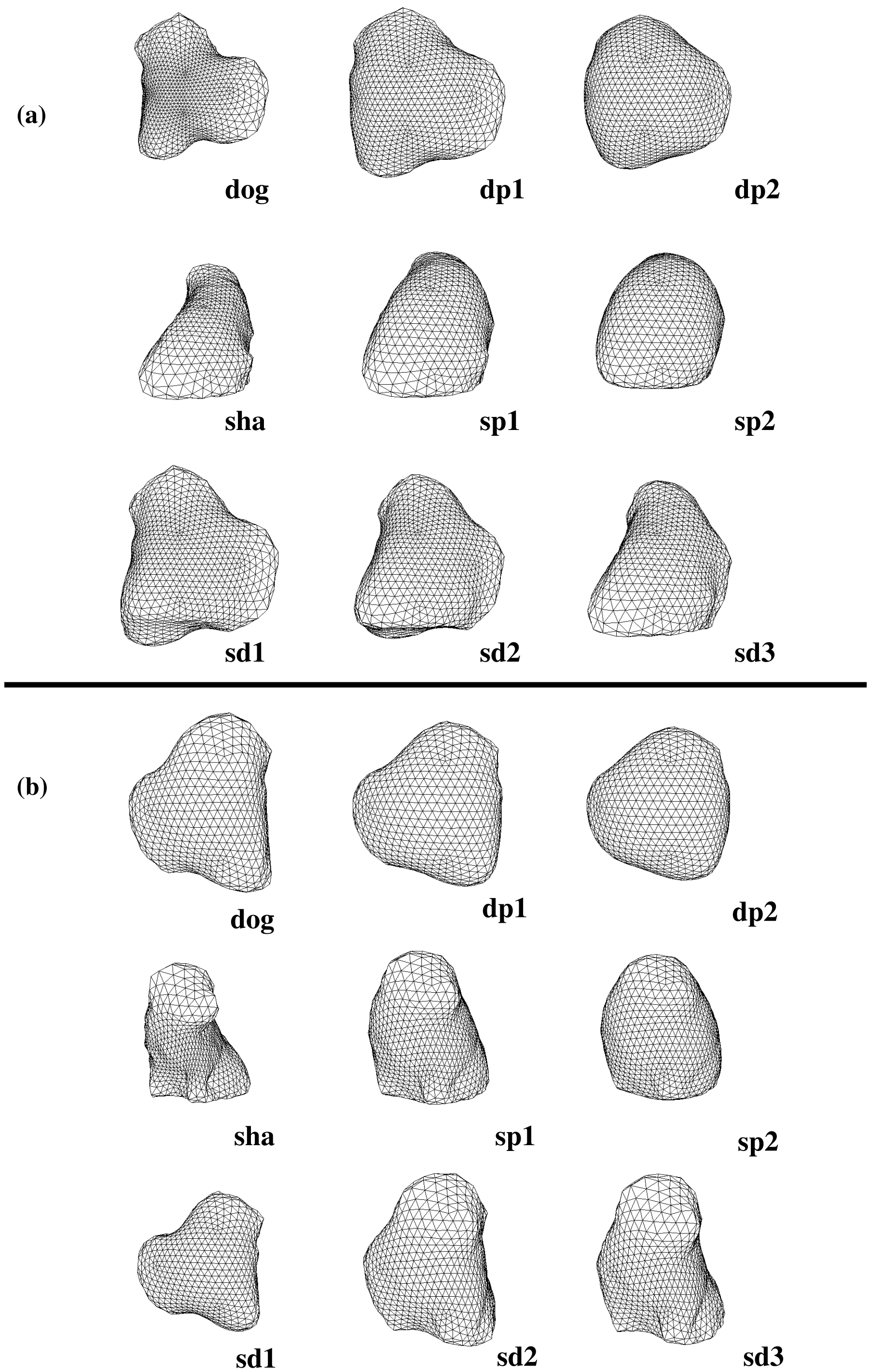


Figure 12 **(a) Free-form objects: dog and sha are generated from real range data [21]; dp1 and dp2 are two approximations of dog; sp1 an sp2 are two approximations of sha; sd1, sd3, and sd3 are three intermediate shapes between sha and dog. (b) A different view of all 9 objects.**

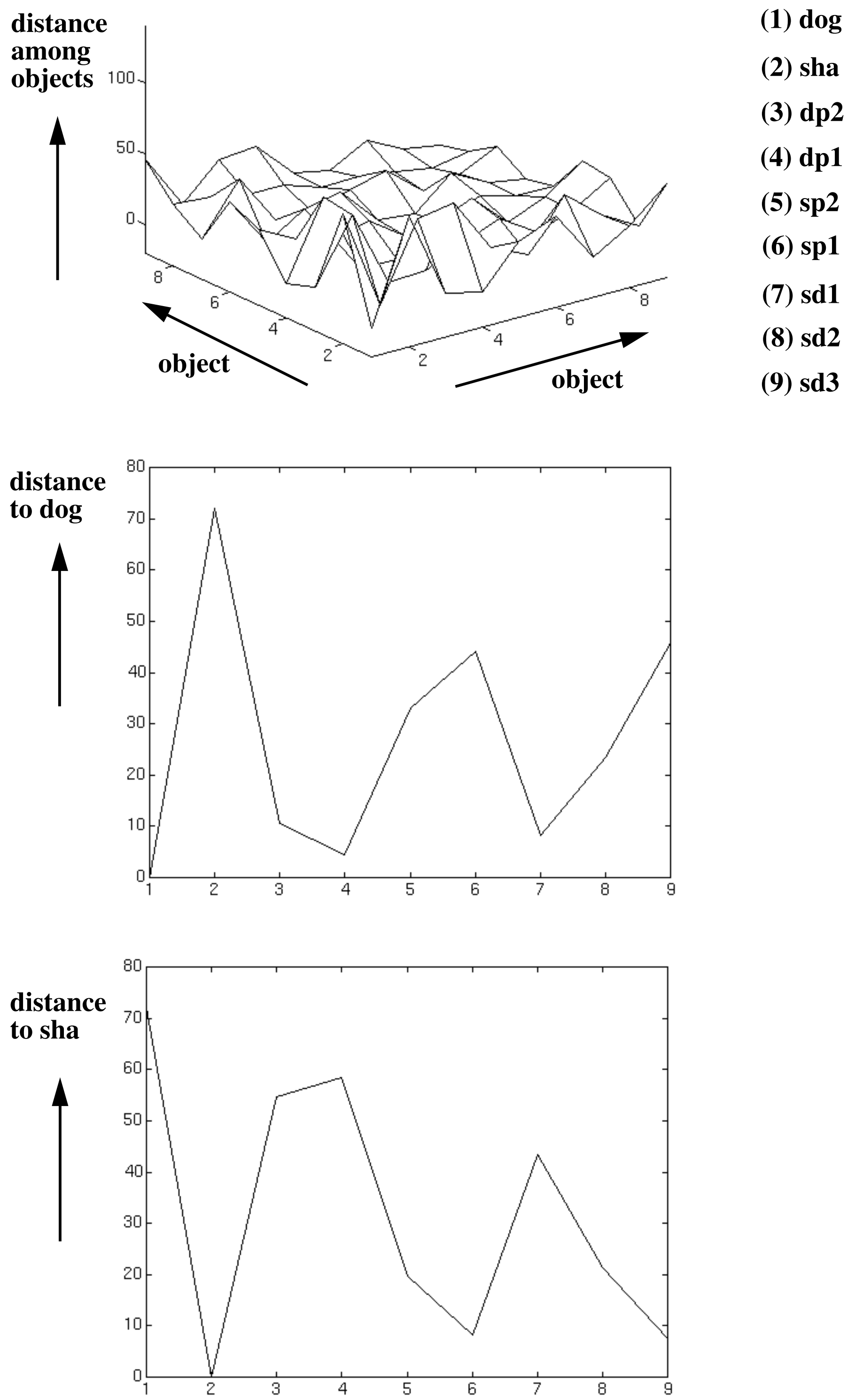


Figure 13 **(a) Shape similarity among all free-form objects: distance of pair-wise comparison. (b) The distance between the object dog and others. (c) The distance between the object sha and others.**

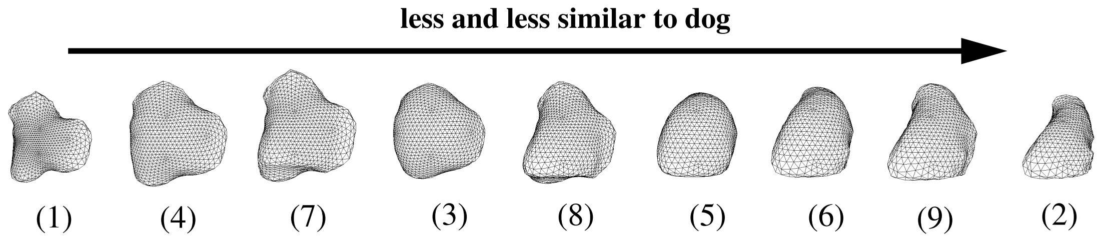


Figure 14 **Shape change from the object dog to others. From left to right: shapes are more and more dissimilar to dog.**

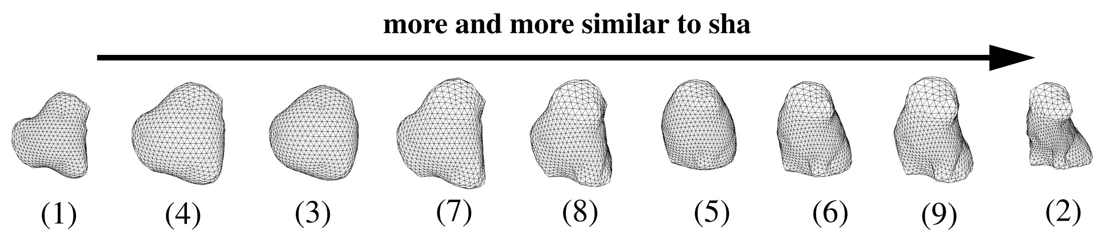


Figure 15 **Shape change from the object sha to others. From left to right: shapes are more and more similar to sha.**

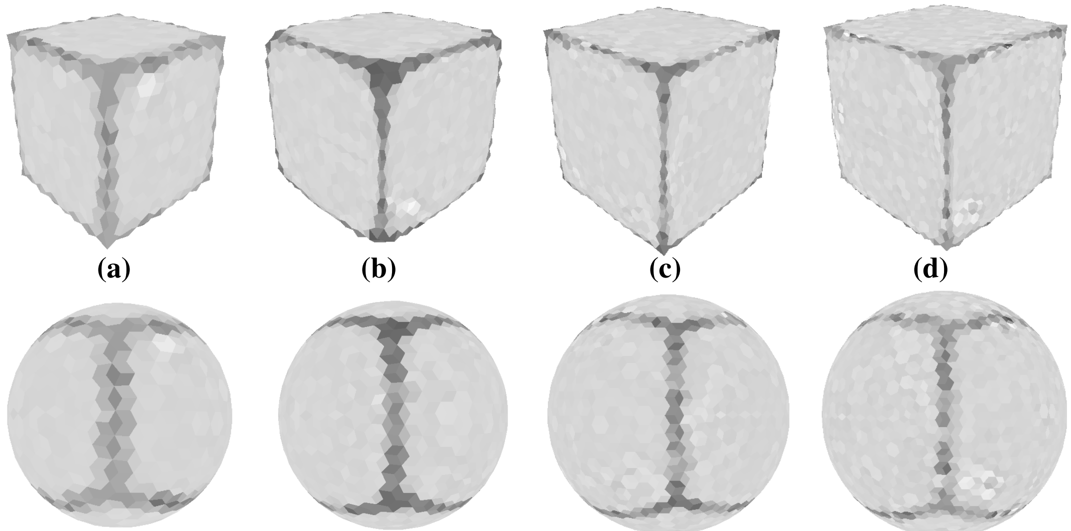


Figure 16 **An example (hexahedron) of curvature distribution of mesh representation at tessellation frequencies: (a) f=7; (b) f=9; (c) f=11; (d) f=13.**

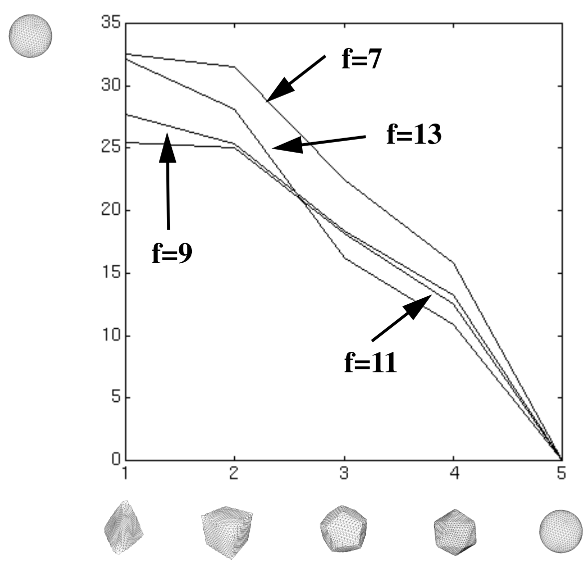


Figure 17 **Effect of tessellation frequency on shape similarity between regular polyhedron and a sphere.**

5 Conclusion

We have employed a new data structure for storing object shapes of genus zero, both free form and polyhedral. We build a spherical mesh representation that has nearly uniform distribution with known connectivity among mesh nodes. We iteratively deform a semi-tesselated sphere so that the deformed mesh representation converges to the original shape. The local curvature computed at each node captures the averaged curvature information in its vicinity. The task of comparing two shapes is essentially one of comparing two curvature distributions generated from deformed meshes. An important observation is that, unlike the curvatures on sparse vertices and edges on polyhedra, the curvature distribution (such as on a mesh representation on a sphere) can be used to compare shapes efficiently and effectively.

Based on the special structure of our mesh representation, we have also proposed an efficient algorithm to computing the shape metric between two objects. We use the distance function to analyze shape similarity between sets of concave and convex objects. Experiments show that our shape similarity metric is robust and invariant under rigid transformation and scaling, easy to compute, and intuitive with human perception on shape.

Our approach is, in spirit, similar to the one used by Schwartz and Sharir [19] where they approximated a 2D curve from noise data points by discretizing the turning function (a 2D curvature in some sense) of two polygons into many equally spaced points. We discretize the polyhedral and/or free-form surfaces into many approximately equally spaced patches. An advantage of our distance function is that it is stable under a certain amount of noise. Even with non-uniform noise, we can keep most part of object well represented.

Currently our approach is restricted to genus zero shape topology. Recent progress on geometrical heat equation and geometry diffusion shed some light on how to compare topological shape similarity as well as geometrical similarity. We will work in this direction.

Acknowledgment

We thank Yan-bin Jia for his many valuable comments on shape metrics, and Michael Nechyba for his help on generating regular polyhedra using Mathematica. Thanks to Marie Elm for proofreading different versions of this paper.

References

[1] E. Arkin, L. Chew, D. Huttenlocher, K. Kedem and J. Mitchell. An Efficient Computable Metric for Comparing Polygonal Shapes. IEEE Trans. PAMI Vol. 13 No. 3, pp. 209-215, 1991.

[2] B. Bhanu. Representation and Shape Matching of 3D objects. IEEE Trans. PAMI Vol. 6, pp. 340-351, 1984.

[3] R. Basri, L. Costa, D. Geiger and D. Jacobs. Determining the Similarity of Deformable Shapes. Proc. of Workshop on Physics-based Modeling in Computer Vision, pp. 135-143, Boston, June 18-19, 1995.

[4] P. Besl and R. Jain. Segmentation Through Variable-Order Surface Fitting. IEEE Trans. PAMI Vol. 10, No. 2, pp. 239-256, 1988.

[5] C. Brechbuhler, G. Gerig, and O. Kubler. Parametrization of Closed Surfaces for 3D Shape Description. Computer Vision and Image Understanding, Vol. 61, No. 2, pp. 154-170, March, 1995.

[6] R. Chin and C. Dyer. Model-based Recognition in Robot Vision. ACM Computing Surveys, Vol. 18, No. 1, 1986.

[7] H. Delingette. Simplex Meshes: a General Representations for 3D Shape Reconstruction. INRIA report 2214, 1994.

[8] M. Hebert, K. Ikeuchi and H. Delingette. A Spherical Representation for Recognition of Free-form Surfaces. IEEE Trans. PAMI Vol. 17 No. 7, pp. 681-690, 1995.

[9] B.K.P. Horn. Extended Gaussian Image. Proc. of IEEE, Vol. 72, No. 12, pp. 1671-1686, December, 1984.

[10] D. Huttenlocher and K. Kedem, Computing the Minimum Hausdorff Distance for Point Sets Under Translation. Proc. ACM Symp. Computational Geometry, pp. 340-349, 1990.

[11] K. Ikeuchi and M. Hebert. From EGI to SAI. CMU-CS-95-197.

[12] S. Kang and K. Ikeuchi. The Complex EGI: New Representation for 3D Pose Determination. IEEE Trans. PAMI Vol. 15, No. 7, pp. 707-721, July, 1993.

[13] J. Koenderink. Solid Shape. The MIT Press, Cambridge, 1990.

[14] K. Kupeev and H. Wolfson. On Shape Similarity. Proc. Int. Conf. Pattern Recognition. pp. 227-237, 1994.

[15] J. Little. Determining Object Attitude for Extended Gaussian Image. Proc. IJCAI, Los Angeles, California, pp. 960-963, August, 1985.

[16] D. Mumford. Mathematical Theories of Shape: Do They Model Perception? SPIE Vol. 1570 Geometric Methods in Computer Vision. pp. 2-10, 1991.

[17] H. Murase and S. Nayar. Learning and Recognition of 3D Objects from Appearance. International Journal of Computer Vision, pp.1-24, Jan 1995.

[18] V. Nalwa. Representing Oriented Piecewise C2 Surfaces. Proc. 2nd ICCV, pp. 40-51, December, 1988.

[19] J. Schwartz and M. Sharir. Identification of Partially Obscured Objects in Two and Three Dimensions by Matching Noisy Characteristic Curves. The Int. J. Robotics Research Vol. 6 No. 2, pp. 29-44, Summer 1987.

[20] S. Sclaroff and A. Pentland. Object Recognition and Categorization Using Modal Matching. Proc. 2nd CAD-Based Vision Workshop, pp. 258-265. Champion, Pennsylvania, Feb. 8-11, 1994.

[21] H. Shum, M. Hebert, K. Ikeuchi and R. Reddy. An Integral Approach to Free-form Object Modeling. CMU-CS-95-135, May 1995.

[22] D. Terzopoulos, A. Witkin and M. Kass. Symmetry-Seeking 3D Object Recognition. Int. J. Computer Vision. Vol. 1, No. 1, pp.211-221, 1987.

[23] A. Tversky. Features of similarity. Psychological Review. 84(4), pp.453-462.