Journal of Computer Science and Cybernetics, V.32, N.3 (2016), 243–258
DOI 10.15625/1813-9663/32/3/7689

REAL-TIME TABLE PLANE DETECTION USING ACCELEROMETER INFORMATION AND ORGANIZED POINT CLOUD DATA FROM KINECT SENSOR

VAN-HUNG LE[1,2], MICHIEL VLAMINCK[4], HAI VU[1], THI-THUY NGUYEN[3], THI-LAN LE[1], THANH-HAI TRAN[1], QUANG-HIEP LUONG[4], PETER VEELAERT[4], WILFRIED PHILIPS[4]

[1]*International Research Institute MICA, HUST - CNRS/UMI-2954 - GRENOBLE INP, Vietnam*
[2]*Tan Trao University, Vietnam; lehung231187@gmail.com*
[3]*Faculty of Information Technology, VietNam National University Agriculture, Vietnam*
[4]*Ghent University/iMinds - Image Processing and Interpretation, Belgium*

Abstract. Table plane detection in the scene is a prerequisite step in developing object-finding-aided systems for visually impaired people. In order to determine the table plane in the scene, we have to detect planes in the scene first and then define the table from these detected planes based on the specific characteristics. Although a number of approaches have been proposed for plane segmentation, it still lacks proper table plane detection. In this paper, the authors propose a table plane detection method using information coming from a Microsoft Kinect sensor. The contribution of the paper is three-fold. First, for plane detection step, the dedicated down-sampling algorithms to original point cloud thereby representing it as the organized point cloud structure in are applied to get real-time computation. Second, the acceleration information provided by the Kinect sensor is employed to detect the table plane among all detected planes. Finally, three different measures for the evaluation of the table plane detector are defined. The proposed method has been evaluated using a dataset of 10 scenes and published RGB-D dataset which are common contexts in daily activities of visually impaired people. The proposed method outperforms the state-of-the-art method based on PROSAC and obtains a comparable result as a method based on organized point cloud where the frame rate is six times higher.

Keywords. Table plane detection, acceleration vector, organized point cloud, plane segmentation.

1. INTRODUCTION

Plane detection in 3-D point clouds is a critical task for many robotics and computer vision applications. In order to help visually impaired/blind people find and grasp interesting objects (e.g., coffee cup, bottle, bowl) on the table, one has to find the table planes in the captured scenes. From the extracted table plane, then the relevant features can be calculated such as its normal vector, center point of the table in the current scene. These features will help to determine the object positions in the current scene. As a prerequisite step, the table plane extraction should be a robust algorithm and furthermore high accuracy and low computational cost. However, 3-D point clouds obtained by low-cost sensors (e.g. from the Microsoft Kinect sensor [13], other depth cameras) are generally noisy and redundant with a huge number of points. Therefore, in common approaches, the plane extraction either produces false positives results or requires huge computational costs. By

exploiting associated data provided by the sensors, the table plane features can be adapted in the 3-D point cloud in a robust way. This paper is motivated by such adaptation in which accelerator data provided by the Kinect sensor to prune the extraction results. The proposed algorithms achieve both real-time performances as well as high detection rate of the table planes. In the experimental evaluations, the proposed method is examined in different contexts to confirm the robustness of the proposed algorithm.

The paper is organized as follows. Section 2 presents related works on plane segmentation. Section 3 describes the proposed method with two main topics: plane segmentation and table plane extraction. Section 4 shows experimental results. Section 5 concludes the paper and gives some ideas.

2. RELATED WORK

The plane extractions/segmentations in a complex scene usually are solved by two main approaches. The first approach uses robust estimation algorithms such as RANSAC [8], and its variants [3], Least Squares [1], Hough Transform [2] for estimating the planes in the scene. For example, Yang et al. [17] proposed to combine the RANSAC algorithm and *'minimum description length'* for plane estimation from point cloud data having complex structures. The point cloud is divided into the blocks, the RANSAC algorithm is performed on each block. Each block is limited from zero to three planes. This combination generates an algorithm that avoids detecting wrong planes due to the complex geometry of the 3-D data. Usually, the point cloud of the scene has millions of points. Therefore, when using the estimation algorithm on this data, it requires a high computational time. On top of that, the RANSAC-based approaches strongly depend on an heuristically chosen threshold to eliminate outliers.

The second approach is based on the local surface normal in the scene [5, 6, 7]. Deschaud et al. [5] proposed an approach for plane detection on unorganized point cloud data. In that paper, the authors implemented the following three steps. The first step estimates the normal vector at each point. The second step computes the score of local planarity in each point, after that the best seed point that represents a good seed plane is selected. The third step is growing this seed plane by adding all points close to the plane. [6, 7] uses the normal vector of points in organized point cloud data. Organized point cloud data is structured so that points are arranged in a grid (similar to image pixels in a matrix structure). Holz et al. [6] proposed an approach for organized point cloud segmentation. This approach performs segmentation in two steps. First, the segmentation step is performed on the normal of each point. After that, points having similar local surface normal orientation are examined and segmented in distance space generate region. This approach is able to process video with frame rates of up to 30Hz. Chen et al. [7] proposed an approach using hierarchical clustering on cloud data based on the normal vectors. The clustering performed similar to the approach in [6], but the authors have represented the data in a graph whose node and edge represent a group of points and their neighborhood respectively. The clustering is performed on the graph. This approach can process video with a frame rate of more than 35Hz for 640×480. Although, all these approaches implemented plane segmentation in the scene, they do not address table plane detection in particular. We have proposed an approach for the table plane detection [10] in complex scenes. We used PROSAC [4] algorithm for plane segmentation and some geometrical constraints for table plane extraction in the complex scene. However, this approach requires prior knowledge about the scenes, such as constrains between the wall and the table, the wall and the floor, size of the table.

Figure 1. Object finding aid for the visually impaired people

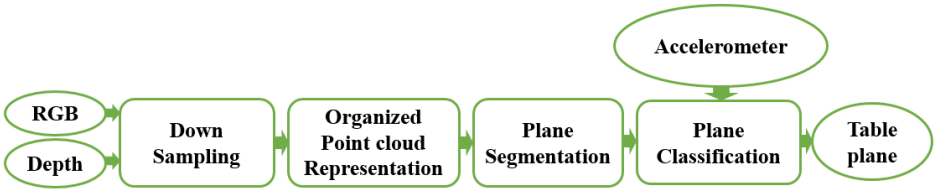

Figure 2. The proposed frame-work for table plane detection

3. TABLE PLANE DETECTION

3.1. Overview

Our research context aims to develop object finding and grasping-aided services for visually impaired people (see Fig. 1). To this end, it is needed to locate the queried objects in the table. In order to develop such a service, this paper first deals with detecting the planes in the current scene and then determining table plane from detected planes. Therefore, we approach this objective as a problem of table plane detection and extraction from a real scene.

In our work, Microsoft Kinect, a low-cost depth and RGB sensor, is utilized. This sensor becomes more and more popular in computer vision applications. An object captured by the Kinect sensor is represented in 3-D space denoted by the coordinates (x, y, z) [13]. This data is generated from both color and depth images in order to form the point cloud data. The proposed frame-work, as shown in Fig.2, consists of four steps: down-sampling, organized point cloud representation, plane segmentation and table plane classification. To achieve low computational costs, the data in the first step is reduced, targeting a lower sampling rate. However, the sampling rate can not be arbitrarily low because it can significantly affect the subsequent steps and lower the overall detection quality.

For plane segmentation, RANSAC or one of its variants can be used as referred to in the related work section. However, this step requires highly accurate and real-time plane extraction. Therefore, we proposed to represent the information captured from the Microsoft Kinect as an organized point cloud and perform plane segmentation using normal vector information. To select the table planes within the extracted planes, the acceleration data from the Kinect sensor is used in the third step. The main constraint is that a table should stand on the floor. Therefore the table plane should be parallel with the floor plane. The accelerometer from the Kinect sensor provides us the normal vector of the ground (floor) plane and other planes, that are parallel with the table plane. The planes which

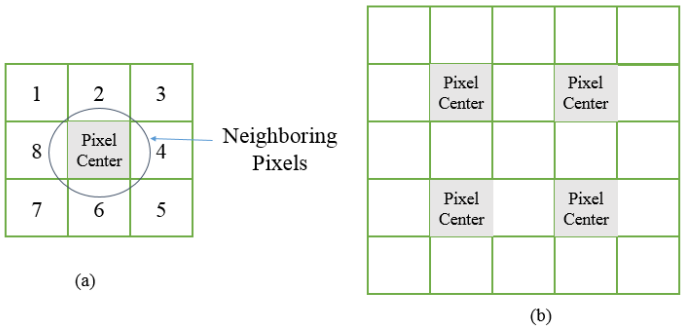

Figure 3. (a) Computing the depth value of the center pixel based on its neighborhoods (within a 3×3 pixels window); (b) down sampling of the depth image

do not meet this criterion are eliminated. A table plane is identified among the remaining planes if it is high enough. The proposed method has to be done in the real-time and archives state-of-the-art detection rate results. In the following sections, four steps of the method are described in detail.

3.2. Point cloud representation

In the scenario of developing an object-finding system that aids the visually impaired people collecting an object based on their query, separating the table plane from a current observation is a pre-processing step so that the interested objects lying on the table are more easily detected and localized. This pre-processing procedure therefore should be high detection rate and low computational costs. However, the collected point cloud data in a certain scene consists of many 3-D data points (each point has the coordinates (x, y, z)). This type of data always requires high computational time but includes many noises. To deal with these issues, we adopt down-sampling and smoothing techniques. Actually, many down-sampling techniques such as [16, 12, 15, 9] could be applied. Because of our work utilizing only depth feature, a simple and effective method for down-sampling and smoothing the depth data is described as below.

Given a sliding window (of size $n \times n$ pixels), the depth value of a center pixel $D(x_c, y_c)$ is computed from the Eq. 1:

$$D(x_c, y_c) = \frac{\sum_{i=1}^{N} D(x_i, y_i)}{N}, \tag{1}$$

where $D(x_i, y_i)$ is depth value of i^{th} neighboring pixel of the center pixel (x_c, y_c); N is the number of pixels in the neighborhood n x n (N=(n x n) -1). An illustration of the down-sampling procedure is given in Fig. 3a. As shown, if the input depth image has the size of 640×480 pixels and the sliding window has the size of 3×3 pixels, then the output depth image is reduced to 320×240 pixels. If the size of the sliding window is increased, then the size of the output depth image is reduced.

It is noticed that this technique does not change the coordinates of pixels and the coordinates of objects in 3-D space, it only reduces the number of points representing an object. By averaging depth values in each window, we only retain the center pixels (x_c, y_c) of all the sliding window, and other pixels (as shown in Fig. 3b) are removed. Therefore, this step also smoothes the collected data.

After down-sampling, the image data is converted into organized point cloud data. Each data point has a 3-D coordinate (x, y, z) and color values (r, g, b). Using the RGB camera intrinsic

parameters [13], each pixel (x_p, y_p) in the RGB image has a color value $C(r_p, g_p, b_p)$ and a depth value $D(x_p, y_p)$ in the corresponding depth image, which is projected into the metric 3-D space using the following Eq. 2:

$$x = \frac{z(x_p - c_x)}{f_x}; \quad y = \frac{z(y_p - c_y)}{f_y}; \quad z = D(x_p, y_p); \quad (r, g, b) = C(r_p, g_p, b_p) \tag{2}$$

with (f_x, f_y) and (c_x, c_y) being the focal length and principal point, respectively.

The organized point cloud data follows the structure of a matrix as in the image. Each point has a 2-D index (i, j), in which (i, j) are the indices of the row and column of the matrix respectively. They are limited by the size of the collected image. For example, an image obtained from Microsoft Kinect sensor has 640×480 pixels, then $i = 1, ..., row$; $j = 1, ..., col$ with $[row, col] = [480, 640]$. Matrix P presents the organized point cloud data of a scene as below:

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & \cdots & p_{1,col} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} & \cdots & p_{2,col} \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} & \cdots & p_{3,col} \\ p_{4,1} & p_{4,2} & p_{4,3} & p_{4,4} & \cdots & p_{4,col} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{row,1} & p_{row,2} & p_{row,3} & p_{row,4} & \cdots & p_{row,col} \end{bmatrix}; p_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j}) \tag{3}$$

where $(x_{i,j}, y_{i,j}, z_{i,j})$ are values of 3-D coordinate as defined in Eq. 2.

It is noticed that the number of points in the cloud is reduced data after applying the down-sampling technique. The indices of the down-sampling data via center pixels of the sliding windows are preserved. By this way, it is easier to find the corresponding pixel in the original color image.

3.3. Plane segmentation

The third step in the proposed framework (as shown in Fig. 2) is plane estimation that is based on normal vectors extracted from organized point clouds. The planes extraction procedure consists of the following steps: estimating surface normals, segmentation and merging co-planar points to generate the planes. The detailed process of the plane segmentation is given in Algorithm 1. First, for estimating the surface's normal vector, the approach of Holz et al. [6] was used. Illustrations of this technique are shown in Fig. 4. A plane at point p_i is estimated based on itself p_i and its two neighbors, as shown in Fig.4(b). To estimate a normal vector at a point p_i, k-nearest neighbours of p_i are determined within a radius r. The curvature value σ (Eq. 5) is estimated by analyzing the eigenvectors of the covariance matrix C as given in Eq. 4. The similar value σ and normal vectors of the points within an organized point cloud P are grouped and clustered to select co-planar points. These co-planar points are starting points of the region growing algorithm for merging and generating planes.

$$C = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{p}_i - \mathbf{p}_{av})(\mathbf{p}_i - \mathbf{p}_{av})^T, \quad C\mathbf{v}_j = \lambda_j \mathbf{v}_j, \quad j \in \{0, 1, 2\}; \tag{4}$$

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}. \tag{5}$$

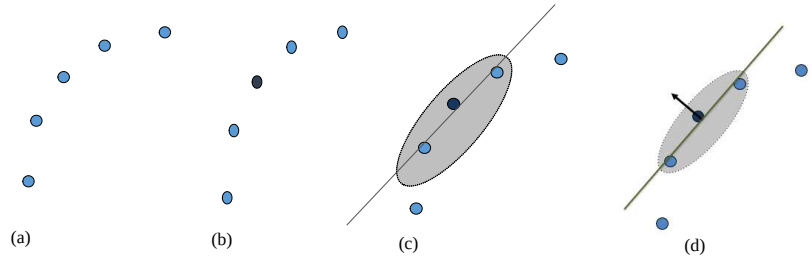

Figure 4. Illustration of estimating the normal vector of a set point in the 3-D space. (a) a set of points; (b) estimation of the normal vector of a black point; (c) selection of two points for estimating a plane; (d) the normal vector of a black point

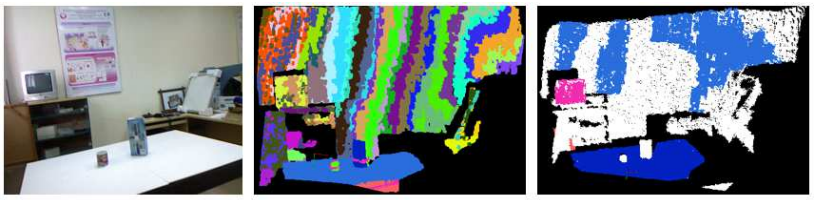

Figure 5. Example of plane segmentation (a) color image of the scene; (b) plane segmentation result with PROSAC [10]; (c) plane segmentation result with the organized point cloud

where $\mathbf{p}_{av}$ represents the 3-D centroid of the nearest neighbors; λ_j is the j^{th} eigenvalue of the covariance matrix; and V_j is the j^{th} eigenvector.

Based on the detected co-planar points, we follow the seed growing regions. Because three-dimensional voxel grid is established and local surface normal is mapped out to the corresponding grid cell. Each cell in the 3-D voxel grid $(n^x_{r,c}, n^y_{r,c}, n^z_{r,c})^T$ (r, c are indices of the each cell) [13] has some points along the normal. The results of this work can generate the cell discretization on the voxel grid. To solve this problem, the average surface normal orientation in two neighbor grid cells are compared. If it is less than the threshold of clustering then they will be added to the existing group. Consequently, the co-planes are merged together.

Thanks to the pre-processing procedures, utilizing the organized point clouds allow for accelerated computational time. Moreover, original technique in [6], that calculated the *'integral image'*, is applied to also reduce computational time. In the experimental evaluations, the table plane detection results are compared in context of utilizing two approaches of the plane segmentation: one inspired by conventional RANSAC algorithms and another is the proposed techniques based on organized point clouds to confirm robustness of the method.

3.4. Table plane detection

Besides color and depth features, a Microsoft Kinect sensor provides acceleration information [13]. This is a vector whose direction points downwards. For each collected frame, there is an acceleration vector, as shown in Fig. 6. Since complex scenes usually contain different planes such

Algorithm 1: Plane segmentation

Input: Organized point cloud: $P = \{p_1, p_2, ..., p_k\}$

Output: Plane candidates: $Planes = \{Planes_i(A_i, B_i, C_i, D_i)\}$

$((n^x, n^y, n^z)^T = ComputeNormal(p_i);$ ▷ Compute normal vector of each point p_i

$(n_i^x, n_i^y, n_i^z)^T = FindNeighBor((n^x, n^y, n^z)^T);$ ▷ Find the k-nearest neighbors of each point p_i

$(n_i^x, n_i^y, n_i^z)^T = SortRegionNeighBor((n_i^x, n_i^y, n_i^z)^T);$ ▷ Sort points in each region including k-nearest neighbors points based on their curvature values

$R_n\{rn_1, rn_2, ..., rn_n\} = Clustering((n_i^x, n_i^y, n_i^z)^T);$ ▷ Clustering region based on normal space

$R_d\{rd_1, rd_2, ..., rd_m\} = Clustering(R_n), (m \leq n);$ ▷ Clustering region based on distance space

$RN(j) = FindNeighbor(R_d), (j = 1, 2, ..., m);$ ▷ Finding r_{nj} neighboring regions of r_j region

$Ang(j) = ComputeAngle(rd_j, RN(j));$ ▷ Compute the angle between normal of rd_j region and $RN(j)$ regions

If$(Ang(j) \leq t)$ **then** ▷ t is the angle threshold to merge regions

{

  $R_i = MergeRegion(rd_j, RN(j));$ ▷ Each region has a fitted plane

  $Push \quad R_i \rightarrow Planes;$ ▷ Add region R_i to $Planes$

}

$Return \quad Planes_i(A_i, B_i, C_i, D_i);$

as the table plane, floor plane, objects' planes, and wall planes. In order to determine the table plane, the non-table planes have to be eliminated based on the criterion as scenario constraints. As a visually impaired people with a mounted Microsoft Kinect on the chest moves around the table, the acceleration vector is perpendicular to the table plane and floor plane.

To do this, a constraint that is the table stands on the floor is utilized. Therefore, the table plane is parallel with the floor plane in the current scene. The acceleration vector gives us a normal vector of the ground/floor plane (and also other planes which are perpendicular to the table plane). The non-table planes that do not meet this criteria are eliminated. Based on this scheme, it is compute the angle (a_i) between the acceleration vector and a normal one of the detected plane pl_i which is obtained from the plane segmentation procedure. If the angle is larger than a threshold t then such detected plane should be eliminated. Otherwise, we consider it as a table plane candidate and move to the second step. The results of the first step are planes that are perpendicular to the acceleration vector. After rotating the y axis such that it is parallel with the acceleration vector, the directions of the acceleration vector and the y axis are the same as Fig. 6. Therefore, the table plane is highest plane in the scene, that means the table plane is the one with minimum y-value. Since the Microsoft Kinect is mounted on the persons chest (as shown in Fig. 1), the table plane is usually the closest plane to the Microsoft Kinect among the detected planes. Therefore, we choose a plane with a minimum y-value to be the detected table plane. Moreover, this plane must have enough number of points *('mininliers')*.

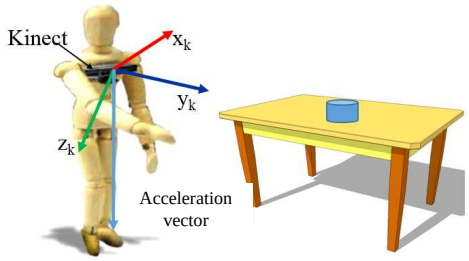

Figure 6. Illustrating acceleration vector provided by a Microsoft Kinect sensor

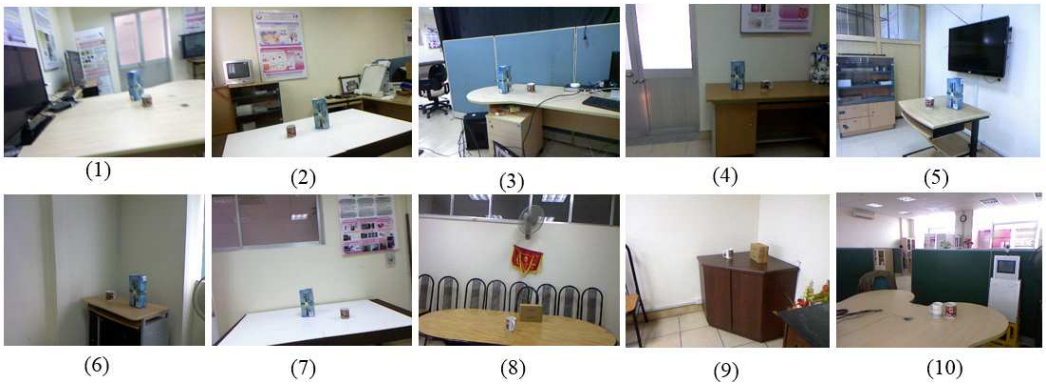

Figure 7. Examples of 10 scenes captured in our dataset

4. EXPERIMENTAL RESULTS AND DISCUSSION

4.1. Setup and dataset

To evaluate the performance of the proposed method, our own dataset and a dataset in [14] are used. Concerning our dataset, the following experiments have been set up: A Microsoft Kinect version 1 is mounted on the person's chest, the person then moves around one table in the room. The distance between the Kinect and the center of the table is about 1.5 m. The height of the Kinect compared with table plane is about 0.6 meter. The height of table plane is about $60 \rightarrow 80$ cm. We capture data of 10 different scenes which include a cafeteria, showroom, and kitchen and, so on. These scenes cover common contexts in daily activities of visually impaired people. Some examples of the captured images of the 10 scenes are shown in Fig. 7. For each scene, the subject is asked to move around a table. Therefore, different view-point is considered in our experiments. The numbers of captured images (at frame rate 5 fps) are given in Table 1. The size of the image is 640×480 pixels. Each frame has a corresponding acceleration vector. The color and depth images are calibrated by using Microsoft Kinect SDK calibration functions.

The second dataset is introduced in [14]. This dataset contains calibrated RGB-D data of 111 scenes. Each scene has a table plane. The size of the image is 640×480 pixels. Some examples of this dataset are illustrated in Fig. 8. Since this dataset does not provide acceleration information, we assume the table plane is the largest plane in a scene.

Table 1. The number of frames of each scene

Scene	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
#frame	950	253	771	292	891	797	411	1717	254	350

Figure 8. Examples of scene in the dataset [14]

4.2. Table plane detection evaluation method

In order to evaluate the proposed technique, it is to prepare the ground-truth of the table planes for two datasets and define three different evaluation measurements. Concerning the ground-truth, the table region on each color image is cropped manually. Such a cropped region gives a mask of the table plane, as shown in Fig. 9b. After that, the corresponding region on the depth image is taken and then presented as a point cloud data. They are referred them as the ground-truth point clouds. For the evaluation measurements, since the table plane detection result could be affected by different factors of the detected planes. In this paper, the accuracy of the detected table plane in term of both table plane's parameters and its size or area is considered. In order to evaluate the parameters, it is to compare the normal vectors delivered from the parameters (e.g., A, B, C) of the detected table plane) and the one extracted from ground-truth data. However, if only using parameters of the plane, the size of the detected table would be omitted. In practical experiments, the size of the detected table can be much smaller than the actual table in the captured scene when projecting the detected plane into the color image. Whereas, if the evaluation bases only on the size/area of the detected table, then the detected plane can be skewed. Therefore, three evaluation measures are needed and they are defined as below.

Evaluation measure 1 (EM1): This measure evaluates the difference between the normal vector extracted from the detected table plane and the normal vector extracted from ground-truth data. First, a 3-D centroid of the ground-truth point cloud data is determined. A normal vector at this point is calculated. The 3-D center point and its normal vector are then extracted from each detected table plane, as illustrated in Fig. 9. After that, an angle (α) between the normal vector of detected table plane and a vector $\mathbf{T}$ that connects the 3-D centroid of the detected table plane to the 3-D centroid of the ground-truth is calculated as shown in Fig. 10(a), (α) is expected to be closed to 90 degree. To evaluate a true detection, a lower and upper threshold for this angle is set up. In this paper, it is to set thresholds of 85 degrees for the lower and 95 degrees for the upper. A limitation of this evaluation is that the planes can consists of many noises, so the evaluation results can be affected by these noises because 3-D centroid is determined from all points (including noises) belonging to the plane. Therefore, the second evaluation measure to overcome this issue is proposed.

Evaluation measure 2 (EM2): By using EM1, only one point was used (center point of the

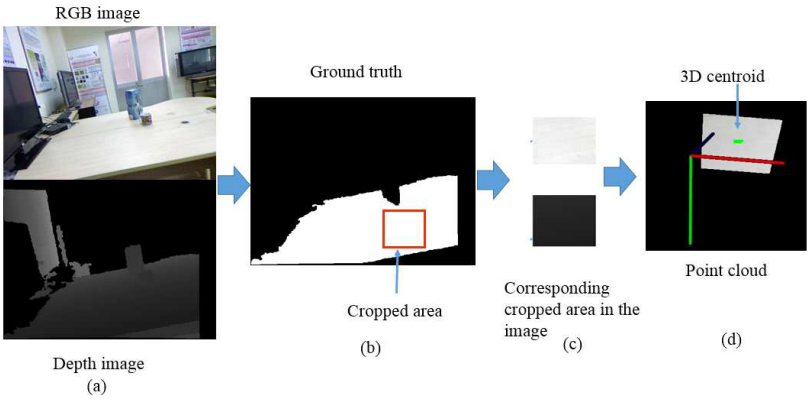

Figure 9. (a) Color and depth image of the scene; (b) mask data of table plane; (c) cropped region; (d) point cloud corresponding of the cropped region, green point is 3-D centroid of the region

ground-truth) to estimate the angle. To reduce the noise influence, more points for determining the normal vector of the ground truth are used. For the EM2, 3 points (p_1, p_2, p_3) are randomly selected from the ground-truth point cloud. After that, a normal vector $\mathbf{n}$ of the plane constituted from two vectors $\mathbf{u}$ and $\mathbf{v}$ is computed as follows:

$$\mathbf{n} = [\mathbf{u}, \mathbf{v}] \tag{6}$$

where $\mathbf{n}$ generated from cross product of two vectors; $\mathbf{u}$ is a vector from p_1, p_2 points; $\mathbf{v}$ is a vector from p_1, p_3 points. The angle (β) between the normal vector of the detected table plane and $\mathbf{n}$ vector is computed. Since two vectors are parallel, the angle β should be 0 degree. However, in order to make the evaluation measure to be tolerant to the noise, an upper threshold for (β) is set. If (β) is lower than this threshold, it is defined as a true detection. It is to set this threshold of 5 degree in our experimental evaluations (see Fig. 10b).

Evaluation measure 3 (EM3): The two evaluation measures presented above do not take into account the area of the detected table plane. Therefore, it is to propose EM3 that is inspired by the Jaccard index for object detection [11]. First, the detected table plane is projected from the point cloud into the RGB image space, named R_d and an area of the table with manually annotated ground truth on the image, named R_g is generated. Then, it is to compute the ratio r between the intersection and union between detected and ground-truth regions as follows:

$$r = \frac{R_d \cap R_g}{R_d \cup R_g}. \tag{7}$$

If r is greater than a threshold, we define it as a true detection, otherwise a false detection. In [7], the author used only the overlapping region for determining a true detection. This may be wrong evaluation if the detected table plane covers whole image. By utilizing EM3, results of the table plane detection could be evaluated more accurate. For example, the detected table plane could be true when the number of data points satisfies the constraints of the EM1, and EM2; but its size is not satisfied according to the EM3 measurement, then it is a false detection. Although, EM1 and EM2 are the traditional evaluations for the plane detection techniques. In this paper, three independent measurements are utilized besides the computational time, to confirm the robustness of the proposed

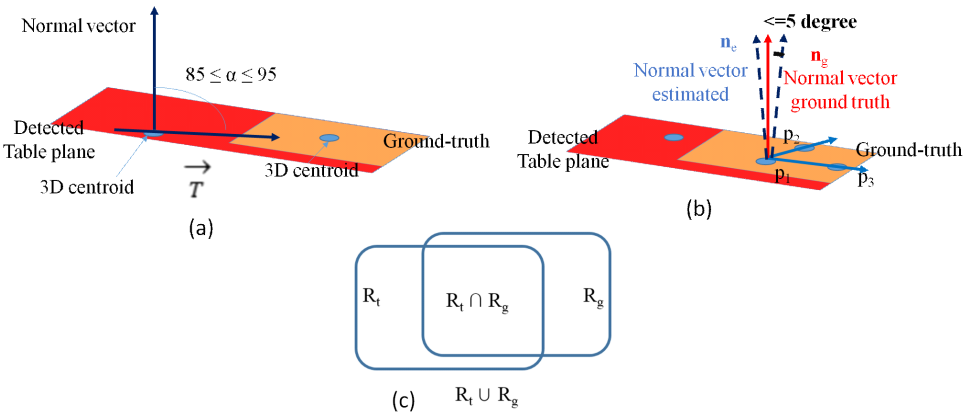

Figure 10. (a), Illustration of the angle between normal vector of the detected table plane and $\mathbf{T}$. (b), Illustration of the angle between normal vector of the detected table plane $\mathbf{n}_e$ and $\mathbf{n}_g$; (c) Illustration of overlapping and union between detected and ground-truth regions

technique. Moreover, also the recall and missing rate of the table plane detection are computed. Recall is defined by the ratio between the number of true detections and the number of table planes in the dataset while the missing rate is defined as a ratio between the number of missed table planes and the number of table planes.

4.3. Results and discussion

In order to evaluate the proposed method, we compare it with two other methods. Three methods are mainly different from one another in term of the implementation techniques for the plane segmentations. The first baseline method, PROSAC is deployed (see details in [10]), whereas the second one utilizes the techniques proposed in [6, 7]. The proposed method, as described in Sec. 3., utilizes the down-sampling and smoothing techniques to reduce computational time.

Some parameters are set for the first method as follows: minimum number of points in a plane are 100 points; the thresholds to determine a point belonging the plane in PROSAC algorithm [10] is $(t = 0.1)$ (that means 10cm); the threshold to remove floor plane h is 120 cm. Regarding the second, our method, the parameters are defined as follows: minimum number of points in the region (mininliers) are 1000 points; the threshold for clustering and merging two regions data: angle threshold is 5 degrees, the distance threshold is 0.05 (that means 5cm); the threshold to remove floor plane h is 120 cm. For the proposed method, we set (mininliers) equal 300 points, down sampling image with size (3×3). The others parameters are identical to the second one.

These methods are compared with the experimental dataset as described in section 4.1. All methods are written in C++ and using the PCL 1.7 and OpenCV 2.4.9 libraries on a PC with Core $i5$ processor RAM 8G. The comparative results of three different evaluation measures on two datasets are shown in Table 2 and Table 3 respectively, while the detail result for each scene of our dataset is illustrated in Fig. 11.

The results of our dataset show that the methods based on organized point clouds (second method and the proposed method) obtain not only good results in terms of precision and low missing rate but also computational efficiency. The proposed method obtains similar results compared to the second

Table 2. The average result of detected table plane on our own dataset(%)

Approach	Evaluation Measurement				Missing rate	Frame per second
	EM1	EM2	EM3	Average		
First Method [10]	87.43	87.26	71.77	82.15	1.2	0.2
Second Method [6]	98.29	98.25	96.02	97.52	0.63	0.83
Proposed Method	**96.65**	**96.78**	**97.73**	**97.0**	**0.81**	**5**

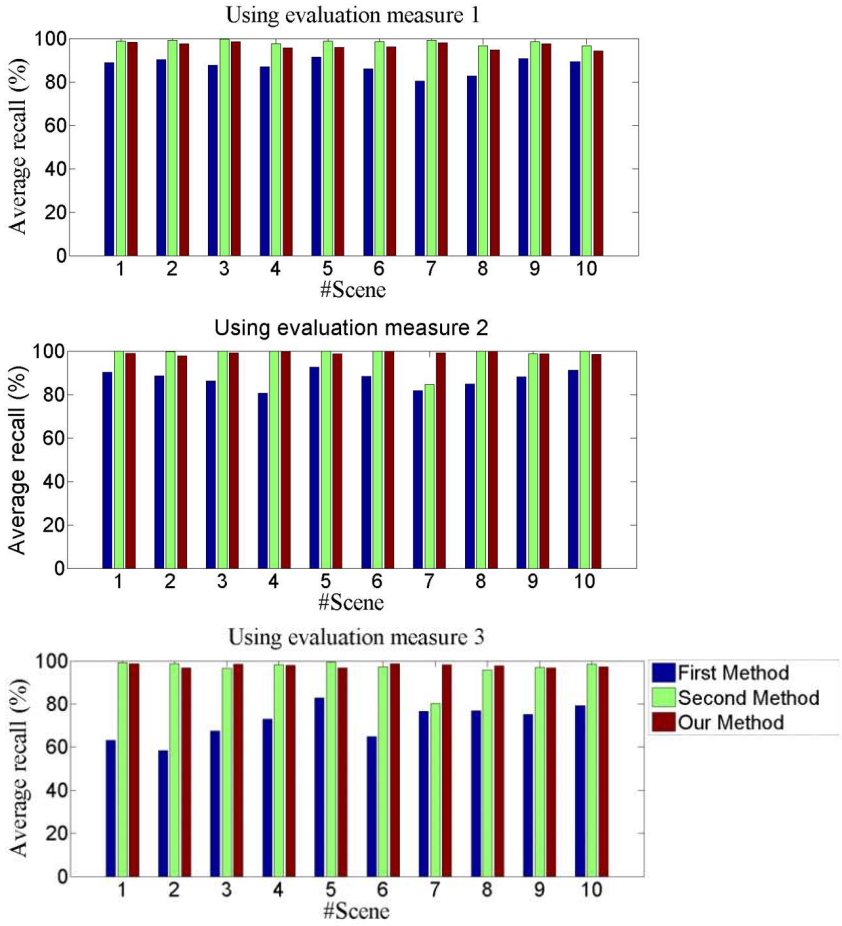

Figure 11. Detailed results for each scene of the three plane detection methods on our dataset: (a) using the first evaluation measure; (b) the second evaluation measure and (c) the third evaluation measure

one (the average recalls are 97% and 97.52% respectively). However, our method gets 5 frames per second while the second method can only process 0.83 frames per second. The first one has the lowest recall (82.15%) and lowest computational time (0.2 frames/second) because, this method is based

on PROSAC algorithm and may generate some wrong planes. In term of missing rate, our method and the second one have a lower missing rate in comparison with the first method. Some examples of table detection results of the proposed method are illustrated in Fig. 12. It is interesting to see that the detection results are coherent with three evaluation measures. Since the third measure uses a strict constraint on detected area, the recall of three methods decreases. Especially for the first detection method that is based on PROSAC, it can generate many planes with a small area. An illustration of this problem is shown in (Fig. 13 - top line). In this example, the detections of the first method are correct if using the first and second evaluation measure. Since the angle (α) and the angle (β) defined in the first and second evaluation measures are 85.68% and 4.17% respectively. However, the r rate of the third measure is 21.3%, which does not satisfy the defined criteria.

The results of the method on the dataset [14] (see Table 3) show that our method obtains a similar accuracy than the methods in [6], [10] on EM1 and EM2 and outperforms these methods on EM3. However, this dataset has more noise than our dataset. Therefore, the obtained accuracy is lower than that of our dataset with EM1, EM2 measures. Concerning EM3 measure, since this dataset is less complex than our dataset in term of number of planes appeared in the scene (one scene in this dataset normally has a table plane, a wall plane and a ground plane), accuracy with EM3 is quite high.

The table detection results on two testing datasets shows that the proposed method achieves good performance (greater than 97% for EM3) with an acceptable frame rate for working application (5 frames per second).

Table 3. The average result of detected table plane on the dataset [14] (%)

Approach	**Evaluation Measurement**				**Missing rate**	**Frame per second**
	EM1	**EM2**	**EM3**	**Average**		
First Method [10]	87.39	68.47	98.19	84.68	0.0	1.19
Second Method [6]	87.39	68.47	95.49	83.78	0.0	0.98
Proposed Method	**87.39**	**68.47**	**99.09**	**84.99**	**0.0**	**5.43**

Since our method applies down sampling in this paper, the table detection result is compared with different down sampling factors. The chosen down sampling factors are (3×3), (5×5) and (7×7). The results are listed on Tab. 4.3.. The table plane detection results are inversely proportional to the down sampling factor and processing time. The choice of down sampling depends on the system requirements. With 33 frames per second, 84.13% of average precision is obtained. In some cases, this result can be accepted because when the system can not detect the table, the visually impaired people can move lightly in the scene until the Microsoft Kinect can capture better the table.

However, in some cases, all methods could not detect well the table plane because the number of points belonging to the table is smaller than a threshold $mininliers$. This case is illustrated in Fig. 13- bottom line. In this failure case, none of the planes could be detected in the plane segmentation step.

5. CONCLUSIONS AND FUTURE WORKS

In this paper, a method for table plane detection using down sampling, accelerometer data and organized point cloud structure obtained from color and depth images of the Kinect sensor is proposed.

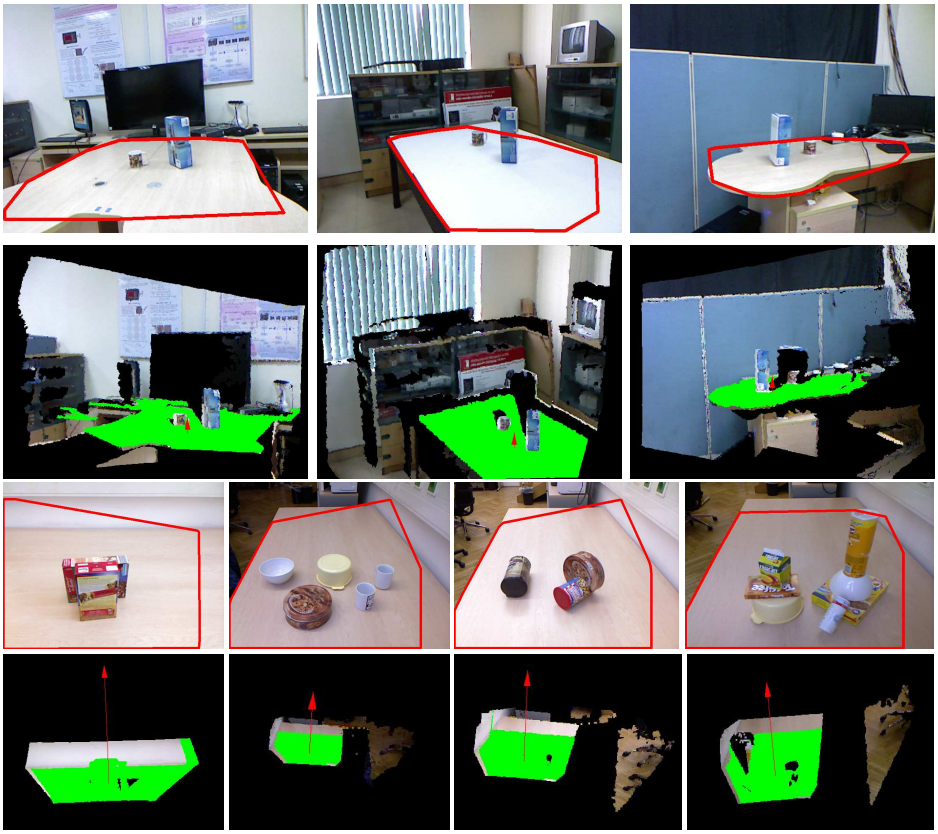

Figure 12. Results of table detection with our dataset (two first rows) and the dataset in [14] (two bottom rows). Table plane is limited by the red color boundary in image and by green color points in point cloud. Arrow with red color is normal vector of detected table

Table 4. The average result of detected table plane of our method with different down sampling factors on our dataset

Down sampling	**Average recall (%)**	**nbframe per second**
(3x3)	97.00	5
(5x5)	92.21	14
(7x7)	84.13	33

The method outperforms the baseline methods in term of both precision and computational time. A table plane dataset with the ground-truth has been built in the context of the object-finding aid system. In order to evaluate the proposed method, the authors have performed a quantitative comparison of the method with two different methods in the literature. Moreover, to confirm the robustness of the proposed method, three different evaluation measures are utilized. It obtained a 97% of precision rate with a frame processing rate of 5Hz on the captured dataset. In this research context,

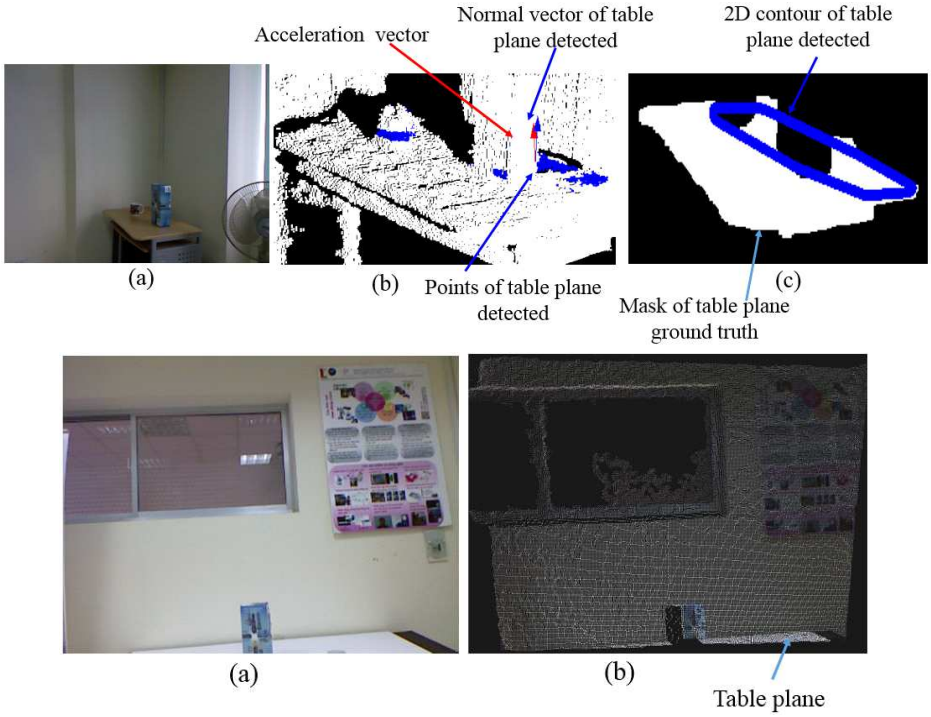

Figure 13. Top line is an example detection that is defined as true detection if using the two first evaluation measures and as false detection if using the third evaluation measure: (a) color image; (b) point cloud of the scene; (c) the overlap area between the 2-D contour of detected table plane and the table plane ground-truth. Bottom line is an example of missing case with our method (a) color image, (b) point cloud of the scene. After down sampling, the number of points belonging to table is 276 that is lower than our threshold

a dataset consisting of 10 different scenes is collected and the frames of each scene were collected from different perspectives. The dataset is collected in the common environments. Therefore, it can satisfy the requirements of an aided-system for object finding. Based on the table plane detection results, we will continue to perform object detection and localization. Finally the aided system is completed with a communication module that sends the object information to visually impaired people.

ACKNOWLEDGMENT

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

REFERENCES

[1] H. Badino, "Least Squares Estimation of a Plane Surface in Disparity Image Space," *Technical report in Carnegie Mellon University Pittsburgh*, 2011.

[2] D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter, "The 3D Hough Transform for Plane Detection in Point Clouds : A Review and a new Accumulator Design," *3D Research*, vol. 2, no. 2, 2011.

[3] S. Choi, T. Kim, and W. Yu, "Performance Evaluation of RANSAC Family," in *Procedings of the British Machine Vision Conference*, 2009, pp. 1–11.

[4] O. Chum and J. Matas, "Matching with PROSAC - progressive sample consensus," in *Proceedings of the Computer Vision and Pattern Recognition*, 2005, pp. 220–226.

[5] J. E. Deschaud and F. Goulette, "A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing," in *Proceedings of the 5th International Symposium on 3D Data Processing (3DPVT)*, 2010.

[6] S. Dirk Holz, R. B. Rusu, and S. Behnke, "Real-Time Plane Segmentation Using RGB-D Cameras," in *LNCS (7416): RoboCup 2011 - Robot Soccer World Cup XV*, 2011, pp. 306–317.

[7] C. Feng, Y. Taguchi, and V. Kamat, "Fast Plane Extraction in Organized Point Clouds Using Ag-glomerative Hierarchical Clustering," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6218–6225.

[8] R. Fischler and M. A. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysisand Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[9] T. Huang, G. Yang, and G. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 27, no. 1, pp. 13–18, 1979.

[10] L. V. Hung, V. Hai, N. T. Thuy, L. T. Lan, and T. T. T. Hai, "Table plane detction using geometrical constraints on depth image," in *Proceedings of the National conference on Fundamental and applied IT research (FAIR)*, 2015, pp. 647–657.

[11] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[12] N. Jagadeesan and R. Parvathi, " An Efficient Image Downsampling Technique Using Genetic Algorithm and Discrete Wavelet Transforman," *Journal of Theoretical and Applied Information Technology*, vol. 61, no. 3, pp. 506–514, 2014.

[13] J. Kramer, N. Burrus, F. Echtler, H. C. Daniel, and M. Parker, *Hacking the Kinect*. Apress, 2012. [Online]. Available: http://link.springer.com/10.1007/978-1-4302-3868-3

[14] A. Richtsfeld, T. Mrwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 4791–4796.

[15] J. S. Simonoff, *Smoothing Methods in Statistics*. Springer, 1998.

[16] M. Trentacoste, R. Mantiuk, and W. Heidrich, " Blur-Aware Image Downsampling," *EUROGRAPHICS*, vol. 30, no. 2, 2011.

[17] M. Y. Yang and W. Forstner, "Plane Detection in Point Cloud Data," *Technical report Nr.1 of Department of Photogrammetry, Institute of Geodesy and Geoinformation, University of Bonn*, 2010.

Received January 12 - 2016
Revised January 23 - 2017