2023 20th International Conference on Ubiquitous Robots (UR)
June 25-28, 2023. Hawaii Convention Center, Honolulu, Hawaii

# Fast Multiple 3D Plane Detection from Depth Images

Hyunkil Hwang[1], Jun Hyeok Choi[2], Chan Ho Seo[2], and Sunglok Choi[2]

*Abstract*— **Plane detection is a useful and necessary part of many robotic applications such as localization, SLAM, and world modeling. Its computing time is critical in the real-time operation of its applications. This paper proposes 3D plane detection using depth images acquired by stereo/RGB-D cameras or 3D LiDARs. The proposed method directly generates surface normal vectors from a depth image using simple pixel-wise subtractions and multiplications. The surface normal generation does not require any conversion to a point cloud and neighborhood search, which is the key idea to accelerate 3D plane detection in the proposed method. The surface normal vectors are substantial priors for 3D planes. The flood-fill algorithm on the surface normal map can simply lead to 3D plane segmentation quickly. Our implementation in Python experimentally has demonstrated almost 4-5 times less computing time than the previous RANSAC-based approach.**

## I. INTRODUCTION

Planes are useful observations in many robotic applications. Many visual/LiDAR odometry and SLAM utilized planes as high-level features or motion/structure priors. Especially, indoor environments contain many (mostly orthogonal) planes which can correct robot orientation more accurately [1]. The ground plane or floor is also a strong prior for robot motion and poses [2], [3]. Many indoor 3D reconstructions [4] also simplify their world models using planes instead of more general polygon meshes or point clouds. Planes with textures enable more compact but high-quality world models.

Plane detection retrieves planes from 2D data (e.g. RGB image) [5] or 3D data (e.g. depth image and point cloud) [1], [6]. This paper focuses on 3D plane detection from depth images. Even though this paper is based on depth images, depth images can be easily generated from point clouds or 3D range data with the given projection parameters. The 3D data often contain multiple planes. The proposed method investigates multiple 3D plane detection. In spite of multiple planes, they can be simply reduced to a single plane (e.g. ground plane) with proper conditions and constraints.

The computing time is one of the most important issues in plane detection because it is usually an early stage of applied algorithms and systems. Plane detection with long computing time will delay the overall algorithms and systems, which is critical in real-time applications such as localization and SLAM. RANSAC [7] is the most popular method for 3D plane fitting [1], [6]. RANSAC can select the most supportive *single* plane after sufficient iterations of plane generation (from random point samples) and its evaluation. The RANSAC-based approaches can be extended for finding *multiple* planes while keeping a generated plane if the plane is supported more than the given number of points. However, the RANSAC-based approaches usually suffer from long computing time due to their nature of the sequential and iterative framework. The structure prior [1] or simplified plane model [6] can speed up the RANSAC-based approaches.

This paper proposes fast 3D plane detection using depth images, not point clouds. In contrast to the point cloud, depth images are possible to access neighborhood points with simple 2-dimensional indexing. The surface normal vectors can be derived from the neighborhood points using simple pixel-wise subtractions and multiplications. Actually, the neighborhood points do not guarantee the closest point set, but normal vector calculation does not require the closest point set. Since the normal vectors are powerful information for getting 3D plane equations, 3D planes are simply classified using the flood-fill algorithm with simple pixel-wise distance measures. The proposed method is described in Section II in detail. Section III demonstrates the preliminary results of the proposed method in comparison to the RANSAC-based approach. Finally, Section IV summarizes our idea and results with further research direction.

## II. PLANE DETECTION FROM DEPTH IMAGES

### A. Overall Procedure

The proposed 3D plane detection is composed of two steps: 1) normal map generation and 2) flood-fill plane segmentation. As we mentioned before, input data is a depth image $D$ whose each pixel at $(u, v)$ contains its depth value, $D(u, v) = z$. Therefore, each pixel leads to its 3D point using the inverse camera projection as follows:

$$x = \frac{u - c_x}{f_x} z \;\text{ and }\; y = \frac{v - c_y}{f_y} z \;\text{ where }\; z = D(u, v)\,, \quad (1)$$

where $(f_x, f_y)$ are x- and y-directional focal lengths, and $(c_x, c_y)$ is the principal point on the depth image.

In the first step, the normal vector at each pixel is generated from its four neighborhood pixels. The neighborhood pixels of $(u, v)$ are selected using 2-dimensional array access such as $(u-s, v)$, $(u+s, v)$, $(u, v-s)$, and $(u, v+s)$. After the

*This research was supported by the National Research Foundation of Korea (NRF) Grant funded by the Ministry of Science and ICT for Bridge Convergence R&D Program (NRF-2021M3C1C3096810).

[1]Hyunkil Hwang is with the Department of Mechanical and Automotive Engineering, Seoul National University of Science and Technology (SEOULTECH), Seoul, Republic of Korea. (E-mail: hyunkil76@gmail.com)

[2]Jun Hyeok Choi, Chan Ho Seo, and Sunglok Choi are with the Department of Computer Science and Engineering, Seoul National University of Science and Technology (SEOULTECH), Seoul, Republic of Korea. (E-mail: sunglok@seoultech.ac.kr)

first step, each pixel contains its normal vector $\mathbf{n} = [a, b, c]^\top$ which is unit length ($a^2 + b^2 + c^2 = 1$).

In the second step, the flood-fill algorithm classifies each pixel into multiple 3D planes. A 3D plane is represented by $ax + by + cz + d = 0$ whose three variables $a$, $b$, and $c$ are derived from the normal vector. From a given seed pixel, the flood-fill algorithm spreads the coverage of the given pixel to its neighborhood pixels until they satisfy two metrics: 1) normal vector similarity $S_\mathbf{n} > \tau_\mathbf{n}$ and 2) depth value distance $D_z < \tau_z$. The two variables, $\tau_\mathbf{n}$ and $\tau_z$, are predefined threshold values for each metric. The normal vector similarity is defined using the cosine similarity of two unit vectors, $\mathbf{n}_i$ and $\mathbf{n}_j$, as follows:

$$S_\mathbf{n}(\mathbf{n}_i, \mathbf{n}_j) = \mathbf{n}_i \cdot \mathbf{n}_j \,. \tag{2}$$

The depth distance is defined using the difference of two depth values, $z_i$ and $z_j$, as follows:

$$D_z(z_i, z_j) = |z_i - z_j| \,. \tag{3}$$

The flood-fill algorithm groups connected pixels belonging to the same 3D plane based on two metrics.

### *B. Pixel-wise Normal Map Generation*

A surface normal vector of the given pixel is calculated using its neighborhood pixels on a depth image. It is much faster than normal vector calculation using point cloud because such calculation needs neighborhood search and principal component analysis (PCA).

At first, the proposed pixel-wise normal vector calculation needs a depth gradient as follows:

$$\nabla D(u, v) = \begin{bmatrix} \frac{\partial D}{\partial u} \\ \frac{\partial D}{\partial v} \end{bmatrix} = \begin{bmatrix} \frac{D(u+s,v) - D(u-s,v)}{2s} \\ \frac{D(u,v+s) - D(u,v-s)}{2s} \end{bmatrix} , \tag{4}$$

where $s$ is the step length to indicate neighborhood pixels. Figure 1 presents the effect of the step length $s$. It is possible to observe that a large value of $s$ led to smooth normal vectors without additional smoothing steps. However, the large value of $s$ missed the detail of the normal map according to the size of objects. For example, boundaries of bricks in Figure 1 were highlighted in $s = 1$ but degraded in $s = 10$ and $s = 30$. The pipe in Figure 1 was identified in $s = 1$ and $s = 10$ but partially disappeared in $s = 30$.

A normal vector is calculated using the cross-product of two gradient vectors. The cross-product lead to an orthogonal vector of two given vectors on a candidate plane, which is the normal vector to the plane. The common approach can simplify the cross-product as

$$\mathbf{n}(u, v) = \begin{bmatrix} 1 \\ 0 \\ \frac{\partial D}{\partial u} \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ \frac{\partial D}{\partial v} \end{bmatrix} = \begin{bmatrix} -\frac{\partial D}{\partial u} \\ -\frac{\partial D}{\partial v} \\ 1 \end{bmatrix} . \tag{5}$$

However, the common approach does not estimate the normal vector correctly as shown in Figure 2. Figure 2 (a) presents the gradually varying normal vectors on the same plane by the common approach. This problem results from different units of two vectors in Equation (5). Even though $\frac{\partial D}{\partial u}$ and $\frac{\partial D}{\partial v}$ has the unit of *meters per pixel*, but 1 has the unit of *pixels per pixel*, that is dimensionless.

We proposed a more accurate calculation of normal vectors with scaling using depth values. To compensate for different units, we apply the scale factor $\alpha$ to the last element as follows:

$$\bar{\mathbf{n}}(u, v) = \alpha \mathbf{k} * \mathbf{n} = \begin{bmatrix} -\frac{\partial D}{\partial x} & -\frac{\partial D}{\partial y} & \alpha \end{bmatrix}^\top , \tag{6}$$

where $\mathbf{k}$ is a unit vector $[0, 0, 1]^\top$ and $*$ is the element-wise multiplication. The scale factor $\alpha$ should be different from each pixel. We define the varying scale factor using normalized depth values $\bar{D}$ as follows:

$$\bar{\alpha}(u, v) = \frac{\bar{D}(u, v)}{\max \bar{D}(u', v')} \quad \text{where} \tag{7}$$

$$\bar{D}(u, v) = D(u, v) - \min D(u', v') \,, \tag{8}$$

$\min D(u', v')$ is the minimum value over the given depth image, and $\max \bar{D}(u', v')$ is the maximum value over the given normalized depth image. The depth-based scaling was inspired by camera projection, that is, farther 3D point has proportionally longer *meters per pixel*. The proposed method generated the normal map as shown in Figure 2 (b).

## III. EXPERIMENTS

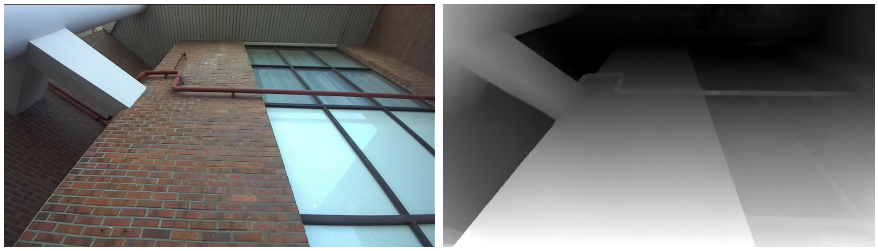
We performed experiments with real depth data using Stereolabs's ZED2 stereo camera. The ZED2 camera was configured as $1280 \times 720$ resolution for its left and right cameras, respectively. Its depth images were acquired by ZED SDK with *neural* depth mode. Figure 1 (b) is an example of depth images by our configuration.

We implemented the proposed algorithm in Python with NumPy and OpenCV. We assigned 32 seed points uniformly distributed on the given images. For comparison, we also implemented a common RANSAC-based plane detector with multiple plane fitting. The number of iterations of RANSAC was configured as 1000. The inlier threshold was assigned as 0.02 meters (for narrow indoors) and 0.2 meters (for wide outdoors), respectively. We measured the computing time of each method using Python Standard Library, `time`.

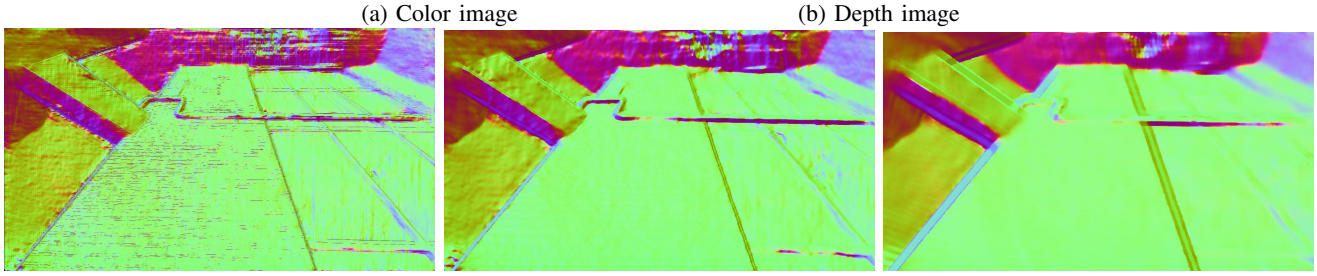
Table I shows the computing time of the previous RANSAC-based method and the proposed method, respectively. The proposed method was almost 4-5 times faster than the RANSAC-based method. The proposed method had different computing times according to the step length $s$ because longer $s$ made less normal vector calculation around image boundaries. As shown in Figure 3, we also observed that RANSAC-based plane fitting classified wrong points because it does not consider proximity (or connectivity) of 3D points.

## IV. CONCLUSION

This paper proposed fast 3D plane detection using depth images. Since it is possible to access neighborhood 3D points on depth images using 2-dimensional array indexing, the normal vector of each pixel is simply derived using pixel-wise subtractions and multiplication. The common normal

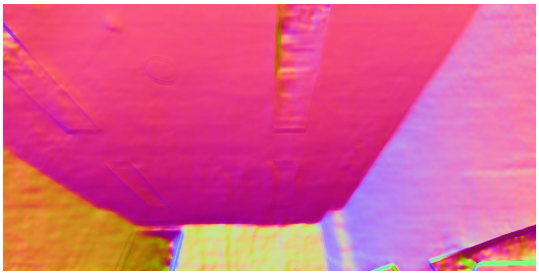


(a) Color image (b) Depth image

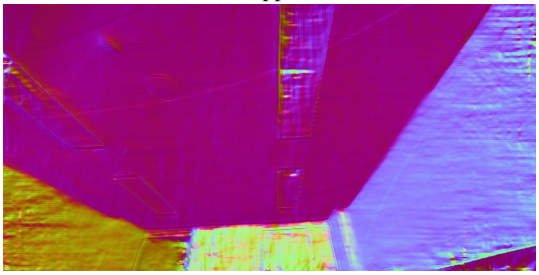


(c) Normal map ($s = 1$) (d) Normal map ($s = 10$) (e) Normal map ($s = 30$)

Fig. 1: An example of color and depth images with its normal maps with varying values of the step length $s$



(a) Common approach ($\alpha = 1$)



(b) Proposed approach ($\alpha = \bar{\alpha}(u, v)$)

Fig. 2: Normal vectors from (a) the common approach and (b) the proposed scaling approach

vector calculation estimates incorrect normal vectors due to inconsistent units of normal vectors. We adopted the varying scale factor $\bar{\alpha}$ to compensate the inconsistency. Our experimental results presented that the proposed method is almost 4-5 times faster than the previous RANSAC-based plane fitting.

TABLE I: COMPUTING TIME (UNIT: [MSEC])

| ALGORITHMS | COMP. TIME |
|---|---|
| RANSAC | 389 |
| Proposed ($s = 1$) | 102 |
| Proposed ($s = 10$) | 94 |
| Proposed ($s = 30$) | 78 |

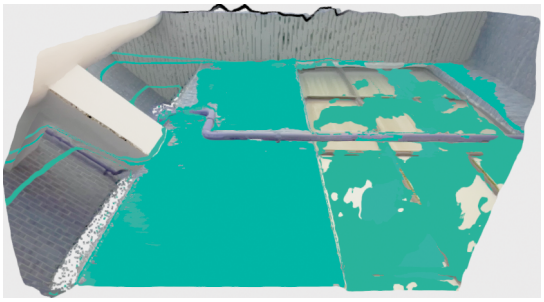


Fig. 3: An example of RANSAC-based 3D plane fitting

## REFERENCES


[1] K. Joo, P. Kim, M. Hebert, I. S. Kweon, and H. J. Kim, "Linear RGB-D SLAM for structured environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, 2022.

[2] S. Choi, J. Park, and W. Yu, "Simplified epipolar geometry for real-time monocular visual odometry on roads," *International Journal of Control, Automation and Systems*, vol. 13, no. 6, 2015.

[3] S. Choi and J.-H. Kim, "Fast and reliable minimal relative pose estimation under planar motion," *Image and Vision Computing*, vol. 69, 2018.

[4] S. Ikehata, H. Yang, and Y. Furukawa, "Structured indoor modeling," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[5] S. Choi, J. H. Joung, W. Yu, and J.-I. Cho, "What does ground tell us? monocular visual odometry under planar motion constraint," in *Proceedings of 2011 11th International Conference on Control, Automation and Systems (ICCAS)*, 2011.

[6] S. Choi, J. Park, J. Byun, and W. Yu, "Robust ground plane detection from 3d point clouds," in *Proceedings of 2014 14th International Conference on Control, Automation and Systems (ICCAS)*, 2014.

[7] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, 1981.