Efficient Recognition of Highly Similar 3D Objects in Range Images

Hui Chen and
Bir Bhanu, *Fellow*, *IEEE*

Abstract—Most existing work in 3D object recognition in computer vision has been on recognizing dissimilar objects using a small database. For rapid indexing and recognition of highly similar objects, this paper proposes a novel method which combines the feature embedding for the fast retrieval of surface descriptors, novel similarity measures for correspondence, and a support vector machine-based learning technique for ranking the hypotheses. The local surface patch representation is used to find the correspondences between a model-test pair. Due to its high dimensionality, an embedding algorithm is used that maps the feature vectors to a low-dimensional space where distance relationships are preserved. By searching the nearest neighbors in low dimensions, the similarity between a model-test pair is computed using the novel features. The similarities for all model-test pairs are ranked using the learning algorithm to generate a short list of candidate models for verification. The verification is performed by aligning a model with the test object. The experimental results, on the University of Notre Dame data set (302 subjects with 604 images) and the University of California at Riverside data set (155 subjects with 902 images) which contain 3D human ears, are presented and compared with the geometric hashing technique to demonstrate the efficiency and effectiveness of the proposed approach.

Index Terms—3D ear indexing, 3D ear recognition, biometrics, ear databases, feature embedding, rank learning, local surface patch representation.

1 Introduction

THREE-DIMENSIONAL object recognition is an important research field of computer vision. In this paper, we discuss the problem of efficient recognition of highly similar 3D objects in range images using indexing techniques. Various techniques have been proposed for 3D object recognition and indexing, for instance, geometric hashing and surface descriptor matching [4]. However, most of the research has focused on the recognition of 3D dissimilar objects using a small database. It is desirable to design a scalable and efficient 3D object recognition system.

In this paper, we present a new framework which handles the recognition of highly similar 3D objects with a good scalability performance on large databases. We use our local surface patch (LSP) descriptor, which has been shown to be more effective and efficient than the popular spin image representation [6]. We develop an efficient framework based on the LSP representation, but any other representation, such as the spin image, can be used. The core component of an LSP descriptor is a 2D histogram, whose dimensionality is large (in hundreds). Search of the closest LSPs in a high-dimensional space is time consuming. Further, most of the current 3D object recognition systems identify objects by matching a test object to every model object. This is definitely not efficient. As a result, the geometric hashing types of techniques are used. We present an approach that combines the feature embedding for the fast retrieval of surface descriptors and an SVM-based technique for ranking the hypotheses to generate a short list for the verification.

- *H. Chen is with the Motorola Biometrics Business Unit, 1250 N. Tustin Ave., Anaheim, CA 92807. E-mail: hui.chen002@gmail.com.*
- *B. Bhanu is with the Center for Research in Intelligent Systems, University of California, Riverside, CA 92521. E-mail: bhanu@cris.ucr.edu.*

Manuscript received 16 Sept. 2007; revised 6 June 2008; accepted 30 June 2008; published online 8 July 2008.
Recommended for acceptance by L. Van Gool.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-09-0604.
Digital Object Identifier no. 10.1109/TPAMI.2008.176.

2 Related Work and Contributions

2.1 Related Work

Campbell and Flynn [4] provided a survey on 3D free-form object recognition. There exists a large amount of work on 3D face recognition in range images, but it does not consider indexing. In this paper, we are focused on 3D object recognition using indexing techniques and the related work is summarized in Table 1. Geometric hashing has been a popular technique used for generating the hypotheses for 3D object recognition and fingerprint recognition [3], [7], [14], [18], [19]. However, for 3D object recognition, experiments on a small data set ($\sim$20 objects) of dissimilar objects are performed and the time and space complexity of hashing is polynomial in the number of feature points.

2.2 Contributions

The main contributions of this paper are as follows: 1) A novel computational framework that integrates feature embedding and rank learning for efficient recognition of highly similar 3D objects is presented. This innovative combination with novel features and associated similarity measures solves the object recognition problem with the integrated indexing in a systematic way. There exists no paper in the computer vision field on indexing using 3D data that used highly similar objects like the human faces or the human ears, as shown in Figs. 3 and 4. The paper [13] used ground vehicles (sedans, sport utility vehicles, jeeps and wagons, minivans, buses and vans, construction vehicles, trucks and pickups, and military vehicles). As compared to human ears, these ground vehicles are quite distinct from each other. Our approach is general and applicable to other data sets in computer vision. 2) The grouping algorithm based on geometric constraints clusters the correspondences and then new features, devised to measure the similarity of correspondences, are computed to rank hypotheses using the SVM learning technique. 3) Extensive experiments on two large public data sets (155 subjects with 902 images and 302 subjects with 604 images) of highly similar 3D objects are presented and compared with the geometric hashing to show the effectiveness of the approach.

3 Technical Approach

The system diagram is illustrated in Fig. 1. Given a model object, we extract the feature points that are defined as either the local minimum or the local maximum of shape index values. Then, we calculate LSP descriptors for the feature points and their neighbors. An "LSP" is defined as the region consisting of a feature point and its neighbors. The LSP representation includes a feature point, its surface type (convex/concave/saddle), the centroid of the patch, and a 2D histogram of shape index values versus dot product of the surface normal at the feature point and its neighbors [6]. Based on the surface type of an LSP, an LSP is classified into three types (convex/concave/saddle). For each type of LSP, we apply a feature embedding algorithm to embed the original feature vector (the 2D histogram of an LSP, concatenated as a feature vector) into a low-dimensional space such that the distance relationships are preserved. The K-d tree structure is used to perform the search in the low-dimensional space. Given a test image, we repeat the same procedures to map LSPs into the corresponding low-dimensional embedded space based on its surface type. By searching the nearest neighbors of the embedded feature vectors, we find the potential corresponding LSPs between a model-test pair. The initial correspondences are filtered and grouped to remove false correspondences using geometric constraints. Based on the set of

TABLE 1
Three-Dimensional Object Recognition from Range Images Using Indexing Techniques

Authors	Technique	Database
Yi and Chelberg [23]	Bayesian framework to achieve efficient indexing.	20 objects
Johnson and Hebert [12]	Used the principal component analysis (PCA) to compress the surface descriptors, the spin images.	20 objects
Matei et al. [13]	Combined the locality sensitive hashing [9] for approximate nearest neighbor search and the joint 3D-signature estimation for generation and evaluation of alignment hypotheses.	Two non-publicly available vehicle databases (89 and 366 objects)
Mokhtarian et al. [14]	Used geometric hashing for the generation of hypotheses.	20 objects
Stein and Medioni [18]	Used geometric hashing for the generation of hypotheses.	9 objects
Chen and Bhanu [5]	Used geometric hashing for the generation of hypotheses.	9 objects
Muller et al. [15]	Incorporated spatio-temporal invariance into the geometric features and proposed efficient indexing methods.	Synthetic motion capture data (Not the range data)
This paper	A novel method that combines the feature embedding, local structural similarity measures and rank learning techniques.	Two public databases of highly similar objects (155 and 302 objects)

correspondences, a set of features is computed to measure the similarity between a model-test pair. Then, the hypotheses are ranked using the SVM rank learning algorithm to generate a short list of candidate models for verification. The parameters of the SVM classifier are learned on a subset of the database. For verification, we perform surface matching by applying the Iterative Closest Point (ICP) algorithm in which the initial transformation is obtained from the corresponding LSPs.

3.1 Local Surface Patch Representation

We use our LSP representation as the surface descriptor. The LSP descriptor has been shown to be effective and distinctive for recognizing 3D similar objects [6]. An LSP is described by a 2D histogram, surface type, and the centroid. The 2D histogram and surface type are used for comparison of LSPs and the centroid is used for computing the rigid transformation. The patch encodes the geometric information of a local surface.

Since the LSP representation is described by a histogram, the χ^2-divergence and Earth Movers Distance (EMD) [17] are two proper distances. However, the χ^2-divergence is nonmetric and EMD is computationally expensive, so we choose euclidean distance to measure the distance between two descriptors.

3.2 Feature Embedding

Given a query feature vector in a high-dimensional space, searching its closest matches in a large database is time consuming.

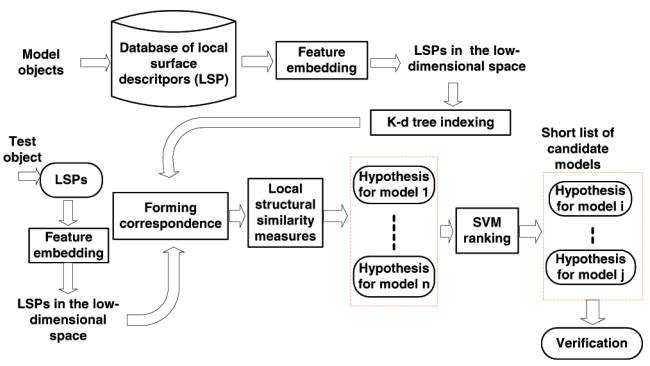

Fig. 1. System diagram for indexing and recognition of highly similar 3D objects.

Various methods have been proposed to speed up the nearest-neighbor retrieval, including hashing and tree structures. However, the complexity of these methods grows exponentially with the increasing dimensionality. In recent years, a number of approaches which *embed* feature vectors from a high-dimensional space into a low-dimensional space have been proposed [1], [8], [10], [16], [20], [21], [24]. Multidimensional scaling (MDS) [24], LLE [16], and ISOMAP [20] cannot handle online query efficiently. Lipschitz embedding [10], FastMap [8], MetricMap [21], and BoostMap [1] can handle online query efficiently.

As compared to the above algorithms, which can handle online query, the *FastMap* embedding algorithm has the following attractive advantages: 1) It only needs $O(Nk)$ distance calculations for the offline embedding in which N is the number of feature vectors and k is the dimensionality of the embedded space. 2) Given a query feature vector, it only takes $O(k)$ distance calculations to map it into the k-dimensional space. 3) It makes no assumption about data distributions.

The FastMap algorithm is used in this paper to map the high-dimensional LSP feature vectors to a low-dimensional space where the distance relationships are preserved. For the FastMap algorithm, a key question is how to choose parameter k, the dimensionality of the embedded space. In this paper, we use a *Stress* function to guide the choice of k. The *Stress* function, a measure of the goodness-of-fit, is defined as $S = \sqrt{\frac{(d'_{ij}-d_{ij})^2}{\sum_{ij} d_{ij}^2}}$, where d_{ij} is the distance between objects i and j in the original space and d'_{ij} is the distance in the embedded space. Once the embedding has been obtained, the actual nearest-neighbor search is performed in the low-dimensional embedded space. In our case, the local surface descriptor has three different types (convex/concave/saddle) based on the shape index value of the feature point. For each type of local surface descriptors, the embedding algorithm is run to map the original feature vector into a low-dimensional feature space and the K-d tree structure is used to build the index in the low-dimensional space. Even though some of the LSP descriptors may map to the same point in the low-dimensional space, it will not affect the recognition performance since the correspondences are grouped and filtered out using geometric constraints, as described below.

3.3 Forming Correspondences

Given a test image, we extract feature points and compute the LSP descriptors. Then, every descriptor is embedded into a

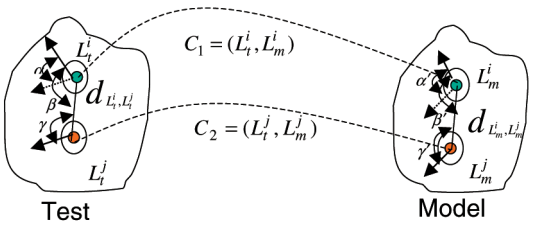

Fig. 2. Geometric constraints for grouping LSPs.

low-dimensional space based on its types and the similar LSPs are retrieved efficiently using the K-d tree structure. The embedding algorithm introduces some errors in that the closest LSPs in the original space may not be the closest in the embedded space. This problem is alleviated by returning a set of nearest neighbors and using the geometric constraints to group the correspondences. The potential corresponding LSP pairs are filtered and grouped based on the geometric constraints of (1), which are illustrated in Fig. 2:

$$\begin{aligned} & d_{C_1,C_2} = |d_{L_t^i,L_t^j} - d_{L_m^i,L_m^j}| < \epsilon_1; \quad max(d_{L_t^i,L_t^j}, d_{L_m^i,L_m^j}) > \epsilon_2; \\ & (|\alpha - \alpha'|, |\beta - \beta', |\gamma - \gamma'|) < \epsilon_3, \end{aligned} \tag{1}$$

where $d_{L_t^i,L_t^j}$ and $d_{L_m^i,L_m^j}$ are the euclidean distances between the centroids of the two surface patches. In the experiments ϵ_1, ϵ_2, and ϵ_3 are 9.4 mm, 3.7 mm, and 30 degrees, respectively. α is the angle between the surface normals at the feature points of the two surface patches (L_t^i, L_t^j), β is the angle between the surface normal of the patch L_t^i and the line connected by the centroids of the two patches (L_t^i, L_t^j), and γ is the angle between the surface normal of the patch L_t^j and the line connected by the centroids of the two patches (L_t^i, L_t^j). α', β', and γ' are defined in the same way. The first distance constraint and the three orientation constraints guarantee that the two corresponding pairs (L_t^i, L_t^j) and (L_m^i, L_m^j) are consistent; the second constraint removes the correspondences which are too close. We use these geometric constraints to partition the potential corresponding pairs into different groups. The larger the group is, the more likely it is that it contains the true corresponding pairs. Given a list of corresponding pairs, the grouping procedure for every pair in the list is as follows: Initialize each pair of a group. For every group, add other pairs to it if they satisfy (1), repeat the same procedure for every group, sort the groups in descending order based on the size of groups, and select the groups on the top of the list.

3.4 Computing Similarities

Once we find the n corresponding pairs $\{L_m^i, L_t^i\}$, $i = 1, 2, \cdots, n$, where L_t^i is the ith local surface descriptor in the test object and L_m^i is its corresponding descriptor in the model m, we compute seven novel features proposed in this paper to measure the similarity between them. The seven features are computed based on (2). In (2), N_t is the number of local surface descriptors in the scene, T is the rigid transformation obtained from the n correspondences which aligns the model and test, $\bullet$ denotes the dot product, $\rho = \frac{2}{n(n-1)}$, $d(\cdot)$ is the euclidean distance between the two 3D coordinates, $f_c(L_i)$ gets the 3D coordinates of the LSP L_i, and f_n gets the surface normal vector of the LSP L_i. The *ratio* counts the fraction of local surface descriptors in the scene which find the correspondences, e_1 is the registration error, e_2 is the average pairwise distance between the corresponding LSPs, e_3 measures the average distance between the surface normal vectors of a corresponding LSP pair, and e_4, e_5, and e_6 are the pairwise angle difference between the corresponding LSPs:

$$\begin{aligned} & \text{ratio} = \frac{n}{N_t}; \quad e_1 = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(f_c(L_t^i) - T\left(f_c\left(L_m^i\right)\right)\right)^2}, \\ & e_2 = \rho\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\left(d(f_c(L_t^i), f_c(L_t^j)) - d\left(f_c\left(L_m^i\right), f_c\left(L_m^j\right)\right)\right)^2, \\ & e_3 = \frac{1}{n}\sum_{i=1}^{n}\left(f_n\left(L_t^i\right) \bullet T\left(f_n\left(L_m^i\right)\right)\right); \quad e_4 = \rho\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\left(\left|\alpha_{ij} - \alpha'_{ij}\right|\right), \\ & e_5 = \rho\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\left(\left|\beta_{ij} - \beta'_{ij}\right|\right); \quad e_6 = \rho\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\left(\left|\gamma_{ij} - \gamma'_{ij}\right|\right). \end{aligned} \tag{2}$$

3.5 Ranking the Hypotheses Using SVM

In [13], a posterior probability of a model, given the scene surface descriptors, model surface descriptors, and the alignment parameter, is computed to rank the candidate models. In order to compute this probability, several unrealistic assumptions are made, for instance, the uniform distribution of a model, the independence of the scene surface descriptors, and the Gaussian distribution of the residuals. These assumptions may not hold in the real data. In our case, given a test object, we compute a set of features as a measure of the similarity for every model in the database. We rank the candidate models in descending order based on these features without making any assumptions. We learn the ranking function, which makes use of the two advantages of SVM, "large-margin" and "kernel trick," for supporting the nonlinear ranking [11].

The problem of ranking is formalized as follows: For a query q and a collection of data $D = \{d_1, d_2, \cdots, d_m\}$, we say $d_i < rd_j$ or $(d_i, d_j) \in r$ if d_i is ranked higher than d_j for an ordering r. The ranking function f can be learned from the training data. Assume the ranking function is linear such that: $(d_i, d_j) \in f_{\mathbf{w}(q)} \Longleftrightarrow \mathbf{w}^T\phi(q, d_i) > \mathbf{w}^T\phi(q, d_j)$, in which $\mathbf{w}$ is a weight vector adjusted by learning and $\phi(q, d)$ is a mapping onto features that describe the match between query q and data d. Here, $\phi(q, d)$ is the feature vector which consists of seven novel features that are used in computing the similarity between q and d. The task of the learner is to minimize the number of discordant ranking pairs. Though this problem is known to be NP-hard, the solution is approximated by introducing nonnegative slack variables $\xi_{i,j,k}$. Therefore, the problem is converted to the following optimization problem:

$$\begin{aligned} & \text{minimize}: \quad V(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} + C\sum \xi_{i,j,k} \\ & \text{subject to}: \\ & \forall(d_i, d_j) \in r_1^* : \mathbf{w}^T\phi(q_1, d_i) \geq \mathbf{w}^T\phi(q_1, d_j) + 1 - \xi_{i,j,1} \\ & \quad \cdots \\ & \forall(d_i, d_j) \in r_n^* : \mathbf{w}^T\phi(q_n, d_i) \geq \mathbf{w}^T\phi(q_n, d_j) + 1 - \xi_{i,j,n} \\ & \forall_i\forall_j\forall_k : \xi_{i,j,k} \geq 0. \end{aligned} \tag{3}$$

Here, C is a parameter that controls the trade-off between the margin size and training error. By rearranging the constraints in (3) as $\mathbf{w}^T(\phi(q_k, d_i) - \phi(q_k, d_i)) \geq 1 - \xi_{i,j,k}$, it becomes equivalent to that of SVM classification on pairwise difference vectors $(\phi(q_k, d_i) - \phi(q_k, d_i))$.

In the training stage, given a test object, its corresponding model should be ranked at the top. For each test-model pair, we compute the seven features to measure the similarity between them. We also know the ranking order of the model objects. Therefore, this training data is input to the SVM learning algorithm to learn the optimal ranking function. Given a test q, the model objects can be sorted in descending order based on the value of $rsv(q, d_i) = \mathbf{w}^{*T}\phi(q, d_i) = \sum \alpha_{k,l}^*\phi(q_k, d_l)^T\phi(q, d_i)$, where $\alpha_{k,l}^*$ is derived from the values of the dual variables as the solution.

Fig. 3. Examples of side face range images of three people in the UND data set Collection F.

Therefore, the ς percent of the models on the top of sorted list are selected for the verification.

3.6 Verification

After the initial rigid transformation is estimated from the corresponding pairs between a model-test pair, the ICP algorithm [2] is run to refine the transformation, which brings the model and test into the best alignment. Since the ICP algorithm requires that the test be a subset of the model database, a method to remove outliers based on the distance distribution is used [25]. Starting with the initial transformation, the modified ICP algorithm is run to refine the transformation by minimizing the distance between the control points of the model and their closest points of the test. For every model in the short list selected by the SVM ranking algorithm, the control points are randomly selected and the modified ICP is applied to those points. For a selected model object, we repeat the same procedure 15 times and choose the rigid transformation with the minimum root-mean-square (RMS) error. The model in the short list of the database with the minimum error is declared the recognized object.

4 Experimental Results

We apply the proposed framework to recognize highly similar 3D objects. We perform extensive experiments on two publicly available large 3D ear databases, part of the University of Notre Dame (UND) data set Collection F (302 subjects with 604 shots) [22] and the University of California at Riverside (UCR) data set (155 subjects with 902 shots) [6], to demonstrate the effectiveness of the approach. The ears are automatically cropped using the approach described in our paper [6]. All of the times reported in the following are measured *in seconds* on a Linux machine with an *AMD Opteron* 1.8 GHz processor. Two data sets are used:

- **The UND data set**.The data collected at UND were acquired with a Minolta Vivid 910 camera. The camera outputs a 480×640 range image and its registered color image of the same size. The UND data set that we used is from Collection F. At the time when we requested the data from UND, the Collection F contained 302 subjects with 302 time-lapse pairs of images. It is a subset of 415 person data set as it exists now. Fig. 3 shows side face range images of three people from this collection.
- **The UCR data set**. The data collected at UCR were captured by a Minolta Vivid 300 camera. The camera outputs a 200×200 range image and its registered color image. There are 155 subjects with a total of 902 shots and every person has at least four shots with two frontal views. For each subject, we capture their images on the same day. There are three different poses in the data: frontal, left, and right. Fig. 4 shows side face range images of three people. The pose variations, the earrings, and the hair occlusions can be seen.

Fig. 4. Examples of side face range images of three people (six shots) in the UCR data set.

Using these two data sets, we perform the following experiments.

4.1 Dimensionality of the Embedding

As described in Section 3, the *Stress* S is used to determine the dimensionality k of the embedded space. We perform experiments on a subset of the UCR data set and compute the *Stress* with respect to different k. For $k =$ 12, 16, 20, and 24, the computed values of Stress S are 0.351, 0.312, 0.277, and 0.246, respectively. Similar results are obtained on the UND data set. We observe that S decreases as k increases. Since the "curse of dimensionality" is a problem for the K-d tree, we choose $k = 24$ for the two data sets. In Table 2, for different values of k, we show the times for searching the nearest neighbors with and without feature embedding on the two data sets. We see that the speed for searching the nearest neighbors in the embedded space is directly proportional to the dimensionality k. Since we select $k = 24$, it results in a speedup of $\sim$90 times as compared with using the sequential search in the original feature space.

4.2 Correspondences Using the Geometric Constraints

Once the LSPs are embedded in the low-dimensional space, the correspondences are obtained by searching the nearest neighbors which are filtered by the geometric constraints. Fig. 5 shows one

TABLE 2
Time (in Seconds) for Searching Nearest Neighbors with and without Feature Embedding

No embedding		(985.4, 93.5)			
With embedding	k	12	16	20	24
	Time	4.69, 0.47	7.1, 0.71	9.27, 0.82	11.1, 1.06
	Speed up	210, 199	139, 122	106, 114	89, 88

The first number is on the UND data set (480,000 LSPs) and the second one is on the UCR data set (300,000 LSPs).

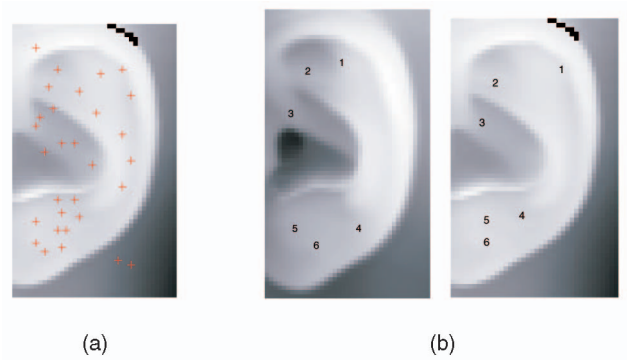

Fig. 5. An example of forming groups for corresponding LSPs for a pair of ears. (a) Feature points marked by + signs extracted from a test ear. (b) Corresponding pairs obtained by applying the geometric constraints (1). In (b), the model ear is on the left side.

example of recovered correspondences. Fig. 5a shows the feature point extraction results marked by the red pluses for a test ear and Fig. 5b shows the recovered correspondences, in which every pair is represented by the same number superimposed on the test and model images. We can see that the true corresponding pairs are obtained by searching the nearest neighbors and using the geometric constraints. Each group of the correspondences belongs to either the matched pairs or to the nonmatched pairs. For each of them, we compute seven features as a measure of similarity between a pair. The distributions of these features are shown in Figs. 6 and 7. If a Bayesian classifier is used to classify a group either from a matched pair or a nonmatched pair, it may not work well since the feature distributions for matched and nonmatched pairs have a significant overlap, which can be clearly observed in Figs. 6 and 7. Instead, the SVM rank learning algorithm is used to rank the candidate models based on the proposed seven features, without making any assumption about the feature distributions.

4.3 SVM Rank Learning Algorithm

To evaluate the performance of the approach, each of the two data sets is divided into disjoint subsets for training and testing. For the SVM rank learning, we randomly select 30 percent of the subjects (90 people for the UND data set and 46 people for the UCR data set) as the training set to learn the parameters. The range images

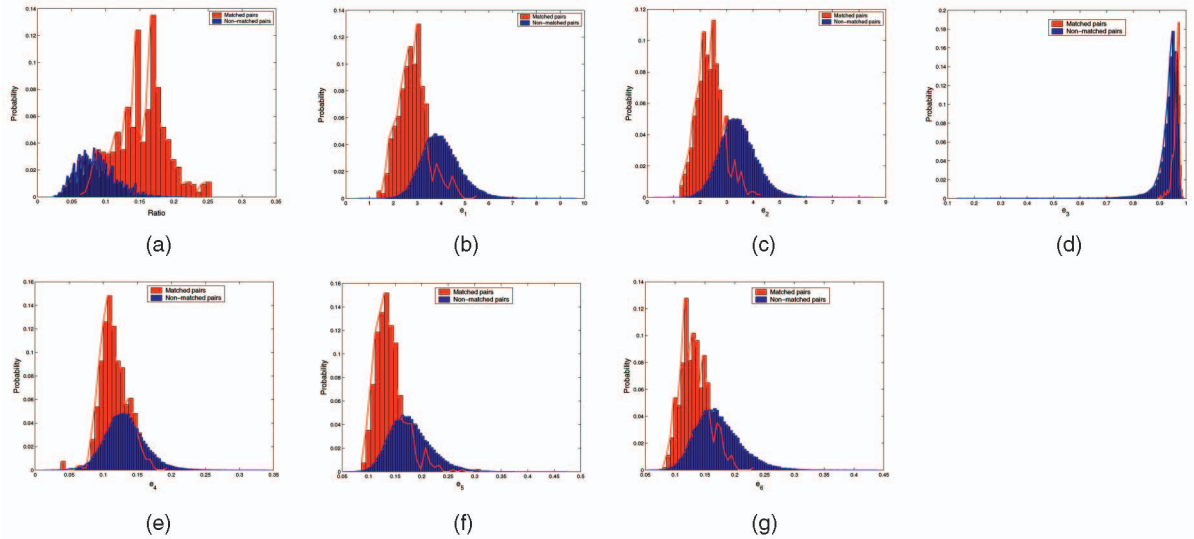

Fig. 6. UND data set: Distributions of the seven features for the matched and nonmatched pairs. (a) Ratio. (b) e_1. (c) e_2. (d) e_3. (e) e_4. (f) e_5. (g) e_6.

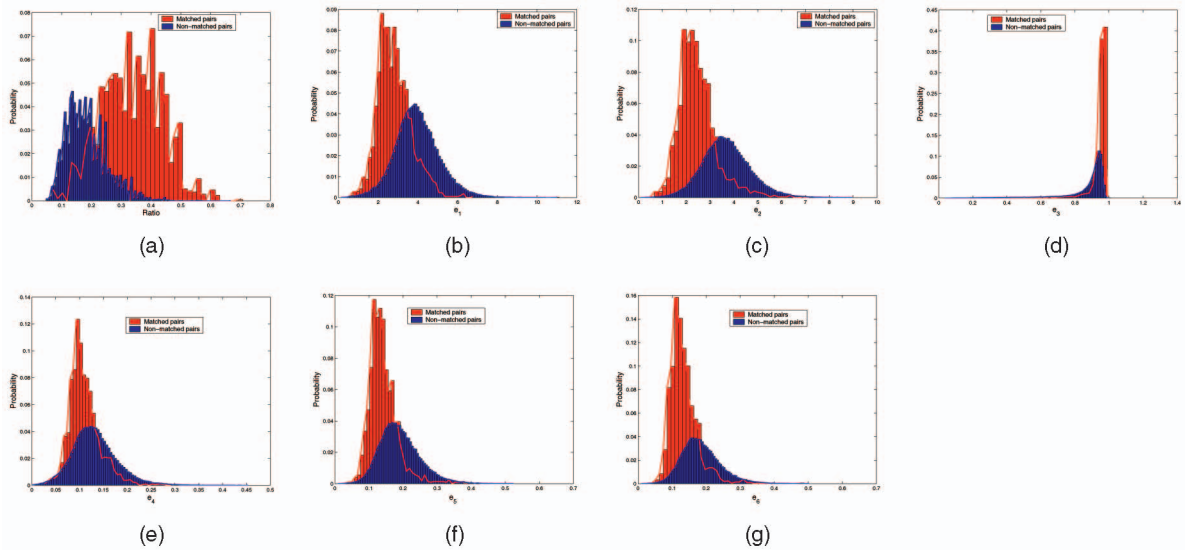

Fig. 7. UCR data set: Distributions of the seven features for the matched and nonmatched pairs. (a) Ratio. (b) e_1. (c) e_2. (d) e_3. (e) e_4. (f) e_5. (g) e_6.

TABLE 3
Indexing and Recognition Performance

ς (Fraction of dataset)	Performance
0.5%	[53.3%, 53.3%], [61.7%, 61.7%]
5%	[82.54%, 81.6%], [81.72%, 81.55%]
10%	[87.74%, 87.74%], [87.41%, 85.19%]
15%	[91.51%, 90.09%], [90.69%, 87.38%]
20%	[92.92%, 92.92%], [91.89%, 88.83%]
25%	[94.39%, 93.87%], [93.34%, 90.53%]
30%	[94.81%, 94.34%], [94.38%, 90.78%]
35%	[95.75%, 94.81%], [95.35%, 91.26%]

The first and second brackets in a row are for the UND and UCR data sets. The first and the second numbers in a bracket are the indexing and recognition performance.

associated with the rest of the people in the data set are used to evaluate the performance of the approach. The UCR data set has at least four images per person and the UND data set has two images per person. We put two frontal ears of a subject in the gallery set and the rest of the ear images of the same subject in the probe set for the UCR data set and we put one image per person into the gallery and the other one in the probe set for the UND data set. When training the SVM, the RBF kernel $K(a,b) = exp(-\mu|a-b|^2)$ is used. The kernel parameter μ and the trade-off control parameter C are selected from $C \in \{0.001, 0.01, 1, 10, 100\}$ and $\mu \in \{0.001, 0.01, 0.1, 0.5, 1, 1.5, 4, 16\}$ by minimizing the 10-fold cross-validation error on the training set. We repeat the random selection three times and report the average results in the following.

4.4 Indexing and Recognition Results

The SVM rank learning algorithm outputs a ranked list of H hypotheses. If the corresponding object is in the list of top H hypotheses, we take the indexing result as correct. The indexing performance is evaluated by computing the ratio between the number of correctly indexed objects in the H hypotheses and the total number of test objects. Let H, the number of hypotheses, be a fraction ς of M which is the number of models in the database, then we calculate the indexing performance and perform the verification for the selected ς candidate models. The indexing and recognition results are listed in Tables 3 and 4. We observe that 94 percent of the objects are correctly indexed with a list of 30 percent of model objects in the database as hypotheses on the two data sets. The relatively large number of retrieved models is due to the high degree of similarity among models.

TABLE 4
Recognition Time (in Seconds) and the Performance on Three Cases (See Text)

	Case 1	Case 2	Case 3
Time per test	(1270, 436)	(400, 215)	(192, 72)
Recognition rate	(96.7, 96.4)	(96.7, 94.9)	(94.3, 90.8)

The first number in the parentheses is on the UND data set and the second one is on the UCR data set.

Table 4 shows results under three cases: Case 1 matching a test with every model object in the database without the feature embedding, Case 2 matching a test with every model object in the database with the feature embedding, and Case 3 matching a test only with the 30 percent candidate models selected from the ranked list with the feature embedding and SVM rank learning. We see that the recognition time per test with the feature embedding and rank learning is reduced by a factor of 6.6 with the 2.4 percent degradation recognition performance on the UND data set, and, on the UCR data set, the time is reduced by a factor of 6 with the degradation of 5.8 percent in the recognition performance. This could be reduced if we embed the LSPs into a higher-dimensional space. We notice that the average recognition time per test is longer on the UND data set than that on the UCR data set since the UND data set has a much higher resolution (640×480 on the UND data set versus 200×200 on the UCR data set) and it has a larger number of LSPs. From Tables 3 and 4, we also observe that the indexing and recognition performances on the UND data set is better since the UCR data set has more pose variations. For ear recognition results without indexing, the reader is referred to [6], [22].

Fig. 8 shows three examples of the correctly recognized model-test ear pairs. Fig. 8a shows the model ear and the test ear before alignment and Fig. 8b shows the model ear and the test ear after alignment. We observe that the model ear is aligned well with the test ear.

During the recognition, some errors are made and the two error cases are illustrated in Fig. 9. Figs. 9a and 9b show the range images of two visually similar test and model ears that belong to different subjects, Fig. 9c shows the true model ear overlaid on the 3D test ear after registration, and Fig. 9d shows the falsely

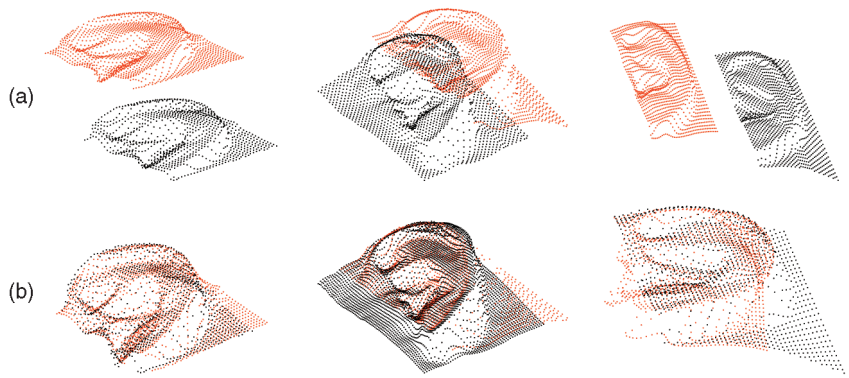

Fig. 8. UCR data set: three cases of correctly recognized model-test pairs. Each column shows one case. The model ears represented by the red pluses are overlaid on the test ears represented by the black dots. (a) Model and test ears before alignment. (b) Model and test ears after alignment. In Case 1, the rotation angle is 12.7° and the axis is $[0.4566, -0.8561, 0.2423]^T$. In Case 2, the rotation angle is 20.3° and the axis is $[-0.0204, -0.9972, 0.0713]^T$. In Case 3, the rotation angle is 25.4° and the axis is $[-0.0496, 0.9970, -0.0598]^T$.

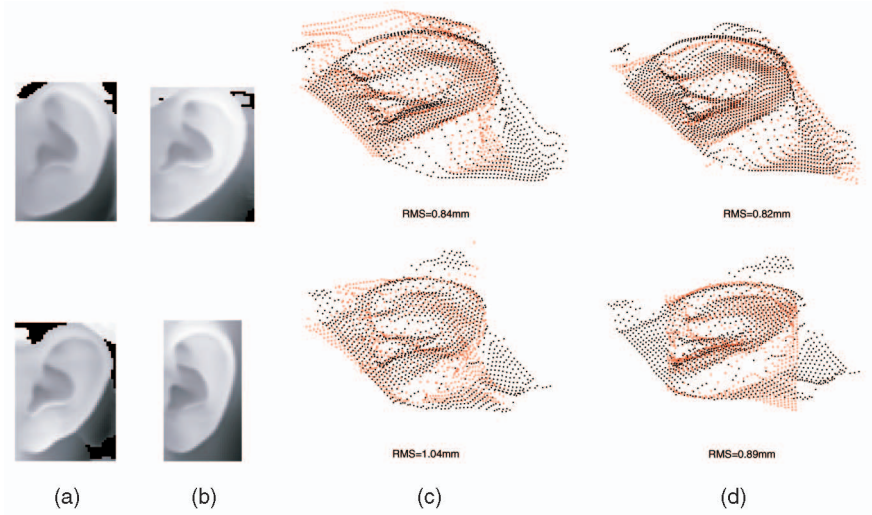

Fig. 9. UCR data set: Two cases of incorrectly recognized gallery-probe pairs. Each row shows one case. The model ears represented by the red pluses are overlaid on the test ears represented by the black dots. (a) Range images of the test ears. (b) Range images of falsely recognized model ears. (c) True model ears after alignment are overlaid on the test ears. (d) The falsely recognized model ears after alignment are overlaid on the test ears. Note that, for the incorrect matches, the model ears in column (d) achieve a smaller value of RMS error than the model ears in column (c).

recognized model ear overlaid on the 3D test ear after alignment. In Fig. 9d, the RMS error for the falsely recognized ear is smaller than the error for the correct ear in Fig. 9c. In this figure, we obtain good alignment between the model and test ears from different persons since these ears are quite similar in 3D.

4.5 Effect of Feature Embedding

We would like to evaluate the effect of the feature embedding on the verification performance for the above first two cases (see Table 4) with sequential matching. Therefore, we perform experiments on the first two cases and demonstrate the verification performance using the receiver operating characteristic (ROC) curve and the equal error rate (EER). The ROC curve is the plot of genuine acceptance rate (GAR) versus the corresponding false acceptance rate (FAR). GAR is defined as the percentage of the occurrences for which an authorized user is correctly accepted by the system, while FAR is defined as the percentage of the occurrences for which a nonauthorized user is falsely accepted by the system. The EER, which indicates the rate at which the false rejection rate $(FRR = 1 - GAR)$ and the FAR are equal, is a threshold independent performance measure. Figs. 10 and 11 show the verification performance on the first two cases on the UND and UCR data sets, respectively. We observe that the verification performance in Case 2 is slightly worse than that in Case 1 (EER increases from 0.018 to 0.020). From Table 4 and Figs. 10 and 11, we observe that the time per test with the feature embedding is reduced with a slight reduction in performance.

4.6 Comparison of the Proposed Approach with Geometric Hashing

We compare the proposed indexing approach with the popular GH technique. All of the LSPs extracted from the model objects are saved into a hash table. Given a test object, we extract feature points and get LSPs. Then, we calculate the mean and standard deviation of the shape index values for each LSP and use them to access the hash table and cast votes to model objects if the histogram dissimilarity is small and the surface type is the same. By tallying the votes from the hash table, the model objects are ranked in the descending order based on the votes they received. We perform the experiments described in Experiment 4 above on the same data sets. The comparison results with GH are listed in Table 5. We observe that the indexing performance of the proposed approach outperforms the GH on the two data sets. Although the search time for the nearest neighbors using GH on the UCR data set is about half of the time using the proposed approach, there is not much difference (9.6 versus 11.1) in time on the UND data set since it contains a larger number of LSPs. We also notice that the GH performs poorly on the UND data set since a larger number of LSPs in this data set increase the chances of collisions caused by the keys hashing to the same index.

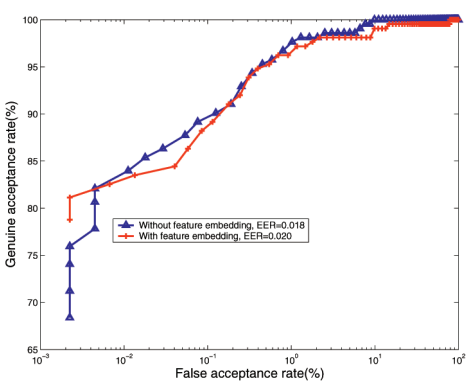

Fig. 10. UND data set: Verification performance on the first two cases in Table 4.

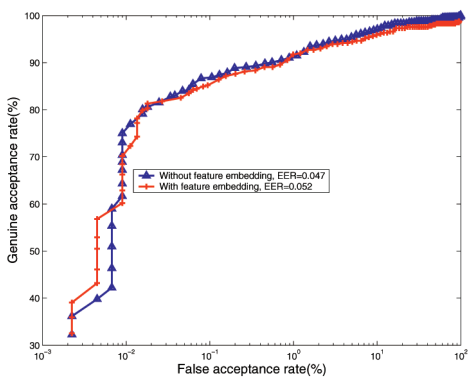

Fig. 11. UCR data set: Verification performance on the first two cases in Table 4.

TABLE 5
Comparison of the Proposed Approach with GH in Terms of the Indexing Performance and the Search Time (in Seconds) for the Nearest Neighbors

Indexing Performance			Time	
ς	This paper	GH	This paper	GH
0.5%	(53.3%, 61.7%)	(1.65%, 32.7%)	(11.1, 1.06)	(9.6, 0.54)
5%	(82.54%, 81.72%)	(23.59%, 57.68%)		
10%	(87.74%, 87.41%)	(35.61%, 69.93%)		
15%	(91.51%, 90.69%)	(45.99%, 76.49%)		
20%	(92.92%, 91.89%)	(53.77%, 81.55%)		
25%	(94.39%, 93.34%)	(59.67%, 85.88%)		
30%	(94.81%, 94.38%)	(64.39%, 89.57%)		
35%	(95.75%, 95.35%)	(69.33%, 91.42%)		

The first number in the parentheses is on the UND data set and the second one is on the UCR data set.

5 Conclusions

In this paper, we have presented a general framework for efficient recognition of highly similar 3D objects which combines the feature embedding and SVM rank learning techniques. Unlike the previous work for fast object recognition in range images, we achieved a sublinear time complexity on the number of models without making any assumptions about the feature distributions. Experimental results on two large real data sets containing highly similar objects in shape confirmed the effectiveness and efficiency of the proposed framework. Furthermore, a comparison with the GH shows that the proposed approach performs much better.

Since the ears are highly similar, one has to examine a larger part of the database (25-30 percent) to achieve a decent recognition rate. However, the recognition time per test with feature embedding and SVM rank learning can be reduced by a factor of 6.6 and 6 on the UND and UCR data sets, respectively, as shown in our experiments. Considering the fact that the two data sets used here contain a large number of highly similar 3D objects, the proposed approach is promising for general 3D object indexing and recognition and it is expected to work better and faster since the LSP features will be more distinct.

Acknowledgments

The authors would like to thank the computer vision research laboratory at the University of Notre Dame for providing them with their public biometrics database Collection F that is used in this paper.

References

[1] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios, "BoostMap: A Method for Efficient Approximate Similarity Rankings," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 268-275, 2004.

[2] P. Besl and N.D. Mckay, "A Method of Registration of 3-D Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 2, pp. 239-256, Feb. 1992.

[3] B. Bhanu and X. Tan, "Fingerprint Indexing Based on Novel Features of Minutiae Triplets," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 5, pp. 616-622, May 2003.

[4] R.J. Campbell and P.J. Flynn, "A Survey of Free-Form Object Representation and Recognition Techniques," *Computer Vision and Image Understanding,* vol. 81, pp. 166-210, 2001.

[5] H. Chen and B. Bhanu, "3D Free-Form Object Recognition in Range Images Using Local Surface Patches," *Proc. 17th Int'l Conf. Pattern Recognition,* vol. 3, pp. 136-139, 2004.

[6] H. Chen and B. Bhanu, "Human Ear Recognition in 3D," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 4, pp. 718-737, Apr. 2007.

[7] C. Chua and R. Jarvis, "Point Signatures: A New Representation for 3D Object Recognition," *Int'l J. Computer Vision,* vol. 25, no. 1, pp. 63-85, 1997.

[8] C. Faloutsos and K. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets," *Proc. ACM SIGMOD '95,* pp. 163-174, 1995.

[9] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," *Proc. 25th Int'l Conf. Very Large Data Bases,* pp. 518-529, 1999.

[10] G. Hjaltason and H. Samet, "Properties of Embedding Methods for Similarity Searching in Metric Spaces," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 5, pp. 530-549, May 2003.

[11] T. Joachims, "Optimizing Search Engines Using Clickthrough Data," *Proc. ACM Conf. Knowledge Discovery and Data Mining,* pp. 133-142, 2002.

[12] A. Johnson and M. Hebert, "Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 5, pp. 433-449, May 1999.

[13] B. Matei, Y. Shan, H. Sawhney, Y. Tan, R. Kumar, D. Huber, and M. Hebert, "Rapid Object Indexing Using Locality Sensitive Hashing and Joint 3D-Signature Space Estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 7, pp. 1111-1126, July 2006.

[14] F. Mokhtarian, N. Khalili, and P. Yuen, "Multi-Scale Free-Form 3D Object Recognition Using 3D Models," *Image and Vision Computing,* vol. 19, pp. 271-281, 2001.

[15] M. Muller, T. Roder, and M. Clausen, "Efficient Content-Based Retrieval of Motion Capture Data," *Proc. ACM SIGGRAPH '05,* pp. 677-685, 2005.

[16] S. Roweis and L. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science,* vol. 290, pp. 2323-2326, 2000.

[17] Y. Rubner, C. Tomasi, and L.J. Guibas, "A Metric for Distributions with Applications to Image Databases," *Proc. Sixth IEEE Int'l Conf. Computer Vision,* pp. 59-66, 1998.

[18] F. Stein and G. Medioni, "Structural Indexing: Efficient 3-D Object Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 2, pp. 125-145, Feb. 1992.

[19] X. Tan, B. Bhanu, and Y. Lin, "Fingerprint Identification: Classification versus Indexing," *Proc. IEEE Int'l Conf. Advanced Video and Signal Based Surveillance,* pp. 151-156, 2003.

[20] J. Tenenbaum, V. Silva, and J. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science,* vol. 290, pp. 2319-2323, 2000.

[21] X. Wang, J. Wang, K. Lin, D. Shasha, B. Shapiro, and K. Zhang, "An Index Structure for Data Mining and Clustering," *Knowledge and Information Systems,* vol. 2, no. 2, pp. 161-184, 2000.

[22] P. Yan and K.W. Bowyer, "Biometric Recognition Using 3D Ear Shape," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 8, pp. 1297-1308, Aug. 2007.

[23] J.H. Yi and D.M. Chelberg, "Model-Based 3D Object Recognition Using Bayesian Indexing," *Computer Vision and Image Understanding,* vol. 69, no. 1, pp. 87-105, 1998.

[24] F. Young and R. Hamer, *Multidimensional Scaling: History, Theory and Applications.* Lawrence Erlbaum Assoc., 1987.

[25] Z. Zhang, "Iterative Point Matching for Registration of Free-Form Curves and Surfaces," *Int'l J. Computer Vision,* vol. 13, no. 2, pp. 119-152, 1994.