UNSYS digital

The Journal of Instrumentation, Automation and Systems

Robust Depth-Based Planar Segmentation Algorithm Based on Gradient of Depth Feature

Bashar Enjarini and Axel Gräser

IAT Institute, Bremen University, Germany

***Abstract*—**In this paper, a new algorithm for segmentation of planar regions from depth images is proposed. The development of a new segmentation algorithm was motivated by the results of other state-of-the-art algorithms which are optimized to segment depth images acquired by the laser scanners or structure light cameras and which however, have delivered low accuracy results when applied on depth images computed from stereo cameras. The proposed planar segmentation algorithm is based on a novel feature to be called Gradient of Depth feature (*GoD*). The proposed *GoD* feature is a parameter-free and it is computed in the two-dimensional (*2D*) image space. The later makes it, in comparison to local surface normal, easy to compute in terms of complexity and faster to compute in terms of computational time. The proposed *GoD* feature is implemented into an algorithm which utilizes the one-dimensional (*1D*) feature space of the *GoD* feature which resulted in increasing the robustness of the algorithm to parameters change. The intensive experimental results presented in this paper confirm the robustness of the proposed algorithm in segmentation of the planar regions of planar and non-planar (i.e. cylindrical or curved) objects in different types of images and in different scenarios. In terms of segmentation accuracy, the *GoD*-based algorithm meets or outperforms the performance of other state-of-the-art algorithms.

***Keywords*—**Planar segmentation, gradient of depth, depth image, robot vision. service robot.

I. Introduction

The importance of the service robots is increasing every day. They are used to support care for elderly citizens and those in need of care to live independently on their own in their homes for longer [1]. They are also developed as support systems for paralyzed individuals to help them to go back to professional life [2].

Working in a human environment imposes a direct challenge to the service robots in sensing and understanding the surrounding scene. A possible solution for decreasing the complexity of understanding the sensed scene for the robot is to segment the scene into a set of planar regions. Usually, most of the objects in the surroundings of the robots are flat, or can be decomposed into planar regions [3].

There are mainly two features used in literature for the segmentation process: local surface normal and scan line segmentation. The local surface normal at a point on a surface is a vector that is perpendicular to the tangent plane to that surface at that point [4]-[6]. The surface normal for each point is computed by fitting a 3D plane to the nearest k neighborhood points. Scan line segmentation technique, on the other hand, is used to detect edges between different planar regions in depth images [7]-[9]; each scan line (row, column and diagonal) is divided into different segments based on the distance between the fitted model (polynomial function) and the points on the scan line.

This paper presents a novel feature, so-called Gradient of Depth feature (GoD). In contrary to the previous features, the proposed Gradient of Depth (GoD) feature is parameter-free and it is computed in the 2D image space which makes it, compared to local surface normal, easier to compute in terms of complexity and faster in terms of computational time. The proposed feature is defined by two components, the Magnitude Gradient of Depth ($MGoD$) and the Directional Gradient of Depth ($DGoD$). The $DGoD$ value relates to the 3D orientation of the plane while the $MGoD$ value relates to the jumping edge through the depth levels. The feature space of the $DGoD$ component is one-dimension [0° 360°] which makes the GoD feature convenient for clustering algorithms. The GoD feature is implemented into a robust algorithm which has many advantages over the other related algorithms: In addition to the high segmentation accuracy, the proposed algorithm is not limited to segmentation of the planar objects, but also can be used for segmentation of the cylindrical and curved objects. Moreover, the algorithm is robust to parameters change which makes it more convenient to be used for segmentation of different images captured in different scenarios. The ABW dataset [10], Perceptron dataset [11] as well as depth images taken from a real-world scenario were used to evaluate the performance of the proposed GoD-based algorithm and to compare it to other state-of-the-art algorithms. As confirmed by the presented results; the proposed algorithm outperforms other state-of-the-art algorithms. In terms of segmentation accuracy, the proposed algorithm meets and, in some cases, outperforms state-of-the-art algorithms. In terms of robustness, the proposed algorithm manages to segment correctly planar regions of cylindrical and curved objects in depth images computed from stereo camera in which other segmentation algorithms failed.

II. State of the Art

Planar segmentation algorithms can be classified into three types based on the framework used for the segmentation process. These are: region growing algorithms, clustering

Corresponding author: B. Enjarini (e-mail: enjarini@iat.uni-bremen.de).
This paper was submitted on February 10, 2014; revised on July 20, 2014; and accepted on July 20, 2014.

algorithms, and edge detection algorithms. In the edge detection algorithms; the edges between different regions are computed using the scan line technique. From the computed edge image; close boundaries are extracted where each close boundary refers to a segmented planar region [8]. In the region growing framework, neighboring points are merged into adjacent regions based on some similarity conditions [5] while clustering algorithms tends to cluster different points with similar features into one cell in the feature space. A merging process is then implemented to merge different cells into one plane [6].

The first full comparison of different planar segmentation methods was given in [12]. Four different algorithms have been evaluated using the ABW dataset [10] and Perceptron dataset [11]. As reported in [12], UE method which is based on the local surface normal and a combination framework of clustering and region growing is ranked as first in terms of correct segmentation. In the UE method; the mean and the Gaussian curvature for each point in the point cloud is estimated and each point is then assigned to a cluster based on the combined signs of both mean and curvature values. Each cluster is considered as an initial region, and a region growing process is implemented to add adjacent pixels to the initialized regions based on some similarity conditions (i.e. 3D distance and the normal difference between the point and the region). However, a major drawback of such algorithm is the number of parameters to tune (nearly twelve parameters) that should be re-tuned when using different camera/scene.

The second best algorithm reviewed in [12] is called UB method and it is based on the scan line segmentation and region growing algorithm. Each row, column and diagonal in the depth image is divided into different line segments. A seed region is defined as a triple of line segments on three adjacent scan lines that satisfy conditions with respect to a minimum length, a minimum overlap, and a maximum distance between the neighboring points on two adjacent scan lines. This seed region grows by adding adjacent line segments which satisfy the perpendicular distance condition between the end points of the line segment and planar equation of the region.

In [7], an improvement to the UB algorithm was proposed which resulted in a better performance on both the ABW and Perceptron datasets. The basic idea behind the algorithm in [7] is based on detecting the edges between the different regions using the scan line approximation technique. The planar regions are then extracted by searching the close boundaries in the resulted edge image. To overcome the problem of the open boundaries produced by the algorithm, an iterative process based on dilation and planar verification is performed on the edge image [8]. Similar to UB method, another algorithm presented in [9] uses the scan lines and a Flood Fill algorithm to segment planar regions from noisy input range data.

Recent applications and algorithms, however, are based mainly on the local surface normal feature. In [5] the point cloud of industrial installations is segmented using the region growing framework. The surface normal alongside with its residual are computed for each point, and points with minimum residuals are defined as seed points. Points are merged into adjacent regions if the difference in angles between the normal of a point and the seed point of its adjacent region is lower than a predefined threshold. Another segmentation algorithm that is based on the local surface normal and voxel grid clustering was used in [6] and implemented in the Point Cloud Library (PCL) [13]. Two clustering steps are used to segmenting the local surface normal: first, initial clusters are defined in the normal space from points whose surface normal fall in the same cell and second, the initialized clusters are re-defined in the distance space by separating clusters which represent more than one plane. In [14], the presented algorithm uses a combination of Mean Shift segmentation and Graph theory to segment the point cloud. However, performance evaluation on a known dataset is missing in both [6] and [14]. In [4], a Multi-Resolution plane segmentation of 3D point cloud is proposed where the cloud is sampled at different resolutions and at each resolution planar regions are extracted using Hough transform. The presented approach shows good results, however it failed in segmentation of the small planar regions.

Each of the previous features has its own limitation and drawbacks. The local surface normal feature is parameter sensitive; the size of the neighborhood for each point in the input data plays an essential role in the performance of the algorithm; a small value will result in noisy normals while a big value will lead to diminishing of sharp features such as edges and corners. Another drawback is the expensive processing time needed to compute the normal for each point in the input data by fitting planar model to k neighborhood points. On the other hand, using the scan line segmentation does not guarantee to extract close boundaries. The limitation of using the scan line segmentation technique on depth images computed from stereo camera was also confirmed in the experiments presented in this paper. As given in Section IV, it failed to produce a close boundary even in the case of a simple box-like object. Moreover, the algorithm produces a lot of noise edges.

The development of a new planar segmentation algorithm was motivated by the results from other state-of-the-art algorithms which are tailored to segment depth images generated from the laser scanners and structure light cameras, but failed to segment depth images computed from stereo camera of normal textured objects which are usually present in many indoor scenarios.

III. GoD-Based Planar Segmentation Algorithm

The GoD-based planar segmentation algorithm is depicted in **Figure 1**. The Input to the algorithm is a depth image; that includes depth images taken using laser scanner, structure light cameras or depth images computed from stereo camera. However, low resolution depth images (below 200×200 pixels) with a high level of noise generated from Time of Flight (ToF) cameras (i.e. SwisRanger 3D camera) are considered challenging to be segmented by the proposed algorithm.

The algorithm computes at first the Gradient of Depth (GoD) feature from the input depth image. To overcome the problem of noise in depth images and to take the advantage of the 1D feature space; a voting-based clustering process is utilized followed by a verification process. The verification process is based on RANSAC outlier detection [15] and has two benefits: 1) it helps to avoid under-segmentation that could result from the clustering process and 2) it makes the proposed algorithm robust to parameters change and possible for the use in wide range of applications. Over-segmentation problem is solved by using a region merging process which tends to merge over-segmented regions based on the 3D distance between two regions. A post processing step which includes adding non processed pixels and fine tuning of the regions boundary is also implemented to increase the accuracy of the segmentation.

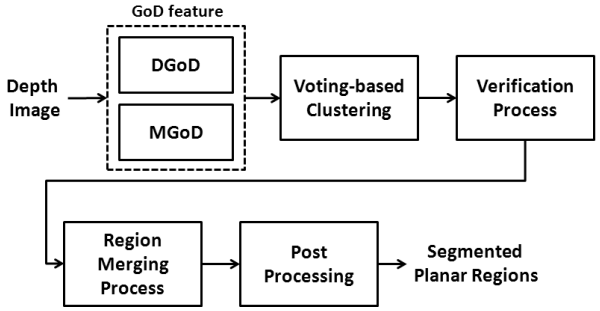


Figure 1 Block diagram of the *GoD*-based Planar Segmentation algorithm

A. Gradient of Depth Feature

The Gradient of Depth (GoD) feature is computed in the depth image space. The GoD feature for each pixel p is defined by two components: Magnitude Gradient of Depth ($MGoD$) and Directional Gradient of Depth ($DGoD$). The $DGoD$ value at pixel p is computed using the following equation:

$$DGoD_{p(y,x)} = \tan^{-1}\frac{dy}{dx} = \frac{p(y+1,x) - p(y-1,x)}{p(y,x+1) - p(y,x-1)} \quad (1)$$

where $p(y,x)$ is the pixel value (i.e. depth value) at y image row and x image column coordinate.

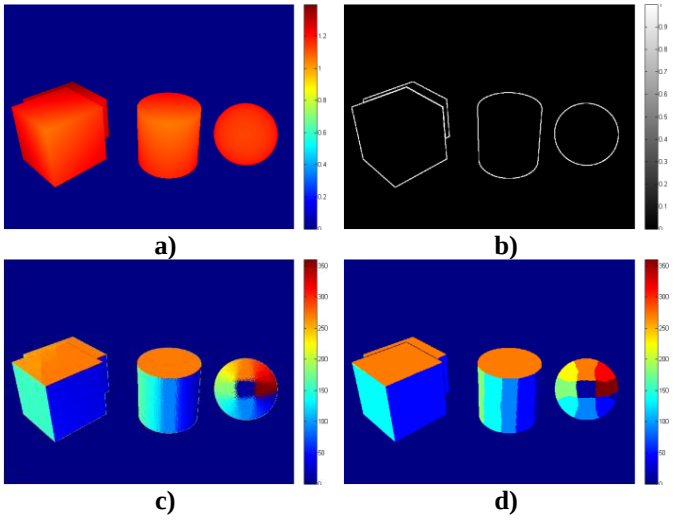


Figure 2 a) Synthetic range image; b) *MGoD* image (thresholded for clarity); c) *DGoD* image; d) Clustered image

The output of (1) is in the range of [0° 360°]. **Figure 2** illustrates the $DGoD$ values computed for pixels of a synthetic range image. The range image contains two boxes on the left where one box occludes the other, a cylindrical object in the middle and a spherical object on the right of the image. As it can be seen, pixels belonging to the same planar surface and parallel surfaces have the same $DGoD$ values while pixels belong to different surfaces, which are not parallel, have different $DGoD$ values.

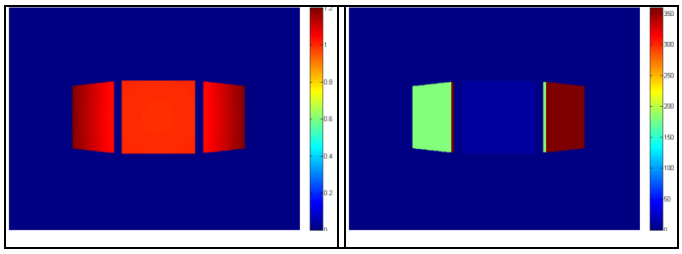


Figure 3 Special case of three planes, see the text for description

However, there are two special cases when computing the $DGoD$ value that should be taken into consideration:

1. $dx = 0 \;\&\; dy = 0$
2. $dx \neq 0 \;\&\; dy = 0$

Figure 3 illustrates an example on the two special cases. The $DGoD$ value in both cases according to (1) is equal to zero; however, each case belongs to a different plane. The first case represent a plane that is perpendicular to the optical axis of the camera (i.e. the plane is exactly parallel to the image plane, see the middle plane in **Figure 3**). On the other hand, the second case relates to a plane that is tilted around the X axis and the gradient along the Y axis is zero (see the left and right planes in **Figure 3**). In order to distinguish between the two special cases, the $DGoD$ value is set to Zero in the first case and the $DGoD$ value in the second case depends on the sign of the dx: if $dx > 0$ then $DGoD$ value is set to 360° whereas $DGoD$ value is set to 180° in case of $dx < 0$.

Equation (1) gives good results on depth images generated from range cameras with sub-pixel accuracy. However, as shown in **Figure 4**, it failed to give the expected performance on depth images with step changes in depth values. To overcome the problem of less accurate depth images; Equation (1) is modified as follows:

$$DGoD_{p(y,x)} = \tan^{-1}\frac{p(y+i,x) - p(y-i,x)}{p(y,x+i) - p(y,x-i)} \quad (2)$$

where $i = \{1, \ldots i_{max} \;||\; (dx \neq 0 \;\&\; dy \neq 0)\}$. In other words, i is set to 1 and dx and dy are computed according to (2), if both dy and dx equal to Zero; i is increased by one and (2) is computed again until both dx and dy are not equal to Zero or the maximum number of iteration i_{max} is reached. If i_{max} is reached and both dx and dy are Zero, then the pixel under the consideration belongs to a planar region parallel to the image plane (i.e. the special case 1). From the experiments, it is observed that $i_{max} = 5$ produce good results. All the experiments conducted in the work presented in this paper use

(2) in computing $DGoD$ and $i_{max} = 5$. **Figure 4** shows the difference between using (**1**) and (2) for the computation of the $DGoD$ component on a depth image from the ABW dataset with a step changes in the depth value. Comparing both images reveals that the $DGoD$ component computed using (2) produce a better results than the $DGoD$ component computed using (**1**).

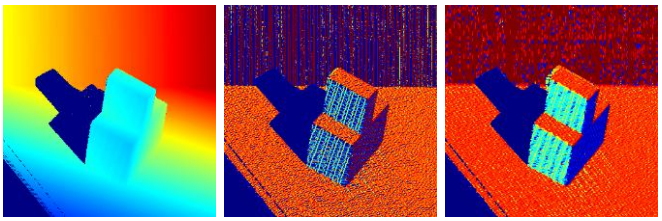
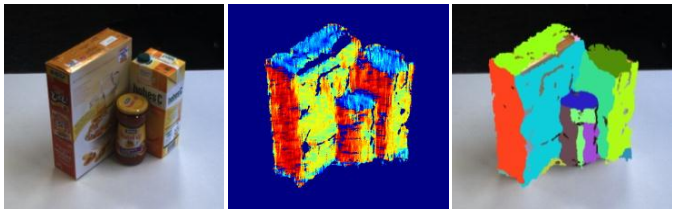


Figure 4 The difference in computing $DGoD$ feature on a depth image with step changes. Left: the input range image from ABW dataset, middle, the $DGoD$ feature map computed using (1); Right: the $DGoD$ feature map computed using (2)

The Magnitude Gradient of Depth ($MGoD$) at pixel p is given by the following equation:

$$MGoD_{p(y,x)} = \sqrt{\begin{matrix}\big(p(y+1,x) - p(y-1,x)\big)^2 \\ +(p(y,x+1) - p(y,x-1))^2\end{matrix}} \quad (3)$$

Pixels with a $MGoD$ value larger than a pre-set threshold T_{MGoD} are considered as jumping edges through the depth values. Jumping edges are important to separate adjacent parallel planes that belong to different depth levels such as the top sides of the boxes in **Figure 2**. In the work presented in this paper, T_{MGoD} is set to 2 cm, this value is sufficient to highlight the jumping edges between two parallel planes with the distance between each other of 2 cm. It is worth to mention that the T_{MGoD} value should be larger than the depth error of the camera used. In other words, if the camera depth error is 1 cm, then choosing a T_{MGoD} value below the camera depth error will result in detecting a lot of pixels as false jumping edges. Another note to consider here is that the computation of (3) using the iterative way as in (2) will decrease the segmentation accuracy since more pixels across the real jumping edges will have high $MGoD$ values which makes it hard to distinguish the real edges from the spurious edges. More details about using $MGoD$ value is explained next in the clustering process.



a) **b)** **c)**

Figure 5 Example of the GoD-based planar segmentation on a real indoor scene; a) The left input image of a stereo camera; b) The $DGoD$ feature map; c) The final segmented image

As the computation of the GoD feature is a pixel wise operation; the computation time for this operation is linear with a complexity of $O(N)$ where N is the number of pixels in the image.

In the ideal case, the $DGoD$ value of pixels belonging to the same planar surface or to the parallel surfaces is equal. However, in the real-world applications and due to noise, the $DGoD$ value of pixels belong to the same plane may shift from the ideal value, as seen in **Figure 5b**, so that a clustering process is needed.

B. Voting-Based Clustering

In this paper, a modified clustering process based on a voting histogram is proposed. Voting based histogram, also known as orientation histogram, has been used in different works [16]-[17], and it has been adapted in this work for the clustering process. The adaptation of a clustering process is motivated by the 1D feature space of the $DGoD$ component. A mathematical model of the voting histogram could be described as follows:

$$h_{voting} = \left\{V_c;\ \forall c \in \left\{[0\colon m-1] \times \frac{360}{m-1}\right\}\right\} \quad (4)$$

where V_c is the number of votes per cluster c, m is the number of clusters in the voting histogram. The X axis of the histogram is the clusters' bins and the Y axis is the number of votes per cluster. The clusters are distributed equally over the 1D feature space of the $DGoD$. For instance, $m = 5$ will give a set of clusters $c = \{0,90,180,270,360\}$.

Figure 6 illustrates the idea using the voting histogram in the clustering operation. For each pixel in the computed $DGoD$ image, a voting histogram with predefined number of clusters m is initialized for $(n \times n)$ neighborhood region. Each pixel in the neighborhood region votes for a specific cluster in the histogram, and the pixel value in the output clustered image belongs to the cluster with the highest number of votes.

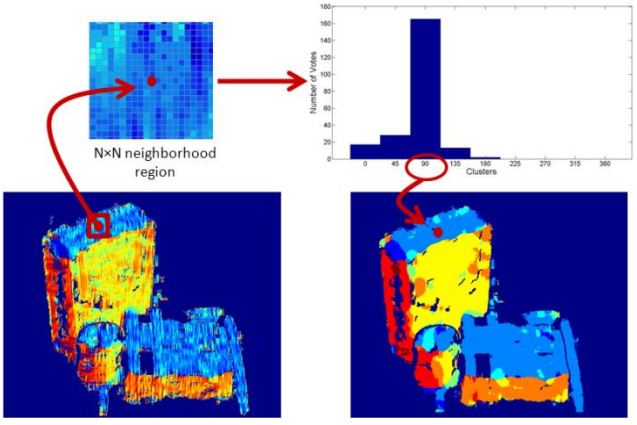


Figure 6 Illustration of the voting-based clustering process

To prevent the merging of two parallel adjacent planes into one cluster; pixels with $MGoD$ value larger than the predefined threshold T_{MGoD} are removed from the clustered image (see the upper surfaces of the two boxes in **Figure 2**). The removed pixels are defined as unspecified pixels and they are added to

the segmented image as it will be explain in Section III.E. The output clustered image contains disjoint clusters which are considered as initial segmented regions.

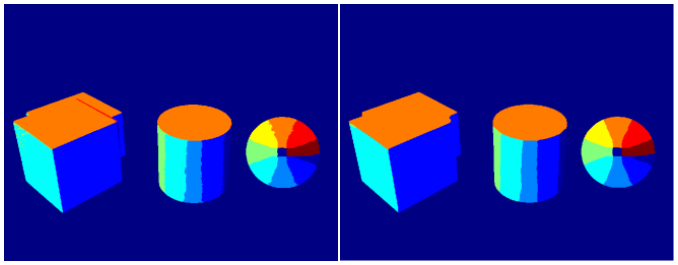


Figure 7 Clustering process on the synthetic range image shown in Figure 2 using different n values. Left: n = 7. Right: n = 19

There are two parameters to be defined in the clustering process. The first parameter is the size of the neighborhood region n. On the first sight, it might be seen that the size of n is similar to the size of the neighborhood region k in the local surface normal, however, it is not the case. Unlike the local surface normal, changing the value n does not really affect the quality of the final segmented Image. **Figure 7** shows the result of the clustering process on a synthetic image using two different values for n. The clustered images for both n values look almost identical for the three objects in the scene. This means that n does not have any influence on segmentation of range images with low noise level. However, as seen in **Figure 8**, using different n values in the clustering process on range images of a real object will result in slightly different clustered images. On one hand, choosing a low n value tends to produce more initial regions for the same surface; on the other hand, using a large n value tends to produce less initial regions without affecting the geometrical shape of the object (i.e. edges and corners) and the final segmented images are not affected by changing the value of n. Having more initial regions in the clustered image will affect the processing time of the algorithm, though. More on that will be given in the performance evaluation Section IV.

The second parameter to be defined in the voting process is the number of clusters, m, in the voting histogram. On one hand, choosing a large m value could lead to over segmentation problem and to an increase in the processing time. On the other hand, choosing a low m value could lead to under segmentation problem. To overcome this problem; a fixed number of clusters is chosen: by default, nine cluster, $m = 9$, $i = \{0, 45, 90, 135, 180, 225, 270, 315, 360\}$, are used to initialize the voting histogram. The clustering process is followed by a verification process to make sure that each disjoint clustered region contains only one planar region. The verification process is described in details in the next process.

The experiments presented in this paper show that choosing different parameters in the clustering process does not have a big influence on the segmentation accuracy thanks to both verification process and merging process.

The clustering process is a pixel wise operation and the processing time is $O(N)$ where N is the number of pixels in the image.

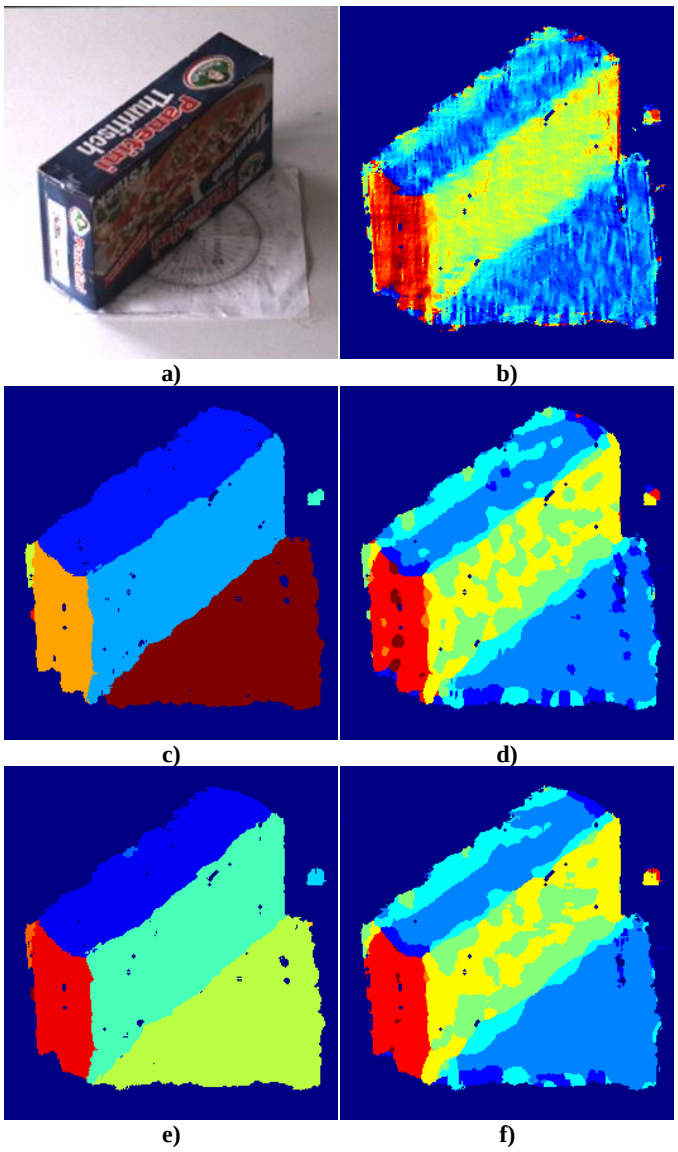


Figure 8 Clustering process in segmenting a real object using different n values. a) The left input image from stereo camera; b) The corresponding $DGoD$ feature map; d), f) The clustered image of the $DGoD$ feature map using $n = 11, 17$ respectively; c), e) The corresponding final segmented images

C. Verification Process

This process will solve any under-segmentation problem resulted from the clustering operation and will also increase the accuracy of the segmentation algorithm. The verification process is based on the RANSAC algorithm for outlier detection [15]. For each initial segmented region resulted from the clustering process; a 3D plane model is fitted using the RANSAC algorithm.

In a simple case when the initial segmented region belongs to only one planar surface, only few pixels would result as outliers due to noise. In that case, the initial segmented region is added to the output image after removing the outlier pixels. The outlier pixels are added to the unspecified points resulted from the clustering process. In an under-segmented case, the number of outlier pixels will be high and they will form a new disjoint

region. The outlier disjoint region is then added to the output image of this process as a new disjoint region. **Figure 9** shows an example of the verification process on an under-segmented region.

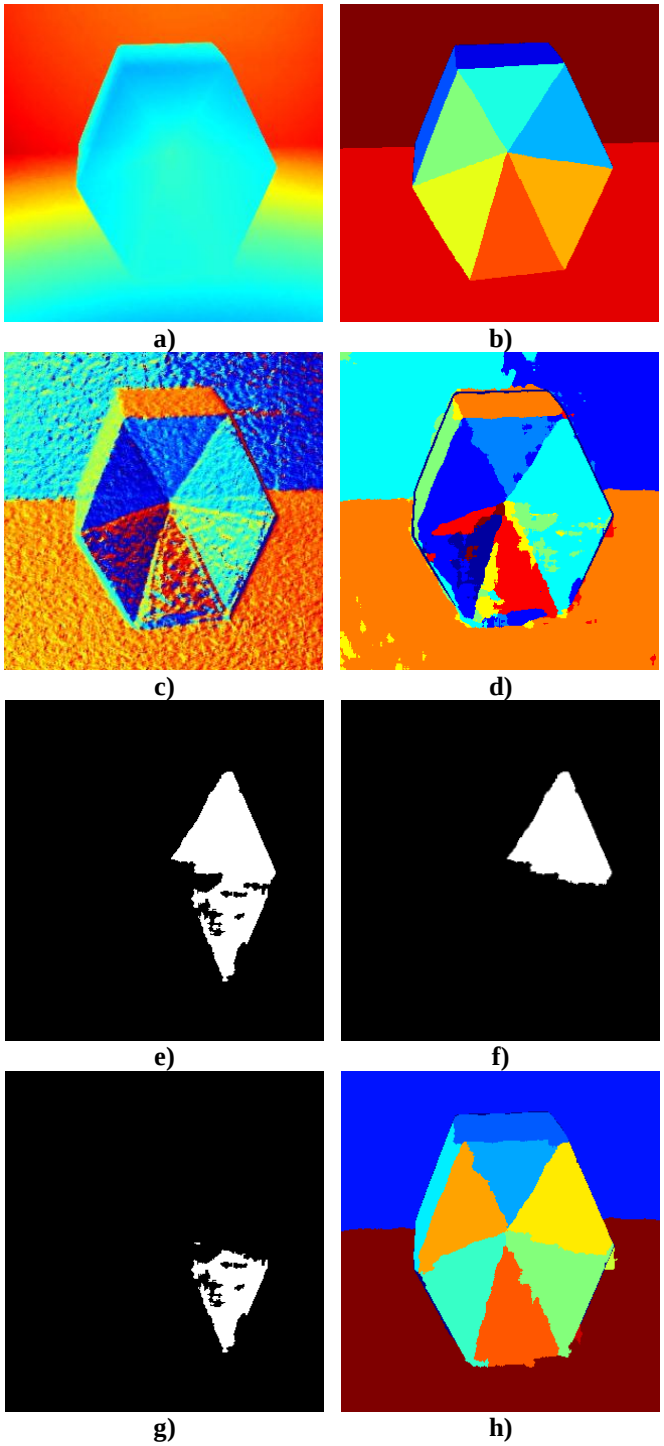


Figure 9 An example of the verification process on an image from the Perceptron dataset. a) The input range image; b) The ground truth image; c) The $DGoD$ feature map; d) The clustered image showing the regions on the right that are under-segmented; e) The under-segmented region; f) The inliers; g) The outliers; h) The final segmented image

Since the verification process is executed on every region in the input image and since processing of a region does not depend on the results of other regions, the verification process is linear and the processing time is $O(N)$ where N is the number of regions in the image. The output of this process is an image that contains set of segmented regions where each region represents a planar surface.

D. Merging Regions Process

Applying a verification process on the clustered image helps in solving the under-segmentation problem which could be resulted from the clustering process. However, it could happen, due to noise, that a real planar surface is over-segmented. Hence, a merging process is needed to solve any over-segmentation problem.

Two initial segmented regions, say R_I and R_J, are said to belong to the same planar surface if the following condition holds:

$$\left(\frac{\sum_{i=0}^{N} d(p_i, D_J) < T_{dis}}{N} > T_{tol}\right) \;\&\&\; \left(\cos^{-1}\frac{\hat{n}_I \cdot \hat{n}_J}{|\hat{n}_I||\hat{n}_J|} < T_{ang}\right) \quad (5)$$

where N is the number of points in the region R_I, D_J is the resulted 3D plane fitted on the region R_J, $d(p, D)$ is the function to compute the $3D$ distance of a point p to the plane D, T_{dis} is the threshold distance to consider the point p as belonging to the plane D, T_{tol} is the tolerance threshold to consider both regions R_I and R_J as belonging to one planar surface and $\hat{n}_I$ and $\hat{n}_J$ are the normals of the planar regions R_I and R_J respectively.

The first part of (5) is responsible to examine the distance between the all points in R_I and the 3D plane of R_J. Theoretically, if both regions R_I and R_J belong to the same planar surface, $d(p_i, D_J)$ should be near to Zero for each point in R_I. However, this is not the case in a real-world image where the distance could be up to 1 cm (depends on the depth error of the camera). Moreover, and due to noise, it is possible to have some points in region R_I that does not belong to the plane D_J, so that a tolerance threshold T_{tol} is proposed to overcome the problem of noisy points.

The second part of (5) is responsible to examine the angle between the both regions R_i and R_j. This condition is essential in preventing a small region with only a few points from being merged with another region where both regions do not belong to the same planar surface. In an ideal case, the angle between the two regions should be near to Zero if they both belong to the same planar surface. In a real case, however, the angle could be up to 20 degrees which depends on the depth error of the camera. Therefore, an angle threshold T_{ang} is used to overcome that problem. In this paper, the value of T_{tol} and T_{ang} are set to

0.85 and 25° respectively, The value T_{dis} depends on the depth error of the camera and should be manually set per camera type.

In addition to the condition described in (5), two regions are merged if they are both adjacent (connected in the 3D space). The *Adjacency Matrix* is used in this paper to describe the connectivity between different regions in the image. *Adjacency Matrix* is a matrix of size $M \times M$, each element in the matrix (i, j) represents the connectivity between the regions R_I and R_J. *Adjacency Matrix* is given by the following:

$$ADJ = \begin{bmatrix} adj_{(0,0)} & \cdots & adj_{(0,M)} \\ \vdots & \ddots & \vdots \\ adj_{(M,0)} & \cdots & adj_{(M,M)} \end{bmatrix} \quad (6)$$

$$adj_{(i,j)} = \begin{cases} 1 & if\ R_I \cap R_J \neq 0 \\ 0 & otherwise \end{cases} \quad (7)$$

where $R_I \cap R_J$ is the adjacency of the two regions R_I and R_J, M is number of regions in the initial segmented image. It is worth to mention that $adj_{(i,j)} = adj_{(j,i)}$ which makes the *Adjacency Matrix* symmetric ($ADJ = ADJ^T$), so that the complexity of the merging process during the connectivity test is reduced.

Merging Regions Process

// Inputs:

Initial Segmented Regions $\{R\}$, Adjacency Matrix ADJ

// Initialize:

Merging Cost Matrix $MRG \leftarrow INF$, Neighboring Regions List $\{R_c\} \leftarrow \emptyset$

// Recursive Merging Process

FOR $i = 1{:}\, length(\{R\})$

$R(i) \rightarrow R_i$

IF R_i is already merged, then continue to the next region

GET the adjacency regions from $ADJ \rightarrow \{R_c\}$

$length(\{R_c\}) \rightarrow N$

$1 \rightarrow j$

WHILE $j \leq N$

$R(j) \rightarrow R_j$

IF Eq. **(5)** holds

COMPUTE the merging cost according to Eq. **(9)** $\rightarrow cost$

GET the minimum merging cost for R_j from $MRG \rightarrow Cost_{min}$

IF $cost < cost_{min}$

UPDATE the adjacency list $\rightarrow \{RC\}$

UPDATE $length(\{R_c\}) \rightarrow N$

UPDATE the merging cost matrix $\rightarrow MRG$

SET $1 \rightarrow j$

MERGE R_j to R_i

ELSE

$j++$

Figure 10 Pseudo code of the merging regions process

During the merging process, it is possible to get one region as belonging to two different surfaces which could confuse the merging process. Such a case is normally seen by small regions located between two planar surfaces. Therefore, a *Merging Cost Matrix* is proposed to solve that problem. A *Merging Cost Matrix* is of size $M \times M$, each element in the *Merging Cost Matrix*, $cost_{(i,j)}$, represents the cost of merging region R_I with R_J. The merging cost matrix is given by the following:

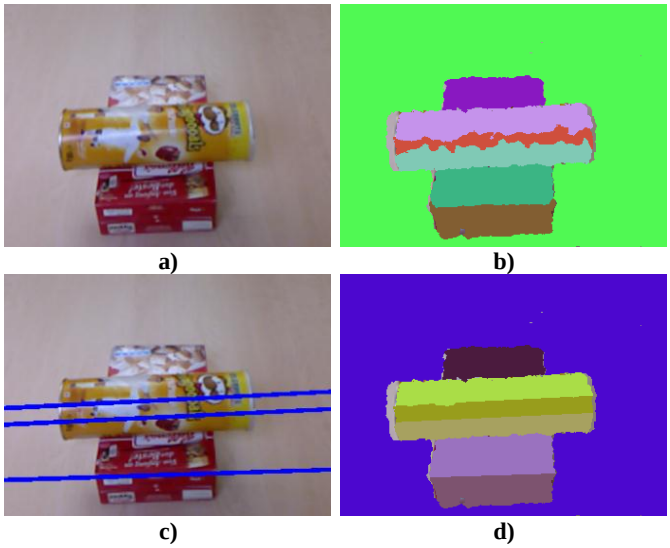
$$Merg\_Cost = \begin{bmatrix} cost_{(0,0)} & \cdots & cost_{(0,M)} \\ \vdots & \ddots & \vdots \\ cost_{(M,0)} & \cdots & cost_{(M,M)} \end{bmatrix} \quad (8)$$

$$cost_{(i,j)} = \begin{cases} RSS_{(I,J)}, & if\ (R_I, R_J) are\ merged \\ inf, & otherwise \end{cases} \quad (9)$$

$$RSS_{(I,J)} = \frac{\sum_{i=0}^{N} d(p_i, D_J)}{N} \quad (10)$$

If a region R_I is to be merged with two regions, say R_{J1} and R_{J2}, the *Merging Cost Matrix* is examined and the region R_I is merged to the region that produces the lowest merging cost. **Figure 10** shows a pseudo code of the merging process.

The merging process is built in a recursive manner; each initial segmented region is tested with all the neighboring connected regions. If two regions belong to one planar surface, then both are merged. This process is iterative and it is repeated until no more regions are merged. The processing time for this process is $O(N^2)$, where N is the number of regions in the input set.



a) **b)** **c)** **d)**

Figure 11 Example image from the OSD dataset [18] showing the result of the re-assignment step. a) The RGB input image; b) The segmented images resulted from the merging process; c) The intersection lines between the segmented planes; d) The result of post processing step

E. Post Processing

The post-processing step is responsible to add to the segmented image resulted from the merging process the unspecified points which are removed as having high $MGoD$ values and pixels defined as outliers in the verification process. Unspecified points that are inside the contour of a segmented

region are added to that region and points that lay on the boundary between two regions are assigned to the nearer region.

An additional step is added to the post-processing process which is aimed to fine tuning of the boundaries between the segmented regions to increase the segmentation accuracy. The additional step is similar to what has been used in the UE method presented in [12]. The intersection line of two adjacent planar regions is computed and re-projected into the 2D image space. The boundary pixels between the two regions are then re-assigned based on the position from the intersection line.

Suppose the line $L_{I,J}$ is the 2D projected line resulted from the intersection of both regions R_I and R_J in the 3D space. Pixels in region R_I located on the opposite side of the line $L_{I,J}$ from the centroid of R_I are removed from R_I and re-assigned to the region R_J. The same process is executed on the region R_J.

The experiments show that the implementation of the re-assignment step increases the quality of the segmentation, especially in the case of segmentation of the planar regions from cylindrical and curved surfaces as seen in **Figure 11**. Note that the re-assignment step is more evident on cylindrical objects compared to cuboid objects.

The processing time in this process is $O(N^2)$ where N is the number of regions in the image.

IV. Performance Evaluation

Three different sets of experiments were conducted and the results are presented in this section. The first set of experiments evaluates the performance of the proposed algorithm on both the ABW dataset and Perceptron dataset. It also compares the results of the GoD-based algorithm to other state-of-the-art algorithms. The second set of experiments evaluates the robustness of the proposed algorithm to the change of the clustering parameters (the number of clusters in the clustering histogram (m) and the clustering window size (n)). The third set of experiments compares the results of the proposed algorithm with other algorithms applied on depth images computed from stereo camera of a typical top-table scenario, which include planar and cylindrical objects.

Each image in the Perceptron dataset [11] and ABW dataset [10] contains up to five polyhedral objects placed on a supported horizontal plane. The datasets were randomly divided into two sub-sets: 10 images for training and 30 images for testing. **Figure 12** shows the results of the GoD-based algorithm on two images, one from the ABW dataset and one from the Perceptron dataset. The GoD-based algorithm is evaluated on the test images of the two datasets using set of parameters that are tuned manually to give the best results. The set of parameters are ($n = 15, m = 17, T_{dis} = 3$) for the ABW and ($n = 15, m = 9, T_{dis} = 4$) for the Perceptron.

In order to evaluate the performance of the proposed algorithm, two different metrics are used. The first metric evaluates the algorithm on the pixel level. Let R_n be the ground truth region and S_m is the corresponding segmented region from the proposed algorithm. Let $TP_i = R_n \cap S_m$ be the overlapping pixels (True Positive), $FP_i = S_m \backslash TP_i$ and $FN_i = R_n \backslash TP_i$ are the False Positive and False Negative pixels respectively. Then,

$$TPR = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{R_i}, FPR = \frac{1}{n}\sum_{i=1}^{n}\frac{FP_i}{S_i}, \quad (11)$$

are the True Positive Rate (sensitivity) and False Positive Rate (1-precision) respectively, n is the number of surfaces in each image.

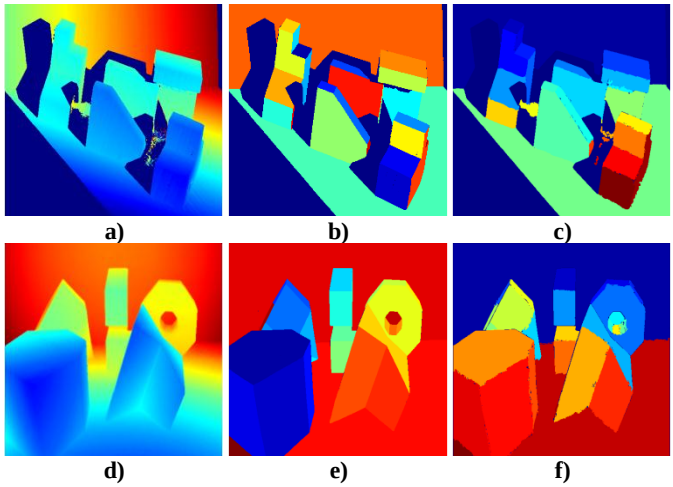


Figure 12 Two different images from both the ABW dataset (top) and Perceptron dataset (bottom). a),d) The input range image; b), e) The ground truth image; c), f) The final segmented image using the GoD-based algorithm

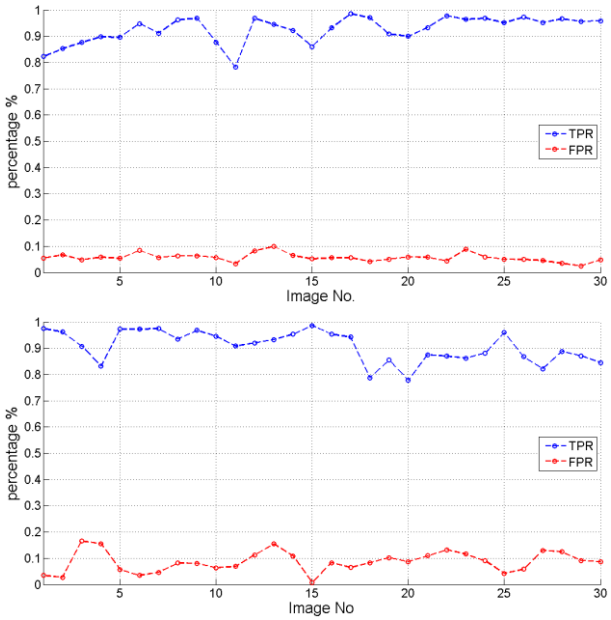


Figure 13: The True Positive Rate (TPR) and the False Positive Rate (FPR) for each image in the ABW dataset (top) and for each image in the Perceptron dataset (bottom) using the GoD-based algorithm

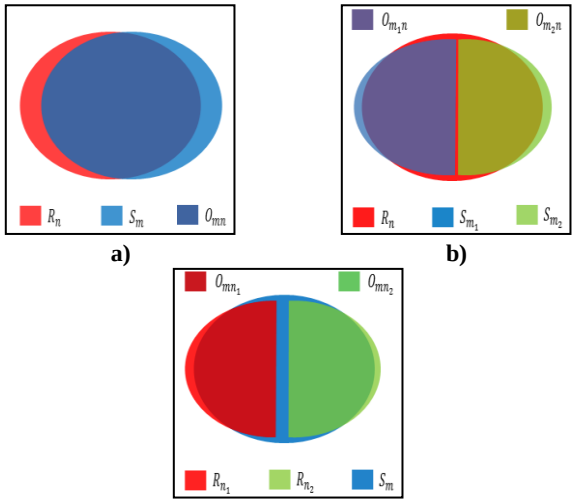
Figure 13 shows the result of the GoD-based algorithm on each image in both datasets using the described metric. As evident, the proposed algorithm managed to segment correctly in average 92.6% of pixels in the ABW and 90.6% of the pixels in the Perceptron dataset. The average False Positive Rate is 5.6% and 8.6% in the ABW and Perceptron datasets respectively.

The second metric used to measure the performance of the proposed algorithm is described in [12]. There are five types of segmentation outputs that have been considered: correct segmentation, over-segmentation, under-segmentation, missed and noise regions.

Let R_n be the area (the number of pixels) of the manual segmented region in the ground truth image and S_m is the area of the corresponding segmented region resulted from the used segmentation algorithm. Let $O_{mn} = R_n \cap S_m$ represents the area of the intersection of both regions R_n and S_m. O_{mn}/S_m represents the percentage of the intersection region O_{mn} with respect to the segmented region S_m. O_{mn}/R_n represents the percentage of the intersection region O_{mn} with respect to the ground truth region R_n. An object is called to be correct segmented if and only if

$$\frac{O_{mn}}{S_m} \geq T \;\&\; \frac{O_{mn}}{R_n} \geq T \qquad (12)$$

where T is the percentage threshold. A lower value of T will relax the definition of a correct-segmented object.



a) **b)**

c)

Figure 14 a) Correct segmentation; b) Over segmentation; c) Under segmentation

An instance of over-segmented object is defined when a region in the ground truth image is segmented into a set of regions $\{S_{m1}, S_{m2}, \ldots S_{mx}\}$ in the segmented image fulfilling the following condition:

$$\frac{O_{m_in}}{S_{m_i}} \geq T \;(\forall i \in x) \;\&\; \frac{\sum_{i=1}^{x} O_{m_in}}{R_n} \geq T \qquad (13)$$

where O_{m_in} is the intersection between a region S_{m_i} and the corresponding ground truth region R_n. x is the number of regions resulted from the segmentation algorithm where the union of them makes the corresponding ground truth region R_n. Similarly, an instance of under segmentation is defined when a set of regions $\{R_{n1}, R_{n2}, \ldots . R_{nx}\}$ in the ground truth image are merged into one region in the segmented image as follows:

$$\frac{\sum_{i=1}^{x} O_{mn_i}}{S_m} \geq T \;\&\; \frac{O_{mn_i}}{R_{n_i}} \geq T \;(\forall i \in x) \qquad (14)$$

A missed region is defined when a ground truth region R_n does not participate in any instance of correct-segmentation, over-segmentation or under-segmentation while a noise region is defined when a region in the segmented images does not have any correspondence in the ground truth image. **Figure 14** shows an example of the correct segmentation, over segmentation and under segmentation.

Figure 15 shows the results of the *GoD*-based algorithm using the second metric on the ABW and the Perceptron datasets through the whole threshold value range ($0.51 \leq T \leq 1$).

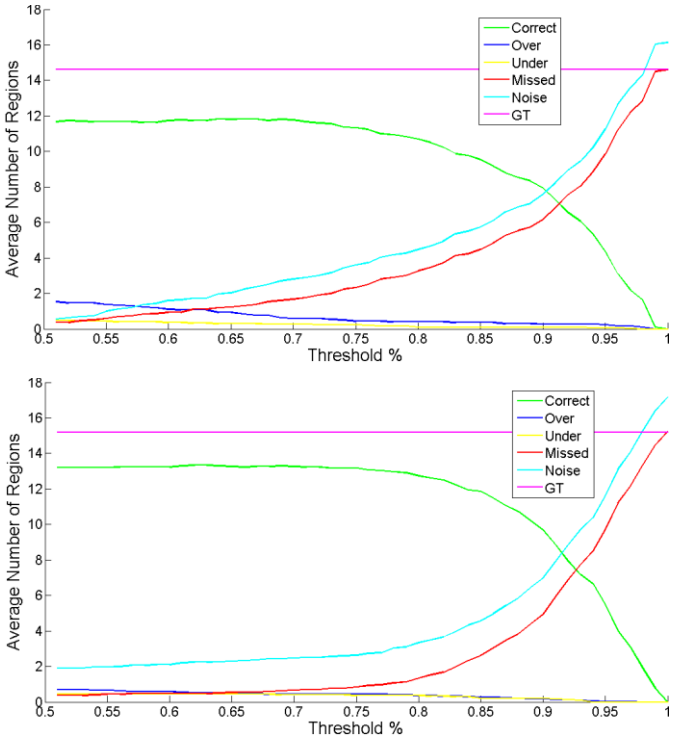


Figure 15 The GoD-based algorithm results on both the Perceptron dataset (top) and ABW dataset (bottom) as described by the second metric. GT in the figures is the ground truth value for the correct segmented regions

TABLE I and **TABLE II** describe the performance of the GoD-based algorithm in comparison to other algorithms that have been evaluated on the same datasets. The results are compared at the threshold level $T = 80\%$. The algorithms used for the comparison are: USF (surface growing algorithm [12]), WSU (principal components clustering [12]), UB (region growing using scan line segmentation [12]), UE (Gaussian and mean curvature clustering [12]), EG (edge detection based on scan line approximation [7]) and ML (Multi-Resolution Segmentation [4]).

On the Perceptron dataset, the GoD-based algorithm ranks as first in number of the correct segmented regions and as second in both the under-segmented and missed regions. On the ABW dataset, the proposed algorithm ranks as first in the under-segmented regions and as second in the over-segmented regions but comes slightly behind at the third place in terms of correct segmented regions.

TABLE I Comparison of different algorithms on Perceptron dataset at threshold value 80'% (x: not available)

Method	Ground Truth	Correct Seg.	Over Seg.	Under Seg.	Missed	Noise
USF	14.6	8.9	0.4	0.0	5.3	3.6
WSF	14.6	5.9	0.5	0.6	6.7	4.8
UB	14.6	9.6	0.6	0.1	4.2	2.8
UE	14.6	10.0	0.2	0.3	3.8	2.1
EG	14.6	10.6	0.1	0.2	3.4	1.9
ML	x	x	x	x	x	x
GoD	14.6	10.7	0.4	0.1	3.6	4.4

TABLE II Comparison of different algorithms on ABW dataset at threshold value 80'%

Method	Ground Truth	Correct Seg.	Over Seg.	Under Seg.	Missed	Noise
USF	15.2	12.7	0.2	0.1	2.1	1.2
WSF	15.2	9.7	0.5	0.2	4.5	2.2
UB	15.2	12.8	0.5	0.1	1.7	2.1
UE	15.2	13.4	0.4	0.2	1.1	0.8
EG	15.2	13.5	0.2	0.0	1.5	0.8
ML	15.2	11.1	0.2	0.7	2.2	0.8
GoD	15.2	13.2	0.3	0.2	1.1	1.8

The second set of experiments is used to validate the robustness of the GoD-based algorithm regarding the change of the two parameters (n and m) in the clustering process. The proposed algorithm was run three times using three different parameters set on the ABW dataset. The parameters in the first run were ($n = 15$, $m = 17$), in the second run were ($n = 15$, $m = 9$) and in the third run were ($n = 7$, $m = 9$). As demonstrated by the results shown in **Figure 16**, the GoD-based algorithm shows slight change in the number of correct segmented regions between the three runs which proofs the robustness of the proposed algorithm.

In the third experiment set, the proposed algorithm is compared against two different algorithms applied onto depth images generated from stereo vision. The two algorithms used for the comparison are the scan line segmentation algorithm and the planar segmentation function implemented in PCL [13]. Scan line segmentation algorithms is implemented by the authors of this work using 2D line model. The goal of this experiment is to examine the ability of other state-of-the-art algorithms for segmentation of planar regions of simple cuboids as well as of non-planar objects on depth images computed from stereo camera which have a low depth accuracy compared to range Images generated from laser scanners or structure light cameras.

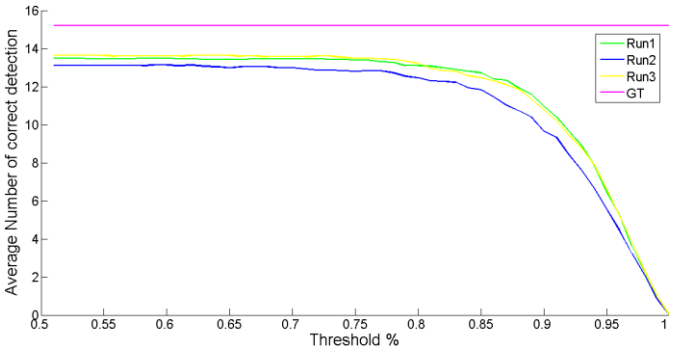


Figure 16 The number of correct segmented regions by the GoD-based algorithm on ABW dataset using three different sets of parameters

Figure 17 shows three different scenes and their corresponding depth images. The first scene includes a simple cuboid object while the second scene includes two bottles one partially occluding the other. The third scene contains cuboids objects together with cylindrical object. Block matching algorithm [19] is used to compute stereo correspondence between left and right stereo images.

Although the scan line segmentation algorithm scored good results on both the ABW and Perceptron datasets, the results on depth images computed from stereo camera show a lot of noise edges which makes it hard to extract close contours. That conclusion is supported also by the results presented in [9] where the quality of the segmentation is low due to noise in range images. PCL implementation on the other hand shows good results in segmenting planar regions from cuboid objects, but failed to segment planar regions from cylindrical objects even after trying to tune manually the parameters. On the other hand, the proposed GoD-based algorithm shows good results in segmenting planar regions from both cuboid and cylindrical objects.

TABLE III shows the average processing time of each process in the GoD-based algorithm on the images shown in **Figure 17**. The processing time of GoD-based algorithm has been computed using sequential process in C++/Windows. The average time needed to compute the GoD feature on an image of the size (1024×768) was 30 ms. Compared to other features known in literature, the fastest time to compute the local surface normal was 4 ms [6] using the PCL on an image of the size 160×120 and a neighborhood size of 3×3. It is worth to mention that PCL is hardware accelerated (GPU acceleration) library

while the proposed GoD feature has been computed using a sequential operation.

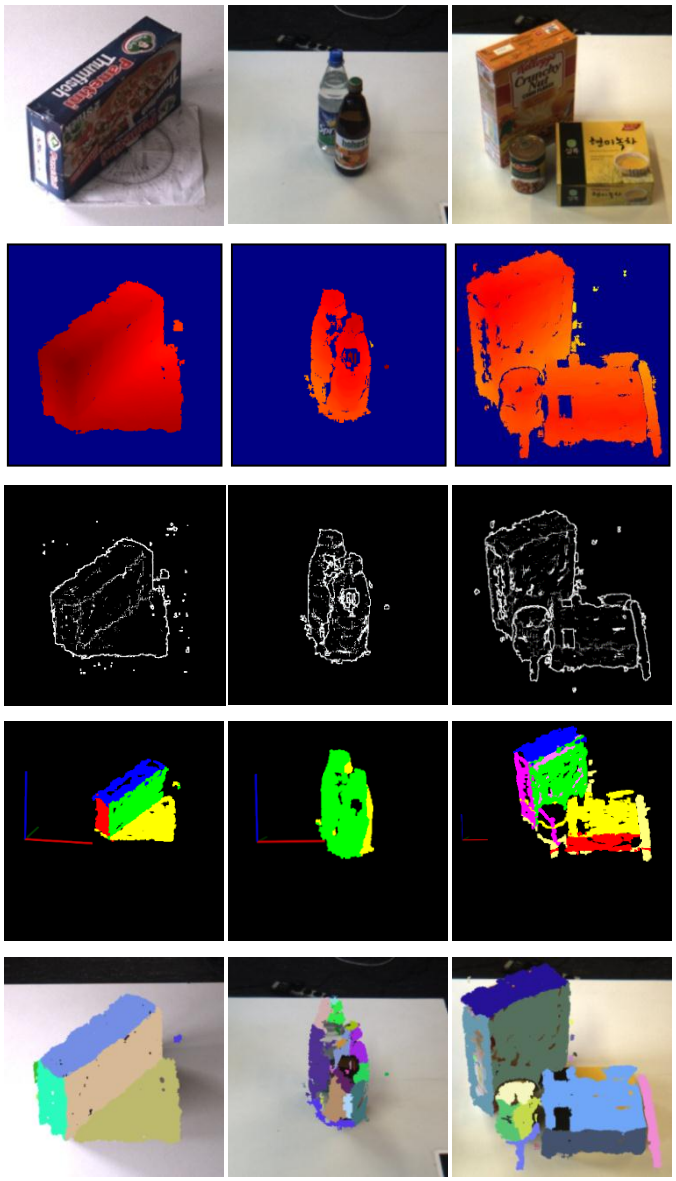


Figure 17 Comparison of different algorithms using depth images computed from stereo camera. First row: left stereo input images (cropped); Second row: the computed depth images; Third row: scan line based segmentation algorithm; Forth row: planar segmentation using PCL library; Last row: planar segmentation using the proposed GoD-based segmentation

The computation of the GoD feature and the total processing time of the proposed algorithm could be accelerated using multi-thread process. Computing the GoD feature and the clustering process could be parallelized since both operations are pixel-wise operation and the processing time would be $O\left(\frac{N}{P}\right)$ where N is the number of pixels in the image and P is the number of threads used for the process. The verification process and the post-processing process could be also parallelized using multi-thread process since both of them are region-wise operations and the processing of one region does not depend on the results of other regions. Hence, the processing time for the verification process and post-process would be $O\left(\frac{N}{P}\right)$ and $O\left(\frac{N^2}{P}\right)$ respectively where N is the number of regions in the image and P is the number of threads used for the process. However, the bottleneck in the processing time of the proposed algorithm is the merging process. The current implementation of the merging process does not allow for a parallelization process. The processing time of the merging process depends highly on two factors: the number of initial segmented regions resulted from the clustering and verification processes and the number of the ground truth regions in the range image. In the case where the input range image is noisy (like the Perceptron range images), the number of the initial segmented regions will be high so that larger processing time is needed for the merging process due to its iterative nature.

TABLE III The average processing time of the processing steps pf the GoD-based algorithm applied on the images shown in Figure 17

Process	GoD	Clust.	Ver.	Merg.	Post
Average Time (s)	0.03	0.7	1.2	2.5	0.3

V. CONCLUDING REMARKS

In this paper, a new algorithm for segmentation of planar regions from depth images is presented. The new algorithm is based on the novel Gradient of Depth feature (GoD), accordingly, the proposed algorithm is to be called GoD-based planar segmentation algorithm. In contrast to the local surface normal, the GoD feature is computed in the 2D image space (i.e. directly on the depth image while the local surface normal is computed in the 3D point cloud space). Hence, the GoD feature is faster to compute and less complex than the local surface normal which is considered as the state-of-the-art feature. The feature space of the proposed GoD feature is 1D which makes it convenient for clustering algorithms. As evidenced by the presented results, the presented GoD-based algorithm is robust to parameters changes and produces accurate results thanks to both verification and merging processes. In terms of segmentation accuracy, the GoD-based algorithm meets the performance of other state-of-the-art algorithms and in some cases, it outperforms them. It is even able to segment planar regions from depth images computed from stereo camera which are known for their high depth error on which other state of art algorithms fails.

Accordingly to the presented performances, the GoD-based algorithm could be utilized in robot vision to segment planar regions from different surfaces (planar, cylindrical and curved) in different user support scenarios using different camera types. As prove of the robustness of the algorithm in segmenting planar regions from different cameras in different scenarios; the GoD-based algorithm has been implemented in the assistive robotic system "FRIEND" to develop a stereo-based book segmentation algorithm in library scenario [2].

A limitation of the presented algorithm is the processing time in the merging process. The current implementation of the merging process does not allow for a parallelization process. Therefore, the authors are already exploring and evaluating other implementations for the merging process which allow the parallelization of the process. It is also planned to parallelize the processing of the algorithm to bring it to the real-time processing.

REFERENCES

[1] "Care-O-bot", [Online]. VIEW ITEM

[2] A. Gräser, T. Heyer, L. Fotoohi, U. Lange, H. Kampe, B. Enjarini, S. Heyer, C. Fragkopoulos and D. Ristic-Durrant, "A Supportive FRIEND at Work: Robotic Workplace Assistance for the Disabled", *IEEE Robotics and Automation Magazine*, To be published.

[3] A. Bartoli, "Piecewise Planar Segmentation for Automatic Scene Modeling", in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2011.

[4] B. Oehler, J. Stueckler, J. Well, D. Schulz and S. Behnke, "Efficient Multi-Resolution Plane Segmentation", in *In Proceedings of the 4th International Conference on Intelligent Robotics and Applications (ICIRA)*, 2011. CrossRef

[5] T. R. Shah, "Automatic Reconstruction of Industrial Installations", *Publications on Geodesy* 62, 2006.

[6] D. Holz, S. Holzer, R. B. Rusu and S. Behnke, "Real-Time Plane Segmentation using RGB-D Cameras", in *RoboCup Symposium*, 2011.

[7] X. Jiang and H. Bunke, "Edge Detection in Range Images Based on Scan Line Approximation", *Computer Vision and Image Understanding*, Bd. 73, Nr. 2, p. 183–199, 1999. CrossRef

[8] X. Jiang and H. Bunke, "Range Image Segmentation: Adaptive Grouping of Edges into Regions", in *Computer Vision — ACCV'98, Lecture Notes in Computer Science*, 1997. CrossRef

[9] A. Sabov and J. Krüger, "Segmentation of 3D Points from Range Camera Data using Scanlines", in *15th International Conference on Systems, Signals and Image Processing*, 2008.

[10] ABW, "ABW dataset", [Online]. VIEW ITEM

[11] Perceptron, "Perceptron dataset", [Online]. VIEW ITEM

[12] A. Hoover, G. J. Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon and R. B. Fisher, "An experimental comparison of range image segmentation algorithms", *IEEE Transactions on pattern analysis and machine intelligence*, 1996. CrossRef

[13] PCL, "Point Cloud Library", [Online]. VIEW ITEM

[14] G. M. Hegde and C. Ye, "A Recursive Planar Feature Extraction Method for 3D Range Data Segmentation," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2011.

[15] M. A. Fischer and R. C. Bolles, "Random sample consensus:A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, Bd. 24, Nr. 6, pp. 381-395, 1981. CrossRef

[16] F. Alhwarin, D. Ristic-Durrant and A. Gräser, "VF-SIFT: Very Fast SIFT feature matching", in *Pattern Recognition, Lecture Notes in Computer Science*, 2010. CrossRef

[17] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[18] OSD, "Object Segmentation Database (OSD)", [Online]. VIEW ITEM

[19] T. Tao, J. C. Koo and H. R. Choi, "A fast block matching algorithm for stereo correspondence", *IEEE Conference on Cybernetics and Intelligent Systems*, 2008.