Real-Time Plane Segmentation and Obstacle Detection of 3D Point Clouds for Indoor Scenes

Zhe Wang, Hong Liu, Yueliang Qian, and Tao Xu

Key Laboratory of Intelligent Information Processing &&
Beijing Key Laboratory of Mobile Computing and Pervasive Device
Institute of Computing Technology, Chinese Academy of Sciences
Beijing 100190, China
{wangzhe01,hliu,ylqian,xutao}@ict.ac.cn

Abstract. Scene analysis is an important issue in computer vision and extracting structural information is one of the fundamental techniques. Taking advantage of depth camera, we propose a novel fast plane segmentation algorithm and use it to detect obstacles in indoor environment. The proposed algorithm has two steps: the initial segmentation and the refined segmentation. Firstly, depth image is converted into 3D point cloud and divided into voxels, which are less sensitive to noises compared with pixels. Then area-growing algorithm is used to extract the candidate planes according to the normal of each voxel. Secondly, each point that hasn't been classified to any plane is examined whether it actually belongs to a plane. The two-step strategy has been proven to be a fast segmentation method with high accuracy. The experimental results demonstrate that our method can segment planes and detect obstacles in real-time with high accuracy for indoor scenes.

Keywords: plane segmentation, point cloud, obstacle detection.

1 Introduction

Understanding the structural information of the surrounding environment is a principal issue for indoor service robots and wearable obstacle avoidance devices. There are many man-made planes in indoor scene. Plane detection and segmentation is the fundamental technique for understanding the structure and can be used in many important applications, such as robot navigation, object detection, scene labeling and so on. The accuracy of plane segmentation is highly related to the performance of the whole scene understanding system. Furthermore, as the basic step of scene understanding systems, plane segmentation should be processed in real-time.

Recently, depth cameras, such as the Microsoft Kinect, generate a booming effect in the computer vision filed. Kinect acquires depth information at high frame rates (30Hz) and the accuracy is roughly equal to 3D laser scanners in low ranges. With depth data, the shape, size, geometric information and several other properties of the objects can be more precisely determined. As a result, depth cameras greatly stimulate researchers' enthusiasm and have been widely used in plenty of computer vision systems, such as mobile robot navigation[1], 3D reconstruction[2] and indoor navigation systems[3].

Based on single depth camera from one view, we focus on fast plane segmentation method and use it to detect obstacles in indoor scenes. Depth image grabbed by single depth camera has some noises. Furthermore, the resolution of depth image by Kinect is 640×480 pixels, which has excessive amount of data for real-time processing.

In this paper we present an algorithm to accurately segment all planes using depth image, and then detect the obstacles in indoor scenes. A novel two-step strategy PVP for fast **P**lane segmentation mainly consists of **V**oxel-wise initial segmentation and **P**ixel-wise accurate segmentation is proposed. Firstly, we convert the depth image into 3D point cloud and divide the point cloud into voxels. Then the normal of each voxel is calculated and an area-growing based algorithm is used to extract the candidate planes according to the normal of each voxel. Secondly, each point in point cloud that hasn't been classified to any plane is examined whether it actually belongs to a plane or not. Then the fragments of the same plane are merged together. The experimental results show the proposed two-step strategy PVP is a fast segmentation method with high accuracy.

2 Related Work

Many plane segmentation algorithms for 3D data have been proposed in recent years. Some researchers treat depth images as 2D images and apply 2D segmentation methods on depth images directly. The only difference is that the value of each pixel in depth images is not in the range of 0 to 255. When adopting advanced segmentation algorithms like mean-shift clustering[4] or graph-based segmentation[5], this approach gives good results in some scenes. However, in other situations such as the case of close or touching objects, these methods perform badly. Some algorithms, like[6] or[7], use RANSAC or J-linkage[8] to robustly estimate the parameters of all planes in a depth image. But this kind of algorithms is usually slow and can hardly be adopted in real-time systems. Furthermore, these methods are unsuitable for large and complex scenes such as office rooms. So some researchers begin to find other more efficient methods. Holz et al.[9] use integral images to compute local surface normals of point clouds. Then the points are clustered, segmented, and classified in both normal space and spherical coordinates. The experiments show the algorithm could achieve a frame rate at 30Hz at the resolution of 160×120 pixels. However other advanced tasks like object classification may need a higher resolution. Dube et al.[10] extract planes from depth images based on the Randomized Hough Transformation. They use a noise model for the sensor to solve the task of finding proper parameter metrics for the Randomized Hough Transform. This algorithm is real-time capable on the platform of a mobile robot. However it can only detect planes, and cannot accurately segments the planes.

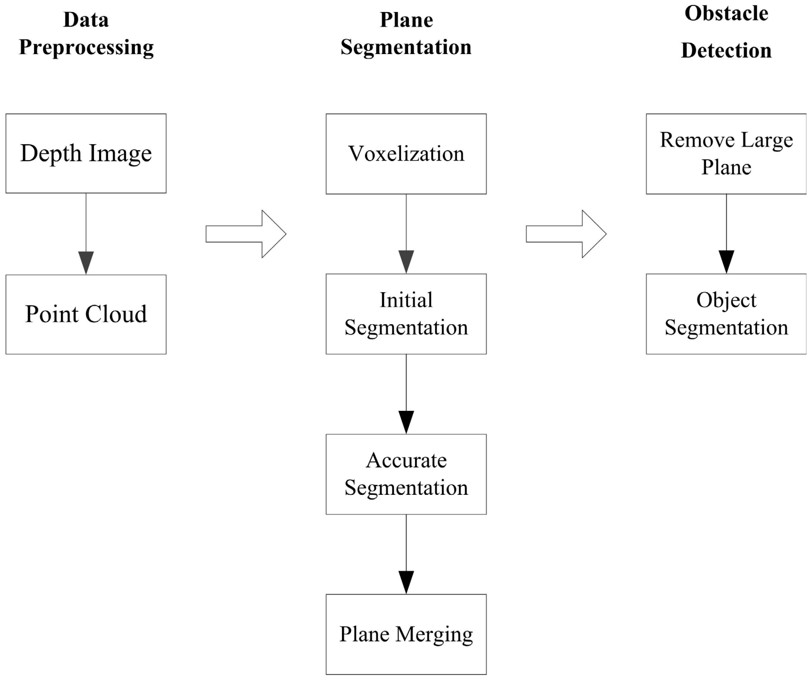

Fig. 1. System architecture

3 Overview of Our Approach

Fig. 1 gives the architecture of our system. Basically, our system contains three stages: data preprocessing, plane segmentation and obstacle detection. First of all, in the data preprocessing stage, the depth image obtained from the Kinect is converted to the corresponding 3D point cloud. Then a voxel grid is created on the point cloud. For each voxel, a plane equation is estimated using least square estimation technique and the normal of the plane can be obtained from the coefficient of the equation. Next, in the plane segmentation stage, an area-growing based algorithm is firstly adopted to extract candidate planes at the initial segmentation step. Then the remaining points are examined one by one to achieve accurate segmentation. For each point, we search in the 3D space within a certain radius to determine whether it belongs to a plane or not. Plane fragments are merged together at the last step of plane segmentation. In the obstacle detection stage, all the planes are removed from the point cloud. Then we use an area-growing based algorithm to extract point clusters and each point cluster is considered to be an obstacle.

4 Real-Time Plane Segmentation Algorithm PVP

3D point cloud contains much structural information that can be used for plane segmentation. However, the point cloud obtained from the Kinect has a large amount of data. So algorithms for point cloud are often time consuming. In this paper, we propose a novel two-step plane segmentation algorithm PVP that firstly applies a voxel-wise initial segmentation and then a pixel-wise refinement. The voxel-wise segmentation gives a rough but extremely fast result and then the pixel-wise refinement greatly enhance the accuracy. The combination of the two steps is a great balance between speed and accuracy.

4.1 Initial Plane Segmentation

Normally, plane segmentation algorithms would check each pixel of the image to determine whether this pixel can be classified to a certain plane. However, we find that in 3D point cloud, planes always consist of many flat pieces of points. These pieces have the same normal and are next to each other. Inspired by this idea, we construct voxel grid on the point cloud and let each voxel be the smallest unit to do initial plane segmentation. A voxel is a small 3D box containing several points as Fig. 2 shows. The side-length of the voxel used in our experiments is 20 cm. For each voxel, we use the least square estimation technique to estimate a plane equation that fits all the points in the voxel. Assume that the plane equation is $Ax + By + Cz + 1 = 0$, the least square estimation for A, B, C is

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i z_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i z_i \\ \sum x_i z_i & \sum y_i z_i & \sum z_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum x_i \\ \sum y_i \\ \sum z_i \end{bmatrix} \tag{1}$$

where $(x_i, y_i, z_i), i = 1, 2, 3, ..., N$ represents a point in the voxel. It is an obvious result in analytic geometry that vector $[A\ B\ C]^T$ is exactly the normal of this plane. After calculating the plane equations for all the voxels, we use the normals to extract initial planes.

Plane segmentation can be treated as a clustering problem. However, if we directly cluster points in the 3D point cloud, the efficiency of the algorithm would not meet the demand of a real-time system. We notice that planes in point cloud consist of small flat pieces, and the pieces of the same plane have the same normal direction. After constructing voxel grid on 3D point cloud, each voxel actually is a piece of points. If the adjacent voxels have the same normal direction, they can be considered in the same plane. So we obtain all the clusters of voxels and in each cluster the normal direction of the voxels are the same. Finally if a voxel cluster is larger than a threshold, it is determined to be a plane. The detailed algorithm is described in Algorithm 1. The number of adjacent voxels is a parameter whose value is 26 in our experiments.

After these steps, we can get many voxel clusters. The cluster contains more than a certain number of voxels is considered as a candidate plane. The voxels

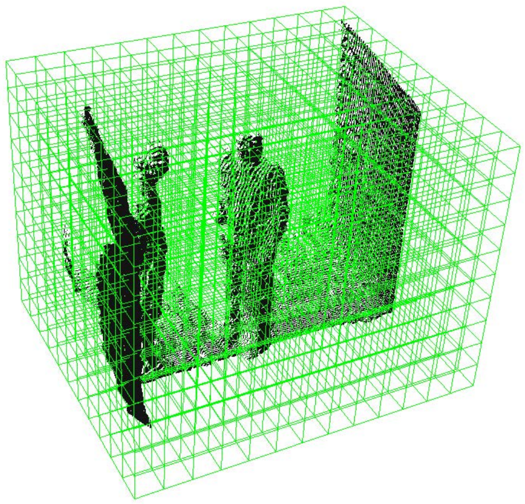

Fig. 2. Construct voxel grid on 3D point cloud

belong to a plane are called planar-voxel, and the remaining voxels are called non-planar-voxel. Points in all the non-planar-voxels should be accurately examined pixel by pixel in the refined segmentation. Furthermore, the voxel-wise plane segmentation is less sensitive to noise because several noise points would not influence the normal of a voxel.

4.2 Refined Plane Segmentation

After initial plane segmentation, we get several roughly segmented candidate plans. In order to obtain an accurate segmentation result, we check each point in the non-planar point cloud to determine whether it belongs to a certain existed plane. We know that if a point is on a plane, then the distance from the point to that plane is near zero. For each point, the distance between it and a candidate plane is calculated. If the distance is small, the point is considered belongs to that plane. However, comparing with every candidate plane is too global, neglecting the local structural restrictions. So we search the neighborhood of a point for candidate planes. And if there are planes near this point, then we just calculate the distance between the point and these neighbor planes to determine which plane the point belongs to.

More specifically, we traverse the voxels grid and find the voxels that haven't been clustered to a plane. Then we take one of these voxels as the center and search the space surrounding the center with radius r. In order to speed up the algorithm, we just search in a cube of side r cm and r is an integral multiple of the length of the voxel's side. If there are no planar-voxels near the center, then all the points in the central voxel are considered as non-planar points. Otherwise, if there are planar-voxels near the center, then for each point in the central voxel,

Algorithm 1. Initial Plane Segmentation

```
1: Traverse the voxel grid and find a voxel $V_0$ that hasn't been processed before.
   Calculate the average distance $d$ of the points in $V_0$ to the estimated plane of $V_0$.
   If $d$ is smaller than a threshold then create a queue $Q$ and add $V_0$ to $Q$. Otherwise,
   find another $V_0$;
2: Examine each of the 26 adjacent voxels $V_i (i = 1, 2, ..., 26)$ surrounding $V_0$. If $V_i$
   hasn't been processed, then go on with the calculation. Let $n_i = (x_i, y_i, z_i)$ denote
   the normal of $V_i$ and $n_0 = (x_0, y_0, z_0)$ denote the normal of $V_i$. Calculate the cosine
   of the angle of $n_i$ and $n_0$ :
       $\cos\theta = \frac{x_i \cdot x_0 + y_i \cdot y_0 + z_i \cdot z_0}{|n_i| \cdot |n_0|}$   (2)
   If $|\cos\theta|$ is larger than a threshold, record $V_i$ and $V_0$ belongs to the same plane
   and then insert $V_i$ into $Q$;
3: Pick up an element from $Q$ and regard it as $V_0$. Then return to step 2;
4: If $Q$ is empty, return to step 1;
5: Repeat step 1 to step 4 until all the voxels are examined;
```

we calculate the distances between the point and every neighbor planar-voxel. The plane equation of each voxel has been calculated when we create the voxel grid, so we can directly use the result to calculate the distance. Let p denotes the coordinate of one point in the central voxel, V_{ij} denotes the ith planar-voxel in the neighborhood that belongs to plane P_j and the plane equation of V_{ij} is $Ax + By + Cz + 1 = 0$. Then the distance from p to V_{ij} is:

$$d_{ij} = \frac{|Ax^{'} + By^{'} + Cz^{'} + 1|}{\sqrt{A^2 + B^2 + C^2}} \tag{3}$$

For each planar-voxel we recorded, the distance is calculated. Then we choose $d = \min d_{ij}$. If d is smaller than a threshold, then point p is considered a point in the corresponding plane P_j. Otherwise point p is a not a point in a plane. All the points in the central voxel are determined like this. After applying this process for all the non-planar-voxels, we can get a refined plane segmentation result.

After the fine segmentation, all the planes are segmented. However, some planes may be divided into several parts. So the planes actually belong to one big plane should be merged together. For each segmented plane, we estimate the plane normal based all the points that belong to the plane. If the angle of two normal directions of two planes is smaller than a threshold and the two planes are adjacent with each other, then they are merged as one plane and the new plane would replace these two planes. Repeat these steps until no plane can be merged with another. Fig. 3 presents an example of the result after accurate segmentation and plane merge.

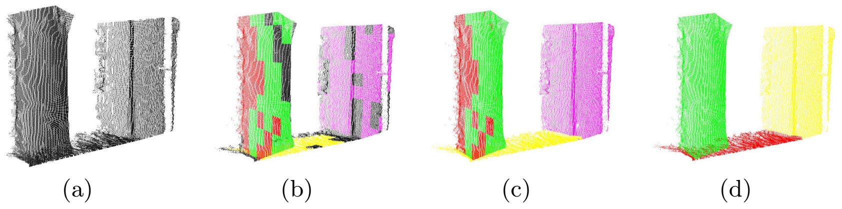

Fig. 3. An example of plane segmentation. (a) is the original point cloud. (b) shows the result of initial segmentation. Gray points represent the points that haven't been divided to a plane. And points are painted with color if they belong to a plane. (c) is the result of accurate segmentation. (d) is the result after plane merging.

5 Obstacle Detection

Plane segmentation is a fundamental technic and can be used in obstacle detection, scene segmentation, environment understanding and so on. We apply our plane segmentation algorithm PVP in an indoor obstacle detection system and it performs very well. The idea of our obstacle detection algorithm is that ground and walls are planes in indoor environment. Obstacles often lie on the ground or beside the walls. If the ground and walls can be removed, the remaining objects are all obstacles.

Following this idea, we propose an algorithm to detect obstacles in the point cloud. At first, all the planes are extracted using our plane segmentation algorithm PVP. Then we set a threshold to filter out large planes because they may be ground or walls in the scene, while the small planes could be the surfaces of desks or boxes. Next, the large planes are removed from the point cloud, so the remaining points are potential obstacles. To segment all the obstacles, we adopt an area-growing based algorithm. We do not directly cluster the points in the point cloud because it's too slow. Here we use the same idea as the initial segmentation of our plane segmentation algorithm. A voxel grid is created at first, and then we use the area-growing method to extract every voxel cluster. In area-growing, voxels are clustered if they are adjacent and both contain more than a certain number of points. The area-growing algorithm used for obstacle detection is different with that of the plane segmentation as the growing strategy is different. This technique can correctly handle weakly connected obstacles, preventing them to be clustered together. After area-growing, we can get several voxel clusters. All the points in voxels of the same cluster form an obstacle.

The method is simple and fast, which makes it very suitable for indoor navigation system or guiding system. Furthermore, as we detect obstacles in 3D environment, the size, direction and distance of each obstacle can be easily calculated. The above information can be used for automatic navigation of indoor robots or wearable guidance devices.

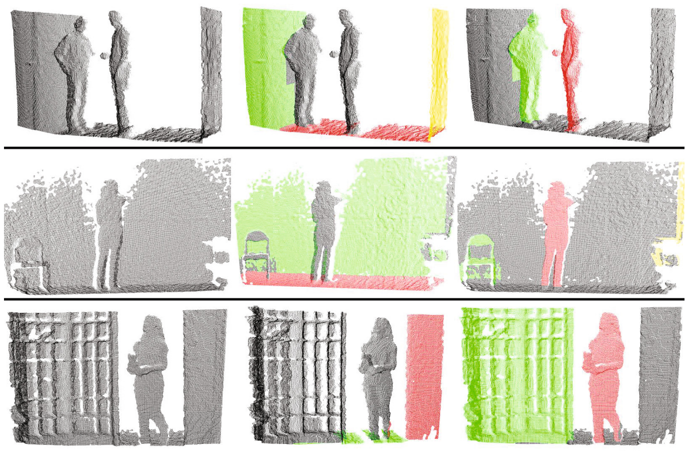

Fig. 4. Plane segmentation and obstacle detection results of three scenes. The first row is a sample of the corridor. The second is the classroom and the third is the office room. In each row, the first column is the original point cloud generated from depth images. The second column is the result of plane segmentation. Different planes are painted with different colors. The third column is the result of obstacle detection. Different obstacles are painted with different colors.

6 Experimental Results

We test the proposed algorithm on the dataset from[11]. The dataset is acquired from a Kinect mounted on a remotely controlled robot. The results have been measured over 30 depth images taken in 3 different scenes: classroom, office room and corridor. Each scene contains 10 images and totally 30 images. Firstly we use our algorithm to segment large planes like ground and walls. Then use the result to detect obstacles in the scene. Fig. 4 shows the results of plane segmentation and obstacle detection of each scene.

Our system is deployed on an Intel Core i7 2.0 GHz CPU laptop computer. The algorithms run sequentially on a single thread within a single core. As the detection rates would not be affected by the input image and the CPU we used is the bottom model of i7 series, which means our CPU would be no powerful than the CPU used in[9], we just take the result from[9] to make a comparison. The results are recorded in Table 1. Clearly our method achieves real-time processing at VGA resolution while Holz's method only runs at 7Hz under this resolution. When the image resolution reduced, our method also performs much better than Holz's method. Data shows that our algorithm is truly capable of real-time processing. And if we take advantage of multi-thread technique on multi-core CPU, the frame rate would be improved significantly.

A subjective evaluation is also conducted to quantitatively analyze the proposed algorithm. For each depth image, we label all the planes and obstacles in the image and then manually analyze the performance of our system according to the ground truth labels. In order to quantitatively measure the accuracy of our system, we calculate two commonly used benchmark recall and precision in our dataset. The results are presented in Table 2. The performance is related to the complexity of the scene. In complex scenes, there are less planar areas so that the plane segmentation results are bad and thus influence the performance of obstacle detection. It is clear that the scenes of the classroom and the corridor are simple and the scenes of the office room are complex. So it is natural that both the recall and precision are high in classroom and corridor. And for a more complex environment, like office room in our experiment, the performance is degraded.

Table 1. Frame rate of different resolution. The processing time of our method is measured and compared with Holz's method.

Image resolution	Holz's method	Our method
VGA(640 × 480 pixels)	7Hz	25Hz
QVGA(320 × 240 pixels)	27Hz	67Hz
QQVGA(160 × 120 pixels)	30Hz	130Hz

Table 2. Recall and precision of the system. For each scene, the recall and precision is calculated. And then the total results are given.

Scene	Plane segmentation		Obstacle detection	
	recall	precision	recall	precision
Classroom	100%	100%	100%	85%
Office room	90.91%	96.30%	97.83%	91.30%
Corridor	100%	100%	100%	92%
Total	97.18%	98.70%	98.94%	89.19%

7 Conclusion

In this paper, we propose a two-step plane segmentation method which is capable of real-time segmenting VGA resolution depth images. The depth image is converted into 3D point cloud and a voxel grid is created on it. Then we apply the voxel-wise initial segmentation on the voxel grid. Next, the pixel-wise accurate segmentation is used to refine the result. After plane segmentation, we use the result to implement obstacle detection in indoor environment. We implement the system on a laptop computer and conduct several experiments to evaluate the algorithms. The frame rate is measured and compared with another algorithm and the accuracy is quantitatively analyzed. The experimental result shows our method is fast and accurate. Thus, our system proves to be very suitable for indoor robots navigation or object avoidance systems, which require high efficiency. As an extension of this work, we are going to combine gray or color information acquired simultaneously from the Kinect to recognize detected objects.

Acknowledgments. The research work is supported by the National Nature Science Foundation of China No.60802067 and the Nature Science Foundation of Ningbo No.2012A610046.

References

1. Benavidez, P., Jamshidi, M.: Mobile robot navigation and target tracking system. In: International Conference on System of Systems Engineering (SoSE), pp. 299–304 (June 2011)
2. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., Fitzgibbon, A.: Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, pp. 559–568 (2011)
3. Mann, S., Huang, J., Janzen, R., Lo, R., Rampersad, V., Chen, A., Doha, T.: Blind navigation with a wearable range camera and vibrotactile helmet. In: Proceedings of the 19th ACM International Conference on Multimedia, pp. 1325–1328 (2011)
4. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 603–619 (May 2002)
5. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision, 167–181 (September 2004)
6. Zuliani, M., Kenney, C.S., Manjunath, B.S.: The multiransac algorithm and its application to detect planar homographies. In: International Conference on Image Processing, pp. 153–156 (2005)
7. Schwarz, L.A., Mateus, D., Lallemand, J., Navab, N.: Tracking planes with time of flight cameras and j-linkage. In: 2011 IEEE Workshop on Applications of Computer Vision (WACV), pp. 664–671 (2011)
8. Toldo, R., Fusiello, A.: Robust Multiple Structures Estimation with J-Linkage. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 537–547. Springer, Heidelberg (2008)
9. Holz, D., Holzer, S., Rusu, R.B., Behnke, S.: Real-time plane segmentation using rgb-d cameras. In: RoboCup Symposium (2011)
10. Dub, D., Zell, A.: Real-time plane extraction from depth images with the randomized hough transform. In: IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1084–1091 (2011)
11. Wolf, C., Mille, J., Lombardi, L., Celiktutan, O., Jiu, M., Baccouche, M., Dellandra, E., Bichot, C.E., Garcia, C., Sankur, B.: The liris human activities dataset and the icpr human activities recognition and localization competition. Technical report, LIRIS Laboratory (March 28, 2012)