A 3D Shape Classifier with Neural Network Supervision

Zhenbao Liu*, Jun Mitani, Yukio Fukui and Seiichi Nishihara
Department of Computer Science,
Graduate School of Systems and Information Engineering,
University of Tsukuba, Japan
E-mail: liuzhenbao@npal.cs.tsukuba.ac.jp, mitani@cs.tsukuba.ac.jp, fukui@cs.tsukuba.ac.jp, nishihara@cs.tsukuba.ac.jp
*Corresponding author

Abstract: The task of 3D shape classification is to assign a set of unordered shapes into pre-tagged classes with class labels, and find the most suitable class for a newly given shape. In this paper, we present a 3D shape classifier approach based on supervision of the learning of point spatial distributions. In this classifier, we first extract the low-level features by characterizing the point spatial density distributions of 3D shapes, and train one feedforward neural network to learn these features by examples. The Konstanz shape database was chosen as the test database, and we grouped the classified objects into two sets, the training set and the test set, which each had an approximately equal number of shapes. We trained the network with the training set, and evaluated the accuracy rate on the test set. We also compared this classifier to *k* nearest neighbors classifier for 3D shapes. This approach can be used to classify 3D shapes and enhance the performance of the existent 3D shape retrieval methods.

Keywords: 3D shape classification; 3D shape classifier; point spatial distributions; neural network supervision.

1 INTRODUCTION

3D data has been widely applied into many applications, such as computer aided design, computer games, virtual reality environments, computer vision and so on. The large number of 3D shapes makes it difficult to find them artificially. Accordingly, there is an increasing need to develop computer algorithms that enable us to find their interesting 3D shape data. 3D shape classification and retrieval has been a recent research focus currently, and researchers have endeavored to extract the features representing one 3D shape (Funkhouser et al., 2004) (Bustos et al., 2005). However, it is challenging to describe one shape by some mathematical functions. We know that it is very easy for human beings to recognize, match and retrieve 3D shapes. But this same task is very difficult for computers. Biology shows that the human brain has more than 10 billion neural cells with complicated interconnections. These neural cells constitute a huge network. Researchers have been attempting to simulate the mechanisms of the human network to accomplish the task of pattern recognition. This paper will discuss the possibility of applying one feedforwoard neural network to supervise 3D shape classification and retrieval.

We present a 3D shape classifier based on supervision of the learning of spatial density distributions of surface points. In this classifier, we first extract the low-level feature of 3D shape by characterizing the point spatial distributions in a bounding cube after pose estimation. One feedforward neural network is trained by the low-level features of 3D shape examples. Then, the network is used to classify some unknown shapes. We chose the Konstanz shape database (Websites 1.) (Bustos et al., 2005), grouped it into a training set and a test set, and trained the classifier through supervision of the training set and evaluated the performance of the classifier on the test set. (Bespalov et al., 2005) also presented several distinctive benchmark datasets for evaluating techniques for automated classification and retrieval of CAD objects. But these datasets includes only CAD objects with a small quantity, and is not adaptable for a training set and a test set.

The outline of the remainder of the paper is as follows. In the next section we briefly review the previous studies on

3D model classification and retrieval. In section 3 we describe the main structure of this classifier. Section 4 describes the technique to extract the low-level features of one 3D shape. Section 5 describes the design of one feedforward neural network. Section 6 gives the experiment results of the accuracy rate, and presents how to enhance the existent retrieval methods by using this classifier. Section 7 discusses the classifier and Section 8 concludes the study.

2 PREVIOUS WORK

In this section, we review recent 3D classification and retrieval methods with an emphasis on feature distribution methods. At the end of this section, we introduce other relevant developments.

The idea of feature distribution originates from statistics. It is fast and easy to statistically analyze the feature distribution of models, since the analysis does not involve any normalization of a 3D mesh-model and does not need any complex mathematic transformation. In an earlier well-known paper, (Ankerst et al., 1999a) used shape histograms to decompose shells or sectors around a model's centroid. (Osada et al., 2002) matched 3D models with shape distributions. The key idea behind this method is that it presents the signature of an object as a shape distribution sampled from a shape function that measures the global geometric properties of the object. (Liu et al., 2008a) utilized multiresolution wavelet analysis on shape orientation and the shape can be described by wavelet coefficients of each shape orientation function. (Liu et al., 2006) presented a novel 3D shape descriptor for effective shape matching and analysis that utilizes both local and global shape signatures. (Liu et al., 2008b) proposed a new 3D shape descriptor based on spherical healpix, and used the healpix structure to analyze the spherical extent function.

The above shape features has been designed for matching tasks, and can not applied directly to semantic classification. The task of classification is to assign a set of unordered shapes into pre-tagged classes with class labels, and find the most suitable class for a newly presented shape. In this sense, classification extends matching by assigning semantic meaning to shapes. One common 3D shape classifier is *k* nearest neighbors classifier (Ankerst et al., 1999b), which compare all the features and arrange *k* nearest 3D shapes for each given 3D shape, and assign it to the group with *k* labels. Another classifier proposed by (Barutcuoglu et al., 2006) is hierarchical classification based on a class hierarchy.

In this study, we defined a new classifier and evaluated the classification performance, and compared this classifier with *k* nearest neighbors classifier (Ankerst et al., 1999b). And we also discussed to use this classifier to enhance the typical existing feature distribution methods – shape histograms on shells (Ankerst et al., 1999a) and shape distributions (Osada et al., 2002) – the two typical methods mentioned above.

3 STRUCTURE OF THE CLASSIFIER

This classifier is divided into two stages. One is the training stage (Figure 1), which is executed under the supervision of training data. The other is the classification test stage (Figure 2) in which the performance of classification and retrieval will be evaluated.

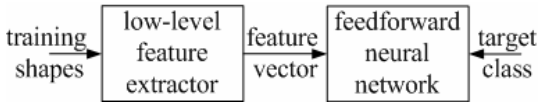

Figure 1 *Training stage*

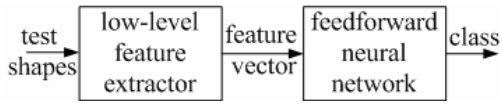

Figure 2 *Classification test stage*

4 LOW-LEVEL FEATURE EXTRACTOR

We propose the point spatial density distributions of one 3D shape as the low-level feature extractor. The distributions can be computed as described below and illustrated in the following flow. The detailed computation including pose estimation, sampling, and point distributions will be seen in the following subsection 4.1, 4.2 and 4.3 correspondingly.

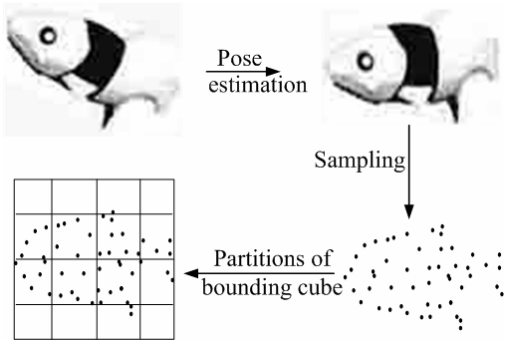

Figure 3 *Flow of low-level feature exaction*

4.1 Pose estimation

(Jolliffe, 1986) introduced the theory of PCA (Principal Component Analysis) method and (Chen et al., 002) applied this theory to obtain a rotation invariant measure of 3D shape. (Podolak et al., 2006) described a planar reflective symmetry transform that captures a continuous measure of the reflectional symmetry of a shape with respect to all possible planes, but this symmetry is not robust for the generic 3D shapes. Here we adopted the classical PCA method for pose estimation. As we know, the eigenvectors of PCA are called principal axes and describe the three orthogonal axes where the scattering of the elements is greatest. The eigenvector corresponding to the largest

eigenvalue coincides with the direction of largest variance of the 3D data set. The direction of the largest variance is usually regarded as the principal *X*-axis direction. A majority of vertices distribute along the direction. During the actual application, the multiresolution of one shape should be considered and the covariance matrix is approximated as follows.

$$C_I = \frac{1}{m}\sum_{i=1}^{m} S_i (g_i - m_I)(g_i - m_I)^T \quad (1)$$

where S_i and g_i is the area of a triangle of a shape, and the center of gravity respectively, and m_I is the center of gravity of a shape, and m is the number of triangles of the shape.

4.2 Random point sampling on the surface of a shape

Unbiased random points can be generated according to the surface area of a triangle on the surface. Here we use Monte-Carlo sampling approach (Madras, 2002, Osada, 2002). Firstly, compute the area of each triangle and store the cumulative area of triangles visited in an array. Secondly, generate a random number between 0 and the total cumulative area and perform a binary search on the array of cumulative area. The probability of finding a triangle is proportional to its area. Lastly, in each selected triangle with vertices (*A*, *B*, *C*), sample a point *P* with respect to the following procedure:

Generate two random float numbers, r_1 and r_2 between 0 and 1. Compute the *P* according to the following equation:

$$P = (1-\sqrt{r_1})A + \sqrt{r_1}(1-r_2)B + \sqrt{r_1}(r_2 C) \quad (2)$$

The square root of r_1 sets the percentage from vertex *A* to the opposite edge. r_2 sets the percentage along this edge. The consideration of taking the square root of r_1 is to get an unbiased random point with respect to surface area (see Figure 4).

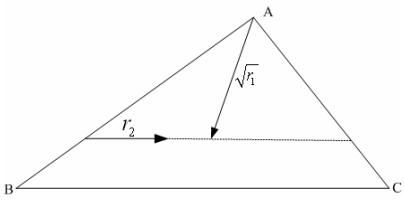

Figure 4 *Sampling a point in a triangle*

4.3 Computation of point spatial distributions

After pose normalization and random sampling, a shape can be represented as a set of points *S*. *S* includes *T* elements (points) in total. From these points, a bounding cube can be calculated. This computation is based on partitioning of the bounding cube. Subdivide the bounding cube to $N \times N \times N$ bins (partitions). With respect to this subdivision, accordingly, the set of points *S* is divided into $L = N \times N \times N$ subsets of points, S_1, S_2, … , S_L. The subset S_l includes all the points residing in the *l*-th bin. Here *L* will be defined as the dimension of input nodes in the following section.

One element V_l of the low-level feature vector *V* can be computed as the following.

$$V_l = N_l / L \quad (3)$$

where N_l is the number of elements in the subset S_l.

The point distributions are described by the low-level feature vector *V* representing the distribution computation of sampling points in the partitions of bounding cube.

5 DESIGN OF TRAINING NETWORK

We will design our training network (Figure 5) according to the basic feedforward neural network (Kishan et al., 1996). This network is composed of three layers, input layer, hidden layer and output layer. And every layer contains many units, and every unit of two layers is connected with weights. We define how the outputs of the neural network depend on the input layer of the input vector and network weights between every layer. The low-level feature vector is input as the input units and the number of input units equals the dimensionality of the low-level feature vector. The output units are designed as the same as the total number of the 3D shape classes. And the number of hidden units can vary for finding the best classification.

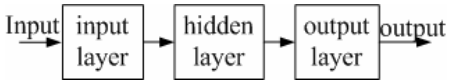

Figure 5 *Feedforward neural network*

5.1 Neuron model

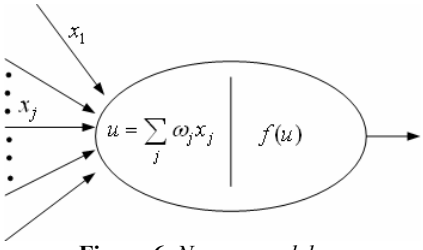

Figure 6 *Neuron model*

Each neuron (Figure 6) computes a weighted sum of its inputs, passes the sum *u* through its activation function *f*(*u*) and presents the result to the next layer. We use sigmoid function as the activation function showed below (Figure 7).

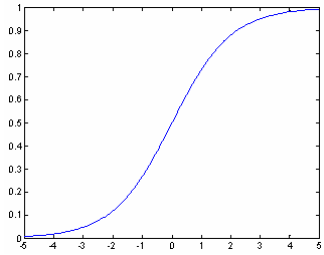

Figure 7 *Sigmoid function f*(*u*)

5.2 Forward computation

Before forward computation (Figure 8), the weights between the input layer and the hidden layer, and other

weights between the hidden layer and output layer are set to random float numbers belonging to [-1.0 1.0]. Pass the low-level feature V to the input units and start the computation.

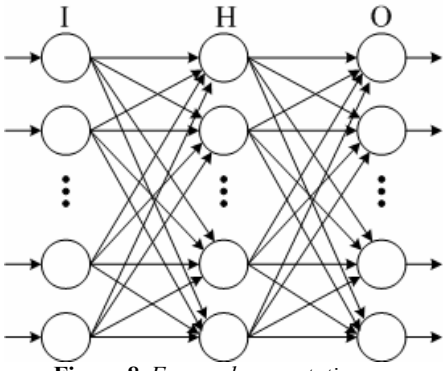

Figure 8 *Forward computation*

$$I_l = V_l \tag{4}$$

$$H_m = f(\sum_{l=1}^{L} \omega_{lm} I_l) \tag{5}$$

$$O_n = f(\sum_{m=1}^{M} \omega_{mn} H_m) \tag{6}$$

Here, V_l is one element of a low-level feature vector V, I_l is one input unit in the input layer, and there are L units in total in the input layer. H_m is the value of one hidden unit in the hidden layer, and there are M units in total in the hidden layer. O_n is one output unit in the output layer, and there are N units in the output layer. And $f(u)$ is the sigmoid function mentioned above.

And all the output units can be calculated in the forward process.

5.3 Error back-propagation method

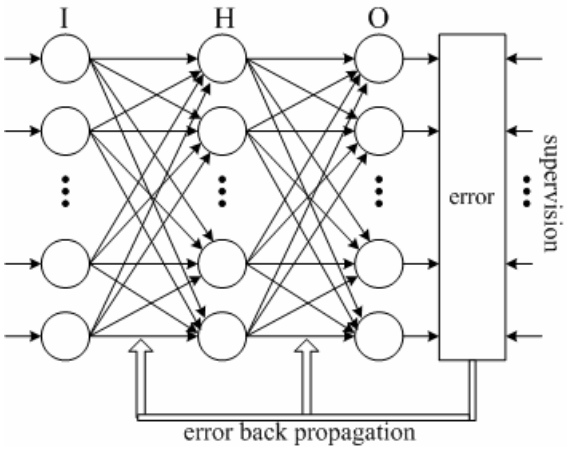

Figure 9 *Error back propagation*

After forward calculation, the error (Figure 9) exists between the supervision outputs (desired outputs) D_n and actual outputs O_n of the network. As for the i-th 3D shape, the error E_i can be computed as the following.

$$E_i = \sum_{n=1}^{N} (D_n - O_n)^2 \tag{7}$$

Integrate the formula (7) and (5), (6), and get (8):

$$E_i = \sum_{n=1}^{N} \{D_n - f(\sum_{m=1}^{M} \omega_{mn} f(\sum_{l=1}^{L} \omega_{lm} I_l))\} \tag{8}$$

We can see the error is the function of all the weights. We need to discover W, the vector consisting of all weights in the network, so that the value of Ei is minimized. One way to minimize the error is based on the gradient decent method. According to gradient descent, the direction of weight change of W should be in the same direction as the minus partial derivative. That is,

$$\Delta\omega \propto -\partial E / \partial \omega \tag{9}$$

We calculate all the partial derivatives of the weights, and then the delta value of each weight ω, lastly, modify the weights according to the delta value as the following.

$$\omega(t+1) = \omega(t) + \alpha \times \Delta\omega(t-1) + \eta \times \Delta\omega(t) \tag{10}$$

Here, α is the momentum, and η is the learning rate. t represents this training time, and t+1 represents the next training time need to change weights.

The error can converge to zero after the modification of each weight in many times. That time, we think the network has been entirely fit for these training examples.

5.4 Result output

All the weights can be frozen after supervision and can be used to compute output values for new input samples. The computation method is the same as the above sub-section "Forward computation".

6 EXPERIMENT RESULTS

In this section, we will analyze the experiments of this classifier, evaluate its classification performance, and discuss how to enhance the existent methods by using this classifier.

6.1 Evaluation means

In recent several years, Konstanz shape database has been widely used to test the classification and retrieval performance of 3D shapes. This database contains 472 classified objects. The classified objects are composed of 55 classes of shapes. We group the classified objects into two sets, the train set and test set in which the number of shapes are approximately equal. That is, the train set contains 55 classes, 248 shapes, and the test set also contains 55 classes, 224 shapes. See the Table 1 for detailed classes.

Class		
ants	jet-planes	axes
rabbits	fighter-planes	glasses
cows	propeller-planes	knives
dogs	other-planes	screws
fishes	zeppelins	spoons
bees	motorcycles	tables
CPUs	sport-cars	skulls
keyboards	cars	human-heads
cans	formula-cars	human-masks
bottles	galleons	books
bowls	submarines	watches
pots	warships	sand-clocks
cups	beds	swords
wine-glasses	chairs	barrels
teapots	office-chairs	birches
biplanes	sofas	flower-pots
helicopters	benches	trees
missiles	couches	weeds
		human-bodies

Table 1 *Shape classes in the database.*

The classification of the train set is used for teacher supervision signal of this neural network, and each shape is an example of the network. Each shape has a record of corresponding relation between input pattern and the correct classification. Each input pattern is the input low-level feature signal. The low-level signal can be computed by point spatial density distributions mentioned in section 4. The correct supervision classification information is a 55-dimension vector. We define that the value of the unit corresponding to the right class is 0.9, and the value of other units is 0.1 entirely, instead of 0 and 1, since sigmoid units can not produce 0 and 1 given finite weights. Now we give one simple example, one ant shape belongs to the first class – ant class, and the supervised classification vector is the following figure.

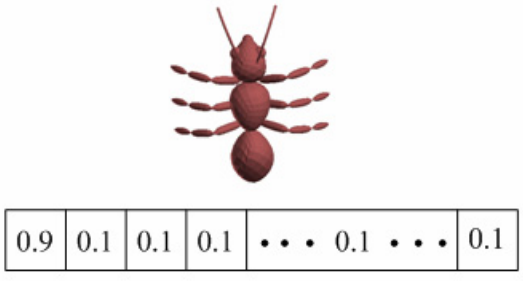

Figure 10 *Supervision classification vector*

6.2 Training process

We set the accuracy rate 90% and approximate 2000 running epochs on the training set as two stopping conditions. This process costs the time on samples drawn from all of the shapes. In the step of setting learning rate, we considered that the high learning rate could cause the learning algorithm to take large steps on the error function, with the risk of missing a minimum, or unstably oscillating across the error minimum. But the lower learning rate can also bring on the longer learning time. Therefore, one medium learning rate is set to 0.5. The momentum is set to 0.3.

We observed how the squared errors including Gross Squared Error (GSE) and Mean Squared Error (MSE) converged. The GSE and MSE are defined as the following, and N is the number of shapes.

$$GSE = \sum_{i=1}^{N} E_i \ , \ MSE = \sum_{i=1}^{N} E_i \Big/ N \qquad (11)$$

The following figures give the relation between GSE or MSE and epochs. We set the accuracy rate 90% on the training set as the stopping condition, but found that the classifier takes only approximate 1500 epochs to reach 95% accuracy on the training set.

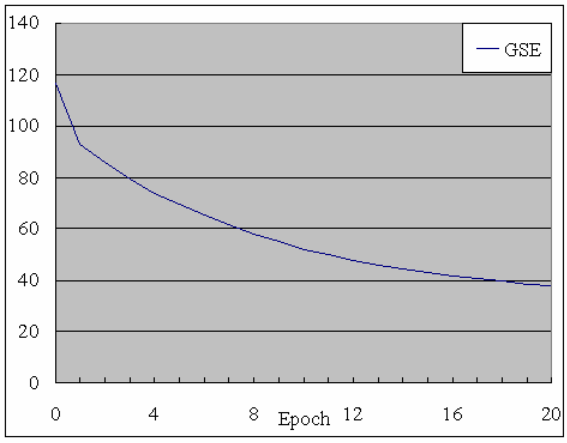

Figure 11 *Epoch (×100) and GSE*

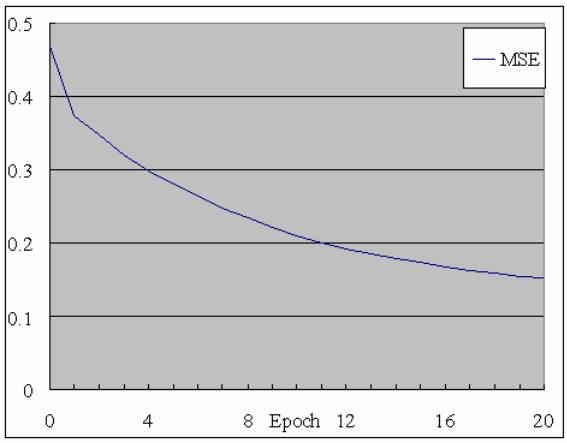

Figure 12 *Epoch (×100) and MSE*

6.3 Accuracy rate of classification

We measured the accuracy rate of classification on the test database, 55 classes, 224 shapes in total. If the output of this classifier is consistent with supervision classification, this classification is thought right. The number R of right classifications is collected statistics. The accuracy rate of classification is the following.

$$accuracy = R/N \qquad (12)$$

where the accuracy represents the accuracy rate and N is the total number of shapes or classification items.

After some experiments, we found the accuracy rate of this classifier could reach 53.6%.

In the experiment, we also viewed that the accuracy of classification had close relations with the number of hidden units in the hidden layer of the neural network. We investigated this relation by setting variable hidden units. Please see the Figure 13.

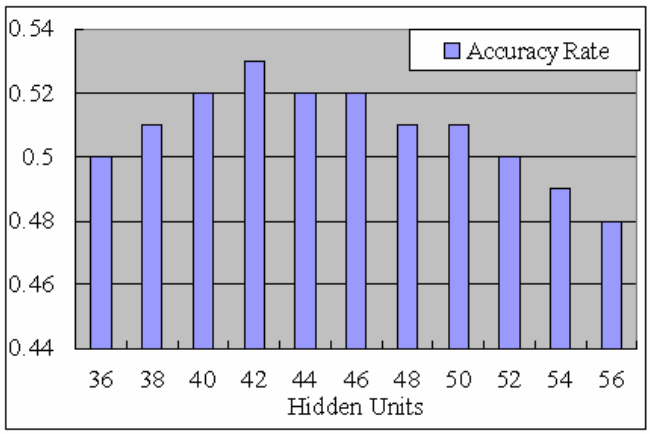

Figure 13 *Relation between hidden units and accuracy rate.*

Lastly, we set the number of the hidden units to 42 to get the best classification result.

We show several examples of our classification results on the test shape database. Firstly, we chose several classes of 3D shapes including human heads, dogs, and helicopters. From any class, input each shape into the classifier, the following table represents the right classification for these input shapes.

Human head Class			
			
Dog Class			
			
Helicopter Class			
			

Table 2 *The examples of classification.*

6.4 Comparison with *k* nearest neighbors classifier

We compared this classifier with *k* nearest neighbors classifier (Ankerst et al., 1999b) on this Konstanz shape database, and used the same feature as this classifier. We measured the accuracy rate of classification on the test database, 55 classes, 224 shapes, in total. The next table gives the result of comparison.

classifier	**accuracy rate**
k nearest neighbors classifier	33.5%
our classifier	53.6%

Table 3 *Comparison between two classifiers.*

6.5 Enhance the existent methods

This sub-sector will discuss how to use this classifier to enhance the existent methods.

We suppose that the dissimilarity of two shapes defined in the existent methods is D_{old}, and the dissimilarity after enhancement is D_{new}. We consider the output supplied by the classifier is one class-number-dimension vector O. The output vector of the one shape in the classifier is O_1, and the other is O_2. The dissimilarity of classifier output between two shapes is D_{class}.

$$D_{class} = \|O_1 - O_2\| \tag{13}$$

We multiply the dissimilarity of the classifier output with one constant weight, and add this impact to the dissimilarity on the existent method, then form the new dissimilarity.

$$D_{new} = D_{old} + c \cdot D_{class} \tag{14}$$

The constant weight c can be designed by making a concrete analysis of concrete problems. And the output vector O can also be filtered by one high-pass filter because we find some output vectors include low-value noise.

Here, we tested the performance of enhancing the existent methods. We chose methods (6.5.1) and (6.5.2), programmed and implemented the two algorithms again, and compared the existent methods with the enhanced version on the same conditions.

We chose three parameters to evaluate the performance, the Recall-Precision (R-P) Curves, the Average Recall Precision (R-P), and Enhancement Rate (ER).

Recall-Precision Curves have been used extensively in 3D retrieval methods. The precision is defined as the fraction of the retrieved objects relevant to the input query, and the recall is given by the fraction of the same kind of objects retrieved from the test database.

The average recall-precision means averaging all the precision values.

The last parameter is defined by us, that is, the enhancement rate (ER).

$$ER = (ARP_{new} - ARP_{old}) / ARP_{old} \tag{15}$$

Here, ARP_{new} is the average recall precision of new enhanced version, and ARP_{old} is the average recall precision of the existent methods.

6.5.1 Shape Histograms on shells (SH) (Ankerst et al., 1999a)**.**

In this method (Figure 14), the 3D shape can be described as histograms of point fractions belonging to different partitioning shape shells.

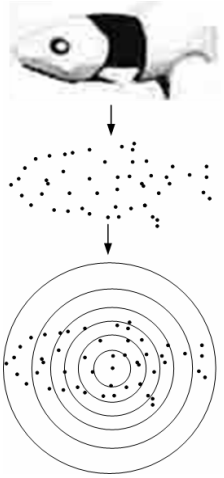

Figure 14 *The flow of shape histograms method.*

We enhanced the retrieval performance of shape histograms under the assistance of our classifier. The plot of recall precision and the table of average recall precision show the result of enhancement.

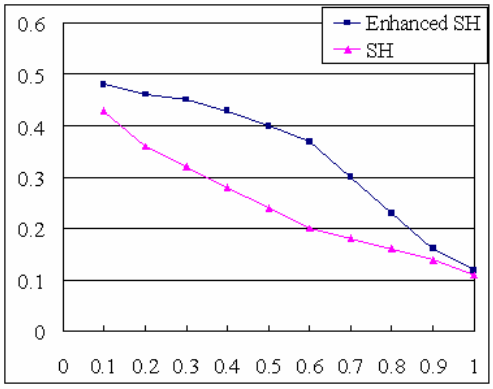

Figure 15 *The effect of enhancement on R-P curves*

Methods	Average Recall Precision
SH	0.242
Enhanced SH	0.34

Table 4 *The effect of enhancement on average R-P.*

From the above recall precision curves and average values we found that the present classifier improved retrieval performance and computed the quantity number of the enhancement, and the enhancement rate is the following.

$$ER = 40\%$$

6.5.2 Shape Distribution (SD) (Osada et al., 2002).

This algorithm (Figure 16) sampled the probability distribution from two random points on the shape surface, and computed a histogram of distances between these pairs of points. The similarity between two shapes can be measured from the distribution.

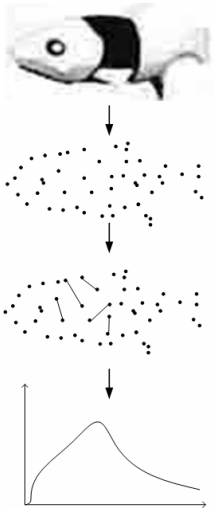

Figure 16 *The flow of shape distributions method.*

We enhanced the retrieval performance of shape distributions under the assistance of our classifier. The plot of recall precision and the table of average recall precision show the result of enhancement.

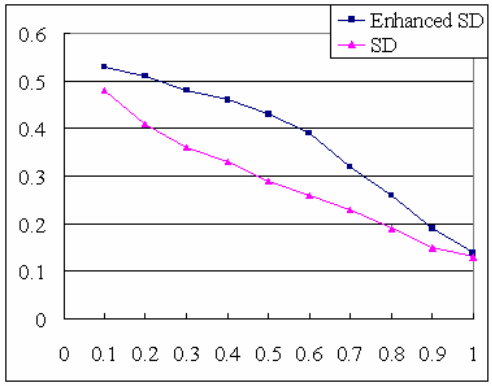

Figure 17 *The effect of enhancement on R-P curves.*

Methods	Average Recall Precision
SD	0.283
Enhanced SD	0.385

Table 5 *The effect of enhancement on average R-P.*

The enhancement rate is the following.

$$ER = 36\%$$

7 DISCUSSION

Based on the test of this classifier on the database, we will discuss some influences. We believe that this classifier has several advantages. At the same time, we discuss the shortfalls of the classification algorithm.

7.1 Influence of resolution of sampling points

Firstly, we tested the several resolutions of sampling points in the stage of low-level feature extractor, and hoped to settle on the influence of sampling resolutions on classification rate. We computed several resolutions of sampling:

$$S_1 = 128*128 = 16384$$

$$S_2 = 64*64 = 4096$$

$$S_3 = 32*32 = 1024$$

And we computed the classification rates on the test set showed in the Table 6. Each accuracy rate of classification corresponds with each resolution.

Resolution	Accuracy Rate
S_1	53.6%
S_2	51.3%
S_3	46.4%

Table 6 *Influence of sampling resolutions.*

From the above Table 6, we could find the relation of accuracy rates with resolutions. When the sampling resolution rises, the accuracy rate of classification is improved. When the sampling rate decreases to 1024, the accuracy rate falls off approximately 7% from 53.6% to 46.4%. This table shows that the high sampling resolution leads to the improvement of accuracy rate of this classifier. However, if the sampling resolution rises to 65536, the need of the sampling time will have a heavy effect on the performance of this classifier. Therefore, we think the sampling rate S_1 has a better balance between classification rate and time.

7.2 Influence of shape normalization

In the stage of pose estimation or shape normalization, the orientation of 3D shape must be determined in advance of the computation of point spatial distributions according to the flow of low-level feature extractor. Here we adopted the Principal Component Analysis as the analysis tool, and computed the direction of the largest variance as the main direction. But this method caused one problem that the orientation of shape may be misunderstood, and two orientations of two similar shapes are obviously different. For example, the following figure presents this problem.

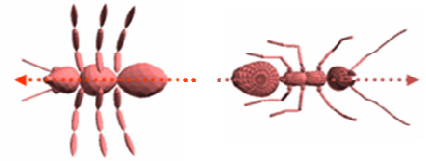

Figure 18 *Orientation mistake in shape normalization*

In the Figure 18, there are two 3D shapes, ants. After shape normalization by Principal Component Analysis, the orientation of each 3D shape is adjusted to the normalized direction. But two shapes have different directions, the left ant with the main direction pointing at the left side, and the right ant with the main direction pointing at the right side. And this error will go into the computation of point distributions, and affect the classification accuracy finally.

After orientation adjusting, the scale factor will be considered because many shapes have not the same scale, for example, small fish and big fish. Here we scaled each shape into unit size of average distances according to the triangle area and distances between the vertices and mass centre. The detailed scale factor s_f is the following.

$$s_f = \frac{1}{f_S} \sum_{i=1}^{n} f_i r_i \qquad (16)$$

Here, f_i is the area of the i-th triangle of 3D shape, and r_i is the mass centre of i-th triangle, and f_S is the mass centre of 3D shape. And each vertex will be scaled according to this scale factor. We know this scale factor will be influenced by triangle area, and low resolution with big triangles will bring in the error of scale factor. And scale error will cause one problem that this classifier cannot compute the point distributions in the bounding cube precisely. And the neural network may classify this shape into one wrong class.

7.3 Merits of the algorithm

1) We discussed one approach to apply one feedforward neural network to classify the 3D shapes. This approach can make good use of examples of man-made classification, and learn the classification mode from human. And this classifier also supplied better classification accuracy than k nearest neighbor classifier.

2) This method has extensible property because the training knowledge can be supplemented by humans step by step, and the performance can also be advanced step by step.

3) This algorithm can enhance the existent 3D shape retrieval methods, and also improve the retrieval performance under supervision of knowledge learning.

7.4 Drawback of the algorithm

The accuracy rate of classification is not higher, and the ideal or desired rate should be about 70% or higher. There are perhaps three reasons.

The main reason is that the training is not enough, and then the neural network cannot recognize some 3D shapes accurately. It is clear that a neural network is useless if it only sees fewer examples of a matching input/output pair, and it cannot infer the characteristics of the input data. This is similar as a child memorizes some animals with different types. In advance, this child needs to observe several examples of each type of animal so that he can classify one new animal of the observed type. After this child is able to distinguish fish from bird and bugs, he also need recognize more fishes because he cannot classify some types of fish, for example, carp, tuna, and whale. It is difficult to require that a child could remember the distinctions between types of animal after seeing them only once. This means that this child must repeat to observe these animals until the necessary information sinks in. The training set we used is

also faced with this problem, and it is not also big and we need one bigger training set and more training shapes. Every desired class needs 100 3D shapes or more, and enough shapes can ensure that the neural network learns more knowledge and does the more right judgments in the test process. This factor impacted the accuracy rate of classification and the performance of retrieval.

Another reason is perhaps that the proposed low-level feature is not discriminative, and brings the neural network the difficulty of distinguishing two shapes. We will try more discriminative feature descriptor as the training subject.

The last reason we must consider is that the error back propagation method adopted the "steepest down" or called gradient descent, which can lead to the local minima of the error function. It is difficult to find the global minimum of the error function, and therefore, the probability of running into the local minima is not low. This drawback of "steepest down" error back propagation brings one problem that the classification system cannot get the best performance. The following figure shows this problem. W and E represent the connection weights and error, and the point P is the start point, and LM and GM are local minimum and global minimum relatively.

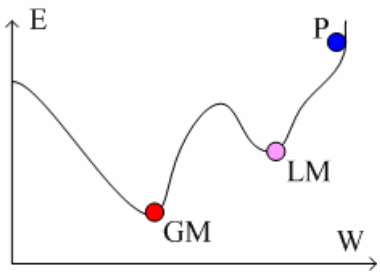

Figure 19 *local minimum and global minimum*

8 CONCLUSION AND FUTURE WORK

In this paper, a 3D shape classifier approach based on supervision of the learning of point spatial distributions is presented. In this classifier, firstly, the low-level feature of 3D shape is extracted, and then we train one feedforward neural network to learn this feature by supervision examples. We tested this classifier on the Konstanz shape database, and evaluated its accuracy rate, compared the classification rate to *k* nearest neighbors classifier for 3D shape, and also analyzed the efficiency of enhancing the existent methods by using this classifier. It can be used to classify 3D shapes and enhance the 3D shape retrieval performance.

In the future, we plan to improve two aspects. Firstly, consider researching more adaptable low-level feature to let classifier have a rise on the classification rate. Secondly, we intend to adopt one better method to avoid the local minimum of error function, for example, Boltzmann machine (Ackley et al., 1985).

REFERENCES

Ackley, D., Hinton, G., and Sejnowski, T. (1985) 'A Learning Algorithm for Boltzmann Machines'. *Cognitive Science*, Vol.9, No.1.

Ankerst, M., Kastenmüller, G., Kriegel, H.P., and Seidl, T. (1999a) '3D shape histograms for similarity search and classification in spatial databases', *Proceedings of 6th International Symposium on Spatial Databases* (*SSD'99*), Springer-Verlag, London, UK.

Ankerst M, Kastenmuller G, Kriegel H.P, Seidl T. (1999b) 'Nearest Neighbor Classification in 3D Protein Databases', *Proceedings of 7th International Conference on Intelligent Systems for Molecular Biology*, American Association for Artificial Intelligence.

Barutcuoglu Z. and DeCoro C. (2006) 'Hierarchical Shape Classification Using Bayesian Aggregation'. *Proceedings of IEEE International Conference on Shape Modeling and Applications* (*SMI'06*).

Bespalov, D., YiuIp, C., C.Regli, W. and Shaffer, J. (2005) 'Benchmarking CAD search techniques', *Proceedings of the ACM symposium on Solid and Physical Modeling*.

Bustos, B., Keim, A.D., Saupe, D., Schreck, T., and Vranic, D.V. (2005) 'Feature-based similarity search in 3D object databases', *ACM Computing Surveys*, Vol. 37, No.4.

Chen, D.Y. and Ouhyoung, M. (2002) 'A 3D Model Alignment and Retrieval System', *Proceedings of International Computer Symposium*.

Funkhouser, T. and Kazhdan, M. (2004) 'Shape-based retrieval and analysis of 3D models', *ACM SIGGRAPH* Courses.

Jolliffe, I.T. (1986) 'Principal Component Analysis', *Springer*.

Kishan M., Chilukuri K. M., and Sanjay R. (1996) 'Elements of Artificial Neural Networks', *MIT Press,* Cambridge, Massachusetts.

Liu, Z., Mitani, J., Fukui, Y. and Nishihara, S. (2008a) 'Multiresolution wavelet analysis of shape orientation for 3D shape retrieval', *ACM International Conference on Multimedia Information Retrieval*, Proceedings of ACM SIGMM.

Liu, Y., Zha, H. and Qin, H. (2006) 'The generalized shape distributions for shape matching and analysis', *Proceedings of IEEE International Conference on Shape Modeling and Applications* (*SMI'06*).

Liu, Z., Mitani, J., Fukui, Y., and Nishihara, S. (2008b) 'A new 3D shape retrieval method using spherical healpix', *Journal of Information Processing*, Vol.16.

Madras, N. (2002) 'Lectures on Monte Carlo Methods', *American Mathematical Society*.

Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D. (2002) 'Shape distributions', *ACM Transactions on Graphics*, Vol.21, No.4.

Podolak, J., Shilane, P., Golovinskiy, A., Rusinkiewicz, S. and Funkhouser, T. (2006) 'A Planar-Reflective Symmetry Transform for 3D Shapes', *ACM Transactions on Graphics* (*SIGGRAPH2006*), Vol.25, No.3.

WEBSITES

1. Konstanz Shape Database, http://merkur01.inf.uni-konstanz.de/CCCC.

Biographical notes: **Zhenbao Liu** received his Master and Bachelor degrees from Northwestern Polytechnical University, China, in 2004 and 2001, respectively. He is now pursuing a doctor's degree at Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba, Japan. His research interests include computer graphics, shape processing, and data retrieval.

Jun Mitani received his Master and PhD degrees from University of Tokyo, Japan, in 2000 and 2004 respectively. He joined Riken as a researcher in 2004. He is currently a lecturer at Department of Computer Science, University of Tsukuba, Japan. His research interests include computer graphics, design aid of paper model, origami. He is also a Presto researcher of Japan Science and Technology Agency. He is a member of Japan Society of Graphics Science, Society for Art and Science of Japan.

Yukio Fukui received his Doctor, Master and Bachelor degrees from University of Tokyo, University of Tokyo, and Kyoto University in 1993, 1980 and 1973, respectively. After graduation of master, he joined Agency of Industrial Science and Technology, Ministry of International Trade and Industry. He served as a research director in National Institute of Bioscience and Human-Technology from 1993. He is currently a professor at Department of Computer Science, University of Tsukuba from 1998. His research interests include shape processing, optical CAD system, shape design system, virtual reality. He is a member of Information Processing Society of Japan, Virtual Reality Society of Japan, Human Interface Society, Japan Ergonomics Society, Japanese Society of Ophthalmological Optics, and ACM.

Seiichi Nishihara graduated from Department of Mathematics and Engineering, Kyoto University, Japan, in 1968. And in the same year, he served as an assistant in Center of Large Scale Computer Systems, Kyoto University, Japan. He is currently a professor of Department of Computer Science, University of Tsukuba. His research interests include constraint satisfaction problem, production of virtual city. He is a member of Information Processing Society of Japan, Japanese Society for Artificial Intelligence. He is also the president of Society for Art and Science of Japan.