IIEEJ **Paper** *-- Special Issue on IEVC2017 --*

Robust 3D Planes Detection under Noisy Conditions Using Scaled Difference of Normals

Jaime SANDOVAL† (*Student Member*), Kazuma UENISHI† (*Member*), Munetoshi IWAKIRI†† (*Member*), Kiyoshi TANAKA† (*Fellow*)

† Shinshu University, †† National Defense Academy of Japan

<**Summary**> 3D planes detection is an important task that has numerous applications in urban environments. However, current methods do not deal appropriately with the noise and quantization artifacts of low-cost sensors. In this paper, we present the Scaled Difference of Normals, a points filter that addresses these issues and is implemented on top of the Fast and Deterministic Planes Detection Based on Hough Transform. We evaluated its precision by comparing the detected planes coefficients with semi-automatically generated ground truth data and confirmed that when compared to state of the art methods, the proposed method is fast and has superior precision even in the presence of high noise levels, quantization artifacts and several variations in the points distribution caused by registration.

Keywords: 3D point clouds, planes detection, model fitting, Hough transform, RANSAC

1. Introduction

With the increasing availability of low-cost 3D sensors and its integration into drones, robots, vehicles and portable devices, 3D information processing and pattern recognition has become an important part of computer vision.

Point clouds are merely a set of points in $\mathbb{R}^3$, they can also come with color or intensity data. However, we will use only spatial data to further generalize.

In urban environments, most objects and scenes are composed of many planar surfaces. Therefore, it is necessary to provide efficient and accurate algorithms to detect 3D planes.

Current methods can be categorized in two types. The first type performs heavy point-wise calculations on point clouds, e.g. Generalized Hough Transform[1]. The second type exploit random sampling to acquire faster speeds, e.g. RANSAC[2]. Therefore, non deterministic methods are the most popular since they are robust and relatively fast. However, they are far from realtime speeds even in modern CPUs, and cannot be parallelized entirely on GPUs due to their serial nature.

In a previous work[3], the authors designed a method that achieved realtime speeds. However, in the presence of noise artifacts it is forced to increase its spatial threshold, hence not being able to detect small planar surfaces. The same problem occurs for the conventional methods in References 2) and 4).

In this paper we propose a novel filtering method, 'the Scaled Difference of Normals', and an improved Fast and Deterministic Hough Transform (FDHT).

First, 'the Scaled Difference of Normals' remove non planar points robustly even in the presence of noise artifacts, then the points distribution and normals are passed to FDHT to detect planes. The improved FDHT uses the already calculated voxel centroids and normals to refine the final results.

We evaluated this algorithm and concluded is robust to noise artifacts and has better accuracy while preserving low computational costs.

2. Related Works

2.1 RANSAC

The RANSAC[2] algorithm is by far the most popular method for planes detection. It is a non deterministic and generalized method for model fitting, and has demonstrated to be effective in feature matching. To increase its robustness when matching correspondences several variations of the original method have been created. For instance, MSAC & MLESAC[5] provide an enhanced evaluation of the model quality function in the verification step. PROSAC[6]enhances the hypothesis generation by sorting samples using a quality function.

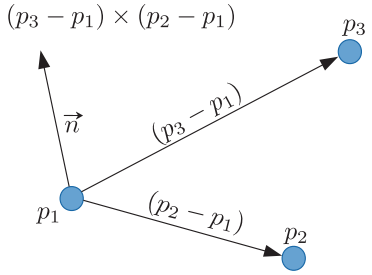


Fig. 1 A plane defined from 3 points and a cross product

It basically consists of picking a small random subset of the data, instantiating a model and deciding whether is a good model or not.

For planes detection the number of required samples is 3. These points, p_1, p_2 and p_3 are randomly selected from the dataset. Then, a model is instantiated by calculating the unit normal vector $\vec{n}$ using the cross product of two vectors over the plane,

$$\tau \vec{n} = (p_2 - p_1) \times (p_3 - p_1), \tag{1}$$

where $\tau = \|\vec{n}\|^{-1}$.

Figure 1 shows how the combination of 3 non collinear points forms a normal vector using the cross product.

By applying the Hessian normal form of a plane, the normal vector and any point of the samples can be used to build up the plane coefficients,

$$\vec{n} \cdot p_n + \rho = 0. \tag{2}$$

A point p_n over a plane is represented by its coordinates (x, y, z). Then the plane equation is defined by the distance from the plane point p_n to the origin as the dot product of p_n and the unit normal vector $\vec{n}$, which has three elements (a, b, c); therefore after the dot product the following equality arises:

$$ax + by + cz + d = 0. \tag{3}$$

It is the plane equation in cartesian coordinates where $\rho = -d$ in Eq. (2).

Once RANSAC obtains the plane equation coefficients (a, b, c, d) from the samples. It evaluates the quality of a model by counting the inliers within a defined distance threshold. If the model is not good enough, is discarded and the process starts again. When a good model is found, it is refined against the inliers using the least squares method.

Nonetheless, as a plane detector in point clouds, its robustness is overshadowed by the fact it is a single-model fitting algorithm. Regarding to point clouds, it has to detect multiple plane models in a scene, e.g. a hallway, buildings, doors. Therefore, it has to be executed iteratively many times within the dataset.

Additionally, in real world data, particularly in the presence of quantized noise there is a well documented study[7] on the bias of RANSAC model fitting in the presence of surface discontinuities.

Coarse-to-Fine RANSAC (CFRANSAC)[8], is an iterative approach for 3D planes detection. It uses RANSAC iteratively and a fine segmentation over a coarse segmentation to increase RANSAC accuracy. Moreover, to segment only points from the same surface, inliers are refined using their normal vectors.

However, CFRANSAC does not perform well in the presence of surface discontinuities patterns[7]. Therefore, to improve the accuracy an ultrafine step was added to the process after the fine segmentation. This UltraFine RANSAC method will be called UFRANSAC.

Figure 2 shows how UFRANSAC works. The points are sideviews of two different planar surfaces. Figures 2 (a) and (b) show how iteratively running RANSAC with a coarse threshold C_{th} and a fine threshold F_{th} can improve the accuracy of the detected model. However, in the presence of close planes as shown in the picture, an additional step with an ultrafine threshold UF_{th} can improve further the accuracy and detect both planar models in Fig. 2(c).

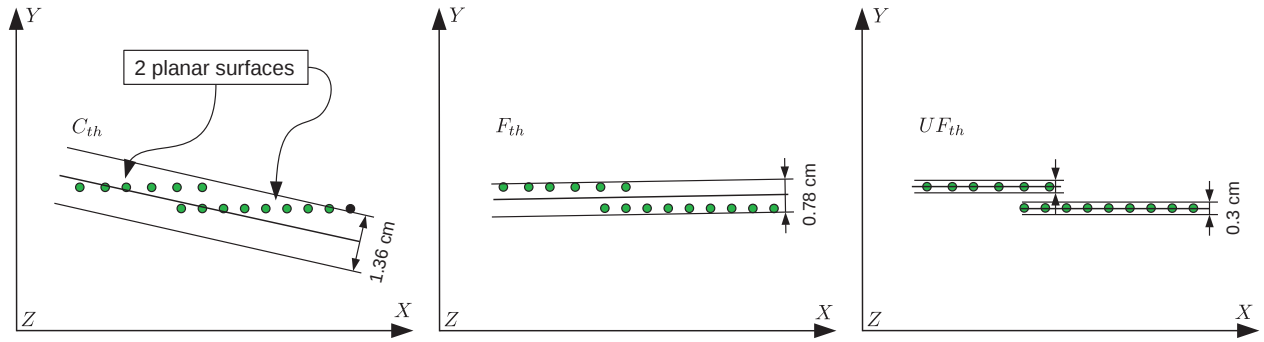
2.2 Randomized Hough transform

It is noted that the main bottleneck of the Generalized Hough Transform is the voting procedure. The Randomized Hough Transform (RHT)[9] is a big improvement to the Hough Voting of the Generalized Hough Transform[1] which was originally proposed by Paul Hough[10].

Analogous to Monte Carlo methods, it randomly selects a subset of the data to generate models and cast votes. After several iterations it can approximate the full accumulator data without having to process the entire dataset.

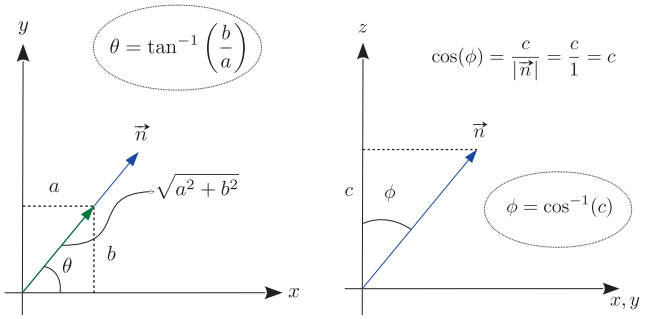
RHT iteratively picks j random samples and constructs a specific model, where j is the minimum number of samples required to build the model. Votes are casted into an accumulator for each iteration. Finally peak detection is used to extract the most prominent models.

RHT for 3D planes detection[4] picks 3 random samples of points within a certain distance and constructs a model. Using the plane coefficients (a, b, c, d) from Eq. (3), the plane coefficients are then transformed into



(a) Step 1, coarse threshold **(b)** Step 2, fine threshold **(c)** Step 3, ultrafine threshold

Fig. 2 UFRANSAC process illustration



(a) θ angle **(b)** ϕ angle

Fig. 3 The angular parameters of the Hough Transform for 3D planes: θ and ϕ

Hough Space using their polar coordinates (θ, ϕ, ρ)

$$x_i \cos\theta \sin\phi + y_i \sin\theta \sin\phi + z_i \cos\phi - \rho = 0, \qquad (4)$$

where their equivalences with the cartesian coordinates coefficients are shown in **Fig. 3**. Here, the distance parameter $\rho = -d$.

Then using the Hough Space coefficients (θ, ϕ, ρ), RHT casts 1 vote into the accumulator. In each iteration the algorithm performs peak detection for the current cell. If a cell reaches a certain number of votes, a plane is detected. Additionally, the processed j samples are removed from the dataset in each iteration.

It finishes when there are a certain number of points left, a specified maximum number of planes is reached or if the algorithm fails to generate a model more than a defined number of times.

One remarkable proposal is a novel accumulator design which improves the accuracy of 3D planes detection[4]. However, it can take longer processing time as the amount of non planar points in the data increases.

2.3 Comparison

RANSAC and RHT are using random sampling to detect 3D planes. The former counts the inliers within a threshold of hypothetical models. The latter counts the frequency of hypothetical models appearances from samples using the Hough Transform.

Table 1 Features of conventional methods

	RANSAC	RHT
Multimodel	×	○
Robustness	△	△
Efficiency	×	△

RANSAC is a single model approach while RHT can detect multiple models in a single iteration of the algorithm. Both are robust to Gaussian noise, nonetheless, their accuracy falls dramatically in the presence of artifacts generated by low cost sensors.

RANSAC is slower because it needs several iterations to detect multiple models. Moreover, it needs to calculate computationally expensive point-wise surface normals to detect 3D planes properly. On the other hand, RHT does not need point normals or further pruning. However, both methods are still computationally expensive for realtime applications.

As seen in **Table 1**, the biggest problems of the conventional methods[2),4)] are their ineffectiveness against noise artifacts and their efficiency.

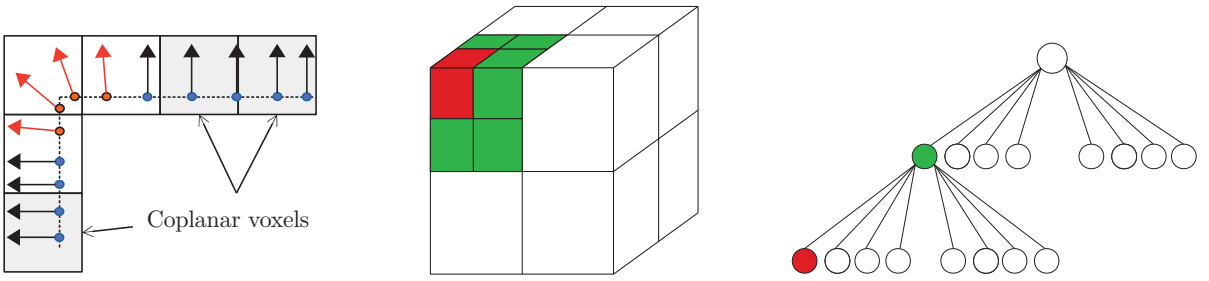
3. Proposed Method

3.1 Fast and deterministic Hough transform

In the Fast and Deterministic Hough Transform (FDHT), instead of reducing computational costs of the voting procedure using random sampling. It reduces computational complexity by voting once for coplanar patches instead of voting several times for each point.

One main concept behind the FDHT is the assumption that normal vectors of coplanar regions are similar.

As seen in **Fig. 4** (a), planar regions have identical or similar normal vectors. Therefore it is more likely for a



(a) Normal vectors of coplanar voxels (b) 3D space subdivision of octrees (c) Voxel hierarchical structure stored in memory

Fig. 4 Coplanar voxels and octrees

patch to be in a plane when is coplanar.

FDHT starts by dividing the 3D space into voxels using an Octree. As seen in Fig. 4 (b) and (c), the octree subdivides the 3D space into 8 cubes recursively until it reaches a defined voxel size V_s. Then voxel information is stored in the leaves, such as occupancy, centroid, points indices, among others. In FDHT, points indices are stored in leaves to query points inside each voxel later.

Consequently, for each voxel the planarity and coplanarity of its points distribution is evaluated. For this purpose, since is fast to calculate centroids, Difference of Centroids[11] is used to calculate the probability of a voxel to be planar. Then, we use the centroid μ_p to generate the covariance matrix and its eigenvalues $\lambda_1 < \lambda_2 < \lambda_3$ using PCA.

These eigenvalues are used to filter planar voxels[12], a planar region can be filtered out by using two constraints: thickness and isotropy. For thickness the relationship between the first and second eigenvalue is used as follows,

$$\lambda_2 > \alpha\lambda_1. \tag{5}$$

For isotropy, the relationship between the second and third eigenvalue is used,

$$\beta\lambda_2 > \lambda_3. \tag{6}$$

Since $\lambda_1 < \lambda_2 < \lambda_3$, valid values for α and β are within the range $(1, \infty)$

With these constraints if is planar enough, the unit normal vector $\vec{n}$ is obtained from PCA, corresponding to the eigenvector associated to the smallest eigenvalue. Then the spanning plane is calculated with the voxel centroid and its normal vector using the Eq. (2).

To detect coplanarity, we measure the difference between two normals at different radius from the current voxel centroid. The normal vector of the voxel plane is set to $\vec{n_1}$ and the normal vector at an expanded radius

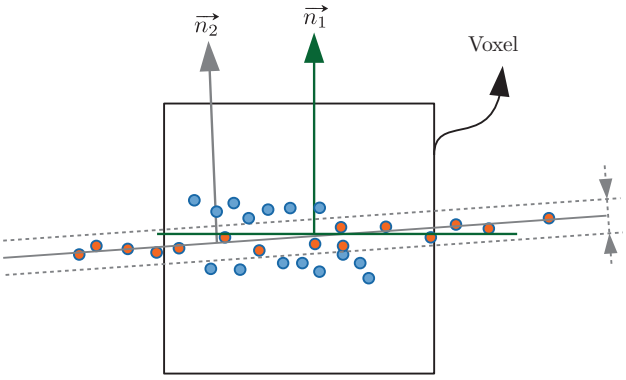


Fig. 5 Voxel plane refinement using coplanar points

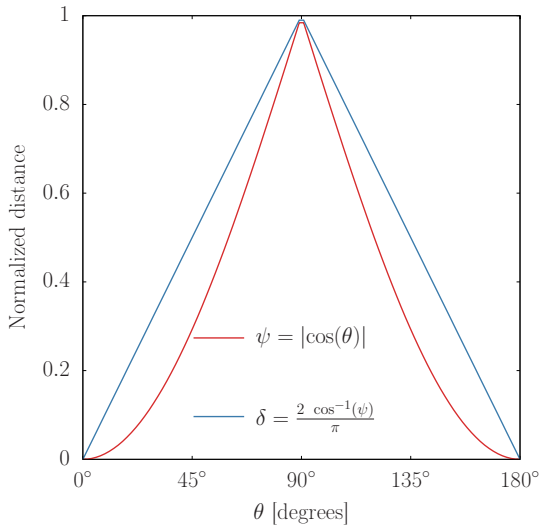


Fig. 6 Comparison of the positive cosine distance and positive cosine similarity functions

is set to $\vec{n_2}$. **Figure 5** shows how the search radius is expanded F times to look for inliers (in orange) within a threshold T_h from the detected plane outside the current voxel.

If the orange points describe another surface $\vec{n_1}$ and $\vec{n_2}$ will point to different directions. Therefore, if the cosine distance δ between $\vec{n_1}$ and $\vec{n_2}$ is less than 0.1, then the voxel is coplanar with its expanded radius. The cosine

distance δ is illustrated in **Fig. 6**. Where the absolute cosine $|\cos(\theta)|$ is defined as

$$\psi = |\overrightarrow{n_1} \cdot \overrightarrow{n_2}| \mapsto [0, 1] \tag{7}$$

since both $\overrightarrow{n_1}$ and $\overrightarrow{n_1}$ are unit vectors. Therefore, its angle becomes the normalized angular distance δ defined as

$$\delta = \frac{2\ \cos^{-1}(\psi)}{\pi}. \tag{8}$$

The coplanarity radius C_r is defined by the voxel size V_s and a factor γ,

$$C_r = \gamma \frac{V_s}{2}. \tag{9}$$

Using the Hessian normal form of the plane from Eq. (2), for every coplanar patch the plane coefficients are calculated using its centroid μ_p and its normal vector $\vec{n}$ already calculated by PCA. Then the plane coefficients from the spatial domain (a, b, c, d) are transformed into the Hough Space (θ, ϕ, ρ) in accordance with Fig. 3.

The next step is Hough Voting. Each dimension of the accumulator is divided into a defined amount of bins $(\theta_{bin}, \phi_{bin}, \rho_{bin})$. The number of bins defines the precision of the accumulator. Votes are casted into the corresponding accumulator cell in Hough Space by using the number of inliers from the coplanarity checking.

Because FDHT votes only for coplanar voxels, a dense accumulator is not required; hence it uses a sparse memory model for the accumulator. Therefore a new accumulator structure was proposed for this method. It consists on a nested map structure of (θ, ϕ, ρ) values as illustrated by **Fig. 7**. The nodes corresponding to ρ stores the accumulator votes data and other useful information such as the detection location and a list of inliers.

From the sparse accumulator, planes are sorted by votes in descending order and extracted until the remaining votes reach a V_t percentage. When cells have higher votes, the corresponding plane has more local representations in the point cloud. Therefore, only the most voted planes are extracted.

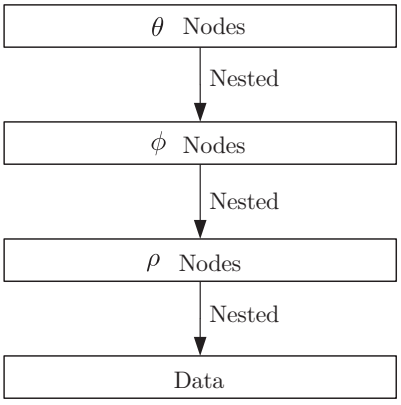


Fig. 7 Nested map structure of the FDHT accumulator

Conventionally, to eliminate duplicates due to subtle variations in the plane equations, their euclidean distance in $\mathbb{R}^4$ space was computed in the past as a measure of similarity. However, planes are more globally describable than locally. Then, to further eliminate the negative influence of noise inside each planar voxel, the following planes refinement has been developed for FDHT.

During the voting procedure, the voxel centroid and normal vector of each coplanar voxel was saved in a centroids point cloud P_c. This downsampled point cloud is used for a final planes refinement.

Iteratively, inliers are selected from inside P_c for each plane reusing the coplanarity threshold. Additionally, the angular distance (Fig. 6) between the normals of the points inliers in P_c and the current plane normal is also considered for inliers selection. Once inliers are defined for each plane, they are removed from P_c. Finally, the refined planes are the ones who have at least 1 inlier in P_c.

Difference of Normals (DoN)[13] is an operator that works on multiple scales, it subtracts two normal vectors $\overrightarrow{n_1}$ and $\overrightarrow{n_2}$ at radius of different sizes, r_1 and r_2. The DoN value is the L2 norm between these two vectors.

$$DoN = \left| \frac{\overrightarrow{n_2} - \overrightarrow{n_1}}{2} \right|_2. \tag{10}$$

Nonetheless, if the normals are not oriented towards a viewpoint, it can miscalculate the value. Therefore, to prevent this issue the normalized positive cosine distance δ of Fig. 6 is used as the DoN value.

The cosine distance is a normalized angular distance between two vectors, its range is $[0, 1]$, where 1 indicates orthogonality and a value near 0 indicates collinearity.

A weakness of DoN is its dependence on normals calculation, hence in PCA. Therefore the method itself remains weak to non-gaussian noise. Particularly in the case of Kinect noise where points are grouped into independent and self-correlated planar clusters.

3.2 Scaled difference of normals

Figure 8 shows a case when normals calculation fails. The dotted line is the real surface of the original model, the vector marked as $\vec{n}$ is its normal vector.

This case can be visualized from the curvature calculated from the eigenvalues of the covariance matrix,

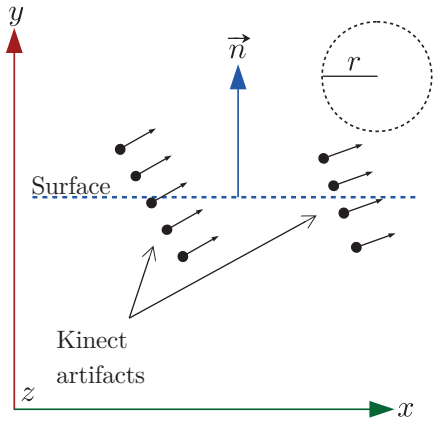
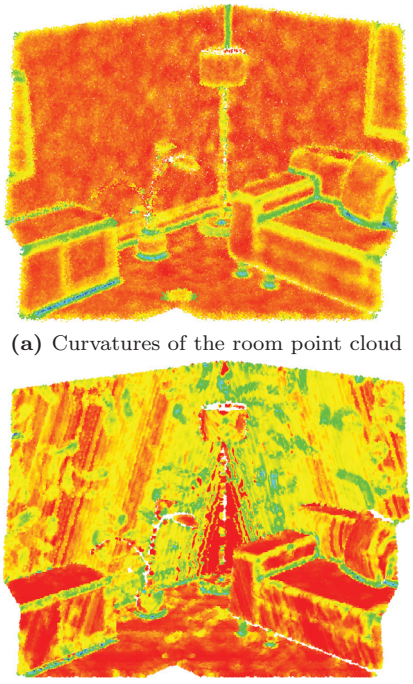


Fig. 8 Sideview of a normals calculations with a small search radius in the presence of Kinect noise



(a) Curvatures of the room point cloud

(b) Curvatures of the room point cloud with Kinect noise

Fig. 9 Curvatures heatmap under Gaussian and Kinect noise

$$c = \left\{ \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \in \left[0, \frac{1}{3}\right) \mid \lambda_1 < \lambda_2 < \lambda_3 \right\}. \quad (11)$$

Figure 9 shows heatmaps of the local curvatures at each point with a fixed radius of 10[mr]. Figure 9(a) has an added gaussian noise of 2[mr]. Figure 9(b) features several types of noise artifacts generated by the simulation of a Kinect sensor. Even though the surface thickness of both point clouds were increased similarly due to noise, Fig. 9(a) shows that when using a uniform search radius to calculate the curvatures using PCA, the curvature pattern is uniform.

However, in Fig. 9(b) in spite of having an analogous search radius. The curvature pattern is not uniform, and the structure of the Kinect noise artifacts can be still seen.

A normal workaround for this noise pattern is to provide a bigger search radius, which can increase the computational costs and affect the normals calculation near sharp surfaces.

Therefore, we propose the Scaled Difference of Normals (SDoN), an extension of FDHT which transforms and filters undesired points to reduce the impact of both Gaussian and quantized noise.

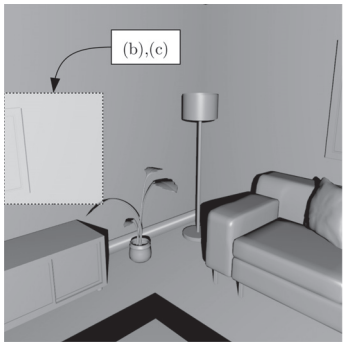
SDoN calculates point normals, Difference of Normals and planarity for each point after resolution downscaling to alleviate quantization artifacts. Consequently, it downscales and calculates DoN scores iteratively. In each iteration point scores are generated and stored in an array which will be used to decide whether to filter out a point or not.

For resolution downscaling we used the Voxel Grid Filtering implementation of the PCL library[14]. Instead of doing computationally expensive point-wise calculations, it subdivides the 3D space using octrees. Then it replaces the point cloud with the centroids of the octree. This method acts as a low-pass filter, which downsamples the points and therefore downscales its resolution. For that reason, in the proposed method we will use the term downscaling to refer the application of Voxel Grid Filtering.

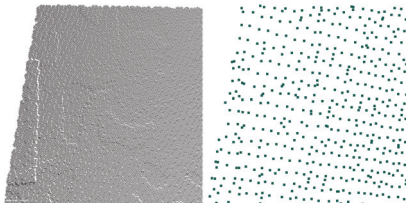
Figure 10 shows an example of minimization of noise artifacts using downsampling. The observed location is pointed by Fig. 10(a). Figure 10 (b) and (c) show respectively the front and side views of points over a plane in a simulated Kinect scan. Both pictures are composed of left and right sections. The left part shows the noisy Kinect scan. The right part shows the same point cloud but with a downscaling of 15[mr].

As seen in Fig. 10(b), the effect of Kinect quantization is not visible because it is view from the sensor perspective. However, after rotating the view, the pattern of Fig. 10(c) arises and planar clusters can be seen across the surface. When comparing both right and left images, it is clear that the negative influence of Kinect noise decreases at the cost of reducing the number of details and small surfaces.

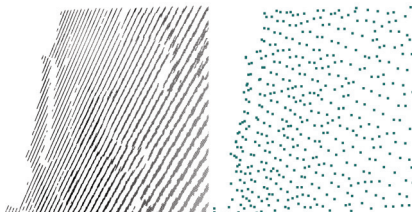
In other words, as the point cloud is downscaled, planar surfaces become clearly visible at the cost of increasing the distance between points. For that reason, SDoN preserves the points from the first downscaling, and analyzes the respective points distribution down through the



(a) Location of the zoomed in region in the render



(b) Front view of the wall with Kinect noise



(c) Side view of the wall with Kinect noise

Fig. 10 Planar section affected with simulated Kinect noise and its points distribution before and after Voxel Grid filtering

scales.

It is noted that the DoN value is treated as a score and not as a distance metric, the proper distance metric from Fig. 6 is used as a reference to define the DoN score:

$$DoN = 1 - \delta. \tag{12}$$

Figure 11 is the flow of the SDoN process. The points of the initial downscaling will be the reference points. These are used as the search radius center for DoN, then the input data for normals calculation is the downscaled point cloud. It is noted that in each iteration, SDoN expands the search radius based on the mesh resolution of the new point cloud to avoid running out of points for the normal vector calculation. This process is run iteratively to build a DoN scores table for each point and for each downscaling.

The scores table is built to decide which of the reference points are filtered out. The decision criteria of the algorithm is to keep only the points that are planar in many scales. For this purpose, DoN scores of the multiple scales are thresholded by a minimum value. If any of them goes

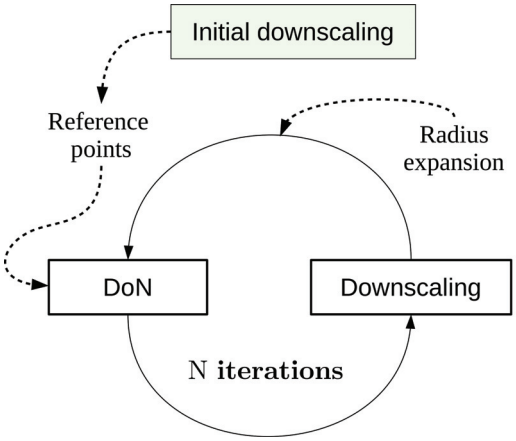


Fig. 11 Simple diagram of the SDoN calculation process

Table 2 Example of SDoN scores table with a minimum score set to 0.9

Downscaling	Point 1	Point 2	Point 3
1st	0.7	0.93	0.91
2nd	0.91	N/A	0.99

down a defined DoN score, the point is discarded.

However, there may be cases in which a reference point does not have enough neighboring points to calculate DoN on the downscaled point cloud, particularly in higher iterations when the point cloud downscaling becomes too aggressive. In that case, the undefined DoN score is ignored and only the remaining scores are utilized.

A minimal example is shown in **Table 2**. In this case SDoN is configured to a minimum score of 0.9. The table shows that only Point 1 is discarded because its score at the first downscaling falls below the minimum: 0.9. It is noticed that Point 2 does not have a DoN value at the second downscaling because the number of points k became insufficient for calculating the normal vector ($k < 3$). Nonetheless, the point is not discarded because in the first downscaling it has a score above 0.9.

SDoN enhances FDHT as it filters and preprocess data for 3D planes detection. Particularly, the normals of reference points are reused for coplanarity checking and the final filtering of FDHT.

Normals calculation is critical to SDoN accuracy. SDoN uses PCA to calculate them. However it was demonstrated that PCA is prone to noise, particularly with non-gaussian noise[15]. Hence, the study of normals calculation in noisy data is out of the scope of this work.

The summary of the proposed method parameters is described in **Table 3**. They are divided into 2 groups:

Table 3 Parameters of the proposed method

Group	Parameter	Description	Unit
SDoN	$SDON_{vs}$	SDoN initial voxel size	[mr]
	$SDON_{it}$	SDoN iterations	#
FDHT	V_s	Voxel Size	[mr]
	α	Planarity factor	N/D
	β	Isotropy factor	N/D
	T_h	Plane inliers threshold for coplanarity	[mr]
	F	Coplanarity radius expansion factor	N/D
	θ_{bin}	Accumulator θ bins	#
	ϕ_{bin}	Accumulator ϕ bins	#
	ρ_{bin}	Accumulator ρ bins	#
	V_t	Votes tolerance	%
	K_t	Angular tolerance	[degrees]

SDoN parameters, and the improved FDHT parameters.

4. Experiments

4.1 Outline of the experiments

The objective of the experiments is to evaluate the efficiency and precision of the 3D planes detectors algorithm and to compare its results with the proposed method.

Three metrics are used for this purpose: processing time T, the number of correctly detected planes n_d and total error E. CFRANSAC, UFRANSAC and RHT, since they are non deterministic, the average values of 50 executions defined the metrics results.

The Fast and Deterministic Hough Transform (FDHT) was also included to show the impact of not having SDoN both in computational efficiency and precision.

It is noted that the processing time T of each method was evaluated using wall time. On the other hand, since the variations of processing time along executions is negligible for the proposed method and FDHT, the result of a single execution was used as their processing time metric.

4.2 Dataset

We used synthetic point clouds generated by Blensor 1.0.17[16], which can provide both clean point clouds and realistic noise simulations from a variety of sensors.

For 3D point clouds generation, we used 3D models with several planar surfaces, a room[17]and a kitchen[18]. Additionally, a model of a car[19] with slightly curved surfaces was used to test out which methods are good at detecting planarity in this scenario. The car model is particularly difficult for the proposed method since for the underlying 3D planes detection, FDHT, might vote for the different tangent planes of the curved surface into the accumulator instead of the global surface.

To prepare a more realistic dataset we did not use clean point clouds. Instead, Gaussian noise of 1[mr] was added to each point of the synthetic clean datasets. These point clouds are not named by any particular annotation; however, the point clouds with Kinect noise are marked as *noisy*.

Figure 12 – 14 show the input point clouds and their details are in **Table 4**. The room point clouds shown in Fig. 13(a) and Fig. 13(b) were created with a single Kinect scanning, while the Kitchen point clouds (Fig. 14(a), Fig. 14(b)) and Car point clouds (Fig. 12(a), Fig. 12(b)) were created by simulating a noisy gaussian and non-gaussian noisy registration. This was done in Blensor by moving the camera 10 times through a path, at each step, both noiseless and noisy Kinect scans were obtained in world coordinates. Finally, the point cloud was concatenated to simulate a perfect registration.

UFRANSAC was run several times over the clean point clouds to capture all the correct planes with high accuracy. Since there exists false positives in the detection, only correct planes were manually selected from in each iteration. Thereupon their final coefficients were saved as the ground truth data.

5. Evaluation

The precision evaluation algorithm is defined in **Algorithm 1**. It consists in generating one-to-many relationships between the ground truth planes and the detected planes. For this purpose it is necessary to provide a list of coefficients of the ground truth planes $Planes_{gt}$ and the detected planes $Planes_{det}$.

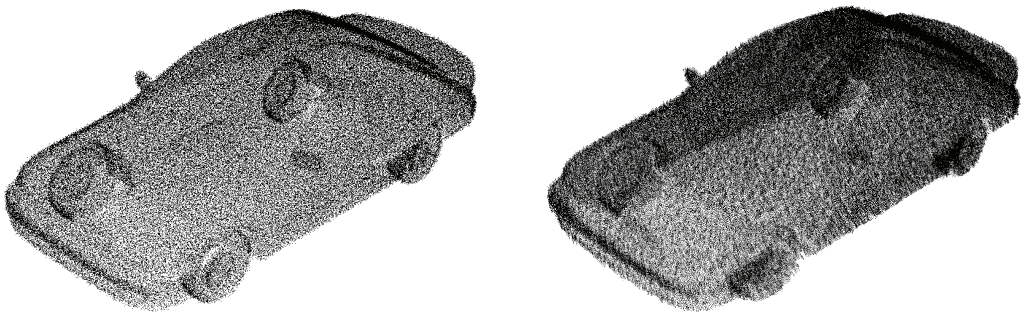
Each plane coefficient is a normalized unit vector in an $\mathbb{R}^4$ vector space. Therefore, to compare plane coefficients the angle between planes coefficients vectors was utilized. This angle δ_θ is based on δ from Fig. 6, where

$$\delta_\theta = 180\ \delta \mapsto [0, 180]. \tag{13}$$

First, each combination of detected and ground truth planes is evaluated using the function *bestmatch* of line 1, the output is the match of ground truth planes and detected planes with the minimum angular distance.

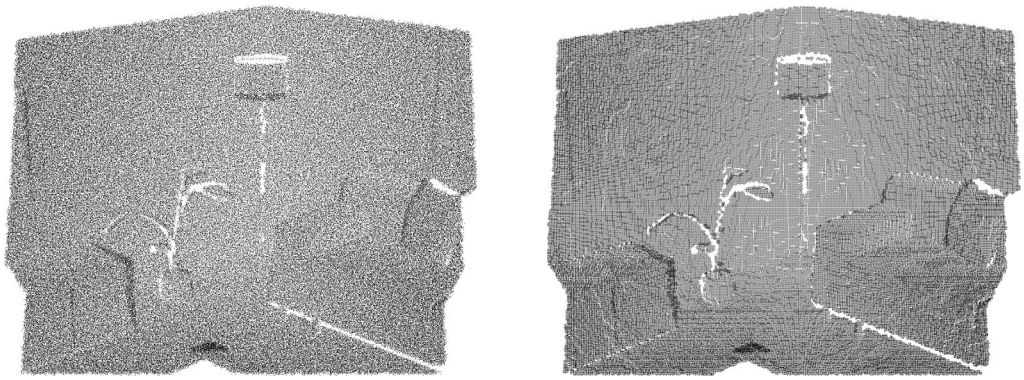
Then, the match is evaluated by thresholding its angle δ_θ under 9 degrees in line 14. Although δ_θ can be adjusted to any value, we found that 9 degrees matches visually with the correct number of detected planes, and lower value leads to a mismatch between the visual and numeric results.

Once we reach the threshold or we run out of detected planes, the algorithm calculates the number of correctly detected planes, i.e., the number of ground truth planes



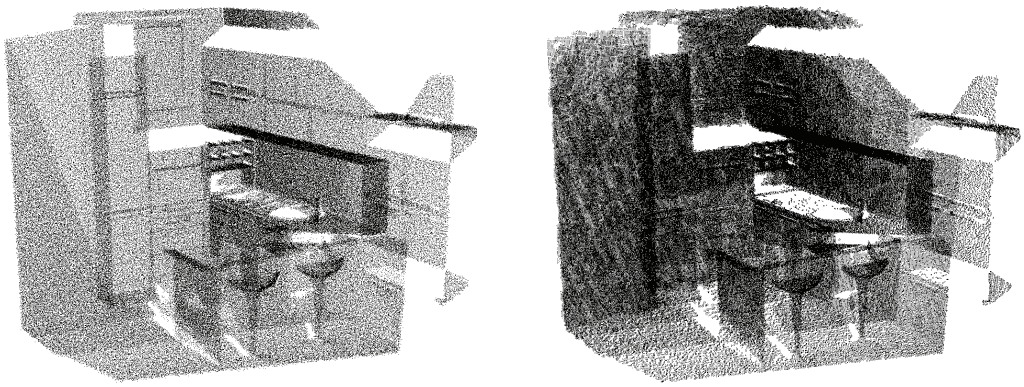
(a) Car **(b)** Car (noisy)

Fig. 12 Noisy and noiseless car point cloud obtained by Kinect simulation



(a) Room **(b)** Room (noisy)

Fig. 13 Noisy and noiseless room point cloud obtained by Kinect simulation



(a) Kitchen **(b)** Kitchen (noisy)

Fig. 14 Noisy and noiseless room point cloud obtained by Kinect simulation

Table 4 Dataset detailed information

Name	**Mesh resolution[mm]**	**Points[#]**	**AABB volume [m^3]**
Car	4.11	180,323	1.02
Car (noisy)	3.98	416,189	1.04
Room	5.08	307,200	19.69
Room (noisy)	5.05	307,200	19.20
Kitchen	6.75	245,928	30.32
Kitchen (noisy)	6.80	494,507	31.01

Algorithm 1 Precision evaluation

```
1: function BESTMATCH(Planes_det, Planes_gt)
2:     m ← emptyList
3:     for P_gt in Planes_gt do
4:         for P_det in Planes_det do
5:             δ_θ ← angle(P_gt, P_det)                ▷ Fig. 6
6:             m.add(P_gt, P_det, δ_θ)
7:     sort(m)                                  ▷ by angle, ascending
8: return m[0]
9: procedure EVALUATE(Planes_gt, Planes_det)
10:    L ← new List(Planes_det)
11:    M ← emptyMap                                    ▷ matches
12:    while l not empty do
13:        m ← BESTMATCH(l,Planes_gt)
14:        if m[δ_θ] ≤ 9 then                          ▷ Degrees
15:            M[match_gt].add(m)
16:            L.del(m_det)
17:        else
18:            break
19:    n_d ← size(M)                                ▷ correct matches
20:    while l not empty do
21:        m ← BESTMATCH(l,Planes_gt)
22:        δ_θ ← angle(m_gt, m_det)                    ▷ Fig. 6
23:        if δ_θ ≤ 9 then
24:            M[match_gt].add(m_det)
25:            L.del(m_det)
26:    for m in M do
27:        Err_gt ← Σ m_i[δ_θ]
28:    E ← Σ Err_gt_i
```

matched with detected planes below 9 degrees.

When there are more planes with higher angular difference, they are matched without thresholding to calculate the total error E in line 28.

6. Results

6.1 Efficiency and error

Figure 15 (a) and (b) show the proposed method has competitive accuracy when compared to conventional methods.

The planes it detects remain high while the error is kept low. However, when it faces slightly curved surfaces such as the point clouds from the car model, its error is not low as expected. This is because the algorithm sometimes fails at detecting locally planar surfaces as global planes.

Figure 15(c) shows that the proposed method is superior in processing time. CFRANSAC and UFRANSAC are by far the slowest methods, particularly in the presence of highly variant noise patterns across the point cloud as in the noisy point cloud of the Kitchen model.

In the Room point cloud results from Fig. 15(a), we can observe that the proposed method has low error and is slightly better than CFRANSAC and UFRANSAC. Notwithstanding, Fig. 15(c) shows it is significantly faster than UFRANSAC, about 11 times according to the numbers.

On the other hand, in the Room point cloud, FDHT is faster than the proposed method. Nonetheless, it has 6.8 times more angular error. Additionally, we can notice RHT looks close to the proposed method efficiency but it has 2.8 times more angular error and is 2.2 times slower.

In the noisiest point cloud, i.e. the noisy Kitchen, we can observe three important patterns. First, the error of CFRANSAC and UFRANSAC is lower than the proposed method. The reason is that RANSAC based methods get advantage of the high points density near the real surface due to a perfect simulated registration. Nonetheless, the proposed method is 37 times faster than CFRANSAC and 41 times faster than UFRANSAC.

Second, the processing time of FDHT and RHT are close to the processing time of the proposed method. However, FDHT has 4.2 times more angular error and for RHT is 2.6 times.

Third, the number of correctly detected planes shown in Fig. 15(b) remained the highest even though the angular error was lower for CFRANSAC and UFRANSAC.

6.2 Qualitative results

Qualitative results examples are provided grouped by point cloud and method. Plane inliers are colored using a random Hue in HSV space for each plane, while outliers are colored in gray.

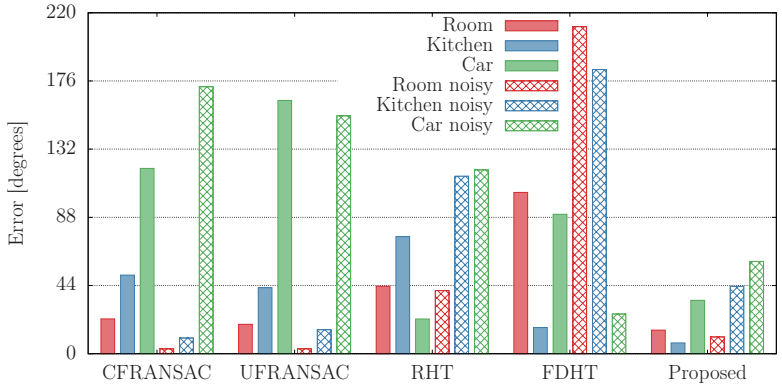
Each figure represents the results of detecting planes in a point cloud, where subfigures are the results for each of the evaluated methods.

Figure 16 – 18 show the qualitative results of detecting planes in the Room, Kitchen and Car point cloud respectively. As seen in all the subfigures, the detection results are acceptable; notwithstanding, FDHT and RHT detected a spurious plane in the pillow surface on the Room point cloud. Furthermore, RHT detected the seats upper region as part of the Kitchen table plane.

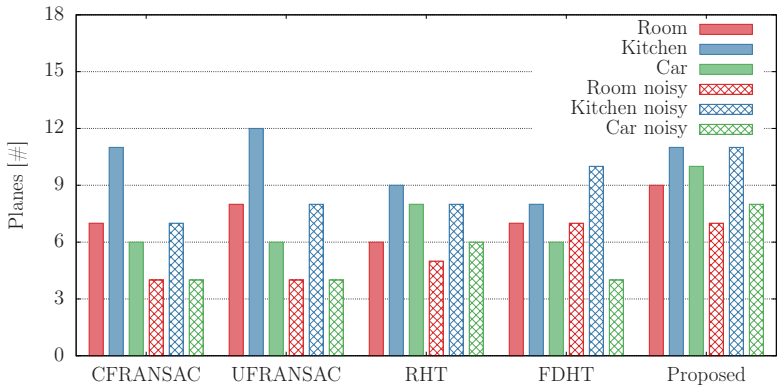
However, in the noisy results we can visualize a drop of the accuracy due to the quantization noise and artifacts.

In **Fig. 19**, CFRANSAC, UFRANSAC and FDHT did not detect the noisiest planar surface. Furthermore, RHT still detected the pillow surface as planar. The proposed method failed to detect smaller planes such as the sofa armrest and the front of the furniture, because a bigger voxel size had to be set to detect noisy planar surfaces, hence not letting the algorithm to detect smaller planes.

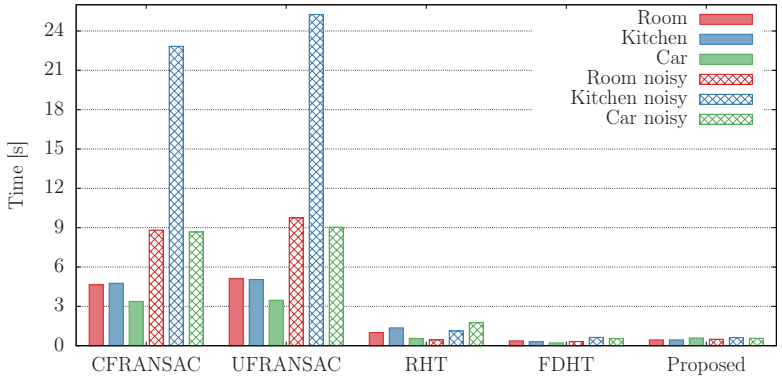
The results of the noisy Kitchen and Car point clouds were the most difficult datasets for the planes detection



(a) Error in vectorial $\mathbb{R}^4$ space, lower is better



(b) Correctly detected planes, higher is better



(c) Processing time[s], lower is better

Fig. 15 Numerical results of the evaluated methods against the datasets

algorithms.

CFRANSAC and UFRANSAC did not detect big planar portions in the noisy Kitchen point cloud of **Fig. 20**. **Figure 21** shows that RHT had it difficult to detect properly most of the planar surfaces. The reason is that RHT does not perform a plane validation using surface features such as normals or cuvatures. Although it showed less angular error than CFRANSAC and UFRANSAC, the qualitatively results were not good.

Finally, we confirmed visually the robustness of the proposed method in the most difficult scenario. It detected correctly the most prominent planar surfaces in the Car point cloud.

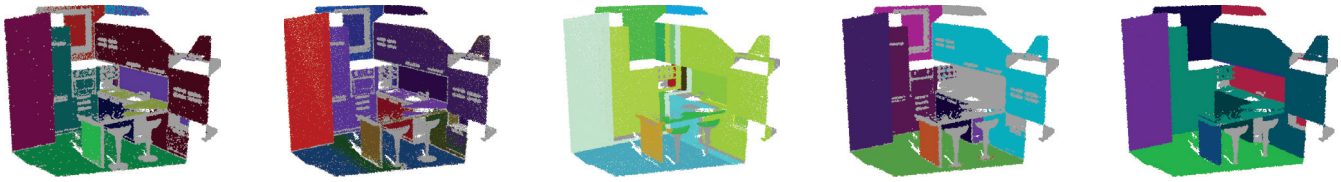
7. Conclusions and Future Works

The conventional methods for planes detection are slow or non deterministic. Moreover, their accuracy tend to drop in the presence of quantized noise and slightly curved surfaces.

Realtime applications in urban environments demand both precision and efficiency. Therefore, we proposed the Scaled Difference of Normals to filter out non-planar

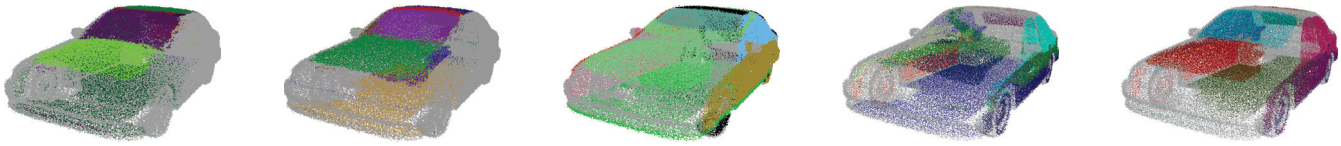
(a) CFRANSAC (b) UFRANSAC (c) RHT (d) FDHT (e) Proposed

Fig. 16 Room graphical results



(a) CFRANSAC (b) UFRANSAC (c) RHT (d) FDHT (e) Proposed

Fig. 17 Kitchen graphical results

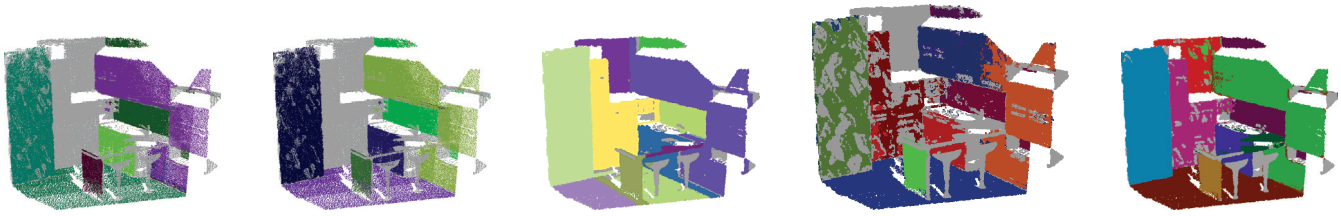


(a) CFRANSAC (b) UFRANSAC (c) RHT (d) FDHT (e) Proposed

Fig. 18 Car graphical results

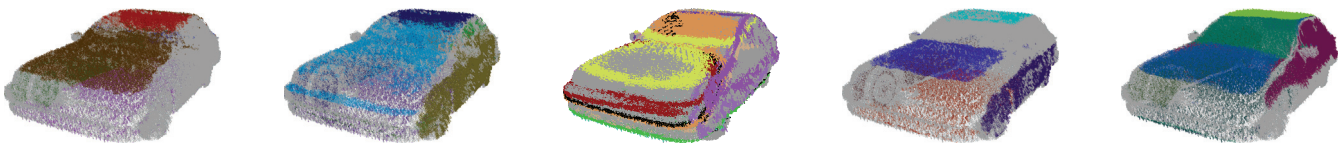
(a) CFRANSAC (b) UFRANSAC (c) RHT (d) FDHT (e) Proposed

Fig. 19 Room (noisy) graphical results



(a) CFRANSAC (b) UFRANSAC (c) RHT (d) FDHT (e) Proposed

Fig. 20 Kitchen (noisy) graphical results



(a) CFRANSAC (b) UFRANSAC (c) RHT (d) FDHT (e) Proposed

Fig. 21 Car (noisy) graphical results

points in the presence of Gaussian noise, and noise artifacts from low-cost sensors. It is a preprocessing step for FDHT, which will be provided by a reduced number of points and surface normals to decrease computational costs, and increase its accuracy.

The second contribution of this paper is an improvement of FDHT, in which a global refinement of planes is performed by validating the detected planes using the voxel centroids and their normal vectors.

To the best of our knowledge, in this paper we propose the first quantitative evaluation of 3D planes detection accuracy. With generated ground truth data, we confirmed numerically which method is good for each scenario. The results show that the proposed method is robust to difficult scenarios such as registered quantized noise and slightly curved surfaces. Additionally, the processing time remained in realtime while mantaining the accuracy high or at least competitive.

It is noticeable that even though the proposed method provides a superior balance between computational efficiency and precision. RANSAC based methods are simple and can provide good results when there are higher points density near the real surface and if processing time is not important.

A known issue of the proposed method is the sensitivity to the voxel size parameter. Even slight variations can cause FDHT and the proposed method to drop its accuracy. The reason is that the octree adapts its bounding box to fit the desired voxel size, hence causing variations in the points distribution of each voxel.

In the future we will assess 3D space subdivision or segmentation alternatives to reduce the sensitivity to the voxel size parameter, as well as to further accelerate the algorithm using GPUs.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 17K20026.

References

1) D.H. Ballard: "Generalizing the Hough Transform to Detect Arbitrary Shapes", Pattern Recognition, Vol.13, No.2, pp.111–122 (1981).
2) M.A. Fischler, C.B. Robert: "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Communications of the ACM, Vol.24, No.6, pp.381–395 (1981).
3) J. Sandoval, K. Uenishi, M. Iwakiri, K. Tanaka: "A Fast and Deterministic Plane Detection Method in Point Clouds based on Hough Transform", Proc. of VC/GCAD (2016).
4) D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter: "The 3D Hough Transform for Plane Detection in Point Clouds: A Review and a New Accumulator Design", 3D Research, Vol. 2, No.2, pp.1–13 (2011).
5) P.H.S. Torr, A. Zisserman: "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry", Computer Vision and Image Understanding, Vol. 78, No.1, pp.138–156 (2000).
6) O. Chum, J. Matas: "Matching with PROSAC-Progressive Sample Consensus", Proc. of IEEE Computer Vision and Pattern Recognition (CVPR), Vol. 1, pp.220–226 (2005).
7) C.V. Stewart: "Bias in Robust Estimation Caused by Discontinuities and Multiple Structures", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.19, No.8, pp.818–833 (1997).
8) K. Uenishi, J. Sandoval, M. Iwakiri, K. Tanaka: "Feature Description Using Planar Surfaces for 3D Virtual Keypoint", Proc. of VC/GCAD (2016).
9) L. Xu, E. Oja, P. Kultanen: "A New Curve Detection Method: Randomized Hough Transform (RHT)", Pattern Recognition Letters, Vol.11, No.5, pp.331–338 (1990).
10) P.V.C. Hough: "Method and Means for Recognizing Complex Patterns", U.S. Patent No. 3,069,654 (1962).
11) T. Hayata, M. Iwakiri: "3D Point Cloud Feature Extraction with the Difference of Centers of Gravity", Proc. of IPSJ Interaction, pp.1085–1087 (2016).
12) F.A. Limberger, M.M. Oliveira: "Real-Time Detection of Planar Regions in Unorganized Point Clouds", Pattern Recognition, Vol.48, No.6, pp.2043–2053 (2015).
13) Y. Ioannou, B. Taati, R. Harrap, M. Greenspan: "Difference of Normals as a Multi-Scale Operator in Unorganized Point Clouds", Proc. of IEEE Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission, pp.501–508 (2012).
14) R.B. Rusu, S. Cousins: "3D is here: Point Cloud library (pcl)", Proc. of IEEE International Conference on Robotics and Automation (ICRA), pp.1–4 (2011).
15) Y. Yu: "Surface Reconstruction from Unorganized Points Using Self-Organizing Neural Networks", IEEE Visualization, pp.61–64 (1999).
16) M. Gschwandtner, R. Kwitt, A. Uhl, W. Pree, BlenSor: Blender Sensor Simulation Toolbox, International Symposium on Visual Computing, Springer Berlin Heidelberg, pp.199–208 (2011).
17) A. Handa, T. Whelan, J. McDonald, A.J. Davison: "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM", Proc. of IEEE Internaitonal Conference on Robotics and Automation, ICRA (2014).
18) Marela kitchen red&white, https://3dwarehouse.sketchup.com .
19) P. Papadakis: "The Canonically Posed 3D Objects Dataset", Proc. of Eurographics Workshop on 3D Object Retrieval (2014).

(Received August 8, 2017)

Jaime SANDOVAL (*Student Member*)
He received his B.E. degree in Computer Systems Engineering from the Universidad del Valle del Fuerte (Mexico) in 2009, and his M.E. degree in Electrical and Electronic Engineering from Shinshu University in 2017. Currently, he is a Ph.D. student in the Interdisciplinary Graduate School of Science and Technology of Shinshu University with a major in Systems Development Engineering. His research interests are 3D Point Clouds Processing, Computer Vision and Image Processing.

Kazuma UENISHI (*Member*)
He received his B.E. degree in Computer Science in 2008, and his M.E. degree in Mathematics and Computer Science in 2014 from the National Defense Academy of Japan. Currently, he is enrolled as a Ph.D. student in the Interdisciplinary Graduate School of Science and Technology of Shinshu University with a major in Systems Development Engineering. He is pursuing research in 3D Point Clouds Processing.

Munetoshi IWAKIRI (*Member*)
He received his B. E. degree in Computer Science in 1993, and received his M. E. degree in Mathematics and Computer Science from National Defense Academy of Japan in 1998. In 1999, he joined Department of Computer Science, National Defense Academy of Japan, as a Research Associate. In 2002, he received Dr. Eng. degree from Keio University, Tokyo, Japan. In 2005 he became Lecturer and in 2015 he became Associate Professor in the same institution. He is pursuing research related to multimedia processing and information security. He is a member of the Information Processing Society of Japan.

Kiyoshi TANAKA (*Fellow*)
He received his B.S and M.S. degrees in Electrical Engineering and Operations Research from National Defense Academy, Yokosuka, Japan, in 1984 and 1989, respectively. In 1992, he received Dr. Eng. degree from Keio University, Tokyo, Japan. In 1995, he joined the Department of Electrical and Electronic Engineering, Faculty of Engineering, Shinshu University, Nagano, Japan, and currently he is a full professor in the academic assembly (Institute of Engineering) of Shinshu University. He is the Vice-President of Shinshu University as well as the director of Global Education Center (GEC) of Shinshu University. His research interests include image and video processing, 3D point cloud processing, information hiding, human visual perception, evolutionary computation, multi-objective optimization, smart grid, and their applications. He is a project leader of JSPS Strategic Young Researcher Overseas Visits Program for Accelerating Brain Circulation entitled Global Research on the Framework of Evolutionary Solution Search to Accelerate Innovation started from 2013. He received IEVC2010 Best Paper Award from IIEEJ, iFAN2010 Best Paper Award from SICE, GECCO2011 Best Paper Award and GECCO2015 Best Paper Award from ACM-SIGEVO, ISPACS2011 Best Paper Award from IEEE, Excellent Journal Paper Award from IIEEJ two times, in 2012 and in 2014, and Best Journal Paper Award from JSEC in 2012. He is a fellow of IIEEJ. He is a member of IEEE, IEICE, IPSJ and JSEC. He is the former editor in chief of Journal of the Institute of Image Electronics Engineers Japan as well as IIEEJ Transactions on Image Electronics and Visual Computing.