# Fast and Accurate Plane Segmentation in Depth Maps for Indoor Scenes

Rostislav Hulik, Vitezslav Beran, Michal Spanel, Premysl Krsek, Pavel Smrz
Brno University of Technology, Faculty of Information Technology
IT4Innovations Centre of Excellence
Bozetechova 2, 61266 Brno, Czech Republic
{ihulik, beranv, spanel, krsek, smrz}@fit.vutbr.cz

*Abstract* — **This paper deals with a scene pre-processing task – depth image segmentation. Efficiency and accuracy of several methods for depth map segmentation are explored. To meet real-time capable constraints, state-of-the-art techniques needed to be modified. Along with these modifications, new segmentation approaches are presented which aim at optimizing performance characteristics. They benefit from an assumption of human-made indoor environments by focusing on detection of planar regions. All methods were evaluated on datasets with manually annotated real environments. A comparison with alternative solutions is also presented.**

***Kinect; depth map segmentation; plane detection; computer vision; range sensing***

## I. INTRODUCTION

Image segmentation is a well-studied computer vision task. Although depth map segmentation has common roots with greyscale image segmentation, respective algorithms generally differ. It is due to the presence of an additional dimension – the depth. The depth information (and consecutive normal estimation from depth images) is beneficial as regions can be distinguished by a step in the depth or the direction of normal vectors. Consequently, segmentation algorithms are based primarily on depth- and normal comparison methods.

Existing approaches differ in their accuracy and speed. To guarantee real-time performance, applications in robotics search for a precise segmentation that makes use of simplest possible methods. The simplification can come from an additional constraint on the environment in which a robot operates. For example, human-made objects (artefacts) can be mainly expected in indoor scenes. The a priori knowledge of their common shapes characterized by planar regions can lead to special segmentation approaches – plane detection (prediction).

The research reported in this paper focuses on plane detection in indoor-scene depth maps. The task is taken as a pre-processing step for further planar object detection (floor, walls, table-tops, etc.) or rough segmentation of foreground and background objects. Although widely-used devices for capturing scene depth data (such as Kinect, PrimeSense, or XtionPRO) provide also visual RGB data, we focus solely on the depth data in this paper. The methods are then easily adaptable to data from other sensors such as LIDARs.

Today, there is a large number of depth segmentation algorithms usable in robotics. However, only few of them meet strict low computational power consumption constraints. We studied and compared existing methods and designed and implemented various modifications to reach real-time capable performance while retaining the accuracy.

Common low-cost depth sensors suffer from specific problems linked to such type of sensors. The major persisting problem is a structural noise present in the depth data. All reported methods are therefore optimized to depress this kind of problem. A general intention is to wider applicability of cheap range sensors in the field of precise and fast environment perception.

**Figure 1.** Indoor kinect depth map example

The rest of this paper is organized as follows: The next section discusses previous work related to our research. Section III takes on existing methods for depth map segmentation and proposes their optimization with the aim to achieve fast performance while keeping their stability and precision. Section IV presents novel approaches for segmentation of depth maps focused on planar regions typical for indoor scenes. Experimental results and a comparison of existing and new methods are discussed in Section V.

## II. Related Work

Last years brought a boom of cheap depth sensors used for localization, map building, environment reconstruction, object detection, and other tasks in robotics. A rapid development of various methods for depth map processing followed. Fast object detection usually incorporates preprocessing of depth data.

Pulli and Pietikäinen [1] apply normal decomposition in their approach. They explore various techniques of range data normal estimation (comparing their performance and accuracy on clean as well as noisy datasets). The techniques include quadratic surface least squares [2] or LSQ planar fitting [3]. A least-trimmed-squares method is utilized for comparison. The normal estimation is done by detecting roof and step edges. A similar approach is discussed in this paper as well.

Another approach that uses a simple extraction of step and roof edges in the depth map was introduced by Baccar et al. [6]. Various approaches to fusion are presented such as an averaging method, Dempster-Shafer combinations or the Bernouilli's rule. After combination rules are applied, an edge gradient map is created which is further used as an input to the watershed algorithm. This algorithm is applied to cope with the noise in depth maps.

Ying Yang and Förstner [7] present a plane detection method that makes use of the RANSAC algorithm. The map is split to tiles (small rectangular blocks). Three points are iteratively tested for a plane region in each block. Detected planes within a certain range are merged at the end. We present an adaptation of this technique for indoor depth map segmentation in this paper.

Other work that compares plane segmentation approaches and that generally inspired our research is presented in [8]. Among other findings, authors mention their experience showing that RANSAC tends to over-simplify complex planar structures, for example multiple small steps were often merged into one sloped plane.

Borrmann et al. [9] present an alternative approach to plane detection in point clouds based on 3D Hough transform. Dube and Zell [10] also employ randomized Hough transform for real-time plane extraction. Non-associative Markov networks are applied for the same task in [11]. A use of another method – multidimensional particle swarm optimization – is reported in [12].

Zheng and Zhang [13] extend the range of detected regular surfaces from planes to spheres and cylinders. Elseberg et al. [14] show how an octree- and RANSAC based method can efficiently deal with large 3D point clouds containing billions of special data points. Sithole and Mapurisa [15] speed up the processing by means of profiling techniques. Deschaud and Goulette [16] deal with efficiency issues as well.

Although our implementation is independent, we appreciate availability of the Point Cloud Library (pointclouds.org) developed by Willow Garage experts [17].

## III. Fast depth map segmentation

As mentioned above, we analysed several approaches to depth image segmentation focusing on efficient strategies enabling fast pre-processing that is potentially integrable into further environment perception tasks.

A first set of explored algorithms comprises modifications of existing segmentation methods. To meet requirements limiting computing time and power in typical robotic scenarios, we simplified the work of Baccar et al. [6]. This resulted in algorithms combining depth and normal information with morphological watershed segmentation.

Baccar et al. distinguish two approaches to depth image edge extraction – one is based on step edges and the other one on roof edges (depth- and normal edge extraction in our terminology). We tested these approaches and evaluated their performance and usability in environment perception tasks.

### A. Depth based edge extraction

The detection of step edges presents the fastest segmentation method as it is simple to compute on depth images. The original work implements the step edge detector using local image approximation by smooth second order polynomials and subsequent computation of first- or second-order derivatives. The authors state that this approach is fast. We experimented with it and decided to go even further and use only ordinal arithmetic to speed up the process.

Because of large structural noise present in Kinect depth images, it is still necessary to emulate the smoothing capability of second-order polynomials. This was done by taking into account the neighbourhood and computing the value of extracted edge according to the following formula:

$$f_D(\mathbf{x}) = \sum_{\mathbf{r}\in W(\mathbf{x})} \begin{cases} 1 & if\ |d(\mathbf{x}) - d(\mathbf{r})| > t \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $t$ is the threshold depth difference specifying a step edge, $W$ is the window of neighbouring pixels and $d$ is a depth information at pixel $\mathbf{x}$. This approach was chosen for its maximum speed and simplicity. A gradient map, with pixels representing steepness of edges, is generated as an output.

### B. Normal based edge extraction

Normal based edge detection, called extraction of roof edges in the original paper, was taken as a part of hybrid segmentation. In order to meet the computational speed requirements, we simplified this method and implemented it in a standalone module.

The edge extraction method is slightly more computationally expensive than the depth based segmentation, because the normal computation on noisy image must forerun. Instead of the normal estimation from second order poly-

nomials, we applied the principle of accumulator edge extraction again. Normal vectors are computed directly (using depth difference or least-squares fitting) and only normal differences are taken into account. Value of $i^{th}$ pixel of edge gradient image is computed as:

$$f_N(\mathbf{x}) = \sum_{\mathbf{r}\in W(\mathbf{x})} \begin{cases} 1 & if\ n(\mathbf{x}) \cdot n(\mathbf{r}) > t \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where *n* is a normal vector (normalized) adjacent to specified pixel of a depth image. This approach is simple but it has proven reliable in the context of noisy data from Kinect.

### C. Fusion based edge extraction

Having the outputs from depth-based and normal-based segmentation methods, a late fusion can be applied to produce accurate and stable results.

Several fusion schemes were presented in [6]. They ranged from simple averaging to sophisticated methods such as the Super Bayesian Combination for fusing two pieces of evidence or the Dempster's rule of combination.

We opted for the fastest combination technique again. Both the depth-based and normal-based edge extractors take advantage of binary accumulators. Since our segmentation algorithm is based on the watershed method (see below), a simple sum of the two outputs provides a robust combination. A necessary condition is to use the same kernel size in the input methods. The output is then given by a linear combination:

$$f(\mathbf{x})_C = w_N . f(\mathbf{x})_N + w_D . f(\mathbf{x})_D, \quad (3)$$

where $w_N$ and $w_D$ are appropriate weights. They are set to 1 to maintain ordinal computation in our experiments.

Table 1 summarizes all presented modifications.

### D. Watershed segmentation

The linear combination used in the fusion-based edge extraction (Section C) can significantly deform edge strengths. For example, the strength of an edge detected by both algorithms will be greater than the strength of an edge detected by only one detector. This observation led us to the use of watershed segmentation which provides robust means to cope with such situations.

| Applied modifications | | |
|---|---|---|
| **Method** | **Baccar et al.** | **Hulik et al.** |
| **Depth-based** | First- or second order derivative computed from second order polynomials | Binary accumulation |
| **Normal-based** | Normals computed from second order polynomials | Binary accumulation |
| **Fusion-based** | Averaging, Super Bayesian or Dempster-Shafer | Weighted sum |

**Table 1.** Summary of the original approach modifications

The concept of watershed segmentation can be explained by the similarity of a gradient image and the Earth surface. When it is iteratively flooded from regional minima and two basins are about to merge, a "dam" separating the two watersheds is raised.

We adapted the segmentation technique by implementing a simple minima-search algorithm. Knowing that the input for segmentation is an edge-strength gradient image (values represented by integers), we simply take each basin as a union of connected points with values lesser than a threshold.

This technique has proven to be reliable in the context of integer-represented edge gradient images. The threshold can be set close to zero as the edge extraction technique is generally insensitive to non-edge regions.

By applying the watershed segmentation to edge detectors discussed above, we obtained three scene segmentation methods, which were evaluated and compared:

1. depth-based segmentation (**DS**)
2. normal-based segmentation (**NS**)
3. segmentation by fusion of DS and NS (**FS**)

## IV. New Segmentation Techniques

In addition to the optimized methods proposed above, we designed two novel approaches specifically tailored for indoor scene segmentation. As mentioned in the Introduction, planar regions are typical for human-made indoor environments. The new techniques make use of this fact by focusing on plane detection in indoor scenes.

### A. Plane prediction segmentation (*PS*)

A novel depth map segmentation method, inspired by state-of-the-art approaches, is based on detecting local gradients. The method benefits from an a priori assumption that a majority of significant objects in the scene (objects to be detected) are human-made. They are supposed to have planar faces or can be approximated by them. Two gradient images are computed:

$$f_P(\mathbf{x}) = \sum_{\mathbf{r}\in W(\mathbf{x})} \begin{cases} 1 & if\ |p(\mathbf{r},\mathbf{x}) - d(\mathbf{r})| > t \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$f_C(\mathbf{x}) = \sum_{\mathbf{r}\in W(\mathbf{x})} \begin{cases} 1,\ if\ \left( \begin{array}{l} |\,p(\mathbf{r},\mathbf{x}) - d(\mathbf{r})| > t\ and \\ |p(\mathbf{r}-\mathbf{1},\mathbf{x}) - d(\mathbf{r}-\mathbf{1})| \le t \end{array} \right) or \\ \quad \left( \begin{array}{l} |\,p(\mathbf{r},\mathbf{x}) - d(\mathbf{r})| \le t\ and \\ |\,p(\mathbf{r}-\mathbf{1},\mathbf{x}) - d(\mathbf{r}-\mathbf{1})| > t \end{array} \right) \\ 0 \qquad \text{otherwise,} \end{cases} \quad (5)$$

where $d$ represents a real depth of a specified pixel and $p$ is a predicted theoretical depth of a pixel computed as follows:

$$p(\mathbf{x},\mathbf{c}) = \nabla d(\mathbf{c}) \cdot \big(d(\mathbf{x}) - d(\mathbf{r})\big) + d(\mathbf{r}) \quad (6)$$

$\mathbf{c}$ defines a center point with specified gradient, $p(\mathbf{x},\mathbf{c})$ is a theoretical depth of a point $\mathbf{x}$ predicted from point $\mathbf{c}$ using its gradient, $f_C$ from formula 5 is the number of changes in

the process of detection in a current window. Changes are defined as differences in thresholding in formula 4, e.g., if a current pixel has been to 1, a precedent one to 0, the change appeared. The value is used to identify a current region – a large number of changes indicate a planar noisy area. This value can be also used for statistical region merging.

The result is represented as an integer edge gradient image, so it is easy to apply the described watershed segmentation method to obtain a desired region map.

### B. *Tiled RANSAC segmentation (RS)*

In search for a very fast and reliable segmentation algorithm for indoor depth scenes, we devised another solution that uses RANSAC for the ground plane search. It came out from the approach presented in [7]. We adapted the method by turning a planar detection algorithm into a depth map planar region segmentation procedure. The resulting algorithm excels in the segmentation of indoor scene images in which planar objects dominate.

To cope with the large computational cost of the RANSAC search, we had to develop a specific algorithm for the plane search which takes into account only small areas of the scene. A depth image is covered by squared tiles which define only a small search area for RANSAC, but sufficiently large area for robust plane estimation from noisy images. The algorithm is sketched in Figure 2.

```
1. Compute normals
2. For each tile
    2.1. Try to find an existing triangle
         using RANSAC
    2.2. If found plane
        2.2.1.  Seed-fill the whole tile
        2.2.2.  Seed-fill
```

**Figure 2.** Tiled RANSAC (RS) algorithm

In step 2.1, RANSAC is used to find an existing plane in a current tile. This means a random search for plane candidates from pixels that has not been segmented yet. A plane is found if it has all three defining point connected, i.e., there are pixels on the triangle plane between all three triangle vertices.

If a plane was found, a seed-fill algorithm will group all connected plane points in the current tile (2.2.1). Seed filling is fast and it is executed only on a detected plane. Each pixel is then seeded only once.

The last step (2.2.2) fills the rest of the depth map for regions reaching borders. This spreads the region out of the tile and prevents creation of artefacts that could result from the tile search. If a large plane is found, this step also reduces the number of tiles searched in further iterations of 2.1 by pre-filling regions. The ability to fill regions outside the tile borders ensures that identified planes are marked in the whole depth image.

Because of its small random sample search, the tiled RANSAC is often used in real-time systems. The algorithm can reach speed of multiple frames per second.

## V. EXPERIMENTAL RESULTS AND DISCUSSION

In order to evaluate proposed methods, we designed a series of tests focusing on performance and accuracy. Each frame of the dataset was manually annotated to represent an ideal segmentation result (the ground truth). Figure 3 shows an example of such annotation. We used 20 different manually annotated frames for the accuracy comparison and 20 30-second frame sequences to evaluate the computation efficiency.

The output of all five methods was collected. The segmentation was compared to the ground truth data and the percentage of correctly/wrongly segmented pixels was counted. Additionally, we provide a comparison with PCL's [17] RANSAC point cloud segmentation method. Due to the parallelism in PCL's method, we used OpenMP [18] library to parallelise our solution as well. An average computational time of the segmentation process run on 640x480-pixel images was also measured (Intel Core i7-2620M, 2.70 GHz). Results are summarized in Table 2. The graph in Figure 4 characterizes the performance of the methods relatively to the results of the slowest/most accurate method.

As expected, the FS and RS algorithms provide the most accurate segmentation. The FS algorithm was designed to precisely detect roof and step edges and the watershed segmentation method contributes to its robustness. The key disadvantage is the high computation time, the second largest among the proposed approaches. It is due to computation of normals for the whole depth map. The computation must be robust so that a large neighbourhood needs to be taken into account (window size 7x7 – 11x11 pixels, larger neighbourhoods would result in imprecise segmentation on object's boundaries as normals would be deformed by the difference in depth).

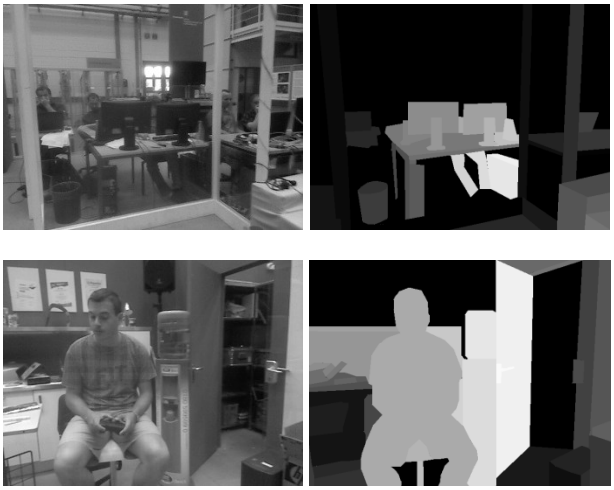

**Figure 3.** Sample images from manually annotated dataset

The tile RANSAC search (RS) has proven to be a precise segmentation method too. The machine comparison results are almost the same as that of the FS method. Moreover, a visual comparison of the segmented images reveals that the method eliminated a problem related to the border noise. On the other hand, it gets into difficulties with planes

| Method | DS | NS | FS | PS | RS | PCL |
|---|---|---|---|---|---|---|
| **Correctly segmented (%)** | 73.11±20.39 | 83.41±16.63 | 87.35±10.86 | 83.40±10.32 | 86.21±10.62 | 78.67±12.95 |
| **Wrongly segmented (%)** | 23.49±18.29 | 13.04±11.49 | 10.22±8.96 | 15.44±9.41 | 11.71±8.25 | 19.02±9.16 |
| **Time (ms)** | 28.49±8.19 | 134.00±27.38 | 151.85±34.72 | 138.20±41.37 | 97.89±13.63 | 254.12±194.31 |

**Table 2.** Table comparing the accuracy and speed of implemented segmentation methods.

consisting of a small number of inlier points – the normals are not computed precisely. Resulting region images need to be post-processed using a region merging metric which join similar planes together. The computation time is also a strong attribute of this method.

The PS method provides the same accuracy as the NS. However, its results are far more acceptable than that of the NS method when one compares the segmentations visually (see Figure 4). It is due to the precise detection of planar regions and the sensitivity to small details. Both methods suffer from the same problem – if the difference between a suggested plane and a real pixel is below a threshold, no line is detected. This poses problems on rounded edges which are detected as a local noise.

Also note that the performance of the PS and RS methods cannot be simply compared with other segmentation algorithms, because of the a priori assumption on the scene shape. The algorithms are expected to produce a noisy output for outdoor scenes.

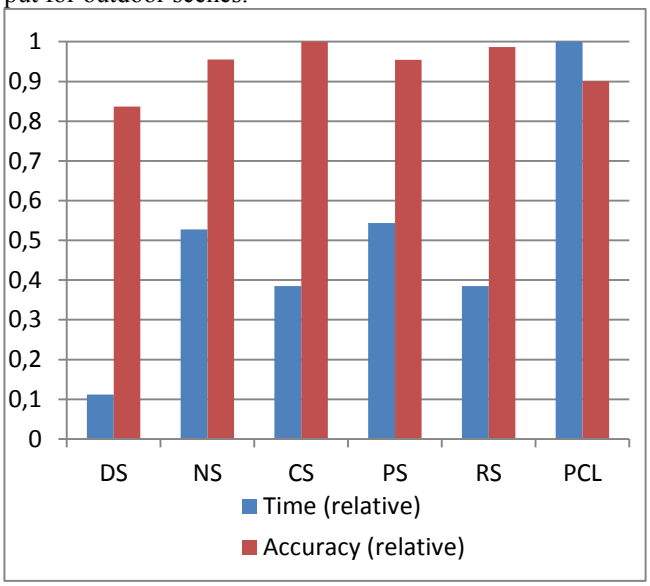


**Figure 4.** Time and accuracy relative to results of the tested algorithms

It is clear that the DS algorithm is far more efficient in speed than the others. It employs minimal floating point arithmetic and minimal image computations. On the other hand, it is also the least precise. This comes from its nature – it does not detect roof shaped edges. Despite that, the method is a good candidate for general pre-processing. It precisely detects depth differences so that clear boundaries of different objects can be easily identified. Accuracy problems arise when the method is used to distinguish large continuous objects such as wall corners.

Comparing our methods with the PCL's RANSAC approach, it is clear that we successfully speeded up the segmentation process while retaining necessary precision. The lower precision in PCL method is due to the global search of compared algorithm. The local approach in our methods has better results for depth image segmentation.

The graph in Figure 4 also clearly shows that the time consumption of the PS method is minimal when compared to its relatively high accuracy. Thus, the technique is also a good candidate for inclusion in very fast depth image pre-processors.

Figure 5 shows a visual output of all the methods on two test samples. To better demonstrate the results, regions are not post-processed by the hole-filling algorithm.

## VI. Conclusions

Five different depth-map segmentation methods were described and evaluated in this paper. We modified three well-known segmentation techniques to minimize their time constraints. Additionally, two new algorithms – the plane prediction segmentation and the tile RANSAC search were presented. They take advantage of the assumption on plane dominance in indoor scenes.

Evaluations were run to assess the performance of the implemented methods. Speed and accuracy figures were compared on a dataset consisting of manually segmented indoor scene images. A visual comparison of the resulting segmentations was also performed. Although the machine-computed accuracy of the methods is similar, the visual comparison shows large differences. There are also significant differences in speed.

The usability of the segmentation methods based on plane detectors depends on the nature of the segmentation task – these methods are precise in planar objects segmentations, non-planar ones can pose problems. It is recommended to post-process segmented images by region merging and hole filling algorithms, which can significantly increase the usability in practical applications.

In future, we are planning to further parallelise and optimise proposed methods to reach the real-time performance (<33.3 s/frame). GPU implementations are not supposed now because of use of these methods primarily on small, embedded systems. Also, further analysis and comparison with today's segmentation methods is advised.

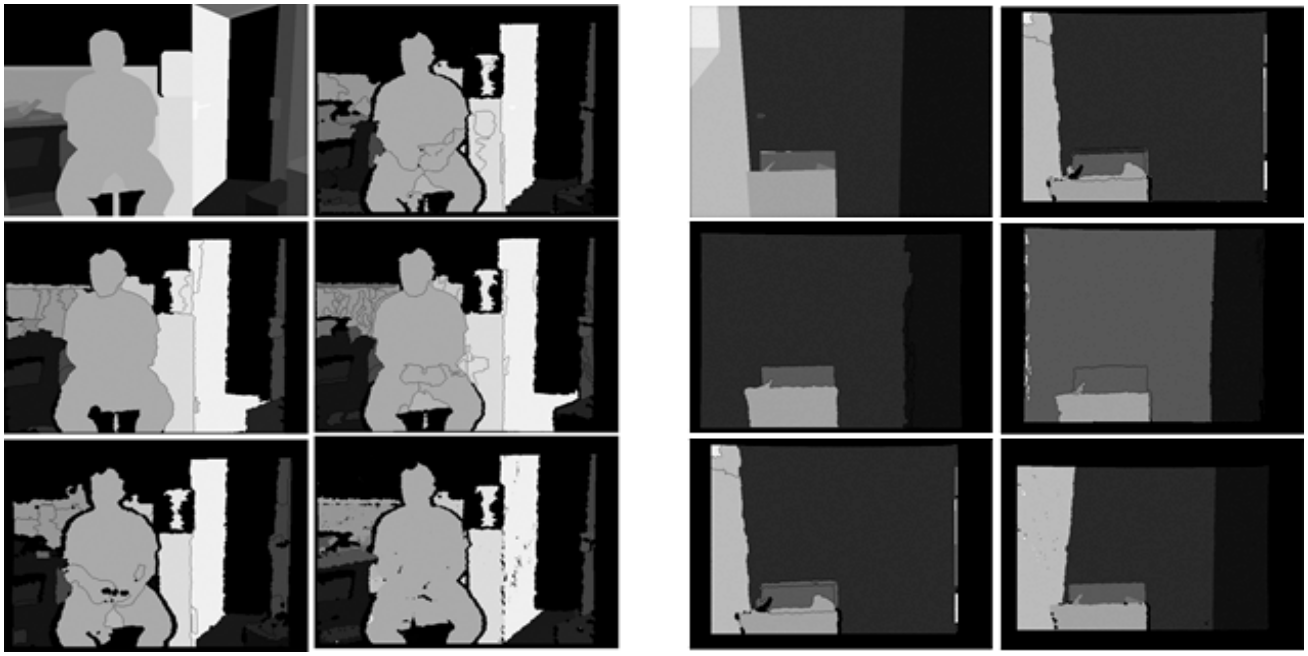

**Figure 5.** Output visualization: upper-left: manual, middle-left: DS, bottom-left: NS,


## ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union, 7th Framework Programme, grant 247772 – SRS, Artemis JU grant 100233 – R3-COP, and the IT4Innovations Centre of Excellence, grant n. CZ.1.05/1.1.00/02.0070, supported by Operational Programme "Research and Development for Innovations" funded by Structural Funds of the European Union and the state budget of the Czech Republic.


## REFERENCES


[1] Pulli, K., Pietikäinen, M.: *Range Image Segmentation Based on Decomposition of Surface Normals*. University of Oulu, Finland, 1988.

[2] Besl P., *Surfaces in Range Image Understanding*, Springer-Verlag. New York, 1988.

[3] Taylor, R., Savini, M., Reeves A.: *Fast Segmentation of Range Imagery into Planar Regions.* Computer Vision, Graphics, and Image Processing, vol. 45, pp. 42-60, 1989.

[4] Rousseeuw, P., Leroy, A.: *Robust Regression & Outlier Detection*. John Wiley & Sons, 1987.

[5] Poppinga, J.; Vaskevicius, N.; Birk, A.; Pathak, K.: *Fast plane detection and polygonalization in noisy 3D range images*. Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on , vol., no., pp.3378-3383, 22-26 Sept. 2008.

[6] Baccar M., Gee, L. A., Gonzalez, R. C. and Abidi, M. A.: *Segmentation of Range Images Via Data Fusion and Morphological Watersheds.* Pattern Recognition, Vol. 29, No. 10. (October 1996), pp. 1673-1687.

[7] Ying Yang, M., Förstner, W.: *Plane Detection in Point Cloud Data.* Proceedings of the 2nd International Conference on Machine Control Guidance Bonn (2010), Issue: 1, Pages: 95-104.

[8] Oßwald, S., Gutmann, J.-S., Hornung, A., Bennewitz, M.: From 3D point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids. In: Proceeding of the 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2011), Bled, Slovenia, October 26-28, 2011

[9] Borrmann, D., Elseberg, J., Lingemann, and K., Nüchter, A.: The 3D Hough transform for plane detection in point clouds – A review and a new accumulator design. 3D research, Springer, Volume 2, Number 2, March 2011.

[10] Dube, D. and Zell, A.: Real-time plane extraction from depth images with the Randomized Hough Transform. In *IEEE ICCV Workshop on Challenges and Opportunities in Robot Perception*, pages 1084 - 1091, Barcelona, Spain, November 2011.

[11] Shapovalov, R. and Velizhev, A.: Cutting-Plane Training of Non-associative Markov Network for 3D Point Cloud Segmentation. In Proceedings of the 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT'11). IEEE Computer Society, Washington, DC, USA, 2011, pp. 1-8.

[12] Wang, L., Cao, J. and Han, C.: Multidimensional particle swarm optimization-based unsupervised planar segmentation algorithm of unorganized point clouds. Pattern Recogn. 45, 11, November 2012. pp. 4034-4043.

[13] Zheng, P. and Zhang, A.: A Method of Regular Objects Recognition from 3D Laser Point Cloud. Lecture Notes in Electrical Engineering, 1, Volume 126, Recent Advances in Computer Science and Information Engineering, 2012. Pages 501-506.

[14] Elseberg, J., Borrmann, D., and Nüchter, A.: Efficient Processing of Large 3D Point Clouds. In Proceedings of the XXIII International Symposium on Information, Communication and Automation Technologies (ICAT '11), IEEE Xplore, ISBN 978-1-4577-0746-9, Sarajevo, Bosnia, October 2011.

[15] Sithole, G. and Mapurisa, W.T.: 3D Object Segmentation of Point Clouds using Profiling Techniques. South African Journal of Geomatics, Vol. 1, No. 1, January 2012.

[16] Deschaud, J. E., Goulette, F.: 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, Visualization and Transmission (3DPVT'10), 2010.

[17] Rusu, R.B. and Cousins, B.:3D is here: Point Cloud Library (PCL). In: Proceedings of the International Conference on Robotics and Automation, 2011, Shanghai, China.

[18] Menon R., Dagum L.: OpenMP: an industry standard API for shared-memory programming. In: IEEE Computational Science and Engineering, Vol. 5, No. 1. (1998), pp. 46-55.