2015 Seventh International Conference on Knowledge and Systems Engineering

3D Object Finding Using Geometrical Constraints on Depth Images

Van-Hung Le[1,2], Hai Vu[2], Thuy Thi Nguyen[3], Thi-Lan Le[2], Thi-Thanh-Hai Tran[2], Michiel Vlaminck[4], Wilfried Philips[4] and Peter Veelaert[4]

[1]Faculty of Information Technology, College of Statistical, Vietnam, hunglv@gso.gov.vn
[2]International Research Institute MICA, HUST - CNRS/UMI-2954 - GRENOBLE INP, Vietnam
[3]Faculty of Information Technology, VietNam National University Agriculture, Vietnam
[4]Ghent University/iMinds - Image Processing and Interpretation, Belgium

***Abstract*—Finding an object in a 3D scene is an important problem in the robotics, especially in assistive systems for visually impaired people. In most systems, the first and most important step is how to detect an object in a complex environment. In this paper, we propose a method for finding an object using geometrical constraints on depth images from a Kinect. The main advantage of the approach is it is invariant to lighting condition, color and texture of the objects. Our approach does not require a training phase, therefore it can reduce the time of preparing data and learning model. The objects of interest have a simple geometrical structure such as coffee mugs, bowls, boxes and are on a table. Overall, our approach is faster and more accurate than methods using 2D features on depth images for training an object model.**

Keywords*—*3D object detection, depth data, Kinect, Geometrical constraint

I. Introduction

Finding an object is a fundamental problem in the field of scene understanding. It is applied broadly in robotics and supporting systems for visually impaired people. There are three main steps for the object finding problem: detecting, classifying and localizing objects in the image. The problem has been widely investigated in both 2D and 3D spaces. However, the real world is hard to be represented in 2D space, whereas detecting objects in 3D space has many challenges such as matching and learning visual objects. The problem depends on object pose, camera viewpoint, partial occlusions, etc. Since its release, the Kinect [1] has become popular and as results it has been the subject in many researches. The device produces both color and depth information and makes it, hence possible to model the environment in a more realistic way. Nevertheless, each object has its own unique shape in 3D space. This shape is at any time only partially visible in the point cloud. The reasoning is therefore performed on some surfaces of the object, each surface containing a set of points. As a result, it is essential to use the geometrical structure of the objects for our problem. Currently, there are two main approaches for 3D object finding including appearance-based and geometry-based [2]. The first approach is based on training feature points of the objects of interest [3], [4], [5]. This approach needs a lot of time to prepare the data for learning the object model and to perform the training. The second approach exploits the geometrical structure of the objects [6]. Some geometrical constraints can be used for certain types of objects in the scenes.

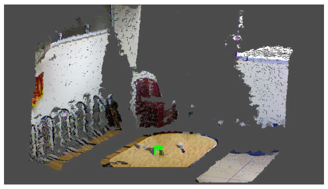


Fig. 1. Cup detection in a particular scene. It illustrates the simplest case. On the table there is only one object, being the cup. This detection is the first step in the process of finding objects in 3D space (preceding classification and localization).

Visually impaired people have a lot of difficulties in daily activities especially in finding and grabbing the object of interest in living environment. With the advance of imaging device such as kinect, developing a reliable system for assisting visual impaired people becomes possible.
In this paper, we propose a new approach for finding objects of interest in a complex scene in order to build an assistive system for visually impaired people. We aim at detection of objects of interest based on a query of impaired person, wherein, the 3D object coordinates of the object are on a table. The result can be emitted to a speaker to notify the impaired person about object localization and number of object on the table. The approach is based on geometry using data from one Kinect. Our approach only deals with objects that have a simple geometrical structure such as cups, bowls, boxes. The coffee mug, bowl or cup can be represented by a cylinder while the box is represented by a cuboid. Fig. 1 illustrates the result of the first step for cup finding, i.e. cup detection.

II. Related work

Object detection is a fundamental problem in computer vision. In this field, detection of geometrical primitives has been studied for years in 2D as well as 3D. In 2D, object detection is often based on lines, ellipses, circles [7] and arbitrary shapes [8]. In 3D, it is often based on planes, cylinders, spheres [6]. Recently, the majority of the researches on 3D object detection use appearance-based methods. Bo et al. [9], [10], [11] presented and improved a family of kernel descriptors (KDES). This approach provides a principled framework for extracting image features from pixel attributes such as gradient, color, local binary patterns, etc. This approach is performed on 2D

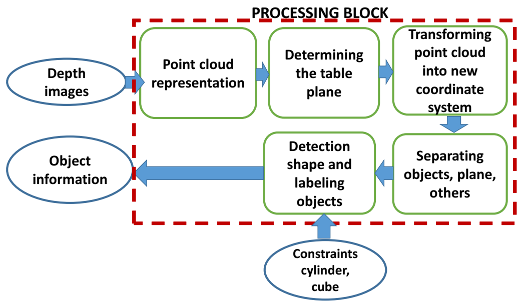

PROCESSING BLOCK
Depth images
Point cloud representation
Determining the table plane
Transforming point cloud into new coordinate system
Separating objects, plane, others
Detection shape and labeling objects
Object information
Constraints cylinder, cube

Fig. 2. The framework for finding an object in 3D

images (RGB, depth image) and 2D features such as SIFT, SURF, LBP. Also in [11], Bo et al. have presented some local features using depth kernel descriptors (gradient, kernel PCA, size, spin and local binary pattern kernel descriptors) for object recognition. It can capture different cues for recognition, including size, shape and edges (depth discontinuities). The authors used a Support Vector Machine (SVM) to train the features (color histograms, SIFT, gradient, and local binary pattern descriptors) on RGB-D images. It has a relatively high accuracy in object recognition. More specifically, it has an accuracy of about 80% on the dataset presented in [12]. However, this approach needs a lot of time for preparing data and learning the object model, i.e. training a classifier given a set of positive and negative examples [2].
Closer to the real world are the researches that represent the image pixels into 3D points (point cloud) in the real world. 3D object detection is performed on point clouds and uses 3D feature points for the object detection. All feature points are calculated on the coordinates of each point in 3D space such as PFH (Point Feature Histogram) [13], FPFH (Fast Point Feature Histogram) [14], VFH (Viewpoint Feature Histogram)[15], CVFH (Clustered Viewpoint Feature Histogram) [16], Intrinsic Shape Signature (ISS) [17]. These features normally are based on the distance and the normal vector of each point on the surface of the object.

III. PROPOSED METHOD

A. Framework

Our work presented in this paper is based on a simple scenario: a visually impaired person wearing a Kinect is going to a kitchen to find and grab an object on a table. The objects have a simple geometrical structure such as cup or box. We propose a framework for finding a simple object based on point clouds generated from depth images from a Kinect. The framework is shown in Fig. 2.

B. Finding an Object

Step 1: Point cloud representation
In this step, the depth data is converted into 3D coordinates. 2D depth image D(x,y) is represented in 3D coordinates in which (x,y) is spatial pixel (from RGB image) and z data corresponds to depth. To represent the data, we use a function of the PCL library (Point Cloud Library). We used a transformation matrix to construct the point cloud data from a depth image. The parameters are obtained from the calibration of the Kinect. Each point of the point cloud data has thus a 3D coordinate (x, y, z). Using the depth camera intrinsic [1], each pixel (x_d, y_d) having depth value $depth(x_d, y_d)$ on the depth image can be projected into metric 3D space using the following formula:

$$x = \frac{z \cdot (x_d - c_x)}{f_x} \quad y = \frac{z \cdot (y_d - c_y)}{f_y} \quad z = depth(x_d, y_d) \tag{1}$$

with f_x, f_y, c_x and c_y respectively the focal length and principal point. Then in order to reduce the size of the data, we present a point in point cloud by means of a voxel grid.
Step 2: Determining the table plane
In [1], the authors bases on the assumption: the table plane is the largest in the scene. In this paper, we also use this assumption. The table plane detection is based on a plane extraction algorithm such as RANSAC, one of its variants, or 3D hough transform. In our approach, we perform a clustering of the point cloud into so called "region point clouds" prior to the actual extraction of the table plane. We use PROSAC (Progressive Sampling and Consensus) [18] for fitting the largest plane. We have evaluated different RANSAC variants and chosen PROSAC because of its performance in accuracy and in computational time. With the similar accuracy, the total gain in processing time compared to RANSAC is 2.364s. The details are presented in Fig. 3.

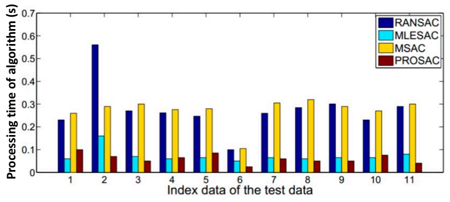


Fig. 3. The processing time of RANSAC and its variants; Our data is 11 point sets of point cloud, chosen from "MICA" dataset.

Step 3: Transforming point cloud data into new coordinate system
Step 2 allows one to limit the search space to objects on the table. In this step, in order to facilitate the computation and the use of geometric constraints, we transformed from the Kinect coordinate system ($O_k x_k y_k z_k$ - origin at Kinect center) into a new coordinate system centered on the table (x,y on table plane, z is the normal vector of table plane). It allows to filter points in which we are not interested and lets us focus on the data on top of the table. In addition, it allows us to use geometrical constraints such as the height of the object. Using the new coordinate system, the height of the object corresponds to the value of the z-coordinate. The processing is as follows: The rotation matrix are:

$$R_x(\alpha) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{vmatrix} \tag{2}$$

$$R_y(\beta) = \begin{vmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{vmatrix} \tag{3}$$

$$R_z(\gamma) = \begin{vmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (4)$$

where α, β, γ are rotation in x, y, z axis
From Eq. 2, Eq.3 and Eq.4, we have the rotation matrix:

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha) \qquad (5)$$

In this paper, we only rotate in the x and y axes. Rotation in y axis by an angle β, and in x axis by an angle α.

$$\beta = \arcsin\frac{a}{\sqrt{(a^2+c^2)}} \quad \alpha = \arcsin\frac{b}{\sqrt{(b^2+c^2)}} \qquad (6)$$

where (a,b,c) is the normal vector of table plane. The new system coordinate after rotation is $(0, 0, Z_N)$. In real data, (x,y) coordinate is very small about 10^{-7}.
Now, we need perform a translation in z axis by a term

$$d = |Z_N|\,. \qquad (7)$$

Step 4: Separating objects, plane, others
First we extract points which could belong to relevant objects based on their distance to the table plane. Specifically, we require $z_i > 0$ (the points should be above the table plane) and $z_i < T$ (the points should be close enough to the table). Moreover, we assume that points with $z_i < t$ (very close to the table) belong to the table and not the objects. Thus m points in the scene are divided into two sets: the first set containing n points belonging to the table plane and the second one consisting of $m - n$ points belonging to objects. After the separation of planes and objects, we perform a clustering of objects based on the Euclidean distance of points. The distance between two point data regions is smaller than $Radius$. We use Kd-Tree [19] for finding the K nearest neighbors of a specific point or location. This algorithm for clustering objects based on the relative position of points is briefly presented in algorithm 1. In this algorithm,

Algorithm 1: Clustering point clouds

```
1 $Region = Nothing; i = 0;$
2 while $length(ListPoint) \neq 0$ do
3     ProcedureSearchKNeastNeigh   $Radius, Region$ ;
4     if $length(Region) \geq ThresSizeRegion$ then
5         CreateRegion($Region$);
6         $ListPoint = ListPoint - Region$;
7     $i = i + 1$;
8     $Return \quad Region$;
```

$ThresSizeRegion$ is the minimum size of the data region. It is the size of projection data on table plane. The threshold may be used to remove the noisy data. $ListPoint$ contains all the points of the objects. $ProcedureSearchKNeastNeigh$ searches for the K nearest neighbors within a $Radius$ and the output of it is a set point belonging to a certain region.
Step 5: Detection of shape and labeling objects
In section I, we mentioned the presentation of interested 3D objects by primitives in 3D space: a cup and a bowl are represented by a cylinder while the box is represented by a cuboid. The constraints for detecting a cylinder in a point cloud are as follows: We fit a cylinder of radius r around the line from the point $A = (x_1, y_1, z_1)$ to point $B = (x_2, y_2, z_2)$. Fig. 4 illustrates the representation of a cylinder.
In this case, point B is defined as follows. A projection of all object points are performed onto the table plane. Point $B = (x_2, y_2, z_2)$ is the circle center and r is the radius of the circle and point B is the location of objects on the table. Since the origin of the coordinate system is centered on the table plane the z axis represent the height of objects. The point $A = (x_2, y_2, z_1)$ and $z1$ is the maximum z value of the object point cloud. Normally, the projection of data of a cup on the table plane is an ellipse. However, in case of incomplete data (the density of data follows a conic shape), we fit an ellipse through the existing points. This ellipse can be too large and the according error function cost is large. This is illustrated in Fig. 5. To 3D object coordinate on the table plane, we fit a circle to the projection data in procedure $GetLocationCup(Center, r), GetLocationCube(Center, rb)$ in algorithm 2. Through each object point cloud a cylinder is fitted and dis values are distance from points to the round axis of cylinder. In the real world, the height of the cup is about h and the radius of the cup about r_c and the threshold $t_{inlier}\%$ is the threshold to determine if a point is an inlier, i.e. belongs to the cylinder model.
The constraints for detecting cuboid in point cloud are as follows: Boxes are represented by their planar surfaces. Normally, a box has 2 or 3 visible sides, but in worst case situations only one sole side is visible. Let the biggest dimensions of the box be b_w and b_h. Let $\vec{n}_1$ be the normal vector of the largest plane belonging to the box and $\vec{n}_2$ the surface normal of the second largest plane. For a cuboid we should have $\vec{n}_1 \times \vec{n}_2 = \vec{0}$. Fig. 6 illustrates the three view cases of the box. The sides are fitted into a cuboid with size

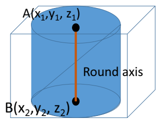


Fig. 4. Construction a cylinder from point A, point B, radius r

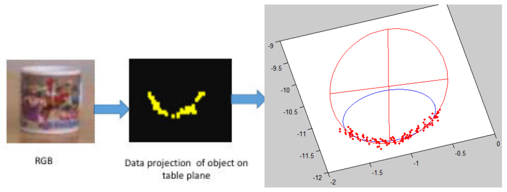


Fig. 5. Fitting the wrong ellipse. The red ellipse and blue circle use the least squares algorithm fitting for projection data on the table.

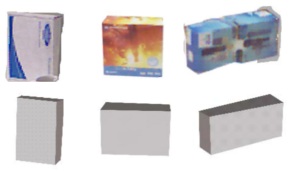


Fig. 6. From left to right, the view of Kinect can see one side, two sides, three sides of the box

Fig. 7. Setup of experiments for collecting dataset

(b_w, b_h, b_w). b_w is the width of cuboid, b_h is the height of cuboid. The evaluation of the inlier rate is similar to fitting a cylinder, but this time a point belongs to the cuboid model if it belongs to nearest the plane. In summary, fitting a cuboid comes down to fitting points into two or three planes, and verifying if these planes are perpendicular to each other. All constraints used for 3D object finding in point cloud are summarized in algorithm 2:

Algorithm 2: Using geometrical constraints for 3D object finding

```
Input: Multi region point cloud
Output: Label cup or box, location of each object on the table
1  $i = 0$;  $Label[i] = 0$;
2  For each region point cloud
3  if $HeighObject \leq ThreHeObject$ then
4      if $HeighObject \leq ThreHeObjectCup$ then
5          ProcedureFitCylinder (AB-axis, rc, dis);
6          if ProcedureCheckCup($h, rc, dis, tinlier$)==true then
7              $Label[i] = 1$;
8              GetLocationCup($Center, r$);
9      else
10         $NumberPlane$ = ProcedureFitPlane(region-point-cloud);
11         $checkcuboid$=ProcedureInlierCuboid($b_w, b_h, b_w, tinlier$);
12         if $NumberPlane == 1$ then
13             if $size(plane) \geq size(b_w, b_h)$ and $checkcuboid == true$ then
14                 $Label[i] = 2$;
15                 GetLocationCube($Center, rb$);
16         else
17             $check$ = ProcedurePerpendicular2Plane($plane$);
18             if $size(plane) \geq size(c_w, c_h)$ and $check == true$ and $checkcuboid == true$ then
19                 $Label[i] = 2$;
20                 GetLocationCube($Center, rb$);
21 $i = i + 1$
22 Return $Label, LocationofObject$
```

IV. Experimental result and discussion

A. Experiment

Fig. 7 illustrates the experimental setup we used to evaluate the proposed approach. We suppose the impaired person to determine the table localization in the room. The focuse is to look for objects on the table. In our data set for experiments, Kinect is mounted on ordinary people and moved near the table. We performed two different experiments. The first experiment was performed in a kitchen near the lab of MICA. We refer to this as the "MICA" dataset. In this experiment, a Kinect was attached to a person's torso. As a result the Kinect was about 60cm above the table plane. The distance between the Kinect and a cup on the table was about 150cm. On the table we had placed some coffee mugs, bowls (cups) and boxes and other objects such as knives and a remote control of TV. The cups all have a cylindrical structure and differed in size (height from 5 to 10 cm, radius from 3 to 5 cm). The boxes have a minimum size about 10×10cm. Our dataset contains 5 scenes (see Fig. 9), in which there are 910 depth images. Both the depth and RGB sensors have been calibrated by MICA platform. The distance between the objects is about 5 to 10cm. The table is put in the center of the room and in other cases near the wall. In some sequences we placed chairs next to the table. The Kinect captured about 3 frames per second. Because Kinect usually collect data with speed 15 Hz - 20 frames per second, but in MICA platform that it takes time for the calibration. The depth images have a resolution of 640×480 pixels. The subject moves around the table in Fig. 8. The second dataset for experiment was captured in TELIN lab at UGhent. We refer to this as the "TELIN" dataset. The experiment is similar to the first experiment. But in this experiment, the table type is different and other objects such as (bottle, clothing, computer screen, keyboard) are on the table. The TELIN dataset has 9 scenes (see Fig. 10) and 705 depth images. We used the following thresholds: The maximum height of objects and boxes is $ThreHeObject = 30cm$; the

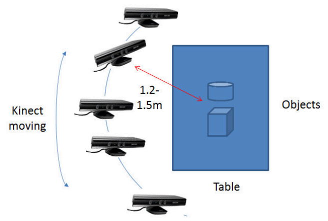


Fig. 8. Person move around the table

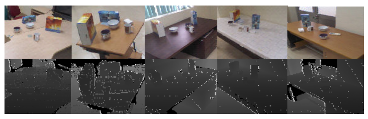


Fig. 9. Some scenes of experiment 1

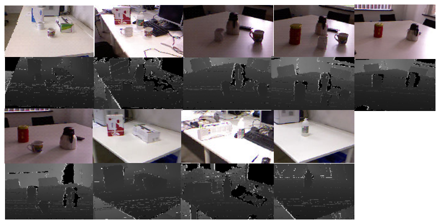


Fig. 10. Some scenes of experiment 2

Scenes	Frames	Objects		True detection		False detection	
		Cups	Boxes	Cups	Boxes	Cups	Boxes
1	230	960	460	893	428	75	53
2	202	606	404	576	384	152	32
3	136	408	272	378	252	65	94
4	177	531	354	502	335	98	46
5	166	498	332	471	314	81	32
Total	**911**	**3003**	**1822**	**2820**	**1713**	**471**	**257**

TABLE I. THE RESULT OF 3D OBJECT DETECTION FOR MICA DATASET

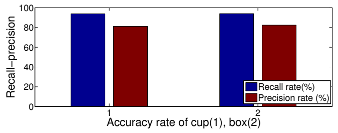


Fig. 11. Precision-recall rate of 3D object detection on MICA dataset (table I); precision rate of cup is 81.2%, box is 82.4 %

maximum height of cups is $ThreHeObjectCup = 10cm$; the maximum of cups radius is $rc = 5cm$ and the maximum width of boxes is $w_b = 20cm$; the size of the biggest visible box plane is maximally $size(b_w, b_h) = 40 \times 30cm$ and the size of the second biggest visible plane is $size(c_w, c_h) = 40 \times 10cm$ and threshold of inlier rate is 80%.

B. Results

We define a true detection of an object as follows: the data points of a cup or box are correctly labeled as such for all points lying on it. A false detection of an object is defined as follows: the region data is part of a cup or a box, but it is not detected or the region data is not an object (cup, box), but it remains to be discovered as a cup or box (false detection). Regarding the location of the objects on the table, the accuracy depends on all the sub processes: point cloud representation (see step 1), point cloud transformation (see step 2), object data projection onto table plane, circle fitting for each data region (see step 5). In this study, we did not evaluate the accuracy of the object location. The evaluation is based on the number of objects that are correctly detected (true detection) and the number of objects that are not detected or wrongly detected (false detection). The results of experiment 1 are summarized in table I and the average accuracy evaluated by Eq. 8 in [20] is shown in Fig. 11.

$$Recall = \frac{TP}{TP + FN} \quad Precision = \frac{TP}{TP + FP} \qquad (8)$$

The result of false detection is high due to several reasons: in scene 2,4,5, the size of small boxes and cups is similar and the noise of cups leads to false cup classification; the noise of objects can lose one part of the object data; the noise of the data can make objects stick together leading to a wrong clustering; some scenes that have a table near the wall wrongly cluster the table data and the wall data. A part of wall data may be detected as a box. But the average accuracy (precision) of 3D cup detection is 81.2% and 3D box detection is 82.4%. Results of experiment 2 are presented in Tab. II and the average accuracy is illustrated in Fig. 12. In Tab. II, the false detection, often due to wrongly clustering table and chairs or a part of the printer and the average accuracy (precision) of 3D cup detection is 80.15%, 3D box detection is 67.31 %.

Scenes	Frames	Objects		True detection		False detection	
		Cups	Boxes	Cups	Boxes	Cups	Boxes
1	96	192	192	178	180	0	25
2	81	162	81	132	67	32	43
3	129	258	0	256	0	9	21
4	96	192	0	181	0	8	19
5	46	0	0	0	0	4	7
6	49	49	45	0	0	8	5
7	75	0	150	0	150	0	15
8	96	0	96	0	91	17	14
9	36	0	0	0	0	1	12
Total	**704**	**853**	**564**	**747**	**488**	**79**	**161**

TABLE II. THE RESULT OF 3D OBJECT FINDING IN TELIN DATASET

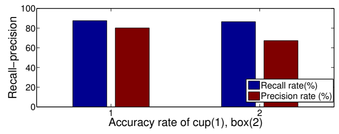


Fig. 12. Precision-recall rate of 3D object detection in TELIN dataset (table II); precision rate of cup is 80.15%, box is 67.31 %

Scenes	Frames	Objects		True detector		False detector	
		Cups	Boxes	Cups	Boxes	Cups	Boxes
1	888	2664	0	2314	0	342	0
2	834	1668	834	1452	756	135	142
3	861	2583	0	2350	0	417	0
4	868	2604	868	2318	783	384	137
5	1128	1926	1028	1628	946	276	136
6	1048	2984	0	2634	0	579	0
7	943	1626	903	1492	871	294	200
8	925	763	0	704	0	98	0
9	732	612	732	552	703	84	117
10	716	1432	0	1393	0	173	0
11	640	0	640	0	589	13	86
12	723	723	0	685	0	100	8
13	462	724	362	637	327	151	84
14	659	1259	0	1191	0	186	32
Total	**11427**	**21568**	**5367**	**19350**	**4975**	**3232**	**942**

TABLE III. THE RESULT OF 3D OBJECT FINDING IN RGB-D SCENES DATASET V.2 [12]

To demonstrate that the results of our approach are better, we evaluated and compared our approach using the RGB-D Scenes Dataset v.2 [12]. This dataset contains visual and depth images of 300 physically distinct objects taken from multiple views. The objects are commonly found in indoor environments. Interested objects are on the table such as a cup, a box, a hat, etc. The dataset is collected using a sensing device consisting of a prototype RGB-D camera, Prime-Sense and a firewire camera from Point Grey Research. The cameras collect both RGB and depth images and the resolution of the frames is 640x480 pixels. The two cameras are calibrated using the Camera Calibration Toolbox for Matlab [21]. This dataset has 14 scenes.
In the experiments, we have converted the video of each scene to individual frames and only evaluated the detection of cups and boxes. In some scenes, the table plane is not the largest plane in the scene. Since we rely on this, we could not use these frames in our evaluation. We evaluated our method using this dataset the same way as we did in experiment 1. We have grouped the coffee mug and bowl into the same class of cup. The results are presented in Tab. III and the average accuracy is shown in Fig. 13. In [12], the authors used a linear support vector machine (SVM) for training feature image from both

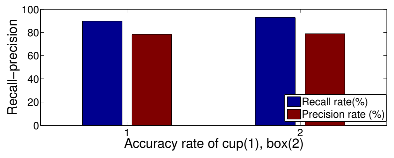
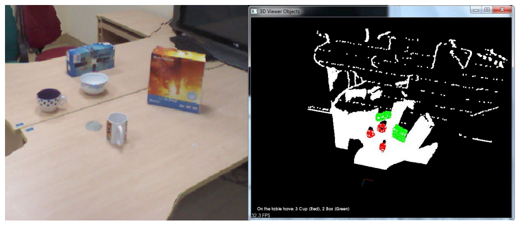


Fig. 13. Precision-Recall of object classification in RGB-D Scenes Dataset v.2 [12]; The average accuracy (precision) of 3D cup detection is 78.02%, 3D box detection is 78.86%

Approach	**Average accuracy (precision)**
Baseline	**74.9%**
Our approach	**78.44%**

TABLE IV. THE AVERAGE ACCURACY ON RGB-D SCENES DATASET V.2 [12] OF OUR APPROACH AND BASELINE APPROACH [12], [23]



On the table have: 3 Cup (Red), 2 Box (Green)

Fig. 14. Result of finding cup and box in the scene

RGB and depth image. The feature is a variant of histogram of oriented gradients (HOG). Object classification is based on the standard sliding window approach (Deformable Part Mode-DPM [22]). In [12], [23] (baseline approach) are evaluated object classification by the approach [22] on depth images of RGB-D Scenes Dataset v.2 [12]. The average accuracy was compared with our approach in Tab. IV.
Our system is implemented in the C++ language using the combination of PCL 1.7 and OpenCV 2.4.8. Out method has been tested on PC with Core i3 processor - RAM 4G. The computational time is 1 scene per second whereas in [12] dataset, the implementation takes approximately 10 seconds to run the four object detectors to label each scene. Our approach is faster and more accurate than the baseline approach. Fig. 14 illustrates the result of finding cup and a box in the scene.

V. CONCLUSION

In this paper, we have presented a new approach for 3D object finding using depth images generated from a Kinect sensor. The proposed method uses some geometrical constraints of the scene and the objects. The approach is different from the appearance-based approach because it does not rely on the learnt model. The obtained experimental results show that by using some geometrical constraints we can achieve a higher accuracy than appearance-based training of an object model to detect objects of interest.

ACKNOWLEDGEMENT

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number FWO.102.2013.08 and VLIR project ZEIN2012RIP19.

REFERENCES

[1] Jeff Kramer, Nicolas Burrus, Florian Echtler, Herrera C. Daniel, and Matt Parker. Hacking the Kinect. *Apress*, 2012.

[2] Sherif Barakat Khaled Alhamzi, M. E. 3D Object Recognition Based on Image Features: A Survey. *International Journal of Computer and Information Technology*, Volume 03 – Issue 03:pp651–660, 2014.

[3] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Detection-based object labeling in 3D scenes. *2012 IEEE International Conference on Robotics and Automation*, pages 1330–1337, ISSN :1050–4729, May 2012.

[4] Shuran Song. Sliding Shapes for 3D Object Detection in Depth Images. *ECCV2014, pp 634-651, Volume 8694*, 2014.

[5] J. Rusu, R.B. ; Willow Garage; Bradski, G. ; Thibaux, R. ; Hsu. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *IROS*, 24(4):345–348, August 2010.

[6] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[7] R. O. Duda and P. E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Comm. ACM, Vol. 15, pp. 11-15(January, 1972)*, 1972.

[8] D.H. Ballard. Generalizing the Hough Transform to Find Arbitrary Shapes. *CVGIP 13, 111–122*, 1981.

[9] Xiaofeng Ren Bo Liefeng and Dieter Fox. Kernel Descriptors for Visual Recognition. *NIPS. Vol. 1. No. 2.*, 2010.

[10] Xiaofeng Ren Dieter Fox. Liefeng Bo, Kevin Lai. Object Recognition with Hierarchical Kernel Descriptors. *CVPR ,Robotic Res. 33(4): 581-599*, 2011.

[11] Xiaofeng Ren Dieter Fox. Liefeng Bo, Kevin Lai. Depth kernel descriptors for object recognition. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[12] X. Ren K. Lai, L. Bo and D. Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. *ICRA,DOI:10.1109/ICRA.2011.5980382*, 2011.

[13] Zoltan Csaba. Blodow Nico. Rusu Radu Bogdan, Marton and Michael Beetz. Learning Informative Point Classes for the Acquisition of Object Model Maps. *ICARCV*, pages pp643 – 650, DOI:10.1109/ICARCV.2008.4795593, 2008.

[14] Michael Beetz Radu Bogdan Rusu, Nico Blodow. Fast Point Feature Histograms (FPFH) for 3D Registration. *ICRA)*, pages pp3212 – 3217, DOI: 10.1109/ROBOT.2009.5152473, 2009.

[15] Romain Thibaux John Hsu Willow Garage Radu Bogdan Rusu, Gary Bradski. Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram. *IROS)*, pages pp2155 – 2162, DOI:10.1109/IROS.2010.5651280, 2010.

[16] Vienna Austria ; Vincze M. ; Blodow N. ; Gossow D. Aldoma, A. ; ACIN Tech. Univ. Wien. CAD-model recognition and 6DOF pose estimation using 3D cues . *ICCV Workshops*, pages pp585 – 592, DOI:10.1109/ICCVW.2011.6130296, 2011.

[17] Stefano Luigi Di Tombari Federico. Hough Voting for 3D Object Recognition under Occlusion and Clutter . *IPSJ Transactions on Computer Vision and Applications 4(0), 20-29, 2012*, 2012.

[18] Czech Tech. Univ. Prague Czech Republic ; Matas J. Chum, O. ; Dept. of Cybern. Matching with PROSAC - progressive sample consensus . *CVPR*, pages pp689 – 696, Volume:1, 2005.

[19] Sajid Hussain and Hå kan Grahn. Fast kd- Tree Construction for 3D-Rendering Algorithms Like Ray Tracing. *ISVC*, pages 681–690, Volume 4842, 2007.

[20] Mark Goadrich Jesse Davis. The Relationship Between Precision-Recall and ROC Curves. *the 23rd International Confer ence on Machine Learning*, pages pp 233–240, ISBN:1–59593–383–2, 2006.

[21] Jean-Yves Bougu. Camera calibration toolbox for matlab.

[22] D. McAllester P. Felzenszwalb and D. Ramanan. A discriminatively trained, multiscale, deformable part model. *In Proc. of CVPR*, pages pp 1 – 8, DOI: 10.1109/CVPR.2008.4587597, 2008.

[23] Marcus Rohrbach Wandi Susanto and Bernt Schiele. 3D Object Detection with Multiple Kinects. *ECCV workshop*, pages pp 93–102, Volume 7584, 2012.