Fast Planar Segmentation of Depth Images

Hani Javan Hemmat, Arash Pourtaherian, Egor Bondarev, and Peter H. N. de With

Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands

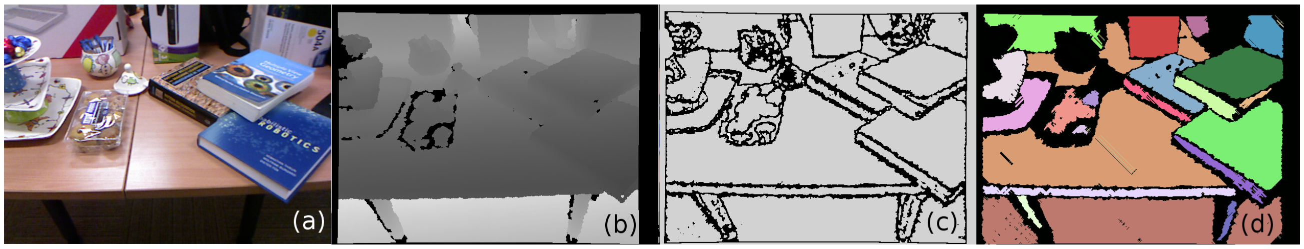

Figure 1. Planar-segmentation process from a sample depth image to its corresponding segmented-planes: (a) RGB image of a sample scene, (b) corresponding depth image, (c) extracted 3D edges, (d) outcome of the fast planar-segmentation algorithm applied on the sample scene.

ABSTRACT

One of the major challenges for applications dealing with the 3D concept is the real-time execution of the algorithms. Besides this, for the indoor environments, perceiving the geometry of surrounding structures plays a prominent role in terms of application performance. Since indoor structures mainly consist of planar surfaces, fast and accurate detection of such features has a crucial impact on quality and functionality of the 3D applications, e.g. decreasing model size (decimation), enhancing localization, mapping, and semantic reconstruction. The available planar-segmentation algorithms are mostly developed using surface normals and/or curvatures. Therefore, they are computationally expensive and challenging for real-time performance. In this paper, we introduce a fast planar-segmentation method for depth images avoiding surface normal calculations. Firstly, the proposed method searches for 3D edges in a depth image and finds the lines between identified edges. Secondly, it merges all the points on each pair of intersecting lines into a plane. Finally, various enhacements (e.g. filtering) are applied to improve the segmentation quality. The proposed algorithm is capable of handling VGA-resolution depth images at a 6 FPS frame-rate with a single-thread implementation. Furthermore, due to the multi-threaded design of the algorithm, we achieve a factor of 10 speedup by deploying a GPU implementation.

Keywords: Planar segmentation, 3D-edge detection, Line-based plane-detection, 3D Reconstruction.

1. INTRODUCTION

The advent of low-cost real-time depth sensors has attracted multiple studies in 3D perception and reconstruction for a wide range of applications including robotics, health-care, and surveillance. Reconstructing 3D indoor environments, however, remains challenging due to the cluttered spaces, extensive variability, and the real-time constraint. Furthermore, as an application point of view, perceiving the geometry of surrounding structures is very important for indoor environments. It is a valid assumption to consider, on the average, up to 95% of indoor-environment structures consist of planar surfaces.[1] Therefore, fast and accurate detection of such features has a crucial impact on quality and functionality of 3D applications, e.g. decreasing model size (decimation), enhancing localization, mapping, and semantic reconstruction.

Multiple planar-segmentation algorithms have been proposed for pointclouds. They mainly utilize region growing,[2,3] 3D Hough Transform,[4] RANdom SAmple Consensus (RANSAC) on the complete set of data[5] or on the pre-selected points using 3D feature descriptors,[6] and a combination of Hough and RANSAC for multiple resolutions.[7] However, these techniques typically introduce computationally expensive algorithms or they suffer

Further author information: (Send correspondence to H. Javan Hemmat) E-mail: h.javan.hemmat@tue.nl

from lack of robustness. Moreover, further processing, which increases the complexity of such methods even more, is required in order to obtain the relation of neighbourhood points.

In this paper, we propose a fast planar segmentation algorithm solely based on identification of intersecting lines on a plane without processing any surface normals. The proposed method is capable of handling VGA-resolution depth images at a high frame rate, upto 6 FPS, with a single-thread implementation. Moreover, the algorithm is optimally designed for multi-threaded GPU platforms. Therefore, the GPU implementation of the algorithm is at least 10 times faster and also capable of dealing with higher resolution images. Figure 1 illustrates the intermediate and final outcomes of the planar-segmentation algorithm applied on a depth image.

This paper is structured as follows. This section is consolidated by a brief review of the related work. Section 2 describes the methods exploited in each part of the proposed algorithm in details. The obtained results are being discussed in Section 3. Finally, Section 4 concludes the paper.

1.1 Related work

Two of the popular plane-segmentation techniques of depth images (and their corresponding pointclouds) are the RANSAC and 3D Hough-transform. RANSAC is utilized to detect the planes with the most number of inliers.[8, 9] The RANSAC is applied iteratively on a depth image to detect multiple planes[10] or region-wise to the connected regions.[11] However, points belonging to the same segment do not always connect and therefore extensive post-processing is necessary to integrate the over-segmented regions. In 3D Hough-transform method, each plane is typically described in the object space with a corresponding point in the parameter space. Hough results are combined with a voting mechanism of detections across the sequence of incoming frames to increase the detection robustness.[12] However, similarly to RANSAC, post-processing steps are required for Hough-based methods to merge neighbouring segments where their parameters do not considerably deviate. Other popular segmentation-techniques based on surface normals and/or curvatures extraction,[13] linear fitting and Markov chain Monte Carlo[14] are computationally expensive and challenging for real-time performance. In these algorithms, a real-time segmentation of planes is normally achieved by sacrificing the image quality.[15] Another segmentation technique is the region growing method which is an efficient technique to extract planes from depth images. However, the choice of segmentation primitives and growth criterion is critical for the complexity of the algorithm. In addition, evaluating the surface normals and curvature estimates[13] also require large computational power. Introducing high-level features (line or curve blocks), as segmentation primitives instead of individual pixels, significantly reduces the amount of processed data.[16]

We propose a fast high-level region growing, assuming that two intersecting straight lines lie on a plane. To this end, edge contours should be extracted first to find connected regions of local surface continuity.[17, 18] Several algorithms for edge detection in intensity images have been developed and extensively discussed in the literature. However, they have a poor performance when applied on the depth maps. The problem of edge detection in intensity images is significantly different when using depth maps. First, definition of an edge in intensity images consists of significant changes in gray level values modelled as the jump edges, whereas in depth maps, corner and curved edges should be also extracted.[19] Second, in depth maps, spatial distribution of range data can be irregular which requires operator adaptivity with respect to shape and size of objects[20] in the scene. Finally, traditional edge-detection operators for intensity images such as Prewitt, Sobel and Canny are limited by orientation and scale and therefore perform poorly in highly noisy and blurred edges of depth images.[21] An edge-detection specific for the depth images has been developed in an early study by detecting residues of morphological operations.[22] In a later work, straight lines in 3D space are detected as depth edges by using a (2+2)–D Hough space.[23] Wani and Batchelor[24] studied spatial curves to design edge masks for edge detection. This methodology was further investigated based on the scan line approximation technique.[25] We have been inspired by the scan line segmentation technique[25] and propose an edge detection algorithm to apply on noisy depth-images.

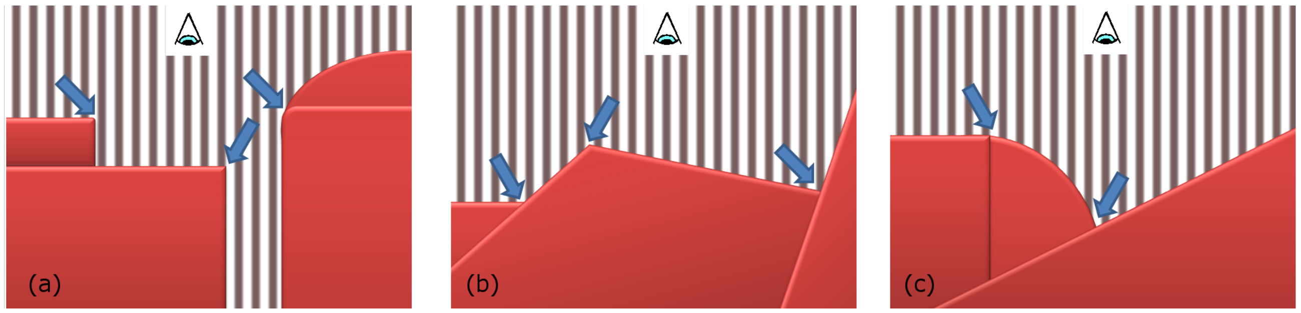

Figure 2. Various types of 3D edges in a depth image. In all the images, vertical lines indicate distance between surfaces and the Kinect camera plane. (a) shows *jump-edges* where there is a gap between two surfaces and/or in case of existence of holes; (b) depicts *corner-edges* where two planar surfaces meet each other; and (c) illustrates the situation that planar surfaces join non-planar ones and *corner-edges* emerge.

2. METHOD

This section introduces the proposed algorithm in details. The planar segmentation of depth images is performed by utilizing two intrinsic properties of planes in a 3D environment:

Property 1. *Inside the framework of a depth image, each planar surface is bounded between 3D edges.*

Property 2. *Based on geometrical propositions, if two straight lines cut one another, then they lie in one plane.*

As indicated by the Property 1, we must search between 3D edges in order to find a planar surface in a depth image. According to the Property 2, each pair of intersecting lines represent a planar surface. Therefore extracting the straight lines in a depth image can lead us to finding planar segments in the image.

The proposed algorithm segments a given depth images into its planar surfaces, based on the mentioned properties, in three main steps. In the first step, it extracts all 3D edges in the depth image. Then it searches for the lines lying between two opposite edges as the second step. And in the final step of the algorithm, it tries to merge the identified lines as far as possible and categorizes them into different groups. These groups of cutting lines are the planar-surface candidates. An extra step of validation is performed whether to verify a group as a planar surface or discard it. The followings focus on methods exploited in each step in detail.

2.1 Edge detection

In order to detect 3D edges in a depth image, the algorithm scans the image in 1-pixel strips in four directions (vertically, horizontally, left- and right-diagonally). An edge is detected according to the depth-value changes. At each scan, three different types of edges are detected: *jump*, *corner*, and *curved* edges. The jump-edges appear due to occlusions or holes in depth images. The corner-edges emerge where two actual planes meet each other, and the curved-edges are resulted from intersection of actual planar and non-planar surfaces. Figure 2 depicts the mentioned types of 3D edges in a depth image.

Based on the following definitions, we formulate the detection criterion for each edge type.

Definition 2.1. $P_i\left(x, y, depth\right)$ *= the* i^{th} *3D point on current line of the scan direction in a given depth image. Actually, it is the pixel located on column* x *and row* y *of the depth image as illustrated in Figure 3. In the rest of this paper, we use the* P_i *instead of* $P_i\left(x, y, depth\right)$ *for a brief representation.*

Definition 2.2. $\mathrm{Threshold_f}$ *= a user-defined value indicating a threshold for the feature* f *(e.g.* $Threshold_{slope}$ *is the criteria to compare any line's slope with).*

Definition 2.3. $\Phi\left(i, \mathrm{s}\right)$ *= a set of* n *neighbours on side* $\mathrm{s} \in \{before, after\}$ *of point* P_i *on the current 1-pixel-wide strip of scan in any of the mentioned directions as depicted in Figure 3.* n *is a user-defined value. We occasionally use* Φ *to briefly represent* $\Phi\left(i, \mathrm{s}\right)$ *in this paper.*

Definition 2.4. $Slope\left(\mathrm{a}, \mathrm{b}\right)$ *= the slope of the line passing through points* a *and* b *in the 3D space.*

Definition 2.5. $\overline{Slope}\left(\Phi\right)$ *= the average slope of the lines passing through each two consecutive points in* Φ.

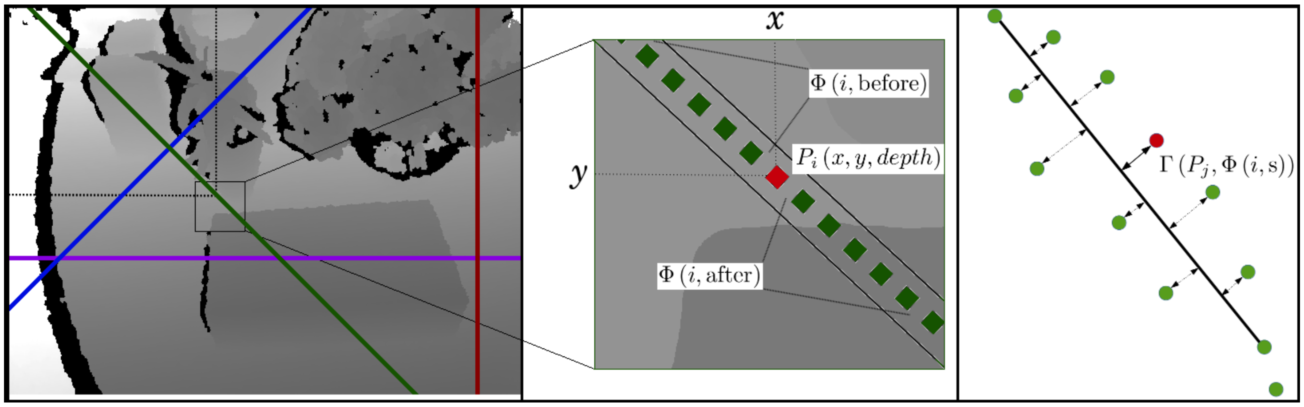

Figure 3. Left: 1-pixel-wide scan strips in four directions (vertically, horizontally, left- and right-diagonally). Middle: a magnified sample-area to illustrate point $P_i(x, y, depth)$ and its neighbours $\Phi(i, \text{before})$ and $\Phi(i, \text{after})$ located before and after it in a vicinity of 5 points, respectively. Right: illustration of the signed distance of $P_i(x, y, depth)$ to the line connecting points at the two ends of its neighbourhoods.

Definition 2.6. $\Gamma(P_j, \Phi(i, \text{s}))$ *= the signed distance of* P_j *to the line passing through* P_{i-n} *and* P_{i+n} *in the 3D space (as shown in Figure 3).*

Jump edge. The 3D points P_i and P_j are located on the opposite sides of a *jump-edge* if they meet the condition represented by Equation (1).

$$|P_i.depth - P_j.depth| > \text{Threshold}_{\text{jump}} \,. \tag{1}$$

$P_i.depth$ is the depth value of pixel P_i. This equation marks two points as the points on a jump edge if their difference in terms of distance to the Kinect plane is more than a user-defined threshold (as illustrated in Figure 2-a).

Corner edge. The *corner-edge* is defined according to the following definition.

Definition 2.7. $Flag_{equal-slope}(\Phi)$ *checks whether all the lines passing through consecutive pairs of points in the neighbourhood* Φ *have the same slope or not.*

Equation (2) formulate this definition according to a user-defined value of $Threshold_{slope}$. The $True$ value for the flag means that points of Φ are located on a straight line (and on a planar surface, accordingly).

$$Flag_{equal-slope}(\Phi) = \begin{cases} \text{TRUE} & \forall\,(P_i, P_j)\; and\; (P_{i'}, P_{j'}) \quad |\; P_i, P_j, P_{i'}, P_{j'} \in \Phi \\ & |Slope(P_i, P_j) - Slope(P_i', P_j')| < \text{Threshold}_{\text{slope}} \,. \\ \text{FALSE} & \text{otherwise.} \end{cases} \tag{2}$$

P_i emerges as a point on a *corner-edge* if both conditions (3) and (4) are met:

$$Flag_{equal-slope}(\Phi(i, \text{before})) \quad \text{AND} \quad Flag_{equal-slope}(\Phi(i, \text{after})) = \text{TRUE} \,, \tag{3}$$

$$\left|\overline{Slope}(\Phi(i, \text{before})) - \overline{Slope}(\Phi(i, \text{after}))\right| > \text{Threshold}_{\text{slope}} \,. \tag{4}$$

Equation (3) checks if two planar surfaces exist on both sides of point P_i and Equation (4) ensures that both surfaces have different slopes (not located on the same plane).

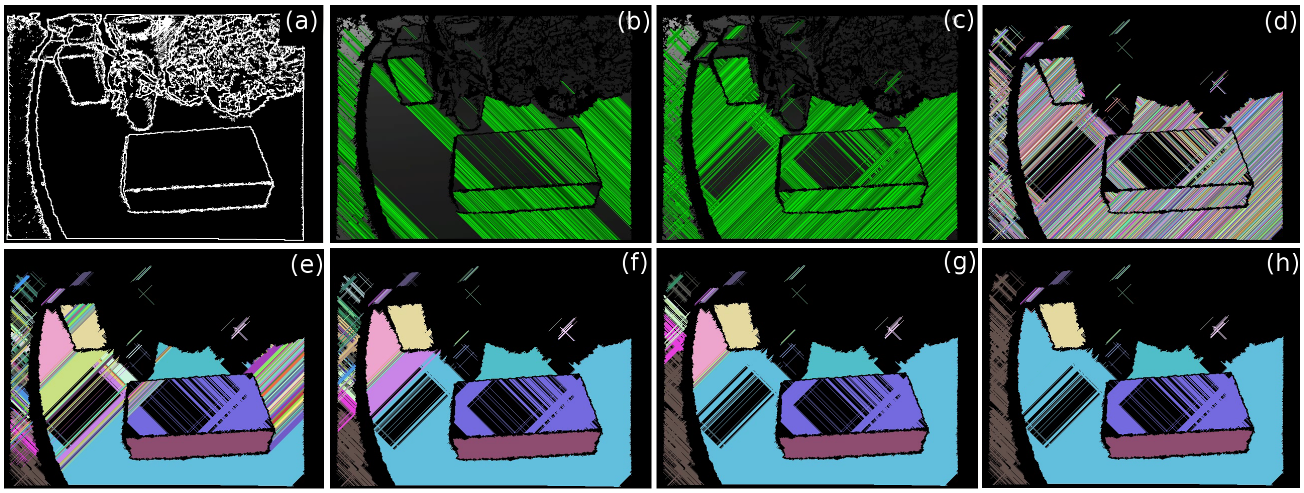

Figure 4. Planar-segmentation process of a sample depth-image from 3D edges to planar surfaces: (a) extracted edges on a given depth-image; (b-c) left- and right-diagonally detected lines between the edges (the vertical and horizontal lines are not depicted for the sake of clarity); (d) segmentation process which begins with labelling each line individually; (e-g) merging process of intersecting lines which share the same label; (h) final outcome of the planar-segmentation process (Note that involving the vertical and horizontal scan-strips can improve the result. Besides this, the proposed enhancement methods can excel the final outcome).

Curved edge. The *curved-edge* is defined based on the following definition.

DEFINITION 2.8. *$Flag_{straight}(\Phi)$ indicates whether all the points of neighbourhood Φ are located on a straight line or not.*

This definition is formulated as Equation (5) based on a user-defined value of $Threshold_{straight}$. Similar to the Equation (2), the $True$ value for the $Flag_{straight}(\Phi)$ means that points of Φ are located on a straight line (and on a planar surface, accordingly). The difference is that Equation (2) is more sensitive on boundaries between two planar surfaces, while Equation (5) performs better where a plane meets a non-planar surface.

$$Flag_{straight}(\Phi) = \begin{cases} \text{TRUE} & \forall\,(P_i, P_j) \quad \mid P_i, P_j \in \Phi \\ & |\Gamma(P_i, \Phi) - \Gamma(P_j, \Phi)| \leq \text{Threshold}_{\text{straight}}. \\ \text{FALSE} & \text{otherwise.} \end{cases} \quad . \tag{5}$$

P_i is on a *curved-edge* if condition (6) is met.

$$Flag_{straight}\left(\Phi\left(i, \text{before}\right)\right) \quad \text{XOR} \quad Flag_{straight}\left(\Phi\left(i, \text{after}\right)\right) \;=\; \text{TRUE}\,. \tag{6}$$

The XOR operation ensures that *one and only one* side is a planar-surface.

2.2 Line-based plane detection

Extracting the 3D edges of a depth image in the first step enables us to perform the second step in which the algorithm searches for straight lines between the edges. Therefore, the algorithm scans the depth image in any of the four directions (vertically, horizontally, left- and right-diagonally) to find lines between points located on the opposite edges. After finding all the lines, the algorithm tries to merge the points on each pair of intersecting detected-lines into a plane according to what described as Property 2. It continues merging points until it covers all the points on each line. Figure 4 illustrates the merging process. Each group of the merged points is a candidate for a planar surface. This step of the algorithm segments a depth image into its planes, but the resulting planar segments need improvements as discussed in the following section.

2.3 Plane Enhancement

After performing the proposed segmentation process, there is still some room for enhancement in order to improve the segmentation outcome. First of all, it is required to evaluate the resulting segments in terms of their curvature in a 3D space (for instance as a pointcloud). Besides this, due to occlusion it could happen that a planar surface is detected as various disconnected-segments. Therefore, a merging process is needed to be perform to coalesce this apart segments into a unified plane. Finally, after accomplishing these two enhancements, we need to evaluate the resulting segments in terms of their size. Based on the various criteria, a user may prefer to discard the planar segments, which has a relatively small number of points. Hence, we process the detected planes further in order to reject diminutive planes.

The three mentioned stages of the enhancement is performed based on the following definitions:

DEFINITION 2.9. *The Eigenvalues for plane Ψ in a 3D space (e.g. as a ponitcloud) are defined by λ_0, λ_1, and λ_2 where $\lambda_2 < \lambda_1 < \lambda_0$.*

DEFINITION 2.10. T_k *is a user-defined threshold for the curvature ratio of $\frac{\lambda_2}{\lambda_k}$, where $k \in \{0, 1\}$.*

The values T_0 and T_1 for a plane represent the ratio of the plane's height to its length and width, respectively. A planar surface must have a very tiny value in terms of these aspects. In the other essence, the smaller values of T_0 and T_1 for a plane mean less curvature and a flatter surface accordingly.

DEFINITION 2.11. $\overrightarrow{\mathbf{n}}(\Psi)$ *is the normal vector of plane Ψ which equals the smallest eigenvector, $\overrightarrow{\nu_2}$, of the plane.*

DEFINITION 2.12. $\overrightarrow{\gamma}(\Psi_a, \Psi_b)$ *is the vector connecting the center of masses of the plane Ψ_a to its corresponding point of plane Ψ_b.*

Curvature validation. To evaluate their curvature values, the plane candidates are converted to corresponding pointclouds and their eigenvalues are calculated by means of the Principal Component Analysis (PCA) method. A plane is considered valid if its curvature values in both $\overrightarrow{\nu_0}$ (height-to-length) and $\overrightarrow{\nu_1}$ (height-to-width) directions are less than thresholds T_0 and T_1, respectively. The equation (7) formulate the curvature-validation process as a condition, which is needed to be fulfilled by each valid plane.

$$(\lambda_2 \leq \lambda_0 \times \mathrm{T}_0) \quad \text{AND} \quad (\lambda_2 \leq \lambda_1 \times \mathrm{T}_1) \quad = \quad \text{TRUE} \, . \tag{7}$$

Merging separate-segments. Due to holes and occlusions in a depth image, different segments of a detected plane may be identified as separate disconnected-planes. In order to merge the separate segments, we check the conditions (8) and (9) for each pair of plane segments Ψ_a and Ψ_b. If both conditions are met then the two separated segments are merged into a single plane. *True* value for the condition represented by Equation (8) means that the separated planes Ψ_a and Ψ_b are parallel to each other; and the condition described by Equation (9) is satisfied when both the separated planes Ψ_a and Ψ_b lie on the same planar surface.

$$\overrightarrow{\mathbf{n}}(\Psi_a) \times \overrightarrow{\mathbf{n}}(\Psi_b) \approx \mathbf{0} \, , \tag{8}$$

$$\overrightarrow{\mathbf{n}}(\Psi_a) \cdot \overrightarrow{\gamma}(\Psi_a, \Psi_b) \approx 0 \quad \text{AND} \quad \overrightarrow{\mathbf{n}}(\Psi_b) \cdot \overrightarrow{\gamma}(\Psi_a, \Psi_b) \approx 0 \, . \tag{9}$$

Size validation. A valid plane is required to contain a certain minimum number of points. This improves segmentation quality by removing the small segments that could not be merged to any other larger planes. Each valid plane Ψ should satisfy the condition described by Equation 10.

$$\Psi.size \geq \text{Threshold}_{\text{size}} \, . \tag{10}$$

Beyond all these post-processing enhancements, a pre-processing step is also performed to improve segmentation results even more. After extracting all the 3D edges of a depth image, we also apply a morphological filter of *closing*closing (straightforwardly of a *dilation* followed by an *erosion*) in order to gain more smooth and connected edges.

Table 1. Average execution time (s) per frame for the proposed method applied on depth images compared to the PCL-SAC various methods applied on corresponding pointclouds.

Methods	Real datasets					Artificial datasets		Mean	$\frac{\text{std. dev.}}{\text{mean}}$
	board	desk	lab.	room	table	boxes	room		
Presented	0.312	0.225	0.249	0.256	0.219	0.197	0.176	0.23	0.19
RANSAC	15.1	21.7	27.4	23.1	11.7	17.7	1.0	16.8	0.52
LMEDS	320.7	449.1	545.7	569.4	257.9	828.9	240.3	458.8	0.46
MSAC	18.4	25.9	32.4	27.7	13.884	21.2	1.2	20.1	0.51

3. RESULTS AND DISCUSSION

We applied the proposed segmentation algorithm on several datasets including both real and artificial VGA-resolution depth-images of indoor scenes. Our algorithm execution time has been compared to various methods from the PCL library[*]. Among all the evaluated methods, RRANSAC, RMSAC and MLESAC had a very long execution time, and PROSAC resulted in invalid segments. Therefore, these four methods were excluded from the evaluation results. Table 1 shows the execution times of our method compared to RANSAC, LMEDS and MSAC [†]. As presented, the proposed algorithm features shorter execution time for all datasets.

An interesting finding is that a large amount of *standard deviation to mean ratio* emerges for all various methods which deal with pointclouds compared to the proposed algorithm performing on depth images. This means that those methods are sensitive to the type and arrangement of the objects in the scene. We explain this according to the nature of their exploited 3D-feature detector, which strictly relies on 3D distribution of points. Depending on each scene, 3D distribution of points could vary hugely. On the other hand, since the presented algorithm is dealing with depth images as a 2D framework, it is less dependent on the distribution of 3D points in the space. Therefore, our algorithm provides a faster solution in terms of execution performance. Another finding is that all algorithms perform faster on noiseless artificial-datasets. Besides this, an interesting point is the huge difference between the execution times for PCL-based methods applied on real and artificial datasets, which reveals higher sensitivity of those methods to noise compared to the presented algorithm.

4. CONCLUSION

In this paper, we have introduced a fast planar-segmentation algorithm for depth images avoiding any normal-estimation calculation. The proposed algorithm searches for 3D edges in a depth image and finds the lines between these edges. Then, it merges all the points on each pair of the intersecting lines into a *plane*. Finally, various enhancement stages are applied to improve segmentation quality. The single-thread implementation of the introduced algorithm outperforms the similar methods dealing with pointclouds instead of depth images in terms of execution time. Furthermore, due to the multi-threaded design of the proposed algorithm, we expect to achieve a factor of 10 speedup by deploying a GPU version.

ACKNOWLEDGMENTS

This research has been performed within the PANORAMA project, co-funded by grants from Belgium, Italy, France, the Netherlands, the United Kingdom, and the ENIAC Joint Undertaking.

REFERENCES

[1] Rusu, R. B., *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*, PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany (2009).

[*] RANSAC: RANdom SAmple Consensus; LMEDS: Least MEDian of Squares; MSAC: M-Estimator SAmple Consensus; RRANSAC: Randomized RANSAC; RMSAC: Randomized MSAC; MLESAC: Maximum Likelihood Estimation SAC; PROSAC: PROgressive SAmple Consensus.

[†] The results relate to a single-thread implementation performed on a target PC with a CPU of Intel® Xeon(R) W3550 @3.07GHz and 20 GB of RAM.

[2] Rabbani, T., van den Heuvel, F. A., and Vosselman, G., "Segmentation of point clouds using smoothness constraint," [*Proc. ISPRS Comssission V Symp. Image Engineering and Vision Metrology*], 248–253 (2006).

[3] Xiao, J., Zhang, J., Adler, B., Zhang, H., and Zhang, J., "3D point cloud plane segmentation in both structured and unstructured environments.," *Robotics and Autonomous Systems* **61**(12), 1641–1652 (2013).

[4] Borrmann, D., Elseberg, J., Lingemann, K., and Nüchter, A., "The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design," *3D Research* **2**(2), 1–13 (2011).

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

[6] Rusu, R. B., Blodow, N., and Beetz, M., "Fast Point Feature Histograms (fpfh) for 3D Registration," in [*Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*], 3212–3217 (2009).

[7] Oehler, B., Stueckler, J., Welle, J., Schulz, D., and Behnke, S., "Efficient multi-resolution plane segmentation of 3d point clouds," in [*Proc. of Int. Conf. on Intelligent Robotics and Applications (ICIRA)*], Jeschke, S., Liu, H., and Schilberg, D., eds., *Lecture Notes in Computer Science* **7102**, 145–156, Springer (2011).

[8] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R., "Indoor Segmentation and Support Inference from RGBD Images," in [*Proc. European Conf. Computer Vision (ECCV)*], Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C., eds., *Lecture Notes in Computer Science* **7576**, 746–760, Springer (2012).

[9] Lee, T.-k., Lim, S., Lee, S., An, S., Oh, S.-y., and Member, S., "Indoor Mapping Using Planes Extracted from Noisy RGB-D Sensors," [*Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*], 1727–1733 (2012).

[10] Lee, K.-M., Meer, P., and Park, R.-H., "Robust adaptive segmentation of range images," *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **20**(2), 200–5 (1998).

[11] Silva, L., Bellon, O., and Gotardo, P., "A global-to-local approach for robust range image segmentation," in [*Proc. IEEE Int. Conf. Image Processing (ICIP)*], 773–776 (2002).

[12] Hulik, R., Spanel, M., Materna, Z., and Smrz, P., "Continuous plane detection in point-cloud data based on 3d hough transform," *Journal of Visual Communication and Image Representation* **25**(1), 86–97 (2013).

[13] Holz, D. and Behnke, S., "Fast Range Image Segmentation and Smoothing Using Approximate Surface Reconstruction and Region Growing," in [*Proc. Int. Conf. Intelligent Autonomous Systems*], 61–73 (2012).

[14] Erdogan, C., Paluri, M., and Dellaert, F., "Planar Segmentation of RGBD Images Using Fast Linear Fitting and Markov Chain Monte Carlo.," in [*Proc. of Conf. on Computer and Robot Vision (CRV)*], 32–39, IEEE (2012).

[15] Holz, D., Holzer, S., Rusu, R. B., and Behnke, S., "Real-Time Plane Segmentation Using RGB-D Cameras," in [*Proc. of RoboCup 2011: Robot Soccer World Cup XV*], **7416**, 306–317, Springer Berlin Heidelberg (2012).

[16] Jiang, X., Bunke, H., and Meier, U., "High-level feature based range image segmentation," *Image and Vision Computing* **18**(10), 817–22 (2000).

[17] Harati, A., Gächter, S., and Siegwart, R. Y., "Fast range image segmentation for indoor 3D-SLAM," in [*Proc. IFAC Symp. on Intelligent Autonomous Vehicles*], 475–480 (2007).

[18] Hulik, R., Beran, V., Spanel, M., Krsek, P., and Smrz, P., "Fast and accurate plane segmentation in depth maps for indoor scenes," *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* , 1665–70 (2012).

[19] Coleman, S. A., Suganthan, S., and Scotney, B. W., "Gradient operators for feature extraction and characterisation in range images," *Pattern Recognition Letters* **31**(9), 1028–40 (2010).

[20] Suganthan, S., Coleman, S. A., and Scotney, B. W., "Using Dihedral Angles for Edge Extraction in Range Data," *Journal of Mathematical Imaging and Vision* **38**(2), 108–18 (2010).

[21] Hoover, A., Jean-Baptiste, G., Jiang, X., Flynn, P. J., Bunke, H., Goldgof, D. B., Bowyer, K., Eggert, D. W., Fitzgibbon, A., and Fisher, R. B., "An experimental comparison of range image segmentation algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)* **18**(7), 673–89 (1996).

[22] Krishnapuram, R. and Gupta, S., "Edge detection in range images through morphological residue analysis," in [*Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*], 630–2 (1992).

[23] Bhattacharya, P., Liu, H., Rosenfeld, A., and Thompson, S., "Hough-transform detection of lines in 3-D space," *Pattern Recognition Letters* **21**(9), 843–9 (2000).

[24] Wani, M. A. and Batchelor, B. G., "Edge-region-based segmentation of range images," *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)* **16**(3), 314–319 (1994).

[25] Jiang, X. and Bunke, H., "Edge Detection in Range Images Based on Scan Line Approximation," *Computer Vision and Image Understanding* **73**(2), 183–99 (1999).