Advanced Plane Properties by Using Level Image

Pavel Chmelar, Ladislav Beran, Natalija Chmelarova, Lubos Rejfek
Department of electrical engineering
Faculty of electrical engineering and informatics
Pardubice, Czech Republic
pavel.chmelar@student.upce.cz, ladislav.beran@student.upce.cz, nataliia.kudriavtseva@gmail.com,
lubos.rejfek@upce.cz

***Abstract*—This paper deals with advanced plane properties in analyzed point clouds. Planes are detected by the level connected component labeling, which is our modification of the classical algorithm for 3D data. According selected detection parameters and a scanning dimension the algorithm detects individual levels in an input point cloud. A level is presented by an image expressing the points' presence at a specific level in a space, we call it level image. The pixel size in a level image is equal to the point cloud distance quantization. This connection allows us to use image processing methods to get important properties about the analyzed 3D space, which can be difficult to get by the standard mathematical solution. Image processing methods offers a simple plane segmentation and visualization or area and perimeter estimation including statistical data description. The results show level image advantages.**

Keywords—plane detection; level image; level connected component labeling; point cloud; area and perimeter estimation

I. INTRODUCTION

A point cloud is a group of points placed in a specific coordinate system. Instead of common points' origin there is no relation between individual points without future processing. Nowadays measurement systems provide detailed and precise point clouds with big amount of measurement points with high memory consumption. Mobile robots are usually composed from small embedded devices. If they are using a 3D map for the navigation, this map has to be simplified to meet robots' memory space. Likewise, when we are processing measurement data, usually individual points losing its meaning. More important is an object or a structure composed from these points.

There are several methods for plane detection. The well-known state-of-art algorithm uses the RANdom SAmple Consensus (RANSAC) approach based on fitting a desired model. For example, in [1] authors show the successful implementation on data measured by the Microsoft Kinect with combination of Point Cloud Library functions. Different research [2] deals with comparison the RANSAC against the Hough Transformation (HT) for the plane detection. Both implementations are compared in real-time application on a robot. The HT overcomes RANSAC in its stability and the number of false detection. The real-time plane detection for the Augmented Reality in unknown environments uses a modified RANSAC [3]. Authors proposing to use the constrained sampling strategy for planes searching. When multiple planes are detected also specific plane properties are used for the score function to get the best match. These modifications also significantly improve the processing time.

The next approach presented in [4] uses Depth-driven Plane Detection (DPD) algorithm based on a region grooving. Close to this is the most recent research [5] presents a complex system for the plane detection and classification. Planes are segmented by the DBSCAN method which respects also curvatures to provide fine segmentation.

The paper [6] describes algorithm for the ground plane detection from depth maps captured by a stereo camera. A camera is moving and the system deals with elimination of the camera's roll angle for the correct plane segmentation. Another research in [7] deals also with the ground plane extraction which enables objects detection present in an analyzed scene.

In our last research [8] we introduced the color extraction method from measurement frames for rotary optical rangefinders including a color point cloud construction. The color information it is used mainly as the point cloud presentation and visualization, but when a large color point cloud with many points is analyzed, it is easier to identify its individual parts, and a whole scanned space looks well arranged.

In the next papers [9] and [10] we described a method for point cloud plane visualization by using the level image. The level image is a tool for expressing the points' presence in specific point cloud levels via an image. Individual points' positions are quantized by the quantization parameter qD into image pixels. The result is a point cloud level described by less points. Moreover, the level image physical origin is known which makes possible to determine each pixels' position in a space. With known detection parameters this allows to segment planes in a space and their visualization. In this paper we would like to present methods for the advanced plane properties as the area and the perimeter estimation by using the level image that connects the point cloud data with possibility of using image processing methods. This offers to easily observe important space properties even for complex plane shapes that needs difficult mathematical solution.

II. LEVEL CONNECTED COMPONENT LABELING

The classical connected component labeling algorithm using only one level. It searches individual element in a binary image and returns their indexes and the elements count. Our aim is to analyze individual point cloud levels and their statistical parameters. Fig. 1 shows an example of analyzed room point cloud.

The research was supported by the Internal Grant Agency of University of Pardubice, the project No. SGS_2018_022.

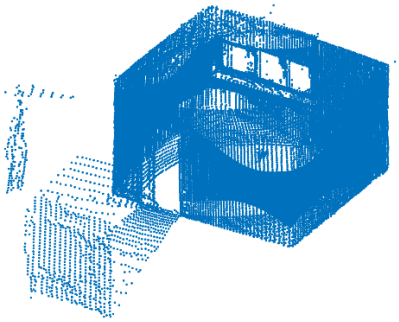

Fig. 1. A point cloud of a indoor room

For this purpose, we developed the new level connected component labeling algorithm (LCCL) for 3D data. It is based on the progressive searching of individual levels in selected scanning dimension dim_s and its marking by indexes in the similar way as the classical component labeling. In our last research [9] and [10] the algorithm was mentioned and in this paper we are describing its general form for using with any input 3D data. The pseudo code of describing algorithm gives Fig. 2.

```
Function [L, numL, (L_stat)] = LCCL(X_L, dim_s)
    idxL = 1
    numL = 0
    maxT = max(X_L(∀, dim_s))
    downT = min(X_L(∀, dim_s))
    upT = downT + lvlS · lvlRS
    while(upT ≤ maxT)
        //actual level
        dL = globalLvlStep(downT, upT, dim_s, numL)
        //select new level
        downT = dL + lvlS                                   (3)
        upT = downT + lvlS · lvlRS                          (1)
        if(upT > maxT)
        //last level
        if(downT ≤ maxT)
            upT = max(X_L(∀, dim_s)) + lvlS
            globalLvlStep(downT, upT, dim_s, numL)
            break
        end
    end
end

Function dL = globalLvlStep(downT, upT, dim_s)
    lvlX_L = (X_L(∀, dim_s) ≥ downT) ∧ (X_L(∀, dim_s) ≤ upT)
    if(lvlX_L ~ ∅)
        dL = max(h_lvl(lvlX_L))
        X_L lvlR = X_L(∀, dim_s) ≥ (dL − lvlS) ∧ X_L(∀, dim_s) ≤
                 ≤ (dL + lvlS) ∧ L(∀) = 0                   (2)
        L(X_L lvlR) = idxL
        (write L_stat(idxL, X_L lvlR))
        idxL = idxL + 1
        numL = numL + 1
    else
        dL = downT + lvlS
    end
end
```

Fig. 2. The level connected component labeling pseudo code

The statistical structure L_{stat} is optional parameter, but useful for a future processing. Analyzed data are marked as the input matrix X_L. The process begins by setting the minimal $minT$ and the maximal $maxT$ input data range in scanning dimension dim_s. For example, in case of image the analyzed data represent the pixel intensity. There are two important parameters $lvlS$ and $lvlRS$ denoting the maximal deviation from a level value and the $lvlS$ multiplication expressing the new level searching range respectively. The algorithm uses two values in each step to express the range of a searching, down threshold $downT$ and up threshold upT. For the first step the $downT$ value is set as the minimal value of the scanning dimension dim_s is $downT = min(X_L(\forall, dim_s))$. The value upT is in each step set by the equation

$$upT = downT + lvlS \cdot lvlRS. \tag{1}$$

The algorithm then searching individual levels according the $globalLvlStep$ till it reaches the maximal value $maxT$. The data range for analyzed level is expressed as

$$X_L lvlR = X_L(\forall, dim_s) \geq dL - lvlS \wedge \\ \wedge X_L(\forall, dim_s) \leq dL + lvlS. \tag{2}$$

In each step is estimated the level range histogram h_{lvl} and the maximal points' concentration in the scanning dimension dim_s is selected as the detected level dL. Fig. 3 gives the level histogram for the level detection from Fig. 4.

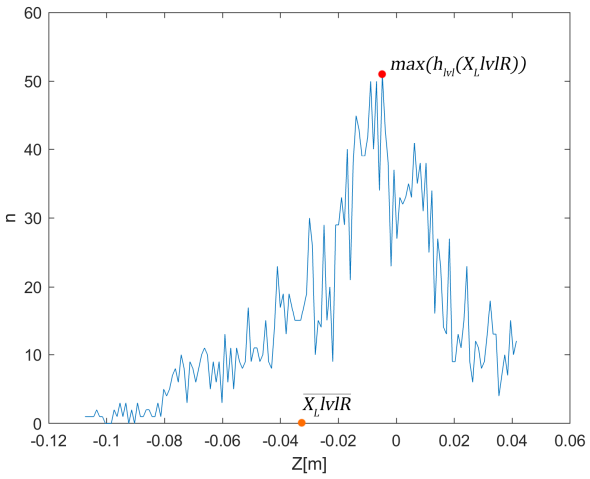

Fig. 3. A point cloud level histogram of the scanning dimension dim_s

Fig. 4 presents the level value dL estimation. The data range in Z scanning dimension is selected by equations (1) and (3). The two dots show the difference between the level estimation by the data range mean (the orange dot) and the histogram $\max(h_{lvl}(lvlX_L))$ (the red dot). In our previous research [9] and [10] we used the mean level estimation which provides inaccurate results in comparison with the histogram method for example in situation of Fig. 3. Data range $X_L lvlR$ is selected by using standard deviation $lvlS$ and the estimated level value in range $[downT, upT]$. After a level estimation in n-th step a new minimum $downT$ for next step $(n + 1)$ defined as

$$downT = dL + lvlS, \quad (3)$$

and a value upT is again set according equation (1). By combination of equations (1), (2) and (3) we achieve all levels distribution of input data X_L according its concentration in analyzed dimension dim_s. In Fix. 4 is shown one level estimation.

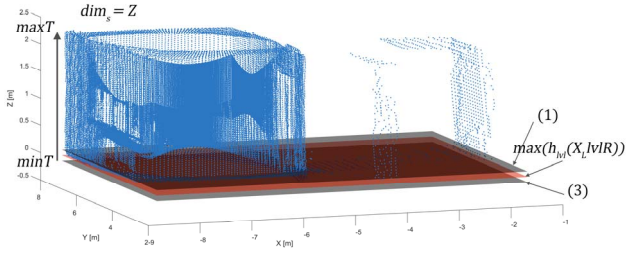

Fig. 4. The level estimation step

The given data range $X_L lvlR$ (2) determines the one level. From these data we can construct the image, see Fig. 5. We call it image level. An image level gives information about data distribution in a detected level. More information about the level image and its construction is described in our previous work [9] and [10]. Its idea is based on the distance quantization by the parameter qD.

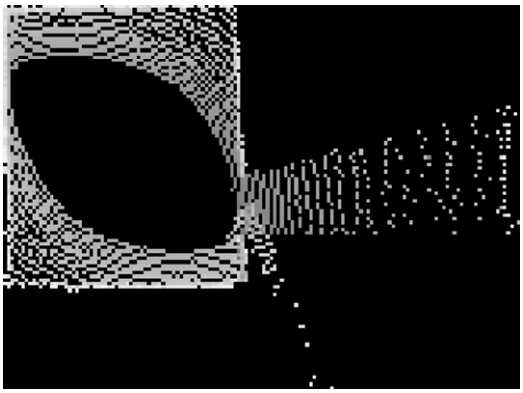

Fig. 5. The image level created from Fig. 4 by using data from (2)

In Tab. 1 are summarized the levels of point cloud from Fig. 1 in Z dimension, which consist from more than 2000 nonzero points in a level image. For a better illustration used parameters are $lvlS = 0.1\ m$, $lvlRS = 1.5\ m$ and $qD = 0.05$.

TABLE I. INDIVIDUAL LEVELS

Lvl	*dL [m]*	*dA [°]*	*Npx*	*Np*	*Lstd.[m]*
1	-0.005	0	3664	2761	0.0396
13	2.15	0	2798	1795	0.0561

The dA is detection angle, the value Np denotes level points, Npx is the nonzero image level pixels' count and $Lstd.$ shows the estimated level standard deviation in the scanning dimension dim_s.

To get compact areas in a level image it is necessary to apply morphological operations on image. Individual levels may consist from several discontinuous areas. Each level image element describes a separated plane. For their analysis it is possible to use a group of connected component algorithms as we are using for color point cloud visualization in [10]. Each level image pixel determines exact position in a space as

$$p[x,y] = I_{L_{phys}}[x,y] + I_L[x,y]qD. \quad (4)$$

Point $p[x,y]$ position is defined by the level image space origin $I_{L_{phys}}[x,y]$, the pixel position $I_L[x,y]$ and the parameter qD. This information allows to determine exact a level image pixel position in a space. For example, we can utilize it for plane visualization as in [10].

To introduce our algorithm we used only Z dimension. For all planes detection is necessary to use left dimension too or use point cloud rotation as we introduce in [9]. When the rotation is used all planes will be detected even without priory information about input point cloud orientation. But this approach is time consuming. The one LCCL scanning for used point cloud takes almost 100 ms in Matlab on Intel i5-2410M. The processing time increases linearly by increasing angle combinations. We advise to use this algorithm only for defined angles or dimensions, which requires only several algorithm steps. Presented level connected component labeling algorithm has wide using range from image analysis to large scale space data.

III. AREA AND PERIMETER CALCULATION

This paper mainly focuses on the plane area and the perimeter estimation by using the level image. Planes can be detected by any state-of-art algorithm or by the LCCL. From a plane points it is necessary to create the level image by using the distance quantization parameter qD, thus we have information about each pixel area. The pixel size is equal to 1 qD. The analyzed plane shape can be irregular and the area or the perimeter estimation can lead to significantly difficult mathematical solution. We can use advantage from the known pixel size and use image processing methods for their determination.

The plane area A_{E_x} calculation by using the level image, where each pixel express points' presence in an analyzed point cloud is defined as an area of all pixels

$$A_{E_x} = N_{E_x} \cdot qD^2, \quad (5)$$

where N_{E_x} is count of nonzero element x pixels in a level image.

The perimeter determination P_{E_x} is not possible to get simply by a count multiplication as the area. To express the element's perimeter, we are using its borders $I_L E_{xB}$, where I_L denotes the level image. Borders B we get by the element E_x with the applied erosion $I_L E_{xE}$ and binary subtraction from the original element $I_L E_x$. It is necessary further borders analysis for the precise perimeter estimation. Simple border pixels count multiplication with one pixel's size gives inaccurate result. The main requirement for the perimeter is to find its precise path, which describes an analyzed element in a level image. All mentioned problems solve the modified connected component algorithm for the borders' position determination with history of searching connected pixels. The algorithm used different

mechanism of searching connected border pixels, see Fig. 6. During the perimeter estimation, the direction of pixels' searching path P_p is also analyzed. Direction values are in a range $[1, 2, \ldots, 8]$ according connected pixel positions. Individual numbers are equal with Fig. 6.

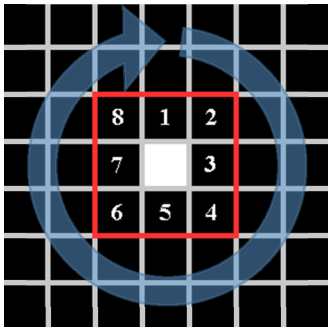

Fig. 6. The neighbor pixels searching direction

The direction analysis helps with the round perimeter detection and mainly with the optimal pixel length estimation in a searching path for the direct and the diagonal directions. For the direct pixel path there are numbers 1, 3, 5 and 7 and their length is qD. Numbers 2, 4, 6 and 8 are used for diagonal direction and their lenght is equal to one pixel's diagonal. For the simpler algorithm description and its implementation is created a set P_d, which express directly the pixel length. The values of this set are binary zeros for the diagonal direction and binary ones for the direct path. The following Fig. 7 gives example of all possible direction during the perimeter pixels searching. The optimal perimeter is marked by the light red color. The number in corner is the border pixel order and bigger numbers are pixels' directions of their detection.

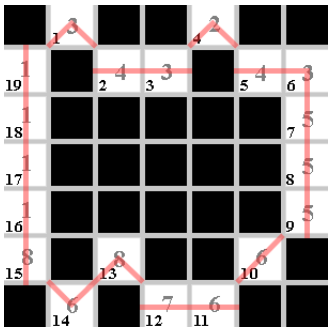

Fig. 7. The correct perimeter determination

For example, the pixel number 11 is detected in the diagonal direction, but in Fig. 7 is the detected direction as the direct path. The condition for this change is the value of the following neighbor pixel $P_d(n+1)$. The condition is defined as following

$$P_d\,(n)_{\sigma P_{d(n)=0} P_d} = \begin{cases} 1 \text{ for } \mathrm{P_d}(\mathrm{n}+1) = 1 \\ 0 \text{ for } \mathrm{P_d}(\mathrm{n}+1) = 0 \end{cases} \quad (6)$$

On positions where $P_d(n)$ is equal to zeros and the following item $P_d(n+1)$ is equal to one, then $P_d(n)$ is one too, because both are in pair. The symbol sigma in (6), (9), (10) and Fig. 8 means the mathematical selection from a set meet the written condition.

For illustration we set the parameter $qD = 0.05$ m. If we do not differentiate perimeter directions and we calculate a perimeter only by pixels count multiplication by the length qD the resulting perimeter is 0.95 m. If we include diagonal directions the result is 1.0536 m. The difference between both values is 10 %, which pose significant error for only 19 pixels.

For the round border detection, the score function is used. According the perimeter directions are added and subtracted score points in individual axes X and Y. The scoring function $X_s Y_s(N_{P_p}, 2)$, where N_{P_p} is count of elements in P_p, begins with zero values in both coordinates on position of the first found border pixel in Fig. 7, the pixel number 1. Successively in the path direction are added score points in the Y axis, direction to right, and in the X axis up direction. In opposite directions the points are subtracted. After filling all direction positions the score function is evaluated as

$$[X_s, Y_s] = \sum_{n=1}^{N_{P_p}} x_s y_s(n, :) \quad (7)$$

The element with round borders has to meet with condition

$$N_{P_p} \geq 3 \wedge |x_s| \leq 1 \wedge |y_s| \leq 1. \quad (8)$$

The given condition (8) express, then an element can have round borders with the minimal pixel count of 3 and the score function values are in absolute value smaller or equal to one. The value one signalize a neighbor pixel with the origin pixel in the direct or the diagonal direction and signalize the round end.

In case the condition (8) is true, the level image perimeter P_{E_x} is defined by the standard equation (9). The count of the direct path in P_d is multiplied by the pixel length qD and diagonal pixels' directions in P_d are multiplied by the pixels' diagonal size, mathematically written as

$$P_{E_x} = n(\sigma_{P_d=0} P_d) qD + n(\sigma_{P_d=1} P_d)\sqrt{2qD^2} \quad (9)$$

In case when the condition (8) is false the values in P_d are multiplied by two

$$P_E I_L = 2n(\sigma_{P_d=0} P_d) qD + 2n(\sigma_{P_d=1} P_d)\sqrt{2qD^2} \quad (10)$$

This equation is valid for narrow elements consist from one line with 1 px width.

The direction of the first pixel in border is estimated as the last direction pixel. When the round condition (8) is true the direction of the first pixel is evaluated from direction of the connection with the last pixel. When (8) is false the connection of the second pixel to the first is evaluated. If the connection is direct the $P_d(1)$ is 1 and vice versa. The algorithm pseudo code for the perimeter estimation is summarized in Fig. 8.

The stated function $ElementBorderPositions$ in Fig. 8 gives element borders. The function uses searching according Fig. 6. When a nonzero pixel is found the algorithm searching its neighbors. If a neighbor pixel is found in the searched direction, the neighbor searching is stopped for the actual pixel and the algorithm continues in searching a neighbor for a new

found pixel in the same way. An element border searching ends when there are no other connected neighbor pixels. More details about this function are written in our last paper [10].

Function P_{E_x} = GetElementPerimeter($I_L E_x$)

$I_L E_{x_B} = I_L E_x - I_L E_{x_E}$

$[E_{x_B}, P_p]$ = ElementBorderPositions($I_L E_{x_B}$)

$$P_d(n) = \begin{cases} 1 \text{ for } P_{p\,(P_p = 1,3,5,7)} \\ 0 \text{ for } P_{p\,(P_p = 2,4,6,8)} \end{cases}$$

$$P_d(n)_{\sigma_{P_{d(n)}} = 0 P_d} = \begin{cases} 1 \text{ for } P_d(n+1) = 1 \\ 0 \text{ for } P_d(n+1) = 0 \end{cases}$$

//fix first direction

if($P_d(2) \neq 1 \wedge P_d(2) \neq 3$)

  $P_d(1) = 0$

end

//check circle path by score function from $P_d(2)$

$$x_s(n) = \begin{cases} +1\ for\ P_p = 4,5,6 \\ +0\ for\ P_p = 3,7 \\ -1\ for\ P_p = 1,2,3 \end{cases}$$

$$y_s(n) = \begin{cases} +1\ for\ P_p = 2,3,4 \\ +0\ for\ P_p = 1,5 \\ -1\ for\ P_p = 6,7,8 \end{cases}$$

Calculate score sum $[X_s, Y_s]$ (7)

cirP = Evaluate round perimeter condition (8)

if(cirP)

  Calculate round perimeter (9)

else

  Calculate line perimeter (10)

end

end

Fig. 8. The function GetElementPerimeter pseudocode

IV. Experimental Results

The final point cloud visualization from Fig. 1 with calculated parameters is in Fig. 9. Detailed parameters are written in Tab. 2.

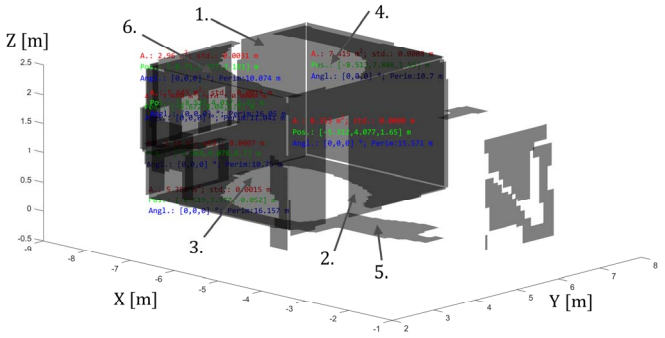

Fig. 9. The point cloud visualisation of Fig. 1

Planes in Fig. 9 are marked by numbers which correspond with numbers in Tab. 2. Table contains only main planes with the perimeter higher than 10 m. Next two tables Tab. 3 and Tab. 4 contains plane parameters with the perimeter higher than 20 m.

The table's first column Idx is plane index equal to corresponding figure. The following columns are: dL the detection level; Pos is a plane origin position in space; A_E the plane area; P_E the plane perimeter, $std.$ is the plane points standard deviation and in the last column is the scanning dimension dim_s.

TABLE II. Planes Parameters of Fig. 9

Idx	*dL [m]*	*Pos [m]*	*A_E [m²]*	*P_E [m]*	*std.[m]*	*dim_s*
1	-5.312	-5.312 4.077 -0.158	8.353	15.571	0.001	*X*
2	-8.525	-8.525 4.017 -0.091	5.645	16.05	0.0011	*X*
3	4.040	-8.672 4.04 -0.048	7.6575	11.041	0.0005	*Y*
4	7.8861	-8.513 7.886 -0.058	7.415	10.7	0.0008	*Y*
5	-0.001	-8.519 3.977 -0.001	9.055	16.157	0.0015	*Z*
6	2.191	-8.71 3.977 2.191	2.965	10.074	0.003	*Z*

The next detected planes visualization of a corridor is depicted in Fig. 10. Detail parameters about marked planes are written in Tab. 3.

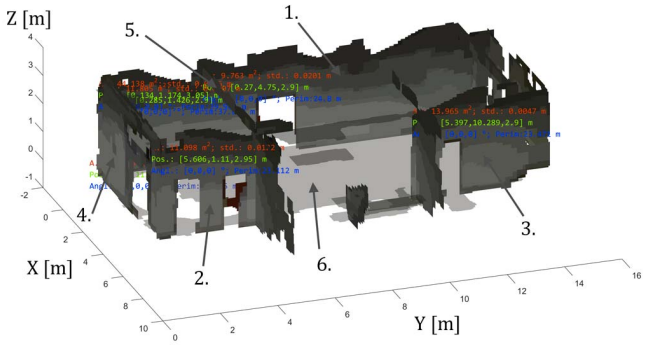

Fig. 10. The coridor point cloud visualisation

TABLE III. Planes Parameters of Fig. 10

Idx	*dL [m]*	*Pos [m]*	*A_E [m²]*	*P_E [m]*	*std.[m]*	*dim_s*
1	0.134	0.134 1.174 0.026	44.138	39.962	0.013	*X*
2	5.606	5.606 1.11 2.95	11.098	23.112	0.012	*X*
3	5.397	5.397 10.28 2.9	13.965	23.471	0.0046	*X*
4	1.426	0.285 1.426 0.01	11.805	37.062	0.0097	*Y*
5	4.749	0.27 4.75 2.9	9.7625	24.8	0.0201	*Y*
6	0.154	0.947 0.154 -0.111	48.233	69.585	0.0128	*Z*

Another color point cloud of a room shows Fig. 11 and its planes are visualized in Fig. 12. Detected plane parameters summarizing Tab. 4.

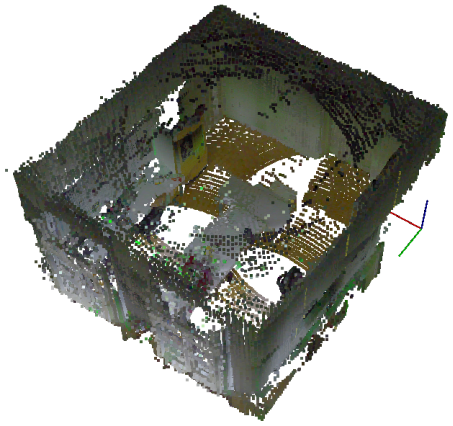

Fig. 11. The point cloud of a room

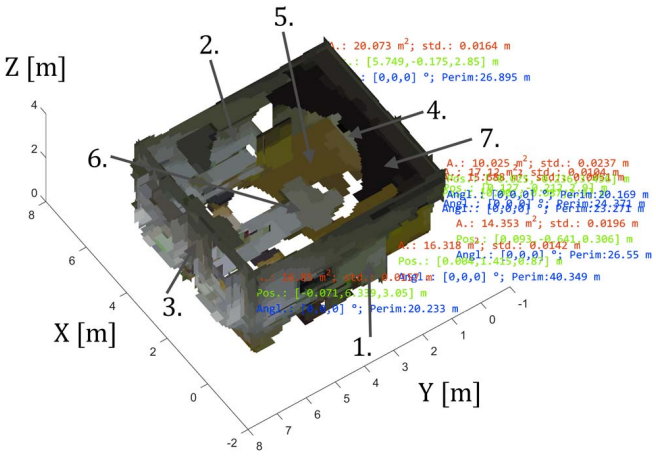

Fig. 12. The point cloud visualisation of Fig. 11

TABLE IV. PLANES PARAMETERS OF FIG. 12

Idx	*dL [m]*	*Pos [m]*	*A_E [m²]*	*P_E [m]*	*std.[m]*	*dim_s*
1	0.127	0.127 -0.212 3.7	17.12	24.371	0.01	*X*
2	5.749	5.749 -0.175 3.65	20.073	26.895	0.0164	*X*
3	6.339	-0.071 6.339 3.85	16.85	20.233	0.0157	*Y*
4	-0.087	0.001 -0.087 3.7	15.688	23.271	0.005	*Y*
5	0.306	0.093 -0.641 0.306	14.353	26.55	0.0196	*Z*
6	0.87	0.004 1.415 0.87	16.318	40.349	0.0142	*Z*
7	3.456	-0.025 -0.236 3.456	10.025	20.169	0.0237	*Z*

The last point cloud proves the algorithm ability to detect objects like cupboards, tables and doors. There are wide possibilities to set parameters for detecting required objects. By increasing individual parameter values we are losing details and vice versa. The parameter qD is also important to express a point cloud density. When the points' density is significantly smaller than qD parameter these points will be expressed as single pixels in a level image and the application of a morphologic operation can suppress these points.

V. CONCLUSION

In this paper we presented advanced methods how to estimate the point clouds' perimeter and the area by using the level image. The analyzed plane shape can be irregular and the standard mathematical solution to estimate these parameters can be incommensurably complicated. The level image offers the connection with image processing methods for solving these tasks. Moreover, it allows to determine other important advanced statistical plane parameters including a plane visualization. The presented practical results show advantages of the image processing methods connection with the real measurement data for future processing. We also introduce the level connected component labeling algorithm for 3D data analysis in general form. This algorithm has the wide using range not only for point clouds.

REFERENCES

[1] R. Kurban, F. Skuka, H. Bozpolat, "Plane Segmentation of Kinect Point Clouds using RANSAC, " 7th International Conference on Information Technology, ICIT, Amman, Jordan, 2015, pp. 545–551

[2] R. Hulik, M. Spanel, P. Smrz, Z. Materna, "Continuous plane detection in point-cloud data based on 3D Hough Transform, " Journal of Visual Communication and Image Representation, Volume 25, Issue 1, 2014, pp. 86-97

[3] D. Kim, S. Chae, J. Seo, Y. Yang and T. D. Han, "Realtime plane detection for projection Augmented Reality in an unknown environment, " 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 5985-5989

[4] Z. Jin, T. Tillo, W. Zou, Y. Zhao, and X. Li, "Robust Plane Detection using Depth Information from a Consumer Depth Camera, " in IEEE Transactions on Circuits and Systems for Video Technology, vol. PP, no. 99, pp. 1-1

[5] T. Czerniawski, B. Sankaran, M. Nahangi, C. Haas, and F. Leite, "6D DBSCAN-based segmentation of building point clouds for planar object classification, " Automation in Construction, vol. 88, pp. 44-58. 2018

[6] P. Skulimowski, M. Owczarek and P. Strumiłło, "Ground plane detection in 3D scenes for an arbitrary camera roll rotation through "V-disparity" representation, " 2017 Federated Conference on Computer Science and Information Systems (FedCSIS), Prague, 2017, pp. 669-674

[7] R. A. Zeineldin, and N. A. El-Fishawy, "Fast and accurate ground plane detection for the visually impaired from 3D organized point clouds, "2016 SAI Computing Conference (SAI), London, 2016, pp. 373-379

[8] P. Chmelar, L. Beran, L. Rejfek, and N. Chmelarova, "The point cloud visualisation for rotary optical rangefinders, " 27th International Conference Radioelektronika (RADIOELEKTRONIKA), Brno, 2017, pp. 1-6

[9] P. Chmelar, L. Rejfek, L. Beran, and M. Dobrovolny, "A Point Cloud Decomposition by the 3D Level Scanning for Planes Detection, " International Journal of Advanced and Applied Science, 2017, pp. 121-126

[10] P. Chmelar, L. Rejfek, L. Beran, N. Chmelarova, and M. Dobrovolny, "Point Cloud Plane Visualization by Using Level Image, " International Journal of Advanced and Applied Science, 2018, in press