High-speed 3D Object Recognition Using Additive Features in A Linear Subspace

Asako Kanezaki, Hideki Nakayama, Tatsuya Harada, and Yasuo Kuniyoshi

***Abstract*— In this paper we propose a method of high-speed 3D object recognition using linear subspace method and our 3D features. This method can be applied to partial models with any size in any posture. Although it is becoming easy to obtain textured 3D models by a 3D scanner, there are few methods for 3D object recognition which take into account both shape and textures of objects. Moreover, it is difficult to achieve high-speed processing of large 3D data. Our 3D features consider the co-occurrence of shape and colors of an object's surface. The additive property of these features makes it possible to calculate the similarity between a query part and the subspace of each object in a database without division, and therefore the time for recognition is quite short. In the experiments, we compare our method with conventional methods using Spin-Images and Textured Spin-Images. We show that our method is appropriate for 3D object recognition.**

I. INTRODUCTION

It is crucially important for robots working in a daily environment to detect and recognize objects in their vicinity with near instantaneous speed. Especially in tasks of fetching, grasping or manipulating objects, it is necessary to obtain information relating to the objects' shape, location, and other properties. Since the state of the art of 3D scanning such as stereo vision and structure from motion has dramatically advanced [1], 3D shape data, as well as associated texture data, of various objects can be obtained instantly. The ability to recognize objects in a 3D scene of the environment would enable a robot to know both what and where a desired object is.

The goal of our work is to develop a system which recognizes objects in a cluttered environment (Fig. 1). The system reconstructs the 3D scene with texture information, obtains the partial query data of various objects, and recognizes them by matching the query data against a set of models in a training database. There are three challenging problems as follows:

- Matching between partial data and complete models
- Incorporating both shape and texture information
- Fast recognition

First, to cope with occlusion by the target object itself and the neighboring objects, it must be possible to perform partial matching between query data obtained from a 3D scene and the complete 3D models in the training database. Secondly, to distinguish objects which have similar shape as well as objects which have similar textures, the system should take into account co-occurrence patterns in the joint shape-texture space. Finally, the required time for recognition must be quite short, since the operation will need to be performed repeatedly to deal with the dynamic nature of daily human environments.

A. Kanezaki, H. Nakayama, T. Harada, and Y. Kuniyoshi are with Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo Bunkyo-ku, Tokyo Japan
kanezaki@isi.imi.i.u-tokyo.ac.jp

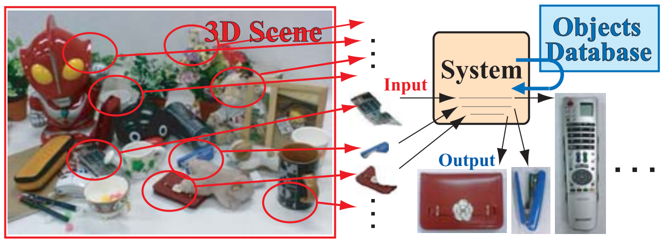

Fig. 1. The goal image of our object recognition system. Each part in a 3D scene is picked up and matched against objects in a database.

In this paper, we propose a high-speed 3D object recognition system using a linear subspace method in a 3D feature space based on Color Cubic Higher-order Local Auto-Correlation (Color-CHLAC) features [2]. Color-CHLAC features are calculated using both shape and color information in the 3D voxel data. These features can be computed from any region in each model. A crucial property of Color-CHLAC features is additivity: the full feature vector of an object equals the sum of the feature vectors of its subparts. This additive property enables fast object recognition with the linear subspace method. By the proposed method of object recognition, the system can judge which database model is the most similar to the query part in a 3D scene. This operation is performed rapidly, and without placing any requirements on the size or posture of the query part, simply by projecting the feature vector of the query part onto a subspace defined by each database model.

The rest of this paper is organized as follows: Section 2 discusses related work of 3D object recognition, Section 3 presents our 3D features, Section 4 describes the proposed recognition method, Section 5 presents the experimental results, while Section 6 summarizes our method and proposes future work.

II. RELATED WORK

Expressiveness of descriptors is critically important for object recognition. Furthermore, descriptors and recognition methods are so closely related that whether fast matching can be applied depends on the property of descriptors. In this section we discuss 3D descriptors from the point of view of expressiveness and ability to apply a fast recognition method.

A. Artificial 3D object retrieval

In the field of artificial 3D object retrieval, various 3D shape descriptors have been proposed. As mentioned in [3], they can be classified into histogram-based, transform-based, graph-based, and 2D view-based. A histogram-based descriptor [4] and transform-based descriptors [5] [6] require the center point of 3D models, making it difficult to apply these techniques to partial matching. Although a graph-based descriptor [7] can be used to perform partial matching, it is difficult to apply such a descriptor to real object recognition since a query part can only be matched against predefined sub-parts of database models. [8] uses a 2D view-based descriptor which can be computed from partial data observed in the real world, however, the memory requirement and computation cost are both high since a query part must be compared with all views of each database model.

Furthermore, these descriptors don't take texture information into consideration. [9] uses shape descriptors based on the curvature of surface patches and color descriptors represented by the average, the maximum, and the minimum value of R, G and B. However, this approach has a difficulty in balancing shape information and texture information. Also, this method does not take into account shape and texture co-occurrence patterns.

B. 3D object recognition in the real world

The Spin-Image (SI) [10] is a computational tool that is able to describe local shape patterns and to perform partial matching, so it is applied to object recognition real environments containing clutter and occlusion. A SI is created for an oriented point at a vertex in the surface mesh by projecting points onto cylindrical coordinates. It is computed by the following equation:

$$(\alpha, \beta) = (\sqrt{\|\boldsymbol{x} - \boldsymbol{p}\|^2 - (\boldsymbol{n} \cdot (\boldsymbol{x} - \boldsymbol{p}))^2}, \boldsymbol{n} \cdot (\boldsymbol{x} - \boldsymbol{p}))$$

where $\boldsymbol{p}$ is the position of an oriented point, $\boldsymbol{n}$ is its normal, and $\boldsymbol{x}$ is the position of another point in the surface mesh. Then the SI is calculated as a 2D accumulator indexed by α and β. [1] Finally, a set of SIs calculated for all the points in the surface mesh of an object is used as the description of the object.

Textured Spin-Image (TSI) [11] is an extension of SI. TSIs are computed from a surface mesh whose points have luminance information. Therefore they can take into account shape and texture co-occurrence patterns. In practice, a TSI is simply a stack of standard spin-images SI(l), where each layer $l \in [1 \dots L]$ corresponds to a given level of luminance. The dimension of a TSI is L times as large as that of a SI.

In the recognition process based on these descriptors, SIs/TSIs of randomly selected points in the query mesh are created, and then the nearest SIs/TSIs to them are found from all SIs/TSIs of each database model. Although an efficient nearest neighbor search algorithm [12] is used in this step, the large number of points in a surface mesh means that the overall computation time is substantial.

[1] Proper upper boundaries of α and $\|\beta\|$ are defined.

III. Our 3D features

Unlike the conventional recognition methods where a set of local descriptors of the query object is matched against a set of local descriptors of the database models, the proposed method is a new approach that calculates the similarity between a partial query data and the entire 3D model of a database object by projecting the query feature vector onto a feature subspace defined by the object. In this approach the required time for calculating similarity is quite small, regardless of the size and the posture of the query part in a 3D scene. Details of this method appear in Section IV. To apply the proposed method, 3D features need to have an additive property: the full feature vector of a model must equal the sum of the feature vectors of its sub-parts. We use Color-CHLAC features [2] which have the additive property. These features take into consideration shape and texture co-occurrence patterns.

Color-CHLAC features are extension of CHLAC features [13]. A CHLAC feature is an integral of the local autocorrelation of 3D voxel data. Color-CHLAC features are computed from color 3D voxel data by measuring the autocorrelation function of the 3D target object at specific points, represented by local patterns. Local descriptors are represented by the co-occurrence of their shape and colors. Because the feature vector is computed by summing over an entire region, it is robust against minor variations caused by noise or other data loss. In this section we describe how to create color 3D voxel data and how to extract Color-CHLAC features.

A. Method of creating color 3D voxel data

The simplest way of creating voxel data from a measured point cloud is to divide 3D space at regular intervals and judge whether or not a given voxel includes a measured point. However, voxel data created in this approach has many holes on view direction because measurement points tend to cluster on the object plane perpendicular to the view direction. In this paper, we first create a surface mesh from a point cloud and then transform the mesh into dense voxel data.

Fig. 2 illustrates the method of transforming the surface mesh into voxel data. First, the 3D space is divided into sufficiently small intervals, e.g. 1mm $\times$ 1mm $\times$ 1mm. Letting A, B and C be the vertices of a mesh triangle, the collision of each voxel and the line AB is detected, and a voxel is marked as "occupied" if a collision occurs. Next, lines are drawn from the voxel collision points on line AB to the line AC (these lines are parallel to BC). Then the collision of these lines and each voxel is detected and voxels are marked as occupied when appropriate. This process is repeated for all triangles in the surface mesh.

Finally, voxel data is resized to a proper resolution. For example, to transform voxel data whose size is 1mm $\times$ 1mm $\times$ 1mm to voxel data whose size is 4mm $\times$ 4mm $\times$ 4mm, $4 \times 4 \times 4$ voxels are joined together to one voxel. In this work each voxel has color values of R, G and B. When resizing, the color values of each voxel in the output voxel data is computed by averaging over the corresponding region of the input voxel data.

B. Color-CHLAC features

Letting $\boldsymbol{x} = (x, y, z)^T$ be the position of a voxel, we use the notation $p(\boldsymbol{x}) = 1$ if the voxel is occupied, and $p(\boldsymbol{x}) = 0$ otherwise. When $p(\boldsymbol{x}) = 1$, the voxel has RGB color values. We represent them as $r(\boldsymbol{x})$, $g(\boldsymbol{x})$ and $b(\boldsymbol{x})$, which are normalized between 0 and 1. By defining $r'(\boldsymbol{x}) \equiv 1 - r(\boldsymbol{x})$, $g'(\boldsymbol{x}) \equiv 1 - g(\boldsymbol{x})$ and $b'(\boldsymbol{x}) \equiv 1 - b(\boldsymbol{x})$, a voxel status $\boldsymbol{f}(\boldsymbol{x}) \in R^6$ is defined as follows:

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{cases} (r(\boldsymbol{x})\ r'(\boldsymbol{x})\ g(\boldsymbol{x})\ g'(\boldsymbol{x})\ b(\boldsymbol{x})\ b'(\boldsymbol{x}))^T & (p(\boldsymbol{x})=1) \\ (0\ 0\ 0\ 0\ 0\ 0)^T & (p(\boldsymbol{x})=0) \end{cases}$$

As a pre-processing of features extraction, $r(\boldsymbol{x})$, $g(\boldsymbol{x})$ and $b(\boldsymbol{x})$ can be binarized. If they are binarized, the resulting voxel status $\boldsymbol{f}(\boldsymbol{x})$ can be categorized into 9 patterns as shown in Fig. 3.

Color-CHLAC features are the integral of $\boldsymbol{f}(\boldsymbol{x})$ or correlations of $\boldsymbol{f}(\boldsymbol{x})$ between neighboring voxels. They are calculated by following equations:

$$\boldsymbol{q} = \int \boldsymbol{f}(\boldsymbol{x}) d\boldsymbol{x} \tag{1}$$

$$\boldsymbol{q}(\boldsymbol{a}) = \int \boldsymbol{f}(\boldsymbol{x})\ \boldsymbol{f}^T(\boldsymbol{x} + \boldsymbol{a}) d\boldsymbol{x} \tag{2}$$

The dimension of Color-CHLAC features calculated by (1) is 6. 14 patterns are used for the displacement vectors $\boldsymbol{a}$ in (2) (Fig. 4). Note that not only $\boldsymbol{f}(\boldsymbol{x})$ correlation between two neighboring voxels but also the correlation between two elements of $\boldsymbol{f}(\boldsymbol{x})$ of one voxel is integrated. Excluding redundant elements, the dimension of Color-CHLAC features calculated by (2) is 480 if color values are binarized, and 489 otherwise.

In [2] we used color binarization, however, this processing does not always work well. By color binarization, the patterns of neighboring voxels whose colors are different each other are emphasized and detected properly. Also, the robustness to small changes in light intensities is achieved. On the other hand, if the colors of the target object are near the thresholds, the features can be sensitive to light variations. Moreover, continuous color values include richer information than binary color values. In this paper, we extract Color-CHLAC features both from binarized color voxel data and from original color voxel data. Then the dimension of Color-CHLAC feature vector becomes 981 (=6+480+6+489). To decide the threshold of color binarization, we apply the histogram threshold selection method of [14] to the R, G and B values respectively, using the voxel colors of all objects in the database as sample data.

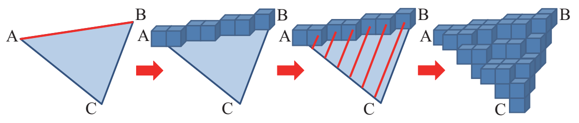

Fig. 2. Illustration of transformation from mesh data into voxel data.

State Variable $f(x)$	Voxel State		State Variable $f(x)$	Voxel State	
$(0\ 0\ 0\ 0\ 0\ 0)^T$	Empty	⬚	$(1\ 0\ 1\ 0\ 0\ 1)^T$	Yellow	🟨
$(0\ 1\ 0\ 1\ 0\ 1)^T$	Black	⬛	$(0\ 1\ 1\ 0\ 1\ 0)^T$	Cyan	🟦
$(1\ 0\ 0\ 1\ 0\ 1)^T$	Red	🟥	$(1\ 0\ 0\ 1\ 1\ 0)^T$	Magenta	🟪
$(0\ 1\ 1\ 0\ 0\ 1)^T$	Green	🟩	$(1\ 0\ 1\ 0\ 1\ 0)^T$	White	⬜
$(0\ 1\ 0\ 1\ 1\ 0)^T$	Blue	🟦	Others	× forbidden	☒

Fig. 3. Patterns of binarized color voxel status.

IV. Proposed Recognition Method

In this section we describe the proposed recognition system based on a linear subspace method [15]. As a preprocessing step, we calculate the Color-CHLAC feature vectors of subdivided parts of each database model. Then we use these feature vectors to compute a basis for a subspace defined by each object. In the recognition process, one feature vector is extracted from a query part in a 3D scene. Then the query part is matched against the database by projecting the query vector into each object's subspace, and calculating a similarity score. Fig. 5 shows the system chart.

A. Color-CHLAC features compression

The dimension of a Color-CHLAC feature vector is 981, which is rather large. Larger feature dimension means longer computational requirements for calculating similarity. In this paper, we compress Color-CHLAC feature vectors by Principal Component Analysis (PCA), using feature vectors extracted from all subdivisions of all database objects. This is the same idea that used for SI [10] and TSI [11].

PCA is effective in our work because it is a linear transformation. Our method requires that the feature vectors have an additivity property. Since PCA is a linear transformation, so the additive property remains in the compressed feature vectors. In this paper we chose to use the 100 top PCA vectors.

B. Creating subspaces of objects in a database

First, the voxel data of each object in a database is subdivided into a voxel grid of a certain size, e.g. $10 \times 10 \times 10$

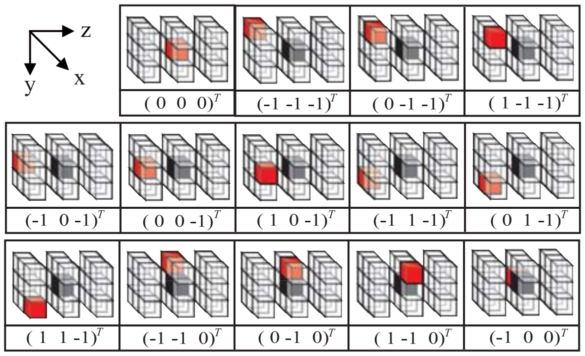

Fig. 4. Patterns of displacement vectors. Position of the center voxel is $\boldsymbol{x}$, while position of the other highlighted voxel is $\boldsymbol{x} + \boldsymbol{a}$.

voxels. Suppose the i-th object in the database is divided into M_i subdivisions. Then M_i Color-CHLAC feature vectors are extracted. To achieve robustness to rotation, features extraction is repeated with various poses of the object. The feature vector of an object which is rotated by 90 degrees can be obtained rapidly through a simple exchange of the elements of the feature vector in the initial posture. This is possible because each displacement vector in Fig. 4 is equivalent to another, rotated by 90 degrees. In this paper, we use this 90 degrees rotation in 24 ways, and rotations of 30 and 60 degrees in 21 ways, resulting in achieving $504(= 24 \times 21)$ varieties of postures for an object. Therefore, the number of Color-CHLAC feature vectors generated from each object is $N_i \equiv 504 M_i$.

We represent a Color-CHLAC feature vector compressed by PCA as $\boldsymbol{z}_t \in R^d, t = 1, 2, \dots N_i$, where d is the dimension of a compressed feature vector. The auto-correlation matrix of these feature vectors is calculated by the following equation:

$$R_i = \frac{1}{N_i} \sum_{t=1}^{N_i} \boldsymbol{z}_t \boldsymbol{z}_t^T$$

The eigenvectors of R_i are then computed by solving the eigenvector problem. Finally, the bases of the subspace of the i-th object in the database, $P_i \equiv (\boldsymbol{v}_{i1} \boldsymbol{v}_{i2} \dots \boldsymbol{v}_{ir})$, are obtained as the r eigenvectors of largest eigenvalue.

C. Calculation of Similarity

The first step in the recognition process is to compute one Color-CHLAC feature vector from the whole of a query part in a 3D scene. Then the compressed feature vector $\boldsymbol{z}$ is computed using the projection matrix which is generated from all Color-CHLAC feature vectors of each database object in the pre-processing phase. Let the similarity between the query part and the i-th object in the database be y_i. y_i is defined as the cosine of the angle between $\boldsymbol{z}$ and the subspace of the i-th object (Fig. 6). y_i is calculated by the following equation:

$$y_i = \frac{\|P_i^T \boldsymbol{z}\|}{\|\boldsymbol{z}\|} \tag{3}$$

The similarity between a query part and objects in a database can be calculated in this way because of the additive property of Color-CHLAC features. Due to this property, the feature vector of the whole of a query part equals the sum of the feature vectors of its sub-parts. Therefore the full feature vector of a query part is included in the proper subspace if the feature vectors of its sub-parts are included in it. Consequently, the similarity measure can be calculated rapidly, without dividing the query part into subregions, regardless of the size and posture of the query part.

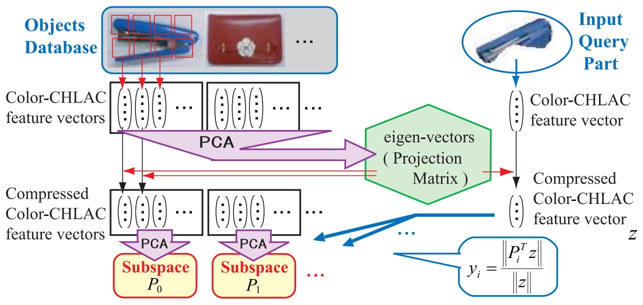
Objects Database
Color-CHLAC feature vectors
PCA
eigen-vectors (Projection Matrix)
Compressed Color-CHLAC feature vectors
PCA
PCA
Subspace P_0
Subspace P_1
Input Query Part
Color-CHLAC feature vector
Compressed Color-CHLAC feature vector
z
$y_i = \frac{\|P_i^T z\|}{\|z\|}$

Fig. 5. System chart.

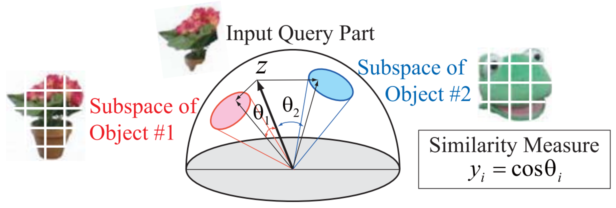

Fig. 6. Illustration of linear subspace method used for our system.

D. Computation and Memory Complexity

Let the dimension of the original feature vector be $D(= 981)$, the number of the database models be m, and the total number of the feature vectors of all database objects be $N(\equiv \sum_{i=1}^{m} N_i)$. The computation complexity of the feature compression step (described in Section IV-A) is $O(ND^2)$, and its memory complexity is $O(D^2)$. In the same manner, the computation complexity of calculating each object's subspace (described in Section IV-B) is $O(N_i d^2)$, and its memory complexity is $O(d^2)$. In order to add a new object to the database, these two steps should be repeated. However, if the database has sufficiently large number of objects, the first step can be skipped. This is because the projection matrix obtained by sampling various objects is expected to be applicable to a new object. Then only the second step for the object added to the database is needed, which can be performed within one minute when the bounding box size of the object is approximately 600mm $\times$ 600mm $\times$ 600mm. The size of the object influences this processing time since it is proportional to N_i.

The computation complexity and the memory complexity of (3) is $O(dr)$, regardless of the size or the shape complexity of the database object. Therefore the recognition time cost of a query part in a 3D scene is $O(dmr)$, which is so fast that the recognition system can work online.

V. Experiment

In this experiment we test the performance of the proposed approach on object recognition in a real 3D scene. Supposing that 3D models of all objects in an environment are preliminarily measured and stocked in a database, the system can recognize them in a new situation by matching them against database objects. The system calculates the similarity between each part of objects in a 3D scene and all objects in a database, and then provides a ranking of the objects in the database. We compare our method with conventional approaches using SI [10] and TSI [11].

A. SI and TSI parameters

Important parameters in SI generation are *bin size*, *image width*, and *support angle*. Details of them are given in [10].

According to [10], choosing the bin size parameter to be equal to the mesh resolution creates descriptive SI. We set the bin size to 4.14mm, which is the average length of edges in all the surface mesh of objects in the database. Image width is set to 15 and support angle is set to 60 degrees, which are the same values as those in [10]. The height of SI is also set to 15 in the same way as [10].

In the recognition step, a fraction of oriented points are selected at random from each query part. Let the number of selected points be K. Following [16], we set K to a fifth of N_q, where N_q is the number of points of the query part. Then K SIs are generated and matched against all SIs of each object in the database. To find closest points, we use the efficient closest point search structure [12], in the same way as [16]. In this approach, one is interested only in the closest point if it is less than a predetermined distance ϵ from the query point. Choosing ϵ to be small allows faster lookup of closest points, but decreases the likelihood of finding the correct closest point. In this paper, we tested five choices, 5, 10, 15, 20, and 25 for ϵ.

After closest point search, the similarity between each point in the query part and the closest point in the i-th object in the database is computed. The definition of the similarity measure between two SIs follows [16]. In [16], a query part in a 3D scene and each object in a database are tightly matched by geometric matching using groups of point correspondences. However, unlike [16], the objective of our work is not to compute a transformation from model to scene but to recognize objects in scene quickly. In this paper, we define the similarity between a query part and the i-th object in the database, that is y_i, as the summation of the similarity measures between each point in the query part and the closest point in the i-th object. Letting the number of SIs of the i-th object be N_i, these SIs be $\boldsymbol{p}_n$, and SIs generated from the query part be $\boldsymbol{q}_k$, y_i is given by

$$y_i = \sum_{k=1}^{K} \max_{n=1,\ldots,N_i} \left(\left(\operatorname{atanh}\left(R\left(\boldsymbol{p}_n, \boldsymbol{q}_k\right)\right)\right)^2 - \lambda \left(\frac{1}{c_{nk} - 3} \right) \right)$$

Where c_{nk} is the number of overlapping pixels used in the computation of correlation coefficient R. The variance of the correlation coefficient transformed by the hyperbolic arctangent function becomes $1/(c_{nk}-3)$. λ weights the variance against the expected value of the correlation coefficient. Further details are given in [16]. We set λ to 3, as is done in [16].

According to [11], the value of the luminance level L in TSI should be small, ranging from 3 to 8. We tested all these values for L. The SI and TSI feature vectors are compressed using PCA. We tested two choices, $d = 25$ and $d = 100$, for the dimension of the compressed feature subspace. Both in [10] and [11] d is set to 25.

Table I shows SI and TSI parameters decided above.

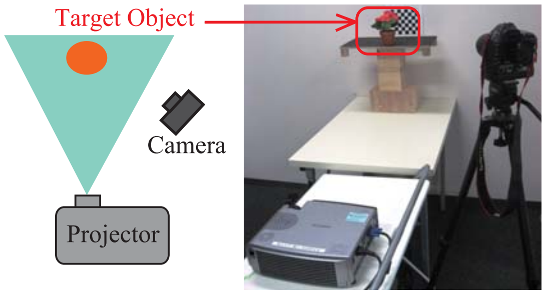

Fig. 7. Experimental Setup. The target object is placed in front of a projector. A camera looks the object from another angle.

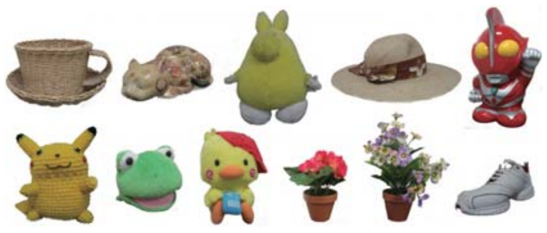

Fig. 8. Pictures of 11 objects in the database.

B. Setup

To obtain 3D data with associated textures, we use a 3D scanner with FlexScan3D Software[2]. To create 3D models, a series of reference patterns is projected onto the subject using a presentation projector, while the scene is captured using cameras. Experimental setup is shown in Fig. 7.

As a pre-processing step, we created textured 3D models of 11 objects shown in Fig. 8. Each model consists of 3D data scanned from eight or ten directions. To register multiple scan data, we used the Leios Studio software package[3].

3D models are obtained as surface meshes with associated textures. For SI and TSI extraction, the resolution of each surface mesh is changed by mesh resampling [17] so that the average length of edges in the surface mesh may be around 4mm. For Color-CHLAC features extraction on the other hand, we transform each model into 4mm $\times$ 4mm $\times$ 4mm color voxel data by the method described in Section III-A. Examples of model's surface mesh are shown in Fig. 9 and examples of model's color voxel data in Fig. 10.

We created 22 test scenes, each of which includes 5 of the 11 objects (Fig. 8). An example of the test scenes is

TABLE I
SI AND TSI PARAMETERS.

Parameter	Value
Bin size	4.14
Image Width	15
Support Angle [deg]	60
Number of points (K)	$N_q/5$
Search distance (ϵ)	5, 10, 15, 20, 25
Luminance level for TSI (L)	3, 4, 5, 6, 7, 8
Compressed dimenstion (d)	25, 100

[2]http://www.3d3solutions.com/products/flexscan3d
[3]http://www.3d3solutions.com/products/leios

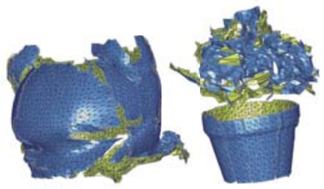
Fig. 9. Surface mesh examples

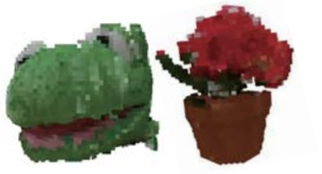
Fig. 10. Color voxel examples

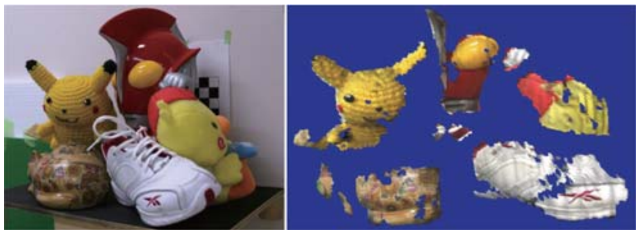
Fig. 11. A test scene example. The left image is a picture captured by the camera. The right image is colored mesh data of the test scene.

shown in Fig. 11. Objects were placed in various postures touching each other, so that the amount of occlusion was significant. Then we generated 110 query parts from test scenes by removing irregular faces and manually selecting areas corresponding to objects. Finally the surface meshes of query parts are smoothed, resampled for SI and TSI extraction, and transformed into color voxel data for Color-CHLAC features extraction.

C. Results

Comparison of the average correct rate against 110 query parts is shown in Table II. The parameter ϵ represents the search parameter used in narrowing down the closest SI/TSI candidates, while the parameter r in our method represents the dimension of subspaces of database objects. First-tier is the percentage of trials in which the correct object is ranked first among the 11 database objects, while second-tier is the percentage where the correct object is first or second, and third-tier is the percentage where the correct object is first, second, or third. In case of using SI or TSI, recognition is repeated ten times using a different choice of K selected points as the query part, and then the average correct rate is computed. Note that the results of TSI shown in Table II are obtained when $L = 4$, which are better than those with all the other choices for L, from 3 to 8.

Comparison of the best scores of average correct rate is shown in Fig. 12. The numbers below the names of descriptors represent the compressed feature vector dimension d. The results of SI and TSI shown in Fig. 12 are reported when $\epsilon = 25$. Regarding the proposed method, the dimension of subspaces of database objects r is 35. As seen in Fig. 12, First-tier, second-tier and third-tier are all the highest when the proposed approach is applied.

Table II also shows the time required for (I) extracting features of a query part and (II) calculating all the similarities between the query part and 11 objects in the database. The time required for feature compression is included in (I). The reported computation time is obtained using a Pentium IV

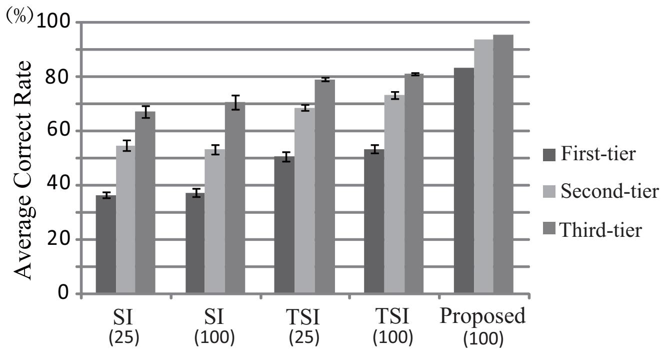

Fig. 12. Comparison of the best score of average correct rate. The numbers below the names of descriptors represent the compressed feature vector dimension d. The results of SI and TSI are reported when $\epsilon = 25$. The result of the proposed method is reported when $r = 35$.

3.4 GHz with 1.0 GB of main memory and a C++ implementation. Note that for equal comparison, we do not use parallel processing. The proposed method shows substantially faster performance, especially in the time required for (II). The computation complexity of calculating the similarity between the query part and the i-th object in the database is $O(dr)$ with the proposed method and $O(dKlog_2N_i)$ with the others. For SI or TSI, the computation time of (II) can be decreased by decreasing K, but this will adversely affect the correct rate. The search distance parameter ϵ also affects the time required for (II). Table II indicates that the computation time of (II) is significantly different between $\epsilon = 5$ and $\epsilon = 25$. On the other hand, in the proposed approach, d and r are the only parameters that influence recognition time. In brief, the proposed method also has the benefit that choosing parameters is easy.

VI. Conclusion and future work

In this paper, we proposed a fast 3D object recognition method using Color-CHLAC features and a linear subspace method. Color-CHLAC features describe the co-occurrence of shape and colors of an objects' surface. Furthermore, these features have the additive property, that is, the property that the full feature vector of a partial model equals the sum of the feature vectors of its sub-parts. Because of this property, the similarity measure can be calculated rapidly by projecting the query feature vector onto the feature subspace defined by each database object. The recognition performance of the proposed approach was evaluated in experiments with real 3D scenes. Our approach proved to be successful compared with other approaches using Spin-Images and Textured Spin-Images.

Our future work is as follows. First, in the current experiment, we created query parts from test scenes by removing irregular faces and selecting each object's area manually. These processes should be automated. Second, it is necessary to verify the ability to detect novel objects. In the case that the query object is not included in the database, the system is required to label it "unknown". This can be enabled by setting a proper threshold of the similarity of

TABLE II

AVERAGE CORRECT RATE AND COMPUTATION TIME. THE BEST SCORES ARE WRITTEN IN BOLD FONTS. NOTE THAT THE SCORES IN SHADED CELLS ARE GRAPHICALLY SHOWN IN FIG. 12.

Feature	Dim.(d)	Parameter ϵ	First-tier [%]	Second-tier [%]	Third-tier [%]	(I) Feature Extraction [msec]	(II) Similarity Computation [msec]
SI	25	5	30.9 ± 2.5	42.6 ± 3.0	50.4 ± 2.3	43	137
		10	33.3 ± 1.1	**55.0 ± 1.5**	**69.0 ± 2.9**		835
		15	30.0 ± 1.6	52.3 ± 1.7	64.1 ± 0.9		2590
		20	33.4 ± 1.5	53.9 ± 2.2	65.5 ± 1.8		5160
		25	**36.6 ± 1.1**	**55.0 ± 2.0**	67.2 ± 2.1		8130
	100	5	31.9 ± 2.5	45.3 ± 2.2	54.0 ± 2.0	46.7	603
		10	34.4 ± 2.0	**56.6 ± 2.6**	**70.7 ± 2.5**		3180
		15	31.4 ± 1.3	50.6 ± 1.1	64.5 ± 2.2		9970
		20	34.3 ± 1.7	52.0 ± 0.9	67.3 ± 1.1		19900
		25	**37.3 ± 1.5**	53.4 ± 1.6	**70.7 ± 2.5**		31000
TSI	25	5	40.3 ± 1.8	58.3 ± 2.4	70.0 ± 2.0	48.4	332
		10	50.6 ± 2.6	66.1 ± 0.5	73.1 ± 1.5		1950
		15	**51.4 ± 1.2**	66.3 ± 1.8	75.4 ± 1.1		4920
		20	50.2 ± 2.2	67.7 ± 1.1	77.5 ± 1.3		8620
		25	50.8 ± 1.9	**68.7 ± 1.0**	**79.2 ± 0.6**		12400
	100	5	39.4 ± 2.2	59.0 ± 2.1	71.8 ± 2.2	55.7	1330
		10	52.4 ± 1.9	66.7 ± 1.2	73.3 ± 1.1		7390
		15	50.9 ± 1.6	68.5 ± 1.0	78.2 ± 0.8		18700
		20	53.3 ± 1.6	69.6 ± 1.2	79.5 ± 0.4		32500
		25	**53.6 ± 1.7**	**73.3 ± 1.4**	**81.1 ± 0.5**		47400
Feature	Dim.(d)	Parameter r	First-tier [%]	Second-tier [%]	Third-tier [%]	(I) Feature Extraction [msec]	(II) Similarity Computation [msec]
Proposed	100	5	63.1	85.6	94.8	2.17	0.191
		10	72.7	81.5	91.9		0.319
		15	73.3	85.1	92.1		0.457
		20	79.5	86.8	94.9		0.581
		25	76.7	84.7	90.2		0.718
		30	83.1	92.1	95.5		0.844
		35	**83.3**	**93.8**	95.5		0.982
		40	81.3	92.1	95.5		1.09
		45	81.3	93.0	**97.4**		1.24
		50	83.2	91.2	96.4		1.38
		55	81.3	93.1	96.4		1.52
		60	80.4	89.6	95.6		1.66
		65	80.5	91.2	95.6		1.80
		70	75.9	89.4	95.7		1.93
		75	74.2	87.6	92.2		2.11
		80	72.3	86.0	91.3		2.23
		85	71.3	85.2	89.5		2.41
		90	74.0	87.9	93.5		2.59
		95	68.4	83.5	90.7		2.68

each database object. Finally, it is also our future work to apply the proposed method to develop an online system for an autonomous mobile robot.

REFERENCES

[1] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops and R. Koch, "Visual modeling with a hand-held camera", *Int. J. of Computer Vision*, Vol. 59, No. 3, pp. 207–232, 2004.

[2] A. Kanezaki, T. Harada and Y. Kuniyoshi, "Partial matching for real textured 3D objects using color cubic higher-order local auto-correlation features", *Proc. Eurographics Workshop on 3DOR*, pp. 9–12, 2009.

[3] C. B. Akgul, B. Sankur, Y. Yemez and F. Schmitt, "3D model retrieval using probability density-based shape descriptors", *IEEE Trans. Pattern Anal. and Mach. Intell.*, Vol. 31, pp. 1117–1133, 2009.

[4] T. Zaharia and F. Preteux, "Shape-based retrieval of 3D mesh models", *Proc. IEEE ICME*, 2002.

[5] D. V. Vranic and D. Saupe, "3D shape descriptor based on 3D fourier transform", *Proc. ECMCS*, pp. 271–274, 2001.

[6] M. Kazhdan, T. Funkhouser and S. Rusinkiewicz, "Rotation invariant spherical harmonic representation of 3D shape descriptors", *Proc. Symposium on Geometry Processing*, 2003.

[7] H. Sundar, D. Silver, N. Gagvani and S. Dickinson, "Skeleton based shape matching and retrieval", *Proc. Shape Modeling International*, pp. 130–139, 2003.

[8] R. Ohbuchi, K. Osada, T. Furuya and T. Banno, "Salient local visual features for shape-based 3D model retrieval", *Proc. IEEE Int. Conf. on Shape Modeling and Applications*, 2008.

[9] S. Park, X. Guo, H. Shin and H. Qin, "Surface completion for shape and appearance", *The Visual Computer*, Vol. 22, pp. 168–180, 2006.

[10] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes", *IEEE Trans. Pattern Anal. and Mach. Intell.*, Vol. 21, pp. 433–449, 1999.

[11] G. M. Cortelazzo and N. Orio, "Retrieval of colored 3D models", *Proc. the Third Int. Sym. on 3DPVT*, 2006.

[12] S. Nene and S. Nayar, "Closest point search in high dimensions", *Proc. IEEE Conf. CVPR*, pp. 859–865, 1996.

[13] T. Kobayashi and N. Otsu, "Action and simultaneous multiple-person identification using cubic higher-order local auto-correlation", *Proc. IEEE Conf. ICPR*, Vol. 4, pp. 741–744, 2004.

[14] N. Otsu, "A threshold selection method from gray-level histograms", *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 9, No. 1, pp. 62–66, 1979.

[15] E. Oja, "Subspace methods of pattern recognition", *Research Studies Press*, 1983.

[16] A. E. Johnson and M. Hebert, "Surface matching for object recognition in complex three-dimensional scenes", *Image and Vision Computing*, Vol. 16, pp. 635–651, 1998.

[17] A. E. Johnson and M. Hebert, "Control of polygonal mesh resolution for 3-D computer vision", *Graphical Models and Image Processing*, Vol. 60, pp. 261–285, 1998.