Fast Object Detection for Robots in a Cluttered Indoor Environment Using Integral 3D Feature Table

Asako Kanezaki, Takahiro Suzuki, Tatsuya Harada, and Yasuo Kuniyoshi

***Abstract*— Realizing automatic object search by robots in an indoor environment is one of the most important and challenging topics in mobile robot research. If the target object does not exist in a nearby area, the obvious strategy is to go to the area in which it was last observed. We have developed a robot system that collects 3D-scene data in an indoor environment during automatic routine crawling, and also detects objects quickly through a global search of the collected 3D-scene data. The 3D-scene data can be obtained automatically by transforming color images and range images into a set of color voxel data using self-location information. To detect an object, the system moves the bounding box of the target object by a certain step in the color voxel data, extracts 3D features in each box region, and computes the similarity between these features and the target object's features, using an appropriate feature projection learned beforehand. Taking advantage of the additive property of our 3D features, both feature extraction and similarity calculation are considerably accelerated. In the object learning process, the system obtains the feature-projection matrix by weighting unique features of the target object rather than its common features, resulting in reducing object detection errors.**

I. INTRODUCTION

It is a frustrating task for people to locate objects in an untidy indoor environment. This is especially true in a communal area such as an office or a laboratory, where objects are often lost because they have been moved somewhere by someone else. Our objective is to develop a mobile robot system (Fig. 1) that can automatically locate objects in a fairly vast indoor environment, in which a number of people reside. While automatically and routinely crawling through the environment, the robot updates the 3D-scene data. When the system receives a request to search for a specific object, it performs a global search of the 3D-scene data and selects several regions that appear most similar to the target object. The robot then proceeds to each of these areas, in descending order of similarity, to locate the object.

This paper focuses on the object detection process using the 3D-scene data from an environment, consisting of a colored-textured surface mesh obtained by a camera and a range sensor on the robot's head. The scanned data are restored using the robot's location information obtained by SLAM, and are registered into a 3D map of the environment once the crawling has been completed.

A. Kanezaki, T. Suzuki, 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
t-suzuki@isi.imi.i.u-tokyo.ac.jp
harada@isi.imi.i.u-tokyo.ac.jp
kuniyosh@isi.imi.i.u-tokyo.ac.jp

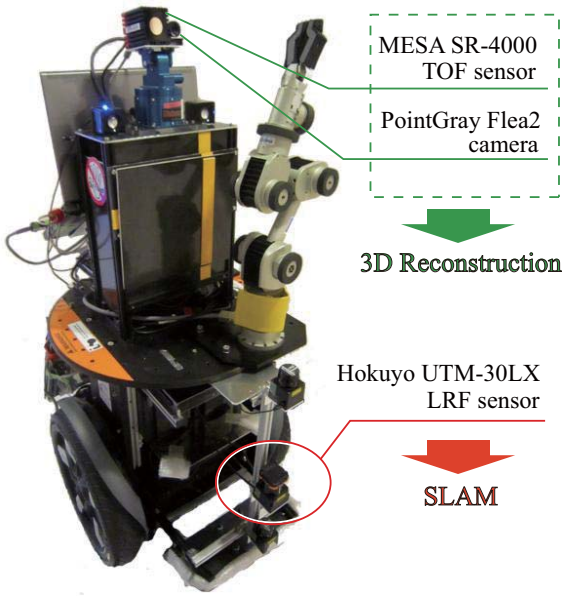

Fig. 1. Our mobile robot system equipped with a Flea2 camera, a SR-4000 TOF sensor, and a UTM-30LX LRF sensor.

There are several approaches to object detection in a 3D environment. The first involves detecting keypoints of SIFT [1] or SURF [2] in 2D images of the environment, comparing these to the keypoints in the images of the target object, and then checking the 3D geometric validity of the selected keypoints. The second approach is to match the 3D points in the environment and those in the target object's 3D model, and then consider the similarity of color textures at these points. The former method is a texture-priority approach, while the latter is a shape-priority approach. Where the textures of the target object are characteristic, but not the shape, (e.g., books) the former approach is appropriate, whereas the latter approach is adequate when the shape, rather than the texture, of the object is salient (e.g., single colored soft toys). A system must switch between these two approaches to cover a larger variety of target objects, resulting in a loss of autonomy. Thus, it would be better to extract features that consider both shape and appearance information, and to select effective features for the detection process by learning.

Another aspect that should be considered in determining the object detection scheme is whether to use a keypoint-based method or a region-based one. A keypoint-based method finds the best matching pairs of points on the

scene and target object using 2D (such as SIFT [1]) or 3D (such as Spin Image [3]) descriptors, and then selects the area with the greatest number of points. A region-based method, such as the "sliding window", selects a small part of the scene, extracts some features (e.g., color histogram), compares these to the target objects features, and then moves to the next region. Although keypoint-based methods are robust to occlusion and rotation, region-based methods that exclude other objects before calculating similarity, are more appropriate for detecting small objects in a vast environment.

We propose a framework for "sliding box" object detection in color 3D voxel data. Whereas an image pixel means a 2D square index at regular intervals, a voxel in 3D data implies a 3D cubic index at regular intervals, with either RGB values or the "empty" property. Thus, extracting features from color voxel data means considering both color-texture and shape information. In our sliding box approach, which is a simple extension of the sliding window approach, a detection box of a certain size is moved a fixed step at a time from the origin to the end of the voxel data. In comparison to the sliding window approach, there are two advantages to the sliding box approach. First, the size of the detection box can be fixed to be equal to that of the bounding box of the target object, since an object's scale is constant in 3D data. Second, each object becomes more likely to be segmented by the detection box, since it moves along not only the x- and y-axes, but also the z-axis. In the sliding window approach in 2D images, the background in the detection window tends to complicate object detection. In the sliding box approach, on the other hand, other objects in both the background and foreground are kept out of the sliding box, even if they appear similar to the target object in the images.

This paper documents the incremental work done since our previous report [4], [5]. The main contribution is the extension of our object classification method [4] and its application to detecting objects in a large environment. We introduce new color 3D features which are improved versions of the features in [4], [5]. Moreover, we propose a method for learning each object's classifier that can be used effectively in an environment containing many unknown objects. To learn such a classifier, the system reduces the weight of common features in the environment and instead increases the weight of unique features observed in the target object's model. This learning process does not require segmenting or labeling objects, other than the target object. In addition, we use a fast sliding box scheme using efficient feature extraction. Finally, constructing a robot system that crawls automatically and collects 3D-scene data in the environment and creating the semi-automatic process of obtaining 3D data of target objects are supplementary contributions.

II. Related Work

The challenge of locating objects using mobile robots has been widely researched in recent years. Viswanathan et al. [6] use the LabelMe database [7] and Kollar et al. [8] the Flickr database to learn the relationship between objects and places in a planned object search. Such a search can be classified as an "indirect search" [9], since the object-search framework makes use of information of other objects or places closely related to the target object. This approach, however, is not successful when the target object is moved to a different place than its usual one. Therefore, we adopt the approach of detecting objects directly in the 3D data of the environment, obtained during the routine crawling of an autonomous mobile robot.

Regarding robot research on direct object search in an environment, [10], [11] use 2D images, while [12], [13] use 3D data. Ma et al. [13] use both photometric information and 3D shape information. As a first step, they narrow down the number of candidates for the target object through a coarse global search using a color histogram, and then they perform a local search with SIFT [1] descriptors in the second step. Although the two-step approach comprising a coarse and fine search, is computationally efficient, the color histogram lacks expressiveness, and therefore, the search may fail if the color of the target object is not salient. In the first step, it is important to reduce the false-positive error, while keeping the false-negative error as low as possible. We focus on increasing the performance of the first step without any increase in computational cost.

The state-of-the-art of automatic 3D reconstruction of indoor environments with color textures has advanced dramatically, both in computer vision with cameras [14], [15] and in robot research with SLAM [16]. Regarding robot applications, colored 3D data of environments can easily be obtained by TOF sensors with associated cameras [17]–[19], which are also used in our work.

There are a variety of well-studied 3D descriptors for recognizing environments. Semantic labeling of 3D points in an environment is discussed in [20], [21]. However, these descriptors, which extract low-level shape patterns, are more suitable for scene description than for specific object detection. In [22] shape primitives are used to detect chairs, although it is difficult to adapt this technique to the detection of objects with arbitrary complex shapes. Regarding detection of 3D free-form objects, [23] uses a Spin Image [3], a well-known rotation-invariant descriptor, while [24] proposed a new descriptor, which is defined by the relative position and orientation of two oriented points. Both these descriptors enable shape description, but do not take color-texture information into consideration. There are also a few descriptors that combine 3D shape and color patterns. Huang and Hilton [25] developed the shape-color histogram, which yields better performance than the shape histogram in recognizing moving objects in multiple-view videos. Nevertheless, although they are effective in applications where the target object can be observed in its entirety, it is difficult to apply shape-color histograms to partial data of objects observed in an everyday environment.

III. System Overview

A diagram of the search system is shown in Fig. 2. The robot routinely crawls around an indoor environment, regularly updating the 3D-scene data of the environment.

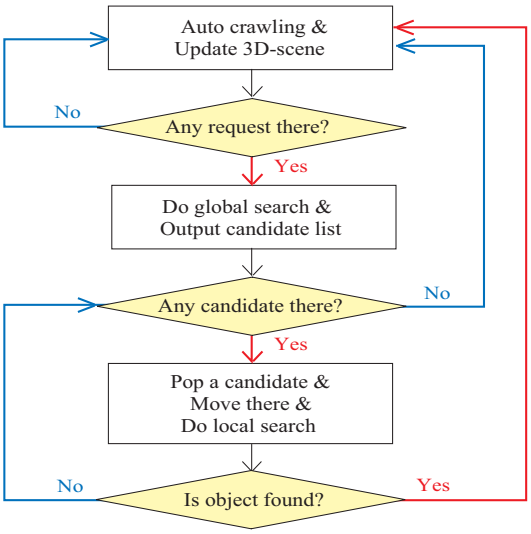

Fig. 2. Search system diagram.

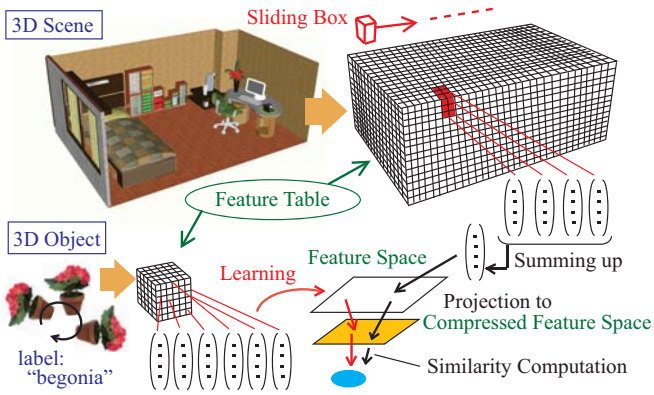

Fig. 3. Overview of object detection, consisting of feature extraction, sliding box, and similarity calculation.

When the system receives a request to locate an object, it executes a global search of the 3D-scene data. In this process, the system performs a sliding-box search to calculate all the similarities between the local regions and the target object, and then outputs a list of regions with higher similarities than a certain threshold. Next, having moved to the first area on the list, in descending order of similarity, the robot searches the area, updates the 3D-scene data, and repeatedly performs object detection. If the target object is not discovered, the robot moves to the next area on the list. If the target object cannot be found in any of the areas on the list, the robot restarts its routine crawling and searches the whole environment.

A. Automatic Routine Crawling in Environment

Our mobile robot has a pan-tilt head at a height of 1.4 meters above the floor, on which a PointGray Flea2[1] camera and a MESA Swissranger SR-4000[2] TOF sensor are affixed parallel to each other (Fig. 1). We equipped the robot with a Hokuyo UTM-30LX[3] LRF sensor located 0.45 meters forward of the robot's center and 0.48 meters above the floor, to create the 2D map of the environment, perform self-localization, and avoid obstacles while carrying out autonomous locomotion. An initial 2D map of the environment is created using CoreSLAM [26] by manually controlling the path of the robot in the environment. Once the map has been created, it is used by the robot for localization while crawling automatically in the environment. The robot navigates along a circular route viewing the outer wall of the environment and avoiding obstacles when the LRF sensor detects adjacency. The robot's motion is defined as the iteration of moving for 3.5 sec, stopping for 1.0 sec, capturing a color image and a range image, and stopping again for 0.5 sec. The motion intervals were appropriately designed for observing the environment in meaningful proportions.

[1]http://www.ptgrey.com/products/flea2/index.asp
[2]http://www.mesa-imaging.ch/prodview4k.php
[3]http://www.hokuyo-aut.co.jp/02sensor/07scanner/utm_30lx.html

B. Creating color voxel data of environment

To match the corresponding points in a color image and range image, the stereo disparity of each point is computed from its depth value in the range image. Then each pair consisting of a color image and a range image is transformed into surface mesh data with color textures. Surface mesh data are automatically obtained by creating edges between two points whose corresponding pixels in the range image are adjacent to each other. Suppose that A(i, j), B$(i+1, j)$, C$(i, j+1)$, and D$(i+1, j+1)$ are neighboring pixels on a range image. There are two ways of creating two triangular faces by connecting these four points; either by connecting A and D or by connecting B and C. The system compares the lengths of segments AD and BC in 3D coordinates, and then selects the shorter. Note that the system does not create a face with an edge longer than 100 mm. The coordinates of vertices in the surface mesh data are transformed to global coordinates using the robot's location.

Finally, all the surface mesh data are registered and transformed into a set of color voxel data using the method described in [4]. Note that the voxels on the surface of each object have RGB values, while those within each object have the empty property.

IV. Object Learning and Detection

An overview of our object detection system is shown in Fig. 3. Color voxel data of an environment are divided into small cubic regions of an $t \times t \times t$ grid, and then feature vectors are calculated per sub-region. The length of the side of a sub-region becomes the step size for the sliding box. When a target object is given, the system finds the maximum length l (mm) of the sides of the target object's bounding box. Let the length of a voxel's side be v mm, and c be an integer value near $l/(vt)$. The detection box is determined to be equal to $c \times c \times c$ sub-regions.

Since our proposed color 3D features have the additive property, the feature vector of the detection box is computed

by summing the $c \times c \times c$ feature vectors computed from corresponding sub-regions. Taking advantage of this additive property, we use efficient feature extraction by preparing a feature table, consisting of feature vectors computed from arbitrary box regions extending from the origin. This idea is a simple extension of the "Integral Image" used in Viola-Jones object detection [27]. We give the details in Sect. IV-C.

The classifier for each target object is learned as a preprocessing step. This semi-automatic learning process requires several different views of the target object shown by the user. We improved the method for classifier learning proposed in our previous work [4], specifically by introducing weighting features as a function of the increasing scale of unique features seen in the target object, as well as the reducing scale of common features seen in most objects. Details are given in Sect. IV-B.

A. Feature extraction and projection-matrix computation

To describe 3D objects, we used the Color Cubic Higher-order Local Auto Correlation (Color-CHLAC) features in [4], [5], computed from color 3D voxel data by measuring the autocorrelation function of colors in two neighboring voxels. The voxel property $\boldsymbol{f}(\boldsymbol{x})$ (where $\boldsymbol{x}$ is the position of the voxel) was defined as a 6-dimensional vector which has L1-norm equality for all variations of RGB values. In this work, we redefine $\boldsymbol{f}(\boldsymbol{x})$ so that it has L2-norm equality and call the proposed new features Circular Color Cubic Higher-order Local Auto Correlation (C^3-HLAC) features.

Let the R, G, and B values of a voxel be $r(\boldsymbol{x})$, $g(\boldsymbol{x})$, and $b(\boldsymbol{x})$, respectively. These values are normalized between 0 and 1. In our previous work [4], [5], $\boldsymbol{f}(\boldsymbol{x})$ was defined as follows:

$$\boldsymbol{f}(\boldsymbol{x}) = [\ \ r(\boldsymbol{x})\ \ 1{-}r(\boldsymbol{x})\ \ g(\boldsymbol{x})\ \ 1{-}g(\boldsymbol{x})\ \ b(\boldsymbol{x})\ \ 1{-}b(\boldsymbol{x})\ \ ]^T$$

if the voxel does not have the empty property; otherwise, $\boldsymbol{f}(\boldsymbol{x})$ becomes a zero vector.

Here, we redefine $\boldsymbol{f}(\boldsymbol{x})$ as follows:

$$\boldsymbol{f}(\boldsymbol{x}) = [\ \ r_1(\boldsymbol{x})\ \ r_2(\boldsymbol{x})\ \ g_1(\boldsymbol{x})\ \ g_2(\boldsymbol{x})\ \ b_1(\boldsymbol{x})\ \ b_2(\boldsymbol{x})\ \ ]^T$$
$$\begin{cases} r_1(\boldsymbol{x}) \equiv sin\left(\frac{\pi}{2} r(\boldsymbol{x})\right), & r_2(\boldsymbol{x}) \equiv cos\left(\frac{\pi}{2} r(\boldsymbol{x})\right) \\ g_1(\boldsymbol{x}) \equiv sin\left(\frac{\pi}{2} g(\boldsymbol{x})\right), & g_2(\boldsymbol{x}) \equiv cos\left(\frac{\pi}{2} g(\boldsymbol{x})\right) \\ b_1(\boldsymbol{x}) \equiv sin\left(\frac{\pi}{2} b(\boldsymbol{x})\right), & b_2(\boldsymbol{x}) \equiv cos\left(\frac{\pi}{2} b(\boldsymbol{x})\right) \end{cases}$$

In this new representation, since the norm of $\boldsymbol{f}(\boldsymbol{x})$ is constant for any color, the distance between colors is even, and thus features can be learned in a proper metric space.

C^3-HLAC features are defined as the summation of $\boldsymbol{f}(\boldsymbol{x})$ ($\sum \boldsymbol{f}(\boldsymbol{x})$) and $\boldsymbol{f}(\boldsymbol{x})$ correlation between two neighboring voxels ($\sum \boldsymbol{f}(\boldsymbol{x})\ \boldsymbol{f}^T(\boldsymbol{x}+\boldsymbol{a})$) over the whole area of the target voxel grid. The displacement vector $\boldsymbol{a}$ has 14 different patterns. Similar to a Color-CHLAC feature vector, a 981-dimensional C^3-HLAC feature vector is obtained by concatenating all the elements in C^3-HLAC features extracted from both the original color voxels and binarized color voxels (see details in [4]). Finally the system carries out Principal Component Analysis (PCA) on the feature vectors and reduces the dimension from 981 to d.

In this work, we compute the PCA matrix by sampling all the feature vectors extracted from sub-regions in the color voxel data of the environment. This projection-matrix is used for compressing feature vectors. In this step, it is possible not only to speed up the similarity calculation, but also to increase the performance of the similarity calculation using whitening [28], the details of which are explained in Sect. IV-B. Note that the projection-matrix is computed only once when the robot observes the environment for the first time, and the system uses the same projection-matrix thereafter.

B. Learning

Learning a good classifier for a certain object is a problem of selecting appropriate features that can separate the object and other unknown objects. An overview of our learning method and similarity calculation is shown in Fig. 4. In the first learning step, the system computes a projection-matrix P through PCA of the features extracted from the environment's color voxel data. Then each principal component axis is divided by the square root of its eigenvalue, an operation known as whitening [28]. Let the transformed axes be P_w. In the next step, the system extracts feature vectors from all the sub-regions in the color voxel data of the target object, with multiple views. Suppose the system is shown K different views of the target object. In order to achieve rotation invariance, the system generates 504 different poses per view by synthetically rotating the initial pose by each 30 degrees for xyz-axes (see details in [4]). Supposing the target object in k-th pose is divided into n_k sub-regions, a total of $N \equiv \sum_k^{504K} n_k$ feature vectors are extracted. Then the system compresses these feature vectors by multiplying P_w and selects d top dimensions. Finally, the system obtains a projection-matrix Q through PCA of the N compressed feature vectors.

To calculate the similarity between the target object and each candidate region in an environment, the system extracts a feature vector $\boldsymbol{z}$ from the current detection box, and then projects it to the target object's feature subspace. In [4], the similarity measure was defined to be the norm of the projected feature vector, based on the Class-Featuring Information Compression (CLAFIC) method [29]. In this work, we weight each axis in the projection-matrix Q by multiplying the square root of its eigenvalue. Let the transformed axes be Q_m. By projecting $\boldsymbol{z}$ to Q_m space, the system obtains the similarity measure called "multiple similarity" [30], which emphasizes the similarity values on the major axes in Q. The definition of the similarity measure in our previous work [4] is given as (1), and the definition in this work as (2).

$$Sim_{previous} = \frac{\|Q^T P^T \boldsymbol{z}\|}{\|P^T \boldsymbol{z}\|} \tag{1}$$

$$Sim_{proposed} = \frac{\|Q_m^T P_w^T \boldsymbol{z}\|}{\|P_w^T \boldsymbol{z}\|} \tag{2}$$

A summary of the proposed similarity calculation is as follows. In the first projection with P_w, the system extracts

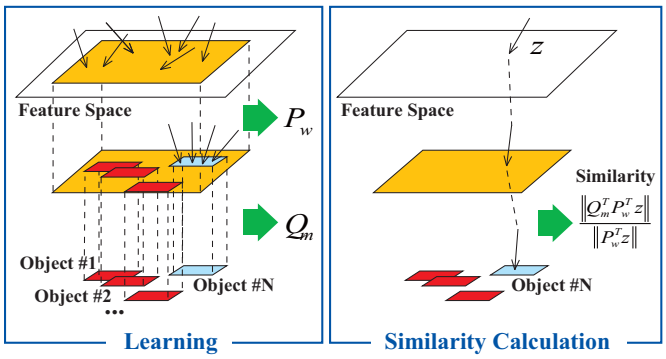

Fig. 4. Illustration of similarity calculation. The P_w represent the principal component axes (divided by eigenvalues) obtained through PCA of the whole environment's features, the Q_m represent the principal component axes (multiplied by eigenvalues) obtained by PCA of the target object's features, and $\boldsymbol{z}$ represents the input feature vector.

principal features that are suited to describing the environment, by suppressing the weight of common features. On the other hand, in the second projection with Q_m, the system selects principal features that can adequately represent the target model, by emphasizing the weight of its primary features. Therefore, the proposed method highlights unique features on the target object, which, in other words, are appropriate features for distinguishing the target object and other objects in the environment.

C. Sliding-box object detection

Viola et al. [27] used an image representation called the "Integral Image" to compute rapidly the summation of pixel values in an arbitrary rectangle. This approach can be applied not only to pixel values' summation, but also to feature vectors represented as the summation of descriptors. In this paper, we call this feature representation the "Integral Feature Table". Since we apply this to voxel data processing, we have extended this approach from 2D to 3D.

The definition of an "Integral Image" is the sum of the image pixels of the upright rectangle stretching from the top left corner to the bottom right corner. In a similar way, the "Integral Feature Table" $\boldsymbol{I}(x, y, z)$ is defined as the d-dimensional compressed C^3-HLAC feature vector extracted from the voxel area ranging from $(0, 0, 0)$ to (x, y, z). Let the feature vector of the voxel area with x ranging from x_1 to x_2, y ranging from y_1 to y_2, and z ranging from z_1 to z_2, be $\boldsymbol{F}(x_1, y_1, z_1, x_2, y_2, z_2)$. This is computed by the following equation:

$$\begin{aligned}\boldsymbol{F}(x_1, y_1, z_1, x_2, y_2, z_2) = \boldsymbol{I}(x_2, y_2, z_2) - \boldsymbol{I}(x_1, y_2, z_2)\\ -\boldsymbol{I}(x_2, y_1, z_2) - \boldsymbol{I}(x_2, y_2, z_1)\\ +\boldsymbol{I}(x_1, y_1, z_2) + \boldsymbol{I}(x_1, y_2, z_1)\\ +\boldsymbol{I}(x_2, y_1, z_1) - \boldsymbol{I}(x_1, y_1, z_1)\end{aligned}$$

Using the "Integral Feature Table", $\boldsymbol{F}(x_1, y_1, z_1, x_2, y_2, z_2)$ can always be computed by adding the 8 cached feature vectors, regardless of the size of the target object. Note that this is not effective when the number of sub-regions included in the detection box is smaller than 8.

Within the large color voxel data of an environment, only the voxels on the surface of each object have RGB values, while other voxels have the empty property. This means that the object detection process can be accelerated by skipping empty regions. Similarly to the "Integral Feature Table", we create a table that stores the number of voxels with the occupied property, in the area ranging from $(0, 0, 0)$ to (x, y, z). Using this table, the number of voxels with the occupied property in the detection box can be computed quickly by adding 8 scalar values. If the number is less than a certain threshold h, the system skips the similarity calculation and moves the detection box forwards. In this work, we set h to the minimum value of the occupied voxel number in the training samples of the target object.

V. Results

A. Experimental Setup

The target environment was our laboratory (Fig. 5(a)), which is 7,950 (length) $\times$ 11,800 (width) $\times$ 2,700 (height) mm. As a pre-processing step, we created a 2D map (Fig. 5(b)) by moving the robot manually around the room, as well as the initial 3D-scene data used for learning the projection-matrix P_w. Note that the robots head was facing outward, and therefore the 3D-scene data consisted of the wall side of the room. To collect 3D-scene test data, the robot moved around the room automatically along a given circular route. In this experiment, we collected 18 different test scenes of the whole room, including 59 target objects (Fig. 7) in different orientations at different locations. Examples of the captured images are shown in Fig. 6.

The robot learned the target objects during a pre-processing step, in which the user displayed them one by one to the robot and then input their labels. Note that the system learns one classifier per object since it performs specific object detection, not the classification of its category. To learn its various aspects, each object was displayed in several orientations. In this learning process, the target object's surface can automatically be segmented from the background by clipping the region with depth in the range of 200 mm of the minimum value.

The parameters were set as follows: the length of the voxel's side v to 10 mm, the size of the sub-regions for learning objects to a $10 \times 10 \times 10$ grid, the size of the sub-regions for creating the "Integral Feature Table" to a $5 \times 5 \times 5$ grid, and the dimension of the compressed feature vector d to 100.

B. Evaluation

To evaluate the effect of the proposed learning method (described in Sect. IV-B), we compared the following four methods:
(a) using P without whitening and Q as CLAFIC,
(b) using P without whitening and Q_m as multiple similarity,
(c) using P_w with whitening and Q as CLAFIC,
(d) using P_w with whitening and Q_m as multiple similarity.
The proposed method in this work is (d), while the method in our previous work [4] is (a).

Fig. 7. Images of 59 target objects, arranged in approximate order of increasing size from top left to bottom right.

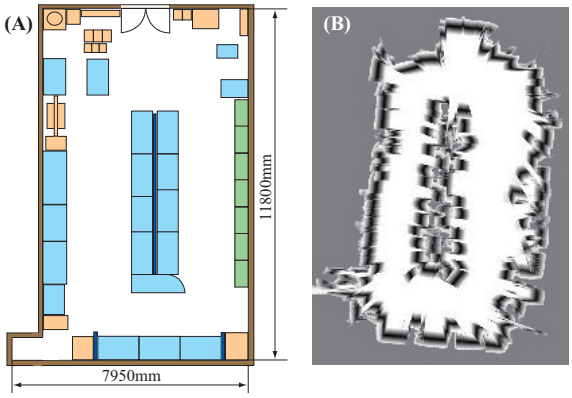

Fig. 5. Target environment: (A) layout of the room, and (B) 2D map created by SLAM.

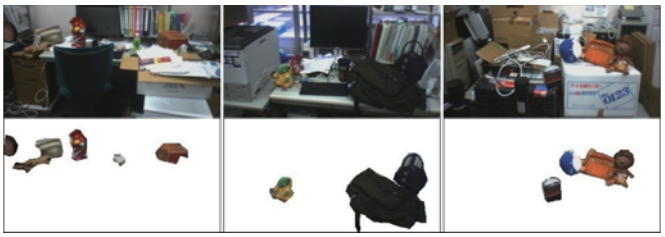

Fig. 6. Examples of Images captured during automatic crawling. Top row shows raw images, and bottom row the ground truth of correct target objects.

In (b) and (d), the dimension of the target object's subspace r, which is the number of axes in Q_m, does not necessarily have to be less than d. This is because the low-rank axes in Q_m have small eigenvalues, and thus, the elimination thereof has no significant effect. However, the computation time for the similarity calculation is faster when r is small. We found experimentally that 20 is an appropriate value of r that does not substantially reduce the accuracy. In (a) and (c), we tested nine choices for r, from 10 to 90, and then chose the one that yielded the highest performance.

A comparison of DET (Detection Error Trade-off) is given in Fig. 8. The vertical axis shows the miss rate, which is the error rate that a correct object was not detected, while the horizontal axis shows the false positive per window (FPPW), which is the error rate that an incorrect object was detected. The closer the DET curve is to the origin, the higher is the performance. As shown in Fig. 8, introducing multiple similarity (method (b)) and whitening (method (c)) achieves better results than the method in our previous work (method (a)). Moreover, the proposed method, which is a combination of multiple similarity and whitening (method (d)), yields the highest performance.

A comparison of the average rate that the correct object was ranked in the top q of the whole environment is shown in Fig. 9. We refer to this rate as the q-rank rate. The q-rank rate represents the probability that when the system outputs the list of q candidate objects, it includes the correct object. For example, if there is a single object somewhere in the room that is being searched for, the system outputs a list of 5 candidates, including the correct one, at a rate of 43.5% using the proposed method. The q-rank rate of each target object in the proposed method is shown in Fig. 10. The average computation time required to detect a single target object in the whole room was 1.96 sec in a single thread, using a Pentium D 3.2 GHz with 6.0 GB main memory.

C. Online trial

After narrowing down the candidate objects through the global search with a sliding box, the robot moves succes-

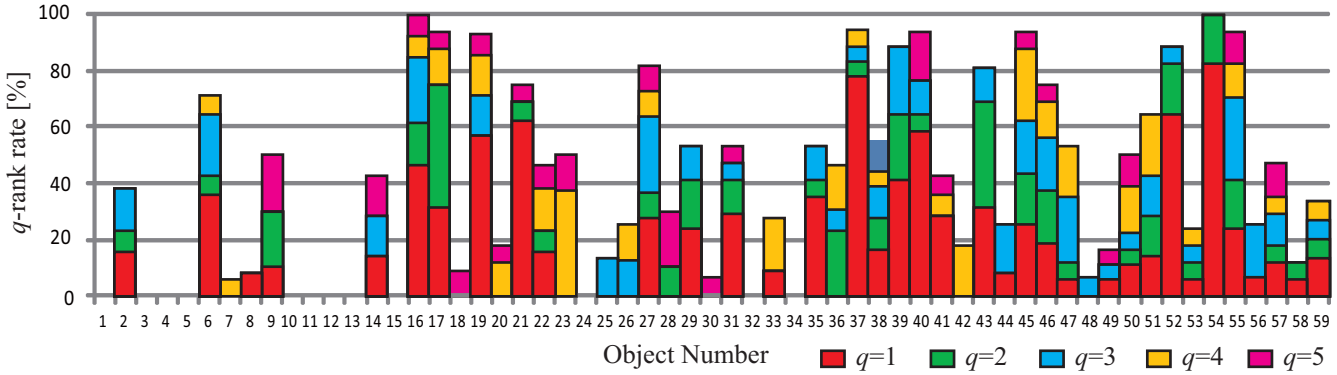

Fig. 10. Average q-rank rate [%] for each object. Each value is shown in red rectangles when q=1, green rectangles when q=2, cyan rectangles when q=3, orange rectangles when q=4, and magenta rectangles when q=5. Each number below rectangles is correspondent to each object's number in Fig. 7.

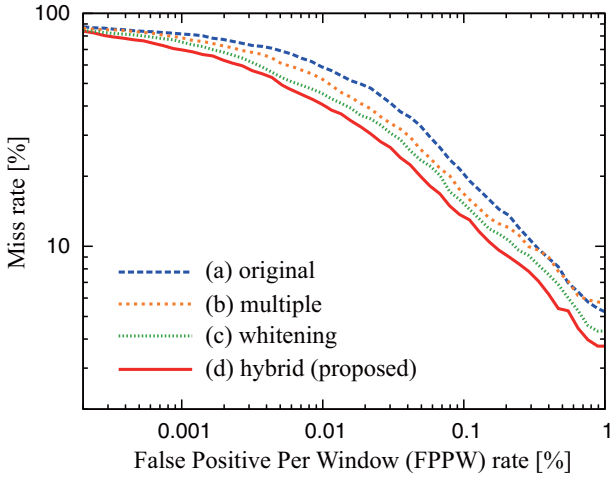

Fig. 8. Comparison of DET.

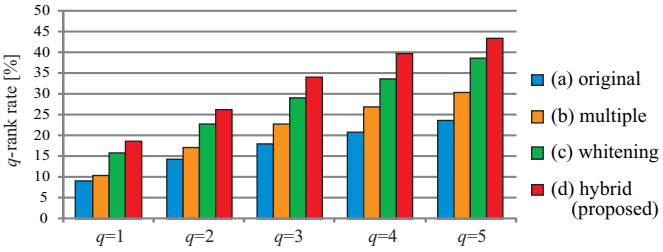

Fig. 9. Comparison of average q-rank rate [%].

sively to each area containing one of the candidate objects, and then looks around for the target object. Considering the aim of this online search from the field of current view, we investigated repeatedly executing all the processing including voxel data reconstruction, feature extraction, and sliding box object detection. The system outputs box regions, whose similarities to the target object are higher than a certain threshold.

Examples of the detection results are shown in Fig. 11. The system basically succeeded in detecting the correct object. On average, the system recorded 2.3 fps in dual thread, using a Core2Extreme QX9300 2.53GHz with 8.0 GB main memory. The results of the online object detection, as well as those of global search, are shown in the attached video.

VI. Conclusion

In this paper, we developed a mobile robot system that performs automatic crawling in an indoor environment, reconstructs color 3D voxel data of the environment, and detects objects through a global search. The system executes sliding box object detection using the color voxel data, where the size of the detection box is fixed to be equal to that of the target object's bounding box. To describe shape-and-color patterns of the color voxel data, we introduced C^3-HLAC features, which are improved versions of the Color-CHLAC features proposed in our previous work. Taking advantage of the additive property of these features, the system can extract features in the detection box very quickly by preparing the "Integral Feature Table".

One of the most significant contributions of this work is the improved similarity measure that includes whitening and multiple similarity. Using this similarity measure, the system highlights unique features on the target object, which are appropriate to distinguish the target object and other objects in the environment. Experimental results show that introducing whitening and multiple similarity increases the detection performance, and that the combination of these is superior to each on its own. The computation time required for sliding box object detection is sufficiently small. All the processing, including voxel data reconstruction, feature extraction, and sliding box object detection, can be performed online at 2.3 fps, if the target environment is confined to the field of current view.

Our proposed object detection scheme aims to achieve a fast global search, and therefore, avoids strict geometry matching between objects in a scene and the reference model of the target object. Implementing geometry matching as the second step in the proposed method, after narrowing down the candidate objects, remains one of our most important future objectives.

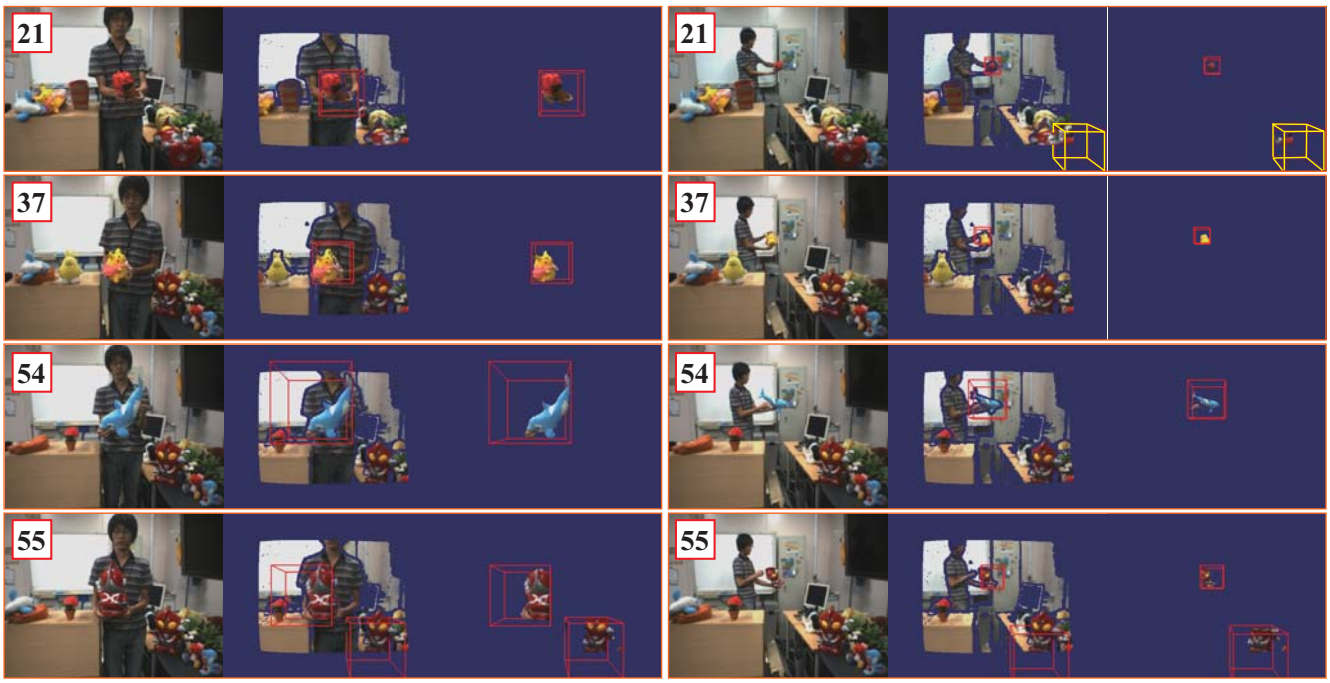

Fig. 11. Examples of image outputs during the online search from the field of current view for each target object, #21, #37, #54, and #55 in Fig. 7. Captured color images are shown in left, 3D-scene images in middle, and object-detected boxes in right. True positive samples are denoted by red boxes and false positive samples by yellow boxes.

REFERENCES

[1] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE ICCV*, 1999.

[2] B. Herbert, T. Tinne, and V. G. Luc, "Surf: Speeded up robust features," in *Proc. ECCV*, 2006.

[3] 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.

[4] A. Kanezaki, H. Nakayama, T. Harada, and Y. Kuniyoshi, "High-speed 3D object recognition using additive features in a linear subspace," in *Proc. IEEE ICRA*, 2010.

[5] A. Kanezaki, T. Harada, and Y. Kuniyoshi, "Partial matching of real textured 3D objects using color cubic higher-order local auto-correlation features," *The Visual Computer*, vol. 26, no. 10, pp. 1269–1281, 2010.

[6] P. Viswanathan, D. Meger, T. Southey, J. J. Little, and A. Mackworth, "Automated spatial-semantic modeling with applications to place labeling and informed search," in *Proc. Canadian Conference on Computer and Robot Vision (CRV)*, 2009.

[7] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *Int. J. of Computer Vision (special issue on vision and learning)*, vol. 77, pp. 157–173, 2008.

[8] T. Kollar and N. Roy, "Utilizing object-object and object-scene context when planning to find things," in *Proc. IEEE ICRA*, 2009.

[9] L. E. Wixson, "Gaze selection for visual search," Ph.D. dissertation, Department of Computer Science, Univ. of Rochester, 1994.

[10] Y. Ye and J. K. Tsotsos, "Sensor planning for 3D object search," *Comp. Vision and Image Understand*, vol. 73, pp. 145–168, 1999.

[11] S. Ekvall, D. Kragic, and P. Jensfelt, "Object detection and mapping for service robot tasks," *Robotica*, vol. 25, no. 2, pp. 175–187, 2007.

[12] F. Saidi, O. Stasse, and K. Yokoi, "A visual attention framework for search behavior by a humanoid robot," in *Proc. IEEE Humanoid Robots*, 2006.

[13] J. Ma and J. W. Burdick, "A probabilistic framework for stereo-vision based 3D object search with 6D pose estimation," in *Proc. IEEE ICRA*, 2010.

[14] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Reconstructing building interiors from images," in *Proc. IEEE ICCV*, 2009.

[15] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *Proc. IEEE CVPR*, 2010.

[16] B. Pitzer, S. Kammel, C. DuHadway, and J. Becker, "Automatic reconstruction of textured 3D models," in *Proc. IEEE ICRA*, 2010.

[17] B. Huhle, P. Jenke, and W. Straser, "On-the-fly scene acquisition with a handy multi-sensor system," *Int. J. Intelligent Systems Technologies and Applications*, vol. 5, 2008.

[18] S. Todt, C. Rezk-Salama, A. Kolb, and K. D. Kuhnert, "GPU-based spherical light field rendering with per-fragment depth correction," *COMPUTER GRAPHICS forum*, vol. 27, pp. 2081–2095, 2008.

[19] A. Kolb, E. Barth, R. Koch, and R. Larsen, "Time-of-flight sensors in computer graphics," in *Proc. Eurographics*, 2009.

[20] A. Nuchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, pp. 915–926, 2008.

[21] R. B. Rusu, Z. C. Marton, N. Blodow, A. Holzbach, and M. Beetz, "Model-based and learned semantic object labeling in 3D point cloud maps of kitchen environments," in *Proc. IEEE IROS*, 2009.

[22] J. Shin, S. Gachter, A. Harati, C. Pradalier, and R. Siegwart, "Object classification based on a geometric grammar with a range camera," in *Proc. IEEE ICRA*, 2009.

[23] S. Ruiz-Correa, L. G. Shapiro, and M. Meila, "A new signature-based method for efficient 3-D object recognition," in *Proc. IEEE CVPR*, 2001.

[24] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Proc. IEEE CVPR*, 2010.

[25] P. Huang and A. Hilton, "Shape-colour histograms for matching 3D video sequences," in *Proc. IEEE ICCV Workshops*, 2009.

[26] B. Steux and O. E. Hamzaoui, "Coreslam : a slam algorithm in less than 200 lines of c code, accepted for the international conference on control, automation, robotics and vision," in *Proc. IEEE ICARCV, to appear*, 2010.

[27] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE CVPR*, 2001.

[28] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford Clarendon Press, 1995.

[29] S. Watanabe and N. Pakvasa, "Subspace method in pattern recognition," in *Proc. 1st Int. J. Conf on Pattern Recognition*, 1973.

[30] T. Iijima, *Theory of Pattern Recognition (in Japanese)*, ser. Basic Information Technology, Vol. 6. Morishita Publishing, 1989.