Hough Transform for Real-Time Plane Detection in Depth Images

Eduardo Vera[a,*], Djalma Lucio[b], Leandro A. F. Fernandes[a,*], Luiz Velho[b]

[a] *Instituto de Computação, Universidade Federal Fluminense (UFF), CEP 24210-240, Niterói-RJ, Brazil*
[b] *Instituto Nacional de Matemática Pura e Aplicada (IMPA), CEP 22460-320, Rio de Janeiro-RJ, Brazil*

Abstract

The automatic detection of planes in depth images plays an important role in computer vision, with applications ranging from robotics to augmented reality. Plane detection from unorganized point clouds usually require complex data structures to organize the points in 3-D space. On the other hand, existing detection approaches tailored to depth images use the structure of the image and the 2.5-D projection of the scene to simplify the task. However, they are sensitive noise and to discontinuities caused by occlusion. We present a real-time deterministic technique for plane detection in depth images. Our approach uses an implicit quadtree to identify clusters of approximately coplanar points in the 2.5-D space. The detection is performed by an efficient Hough-transform voting scheme that models the uncertainty associated with the best-fitting plane with respect to each cluster as a trivariate-Gaussian distribution. Experiments show that our approach is fast, scalable, and robust even in presence of noise and discontinuities.

Keywords: Hough transform, real-time, plane detection, depth images, pattern recognition
2010 MSC: 68U10

1. Introduction

The growing popularity of depth sensors attached to handheld devices, like Occipital's Structure Sensor [1] and Google's Project Tango [2], makes the 3-D scanning technology cheaper and ubiquitous. It also motivates the development of new automatic real-time variations of classical image processing and computer vision techniques. For instance, the ability to detect planar structures from depth images in real-time is an important step towards the solution of many practical problems, whose applications include, but are not limited to, image-based reconstruction [3, 4], camera calibration [5], autonomous vehicle navigation [6], and augmented reality [7]. State-of-the-art plane detection techniques designed for unorganized point clouds are not suitable for handling depth data, because they often require special data structures to handle points in 3-D space, or exploit non-deterministic strategies to reduce the amount of processing. In contrast, surface-growing techniques that detects planar patches in depth images are tailored to the 2.5-D structure of depth data. However, the quality of detections relies on good initial guesses for the location of seeds, on the amount of noise, and on the absence of occlusions and missing portions of depth information, which may lead to multiple detections of the same plane.

We present an efficient technique for plane detection in depth images. Our approach exploit the 2.5-D nature of depth data to organize input points in simple 2-D data structures, and takes advantage of the regular structure of the image to avoid recomputations. Our deterministic solution allows a software implementation of the algorithm to perform in real-time on a personal computer, with low memory footprint. In addition, the proposed approach is robust to noise and to multiple detections of the same plane, even in presence of occlusion and missing data. Fig. 1c shows the result obtained by our algorithm applied to the depth image in Fig. 1b. Each color in Fig. 1c represents a distinct plane detected by our approach in 22 ms on a 3.4 GHz PC.

*Corresponding author
Email addresses: eduardovera@ic.uff.br (Eduardo Vera), dlucio@impa.br (Djalma Lucio), laffernandes@ic.uff.br (Leandro A. F. Fernandes), lvelho@impa.br (Luiz Velho)
URL: http://www.ic.uff.br/~laffernandes (Leandro A. F. Fernandes), http://lvelho.impa.br (Luiz Velho)

Notice that, as expected, the coplanar doors of the balcony where detected as part of the same plane (yellow). Black regions correspond to non-planar surfaces. The input image (Fig. 1b) has 640×480 pixels with 16-bits depth. The image corresponds to frame 294 of the *Living Room* dataset [8].

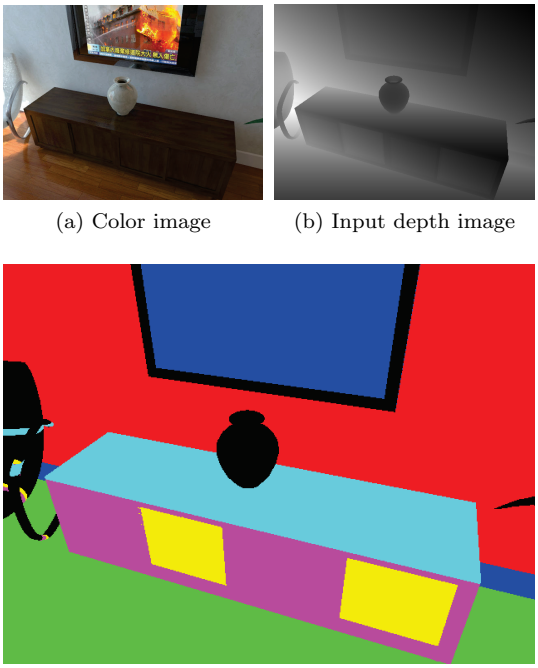
Our technique uses the Kernel-Based Hough Transform (KHT) pipeline developed by Fernandes and Oliveira [9] for straight-line detection in images, and extended by Limberger and Oliveira [10] for plane detection on unorganized point clouds. Specifically (see Fig. 2), we first subdivide the input depth image to produce clusters of approximately coplanar points. In contrast to Limberger and Oliveira's approach, whose subdivision procedure is designed for general 3-D point clouds and has total cost of $O(n \log_8 n)$, our subdivision procedure uses an implicit quadtree to organize input data in 2-D. Also, our procedure applies Summed Area Tables (SATs) to perform efficient analysis of the structure of the projected scene while refining the tree nodes, leading to $O(n)$ cost, where n is the number of input points/pixels. After the clustering step, we compute trivariate-Gaussian distributions that model the uncertainty on the best-fitting plane of each cluster (the kernels). An efficient voting scheme is performed using the kernels to cast votes for each cluster in the sparse spherical accumulator representing the parameter space of planes that can be observed in a depth image. Finally, the most significant planes in the image are retrieved as the peaks of votes in the accumulator map.

The *main contribution* of this paper is a real-time plane detection scheme for depth images: the Depth Kernel-Based Hough Transform (D-KHT), which has asymptotic time complexity $O(n)$, and whose implementation runs twice as faster than the state-of-the-art.

2. Related Works

The Hough transform (HT) [11] is one of the most popular techniques for detecting geometric entities in low-dimensional spaces. It was originally proposed for straight-line detection in binary images but it was further extended for detecting other structures. Duda and Hart [12] proposed the use of the normal equation of the line as parametrization model of the Standard Hough Transform (SHT) for line detection:

$$\rho = x \cos\theta + y \sin\theta, \qquad (1)$$



(a) Color image (b) Input depth image

(c) Detected planes

Figure 1: Result produced by the proposed approach (best viewed in color): (a) Color image of the scene for; (b) Input depth image consisting of 640×480 pixels with 16-bits depth; and (c) Detected planes, identified by colors. Black pixels in (c) correspond to non-planar object. Note that the algorithm behaves well, even on presence of non-planar surfaces, like the vase, the chair, the plant, and the TV frame.

where $(x, y)^T$ correspond to the coordinates of a point on the straight line $(\rho, \theta)^T$, $\rho \in [-R, +R]$ is the distance of the line relative to the center (origin) of the image, and $\theta \in [0, \pi)$ is the angle from the x-axis to line's normal direction. $R = \sqrt{w^2 + h^2}/2$, where w and h correspond to the image's width and height, respectively. Given a binary edge image, for each edge pixel $(x', y')^T$, the SHT identifies all $(\rho, \theta)^T$ pairs satisfying (1), *i.e.,* all lines passing through that pixel, by arbitrating discrete values to θ and computing ρ using (1). In the SHT, the space defined by the line parameters ρ and θ, *i.e.,* the parameter space, must be properly discretized as an accumulator map. The accumulator's bins related to the lines passing through each edge pixel are incremented by one unit per voting pixel. The local maxima created by this voting process correspond to the parameters of the most likely lines in the image. The discretization step assumed for ρ and θ has direct influence on the execution time, memory footprint, and quality of detections.

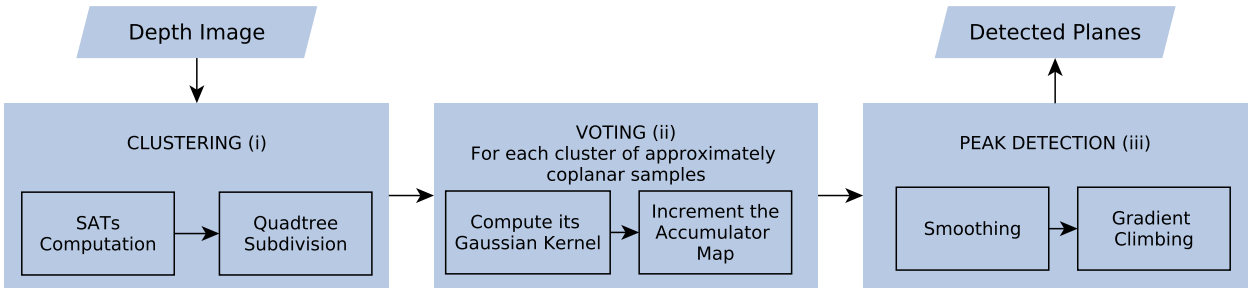


Figure 2: D-KHT workflow: (i) We take a depth image as input and compute some SATs to ensure efficient processing during the clustering step. Then, we use an implicit quadtree to identify rectangular image regions having approximately coplanar points in 3-D. (ii) For each region identified in step (i), we calculate a Gaussian distribution describing the uncertainty of the best-fitting plane for the set of points in the region, and use the Gaussian distribution to increment the spherical accumulator map. (iii) Finally, we use a gradient uphill climbing approach to find peaks of votes in the accumulator map. The coordinates of the peaks correspond to the most likely planes in the image.

Fernandes and Oliveira [9] realized that the performance of the SHT for line detection could be greatly improved by grouping edge pixels and voting for each group instead of for each edge pixel. The Kernel-Based Hough Transform (KHT) algorithm first extracts clusters of approximately collinear pixels from the edge image, and them uses the pixels of each cluster as samples to fit the Gaussian distribution of uncertainty of the line that better explains that cluster, *i.e.,* the Gaussian kernel. Voting is performed by adding the contribution of each kernel to the accumulator map. Wide kernels are neglected from the voting process because they represent sets of ill-defined collinear pixels. The KHT was the first HT-based solution to achieve robust real-time line detection for high-resolution images, even in presence of noise.

Limberger and Oliveira [10] extended the KHT to the detection of the planes that better fit an unorganized point cloud in 3-D space. For plane detection, the line model (1) is naturally replaced by the normal equation of the plane:

$$\rho = x \sin\phi \cos\theta + y \sin\phi \sin\theta + z \cos\phi, \qquad (2)$$

where $(x, y, z)^T$ are the Cartesian coordinates of a point on the plane $(\rho, \phi, \theta)^T$, $\rho \in \mathbb{R}^+$ is the non-negative distance from the plane to the origin, and $\phi \in [0, \pi)$ and $\theta \in [0, 2\pi)$ are the angular coefficients of the plane. The 3-D KHT does not assume the pre-organization of input points. Thus, the authors propose the use of an octree to subdivide the point cloud in such a way that each leaf node of the tree includes a cluster of approximately coplanar points, or a set of outliers. The latter leaves are neglected in the voting process. The 3-D KHT assumes a spherical parameter space for planes and, by extending the ideas of the KHT, the accumulator map is updated by adding the contribution of the trivariate-Gaussian distribution that models the uncertainty of the best-fitting plane of each cluster. Experiments show that the 3-D KHT outperforms previous HT-based techniques for plane detection in point clouds [10].

Vosselman *et al.* [13] also presented a HT-based technique for plane detection. The authors split the detection process in two steps. First, they explore the connectivity of point clouds acquired with laser scanners to calculate the normal vector of the most likely plane passing through each input point. Then, the most frequent plane orientations are identified by voting into the 2-D parameter space of angular coefficients of (2). Finally, the points assigned to the most frequent orientations vote for the distance of the plane to the origin in an 1-D parameter space. Vosselman's *et al.* approach has low memory footprint. However, it is slower than the 3-D KHT when applied to unorganized point clouds. In addition, the use of a single direction per point makes it not robust to high levels of noise. A similar proposal is presented by Nguyen *et al.* [14]. The main differences are how they compute normal vectors and identify the most likely orientations in the first parameter space.

The Random Sample Consensus (RANSAC) [15] is a non-deterministic iterative method for model's parameters adjust. Its main advantage is noise resilience. Unfortunately, only one instance of the desired model is detected per execution, and most of the computational cost is dedicated to handle models computed from unrelated input entries.

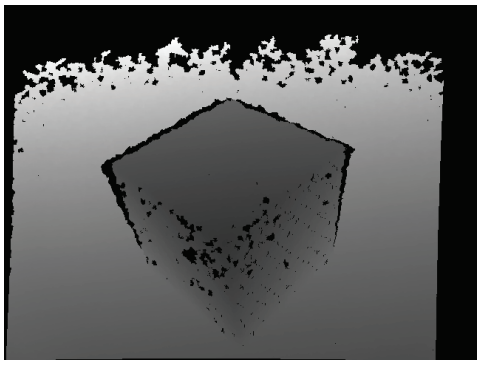
Schnabel *et al.* [16] introduced an optimization to RANSAC using an octree to establish spatial proximity among samples, improving the detection of planes, spheres, cylinders, cones and tori from points in 3-D spaces. However, despite overcome most of the shortcomings of conventional RANSAC, Schnabel's *et al.* approach was shown to be less efficient than the 3-D KHT [10].

Fernandes and Oliveira [17, 18] presented a general voting-based framework for detecting data alignments in large unordered noisy multidimensional datasets. In contrast to classical techniques such as the HT and RANSAC, which are designed for detecting a specific type of alignment on a given type of input, their approach is independent of the geometric properties of the alignments to be detected, as well as independent of the type of input data. Thus, it can be used to plane detection from both exact and uncertain data. However, its performance is still far from real-time.

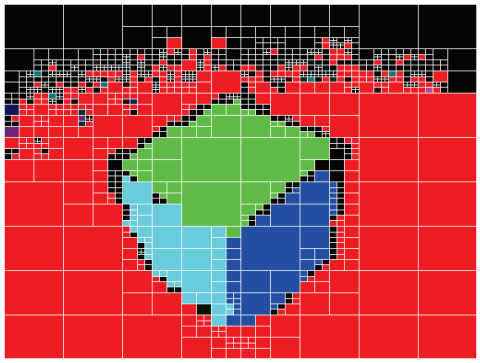
Surface Growing (SG) [19, 20, 21, 22, 23] is a Region Growing [24] technique that detects planar patches by using a smoothness constraint in depth images. It consists in select points on the image (seeds), analyze the seeds' neighborhood, and expand the related patch while the surface keeps its smoothness. In other words, if a data point is inside a relevance threshold with respect to a plane, its neighbors will be evaluated under the same criteria. Although SG works well in presence of a small amount of noise, it has a high computational cost and the quality of detection heavily relies on the selection of the seeds. Thus, the technique is sensitive to a good initial guess. Furthermore, SG requires connectivity between pixels to avoid multiple detections of the same plane, which is not always observed in depth images, since damaged and missing portions of depth information may be caused by occlusion, depth shadowing, and reflective, refractive, and light-absorbing materials [25].

3. Depth Kernel-Based Hough Transform

Fig. 2 illustrates the flowchart of the proposed plane detection pipeline: (i) starting from a depth image, our algorithm uses SATs [26] and a quadtree [27] to efficiently subdivide the set of pixels into clusters of approximately coplanar points, in 3-D (Section 3.1); (ii) for each cluster, it uses first-order error propagation [28] to estimate the uncertain plane that fits clustered points (Section 3.2); then, our approach votes in a spherical accumulator map (Section 3.3) for the main planes using trivariate-Gaussian kernels computed from the planes associated uncertainties (Section 3.4); and (iii) after the voting procedure, the location of local maxima in the accumulator map correspond to the most likely planes in the depth image (Section 3.5).



(a) Input depth image



(b) Quatree and detected planes

Figure 3: The Cube dataset consists of the depth image of a cube on a planar surface (a). It was acquired using a Structure Sensor. Black pixels in (a) depict invalid depth information. Thus, those pixels do not correspond to 3-D point samples. Image (b) shows the leaf nodes of the quadtree built over the depth image. Nodes having the same color indicate clusters of point samples that voted to the same detected plane. Black nodes in (b) correspond to outliers. They do not participate in the voting process.

3.1. Clustering depth data

The use of depth cameras imposes two restrictions over data captured from a particular viewpoint of the scene. For instance, just like ordinary cameras, depth cameras does not register occluded surfaces. Therefore, the depth image is a height map from the camera's perspective, which allows

the organization of 3-D data points (*i.e.,* the samples, one per valid pixel) considering only the 2-D coordinate system of the image. The second restriction is the regular organization of pixels. We use both restrictions to build an implicit quadtree whose rectangular leaves define clusters of either approximately coplanar points or sets of outliers (see Fig. 3). In addition, computations in each tree node are efficiently evaluated using SATs computed before building the quadtree.

The clustering procedure is presented by Algorithm 1 as the recursive construction of a quadtree on the depth image. The first call to `Clustering()` receives as input a node that includes the whole image. The threshold value s_{ms} (used in line 2) indicates the minimum number of samples (*i.e.,* valid depth pixels, depicted by shades of gray in Fig. 3a) that the rectangular region related the node must include to be considered a candidate cluster. The threshold value s_t (used in line 5) indicates the maximum spread a distribution of samples may have in its thinnest dimension to be considered coplanar. Thus, s_t must be defined according to the amount of details one expects to retrieve from the scene. Higher values lead to less but ticker planes (less resolution). Lower values usually retrieve more planes (more resolution). The values assumed for s_{ms} and s_t in our examples are reported in Section 4.

For each node having enough samples, we use the Principal Component Analysis (PCA) [29] of the distribution of samples centered in the mean sample (lines 3 and 4) to decide whether the current set of points define a cluster of coplanar samples (line 6) or must be subdivided according to the quadrants of the current rectangular image region represented by the node (lines 8 to 11). If a node does not have samples enough, it is a leaf node and its samples are considered outliers (line 13). In this case, the node and its samples will not participate of the voting process. Fig. 3b depicts outliers as black nodes.

When using PCA on depth images, one must be aware that the depth value z_{ij} in each pixel $(i,j)^T$ is usually given in the standard unit of measurement assumed by the capturing device. Pixels coordinates $(i,j)^T$, on the other hand, assume a different unit of measure. Usually, non-negative integer values ranging from zero to the width and height of the image, respectively. Thus, in order to compute the covariance matrix in a 3-D space having the same metric in all dimensions, we have to map valid pixels to the actual 3-D space of the scene, up to uniform scale. The 3-D coordinated of a given sample

Algorithm 1: Recursive subdivision of the quadtree for clustering samples.

```
1  Procedure Clustering(node)
       Input: node represents the boundaries of
              the image region associated to the
              current tree node
       Data: s_ms is the minimum number of
             samples required in a cluster; and s_t
             is the maximum thickness of a plane
       Result: Updates the list of clusters
2      if node includes at least s_ms samples then
3          Σ_(x,y,z) ← the covariance matrix of
            samples in node;
4          V_(x,y,z), Λ_(x,y,z) ← the eigenvectors and
            eigenvalues of Σ_(x,y,z), where λ_1 is the
            smallest eigenvalue;
5          if 2√λ_1 < s_t then
6              node is a leaf node. Put Σ_(x,y,z),
                V_(x,y,z), and the limits of node in
                the list of clusters;
7          else
8              foreach quadrant of node do
9                  Make a child node with the
                    boundaries of the quadrant;
10                 Call Clustering(child);
11             end
12         end
13     else
           /* node is a leaf node and its
              samples are outliers.        */
14     end
15 end
```

in the unit of measurement used by the device is given by:

$$\begin{pmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{pmatrix} = \begin{pmatrix} \frac{i-c_x}{\alpha_x} z_{ij} \\ \frac{j-c_y}{\alpha_y} z_{ij} \\ z_{ij} \end{pmatrix}, \tag{3}$$

where $(c_x, c_y)^T$ is the camera's principal point, in pixels, and α_x and α_y represent focal length in terms of pixels. In (3), we assume a pinhole camera having skew equal to zero. In our experiments, the camera parameters were retrieved from the datasets or from the calibration procedure of the device.

Using the coordinates of the samples in the rectangular region $\mathcal{R}$ related to a given node, the mean point and the covariance matrix of the samples are

computed, respectively, as:

$$\mu_{(x,y,z)} = \begin{pmatrix} \mu_x \\ \mu_y \\ \mu_z \end{pmatrix}, \Sigma_{(x,y,z)} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix}, \tag{4}$$

where

$$\sigma_{ab} = \frac{1}{m-1}\left(\sum_{k=1}^{m} a_k\, b_k - \mu_b \sum_{k=1}^{m} a_k - \mu_a \sum_{k=1}^{m} b_k + m\,\mu_a\,\mu_b\right), \tag{5}$$

and

$$\mu_a = \frac{1}{m}\sum_{k=1}^{m} a_k, \quad \mu_b = \frac{1}{m}\sum_{k=1}^{m} b_k. \tag{6}$$

In (5) and (6), m is the number of samples in $\mathcal{R}$, and $\{a_k\}_{k=1}^{m}$ and $\{b_k\}_{k=1}^{m}$ are the coordinates x, y, or z of samples (3). In order to keep the expressions simpler, we have replaced the double-indexing notation $\square_{ij}$ from (3) by single indexing $\square_k$ in (5) and (6). Notice that m may be less than or equal to the number of pixels in $\mathcal{R}$.

The computation of $\Sigma_{(x,y,z)}$ using the expression presented in (4) takes $O\left(n^2\right)$. However, by precomputing SATs for images having x_{ij}, y_{ij}, z_{ij}, x_{ij}^2, y_{ij}^2, z_{ij}^2, $x_{ij}\,y_{ij}$, $x_{ij}\,z_{ij}$, and $y_{ij}\,z_{ij}$ values in their pixels, we have reduced the computation time of each covariance matrix to $O\left(1\right)$, while the computation of the nine SATs is $O\left(n\right)$. A linear algorithm for computing SATs is presented in [26]. The summation of values in a rectangular region of the image (each summation in (5) and (6)) is evaluated in constant time using four SAT references, and three arithmetic operations [26].

The eigenvectors $V_{(x,y,z)} = \{\overrightarrow{v}_1, \overrightarrow{v}_2, \overrightarrow{v}_3\}$ and eigenvalues $\Lambda_{(x,y,z)} = \{\lambda_1, \lambda_2, \lambda_3\}$ computed in Algorithm 1 (line 4), with $0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3$, describe the distribution of samples in 3-D space. By taking the squared root of the smallest eigenvalue ($\sqrt{\lambda_1}$), we have one standard deviation of the distribution of samples along its least significant principal component ($\overrightarrow{v}_1$). In Algorithm 1 (line 5) we check whether two standard deviations in this component includes the s_t threshold. Assuming Normal distribution in the $\overrightarrow{v}_1$ direction, this gives a 95.4% assurance that the average plane to be adjusted to this set of points has a tubular neighborhood around it with at most two times the width defined by s_t.

Limberger and Oliveira [10] use an octree to create clusters of samples. The octree subdivision is guided by two parameters: s_α and s_β, which are the relative tolerances associated with, respectively, sample's thickness and isotropy inside a cluster. According to our experiments, the 2.5-D nature of depth images makes the definition of relative isotropy a challenging task, since the single-layer spread of feature points in 3-D is influenced by the relative angle between the principal axis of the camera and the normal of the plane. Also, we believe that handling a single parameter (s_t) in the same unit of measurement assumed by the device is more intuitive to the user.

3.2. Computing Gaussian Kernels

Let $\mathcal{C}$ be a cluster containing approximately coplanar point samples in a leaf node of the quadtree (the m valid pixels inside $\mathcal{R}$), whose unit eigenvectors $V_{(x,y,z)} = \{\overrightarrow{v}_1, \overrightarrow{v}_2, \overrightarrow{v}_3\}$ and eigenvalues $\Lambda_{(x,y,z)} = \{\lambda_1, \lambda_2, \lambda_3\}$ where computed in Algorithm 1 (line 4). The plane that better fit the samples in $\mathcal{C}$ has normal vector $\overrightarrow{n} = \overrightarrow{v}_1 = (n_x, n_y, n_z)^T$ and passes through the centroid $\mu_{(x,y,z)}$ (4) of the points. This plane is mapped to the spherical parameter space using

$$\mu_{(\rho,\phi,\theta)} = \begin{pmatrix} \mu_\rho \\ \mu_\phi \\ \mu_\theta \end{pmatrix} = \begin{pmatrix} n_x\,\mu_x + n_y\,\mu_y + n_z\,\mu_z \\ \cos^{-1}\left(n_z\right) \\ \tan^{-1}\left(\frac{n_y}{n_x}\right) \end{pmatrix}. \tag{7}$$

Thus, the Gaussian kernel that weights votes from $\mathcal{C}$ will be centered at the parameters $\mu_{(\rho,\phi,\theta)}$ of the plane that better fits the points in the cluster, *i.e.,* the mean plane.

The covariance matrix of the Gaussian kernel can be obtained from the covariance of input samples (4) using first-order error propagation:

$$\Sigma_{(\rho,\phi,\theta)} = \begin{pmatrix} \sigma_{\rho\rho} & \sigma_{\rho\phi} & \sigma_{\rho\theta} \\ \sigma_{\rho\phi} & \sigma_{\phi\phi} & \sigma_{\phi\theta} \\ \sigma_{\rho\theta} & \sigma_{\phi\theta} & \sigma_{\theta\theta} \end{pmatrix} = J\,\Sigma_{(x,y,z)}\,J^T, \tag{8}$$

where J is the Jacobian matrix of (7):

$$J = \begin{pmatrix} \partial_x\rho & \partial_y\rho & \partial_z\rho \\ \partial_x\phi & \partial_y\phi & \partial_z\phi \\ \partial_x\theta & \partial_y\theta & \partial_z\theta \end{pmatrix} = \begin{pmatrix} n_x & n_y & n_z \\ \frac{n_x\,n_z}{\rho\,\omega} & \frac{n_y\,n_z}{\rho\,\omega} & -\frac{\omega}{\rho} \\ -\frac{n_y}{\omega} & \frac{n_x}{\omega} & 0 \end{pmatrix},$$

with $\omega = \sqrt{n_x^2 + n_y^2}$. See [10] for details.

If all the samples of the node are perfectly aligned over a plane, *i.e.,* without any noise, we would obtain $\sigma_\rho = 0$, which would make $\Sigma_{(\rho,\phi,\theta)}$ singular. In order to avoid handling this specific case during the

voting procedure, in our experiments we add a small value $\varepsilon = 0.001$ to σ_ρ. This practice does not lead to implications on results but avoids singularity.

3.3. Spherical Accumulator Map

Borrmann *et al.* [30] demonstrated that the use of an unbiased spherical accumulator map $\mathcal{A}$ is ideal for HTs tailored to detect arbitrary planes in 3-D spaces. They claim that naive accumulator maps favor geometrical objects with certain parameters due to uneven sampling of the parameter space. Borrmann's *et al.* map, on the other hand, is designed to assign approximately the same importance to each cell of $\mathcal{A}$. In D-KHT, we rely on Borrmann's *et al.* proposal to build a map having unbiased cells. In our implementation, the spherical parameter space defines planes whose normal vectors point to the opposite direction of the origin of the 3-D Cartesian space. We take the camera's center and its principal axis as, respectively, the origin and the z-axis of the space. Thus, we define $\theta \in [-\pi, +\pi)$ with its discretization given as function of the elevation angle ϕ, restricted to $\phi \in [0, \pi)$. The amount of accumulators toward θ-axis is [30]:

$$N_\theta(\phi) = \max(1, 2\, N_\phi\, \sin(\phi)), \tag{9}$$

where N_ϕ corresponds to the number of accumulators in ϕ-axis. We assume linear discretization for both ϕ and θ axes of $\mathcal{A}$. It is important to note that the domain we are working on is cyclical. One must take this observation into account while performing the voting and the peak detection procedures described, respectively, in Sections 3.4 and 3.5.

The $\rho \in [0, \rho_{\max}]$ parameter is related to the distance of the plane to the origin. Therefore, its upper limit defines the radius of the hemispherical parameter space. We choose $\rho_{\max}$ as the distance between the camera and the farthest point sample. The discrete values assumed by ρ in $\mathcal{A}$ are defined by linear interpolation of the interval $[0, \rho_{\max}]$ into N_ρ samples. The values of N_ρ and N_ϕ (used in (9)) are defined by the user according to his/her expectations on the granularity of detection results. Section 4 indicates the values assumed in our experiments.

We follow Limberger and Oliveira and keep the accumulator as an associative map that binds each possible pair of discrete values $\{\phi, \theta\}$ to a 1-D array representing the possible values of ρ. All key entries of the associative map are created before the voting process, and the third dimension (ρ) is allocated and bonded as needed during the voting procedure. Such a construction ensures the sparse occupation of the parameter space, leading to the reduction of total memory usage.

3.4. Kernel-Based Voting

We use the Gaussian kernels computed according to Section 3.2 to update the spherical accumulator map $\mathcal{A}$ described in Section 3.3. For a given kernel defined by a mean parameter vector $\mu_{(\rho,\phi,\theta)}$ (7) and a covariance matrix $\Sigma_{(\rho,\phi,\theta)}$ (8), we increment the bins of $\mathcal{A}$ which are within the two standard deviation ellipsoid of the trivariate-Gaussian distribution of the kernel. Therefore, the contribution of a kernel is considered significant only at the bins whose parameter vector $q = (\rho, \phi, \theta)^T$ satisfies:

$$f(q) \geq f\left(\mu_{(\rho,\phi,\theta)} + 2\sqrt{\kappa}\,\overrightarrow{u}\right), \tag{10}$$

where $\{\overrightarrow{u}, \kappa\}$ is any eigenpair of $\Sigma_{(\rho,\phi,\theta)}$, and f denotes the probability density function (PDF) of the Gaussian distribution [31]:

$$f(q) = \frac{1}{\sqrt{(2\pi)^3\left|\Sigma_{(\rho,\phi,\theta)}\right|}} \exp\left(-\frac{1}{2}\,\delta^T\,\Sigma^{-1}_{(\rho,\phi,\theta)}\,\delta\right), \tag{11}$$

for $\delta = q - \mu_{(\rho,\phi,\theta)}$. $|\square|$ and $\square^{-1}$ denote, respectively, the determinant and the inverse of a matrix. Notice that the right side of (10) is a different constant value for each kernel.

Since $\mathcal{A}$ is a discrete domain, we start voting at the bin that includes $\mu_{(\rho,\phi,\theta)}$, whose address is:

$$\rho_{index} = \left\lfloor \frac{\mu_\rho}{\rho_{\max}}\,(N_\rho - 1) \right\rfloor, \tag{12a}$$

$$\phi_{index} = \left\lfloor \frac{\mu_\phi}{\pi}\,N_\phi \right\rfloor, \tag{12b}$$

$$\theta_{index} = \left\lfloor \frac{\mu_\theta + \pi}{2\,\pi}\,N_\theta(\mu_\phi) \right\rfloor, \tag{12c}$$

where $\lfloor\square\rfloor$ denotes the floor function, and $N_\theta(\square)$ is defined by (9). Neighbour bins are reached in a flood-fill fashion, having the condition (10) as stopping criteria for flooding.

It is important to note that the density of points in the clusters varies with camera proximity, while the integral of (11) is one due to the normalization axiom of probability [31]. Therefore, kernel votes in each bin must be composed by multiplying the evaluation of (11) to the $w_\mathcal{C}$ factor (13) computed for a given cluster $\mathcal{C}$ regarding the spatial coverage of its tree node and the number of samples. Similar

to Limberger and Oliveira [10], but with appropriate adjustments for our clustering model, we define

$$w_{\mathcal{C}} = w_a \frac{\mathcal{R}_{\text{area}}}{\mathcal{I}_{\text{area}}} + w_d \frac{m}{n}, \tag{13}$$

where $\mathcal{I}_{\text{area}}$ and $\mathcal{R}_{\text{area}}$ are the areas (in pixels) of the image and of the rectangular region $\mathcal{R}$ of the node, respectively. n is the number of samples in the whole image, and m is the number of samples in the cluster. The values of w_a and w_d are restricted to $w_a + w_d = 1$. In our experiments, we confirm the findings in [10] that better results are reached by using $w_a = 0.75$ and $w_d = 0.25$. Thus, we favor number of pixels (area) against number of samples.

3.5. Detecting Peaks of Votes

After the voting process, the peaks of votes in the accumulator map correspond to the detected planes (Fig. 2, step (iii)). It is essential, however, to apply a low-pass filter to consolidate local maxima [9] before searching for local maxima. In our experiments, we computed the convolution of the accumulator map and a six-connected filter with central weight equals to 0.2002 and neighbor weights equal do 0.1333. This filtering operation smooths the voting map, helping to consolidate adjacent peaks as single detected planes.

We apply a gradient climbing strategy to detect peaks of votes in the smoothed accumulator map. For each kernel used in the voting procedure (Fig. 2, step (ii)), we take the accumulator bin addressed by the parameters of the mean plane and check whether this bin corresponds to a local maximum. If the condition is true, the bin is marked as a detected plane. Otherwise, we took the neighbor bin having more votes than the current bin and repeat the process until find a local maximum.

Once we have the parameters of each plane detected by gradient climbing, the next step is to determine the relevance of plane to the dataset. In this work, such relevance is a function of the amount of input points belonging to the plane. Thus, given the parameter vector $(\rho, \phi, \theta)^T$ of a plane, its normal vector $\overrightarrow{n}$ is computed as:

$$\overrightarrow{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \begin{pmatrix} \sin\phi\,\cos\theta \\ \sin\phi\,\sin\theta \\ \cos\phi \end{pmatrix}. \tag{14}$$

From $\overrightarrow{n}$ and ρ one can extract the relation of pertinence of a point $p = (x, y, z)^T$ to a plane given a threshold s_d, just by satisfying:

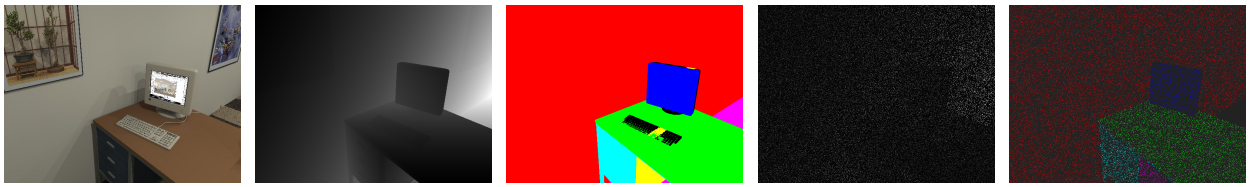
$$|n_x\, x + n_y\, y + n_z\, z - \rho| \leq s_d. \tag{15}$$

Thus, we determine the relevance of a given detected plane based on the amount of points in the depth image that belongs to that plane. Such a verification can be efficiently evaluated by checking only the data points related to the kernels that have climbed to a given local maximum. We have observed that this simple counting strategy lead to better ordering than the use of the number of votes in noisy datasets.

3.6. Time Complexity Analysis

Our plane detection approach is comprised of three steps having linear asymptotic time complexity each. Therefore, the time complexity of the D-KHT is also $O(n)$. Without loss of generality, let's assume an input depth image with $d \times d$ pixels and $n = d^2$ point samples for $d \geq 2$ being power of two. According to Section 3.1, we first compute nine SATs and them we build an implicit quadtree to subdivide the image into clusters of approximately coplanar samples. The cost of computing each SAT is $O(n)$ (see [26]). By assuming the worst-case scenario, the quadtree would subdivide the samples into $n/2$ clusters (tree leafs) defined by 2×2 image regions. In that case, the height of the tree would be $\log_4 n$, leading to $\sum_{l=1}^{log_4 n} 4^{l-1} = (n-1)/3$ nodes. According to Algorithm 1, at each node we have to compute a 3×3 covariance matrix $\Sigma_{(x,y,z)}$ and its eigen decomposition. The SATs allow the computation of $\Sigma_{(x,y,z)}$ in $O(1)$ time. Decomposing $\Sigma_{(x,y,z)}$ is accomplished in $O(1)$ time, too. Therefore, the operations in the whole clustering procedure are performed in $O(n)$.

The voting procedure consists on apply first-order error propagation to compute the trivariate-Gaussian distribution of the uncertain plane of each cluster and update the accumulator array using those distributions (Section 3.4). The first operation is clearly performed in time $O(1)$, per cluster. The number of accumulator bins updated by each cluster depends on the distribution of samples in the cluster, and on the resolution of the accumulator. Nevertheless, it is typically much smaller than n. Therefore, we can safely handle the number of updated bins as a constant value, leading to time $O(1)$, per cluster. In worst case we have $n/4$ clusters. Hence, the asymptotic time for the voting procedure is $O(n)$.



(a) Color image (b) Original depth (c) Detections from (b) (d) Noisy depth (e) Detections from (d)

Figure 4: Frame 2453 from the *Office* dataset illustrates the noise resilience of the D-KHT (best viewed in color). From left to right: (a) color image; (b) original depth image; (c) planes detected in (b); (d) depth image (b) with 90% of random-valued impulsive noise; and (e) planes detected in (d). Notice that the D-KHT retrieves the main planes of the scene even from considerably corrupted depth image. We used $N_\rho = 200$, $N_\phi = 160$, $s_{ms} = 30$, $s_t = 0.5$, and $s_d = 30$ in those examples.

Finally, filtering any given accumulator bin is performed in $O(1)$ time, for a total cost of $O(n)$ because we only have to filter bins that received votes. Gradient climbing is accomplished for each cluster (see Section 3.5), leading to a small amount of accesses to accumulator bins per climbing process (much smaller than n). Thus, the time complexity of the peak detection procedure is $O(n)$, and the overall time complexity of the D-KHT is $O(n + n + n) = O(n)$.

4. Experiments and Results

We have compared the performance of the D-KHT against the state-of-the-art by using our C++ implementation of the proposed approach and the reference C++ implementation of the 3-D KHT provided by Limberger and Oliveira. Both implementations use `dlib` [32] for computing eigendecomposition. However, while the 3-D KHT uses OpenMP to parallelize the clustering procedure, our implementation is strictly sequential. We have not compared the D-KHT against nondeterministic techniques, such as RANSAC and SG, because the former returns only one plane per execution and the latter requires good seeds and does not handle depth discontinuity. Also, both are not as fast as the 3-D KHT [10]. The C++ codes where compiled with GCC (version 4.8.4) and the experiments were performed on an Intel Core i7-4770 CPU with 3.4 GHz and 16 Gb of RAM. The codes were compiled for 64-bits architecture to exploit the full potential of the hardware.

We have used depth images from four different datasets in our experiments. Two synthetic datasets were provided by Choi *et al.* [8]: the *Living Room* and the *Office*, whose frames 294 and 2453 are presented in Figs. 1 and 4, respectively. Synthetic datasets do not include noise nor systematic errors introduced by the capturing device. We also used two real datasets: the *Copy Room* and the *Cube*. The former was captured by Zhou and Koltun [33] with an Asus Xtion PRO LIVE camera. Fig 5 shows frames 1063, 1791, 2297, and 4161 from the *Copy Room* dataset. The *Cube* dataset was captured by us using a Structure Sensor. Its single frame is presented in Fig. 3. Real datasets are comprised of depth images with optical distortions introduced by the device, noise, as well as damaged and missing portions of depth information. All depth images have 640×480 pixels with 16-bits depth, which provides 65.536 levels of sensitivity. We use (3) to convert depth images into input point clouds to the 3-D KHT.

A detailed account for the times involved in each step of the proposed plane detection strategy is presented in Table 1 for seven different shown in Figs. 1, 3, 4b and 5, and in Tables 2 and 3 for, respectively, noisy variations a synthetic (Fig. 4b) and a real (Fig. 5, third column) scene. The complexity of these scenes are indicated by their number of point samples (see the *Samples* column in Table 1) and distribution of planar and non-planar surfaces, which is depicted in their respective depth images. The execution times (in milliseconds) were computed as an average of 50 executions and the parameters were defined as shown in figure captions.

Table 1 shows that the proposed approach can achieve up to 500 fps for real depth images (*e.g., Cube dataset*, frame 1) of simpler scenes, and performance ranging from 33 fps (*e.g., Copy Room* dataset, frame 1791) to 240 fps (*e.g., Copy Room* dataset, frame 4161) for more complex real datasets.

Table 1: Comparison of the performance of the D-KHT and the 3-D KHT regarding all examples presented in this paper.

Input		D-KHT						3-D KHT					
Image	Samples	Clustering	Voting	Peaks	Total	Clusters	Planes	Clustering	Voting	Peaks	Total	Clusters	Planes
Living Room, frame 294	307,200	4.3496	1.6476	0.6939	6.6911	627	4	161.1130	10.0070	3.0020	174.1220	48	5
Office, frame 2453	307,200	1.8626	6.1292	3.7548	11.7466	527	6	147.0960	229.1740	120.6020	496.8720	298	6
Copy Room, frame 1063	260,636	0.7628	14.0387	12.9332	27.7348	234	5	31.0210	66.0470	68.0490	165.1170	9	7
Copy Room, frame 1791	271,359	0.8615	14.5953	14.2071	29.6639	279	5	41.0290	212.1520	138.1070	391.2880	13	7
Copy Room, frame 2297	266,917	0.5042	3.7146	2.7238	6.9425	182	8	26.0080	347.2580	284.2030	657.4690	28	8
Copy Room, frame 4161	270,089	1.3045	1.4618	1.3610	4.1273	313	4	32.0220	509.3650	266.2030	807.5900	6	4
Cube, frame 1	225,296	0.7010	0.7220	0.5570	1.9800	181	4	31.5770	57.0390	40.0280	128.6440	17	4

To evaluate the accuracy and the noise resilience of our approach, we have replaced depth values from dataset images by random-valued impulsive noise ranging from 0 to 10^4. Then, we compared the detection results from a given original depth image (0% level noise) and its noisy counterparts at 10%, 20%, $\cdots$, and 90% level noise. For instance, Figs. 4b and 4d show frame 2453 of the *Office* dataset with 0% and 90% level noise, respectively. Notice that the geometrical structure of the scene is almost indistinguishable in Fig. 4d. However, our approach was capable to correctly detect the main planes in both cases, as can be observed in Figs. 4c and 4e. The only misdetection is the plane of the paintings, which was merged to the plane of the wall in Fig. 4d. Black pixels in Figs. 4c and 4e correspond to point entries filtered out by our technique.

Limitations: As any HT-based approach that uses a discrete accumulator map, the performance and robustness of our technique are constrained to the appropriate discretization of the parameter space and to scene complexity. Furthermore, the presence of non-planar surfaces in the scene may lead to the detection of spurious planes. Fortunately, those planes often have lower importance, being discarded with the appropriate choice of a suppression threshold. Parallel close planes may be retrieved as a single instance in noisy datasets. Finally, depth cameras restricts the set of planes that can be observed in the scene (*e.g.,* planes behind the camera cannot be seen). Thus, the parameter space we have adopted includes regions that are never reached by the voting procedure.

[1]The ϕ_{max}, ρ_{max} and s_{ms} parameters of the 3-D KHT correspond to, respectively, N_ϕ, N_ρ, and s_{ms} of the D-KHT. s_α and s_β are the relative tolerances associated with plane thickness and isotropy. s_{level} is the first octree level for checking for approximate coplanarity.

5. Conclusion and Future Works

We have presented an $O(n)$ HT-based approach for real-time plane detection in depth images, where n is the number of pixels in the input image. The approach uses an implicit quadtree to identify clusters of approximately coplanar points in the 2.5-D projection of the scene. PCA and the desired minimum number of samples (valid depth pixels) in some tree node guide the subdivision criteria of the quadtree. Thus, the maximum height of the tree is known *a priori*, making the computational cost of the detection procedure predictable. For each cluster, our approach casts votes for a reduced set of planes in the sparse spherical accumulator representing the parameter space of possible planes. Casted votes are weighted by trivariate-Gaussian distributions that models the uncertainty on the best-fitting plane of the node samples in each cluster. Our approach is deterministic, fast, and robust to the detection of spurious planes, even in presence of outliers and discontinuities.

We are currently investigating how to reduce the memory footprint of our technique by considering only the subset of planes that cross the camera's frustum. We are also exploring the proposed algorithm as part of a real-time solution for geometric reconstruction of indoor environments by stitching planar regions detected at each frame. In this application, the input depth images are provided on-the-fly by off-the-shelf depth sensors, like Structure Sensor [1] and Project Tango [2].

Acknowledgments

This work was sponsored by CNPq-Brazil grants 456.016/2014-7, 308.316/2014-2, and FAPERJ grants E-26/110.092/2014, E-26/202.832/2015.

References

[1] Occipital, Structure Sensor (2015). URL `http://structure.io`

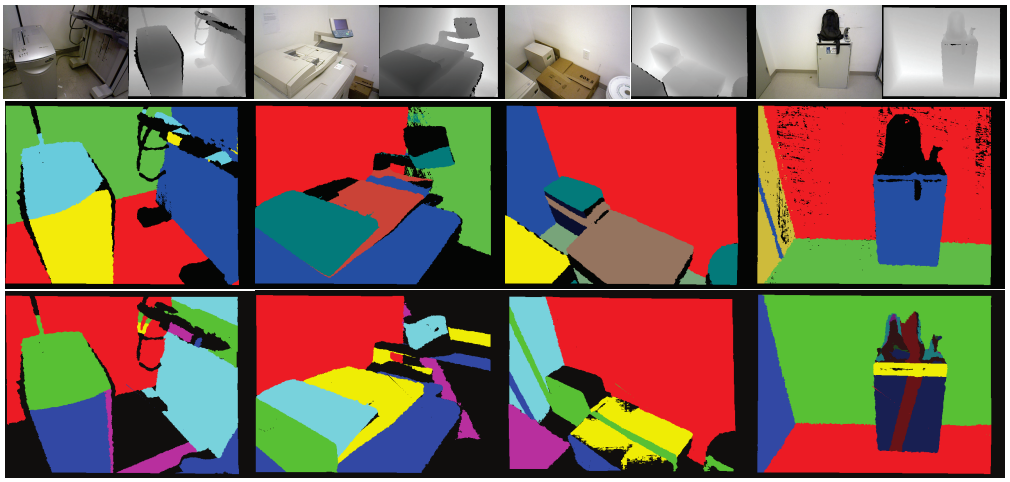


Figure 5: Comparison of plane-detection results on real scenes (best viewed in color). From left to right, frames 1063, 1791, 2297, and 4161 of the *Copy Room* dataset. The first row shows the color and the input depth images. The second and third rows present the planes (identified by colors) detected by the D-KHT and the 3-D KHT, respectively. In these experiments,

Table 2: Comparison of the performance of the proposed approach (D-KHT) and the state-of-the-art (3-D KHT) regarding frame 2453 from the *Office* dataset at different levels of impulsive noise. Time in columns *Clustering*, *Voting*, *Peaks*, and Total is expressed in milliseconds.

Input		3-D KHT						D-KHT					
Noise	Samples	Clustering	Voting	Peaks	Total	Clusters	Planes	Clustering	Voting	Peaks	Total	Clusters	Planes
0%	307,200	124.989	187.514	62.507	375.01	358	6	1.87	249.015	77.199	328.084	522	7
10%	276,728	109.382	187.528	62.523	359.433	343	7	1.706	248.016	78.325	328.047	524	7
20%	245,696	93.744	203.139	62.503	359.386	315	6	1.834	256.659	79.066	337.559	520	7
30%	215,055	93.754	187.513	46.877	328.144	264	7	1.648	251.982	78.774	332.404	523	7
40%	184,182	78.129	171.896	62.502	312.527	224	5	1.619	255.974	78.892	336.485	528	7
50%	153,229	62.493	187.522	46.877	296.892	182	5	1.631	250.447	79.867	331.945	501	7
60%	122,552	62.494	187.512	46.875	296.881	167	5	1.741	249.85	75.779	327.37	360	7
70%	92,134	62.493	203.14	46.876	312.509	149	6	1.268	242.902	72.303	316,473	192	7
80%	61,334	46.877	203.131	62.501	312.509	137	5	0.672	249.809	70.649	321.13	171	7
90%	30,826	189.442	156.264	62.506	408.212	73	5	0.454	244.444	72.029	316.927	120	5

Table 3: Comparison of the performance of the proposed approach (D-KHT) and the state-of-the-art (3-D KHT) regarding frame 2297 from the *Copy Room* dataset at different levels of impulsive noise. Time in columns *Clustering*, *Voting*, *Peaks*, and Total is expressed in milliseconds.

Input		3-D KHT						D-KHT					
Noise	Samples	Clustering	Voting	Peaks	Total	Clusters	Planes	Clustering	Voting	Peaks	Total	Clusters	Planes
0%	266,917	31.234	16.225	31.249	78.708	59	9	0.743	26.893	21.036	48.672	356	10
10%	239,998	31.236	15.635	15.625	62.496	57	8	0.497	12.399	13.937	26.833	196	11
20%	213,492	31.242	16.824	31.25	79.316	60	9	0.535	10.654	15.078	26.267	196	11
30%	187,034	15.618	16.623	15.626	47.867	53	8	0.568	12.906	15.059	28.533	195	11
40%	160,332	15.617	16.309	31.25	63.176	52	11	0.495	9.585	13.996	24.076	197	10
50%	133,163	15.61	15.625	31.25	62.485	56	11	0.489	10.004	13.993	24.486	197	10
60%	106,362	15.626	15.908	31.25	62.784	49	9	0.501	9.745	13.743	23.989	194	10
70%	80,149	46.87	15.321	31.249	93.44	44	9	0.49	3.838	7.854	12.182	186	11
80%	53,327	19.377	15.625	14.531	49.533	43	11	0.238	2.319	4.428	6.985	61	11
90%	26,746	43.75	17.251	19.365	80.366	31	9	0.228	2.453	4.158	6.839	60	11

[2] Google, Project Tango (2015).
URL https://www.google.com/atap/project-tango/

[3] R. I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, 2004.

[4] R. Kaucic, R. Hartley, N. Dano, Plane-based projective reconstruction, in: Proc. of ICCV, Vol. 1, 2001, pp. 420–427. doi:10.1109/ICCV.2001.937548.

[5] B. Triggs, Autocalibration from planar scenes, in: Proc. of ECCV, Vol. 1, 1998, pp. 89–105. doi:10.1007/BFb0055661.

[6] D. Holz, S. Holzer, R. B. Rusu, S. Behnke, Real-time plane segmentation using RGB-D cameras, Lecture Notes in Computer Science 7416 (D) (2012) 306–317. doi:10.1007/978-3-642-32060-6_26.

[7] G. Simon, A. W. Fitzgibbon, A. Zisserman, Markerless tracking using planar structures in the scene, in: Proc. of ISAR, 2000, pp. 120–128. doi:10.1109/ISAR.2000.880935.

[8] S. Choi, Q.-Y. Zhou, V. Koltun, Robust reconstruction of indoor scenes, in: Proc. of CVPR, 2015, pp. 5556–5565. doi:10.1109/CVPR.2015.7299195.

[9] L. A. F. Fernandes, M. M. Oliveira, Real-time line detection through an improved Hough transform voting scheme, Pattern Recognit. 41 (1) (2008) 299–314. doi:10.1016/j.patcog.2007.04.003.

[10] F. A. Limberger, M. M. Oliveira, Real-time detection of planar regions in unorganized point clouds, Pattern Recognit. 48 (6) (2015) 2043–2053. doi:10.1016/j.patcog.2014.12.020.

[11] P. V. C. Hough, Machine analysis of bubble chamber pictures, in: Proc. of Int. Conf. on High Energy Accelerators and Instrum., 1959, pp. 554–558.

[12] R. O. Duda, P. E. Hart, Use of the Hough transformation to detect lines and curves in pictures., Commun. ACM 15 (1) (1972) 11–15. doi:10.1145/361237.361242.

[13] G. Vosselman, B. G. H. Gorte, G. Sithole, T. Rabbani, Recognizing structure in laser scanner point clouds, in: Proc. of Conference on Laser scanners for Forest and Landscape assessment and instruments, Vol. XXXVI, 8/W, 2004, pp. 33–38.

[14] H. H. Nguyen, J. Kim, Y. Lee, N. Ahmed, S. Lee, Accurate and fast extraction of planar surface patches from 3D point cloud, in: Proc. of Int. Conf. on Ubiquitous Information Management and Commun., 2013, p. 84. doi:10.1145/2448556,2448640.

[15] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395. doi:10.1145/358669.358692.

[16] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, Comput. Graph. Forum 26 (2) (2007) 214–226. doi:10.1111/j.1467-8659.2007.01016.x.

[17] L. A. F. Fernandes, M. M. Oliveira, A general framework for subspace detection in unordered multidimensional data, Pattern Recognit. 45 (9) (2012) 3566–3579. doi:10.1016/j.patcog.2012.02.033.

[18] L. A. F. Fernandes, M. M. Oliveira, Handling uncertain data in subspace detection, Pattern Recognit. 47 (10) (2014) 3225–3241. doi:10.1016/j.patcog.2014.04.013.

[19] M. A. Fischler, R. C. Bolles, Perceptual organization and curve partitioning, IEEE Trans. Pattern Anal. Mach. Intell. 8 (1) (1986) 100–105. doi:10.1109/TPAMI.1986.4767756.

[20] D. S. Chen, A data-driven intermediate level feature extraction algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 11 (7) (1989) 749–758. doi:10.1109/34.192470.

[21] P. J. Besl, R. C. Jain, Segmentation through variable-order surface fitting, IEEE Trans. Pattern Anal. Mach. Intell. 10 (2) (2002) 167–192. doi:10.1109/34.3881.

[22] J. Poppinga, N. Vaskevicius, A. Birk, K. Pathak, Fast plane detection and polygonalization in noisy 3D range images, in: Proc. of IROS, 2008, pp. 3378–3383. doi:10.1109/IROS.2008.4650729.

[23] J. Xiao, J. Zhang, J. Zhang, H. Zhang, H. P. Hildre, Fast plane detection for SLAM from noisy range images in both structured and unstructured environments, in: Proc. of ICMA, 2011, pp. 1768–1773. doi:10.1109/ICMA.2011.5986247.

[24] R. Adams, L. Bischof, Seeded region growing, IEEE Trans. Pattern Anal. Mach. Intell. 16 (6) (1994) 641–647. doi:10.1109/34.295913.

[25] J. Zhu, L. Wang, R. Yang, J. Davis, Fusion of time-of-flight depth and stereo for high accuracy depth maps, in: Proc. of CVPR, 2008, pp. 1–8. doi:10.1109/CVPR.2008.4587761.

[26] F. C. Crow, Summed-area tables for texture mapping, SIGGRAPH Comput. Graph. 18 (3) (1984) 207–212. doi:10.1145/964965.808600.

[27] H. Samet, The quadtree and related hierarchical data structures, ACM Computing Surveys 16 (2) (1984) 187–260. doi:10.1145/356924.356930.

[28] L. Parratt, Probability and experimental errors in science, J. Frankl. Inst.-Eng. Appl. Math. 272 (6) (1961) 527–528. doi:10.1016/0016-0032(61)90902-4.

[29] I. Jolliffe, Principal Component Analysis, John Wiley & Sons, Ltd, 2005, Ch. 18, pp. 1094–1096. doi:10.1002/0470013192.bsa501.

[30] 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 2 (2) (2011) 1–13. doi:10.1007/3DRes.02(2011)3.

[31] D. P. Bertsekas, J. N. Tsitsiklis, Introduction to Probability, Athena Scientific, 2002.

[32] dlib, dlib c++ library.
URL http://dlib.net

[33] Q.-Y. Zhou, V. Koltun, Dense scene reconstruction with points of interest, ACM Trans. Graph. 32 (4) (2013) 112:1–112:8. doi:10.1145/2461912.2461919.