Detection of top-planes in cuboids from RGB-D images of packing scenes

Guillermo Camacho-Munoz, *Member, IEEE,*

***Abstract*—This paper presents a top-plane extraction algorithm suitable for packing scenes: cluttered scenes with multiple boxes, each one with different spatial properties (positions, orientations and scales) and under partial occlusion conditions. The main contribution of the paper is the set of criteria to select planes that compose boxes in the scene. The algorithm was designed as a component of a pipeline that resolves the problems of detection, segmentation and pose estimation of boxes from point clouds. The performance of the algorithm is assessed in terms of four indicators: (1) existing planes detected -DP, (2) number of existing planes that were detected multiple times – MD, (3) number of missing planes -MP , (4) number of spurious planes detected -SP.**

***Keywords*—plane extraction, cuboid detection, cuboid segmentation, cuboid pose estimation, packing applications.**

I. Introduction

A packing system is a system used in logistic operations to resolve two main tasks: (1) compute a loading pattern and (2) assembly of the pattern in the container. The first task assigns the cargo to a container, optimizing a cost function subject to a set of constraints [1]. A solution that satisfies those conditions is called loading-pattern. In the second task (usually performed by an operator), the cargo is loaded to the container until complete the loading-pattern. The operation requires converting the loading pattern into a physical packing sequence (PPS): the sequence by which each box is placed inside the container in a specific location [11]. The communication of the PPS (to the operator) with conventional interfaces have shown low effectiveness. An opportunity to improve this interface is identified in the Augmented Reality (AR) systems by its feature of increment the visual perception of the user with virtual objects overlaying the real scene. Those virtual objects can aid the placing of cargo inside the container.

There exists multiple problems to solve in this kind of interface [2], one of them is locating all instances of boxes in an scene represented by a point of clouds. The literature in computer vision [20] identifies two approaches to resolve the detection problem. From those approaches this paper focus on the one called 'recognition by components' and we assume that top-planes are the more suitable components of a box in a packing application. Then we focus on the design of a top-plane extraction algorithm suitable for packing scenes: cluttered scenes with multiple boxes, each one with different spatial properties (positions, orientations and scales) and under partial occlusion conditions.

M. Camacho-Munoz is with the Department of Electrical and Electronic Engineering, University of Valle, Cali, Calle 13 # 100-00, Colombia e-mail: guillermo.camacho@correounivalle.edu.co

Manuscript received April 19, 2020 revised January 11, 2021

II. Basic Concepts

We have selected three main concepts to understand the proposed algorithm: Model of a plane, Algorithms reported to extract planes from point clouds and M-estimator SAmple Consensus (MSAC)

A. Model of a plane

The planes of interest in this paper are three-dimensional surfaces that can be represented by a set of four parameters: $ABCD$. The parameters A, B, C represent the components of the normal vector of the plane: a vector orthogonal to the plane that indicates its inclination. The last parameter (D) represents the distance that the plane must be translated in the orientation of the normal to cross the origin. The scalar equation of plane is presented in equation 1

$$Ax + By + Cz = D \tag{1}$$

B. Algorithms reported to extract planes from point clouds

Algorithms proposed for plane extraction from 3D point clouds can be classified in two groups: (1) RANSAC based, (2) Hough Transform based.

The Hough transform used conventionally to detect lines [13], has been adapted for the detection of basic geometric shapes, including the detection of planes. The most recent adaptation is reported in [18] were they propose a three stage pipeline to detect planes from depth images: (a) clustering, (b) voting, and (c) peak detection. They compare the performance of this technique with a plane extractor based on RANSAC over a dataset composed by real and synthetic data. The reported results allow to conclude a superior performance of their algorithm in terms of (1) detected planes, (2) multiple detection of a planes, (3) missing planes, and (4) spurious planes detection. The source code of this propose is available at [17], however this code exhibits some bugs that to the date of this report were not resolved at the official repository.

The RANSAC algorithm is an iterative and non-deterministic method to estimate parameters of a mathematical model [4], this method has been adapted to search for the best plane model in a 3D point cloud. Methods based on RANSAC usually follow the paradigm of iterative applying RANSAC algorithm on the data while removing inliers corresponding to the currently found plane instance. The comparison reported in [14] allow to conclude a superior performance of RANSAC

TABLE I: Relation between the side and distance of camera for each scene

Side/Distance	d1		d2		d3	
Right	s13	s15	s3	s20	s1	s6
	s34	s35	s21	s31	s7	s29
Center	s8	s10	s11	s16	s2	s4
	s12	s28	s24	s26	s18	s36
Left	s19	s22	s9	s14	**s5**	s23
	s25	s32	s17	s27	s30	s33

implementations in terms of computation time and quality. There exists multiple implementations of this algorithm in academic platforms like $RANSAC\_SAC$ [10] in the Point Cloud Library and MSAC in Matlab [9]. In consequence we will use one of the available RANSAC implementations as an initial point in the resolving of our problem

C. MSAC algorithm to detect a plane

MSAC is a generalization of the RANSAC estimator that adopts the same sampling strategy as RANSAC but chooses the solution to maximize the likelihood rather than just the number of inliers [16], [15]. We have used the implementation of MSAC available in the Matlab environment under the function $pcfitplane$ [9]. This function allows to fit a plane to a point cloud that has a maximum allowable distance from an inlier point to the plane. The function returns a geometrical model that describes the plane.

III. The Dataset

The dataset used for validation of our algorithm was initially presented in [3] and is comprised of point clouds acquired by Microsoft Kinect v2 sensor and annotations suitable to the problems of detection of boxes. It features 36 scenes, each one with controlled environment: fixed number of boxes ($n_b = 4$), random distribution of boxes over a limited area, non-stacked boxes and low level of occlusion. Each scene was sample from a single point of view and with four samples. The dataset used a total of nine different poses of camera with four scenes by each pose as presented in Table I.

The annotation for each scene is composed by: identifiers of each box, box size measured by hand, position of grasp point of the box in a world frame (O_0) and azimuth orientation of the box (angle between the x axis of frame O_0 and the depth vertex of the box) as depicted in Figure 1 and Table II.

We have complemented this database with the computation of a transformation matrix that allow to align the projection of annotations with the point cloud acquired by the sensor. A sample of this align is presented in Figure 1. There are nine different transformation matrix for each of the relation side/distance in Table I.

IV. Methodology

The pipeline proposed for the construction of a map of planes which belongs to the top side of a box, from point clouds is presented in Figure 2.

TABLE II: Annotations Sample of database. Note that Depth have been defined to be less or equal than the Width

scene	boxID	Height	Width	Depth	Azimuth Angle
1	1	0,25	0,1	0,1	26
	7	0,25	0,1	0,1	141
	12	0,252	0,149	0,1	64
	13	0,252	0,15	0,099	171
6	1	0,25	0,1	0,1	110
	8	0,15	0,149	0,1	104
	12	0,252	0,149	0,1	24
	32	0,1	0,15	0,1	59
7	7	0,25	0,1	0,1	5
	8	0,15	0,149	0,1	91
	13	0,252	0,15	0,099	68
	33	0,15	0,1	0,099	97
29	1	0,25	0,1	0,1	52
	2	0,3	0,15	0,099	7
	20	0,15	0,3	0,2	175
	32	0,1	0,15	0,1	149

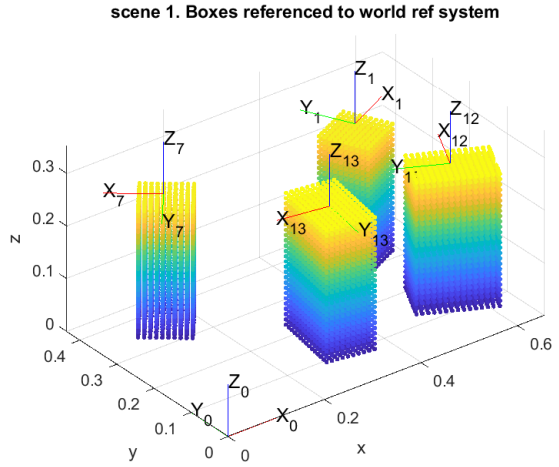


Fig. 1: Projection of annotations in scene one. The boxes are referenced to the world frame O_0

A. Preprocessing

This stage applies three operations on the raw point-cloud: (1) a merge of the n_b available samples of the scene, (2) denoise on the merged point cloud, and (3) extraction of a region of interest (ROI) in the scene. The first operation is applied to increment the density of points in the image with the four available samples by scene in the database. The second operation removes outlier points based on a distance metric. The last operation eliminates samples of objects different of boxes and structural elements of the scene. A sample of an output of this stage is presented in Figure 4.

B. Fitting Planes

The input to this stage is composed by the point cloud pc_{raw} and a threshold $th_{inliers}$ which allow to define the maximum

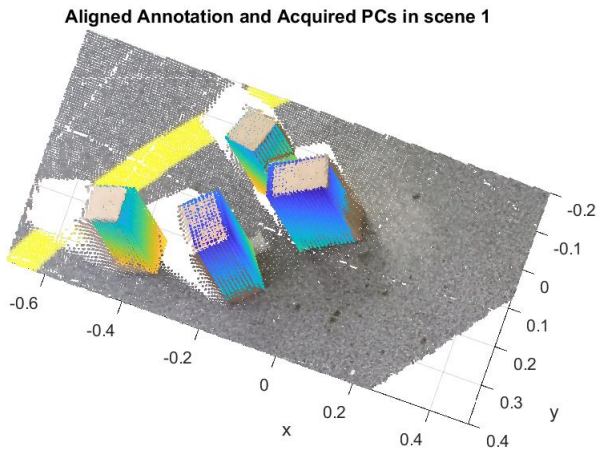


Fig. 2: Point Clouds of sensor and Annotations alligned. The z axis is entering the paper

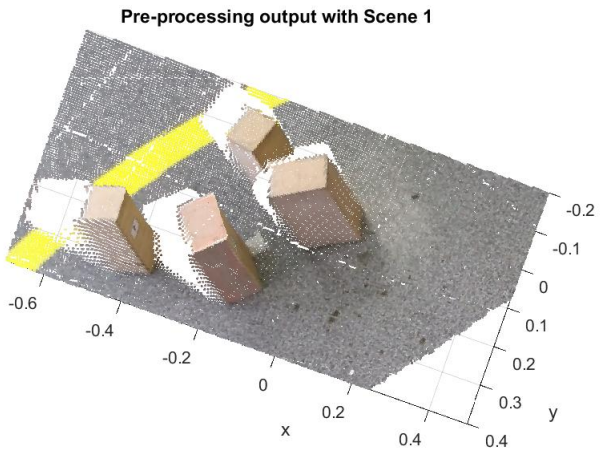


Fig. 4: scene 5. Output of Pre-processing stage as a colored Point Cloud

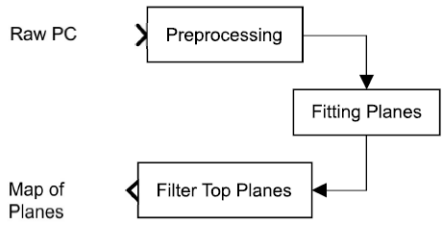


Fig. 3: Pipeline to extraction of Map of planes

distance from an inlier point to the computed plane. At the output we find a set of plane models $modelSet$ with a vector of indexes for each model $inliersPCraw$; those indexes are referenced to the input pc_{raw}.

The computation of plane models is described in algorithm 1. The process is based on the MSAC implementation described in Section II with an iterative strategy that stops when a predefined percentage of points ($w = (97.5\%)$) in pc_{raw} have been assigned to a model.

There are five parameters to set in this algorithm: (1) normal of reference n_{ref}, (2) threshold of angle with the normal of reference th_{angle}, (3) distance to classify a point as inlier or outlier of the fitted model $th_{inliers}$, (4) maximum number of random trials and (5) Confidence percentage for finding maximum number of inliers.

We assume that all the boxes in the packing application relies on a surface parallel to the ground plane, then, the normal of reference is set as the normal of the ground plane $n_{ref} = n_{ground}$ with a low tolerance $th_{angle}(= 5)$; n_{ground} is computed as the first predicted model of a low parametrized fitting model based on RANSAC. For the third parameter we use as reference the resolution of the sensor which is function of the distance to the boxes (1.3-1.8mt for the used database) and we estimate this value on 0.001 mt based on the experiments presented in [19]. The maximum number of random trials are fixed to 1000 and the confidence percentage is fixed in 99.

This phase generates two outputs: (1) a vector $modelSet$ with the parameters of each detected model (A_i, B_i, C_i, D_i) for $i = 1, ...N_{planes}$. The second output is a cell of inliers for each model, those inliers are saved as indexes of the original pc_{raw}.

C. Filter top Planes

Taking the output of Algorithm 1 as input, we apply a set of criteria to select exclusively the planes that belong to the top surface of a box. Those criteria are based on assumptions over the packing scene and on the available descriptors of boxes that are common in packing operations.

- Filter ground plane and its duplicates. This plane is detected as the plane with more inliers in the scene or the plane with more area. Although this model is filtered, the parameters of the model (A_g, B_g, C_g, D_g) are stored cause they are necessary to define other filter criteria. To remove the duplicates of planes we compare the parameter D_g of the ground model with the parameter D_i of the i- model. If ($(abs(D_g) - abs(D_i)) < th_D$) then the current model is filtered. The threshold th_D is defined as the minimum height of box expected in scene (view Table II).
- Remove planes with area major than a threshold $th_{maxArea}$ (=1 mt). This threshold is defined by the expected size of boxes in the scene (view Table II). Note that for scenes with multiple boxes with the same height (as the one depicted in Figure 1), there exist a probability

Algorithm 1: Fitting Planes

Function FittingPlanes(**pc** pc_{raw}; **double** $th_{inliers}$)
Result: $modelSet$, $inliersPC_{raw}$
$th_{angle} = 5$;
n_{ground} = pcfitplane(pc_{raw},$th_{inliers}$);
n_{ref}=n_{ground};
$initSize$ = $pc_{raw}.Count$;
pc = pc_{raw};
$i = 1$;
while $pc.Count$ >*(1-w)*initSize* **do**
  [model inliers outliers]=
   pcfitplane(pc,$th_{inliers}$,n_{ref}, th_{angle});
  $modelSet(i)$=model;
  if $i == 1$ **then**
    $inliersPC_{raw}(i)$=inliers;
  else
    $pc_{inliers}$=extractPoints(pc, inliers);
    $inliersPC_{raw}(i)$=computeIndex($pc_{raw}, pc_{inliers}$);
  end
  pc=extractPoints(pc, outliers);
  $i = i + 1$;
end

to detect a single plane for all the boxes. This plane must be accepted at this stage and splitted in a posterior stage.

- Remove planes with area minor than a threshold $th_{minArea}$. This threshold is computed from the minimum area expected in the scene.
- Split planes with area major than the maximal area expected in the scene. Some of the detected planes have inliers that correspond to points of k objects of the scene (with $k > 1$). This situation requires to design a process that split the plane in k subsets and defines if all the subsets correspond to real or spurious planes. We have designed a cluster process based on the Mean Shift Algorithm [5]. The selection of this algorithm is motivated by its feature of find the number of clusters, which allow to handle variant values for k. To set this algorithm, we assume that the maximum distance between boxes is 5 cm, then the bandwidth was set in 0.05. In this stage we define an additional threshold to reject clusters with number of inliers $< th_{N_{points}} = 49$. This threshold was set experimentally and depends on the distance between the camera and the object.

A sample of the final map of planes after applying those criteria is presented in Figure 5. This figure presents the output for a single scene with two values of $th_{inliers}$. Note the dependence between the $th_{inliers}$ and the number of detected planes/inliers: a lower $th_{inliers}$ is associated with more planes but less inliers.

V. RESULTS

The results are presented in terms of five indicators: (1) existing planes detected -DP, (2) number of existing planes that were detected multiple times – MD, (3) number of missing planes -MP , (4) number of spurious planes detected -SP and (5) number of planes in the detected map of planes. To compute those indicators is necessary to define a function of dissimilitude between a ground-truth plane p_{gt} and a plane that belongs to the map of planes detected by our algorithm p_i. We have formulated a function e_{diss} with three components: d_{area} a difference between areas of the planes, d_{or} a difference between orientation of the planes and d_{dist} the distance between the mean points at each plane. To compute d_{dist} we use the criterion in equation 3, where $dist_{euc}$ is the euclidean distance between the mean points at each plane and $depth_{gt}$ is the depth of the plane as defined in Table II.

$$e_{diss}(p_{gt}, p_i) = d_{area}(p_{gt}, p_i) + d_{or}(p_{gt}, p_i) + d_{dist}(p_{gt}, p_i) \quad (2)$$

$$d_{dist}(p_{gt}, p_i) = \begin{cases} 1, dist_{euc}(p_{gt}, p_i) \geq 0.5 depth_{gt} \\ 0, otherwise \end{cases} \quad (3)$$

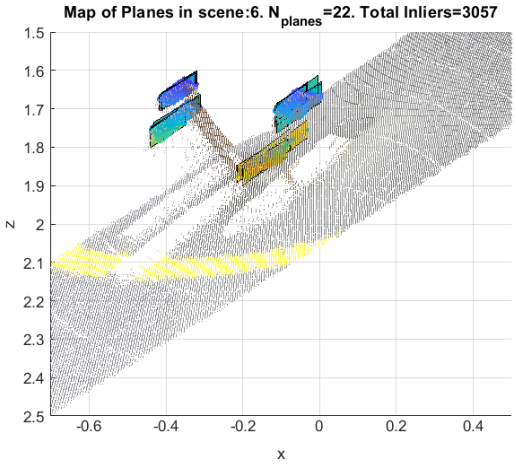
The function of dissimilitude is computed as presented in equation 2. A predicted model p_i is considered correct w.r.t. the ground-truth plane p_{gt} if this function $e_{diss} < th_{diss}(= 0.2)$. Some samples of this function are presented in Figure 6. With this function we have computed the results presented in Tables II and III.

TABLE III: Results by scene using a $th_{inliers} = 0.001$

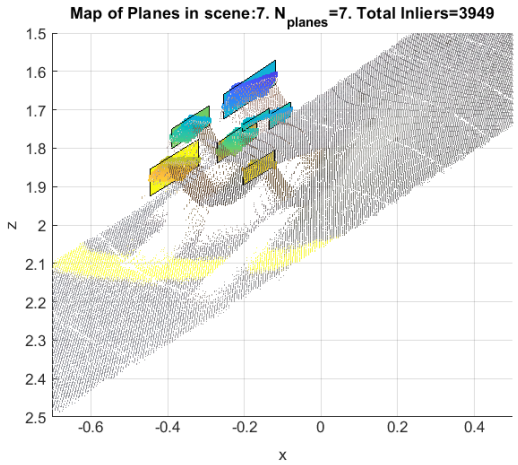
scene	DP	MP	MD	SP	Total Planes in Map
1	3	4	1	14	21
6	1	0	3	21	22
7	2	0	2	15	17
29	2	2	2	31	35

Table II shows the performance of the algorithm with a $th_{inliers} = 0.001$. In scene 1, 75% of the boxes were detected, even though they all had similar heights [0.25 to 0.252] mt. The value of $th_{inliers}$ generated a high volume in the map of planes (21 planes), likewise, it caused several of these planes to overlap. The dissimilarity indicator penalized these overlaps, detecting several of them as spurious planes (14). Some of those planes (4) had enough area to be classified as a multiple detection. In scene 6 at least three different box heights are identified (assuming that 0.25 is very close to 0.252). For this case, only 25% of boxes were detected. A detailed review of the case (see Fig 5a) allows us to observe that (1) the planes were located in the expected area, however, the overlapping of planes associated with the low $th_{inliers}$ increased the dissimilarity measure e_{diss} and therefore 21 of the 22 plans that made up the map were rejected.

In scene 7 two different heights are identified (view Table II). In this scene the algorithm manages to detect 50% of the planes. Multiple detections are reduced compared to scene 1 and most plane become spurious due to differences in area (d_{area}). Scene 29 has four different box heights. For this scene the algorithm identifies 50% of the reference planes. The tendency to penalize most of the planes detected as spurious



(a) Selected Models from a set of 69 models detected by RANSAC algorithm ($th_{inliers} = 0.001$) in scene 6.



(b) Selected Models from a set of 13 models detected by RANSAC algorithm ($th_{inliers} = 0.005$) in scene 7. Note that there are two models located away of the box's top-surface

Fig. 5: Output of the Filtering Stage

TABLE IV: Results by scene using a $th_{inliers} = 0.005$

scene	DP	MP	MD	SP	Total Planes in Map
1	2	1	2	8	11
6	1	0	3	6	7
7	4	0	0	3	7
29	2	0	2	11	13

planes is preserved, due to differences in area. In summary, the use of a low $th_{inliers}$ is associated with a majority of planes with few inliers, which, in most cases, expand in an area less than 20% of the expected area. For this reason those planes are classified as spurious planes.

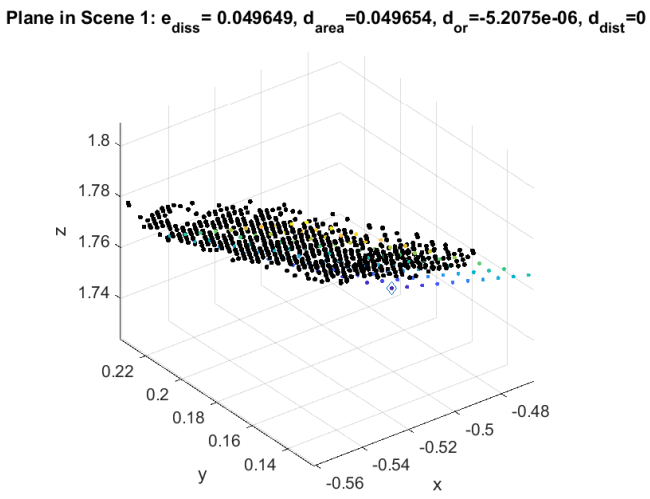
Table IV shows the performance of the algorithm with a $th_{inliers} = 0.005$. As expected, increasing $th_{inliers}$ reduces the number of planes per map and increases the number of inliers (see Figure 5). As shown in Figure 5b, with increasing $th_{inliers}$, some planes are located far from the expected zone. Consequently, and unlike Table III, the planes are also rejected for dissimilarity in distance. Scene seven is striking, where 100% of the shots are located, accompanied by four repetitions (see Figure 5b)

VI. Conclusions

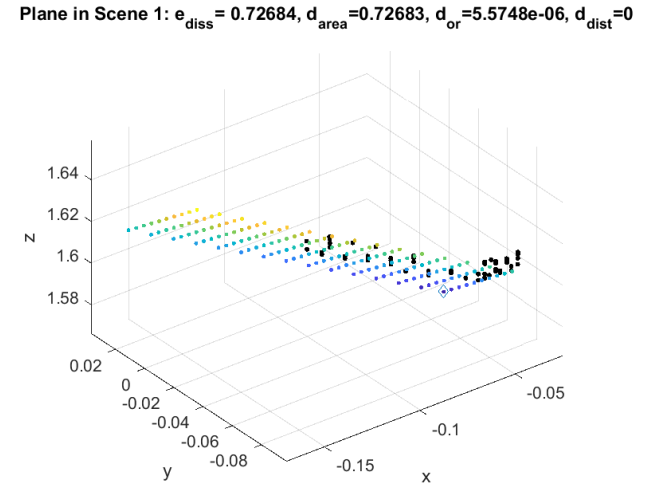
This article presents an algorithm for the extraction of maps of top-plans, from point clouds of packing scenes. The performance of the algorithm is evaluated in terms of four indicators (DP, MP, MD, SP) dependent on a dissimilarity function between the reference plane and the detected planes. The main contribution of the paper is an algorithm that can detect an average of 50% of the top-planes in the assessed scenes combined with a high number of spurious planes. The analysis of the results show a high dependence between the performance of the algorithm and one of the parameters used during the extraction of planes: $th_{inliers}$. Low values of this parameter generate a high number of planes with few inliers in most planes. These few inliers are distributed in areas smaller than 20% of the reference area and for that reason they are classified as spurious planes. Increasing the value of $th_{inliers}$ reduces the number of planes and increases the inliers, however, it generates plane detections at distances of more than 20% of the minor edge in the reference plane. Therefore, several of the planes detected are also classified as spurious. Additional contributions are: (1) a dissimilarity index between planes which shows consistency with the results obtained, (2) a framework useful to perform the detection of planes and assessment of the algorithms.

This work allows identifying several improvements to be implemented on the proposed pipeline:

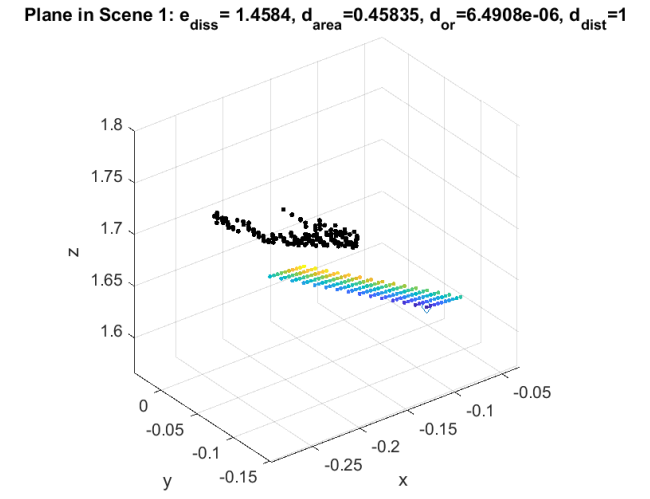
- The Fitting-Plans process currently used is based entirely on the MSAC algorithm available in Matlab. This process have shown high computational cost (aprox 2.5 min by scene) and requires a more efficient implementation. One candidate is the efficient RANSAC algorithm presented in [12].
- The independent processes of Fitting-Plans and Filtering-Top-Plans prevent the inliers of a filtered plane from being used in another plane. One possible solution is to integrate these processes and reprocess the inliers of the filtered plans.
- The Mean Shift-based plane division stage can be improved using techniques that use information from the neighborhood of the point cloud. These techniques can identify discontinuities such as those expected at the edges and use these detections to activate clustering criteria. In this way we can create a connected com-



(a) Accepted model



(b) Rejected Model by Area



(c) Rejected Model by Distance

Fig. 6: Samples of dissimilitude function between a ground-truth plane p_{gt} (colored points) and a plane that belongs to the predicted map of planes p_i (points in black). Note that all the samples have a low d_{or} as a consequence of the parametrization in the RANSAC algorithm: restrict the fitting to planes parallel to ground

ponents on the surface criteria useful for the clustering. A proposal like these is presented in [8]

- The filtering process can use the information on normal of points to reject planes like those highlighted in the Figure 5b
- The detection of ground normal in Algorithm 1 it works cause the clutter in the scene is low. For high clutter in the scene we must use a robust method. One candidate is the gravity estimation algorithm presented in [7], [6].
- Finally, We expect to implement experiments over the whole dataset to conclude about the performance of the algorithm under different distances between camera and objects and different points of view

REFERENCES

[1] Andreas Bortfeldt and Gerhard Wäscher. Constraints in container loading – A state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20, aug 2013.

[2] Guillermo Camacho-Munoz, Humberto Loaiza-Correa, and Sandra-Esperanza Nope. A survey on components of AR interfaces to aid packing operations. 2020.

[3] Cristian Alfonso Celis Gonzales and Fabian Alejandro Leon Almeciga. *Evaluación de la exactitud, precisión y tiempo de procesamiento en un algoritmo de estimación de pose y tamaño a partir de la variación de parámetros de escena en aplicaciones de estibado automático*. Tesis para optar al título de ingeniero industrial, Universidad de La Salle, 2019.

[4] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[5] Keinosuke Fukunaga and Larry D. Hostetler. The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.

[6] Saurabh Gupta, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Aligning 3D models to RGB-D images of cluttered scenes. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:4731–4740, 2015.

[7] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 564–571, 2013.

[8] Hani Javan Hemmat, Arash Pourtaherian, Egor Bondarev, and Peter H. N. de With. Fast planar segmentation of depth images. *Image Processing: Algorithms and Systems XIII*, 9399(c):93990I, 2015.

[9] Mathworks. pcfitplane, 2020.

[10] PointClouds.org. Point Cloud Library (PCL) 1.11.1-dev, 2020.

[11] A. Galrão Ramos, José F. Oliveira, and Manuel P. Lopes. A physical packing sequence algorithm for the container loading problem with static mechanical equilibrium conditions. *International Transactions in Operational Research*, 23(1-2):215–238, 2016.

[12] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. *The Eurographics Association and Blackwell Publishing 2007*, 26(2):214–226, 2007.

[13] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2nd edition, 2020.

[14] F. Tarsha-Kurdi, T Landes, and P Grussenmeyer. Hough-Transform and Extended Ransac Algorithms for Automatic Detection of 3D Building Roof Planes From Lidar Data. *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, XXXVI(1):407–412, 2007.

[15] P. H.S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[16] Phil Torr and Andrew Zisserman. Robust Computation and Parametrization of Multiple View Relations. In *Sixth International Conference on Computer Vision*, pages 727–732, Bombay, India, 1998. IEEE Cat. No.98CH36271.

[17] Eduardo Vera. Implementation of the fast Hough Transform for plane detection in depth images, 2017.

[18] Eduardo Vera, Djalma Lucio, Leandro A.F. Fernandes, and Luiz Velho. Hough Transform for real-time plane detection in depth images. *Pattern Recognition Letters*, 103:8–15, 2018.

[19] S. Zennaro, M. Munaro, S. Milani, P. Zanuttigh, A. Bernardi, S. Ghidoni, and E. Menegatti. Performance evaluation of the 1st and 2nd generation Kinect for multimedia applications. *Proceedings - IEEE International Conference on Multimedia and Expo*, 2015-August(January), 2015.

[20] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object Detection in 20 Years: A Survey. pages 1–39, 2019.