Multi-Scale Classification of 3D Objects

Dongmei Zhang and Martial Hebert

CMU-RI-TR-96-39

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania 15213

November 29, 1996

This research was supported by NSF Grant IRI-9224521. The views and conclusions expressed in this document are those of the authors and should not be interpreted as representing the policies and endorsements of the National Science Foundation, or the U.S. Government.

Keywords: Object recognition, object classification, 3-D mesh smoothing, 3-D shape similarity, shape metric.

Abstract

We describe an approach to the classification of 3-D objects using a multi-scale representation. This approach starts with a smoothing algorithm for representing objects at different scales. In a way similar to the classical scale space representations, larger amount of smoothing removes more details from the surfaces. Smoothing is applied in curvature space directly, thus avoiding the usual shrinkage problems and allowing for efficient implementations. A 3-D similarity measure that integrates the representations of the objects at multiple scales is introduced. This similarity measure is designed to give higher weight to the coarse scale representations, while ignoring the finer scale details of the surfaces. Given a library of models, objects that are similar based on this multi-scale measure are grouped together into classes. We show how shapes in a given class can be combined into a single prototype object. This is achieved by using a powerful property, introduced earlier, of inverse mapping from representation to shape. Finally, the prototypes are used for hierarchical recognition by first comparing the scene representation to the prototypes and then matching it only to the objects in the most likely class rather than to the entire library of models. Beyond its application to object recognition, this approach provides an attractive implementation of the intuitive notions of scale and approximate similarity for 3-D shapes.

Table of Contents

List of Figures

1 Introduction

A number of algorithms and representations have been proposed for the recognition of 3-D objects in 3-D scenes. These techniques are typically demonstrated by using a single model, or a small set of models. In practice, however, one may have to contend with large libraries of objects. In that case, models are matched with the scene in a sequential manner. The difficulty is the inability to represent the notion that many objects in the library have "a similar overall shape" and that, therefore, they could be grouped together and recognized as a group.

Ideally, we would like to be able to first identify candidate objects at a coarse scale, *i.e.*, based on their overall shapes, and to then identify the actual objects present in the scene at a finer scale, *i.e.*, based on a detailed description of their shapes. Such a multi-resolution approach is more efficient since it attempts to match surfaces at full resolution only for those objects identified as candidates at a coarser resolution.

Although such an approach has not been developed for 3-D shapes, the equivalent approach for 2-D images using template matching at multiple resolution and template grouping at coarse resolution is well-known. The basic idea described in this paper is to develop a similar multi-scale approach for 3-D shape matching.

Our approach starts with a simple smoothing algorithm that can be applied to surfaces in a way similar to image smoothing. This smoothing algorithm has the desired intuitive properties. In particular, as the amount of smoothing increases, the surface evolves from a complete description of the object to a coarser description in which details are omitted, and, as the amount of smoothing becomes very large, to a sphere.

This smoothing algorithm allows us to compare models in a model library at multiple scales. Given two models, the comparison is done by combining the values of a similarity measure between the two objects at different scales, i.e., different degrees of smoothing. The combination is defined in such a way that models at coarser scales contribute more to the overall similarity measure. Assuming that all the models are compared in this manner, they can be grouped into classes of similar objects. Each class is represented by a prototype object which is the "average" of all the objects in the group.

The notion of representing and comparing data sets at different degrees of smoothing instead of at a single scale is, of course, not new. The 2-D concept of scale space has been used extensively, both for rep-

resenting features in images and for representing curves [19][8][11][7]. The difficulty is to extend this to three-dimensional surfaces in a way that is simple and computationally efficient.

At recognition time, the representation of the observed scene is first compared to the prototype objects. The class corresponding to the prototype object that is most similar to the scene is the set of candidate objects for recognition. Final recognition is performed by comparing the objects of this group with the scene at full resolution.

Examples of related approaches to 3-D shape classification include the use of global shape parameters [13], which can also support multiple scales; the use of parameterized surfaces [9]; and the use of neural networks for classification [4][3]. Our approach differs in that we explicitly build prototype objects for each class, and that we do not require parametrization or global shape indices.

Beyond its direct application to object recognition this approach answers several challenging questions in the area of object representation. First of all, the smoothing algorithm provides a simple and intuitive way of capturing the notion of “level of detail”, “scale”, and “overall shape” of an object. Second, the grouping algorithm provides a way to implement the intuitive concept of the “average” shape of a group of similar objects. Finally, the multi-scale algorithm is a first step toward hierarchical representation of shape libraries.

The paper is organized as follows: In section 2, we describe the basic shape representation used in this paper. In the following sections, we describe the three components of smoothing, grouping, and matching. We conclude with some examples of real objects.

2 Shape Representation and Shape Similarity

The mesh models used in this paper are built up using the deformable surface technique described in [5]. The object’s surface is represented by a mesh in which each vertex is of degree 3. The number of vertices is determined by the frequency of tessellation. It can be shown that, by using appropriate constraints on the relative positions of the vertices, the nodes of the mesh are distributed in a nearly uniform manner on the surface. A complete discussion of the uniformity properties can be found in [6].

A local shape measure is computed locally at each node of the mesh, as defined in [5]. Although this measure is not a direct approximation of surface curvature, we will refer to it as “discrete curvature” in the rest of the paper for simplicity. Details on the relation between the shape measure used in [5]and other measures of surface curvature can be found in [16].

Under the assumption that the mesh is uniform, it can be shown that a measure of similarity between the two meshes can be computed. In order to compute the similarity measure, it is convenient to intro-

duce an intermediate representation, which is a spherical representation of the discrete curvature distribution. In this representation the discrete curvature value at each node of the surface mesh is mapped to the corresponding node on a unit sphere tessellated using the same mesh topology and number of nodes. We will refer to this representation as the "shape distribution function" or the "curvature distribution function". Therefore, for each object, there is a mesh model in 3-D space and a corresponding spherical shape representation in curvature space to represent it. Using a spherical representation is for convenience only. In particular, it does not imply that the objects are convex or star-shaped.

Given the spherical representations of two objects *A* and *B*, the shape similarity is measured by finding the rotation that minimizes the difference of discrete curvature values at corresponding nodes of the two spheres. Formally, let S_A and S_B be the mesh representations of *A* and *B*, respectively. Let $k(S_A)$ and $k(S_B)$ be the shape distribution functions for S_A and S_B, respectively. We denote by $k_R(S_B)$ the shape distribution function of S_B after rotation by *R*. The L_p distance $d_p(S_A, S_B, R)$ between *A* and *B* for a given spherical rotation *R* is defined as

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

which is the sum of local discrete curvature differences over the sphere. Then the shape similarity measure $D_p(A, B)$ between *A* and *B* becomes

$$D_p(A, B) = min_R d_p(S_A, S_B, R) \quad (2)$$

which is the minimum of d_p over all possible rotations *R*. D_p is a distance between shapes, *i.e.*, it is symmetrical and satisfies the triangle inequality. We refer the reader to [14]for a complete description of the properties of this similarity measure. An efficient algorithm for implementing the minimization in rotation space is described in [6].

This similarity measure is an extension of the similarity measure for 2-D curves defined in [1] or [12], which uses the turning angle along a 2-D curve as the equivalent of our discrete curvature measure. Other approaches to shape similarity are based on shape deformation metrics [2][13].

So far, we have described the "forward mapping" from surface to shape representation. A critical property of this representation is that the "inverse mapping", *i.e.*, constructing a 3-D shape from a given discrete curvature distribution, is also possible. This property is the basis of our approach for representing a group of objects by a single prototype. The inverse mapping is described in detail in [15].

3 Smoothing

There has been extensive previous work on smoothing piece-wise linear shapes of arbitrary dimension and topology. Polygonal meshes are of particular interest because of their wide application in modeling

3-D objects. The main disadvantage of mesh smoothing is the shrinkage of mesh that results from repeated averaging. Although several solutions to the shrinkage problem have been proposed [17][10][18]. Most of those approaches require the use of careful design of the smoothing operator and, to the exception of [17], are hard to extend to 3-D. In this section, mesh smoothing is studied from a new point of view. Instead of smoothing the mesh directly in 3-D space, the shape representation of the mesh is smoothed in curvature space. By reconstructing a new mesh based on the smoothed shape representation, the mesh in 3-D space is indirectly smoothed. This approach to mesh smoothing has two advantages in that it does not generate shrinkage on the mesh; and it does not involve direct computation on the mesh, which is an advantage when only the local discrete curvature distribution is needed.

Before describing the smoothing algorithm, we need to define a few notations: A unit discrete sphere is represented as $S=\{V, E, C\}$, in which V is a list of vertices, E is a list of edges and C is a list of local discrete curvatures. V, E, C are defined more clearly as follows:

$V = \left\{ v_i \middle| 1 \le i \le n_v, \|v_i\|^2 = 1 \right\}$, n_v is the total number of vertices on the sphere;

$E = \{ e_{ij}^k \mid 1 \le i \le n_e, e_{ij}^k = (v_i, v_j) \}$, n_e is the total number of edges on the sphere;

$C = \{ c_i | (1 \le i \le n_v) \}$, C is the set of local discrete curvature values.

A "unit mesh distance" is defined as the distance between two neighboring vertices:

$d_{unit} = \|v_i - v_j\|$, $\exists e_{ij}^k$, $1 \le k \le n_e, 1 \le i, j \le n_v$

The mesh distance d_{ik} from v_k to v_i is defined as the number of unit mesh distances between v_i and v_k.

With those notations, the smoothed discrete curvature value at a node v_i is defined by:

$$c_i^\sigma = \alpha_{norm} \sum_{v_j \in W} e^{\frac{-1}{2}\frac{d_{ij}^2}{\sigma^2}} c(v_j) \tag{3}$$

In (3), d_{ij} is the mesh distance from node v_j to node v_i and W is the neighborhood of v_i defined by $0 \le d_{ij} \le w$, where w is the size of the smoothing operator. $c(v_j)$ is the local discrete curvature at node v_j. c_i^σ is the smoothed local discrete curvature at v_i at scale σ.

In practice, the size of the operator is selected as $w=4\sigma$ and the normalization factor α_{norm} is defined by:

$$\alpha_{norm} = 1 \Big/ \left(\sum_{v_j \in W} e^{\frac{-1}{2}\frac{d_{ij}^2}{\sigma^2}} \right) \tag{4}$$

For a given vertex v_i, the vertices which are d mesh distances away from v_i can be found easily by traversing the edge list E.

This definition is the direct extension to the sphere of Gaussian smoothing on 2-D images:

$$F_{ij} = \frac{1}{2\pi\sigma^2}\sum_k\sum_l e^{\frac{-1}{2}\frac{k^2+l^2}{\sigma^2}} f(i+k, j+l) \tag{5}$$

in which k, l are the pixel distances from pixel (i, j). The extension to the spherical image involves changing the definition of neighborhood and distance to take into account the topology of the mesh.

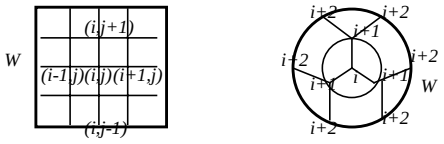


Figure 1: Comparison of pixel distance on an image with mesh distance on a discrete sphere.

There is a limit to the size of the smoothing operator, w. The basic consideration here is that the sphere is a closed surface, which implies that the neighborhood W wraps around on itself if w is too large. Therefore, there is an upper limit w_{max} for w. Starting at a vertex, this w_{max} is calculated by traversing the list E until no more new vertices can be added.

This implementation of smoothing assumes that all the vertices are included in the computation, and therefore, the entire surface of the object is known. In practice, however, parts of the surface may not be visible. As a result, some of the vertices on the sphere may not correspond to the underlying surface. To address this problem, smoothing is not applied to those vertices and $c(v_j)$ is ignored in (3) if v_j does not correspond to the underlying surface.

Using the above definition of smoothing, the local discrete curvature distribution becomes a constant distribution as σ becomes large. In fact, this is true for all 3-D objects which can be represented by the spherical discrete curvature function. This can be shown as follows: As indicated earlier, as σ increases, the neighborhood will reach its maximum size w_{max} due to the boundedness of the sphere. The corresponding neighborhood W is the entire sphere S. At that point, the smoothed value at node v_i is computed as:

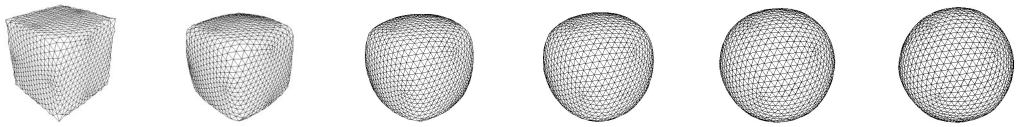
$$c_i^\sigma = \left(\sum_{v_j \in S} e^{\frac{-1}{2}\frac{d_{ij}^2}{\sigma^2}} c(v_j)\right) \Big/ \left(\sum_{v_j \in S} e^{\frac{-1}{2}\frac{d_{ij}^2}{\sigma^2}}\right)$$

As σ increases further, the coefficients of the smoothing operator all converge to 1. Therefore, in the limit, c_i^σ converges to the average value $\bar{c}$ of discrete curvature over the entire sphere:

$$c_i^\sigma = \left(\sum_{v_j \in S} c(v_j)\right) \Big/ N = \bar{c} \tag{6}$$

This convergence property is simply the property equivalent to that of any 2-D image smoothing, which is that the continuously smoothed image converges to a constant image whose value is the average of the input image. In particular, this is independent of the shape distribution function defined on a sphere (In fact, there are many ways to represent local discrete curvature [16]).

A consequence of (6) is that, as σ increases, all objects converge to a shape of identical discrete curvature everywhere, namely a sphere. This behavior is consistent with the intuitive behavior of smoothing: as more and more details are eliminated from an object surface, that object should converge to the simplest object, a sphere.



(a) (b) (c) (d) (e) (f)

Figure 2: Reconstructed meshes of the cube from smoothed shape representations. (a) σ=0; (b) σ=0.5, (c) σ=1.0; (**d)** σ=2.0; (**e)** σ=**4.0; (f)** σ=9.0.

For example, Figure 2 shows that, as the amount of smoothing increases, the cube gradually converges to a sphere. This sequence was obtained by smoothing the spherical distribution and by using the inverse mapping algorithm of [15] to construct the corresponding 3-D shapes.

In practice, different objects modeled by discrete meshes of the same frequency may converge to slightly different values of $\bar{c}$ due to round-off errors, and to the fact that the sampling is not perfectly uniform. However, the differences are small enough that they do not affect the classification algorithms. For example, Table 1 shows the value of the mean and variance of *c* for four different objects for σ =9.0. Those numbers show that the discrete curvature distribution does become constant as the amount of smoothing becomes large.

Table 1: Mean and Variance of *c* for Different Objects After a Large Amount of Smoothing.

Object	Mean of *c*	Variance of *c*
Cube	8.03e-2	1.0e-6
Tetrahedron	7.25e-2	2.6e-5
Octahedron1	7.53e-2	1.3e-5
Octahedron2	7.59e-2	2.2e-5

4 Classification

The basic idea of the classification algorithm is to compute the representations of the objects in the library at different scales and to compute a new similarity measure which includes all the scales. Figure 3 shows a simple example of the comparison of two objects, a cube and a sphere. The graph shows that, as the degree of smoothing is increased, the two objects become more similar. Our goal is to combine all the values of D_p computed at different scales σ into a single measure, which we call the "scaled shape similarity metric".

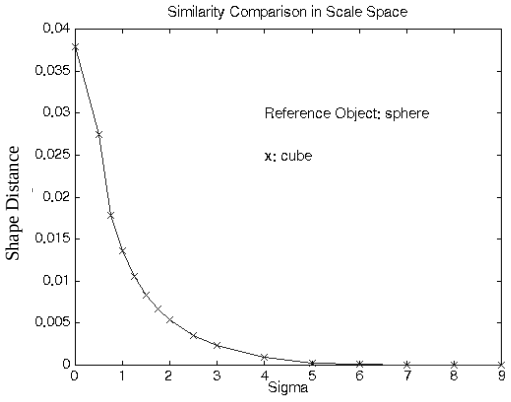


Figure 3: Shape comparison of cube and sphere. The x-axis is the scale variable σ and the y-axis is the shape distance D_p between the cube and the sphere. Here the sphere is selected as the reference object.

Formally, the scaled shape similarity metric $d_\Omega(A,B)$ between two objects A and B is defined as follows:

$$d_\Omega(A, B) = \sum_{\sigma \in \Omega} w_\sigma D_p^\sigma (A, B) \tag{7}$$

In this expression, Ω is the set of scales σ; w_σ is the weight at scale σ; $D_p^\sigma(A,B)$ is the shape similarity metric defined in (2) with σ indicating the comparison is done at scale σ.

In previous sections, we have discussed how to select σ_{max} which defines the range of values that σ may take and how to compute $D_p^\sigma(A,B)$. There remains one parameter w_σ which needs to be determined.

The purpose of w_σ is to give more weight to coarse representations of the object so that $d_\Omega(A,B)$ compares objects based on their overall shapes rather than on detailed surface features. Therefore, the more smoothing is done, the larger w_σ should be. We set w_σ to be the average number of vertices which are within σ mesh distances from any arbitrary vertex v_i on a sphere.

Intuitively, w_σ is the area of the sphere that is used in the computation of the smoothed discrete curvature values. For instance, given a sphere with 980 vertices, w_σ is computed with respect to different scale factor σ and listed in Table 2.

Table 2: Weights for Different Scale Factors.

σ	w_σ
0	1.00
0.5	9.89
1.00	30.31
2.00	103.00
4.00	347.39
9.00	943.91

The scaled shape similarity metric d_Ω has the same properties as that of shape similarity metric D_p because d_Ω is a linear combination of D_p. In particular, d_Ω is symmetric and satisfies the triangle inequality.

Let us illustrate the use of the scaled shape similarity measure for grouping objects into classes using a simple example containing five synthetic objects: cube, tetrahedron, octahedron, octahedron with one indentation (referred as octahedron1) and octahedron with two indentations (referred as octahedron2). Figure 4 shows the mesh models and the corresponding spherical shape representations.

In order to do shape comparison at multiple scales, the modified smoothing algorithm is applied to those five objects. Figure 5 shows the distance of each object to the octahedron as a function of scale.

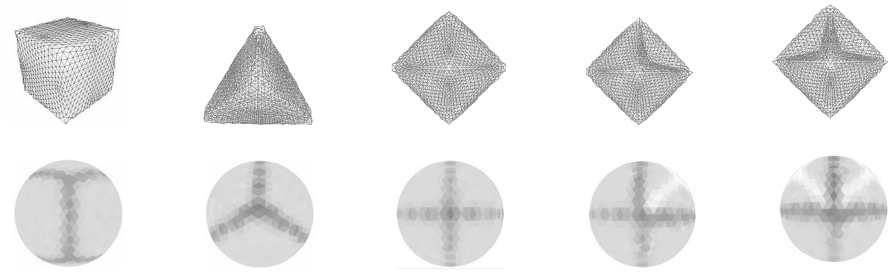


Figure 4: Meshes and their corresponding shape representations. (a) cube; (b) tetrahedron; (c) octahedron; (d) octahedron1; (e) octahedron2.

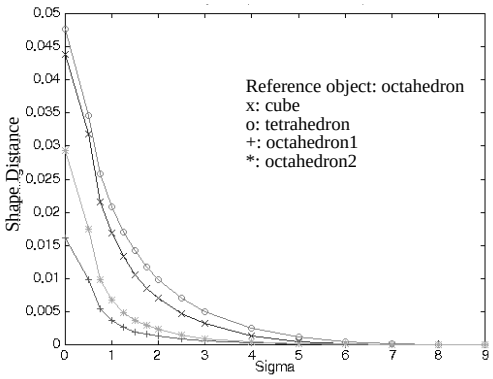


Figure 5: Shape comparison of cube, tetrahedron, octahedron1 and octahedron2 with respect to octahedron.

The scaled shape similarity metric for each of the four objects is shown in Table 3. This example shows that the scaled shape similarity metric classified octahedron1 and octahedron2 into one group, octahedron1/octahedron2.

Table 3: Scaled Shape Similarity Values of Four Objects

Objects	cube	tetra	octa1	octa2
$d_{\Omega}(x, octa)$	6.13	9.15	1.31	2.27

After classification, a prototype object is generated for each class so that, when compared with an object in a scene, the prototypes are used first to determine to which class the scene object belongs.

A natural approach is to take the spherical representations of all the objects in a given a class and to compute the average of the discrete curvature values at each node among all the objects. The resulting spherical distribution does not correspond to a physical object; rather it is the shape distribution of the "average" object of the class.

Given the average shape distribution, a 3-D surface of a new object can be constructed using the inverse mapping algorithm of Figure 15. The resulting average object is the prototype for the class. Note that most of the algorithms operate directly on the spherical distribution, so that the reconstruction of the prototype surface mesh is not always necessary.

Figure 6 shows the prototype obtained for the class octahedron1/octahedron2. The meshes of octahedron1 and octahedron2 are also shown for comparison.

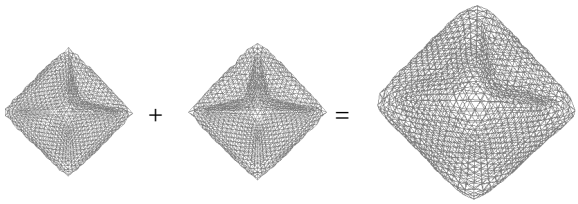
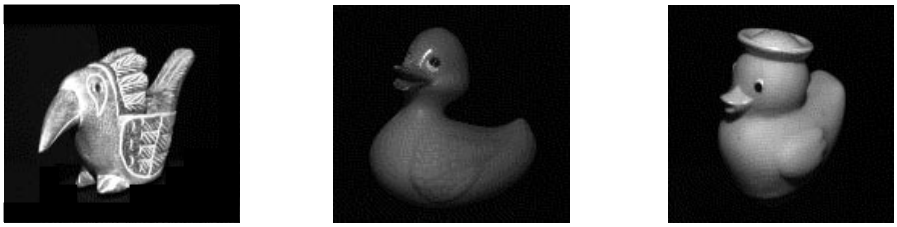


Figure 6: Prototype object for the class octahedron1/octahedron2 reconstructed by inversely mapping the averaged spherical representations to the surface mesh.

5 Classification Example

The classification algorithm was tested using the three objects shown in Figure 7.



(a) (b) (c)

Figure 7: Objects in library. (a) object1; (b) object2; (c) object3.

First, mesh models of the above objects are built by running the deformable surface program. Each of the three meshes has 980 vertices. After the mesh models are obtained, the corresponding shape representation of each object is constructed (Figure 8).

Next, in order to do shape comparison at multiple scales, the above shape representations are smoothed. Figure 9 shows the smoothing results and the corresponding reconstructed meshes of object3.

At the third step, shape comparison is done at multiple scales between pairs of objects. In this example, object2 is selected as the reference object. The comparison result is show in Figure 10.

Figure 10 shows that if object3 and object1 are compared with object2 without any smoothing, then *object1* is more similar to object2 as opposed to our intuition, which is that object3 is more similar to object2. However, this is not totally surprising because, at that level of detail, object3 and object2 are sufficiently different in shape. In addition, the noise in the data is fully reflected in the model, particularly at the high curvature points which contribute the most to the matching, thus generating the wrong value for the similarity.

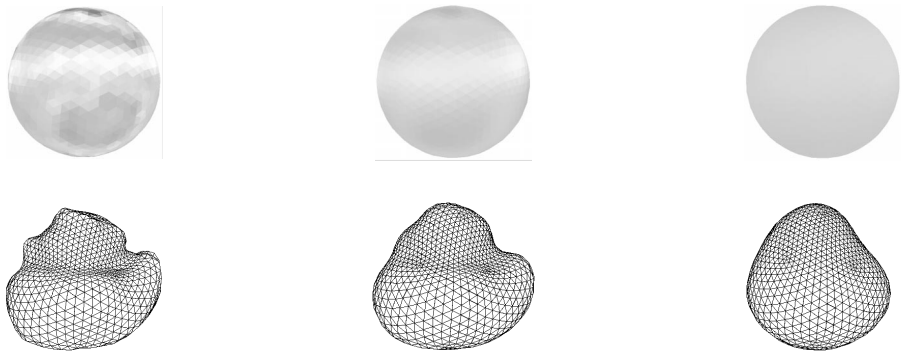
By contrast, the correct decision is made by using the scaled shape similarity d_Ω. More precisely, using the values of w_σ listed in Table 2, the scaled similarity values are:

d_{Ω}(*object3*,*object2*)=15.89, d_{Ω}(*object1*,*object2*)=21.89

Based on those values, object2 and object3 are correctly grouped in the same class. After classification, a prototype object is computed for the object2-object3 class. Figure 11 shows the result.

(a) (b) (c)

Figure 8: Mesh models and their corresponding shape representations. (a) object1; (b) object2; (c) object3.



(a) (b) (c)

Figure 9: Object3: smoothed shape representations and the corresponding reconstructed meshes. (a) σ=0.5; (**b)** σ=2.0; (**c) σ=5.0.**

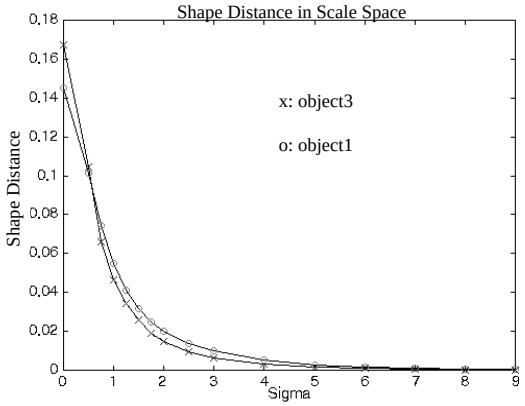


Figure 10: Distance to object2 as a function of scale.

.

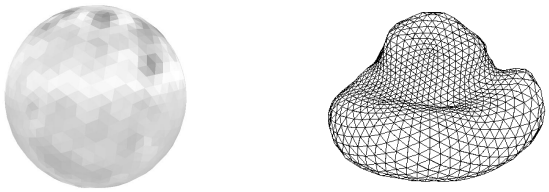


Figure 11: Shape representation and reconstructed mesh model for the object2-object3 class.

Using the example shown in Figure 10:, we can also verify our earlier claim that the weight w_σ correctly reflects the desired contribution of different scales. Figure 12 shows the percentage of each weighted shape similarity value in the final scaled shape similarity metric:

$$p_\sigma = \frac{w_\sigma D_p^\sigma}{\sum_{\sigma \in \Omega} w_\sigma D_p^\sigma}. \tag{8}$$

From Figure 12, we can see that the shape similarity value at σ=0 does not contribute much to d_Ω. This solves the classification confusion we discussed previously (object1 is more like object2). Nor do the similarity values at large σs, e.g. $\sigma \geq 5$, contribute much. This is reasonable because when shapes are smoothed too much, they all become similar, thus making classification difficult. The scales which contribute most are within the range of $1 \leq \sigma \leq 4$. At those scales, the details of the shape are smoothed off effectively, while the important characteristics of the shape are preserved.

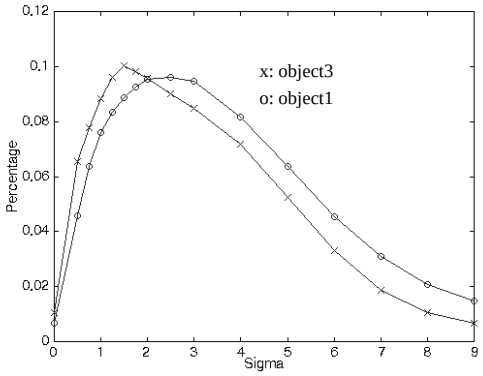


Figure 12: Percentage of shape similarity value at each scale in the scaled shape similarity metric.

This example illustrates the importance of doing classification at multiple scales: When classifying objects, the objects belonging to the same class are not expected to be of identical shape. It is the similarity of their "rough" shapes that is more important for classification. By smoothing their shape representations at different scales, the different details are smoothed off and the "rough" shapes of those objects become dominant in shape comparison.

6 Recognition

At recognition time, a scene description is first matched to the prototypes to determine the most likely class, and then matched with the objects of the best class for final identification.

Figure 13(a) shows a scene which contains object2. The corresponding range data was taken using a different setup from the one that was used for building the object2 model. A mesh model of the partially viewed object2 in the scene is built (Figure 13(b)) and the corresponding shape representation is also obtained (Figure 13(c)). The shape representation of object2 in the scene is then compared to the shape representations of the two classes. The following matching errors are obtained using the shape similarity D_p:

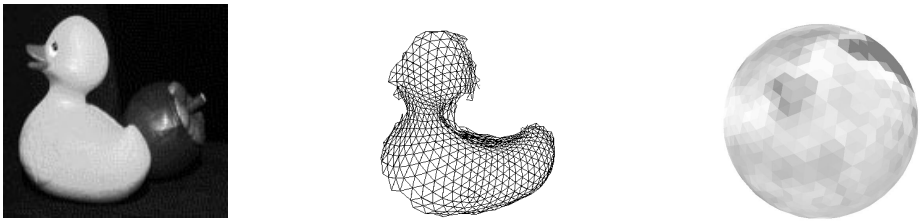
$e_{object2\text{-}object3}$=0.0953, $e_{object1}$=0.0991

Therefore, as we expected, object2 in the scene falls into the object2-object3 class. The prototype for the object2-object3 is the object shown in Figure 11.

Next intra-class recognition needs to be performed in order to determine whether object2 in the scene is actually object2 or object3. The matching errors are shown below, which indicates that the object in the scene is recognized as object2.

$e_{object2}$=0.0851, $e_{object3}$=0.0987

At the final step, the transformation between object2 in the scene and object2 in the library is computed for verification. Figure 14 shows the mesh of object2 in the scene overlapped on the mesh of the model.



(a) (b) (c)

Figure 13: (a) Scene; (b) Mesh; (c) Shape representation.

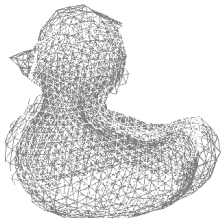


Figure 14: Overlapped scene and model after transformation.

7 Conclusion

We have proposed an algorithm for 3-D object classification. This algorithm mainly consists of three stages. The first stage is the application of our modified smoothing algorithm to spherical shape representations. The advantages of this smoothing algorithm are that smoothing is done in curvature space, thus causing no shrinkage on the mesh in 3-D space; and that, in the scenario of object classification, it saves the computation of smoothing the mesh in 3-D space and obtaining the shape representation from the smoothed mesh.

The second stage of the classification algorithm is the computation of the scaled shape similarity metric. This metric effectively captures the overall shape of an object without being disturbed by shape details, thus making classification robust. The other advantage of this metric is its ease of computation.

The third stage of the classification algorithm is the generation of a prototype object for each class. This is done by using an inverse mapping technique for reconstructing a 3-D shape from its representation. The proposed classification algorithm has been tested using both synthetic and real data.

The preliminary results show the effectiveness of the approach on examples with two classes. Additional work is needed to extend the approach to a general classification technique for any arbitrary number of classes.

In the current implementation, the classification is performed by comparing the objects in the library to a single reference shape. A better approach would be to compare all pairs of objects in the library and to group pairs based on the scaled shape similarity metric. Another improvement is the automatic selection of the scaled shape similarity threshold which is used to decide whether an object falls into the reference object's class.

Acknowledgments

We thank Yoichi Sato and Mark Wheeler for providing some of the range images. Thanks to Marie Elm for proofreading this paper.

Reference

[1] E. Arkin, L. Chew, K. Huttenlocher, K. Kedem and J. Mitchell. An efficient computable metric for comparing polygonal shapes. IEEE Transaction on Pattern Analysis and Machine Intelligence, 13(3): pp. 209-215, 1991.

[2] 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.

[3] N. Burgess, M. N. Granieri and S. Patarnello. 3-D object classification: application of a constructive algorithm. International Journal of Neural Systems (Singapore), 2(4): pp. 275-282, 1991.

[4] J.M. Corridoni, A. Del Bimbo and L. Landi. 3D object classification using multi-object Kohonen networks. Pattern Recognition (UK), 29(6): pp. 919-935, June 1996.

[5] M. Hebert, K. Ikeuchi and H. Delingette. A spherical representation for recognition of free-form surfaces. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(7): 681-689, July 1995

[6] K. Higuchi, M. Hebert and K. Ikeuchi. Building 3-D models from unregistered range images. *CVGIP-Image Understanding*. Vol. 57. No. 4. July 1995..

[7] T. Lindeberg. Scale-space for discrete signals. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(3): 234-254, March 1990.

[8] D.G. Lowe. Organization of smooth images curves at multiple scales. International Journal of Computer Vision, 3:119-130, 1989.

[9] T.S. Newman, P. J. Flynn and A. K. Jain. Model-based classification of quadric surfaces. CVGIP, 58(2): pp. 235-249, 1993.

[10] J. Oliensis. Local reproducible smoothing without shrinkage. IEEE Transaction on Pattern Analysis and Machine Intelligence, 15(3): pp. 307-312, 1993.

[11] G. Sapiro and A. Tannenbaum. Area and length preserving geometric invariant scale-spaces. IEEE Transaction on Pattern Analysis and Machine Intelligence, 17(1): pp. 67-72, 1995.

[12] 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, 6(2): pp. 29-44, 1987.

[13] 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.

[14] H. Shum, M. Hebert and K. Ikeuchi. On 2shape similarity. Proc. CVPR'96, pp. 526-531. June 1996.

[15] H. Shum, M. Hebert and K. Ikeuchi. On 3D shape synthesis. In Object Representation in Computer Vision II. LNCS 1144. Springer-Verlag. April 1996.

[16] H. Shum. Modeling from reality: representation and integration. Ph.D. thesis, Carnegie Mellon University, July 1996.

[17] G. Taubin. Curve and surface smoothing without shrinkage. Proc. ICCV'95, pp. 852-857, June 1995.

[18] M. Wheeler and K. Ikeuchi. Iterative smoothed residuals: a low-pass filter for smoothing with controlled shrinkage. IEEE Transaction on Pattern Analysis and Machine Intelligence, 18(3): pp. 334-337, March, 1996.

[19] A.P. Witkin. Scale-space filtering approach to fair surface design. In Proceedings, 8th. International Joint Conference on Artificial Intelligence(IJCAI), pp. 1019-1022, Karlsruhe, Germany, August 1983.