TABLE PLANE DETECTION USING GEOMETRICAL CONSTRAINTS ON DEPTH IMAGES

Le Van Hung[1,2], Vu Hai[2], Nguyen Thi Thuy[3], Le Thi Lan[2], Tran Thi Thanh Hai[2]

[1] Faculity of Information and Technology, College of Statistical
[2] International Research Institute MICA, HUST - CNRS/UMI-2954 - GRENOBLE INP
[3] Faculity of Information Technology, Vietnam National University Agriculture
hunglv@gso.gov.vn

ABSTRACT*—Plane detection is an important research in the robotics community. Its results are utilized in many applications such as 3D reconstruction, scene analysis and segmentation, objects localization so on. In context of an assistive system forvisually impaired people, plane detection, or more specilized, table plane detection is a prerequisite step for further procedures such as object localization and recognition. Although many approaches have been proposed for the plane detection such as RANSAC, RANSAC variants, Hough Transform, Least squares. Depending on the context of each application, selecting the apprriciate results always required further implementations. But in fact, each specific application can only focus on one plane (interested plane). In this paper, we propose a new method for detecting an interested plane that is table plane in complex scenes based on depth images captured by a Kinect sensor. Our approach proposed algrothims combines PROSAC algorithm (an algorithm for estimating plane model) and geometrical constraints of environments. We compared some approaches together. The experimental results show that the proposed method outperforms the traditional ones.*

I. INTRODUCTION

Plane detection is a fundamental problem in the field of computer vision and robotics. It is applied broadly for many different researches. The researchers can use it for 3D reconstruction [1] from image sequences or RGB-D image, scene analysis [2], segmentation [3] [4], etc. In context of the vision-based application supporting visually impaired people, the plane detection is an important step. Its results can consit of a table plane in room, where locates many interested object such as botle, cups, boxes, so on; or steps on stairs, ground planes, so on. Such results help to automatic navigating direction and obstacle detection [5].

Currently, there are many approaches for plane detection such as RANSAC [7], Hough Transform [8], Least Square [9]. However, in computer vision, one of the most widely used methodologies for plane detection is RANSAC and its variant [7]. This approach has been proven to successfully detect planes in 2D as well as 3D. They also required a low computational cost. The RANSAC [7] is in each iteration step use a part points of point cloud in the estimation model plane. It may reduce the processing time and plane extraction in point cloud have high noise. Moreover, there has been improvement of RANSAC such as [10] PROSAC (Progressive Sample Consensus), LMEDS (Least Median of Squares), MLESAC(Maximum Likelihood Estimation Sample and Consensus), MSAC (M-estimator SAmple and Consensus), etc. They increase the accuracy of fitting and extraction plane model in the real time.

In a complex scene, plane detection results consit of many planes such as ground wall, table plane, etc. These planesare surfaces of different objects such as door, cabinet, chair, etc. In [6], [11], a table plane is detected in a simple case using RANSAC. The author assumesd that table plan is the largest planes in the scene. Such assumption is not suitable when many planes are exisiting in a complex scene. The fact that each specific application requires theselecting the interested plane). In context of the constructing an assistive system for visually impaired people, to help people handles objects on the table plane the system focuses on detecting table plane. Detection of the interested table plane detection is a prerequise step for finding objects on the table. Consequently, we handles practical conditions, where table plane is localed in the complex scenes such as in a cafeteria, sharing-room. Such scenes are more complex than laboratory-based enviroment.

In this paper, we propose a method for the table plane detection in the complex scene by combining of a conventional plane detection algorithm and geometrical constraints. The table plane is selected from various type of planes such as floor plane, wall plane, table plane, others plane by utilizing geometrical constraints. Such contraints are simple and infer context of real enviroments. The proposed method is simple and real time. The experimental results shows it is fast, accurate and consistent for detecting table planes in realistical enviroments. Fig. 1 illustrates detection results in which the the interested table planes are detected in the complex scenes.

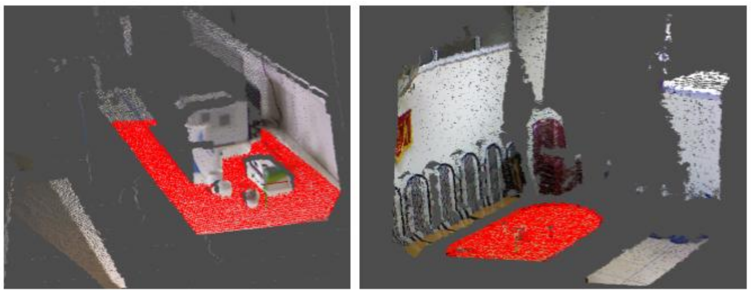


Figure 1: the results of interested plane detection in the complex scene; points have red color belong to table plane.

II. RELATED WORK

The fitting and extracting planes from point cloud have been performed in the many researches [7], [8], [12]. In researches [7, 12], the authors model planes by four pameters (A, B, C, D). To estimate these parameters, they used a threshold method. Points belonging the plane model are inliners, whereas, points not belonging the model are outlines. Schnabel et al. [13] showed that RANSAC algorithm consistents with data which has much outliers or only few inliners. To find the best parameters fitting with the data, it must be run through several iterations. At each iteration, it will be selected randomly a set point in the data for fitting the model. 3D Hough Transform [8], the planes presented by a set point in 3D Hough Space with three parameters (θ, φ, ρ). Each point in Hough Space corresponds with a line ρ (see Eq. 1) in R^3.

$$\rho = p_x \,.\, \cos\theta \sin\varphi + p_y .\, \sin\varphi .\, \sin\theta + p_z \,.\, \cos\varphi \qquad (1)$$

where P (p_x, p_y, p_z) is a point in 3D, θ is the angle between the normal vector at point P on xy-plane, φ is the angle between the line ρ and xy-plane. Least Square [12], it tries to estimate all possible models for a set of data, then choose a best model. Both Hough Transform and Least Squares algorithm used all points in the data for the estimation plane model that have higher computational time. In Hyun et al.[14], the authors compared the processing time of 3D Hough Transform and RANSAC for estimation plane model on the same dataset. The processing time of 3D Hough Transform is 409s, 6.74 s of RANSAC. The deviation between ground truth plane and plane detection of 3D Hough Transform is about 11mm and that approximately is 2mm of RANSAC. Currently, there is some improvement of RANSAC [15]. They are divided into groups differently as follows: accurate group (MSAC, MLESAC), fast group (PROSAC, R-RANSAC), robust group (MAPSAC). Each algorithm has specific advantages and disadvantages. Such MLESAC is more accurate than 5% with RANSAC, the processing time of R-RANSAC is lower than RANSAC. [14], the authors proposed a method which may be plane detected in real time from depth images. This method based on the normal vector of four nearest neighbor points (up, down, left, right). Points belonging the plane model have same the normal vector. This approach is thirteen time as fast as RANSAC on the same dataset.

III. THE PROPOSED METHOD

A. The proposed framework

In this paper, we proposed a framework for detecting an interested plane that is table plane in complex scenes based on depth image captured by a kinect. The input data represented in 3D space by point cloud data. Algorithms using our method combine from PROSAC and geometrical constraints. The processing shows in Fig. 2.

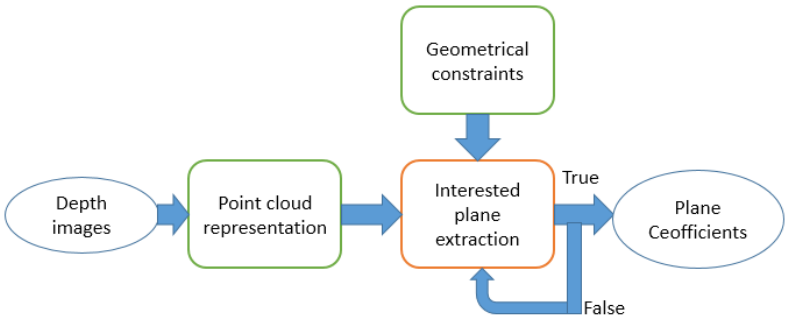


Figure 2: Framework for interested plane detection

In Fig. 2, the step "interested plane extraction" is most important. The output results are the parameter (A, B, C, D) and a set inlier point of table plane.

Interested plane detection

Step 1: Point cloud representation

In this step, the depth data is converted to 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 of depth. To represent the data, we use a function of PCL library (Point Cloud Library). This function combines three values (x, y, z) and parameters of kinect from calibration data. Each point of point cloud data has 3D coordiantes (x, y, z). It generates from using the depth camera intrinsic [11] and projected the each pixel (xd, yd) to metric 3D space as follows the equation (2).

$$X = \frac{Z\cdot(x_d - c_x)}{f_x} \quad Y = \frac{Z\cdot(y_d - c_y)}{f_y} \quad Z = depth(x_d, y_d) \qquad (2)$$

with f_x, f_y, c_x and c_y are focal length and principal point. Then in order reduce the size of the data, we present point cloud data by means of a voxel grid.

Step 2: Table plane extraction

Geometrical constraints:

In [11], [16], the table plane is the largest in the scene that is only an estimation of table plane detection. Our approach detects a plane from ground, wall, a table and other planes. The size of planes is different. We proposed some geometrical constraints for table plane detected as follows:

Scenario is the visually impaired people mounting a Kinect and moving on the cafeteria, sharing-room. This person wants to find some objects on the table. Kinect orientation focuses into table center. The height between Kinect and table plane is h. The angle between person height (straight line) and Kinect orientation is α degree. The origin (O x_k y_k z_k) is Kinect center. The angle between Kinect orientation and normal vector of table plane is 90 + (90 - α) (see Fig. 3.a).
The real world is the table plane perpendicular to the wall plane and the normal vector of the table plane (N_1) parallel with the normal vector of the ground plane (N_2) that consistent the Eq. 3. The angle between Kinect orientation and normal vector of wall plane is β that is about $90 + (90 - \alpha) <= \beta < 180$ degree (see Fig. 3.b) and about $180 - (90 - \alpha) <= \beta<180$ degree (see Fig. 3.c) .β is smallest in the case Kinect orientation perpendicular with normal vector of the wall plane (see Fig. 3.b). Fig. 3, we show some angle constraints with $\alpha=60^0$.

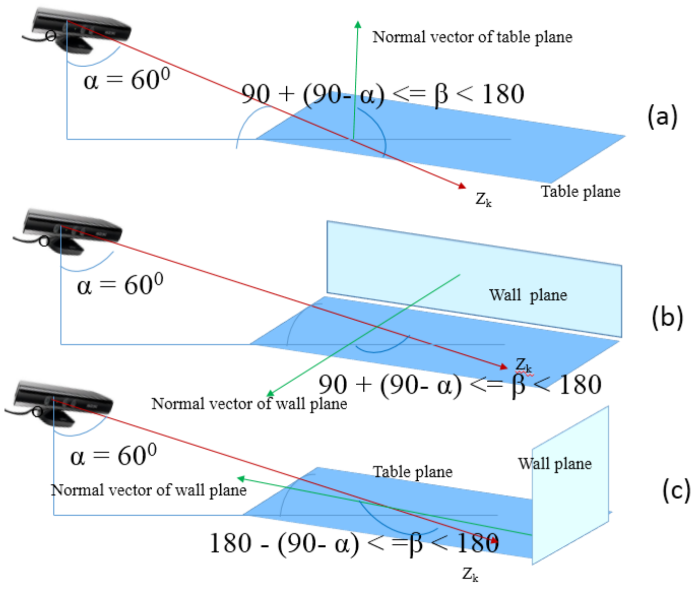


Figure 3: the angle constraints in for table plane detection; (a), (b), (c) are that case $\alpha=60^0$

Based on Kinect location, the direction of the z-axis is in table center. Direction of the x-axis is horizontal. Direction of the y-axis is to down. The y value of the points belonging table plane (V_t) is smaller than points belonging ground plane (V_f). Two types plane are the same normal vector, but they are the different D coefficient.

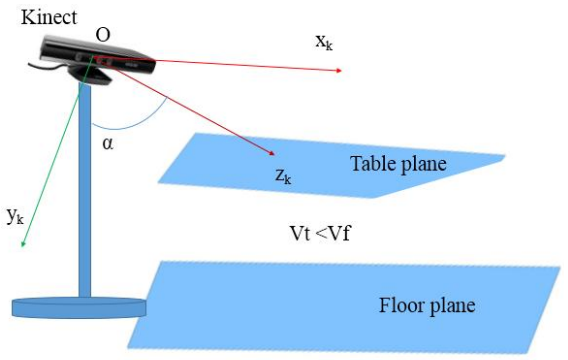


Figure 4: Constraint of location plane; it uses for classification between table plane and ground

However, the classification between table plane and wall plane is unclear in Fig. 3a, Fig. 3b. Because, the angle between the normal vector of wall plane and z-axis is similar to the angle between the normal vector of table plane and z-axis. Assuming, *maxy* is the maximum value following y-axis of point cloud data; *miny* is the minimum value following y-axis of point cloud data. In complex scenes have the table plane, wall plane, ground plane. *maxy* belongs to the wall plane. *miny* belongs to the ground plane. Should we use constraint (Eq. 3) for distinguishing between the table plane and the wall plane.

$$Miny \in U_w < y_i \in Ui < maxy \in U_g \qquad (3)$$

Before performing estimates the planes in complex scenes. We used the Kd-tree [20] structure and K-nearest neighbor algorithm for clustering the point cloud data. It is a structure for clustering unsupervised that base the Euclidean distance between points in cloud point. They applied for clustering the point cloud data that presented the brief [21] in Algo.1.

Algorithm1: Kd-Tree structure in K-neast neighbor for clustering point cloud

```
    Input: Points of point cloud (P)
    Output: Regions
 1: Create a Kd-tree representation for the input point cloud dataset P;
 2: Set up an empty list of clusters C, and a queue of the points that need to be checked Q;
 3: For every point $p_i \epsilon P$, perform the following steps:
 4:   {
 5:     Add $p_i \epsilon P$ to queue Q
 6:     For every point $p_i \epsilon Q$ do
 7:       {
 8:         Search for the set $P^i_k$ of point neighbors of $P_i$ in a sphere with radius $r < threshD$
 9:         For every neighbor $p^i_k \epsilon P^i_k$, check if the point has already been processed, and if not add it to Q;
10:       }
11:     When the list of all points in Q has been processed, add Q to the list of clusters C, and reset Q to
          an empty list;
12:     The algorithm terminates when all points $p_i \epsilon P$ have been processed and are now part of the list
          of point clusters C;
13:   Regions (k) = C;
14:   }
```

where *threshD* is the threshold radius for searching neighbor points.

Extraction plane:

Interested plane extraction is the most important step in our approach. It is combining of PROSAC and geometrical constraints. These geometrical constraints presented in the above section. In this step, we represented to combining detail of PROSAC and geometrical constraints for table plane detection. PROSAC is a RANSAC [17] variant that presented the brief in Algo. 2.

Algorithm 2: RANSAC for plane detection

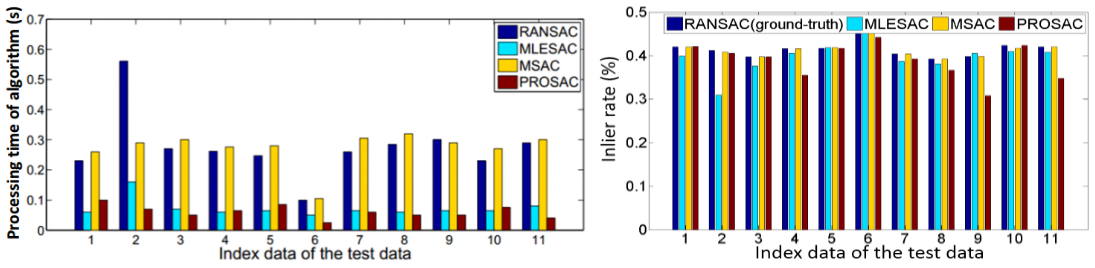
```
    Input: The point cloud
    Output: bestPlaneCeo(1,3)
 1: i=0; bestStd = INT_MAX; bestPlaneCeo(1,3)=(0,0,0);
 2: w = 1 - rateOutlier /length(point-list);
 3: K= round(log(1-P)/(log(1-$w^3$)));
 4: while i<K do
 5:     k pick 3 points randomly among (point-list);
 6:     model= pts2plane(k);
 7:     dis(point-list) = dist2plane(pl, point-list );
 8:     s= find(abs(dis(point-list)) < thre)
 9:     st = Standard-deviation(s)
10:     if (length(s)> bestNumPoint) or  (length(s)= bestNumPoint and st < bestStd) then
11:         Inlier= length(s);
12:         bestPlaneCeo= model; bestStd = st;
13:     end if
14:     i=i+1;
15: end while
```

In this Algo. 2: "thre" - is the distance threshold of a point belong to plane model
"bestNumPoint" is the minimum number point of plane
"w" - is the probability of choosing an inlier. It can be estimated before.
"P" – is the probability for successful algorithm that is between 0.9 - 0.99.

PROSAC algorithm is improved of RANSAC algorithm by processing time. It is faster RANSAC to hundred times [18]. Unlike with RANSAC, its idea is as follows [18], [19]:

- Generate all T_N samples similar RANSAC.
- Choosing U_n good samples in order of decreasing quality of samples T_N .
- In each iteration, choosing sample from U_n of the previous iteration.

To testing the processing time of PROSAC, RANSAC, RANSAC variants, we performed the small testing. We tested to fitting plane on eleven sets of point in MICA dataset with parameters (*P=0.99, thresh* is the threshold for determining inlier, K=100). PROSAC is faster than RANSAC, RANSAC variants (see Fig. 5a). The total of processing time of it is faster RANSAC for 2.364s. The inlier rate is not much lower than RANSAC (ground-truth) (see Fig. 5b). The normal vector of planes is similar.



(a) (b)

Figure 5: The tested result of the estimation plane by RANSAC, RANSAC variants; (a) processing time, (b) inlier rate of model estimation.

Complex scene has many planes (*ThreNumPlane* planes) such as table, wall, ground and other planes. Each estimation they had a plane. To find the interested plane, we checked the geometrical constraints in each estimation of PROSAC. If a plane consistent with constraints, then stop whereas next iterations. The detail presented in Algo. 2. The real, number point of complex scene is about 100000 to 400000 points. PROSAC algorithm need to determining minimum of plane size (*minpoint*); number of iterations N; probability for running successful algorithm P (success). In this paper, we set of them as follows:

minpoint= 10% number point of the scene.
K = 70, because we estimate 40% inlier.
P (success) = 0.99

In this paper, we see two cases appear as follows:

The first, no clustering between table plane and wall plane: the region point cloud of table plane and wall plane is the largest data (number point). We find to it in the regions data. Supposed, it is greater than about "*pc*"=50% number point of the scene. After that, we repeat combination fitting plane (PROSAC) and geometrical constraints for table plane detected satisfying the constraints.

The second, clustering between table plane and wall plane: we repeat combination fitting plane (PROSAC) and geometrical constraints for table plane detected satisfying the constraints. The processing of our approach is presented in algorithm 2:

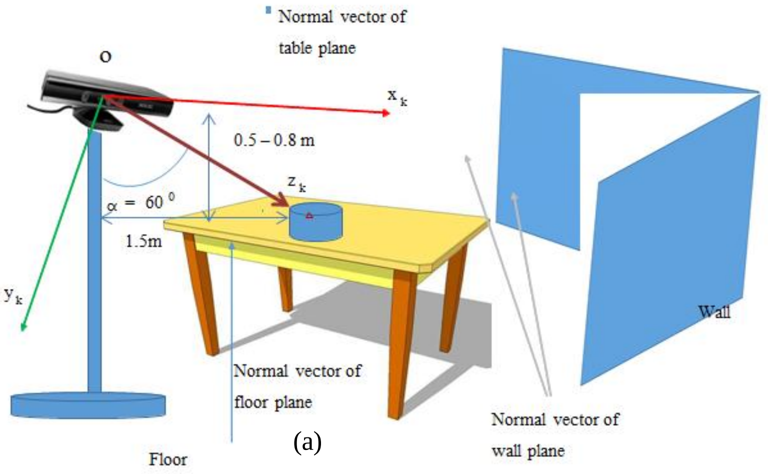
Algorithm 3: Combining PROSAC and geometrical constraints for interested plane detection

```
Input: The point cloud
Output: tablePlaneCeo(3,1)
1: i=0; bestStd = INT_MAX; tablePlaneCeo(1,3)=(0,0,0);
2: p = clusterPointcloud(scene); pc=0.5*numberpoint(scene);
3: N= length(p); k=0.1*numberpoint(scene);
4: while i<N do
5:         pi=Regions(i);
6:         if (numpoint(pi)>=pc) then
7:                 kt=false;
8:                 while (kt==flase) do
9:                         pf= PROSACfitPlane(pi);
10:                        if (numberpoint(pf)>=k) and (geometricalConstraints(pf)==true) then
11:                                kt=true; tablePlaneCeo(3,1)= pf;
12:                        end if
13:                end while
14:        else
15:                pf= PROSACfitPlane(pi);
16:                if (numberpoint(pf)>=k) and (geometricalConstraints(pf)==true) then
17:                        tablePlaneCeo(3,1)= pf;
18:                end if
19:        end if
20:        i=i+1;
21: end while
22: if (tablePlaneCeo(1,3)==(0,0,0)) then
23:        return false;
24: end if
25: Return tablePlaneCeo;
```

IV. EXPRIMENTAL RESULTS AND DISCUSSION

A. Experimental

Fig. 6a illustrates the experimental setup we used to evaluate the proposed approach. We were performed in the kitchen near the lab of MICA. We refer to this as the "MICA" dataset. We set up experiments to collect data as follows: Kinect is mounted on a shelf; the angle between Kinect orientation and the vertical line of shelf is 60 degree; the table put is in the center of the room and in other cases near the wall; in some sequences we placed chairs next to the table it; may be the ttable work, table eat; the height of table and floor is about 0.8m; the high between Kinect and the table plane was about 0.5 - 0.8 m; the distance between Shelf of mounted Kinect and about table center is about 1.5m. Some the location of kinect is illustrated in Fig. 6b. The environment is indoor (kitchen, our work). The Kinect captured to 5 frames / second. Because Kinect usually collect data with speed 15 Hz - 20 frames / second, but in MICA platform that it takes time for the calibration. The each height location of Kinect (0.5 – 0.8m) and the each scene, we collected the images in 5s. The images had been collected a resolution 640x480 pixels. Each scene is always only one table. Both the depth and RGB sensors have been calibrated by MICA platform.


Normal vector of table plane
O
x k
0.5 – 0.8 m
z k
α = 60 0
1.5m
y k
Wall
Normal vector of floor plane
Normal vector of wall plane
Floor
(a)

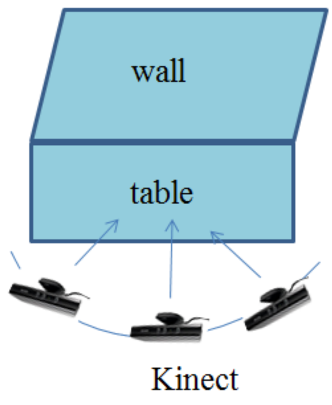


Figure 6: (a) setup Kinect to collect data (RGB-D images); (b) some location Kinect puts in the experimental for collecting data.

In this experiment, the number of image is about 800 images. The Fig. 7 is shown eight the scenes of experimental.

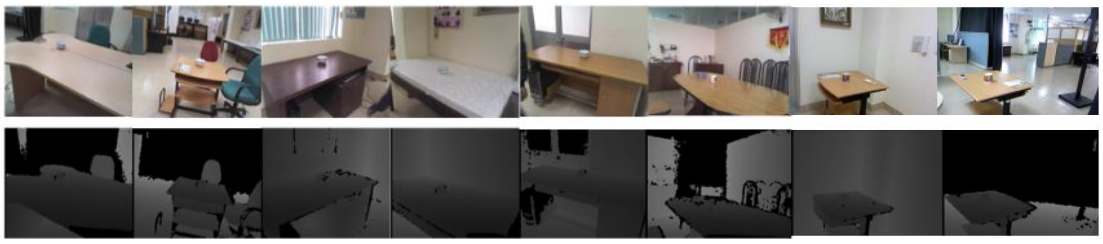


Figure 7: Some scenes of experiment

B. Results

In order to evaluate the table plane detection, we prepared the ground-truth table plane for evaluating of table plane detection as follows: From depth images to crop table image in the block, then point cloud representation and get a normal vector of the plane (using RANSAC algorithm), 3D center point of the plane (see Fig. 8).

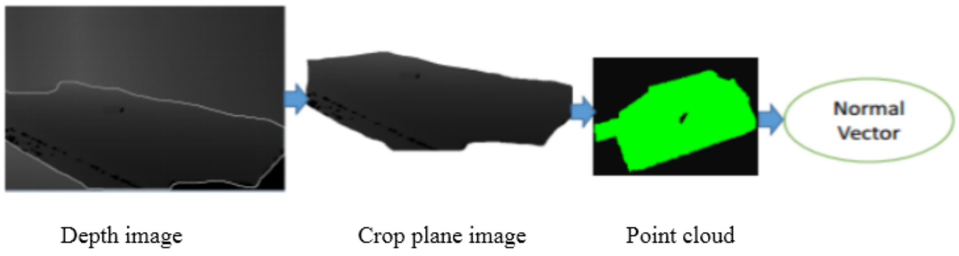


Figure 8: Preparing ground-truth table plane for evaluating table plane detection

we illustrated to get the normal vector of the plane in preparing plane ground truth in Fig. 9.

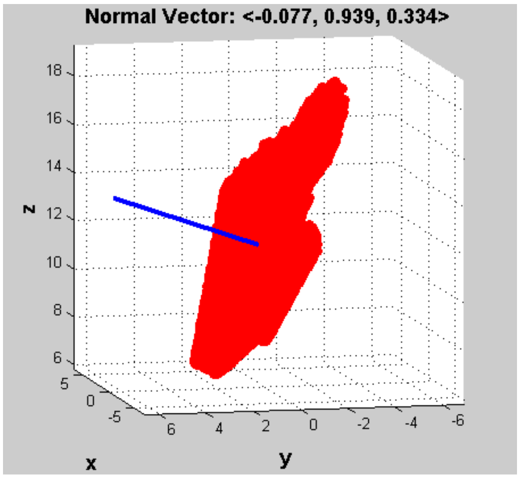


Figure 9: The step get normalvector for preparing of the plane ground truth

We defined a true table plane detection as follows: the angle between the normal vector of the plane detection and the normal vector of the ground-truth table plane less than 5 degrees and the 3D center point of ground truth belong to the plane detection. If the angle is more than 5 degrees or the 3D center point of ground-truth table plane do not belong to the plane detection, it was the false detection. Results based on two directions: the first is the rate of true detection; the second is the computation time. We are evaluated on three approaches:

- The baseline approach is combined from clustering point cloud and choosing table plan is the largest plane in the scene and using PROSAC for fitting plane.
- Others approaches are combining from clustering point cloud and using Least Squares, 3D Hough Transform, RANSAC, MLESAC, MSAC for fitting plane and geometrical constraints of the environment.
- Our approach, we combined from clustering point cloud and using PROSAC for fitting plane and geometrical constraints of the environment.

We evaluated on 800 images of the experiment. Our system is written in the C++ language and development using the combination of PCL 1.7 and OpenCV 2.4.8. Out method has been tested on PC with Core i5 processor – RAM 4G. The results of table plane detection (precision) in Tab. 1.

Approach	Algorithm for fitting plane	True table plane detector (%)	Processing time(s)
Baseline	**PROSAC - Code C++**	26.2%	248 s
Others approaches	**Least Square[12] -Code C++**	89.3%	633s
	3D Hough Transform[8] - Code MatLab	88.4%	1021 s
	RANSAC[7] -Code C++	90.7%	376 s
	MLESAC[10] -Code C++	92.9%	308 s
	MSAC[10] -Code C++	93.1%	365 s
Our approach	**PROSAC[10] -Code C++**	**92.6%**	**275 s**

Table 1: The results of table plane detection (precision) on "MICA" dataset

We are evaluated the cost error function between plane model ground truth and plane model detected. The equation (Eq. 4) of error function presented as follows [22] (4):

$$Error_f = sqrt((a' - a)^2 + (b'-b) + (c'-c)) \qquad (4)$$

where (a,b,c) is the parameters of plane ground truth; (a',b',c') is the parameters of plane detected. Cost error function of scenes in MICA dataset are shown on Fig. 10.

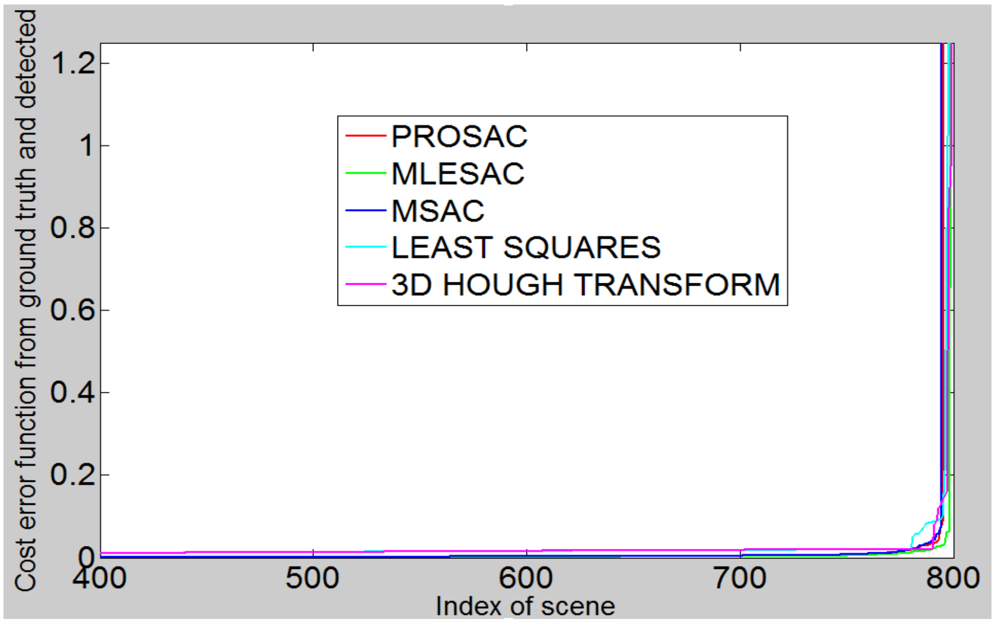
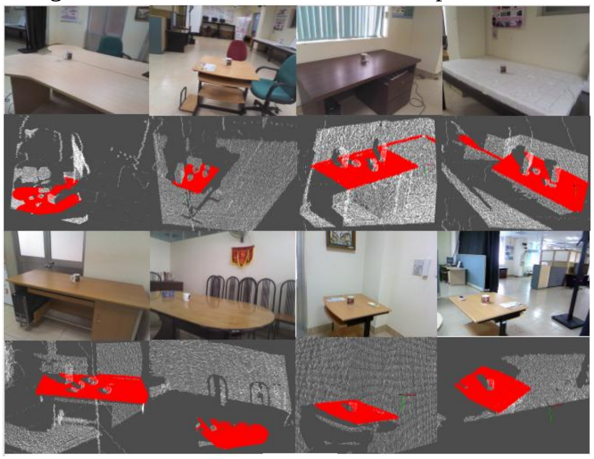
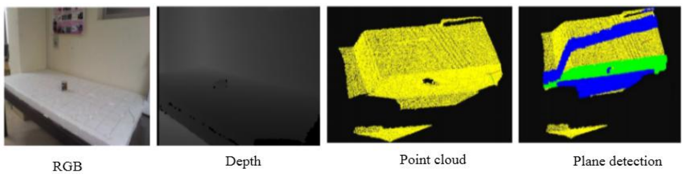


Figure 10: Cost error function of scenes in MICA dataset presented with some algorithms. They compared with the ground truth parameters (RANSAC).

In Fig. 11a illustrates the results of table plane detection in some scene.



(a)



(b)

Figure 11: (a) some results of table plane detection on point cloud; the points have red color on plane is belong to table plane. (b) the wrong table plane detection; Points have blue color belong to wrong table plane detection

In Tab. 1, the results of our approach are faster and more accurate than Least Squares, 3D Hough Transform, RANAC. The rate of true table plane detection is 92.6% and the processing time is 275 s equal 3 scenes / second. It is lower than the MLESAC, MSAC, but it is faster than them. In baseline approach, the result is 26.2% that it proves the number of scenes have table plane is the largest plane in the scene.

The wrong table plane detection focus into the cases not clustering table plane and wall plane. In this case, it fit a plane have points belong to both plane: wall plane, table plane (see Fig. 11b).

IV. CONCLUSION

In this paper, we have shown a new approach for interested plane detection from point cloud of depth images of Kinect sensor. We have successfully built a system to table plane detection by combining PROSAC algorithm and geometrical constraints of the environment. Our approach is simple and processing in real time. The results of our approach are more accurate than other approaches for table plane detection in complex scene and it also confirmed PROSAC is a very fast algorithm for estimation and fitting model. Results may be applied to localizing objects in robotic or construction visually impaired supporting system. The results of the experiment have shown us.

V. REFERENCES

1. Kaucic, R., R. Hartley, and N. Dano, Plane-based projective reconstruction. Computer Vision, IEEE International Conference on 1, 2001(420).
2. Kahler, O.a.J.D., Detection of planar patches in handheld image sequences. In PCV06, 2006.
3. Biosca, J.a.J.L., Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods. ISPRS Journal of Photogrammetry and Remote Sensing 63 (1), 2008: p. 84-98.
4. Bastian Oehler, J.S., Jochen Welle, Dirk Schulz, and Sven Behnke, Efficient Multi-Resolution Plane Segmentation of 3D Point Clouds. ICIRA, 2011.
5. Zhe Wang, H.L., Yueliang Qian, Tao Xu Real-Time Plane Segmentation and Obstacle Detection of 3D Point Clouds for Indoor Scenes. ECCV 2012. , 2012: p. pp 22-31.
6. Radu Bogdan Rusu, G.B., Romain Thibaux, John Hsu Willow Garage, Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram. IROS, 2010.
7. Bolles, M.A.F.R.C., Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysisand Automated Cartography. Communications of the ACM, 1981. 24(6): p. 381--395.
8. Dorit Borrmann , J.E., Kai Lingemann , Andreas Nüchter The 3D Hough Transform for Plane Detection in Point Clouds : A Review and a new Accumulator Design. 3D Display Research Center,© 3D Research Center and Springer 2011 2011. 02003.
9. Badino, H., Least Squares Estimation of a Plane Surface in Disparity Image Space. Carnegie Mellon University Pittsburgh, PA 15217, USA, 2011.
10. Choi, S., T. Kim, and W. Yu, Performance Evaluation of RANSAC Family. Procedings of the British Machine Vision Conference 2009, 2009.
11. Kramer, J.B., Nicolas Echtler, Florian Daniel, Herrera C. and M. Parker, Hacking the Kinect. Apress, 2012.
12. Eberly, D., Least Squares Fitting of Data. 2015: p. 1--9.
13. Schnabel, R.W., R Klein, R, Efficient RANSAC for Point-Cloud Shape Detection. Computer Graphics Forum, 2007. 26(2): p. 214--226.
14. Yoo, H.W.K., Woo Hyun Park, Jeong Woo Lee, Won Hyong Chung, Myung Jin, Real-time plane detection based on depth map from Kinect. Ieee Isr 2013, 2013. 2: p. 1--4.
15. Choi, S.K., Taemin Yu, Wonpil, Performance Evaluation of RANSAC Family. Procedings of the British Machine Vision Conference 2009, 2009: p. 81.1--81.12.
16. Aitor Aldoma, Z.-C.M., Federico Tombari, Walter Wohlkinger, Christian Potthast and R.B.R. Bernhard Zeisl, Suat Gedikli, and Markus Vincze, Three-Dimensional Object Recognition and 6 DoF Pose Estimation. Robotics Automation Magazine, 2012(September).
17. Michael Ying Yang, W.F., Plane Detection in Point Cloud Data. Department of Photogrammetry Institute of Geodesy and Geoinformation University of Bonn, 2010.
18. Chum, O.D.o.C., Czech Tech. Univ., Prague, Czech Republic ; Matas, J., Matching with PROSAC - progressive sample consensus Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on 2005. 1: p. 220 - 226
19. Matas, J., RANSAC in 2011 (30 years after). CVPR20111, 2011.
20. Sajid Hussain and Hå kan Grahn. Fast kd- Tree Construction for 3D-Rendering Algorithms Like Ray
Tracing. ISVC, pages 681–690, 2007.
21. http://www.pointclouds.org/documentation/tutorials/cluster_extraction.php.
22. Ngo Thanh Trung, Hajime Nagahara, Ryusuke Sagawa,Yasuhiro Mukaigawa, Masahiko Yachida,Yasushi Yagi, Highly Robust estimator Using a Case- Dependent Residual Distribution Model, IPSJ Transaction on Computer Vision and Applications, 2009

PHÁT HIỆN MẶT BÀN SỬ DỤNG CÁC RÀNG BUỘC HÌNH HỌC TRÊN ẢNH ĐỘ SÂU

Lê Văn Hùng[1,2], Vũ Hải[2], Nguyễn Thị Thủy[3], Lê Thị Lan[2], Trần Thị Thanh Hải[2]

[1] Khoa công nghệ thông tin, Trường Cao đẳng Thống kê
[2] Viện quốc tế MICA,Trường Đại học Bách khoa Hà Nội
[3] Khoa công nghệ thông tin, Học viện Nông nghiệp Việt Nam
hunglv@gso.gov.vn

TÓM TẮT*—Phát hiện mặt phẳng là một nghiên cứu quan trọng trong cộng đồng nghiên cứu về robot, đặc biệt là trong các nghiên cứu xây dựng các hệ thống hỗ trợ cho người khiếm thị. Nó đã được thực hiện trong nhiều hướng tiếp cận như RANSAC, các biến thể của RANSAC, 3D Hough Transform, Least Squares. Nó cũng được áp dụng trong nhiều ứng dụng như: xây dựng lại không gian 3D, phân tích cảnh, phân đoạn các vùng dữ liệu, xác định vị trí của các đối tượng,...vv. Kết quả của các hướng tiếp cận trước là một hoặc nhiều mặt phẳng trong cảnh. Nhưng trong thực tế, đối với mỗi ứng dụng riêng có thể chỉ tập trung vào một mặt phẳng (mặt phẳng quan tâm). Trong bài báo này chúng tôi đề xuất một hướng tiếp cận mới cho phát hiện một mặt phẳng quan tâm trong cảnh phức tạp và xử lý thời gian thực từ đám mây điểm(point cloud) của ảnh độ sâu(depth) của Kinect. Nó dựa trên sự kết hợp của PROSAC(một giải thuật cho ước lượng mô hình mặt phẳng) và các ràng buộc hình học của môi trường. Chúng tôi có so sánh hướng tiếp cận mà chúng tôi đề xuất với một số hướng tiếp cận khác cho bài toán này. Các kết quả thí nghiệm cho thấy hướng tiếp cận của chúng tôi cao hơn so với các hướng tiếp cận truyền thống.*