Doctoral Dissertation (Shinshu University)

Robust and efficient geometric primitive detection in 3D point clouds

September 2020

Sandoval Gálvez Jaime Alberto

I dedicate this thesis to my family for all their unconditional love and support

ABSTRACT

Recently, with the advent of low-cost 3D sensing technology, the need for processing, analyzing, and detecting patterns in 3D point clouds has been getting the focus of industry and the academy.

Assistant robots and the introduction of intelligent robotics in the industry will play a major role in how well we address population aging and even disaster prevention and management. In reverse engineering, we are able to create new pieces without blueprints. The acquisition of 3D point clouds and the detection of its geometry are crucial to numerous applications.

One main characteristic of urban environments and man-made objects is their geometric nature. They are comprised of several geometric shapes such as planes, spheres, and cylinders. Due to their broad applications, detecting them is an important task in 3D point cloud processing. However, their detection in point clouds is not trivial.

The first part of this thesis focuses on describing different 3D sensing technologies and the characteristics of their resulting point clouds as well as the discussion of geometric primitive detection. Conventional geometric primitive detection algorithms rely on the correct estimation of local features such as normal vectors. In most cases, their efficient and correct computation is not yet possible. RANSAC based methods and the Randomized Hough Transform (RHT), use random sampling to find hypothetical models faster. In RANSAC based methods, deficiencies arise with a poor inliers/outliers ratio, which is a common scenario in high-range sensors or even low-cost sensors due to noise. Sometimes, the iterations are not enough to detect shapes diminishing their efficiency and accuracy.

Hough Transform methods suffer from the quantization of their accumulator, they need to rely on further refitting the detected shapes during the voting process. Moreover, for 3D shapes, it is only defined for planes since an accumulator of more than 3 dimensions would need an unfeasible amount of memory.

In the first part of this thesis, I explain the basic concepts behind 3D sensors. Therefore, in this thesis, I focus on solving the problem of geometric primitive detection in 3D point clouds.

As planes are the most common geometric primitive, the second part of this thesis focus on a more efficient approach for plane detection: the Fast and Deterministic Hough Transform based plane detector (FDHT). The key for its efficiency is that we opted for analyzing fewer voxelized regions as opposed to pixel-wise neighborhoods to detect planes. Points are filtered using a Scaled Difference of Normals (SDoN) over these voxels. Then, instead of random sampling, the proposed plane detector analyzes deterministically the whole point cloud with higher efficiency. Because the detection mechanism is based on Hough voting, I proposed a novel memory model for the accumulator based on

nested trees instead of contiguous arrays to avoid memory saturation in Hough space. Nevertheless, it still requires improvement in robustness and efficiency since it needs a pre-filtering mechanism (SDoN) which adds computational complexity.

Therefore, based on existing knowledge about sliding windows in object detection, in the third part of this thesis, I propose a novel plane detector approach based on sliding voxels.

Instead of analyzing independent voxels of the point clouds, overlapping voxelized regions of the point cloud. A 3x3x3 sliding voxel is defined for each voxelized region in the point cloud. An efficient non-planar filtering method can be performed simultaneously while the sliding voxels are sorted by their planarity. These planar regions define hypothetical planes that are sorted by their number of inliers. Finally, by removing planes from bigger to smaller we can detect planes with superior robustness and efficiency. Experiments from realistic point cloud simulations show that the proposed sliding voxel plane detector is drastically more efficient and robust than the state-of-the-art planes detectors.

Since the main key of the sliding voxel algorithm is its efficiency generating hypothetical models from point clouds. We explored this paradigm to sphere detection.

As opposed to planes, which are simple and linear geometric models, spheres are closed surface quadrics. The sliding voxel for sphere detection provides an efficient local sphere fitting algorithm that gets the most probable sphere at each sliding voxel using robust statistics. This allows us to robustly generate hypothetical spheres with high efficiency. For each hypothetical sphere, Hough voting is performed on a 4D sparse accumulator to detect the most prominent spheres. After extracting spheres from the accumulator, the algorithm refits them and filter by a novel measure of completeness.

The sliding voxel sphere detector has drastically higher efficiency while being more precise than the conventional method in realistic point cloud simulations. It demonstrated superior robustness by keeping its high accuracy while increasing Gaussian noise. Furthermore, it shines when processing LIDAR point clouds. They tend to be massive and with high range; hence, their number of model outliers are overwhelming for methods based on random sampling. There, the proposed sphere detector had not only superior efficiency but its accuracy was exceptionally high.

The sliding voxel paradigm showed that we can design highly efficient and robust shape detectors. Further works in the detection of arbitrary shapes is a promising line of research that opens from this thesis.

PUBLICATIONS

The contributions of this thesis have appeared previously in my publications[68–70].

ACKNOWLEDGMENTS

I am eternally grateful to my family, my wife, my daughter, and my mother who gave me the courage to keep on track during difficult times.

I wish to express my deepest gratitude to my supervisor, Professor Tanaka Kiyoshi, for all his valuable kind support and encouragement through the master and doctoral courses. Every monthly seminar we have been holding in Tokyo and Nagano, my supervisor, and Professor Iwakiri Munetoshi always gave me excellent feedback. I am indebted to them for all the advice they kindly brought me.

I also want to thank Professor Aguirre Hernan for his valuable academic and personal support through these 5 years. My senpai Mr. Uenishi Kazuma, who guided me in putting into practice point cloud processing in the PCL library.

I want thanks to faculty members, Ms. Watanabe Masako, Ms. Takayanagi Azusa, and Ms. Nishizawa Keiko for their support in administrative work and my adaptation to life in Japan.

Last but not least, I want to thank the Japanese taxpayers and the Japanese Government because I was able to pursue a postgraduate diploma thanks to the scholarship for foreigners of the Ministry of Education, Culture, Sports, Science, and Technology (MEXT).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

RECENT DIRECTIONS AND TRENDS IN INFORMATION AND COMMUNICATION TECHNOLOGIES

1.1 COMPUTER PERCEPTION AND 3D INFORMATION

Recently, Information and Communication Technologies (ICT) have been playing a central role in improving the way we live, socialize, and work. Since the widespread availability of digital cameras and broadband connections, we can communicate instantly with others with high-reliability thanks to advances in communications and image and audio compression. Also, image sensing techniques allowed physicians to diagnose patients without invading the body of their patients. These advances are possible because we can acquire, process, transmit, and visualize data from sensors to something understandable by humans. However, due to technological limitations, those sensing techniques were constrained to 2D images.

One of the goals of computer vision is to percept and analyze the environment automatically such that computers or automata can perform actions we need such as navigate, manipulate, categorize, and so on. This has played a main role in reducing operating costs, also reduces the risks of operation when we need to work in hazardous environments for humans [90].

Many European countries and particularly Japan are experiencing acute aging of their population. Population aging is known to impact seriously in the economy, productivity, and the sustainability of pensions among others. Automatization is one of the approaches to alleviate this problem but some tasks require precise information about the 3D world.

Images are just projections on the camera sensor; hence, we lose depth information when taking photographs. This limits their applications in many important areas such as reverse engineering [64], user-centered product development [54], patient-specific 3D models in medicine [66] just to mention a few of them. With 3D spatial information we can give precise information to devices that operate in the real world, right now, we cannot devise fully autonomous navigation, disaster management, topography, augmented reality, among others without 3D information. Their applications will increase exponentially as we figure out how to efficiently and accurately obtain, process, and analyze the structure of surfaces of the real world in 3D.

Therefore, because of its future potential, methods for obtaining 3D information have been in very active research in the past few years. Some of them are already implemented in small devices such as smartphones or tablets.

(a) Color image (b) Depth image

Figure 1.1: Color and depth image taken from a Kinect™ sensor [28]

1.1.1 *Types of 3D information*

Several methods to capture spatial or 3D information exists as well as the formats they output. 3D sensing methods can be categorized as passive or active.

Their output formats are also categorized in mainly 3 types of formats: depth images, point clouds, and 3D Models. Depth images are images in which their intensity channel is replaced by a measure of depth (usually quantized for better performance). Spatial information from this type of image is called 2.5D because they are not completely 3D, i. e., we can not get depth information that was not exposed to the sensor field of view. Low-cost and entertainment-focused sensors produce this type of 3D information.

Figure 1.1 shows both a color and depth image of a PrimeSense sensor from left to right. Depth is encoded as a grayscale image in Figure 1.1(b). Comparing depth with Figure 1.1(a), we can notice a lack of spatial information in some regions due to the physical limitations of the sensor.

3D Models are set of polygons (usually triangles) that as a whole represent the shape and texture of objects. Usually, 3D models are the cleanest and most complete type of 3D information we can get Nevertheless, they are obtained later in a computationally expensive 3D reconstruction pipeline. Therefore, it is avoidable unless is necessary for the application.

Point clouds are just sets of points in a 3D space that represent the surface of objects. They might include other types of information like color or intensity and are considered a core type in 3D spatial information. Depth images and 3D models can be transformed into their point clouds form. In the case of 3D models, we can preserve some geometric information (normal vectors) but we lose the relationship between points in the form of polygons. Therefore, in this thesis, we focus on point cloud processing.

1.1.2 *A formal definition of point clouds*

Let $\mathcal{P}$ be a set of points p_i of at least 3 dimensions. The set of minimum dimensions represent only the $[x, y, z]^T$ coordinates of each point. As each point can be integrated with different kinds of information the number of dimensions can grow.

Depending on the *acquisition method* a point cloud can include other features that come from sensors such as color or intensity. Moreover, we can estimate point-wise unit normal vectors by adding another 3 dimensions, or curvatures by adding 1 dimension. However, these estimations come with a degree of uncertainty and their correctness or availability should not be taken for granted. Therefore, Point Clouds processing algorithms can utilize this information but should not be limited or negatively affected by this extra information.

Figure 1.2 shows a photo, the 3D model, and a point cloud of the reconstruction process of the Stanford bunny [43]. The bunny figure from Figure 1.2(a) was scanned using a rotating table laser scanner. The point cloud in Figure 1.2(c) are just the vertices of its 3D model Figure 1.2(b). As the point cloud is aligned wit the XY plane, the z coordinates of the plane were encoded with a heatmap to understand the point's depth in the picture.

1.2 ADVANCED SENSING TECHNOLOGIES IN POINT CLOUD ACQUISITION

The upper part of Figure 1.3 illustrates the different acquisition methods, and the lower part show their applications. Each application drives the acquisition method. For instance, low-range scanners are suitable for indoor scenes or small objects reconstruction since they offer a short minimum scan distance. Large range scanners are suitable for Simultaneous Localization and Mapping since they can obtain a better picture of objects far away.

There are several acquisition methods to obtain spatial information or 3D point clouds. We can put them into 2 categories: *Passive 3D sensing* and *Active 3D sensing*.

(a) Photo (b) 3D Model (c) Point cloud

Figure 1.2: Stanford bunny reconstruction [43]

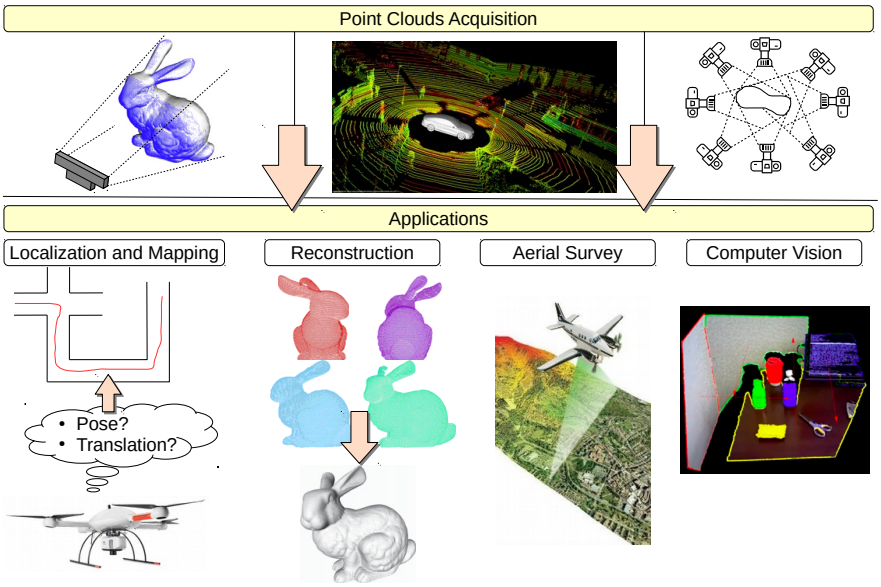


Figure 1.3: 3D point cloud acquisition and applications

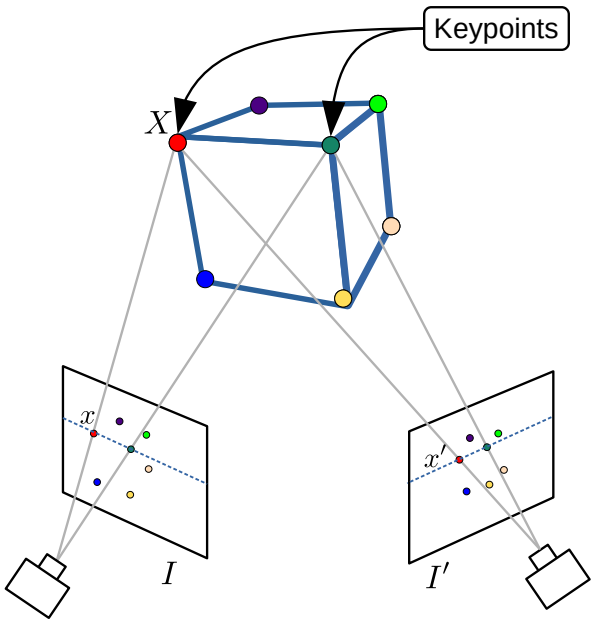


Figure 1.4: Stereovision

1.2.1 *Passive 3D sensing*

Passive 3D sensing refers to a 3D scanning technique that utilizes natural light and conventional cameras. Stereovision and structure-from-motion (SfM) are two popular passive 3D sensing techniques. Both require a well-illuminated scene with textured objects and two images. The main difference between stereovision and SfM is that stereovision

expects both images to be taken at the same time and SfM expects both images were taken at different times.

Figure 1.4 shows a simplified example of stereovision. It assumes a pair of images are taken from two cameras at the same time. Let x and x' be corresponding point pairs of X from I and I' of respectively. These correspondences are estimated by keypoints detection and feature extraction.

Keypoints are characteristic pixels of each image that have a high probability of being found if we capture the same object with similar visibility settings. Once matches are found, the algorithm needs to find the epipolar lines. To calculate them one must compute the *Fundamental Matrix* F in which is the algebraic representation of epipolar geometry. Epipolar lines and translation between correspondences can be performed with F. To simplify the processing, both images are rectified by applying a transformation such that both rely on the same plane [58] and the epipolar lines become parallel to the horizontal axis of the rectified image.

As seen in Figure 1.5, SfM works similarly but by moving a single camera over the scene. This makes matching more complex since objects can be moved, soft-objects can bend, and the cameras might not be the same thus requiring a more robust matching of the keypoints inside the scene.

One strong point of this type of sensing is that we can generate colored point clouds when we have a minimum of 8 matches in the image [46]. However, these techniques are not suitable for low-light and texture-less environments with homogeneous geometry [41]. The lack of a *reference frame* forces every passive stereovision-based method to use the first image as the reference frame for rotation and a translation. Therefore, 3D information generated by these algorithms in 2 or more separated batches will have arbitrary scales.

1.2.2 *Active 3D sensing*

On the other hand, in *Active 3D sensing*, a device emits laser beams to illuminate a scene and detect 3D shapes. There are mainly 3 different types of Active 3D sensing techniques, Structured Light, Light Detection and Ranging (LIDAR) and Time-of-Flight (ToF).

Structured Light is a type of active stereovision. Figure 1.6 exemplify the basic operation of a structured light sensor. It consists on a projector that emits a known pattern of infrared laser beams, and a camera that captures the distortion of those patterns to detect the shape of objects. To calculate depth, the camera needs to be calibrated and the baseline needs to be known with precision. Structured light sensors such as the Microsoft Kinect™and Asus Xtion™ PRO use a set of IR and RGB cameras to get both color and depth.

LIDAR and ToF utilize laser pulses to detect the relative position of the objects. Both measure the time the emitted pulses takes to be captured into the sensor to measure depth.

LIDAR was once an acronym for Laser RADAR because it works similarly. Since it was the first ToF-based technology, it has been widely used in Terrestrial and Aerial Surveying. It is comprised of a rotating laser beam that emits pulses that are captured by an infrared sensor. As the pulse is captured, it measures the distance on the laser beam direction by using the time that took to reach the sensor.

When a LIDAR has only 1 laser beam, we can say it is a 2D LIDAR because it detects depth over the plane the laser beam is rotating. Figure 1.7 shows the basic mechanism of a 2D LIDAR, a set of fixed and rotated mirrors is used to reflect and capture the laser pulse.

If multiple laser beams are emitted, the sensor is a 3D LIDAR. Figure 1.8 illustrates a Velodyne 3D LIDAR sensor [25]. It emits several laser pulses from LEDs fixed vertically in the sensor. In this type of sensors, its light-emitting head rotates and produces a 360-degree field-of-view point cloud. Aside from measuring distance, the reflectivity of objects as the intensity of the laser pulses can be captured with this technique as an additional sensor-dependent feature of point clouds.

Figure 1.9 shows two images of a LIDAR point cloud taken from a Velodyne HDL-64E in movement. The intensity of the detected pulses is color-encoded in a heatmap for better visualization. In Figure 1.9(a) we can observe a near-ground level projection of the point cloud. The rotating patterns are visible even in near objects, and the empty space in the middle indicates the sensor's location, which defines the minimum distance required for sensing.

Figure 1.9(b) is an aerial view of the same point cloud. We can observe that as we get far from the sensor, the distance between scanlines is increased, the distance between points is also getting bigger. Recently,

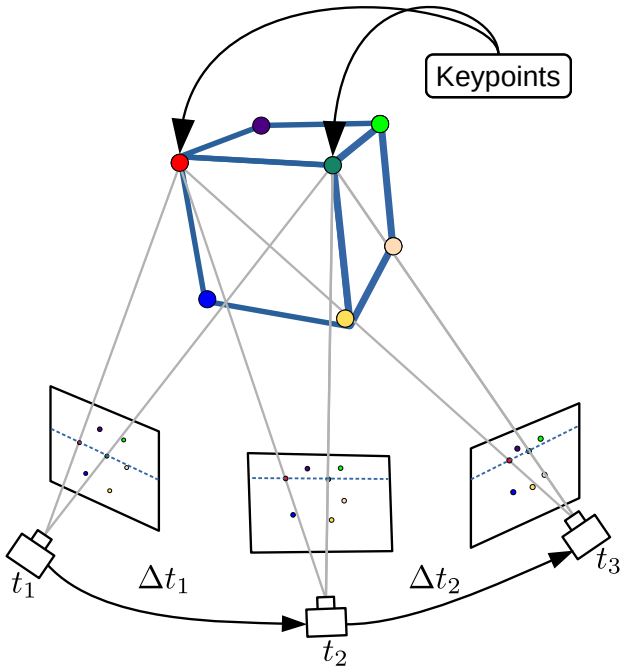


Figure 1.5: Structure-from-motion (SfM)

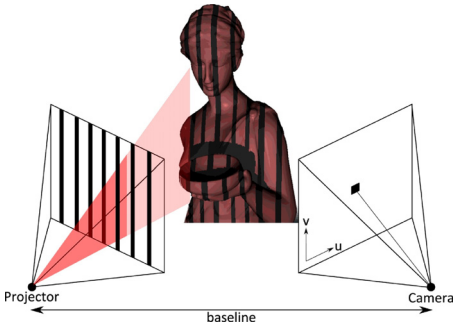


Figure 1.6: Structured light sensing [71]

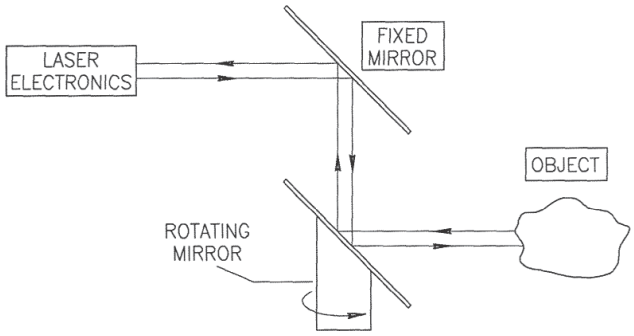


Figure 1.7: 2D LIDAR sensor mechanism [24]

in aims to introduce more features to these 3D LIDAR point clouds, advanced techniques to add color information have been developed [25, 65, 84].

A ToF camera emits modulated wave signals to objects and an IR camera captures all the signals at once [41]. When 3D LIDAR sensors posses rotating elements, this type of camera is assisted When the camera detects a different signal it measures the time it has taken and by using a simple formula one can detect per-pixel depth. Similarly to Structured Light sensors, additional RGB cameras paired with IR cameras can be used to obtain color.

Another type of point cloud acquisition method are model sampling and sensor simulation. While model vertices can be utilized as point clouds, it is inconvenient when the model has been built using CAD software or has been simplified using geometric primitives. The Point Cloud Library [67] offers two tools to accomplish these tasks mesh_sampling

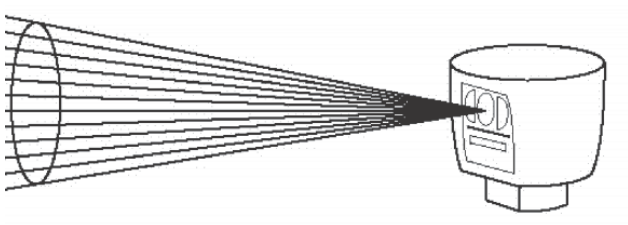
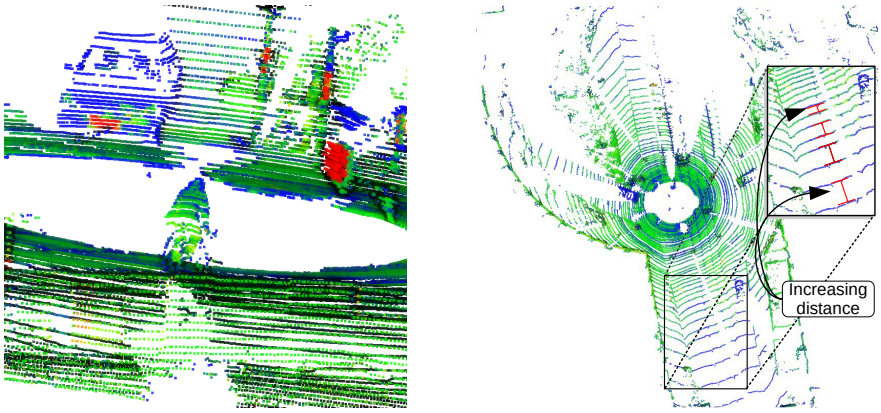


Figure 1.8: 3D LIDAR sensor mechanism [25]



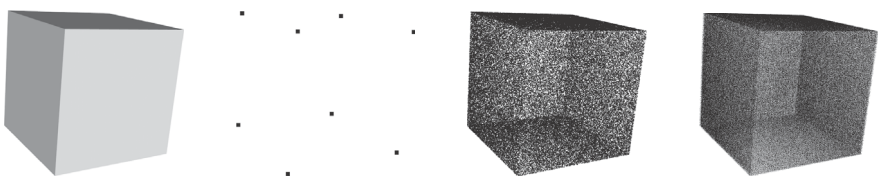
(a) Ground view (b) Aerial view

Figure 1.9: 3D LIDAR point cloud example from the KITTI dataset [18]

and mesh2pcd. mesh_sampling uses raytracing to sample 3D points from the models and mesh2pcd performs uniform random sampling over the model faces. In Figure 1.10 we can visualize the different approaches of sampling point clouds from models. Figure 1.10(a) shows a cube model rendering. We can visually understand that the vertices of Figure 1.10(b) are descriptive enough to analyze its shape because of the lack of points describing the surface of the cube. However, Figure 1.10(c) shows the random patterns of the mesh_sampling tool. In Figure 1.10(d) we can observe the model has been sampled much more uniformly. However, it has an induced thickness due to the quantization of the 3D space while performing raytracing over a grid.

3D sensing techniques can be simulated in software. Blensor [23] is a version of Blender, a 3D computer graphics software, that integrates a sensor simulation plugin. This sensor simulation is highly realistic and is capable of simulating ToF, LIDAR, and Structured Light (Kinect™V1).

3D point cloud sensing is becoming ubiquitous and its applications are in increasing demand. From terrestrial and aerial surveys a few decades ago, we were witnesses of their introduction to consumer electronics with the Kinect™sensor, and the recent introduction of massive consumer electronics such as smartphones and tablets. Therefore, their applications are expected to be extended with high speed in the years to come.



(a) Cube model (b) Cube vertices (c) mesh_sampling point cloud (d) mesh2pcd point cloud

Figure 1.10: Cube model and point clouds sampled from different methods

1.3 BRIEF OVERVIEW OF 3D POINT CLOUDS APPLICATIONS

The applications of 3D point clouds have been increasing in recent years due to increasing computing power and the development or advanced algorithms. Robotics is one of the most direct applications of this technology but not limited to it. 3D-guided navigation has been used for numerous situations, even in missions to Mars [47].

Reconstruction of objects and scenes [28] is increasingly in demand by industry and entertainment. Kinect fusion is a technique that fuses point clouds in a grid called the Truncated Signed Distance Function (TSDF), and reconstructs a model from the integration of several registered depth images in a volume.

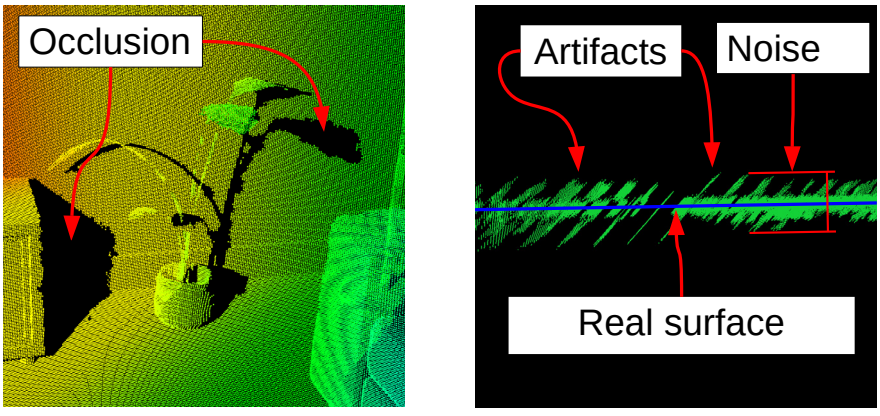
1.4 CHALLENGES IN POINT CLOUDS PROCESSING

1.4.1 *Noise and sensor artifacts*

Different 3D sensing techniques share some defects such as variable sparsity, occlusion and the presence of artifacts. Noise is present in every type of sensing method as fluctuations (wiggling) in the position of points from the sampled surfaces. Deformations in objects from poor calibration in Structured Light sensors can diminish the accuracy of algorithms. Also, empty holes in point clouds derived from occlusion is a constant in almost every type of sensing as shown in Figure 1.11(a).

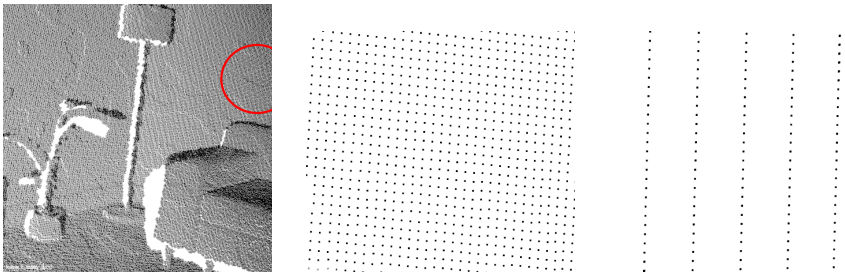
Aside from regular gaussian-like noise, every sensing technique have their type of artifacts and variations in points density. For instance, Structured Light Sensors suffer from adverse effects of depth quantization [10, 11] as shown in Figure 1.11(b).

Figure 1.12 shows a 3D point cloud of a wall sensed using a noiseless Kinect™V1 sensor simulation in Blensor. In Figure 1.12(a) we can observe a point cloud, where the circled region lies on a wall. This wall as seen from the sensor perspective is completely planar, we can



(a) Occlusion (b) Noise and sensor artifacts

Figure 1.11: Nuissances of point clouds



(a) Point cloud (b) Front (sensor) view (c) Side view

Figure 1.12: Structured light sensors depth quantization pattern

confirm this in Figure 1.12(b). Nonetheless, Figure 1.12(c) is the region but taken in which we can notice with high clarity the structured light sensor depth quantization.

This pattern, while combined with x-noise [10], and point fluctuations makes it extremely difficult to define thresholds for search radius, voxel size, or other algorithm parameters that are dependent on the point cloud's resolution.

It is well known that points are more distant from each other as they are farther from the sensors. LIDAR sensors, as high range devices, their precision is much worse near the limits of their range as seen in Figure 1.9(b). ToF sensors suffer from *flying pixels* [87] when under high solar radiation or when a cell in the sensor receives 2 pulses [6], i. e.when the pulse is just at the edge of an occlusion.

1.4.2 *Unorganized nature of point clouds*

Point clouds can be divided into organized or unorganized. Organized point clouds are originated from passive or active stereovision-based techniques because they can be organized as a collection of pixels and depth. This eases the search for neighboring points since the points are already organized in a grid. Organized point clouds are 2.5D since they normally include highly occluded scenes. Objects and scenes are not fully 3D since we are unable to sense beyond the sensor's field-of-view.

On the other hand, unorganized point clouds are sensed from image-less sensing techniques such as LIDAR or SfM. They are just sets of points without any inherent organization, not even point sorting gives a glimpse of their underlying geometry or structure. To get basic information about the extension, density, and organization of the point cloud one has to compute 3D bounding boxes, 3D grids, or even more complex structures such as kd-trees or octrees [36, 51].

While image compression is a well-established area of research, point cloud compression non-trivial many approaches are exploring geometric and neural-based techniques [8, 34, 52, 72, 75, 78].

For the sake of generalization, we treat all point clouds as unorganized unless it is necessary to exploit the organized nature of some point clouds to increase the efficiency of algorithms [29, 30, 82].

2 RESEARCH OBJECTIVES

2.1 GEOMETRIC PRIMITIVE DETECTION AND ITS APPLICATIONS

In urban environments, and when surrounded by man-made things, the most common types of structures are geometric because they have convenient properties: they are stable and easier to manufacture. For instance, buildings and streets are comprised of many planar surfaces while car wheels are similar to a torus. Complex objects with many geometric features such as statues and people can be approximated by several geometric primitives[73].

Geometric primitives are parameterized objects that can be expressed mathematically. The most common geometric objects are planes, spheres, and cylinders. We can reconstruct a scene full of geometric objects easily if we know their coefficients. However, their detection in point clouds is not a trivial task and has been addressed in several works in the literature[5, 15, 29, 57, 59, 73].

The detection of geometric primitives can also be useful for reverse engineering since the reconstructed 3D models tend to approximate a surface by small polygons and their geometry needs to be simplified in a post-processing stage. Simplifying the geometry of point clouds also opens a door for geometric compression[8, 34, 52, 78].

However, geometric primitive detection is not a trivial task. Since we are working with sparse and unorganized points, we have to infer the underlying geometric efficiently, and to do it robustly even in the presence of highly noisy environments with huge variations of noise patterns.

In general terms, a geometric primitives detector is a software that inputs a set of points and outputs a set of coefficients. Figure 2.1 illustrates a diagram of a generic primitive shapes detector. It inputs a point cloud $\mathcal{P}$ with at least 3 dimensions ($n_p \geq 3$) alongside with noise and artifacts. The detector can be defined as a map $f(\mathcal{P}, \epsilon) : \mathbb{R}^{n_s} \mapsto \mathbb{R}^{n_s}$, where n_p is the dimensionality of the point cloud and n_s is the dimensionality of the detected primitive shapes $\mathbf{S}^{det}$.

The dimensionality of $\mathbf{S}^{det}$ depends on the number of parameters required to describe each shape. It can be variable, for instance, a plane in space can be represented by its equation coefficients (4), a point and a normal vector (6), or by applying the Hough Transform (3).

Planes and spheres are the most basic geometric primitives. They possess uniform geometry in two different ways:

- A plane has zero Gaussian curvature, they expand linearly and indefinitely over any direction as long as it is orthogonal to its normal vector.

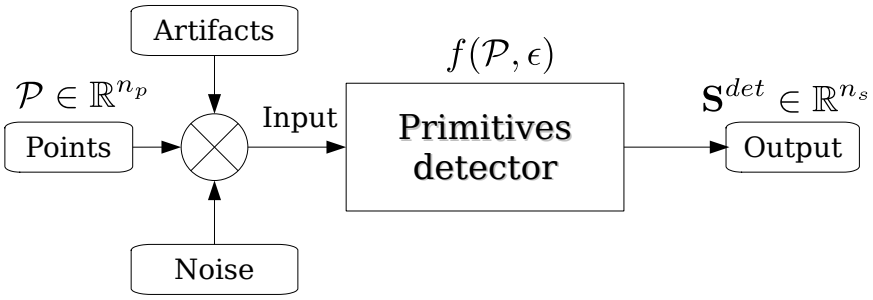


Figure 2.1: Primitives detector diagram

- A sphere has constant positive Gaussian curvature, i. e., it bends uniformly in every direction.

These characteristics endow them with interesting applications. Planes are also developable surfaces; therefore, they can be used as a representation of tangential portions of curved surfaces. Because of their constant curvature, spheres preserve geometric features independently of the point of view of a sensor making them excellent targets for 3D scanning.

Because of their wide area of applications, in this thesis, **we focused on solving problems of plane and sphere detection in unorganized point clouds**.

2.2 PLANE DETECTION AND REPRESENTATION

Planes are the most common primitives. In intelligent robotics, a core task is to recognize environment patterns, especially those from human crafted objects and scenes, which are describable by a set of planar structures.

Therefore, the detection of planar surfaces such as obstacles, floor, stairs, and walls is crucial to several applications; i.e., virtual keypoint detection [83], object detection [45], reconstruction [93], localization and mapping [42, 61, 62, 88], roof detection [79], forest mapping[63], and augmented reality [22]. Besides, plane detection is involved in essential robotics tasks such as placing objects onto planar surfaces [57] or climbing stairs [59].

Planes can be expressed in equations by their their Hessian normal form, where the normal vector and a point on the plane can be used

$$\hat{n} \cdot p_n + \rho = 0. \tag{2.1}$$

A point p_n over a plane is represented by its coordinates $(x, y, z)^T$. Then, the plane equation is defined by the distance from the plane point p_n to the origin in its normal vector $\hat{n}$ direction. Figure 2.2 illustrates the role of the main components of the Hessian normal form of a plane. The dot product of $p_n \cdot \hat{n}$ results in:

$$ax + by + cz + d = 0, \tag{2.2}$$

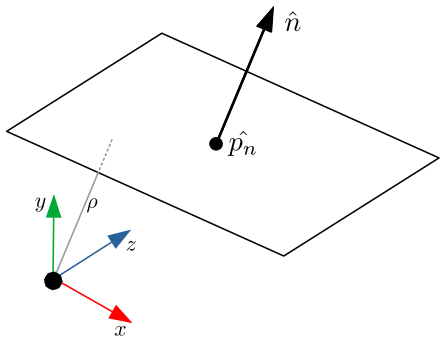


Figure 2.2: Plane diagram and parameters

where $\rho = -d$ and $\hat{n} = (a, b, c)^T$ in Equation (2.1). This is the general and canonical plane equation. Its coefficients are (a, b, c, d), and a set of them is expected in the output of a plane detector.

Planar surfaces extend indefinitely over the space. Therefore, some algorithms may opt to offer a set of inlier points that belong to $\mathcal{P}$ or to provide a transformation of $\mathcal{P}$.

2.3 SPHERE DETECTION AND REPRESENTATION

Sphere detection is an important technique in 3D computer vision with applications in broad areas such as materials engineering[37], measuring [53], medicine[19] among others. They have promising applications in point cloud registration [17, 86, 92].

A sphere is defined by its center coordinates $C = (C_x, C_y, C_z)^T$ and its radius r

$$(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2 = r^2. \tag{2.3}$$

Figure 2.3 shows the graphical explanation of the sphere parameters.

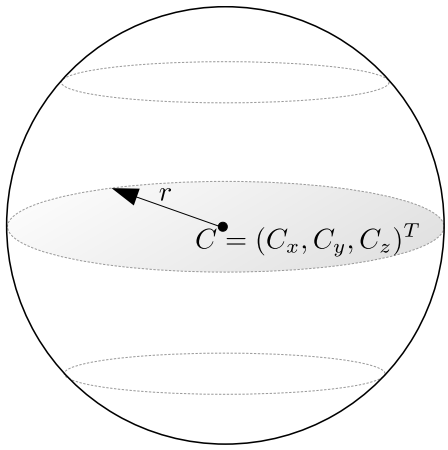


Figure 2.3: Sphere diagram and parameters

Both its center and radius are expressed in the same units. Therefore, in point clouds obtained from active sensing, their units are usually meters.

2.4 PROBLEMS WHEN DETECTING GEOMETRIC PRIMITIVES

Aside from the challenges of processing point clouds metioned in Section 1.4, there are still many problems to solve in the detection of parametric shapes in point clouds. A shape detector can be seen as a search problem, in which, the strategy of the search defines the performance in terms of efficiency and accuracy of the algorithms. Although conventional methods will be described later in detail, in this section, I will briefly explain their schemes and direct implications in efficiency and accuracy.

GEOMETRIC FEATURES OF POINTS NEIGHBORHOODS A single point is not descriptive enough to define any shape; hence, every method use each point as a reference, and search for its nearest neighbors. Since point clouds have no definite order, nearest neighbor search is performed by employing 3D space subdivision such as kd-trees or octrees. Once we can estimate the nearest neighbor set for each point, features can be extracted from their points' distribution. Therefore, parametric shape detection algorithms are expecting or computing features for each point and their neighbors. However, efficiently finding their neighbors and defining an appropriate search extent are processes that depend on the local distribution of points. Moreover, the processing time increases exponentially as points and their neighbor's density increase because these operations are performed point-wise.

2.4.1 *Inefficiency*

NORMALS VECTORS ESTIMATION The most basic and commonly used feature of point clouds is a *normal vector*. Normal vectors are unit vectors that define the estimated orientation of a surface at each point using their nearest neighbors. They are calculated point-wise using PCA; hence, its processing time increases exponentially with respect to an increment in the number of points. Moreover, nearest points share geometric properties and neighborhoods point sets with high overlap will output similar normal vectors. Therefore, their point-wise estimation is causing a severe inefficiency issue in conventional methods that use this feature [73, 83].

Additionally, conventional methods are using random sampling by assuming the number of inliers of the current detection target is relatively high. Although this is true for planes in urban environments, the search becomes more difficult, like finding a needle in a haystack, hence, time consuming as other less common primitives such as spheres or cylinders are meant to be detected.

POINT-WISE DETECTION A greedy approach to object detection would be to fix a minimum number of samples and generate hypothetical shapes by iterating over all possible combinations. Therefore, instead of searching for all combinations, algorithms usually rely on Random Sample Consensus or Random Subsampling which drastically reduces processing times and the feasibility of the shape search at the cost of falling in the aforementioned problems of *finding needle in a haystack* or losing geometric detail by an aggresive subsampling. In this way, their will poorly scale to point clouds with numerous points like those from dense reconstructions or long-range 3D LIDAR scans.

Another greedy approach is to use the Standard Hough Transform (SHT) to *vote* for all the possible shapes that can be defined for each point. That is, for each point we have to iterate over all the parameter space, which can be of higher dimensions that the points. This is a notorious costly process which can be unfeasible for high dimensions (> 2) parametric shapes. Therefore, instead of iterating over the whole parameter space, a random sampling strategy is selected to cast a vote for each random sample. Other non-deterministic approach consists on reducing the amount of points using random subsampling. Nonetheless, we end falling with the aforementioned problems when the search space becomes too narrow.

2.4.2 *Inaccuracy*

Conventional methods are expected to work with a relatively high number of inliers. According to [73], their shape detector start to detect false positives when the outliers are 20% of the points or above. These outliers can be present in any kind of point cloud but more often in noisy low-cost sensors or Terrestrial Laser Scanning (TLS) point clouds.

Their accuracy highly depends on normal vectors estimation, which is performed by PCA; a method known to be susceptible to noise and outliers. When the point cloud is organized, there exist faster algorithms to compute them [29, 30] at the cost of losing accuracy. Even now, their efficient and robust computation of normal vectors in unorganized point clouds is still an open topic.

These issues related to the complexity of the search space, and their non-deterministic schemes limit dramatically the applications of geometric primitive detection. Therefore, in the following chapters of this thesis, we will describe 3 different approaches in chronological order about how these issues were analyzed and addressed.

2.5 RESEARCH OBJECTIVES

To solve the aforementioned issues of geometric primitive detection, in this thesis, we defined the following objectives.

- To improve the applicability of geometric primitive detection: its accuracy and efficiency.

- To study the problems of geometric primitive detection regarding accuracy and efficiency.
  - Study the schemes of conventional planes and spheres detector.
  - Measure accuracy numerically and qualitatively with ground truth data.
  - Measure efficiency with processing time.
  - Identify failure cases when accuracy and efficiency drops, and hypothesize the reasons.
  - Assess hypothetical solutions derived from the previous objective, on how to improve accuracy and efficiency without being affected with tradeoffs between them.
- To develop a drastically more accurate and efficient geometric primitives detector.
  - Analyze the reduction of the computations needed to generate hypothetical shape models without losing accuracy or robustness to noise and outliers.
  - Develop a method that validates hypothetical shapes efficiently using geometric features to improve robustness.
- To validate the results visually and numerically against ground truth data.
  - Generate point clouds and ground truth shapes from realistic sensor simulation.
  - Select an appropriate evaluation metric for shape detection accuracy and efficiency.
  - Execute comprehensive experiments on the performance of the evaluated methods in at least two metrics corresponding to accuracy and efficiency.
    - Conduct experiments on noiseless and noisy data with challenging sensor patterns.
    - Conduct experiments with short and high-range point clouds.
  - Visualize the experiments results in order to qualitatively compare them against ground truth data.
  - Execute experiments with real sensors data.

2.6 ORGANIZATION

In the following sections, I will describe the prior art in sphere and plane detection focusing on briefly describing their key mechanisms and foundations, and pointing out hints about their weak points.

Conventional plane detection methods are briefly explained in Chapter 3. Then, a plane detector based on the Hough transform is described

in Chapter 4. Consequently, derived from the lessons learned from this plane detector, a novel plane detector based on sliding voxels is introduced in Chapter 5. Under the hypothesis that the sliding voxel approach can be generalized to more shapes, we applied this approach to sphere detection. Therefore, we proposed a sphere detection method based on sliding voxels in Chapter 6.

3

CONVENTIONAL PLANE DETECTION METHODS

3.1 RANDOM SAMPLE CONSENSUS (RANSAC)

RANSAC [16] is a simple but robust model-fitting algorithm, in computer vision is widely used for feature matching but also can be used for lines and plane fitting. It is the most popular method for plane detection. As its name indicates, 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, many variations of the original method have been created. For instance, MSAC & MLESAC [81] provide an enhanced evaluation of the model quality function in the verification step. PROSAC [12] enhances the hypothesis generation by sorting samples using a quality function.

RANSAC is an iterative approach consisting mainly in hypothesis and verification steps. A subset of n random samples are drawn from the point cloud and a hypothetical model $\mathcal{M}$ is generated. For each $\mathcal{M}$, an error metric between the points and $\mathcal{M}$ is computed. The model is verified by counting the number of inliers τ within an error threshold ϵ. If τ does not reach a threshold τ', the process is repeated until k iterations. Once a proper $\mathcal{M}$ is found, then its coefficients should be refined by other methods such as least squares.

3.2 RANSAC FOR PLANE DETECTION

Figure 3.1 illustrates the general strategy of RANSAC for plane detection. Variations and extensions of RANSAC follow this basic approach or an extension. It draws random samples from the point cloud until it finds a good model by thresholding its number of inliers.

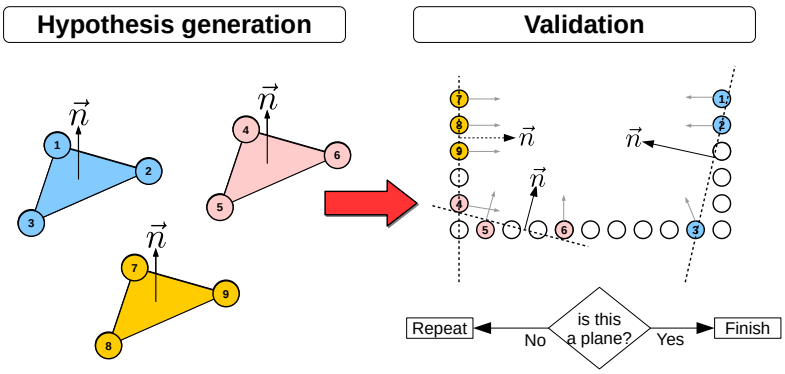
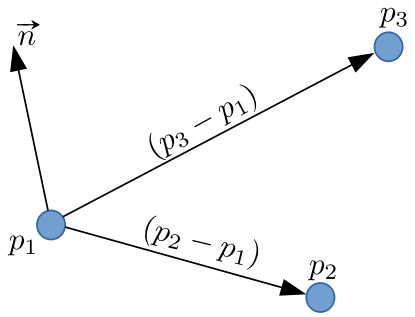


Figure 3.1: RANSAC general strategy for plane detection



$$\vec{n} = (p_3 - p_1) \times (p_2 - p_1)$$

Figure 3.2: A plane defined from 3 points and a cross product

For plane 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 $\hat{n}$ using the cross product of two vectors over the plane,

$$\hat{n} = \tau \vec{n} = (p_2 - p_1) \times (p_3 - p_1), \tag{3.1}$$

where $\tau = |\vec{n}|^{-1}$ and $|\cdot|$ represents the Euclidean norm.

Figure 3.2 shows how the combination of 3 non-collinear points forms a normal vector using the cross product.

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 until k iterations. k is calculated from the probability of drawing good samples from the data, i. e., samples that generate accurate models. Let w be the probability of drawing an inlier from the data, then $b = w^n$ will be the probability of finding a good set of samples from $\mathcal{P}$, then to ensure a probability z of RANSAC to withdraw a valid model it defines k as

$$k = \frac{\log(1 - z)}{\log(1 - b)}. \tag{3.2}$$

In real-world data, particularly in the presence of quantized noise, there is a well-documented study [77] on the bias of RANSAC model fitting in the presence of surface discontinuities.

Additionally, as we encounter with point clouds with a small inliers/outliers ratio, its performance decreases even if we set the probability of success to 99%. For instance, suppose we have a noiseless point cloud with 3 planes. If we want RANSAC to find one plane model with high probability, let's say $z = 0.99$, $w = 1/3$ and $n = 3$, then we have to iterate up to 122 times per plane.

However, if we have noise and other non-planar points (outliers) then we have to modify w and set a lower value, then the maximum iterations

per plane increase drastically. When we test for $w \in \{1/4, 1/5, 1/6\}$ the maximum iterations per-plane increases from 122 to 292, 573, and 992 respectively.

This method is non–deterministic and it has two main disadvantages when applying it to point clouds. First, to find a model we have to assume that the inliers/outliers ratio w is relatively low since even in noiseless point clouds it should approximate the ratio between the number of points of the smallest plane over the whole point cloud, thus increasing the maximum iterations per plane. Second, since point clouds are multi-model, we have to execute RANSAC several times. Clearly, this adds finishing conditions difficult to configure.

Furthermore, from a geometric perspective, RANSAC is only testing plane models against point-model distance inliers, i.e., it is not aware of points curvatures, hence generating spurious models across noisy regions in the case of realistic point clouds.

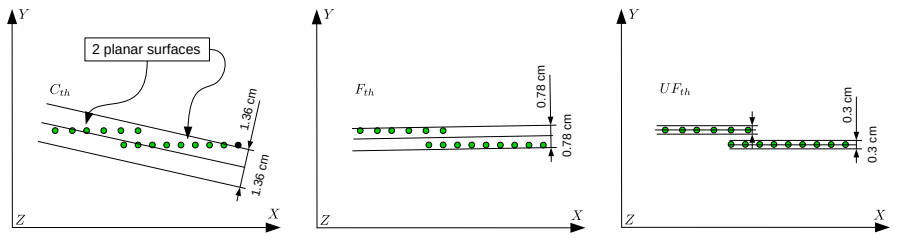
3.2.1 *Coarse-to-Fine RANSAC and Ultrafine RANSAC*

An iterative Coarse-to-Fine RANSAC, hereinafter referred to as CFRANSAC, was developed to be used for virtual keypoints detection [83]. This method employs RANSAC iteratively to detect each model in a coarse-to-fine approach, with Euclidean clustering spatially separating plane inliers in segments before refining its coefficients.

Geometry-aware RANSAC methods like this filter distance inliers by using point normals: a local feature of point clouds that is not sensed and has to be estimated. Apart from the distance threshold ϵ, it introduces an angular threshold θ between the hypothetical plane normal vector and each point normal vector.

First, the coarse step detects surfaces that are roughly planar with big threshold values. However, when these surfaces are neighbors of similar planar surfaces RANSAC tends to report inaccurate results. Therefore, the coarse inliers are refined by using RANSAC again but with stricter thresholds.

Since nearly parallel planes far from each other can be erroneously detected as one, it can produce an error when refining the model coefficients. Thus, planes inliers are spatially clustered, where each cluster is treated as a different model if they meet a specified points



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

Figure 3.3: UFRANSAC process illustration

number threshold τ'. Lastly, plane candidates are refitted against their inliers in a least squares sense.

CFRANSAC does not perform well in the presence of surface discontinuities patterns [77]. Figure 3.3 shows how UFRANSAC works. The points are side-views of two different planar surfaces. Figure 3.3(a) and Figure 3.3(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, an additional step with an ultrafine threshold UF_{th} can improve further the accuracy and detect both planar models in Figure 3.3(c). Therefore, to improve the accuracy an ultrafine step was added to the process after the fine segmentation. This UltraFine RANSAC method is called UFRANSAC.

3.2.2 *Efficient RANSAC*

Efficient RANSAC[73] (EFRANSAC) executes iteratively RANSAC on disjoint random subsets under the assumption that valid planes models will be detected in most of the subsets. Therefore, it only accepts models that are prominent in the number of inliers, and were found in most of the disjoint subsets. Every accepted model removes its inliers from the point cloud and the process iterates again until the finishing criteria is met.

Unlike CFRANSAC, EFRANSAC filters out points with large deviations among their respective normal vectors during random sampling. It also filters points with normals deviating more than a defined angle from the plane normal vector. Moreover, it filters from the inliers only the largest connected component on the plane model by discretizing the inlier points translated to the plane coordinates. Finally, the candidate shapes are refitted using their inliers and removed from $\mathcal{P}$.

Even though its core functionality is faster than conventional RANSAC methods, it requires point-wise normals estimation, increasing its computational cost drastically.

3.3 RANDOMIZED HOUGH TRANSFORM (RHT)

The Hough Transform is an important parametric shape detection method in pattern recognition and image processing literature. It consists of transforming a datum into all the possible instances of a shape it can represent in their Hough space (or parameter space). Models can be anything that can be parameterized.

A well-known bottleneck of the Generalized Hough Transform is the voting procedure. The Randomized Hough Transform (RHT) [89] is a big improvement to the Hough Voting of the Generalized Hough Transform [3] which was originally proposed by Paul Hough [32].

While the Standard Hough Transform performs numerous point-wise calculations, the Randomized Hough Transform [89], RHT, exploits random sampling to accelerate the voting process.

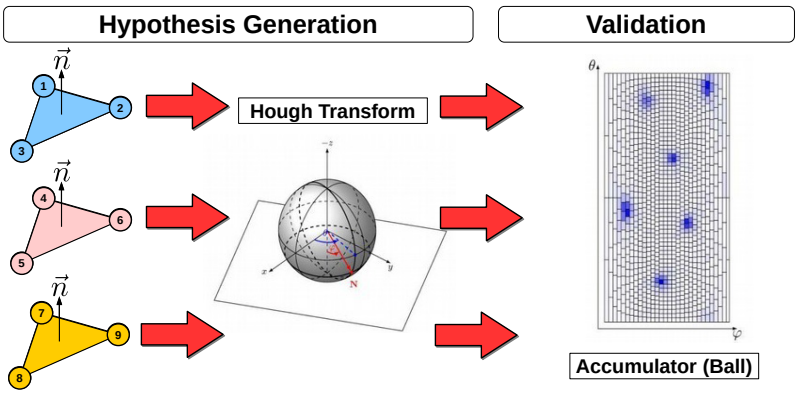


Figure 3.4: General flow of the RHT for plane detection [5]

Similar to RANSAC, in each iteration, RHT selects a sample of 3 points within a given distance, then the spanning plane for those points is calculated using their cross product. Finally, the plane is voted once in an accumulator. Then, a plane model is detected by the algorithm when a cell in the accumulator reaches a given amount of votes. After several iterations, it can approximate the full accumulator data without having to process the entire parameter space.

Figure 3.4 illustrates the general process of the RHT for plane detection [5]. The RHT picks 3 random samples of points within a certain distance and constructs a model. Using the plane coefficients (a, b, c, d) from Equation (2.2), the plane coefficients are then transformed into Hough Space using their polar coordinates (θ, ϕ, ρ)

$$x_i \cos\theta \sin\phi + y_i \sin\theta \sin\phi + z_i \cos\phi - \rho = 0, \tag{3.3}$$

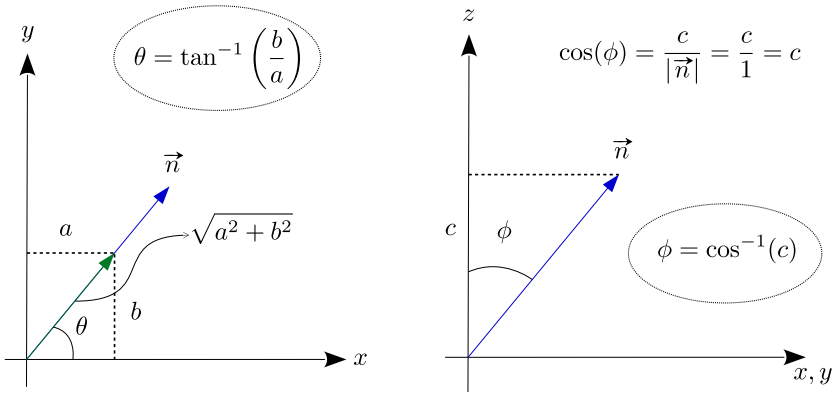
where their equivalences with the cartesian coordinates coefficients are shown in Figure 3.5.

The RHT is not aware of points geometry, but of the parameter space during the voting process. Thus, it does not need normal vectors to compute planes coefficients with reasonable accuracy and speed. A drawback of this approach is that it is not aware of the points distributions of the detected plane models.

Even though, the ball accumulator solved some accumulator problems [5], it is non–trivial to adjust the discretization parameters, among others who affect directly the precision of the algorithm, including measurements to control the minimum and maximum distance between the random samples and a restriction on the smallest eigenvalue size of the spanned planes.

The latter restrictions are not possible only with the plane voting process. Therefore, the RHT needs to be aware of the inliers points distribution by recurring to a sorted clustering of the most prominent planes.

Since it is non-deterministic, the finishing conditions have to be chosen carefully. The algorithm will stop when the remaining points go lower a given threshold, or if the algorithm fails to build a plane model in a given



(a) θ angle (b) ϕ angle

Figure 3.5: The angular parameters of the Hough Transform for 3D planes: θ and ϕ. The distance parameter $\rho = -d$

amount of iterations. If these conditions are not properly configured, the algorithm may not find all the most prominent planes, or it may take longer to stop. Additionally, the performance of the RHT decrease drastically when there is a large number of planes to be detected, or if the number of non-planar points is increased as shown in the hall and arena model experiments of Borrmann et. al [5].

$$\begin{aligned} \rho &= -d \\ \theta &= atan2\left(\frac{b}{a}\right) \\ \phi &= cos^{-1}(c). \end{aligned} \tag{3.4}$$

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[5]. However, it can take longer processing time as the amount of non-planar points in the data increases.

4

FAST AND DETERMINISTIC METHOD FOR PLANE DETECTION

4.1 INTRODUCTION

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 into two types. The first type performs heavy point-wise calculations on point clouds, e.g. Generalized Hough Transform [3]. The second type exploits random sampling to acquire faster speeds, e.g. RANSAC[16]. 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 this chapter we propose a novel filtering method, the Scaled Difference of Normals (SDoN), and an improved Fast and Deterministic Hough Transform (FDHT).

First, SDoN removes 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 it is robust to noise artifacts and has better accuracy while preserving low computational costs.

4.2 PROPOSED METHOD

4.2.1 *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.

Figure 4.1 shows a diagram that describes the flow of the proposed FDHT algorithm. The main concept behind the FDHT is the assumption that normal vectors of coplanar regions are similar.

As seen in Figure 4.2(a), planar regions have identical or similar normal vectors. Therefore it is more likely for a patch to be in a plane when is coplanar.

FDHT starts by dividing the 3D space into voxels using an Octree. As seen in Figure 4.2(b) and Figure 4.2(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,

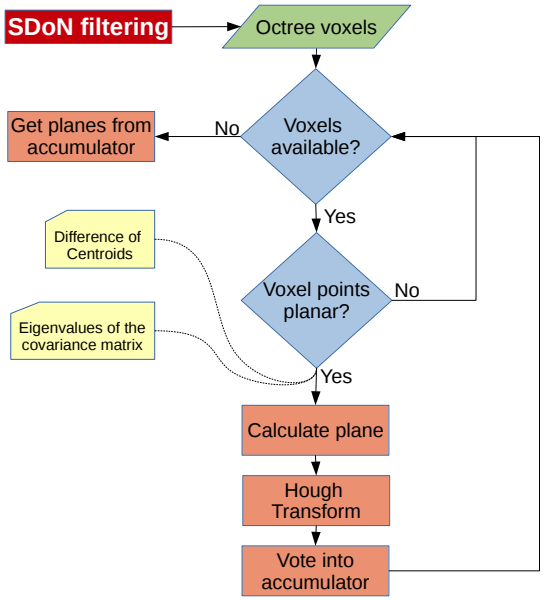


Figure 4.1: Diagram of the FDHT algorithm

centroid, points indices, among others. In FDHT, points indices are stored in leaves to query points inside each voxel later.

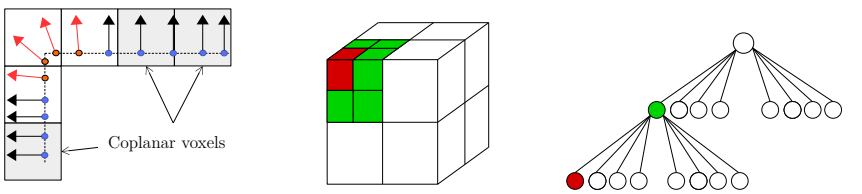
Consequently, for each voxel the planarity and coplanarity of its points distribution is evaluated. For this purpose, since is fast to calculate centroids, Difference of Centroids[27] is used to calculate the probability of a voxel to be planar. Then, we use the centroid μ_p to generate the covariance matrix and its eigenvalues $\lambda_1 < \lambda_2 < \lambda_3$ using PCA.

These eigenvalues are used to filter planar voxels [44], 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,

$$\lambda_2 > \alpha\lambda_1. \tag{4.1}$$

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

$$\beta\lambda_2 > \lambda_3. \tag{4.2}$$



(a) Normal vectors of coplanar voxels

(b) 3D space subdivision of octrees

(c) Voxel hierarchical structure stored in memory

Figure 4.2: Coplanar voxels and octrees

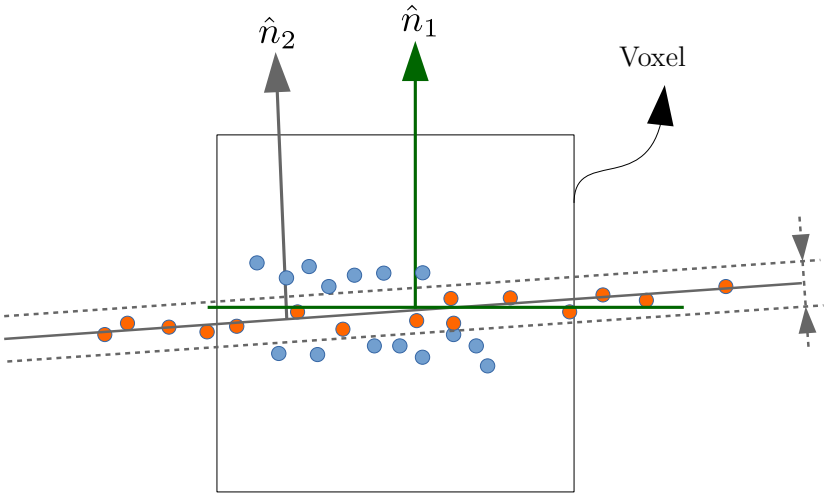


Figure 4.3: Voxel plane refinement using coplanar points

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 $\hat{n}$ 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 Equation (2.1).

To detect coplanarity, we measure the difference between two normals at a different radius from the current voxel centroid. The normal vector of the voxel plane is set to $\hat{n}_1$ and the normal vector at an expanded radius is set to $\hat{n}_2$. Figure 4.3 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 $\hat{n}_1$ and $\hat{n}_2$ will point to different directions. Therefore, if the cosine distance δ between $\hat{n}_1$ and $\hat{n}_2$ is less than 0.1, then the voxel is coplanar with its expanded radius. The cosine distance δ is illustrated in Figure 4.4. Where the absolute cosine $|\cos(\theta)|$ is defined as

$$\psi = |\hat{n}_1 \cdot \hat{n}_2| \mapsto [0, 1] \tag{4.3}$$

since both $\hat{n}_1$ and $\hat{n}_1$ are unit vectors. Therefore, its angle becomes the normalized angular distance δ defined as

$$\delta = \frac{2 \cos^{-1}(\psi)}{\pi}. \tag{4.4}$$

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

$$C_r = \gamma \frac{V_s}{2}. \tag{4.5}$$

Using the Hessian normal form of the plane from Equation (2.1), for every coplanar patch the plane coefficients are calculated using its

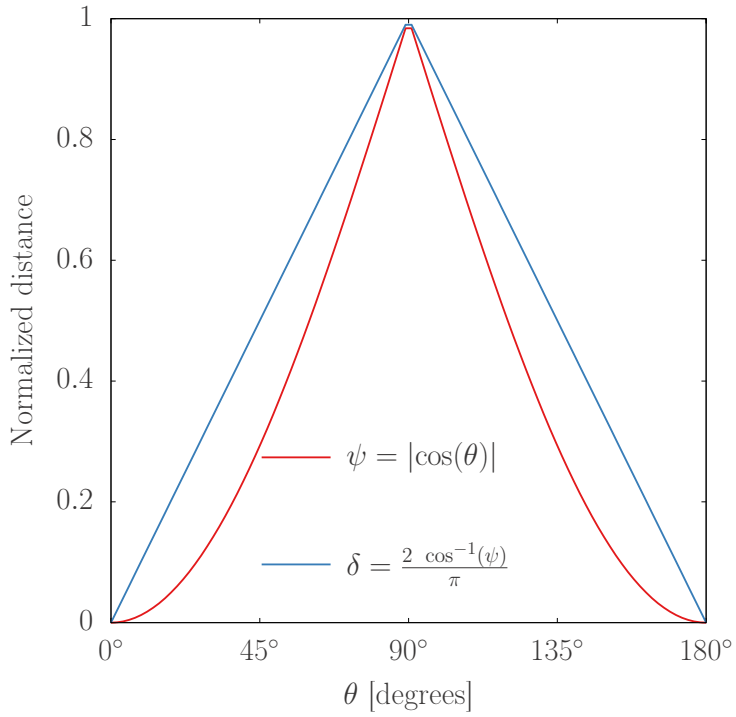


Figure 4.4: Comparison of the positive cosine distance and positive cosine similarity functions

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).

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 cast 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 Figure 4.5. 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.

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

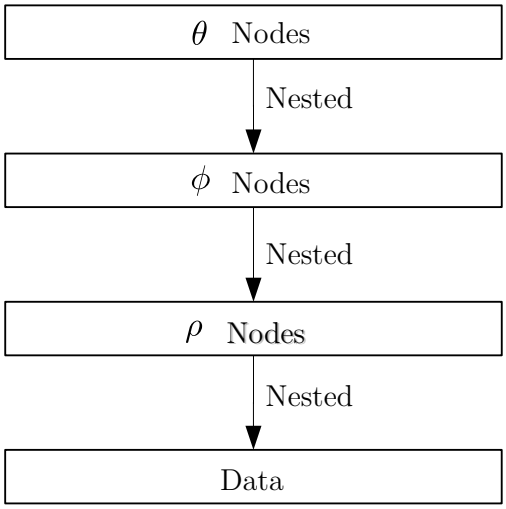


Figure 4.5: Nested map structure of the FDHT accumulator

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 plane refinement.

Iteratively, inliers are selected from inside P_c for each plane reusing the coplanarity threshold. Additionally, the angular distance (Figure 4.4) 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)[35] is an operator that works on multiple scales, it subtracts two normal vectors $\vec{n_1}$ and $\vec{n_2}$ at radius of different sizes, r_1 and r_2. The DoN value is the L2 norm between these two vectors.

$$DoN = \left| \frac{\vec{n_2} - \vec{n_1}}{2} \right|. \tag{4.6}$$

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 Figure 4.4 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.

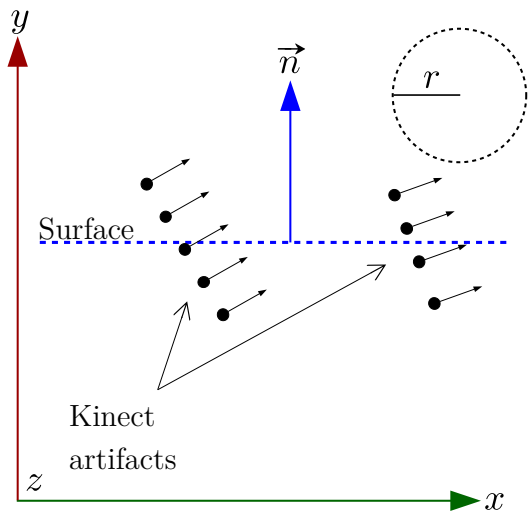


Figure 4.6: Sideview of a normals calculations with a small search radius in the presence of Kinect noise

4.2.2 *Scaled Difference of Normals*

Figure 4.6 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,

$$c = \left\{ \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \in \left[0, \frac{1}{3}\right) \;\middle|\; \lambda_1 < \lambda_2 < \lambda_3 \right\}. \tag{4.7}$$

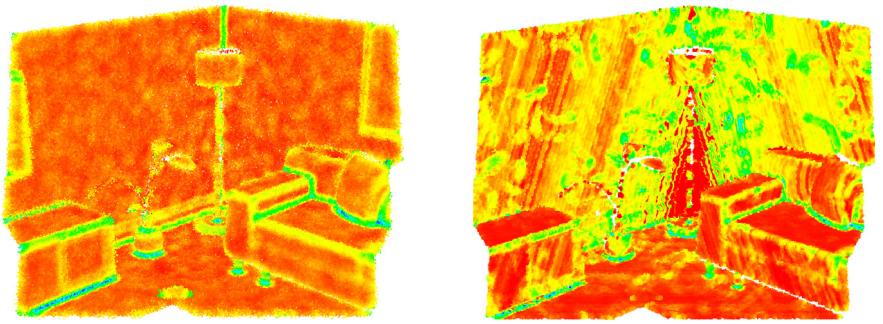
Figure 4.7 shows heatmaps of the local curvatures at each point with a fixed radius of 10[mr]. Figure 4.7(a) has an added gaussian noise of 2[mr]. Figure 4.7(b) features several types of noise artifacts generated by the simulation of a Kinect sensor. Even though the surface thickness of both point clouds were increased similarly due to noise, Figure 4.7(a) shows that when using a uniform search radius to calculate the curvatures using PCA, the curvature pattern is uniform.

However, in Figure 4.7(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 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



(a) Curvatures of the room point cloud

(b) Curvatures of room point cloud with Kinect noise

Figure 4.7: Curvatures heatmap under Gaussian and Kinect noise

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[67]. 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 4.8 shows an example of minimization of noise artifacts using downsampling. The observed location is pointed by Figure 4.8(a). Figure 4.8(b) and Figure 4.8(c) show respectively the front and side views of points over a plane in a simulation of a Kinect scan. The left part shows the noisy Kinect scan. The right part shows the same point cloud but with a downscaling of 15[mr].

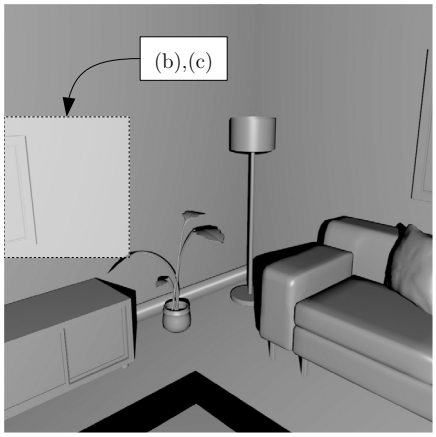
As seen in Figure 4.8(b), the effect of Kinect quantization is not visible because it is view from the sensor perspective. However, after rotating the view, the pattern of Figure 4.8(c) arises and planar clusters can be seen across the surface. When comparing both right and left images, it is clear that 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 scales.

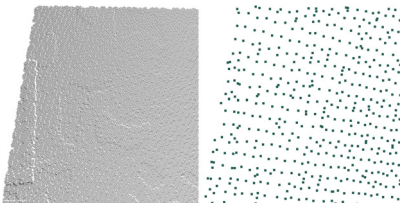
It is noted that the DoN value is treated as a score and not a distance metric, the proper distance metric from Figure 4.4 is used as a reference to define the DoN score:

$$DoN = 1 - \delta. \tag{4.8}$$

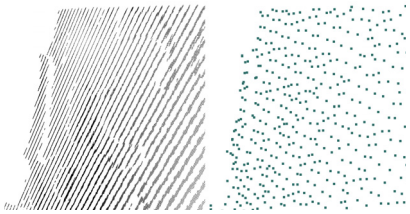
Figure 4.9 is the flow of the SDoN process. The points of the initial downscaling will be the reference points. These are used as the search



(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

Figure 4.8: Planar section affected with simulated Kinect noise and its points distribution before and after Voxel Grid filtering

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 score table for each point and also 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 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 4.1. 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

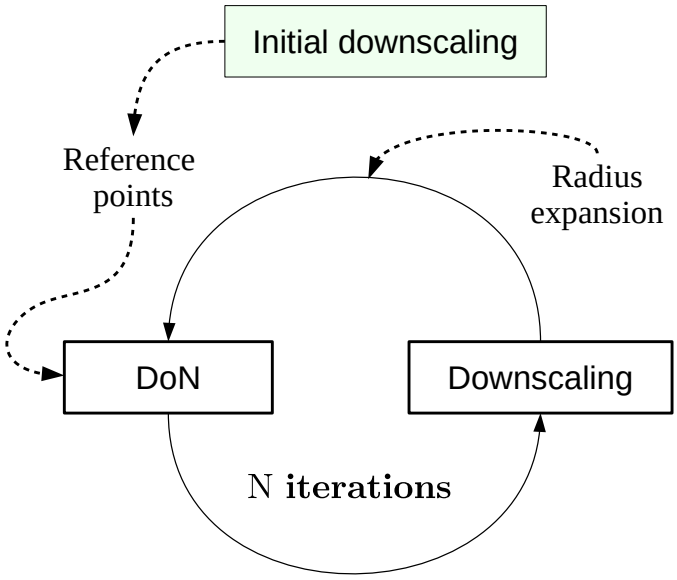


Figure 4.9: Simple diagram of the SDoN calculation process

Table 4.1: 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

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[91]. 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 4.2. They are divided into 2 groups: SDoN parameters, and the improved FDHT parameters.

4.3 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.

Table 4.2: 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]

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.

4.3.1 *Datasets*

We used synthetic point clouds generated by Blensor 1.0.17[23], 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[26] and a kitchen[1]. Additionally, a model of a car[60] with slightly curved surfaces was used to test out which methods are good detecting planarity in this scenario. The car model is particularly difficult for the proposed method since for the underlying 3D planes detection, FDHT, it 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, a Gaussian noise of 1[mr] was added to each point of the synthetic clean datasets. These point clouds are not named by any

1 Marela kitchen red&white, https://3dwarehouse.sketchup.com

Table 4.3: 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

particular annotation; however, the point clouds with Kinect noise are marked as *noisy*.

Figure 4.10, Figure 4.11 and Figure 4.12 show the input point clouds and their details are in Table 4.3. The room point clouds shown in Figure 4.11(a) and Figure 4.11(b) were created with a single Kinect scanning, while the Kitchen point clouds (Figure 4.12(a), Figure 4.12(b)) and Car point clouds (Figure 4.10(a), Figure 4.10(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.

4.4 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 plane coefficients vectors was utilized. This angle δ_θ is based on δ from Figure 4.4, where

$$\delta_\theta = 180\ \delta \mapsto [0, 180]. \tag{4.9}$$

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

Algorithm 1 Precision evaluation

1: **function** BESTMATCH($Planes_{det}$, $Planes_{gt}$)
2: $m \leftarrow$ emptyList
3: **for** P_{gt} in $Planes_{gt}$ **do**
4: **for** P_{det} in $Planes_{det}$ **do**
5: $\delta_\theta \leftarrow angle(P_{gt}, P_{det})$ ▷ Figure 4.4
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: $L \leftarrow$ new List($Planes_{det}$)
11: $M \leftarrow emptyMap$ ▷ matches
12: **while** l not empty **do**
13: $m \leftarrow$ BESTMATCH(l,$Planes_{gt}$)
14: **if** $m[\delta_\theta] \leq 9$ **then** ▷ Degrees
15: $M[match_{gt}].add(m)$
16: $L.del(m_{det})$
17: **else**
18: break
19: $n_d \leftarrow size(M)$ ▷ correct matches
20: **while** l not empty **do**
21: $m \leftarrow$ BESTMATCH(l,$Planes_{gt}$)
22: $\delta_\theta \leftarrow angle(m_{gt}, m_{det})$ ▷ Figure 4.4
23: **if** $\delta_\theta \leq 9$ **then**
24: $M[match_{gt}].add(m_{det})$
25: $L.del(m_{det})$
26: **for** m in M **do**
27: $Err_{gt} \leftarrow \sum m_i[\delta_\theta]$
28: $E \leftarrow \sum Err_{gt_i}$

planes, and lower value leads to a mismatch between the visual and numeric results.

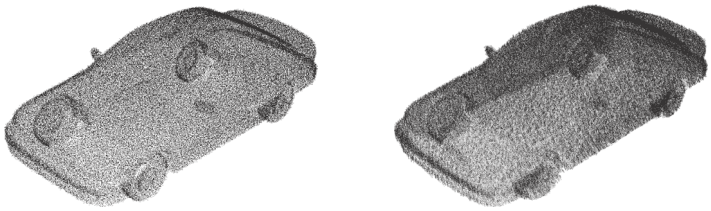
Once we reach the threshold or we run out of detected planes, the algorithm calculates the number of correctly detected planes, i.e., the number of ground truth planes matched with detected planes below 9 degrees.

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

4.5 RESULTS

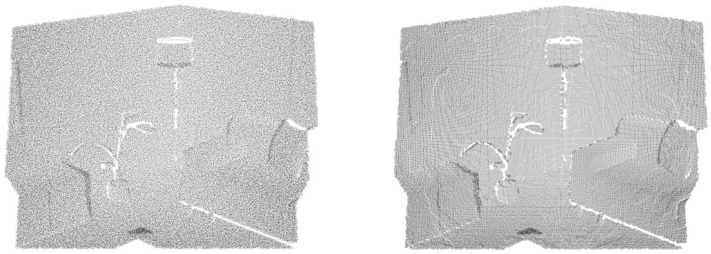
4.5.1 *Efficiency and error*

Figure 4.13(a) and Figure 4.13(b) show the proposed method has competitive accuracy when compared to conventional methods.



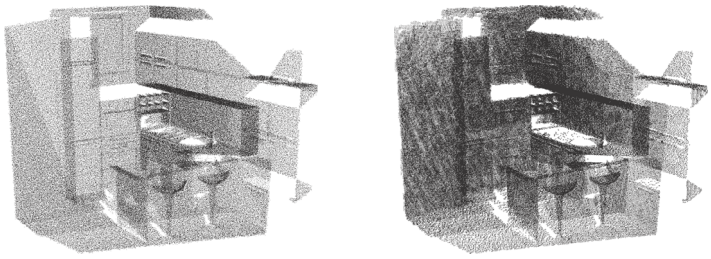
(a) Car (b) Car (noisy)

Figure 4.10: Noisy and noiseless car point cloud obtained by Kinect simulation.



(a) Room (b) Room (noisy)

Figure 4.11: Noisy and noiseless room point cloud obtained by Kinect simulation.



(a) Kitchen (b) Kitchen (noisy)

Figure 4.12: Noisy and noiseless room point cloud obtained by Kinect simulation

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 4.13(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 Figure 4.13(a), we can observe that the proposed method has a low error and is slightly better than CFRANSAC and UFRANSAC. Notwithstanding, Figure 4.13(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 take advantage of the high density of points 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 Figure 4.13(b) remained the highest even though the angular error was lower for CFRANSAC and UFRANSAC.

4.5.2 *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 4.14, Figure 4.15 and Figure 4.16 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 upper region of the seats as part of the Kitchen table plane.

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

In Figure 4.17, CFRANSAC, UFRANSAC and FDHT did not detect the noisiest planar surface. Furthermore, RHT still detected the pillow surface as a plane. 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 plane detection algorithms.

CFRANSAC and UFRANSAC did not detect big planar portions in the noisy Kitchen point cloud of Figure 4.18. Figure 4.19 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 curvatures. Although it showed less angular error than CFRANSAC and UFRANSAC, the qualitative 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.

4.6 CONCLUSIONS AND FUTURE WORKS

The conventional methods for plane 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 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 work 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.

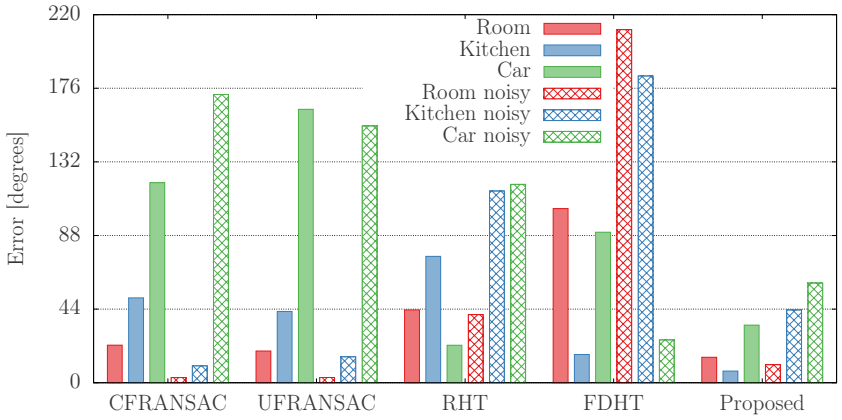
To the best of our knowledge, in this work we propose the first quantitative evaluation of 3D planes detection accuracy. With generated ground truth data, we confirmed numerically which method is good for each scenario.

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 keeping the accuracy high or at least competitive.

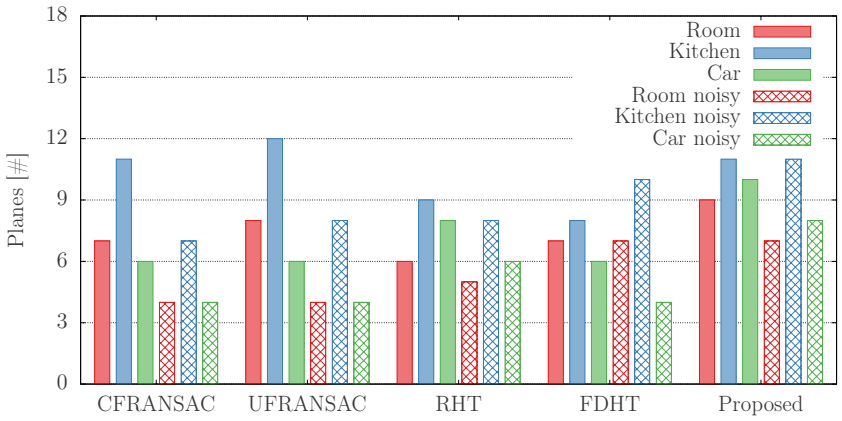
It is noticeable that even though the proposed method provides a superior balance between computational efficiency and precision. RANSAC based methods are simple and can provide good results when there are higher points density near the real surface and if processing time is not important.

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.

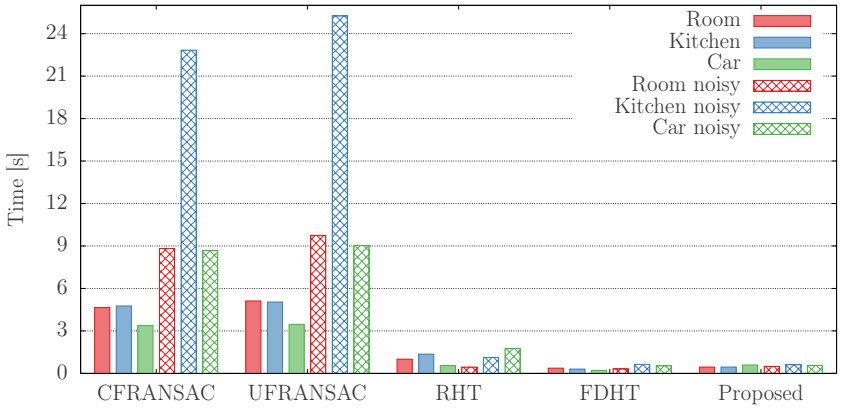
In the future we will assess 3D space subdivision or segmentation alternatives to reduce the sensitivity to the voxel size parameter, as well as to further accelerate the algorithm using GPUs.



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

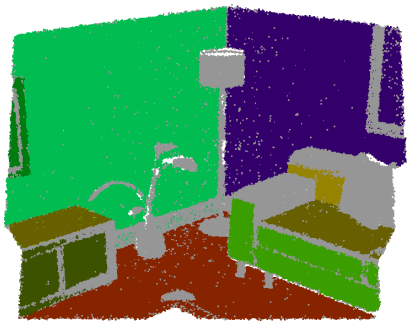


(b) Correctly detected planes, higher is better

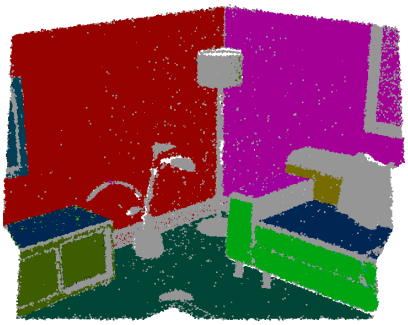


(c) Processing time[s], lower is better

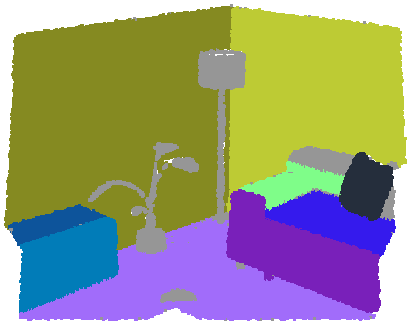
Figure 4.13: Numerical results of the evaluated methods against the datasets



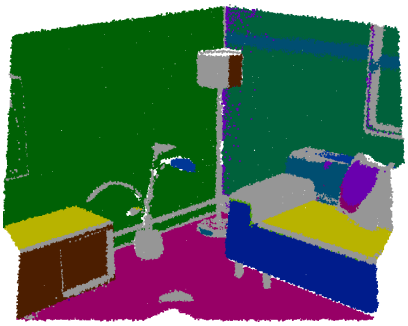
(a) CFRANSAC



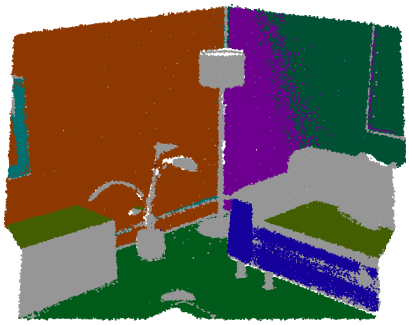
(b) UFRANSAC



(c) RHT

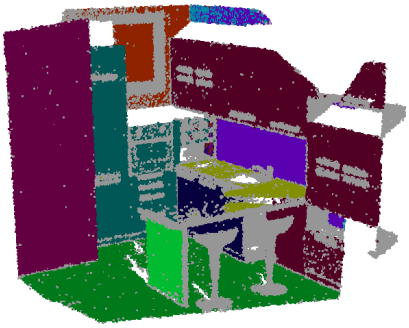


(d) FDHT

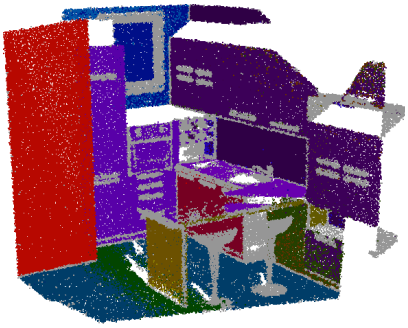


(e) Proposed

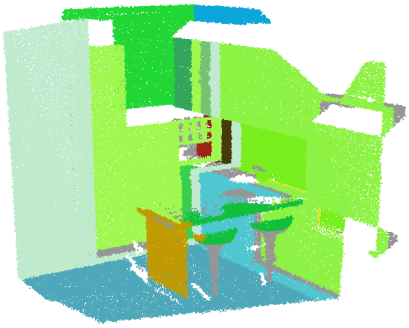
Figure 4.14: Room graphical results



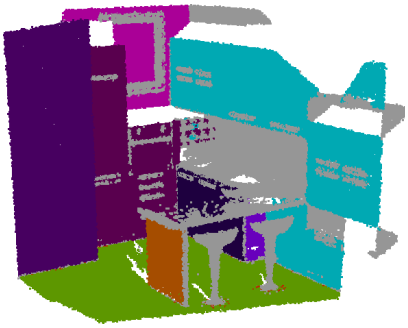
(a) CFRANSAC



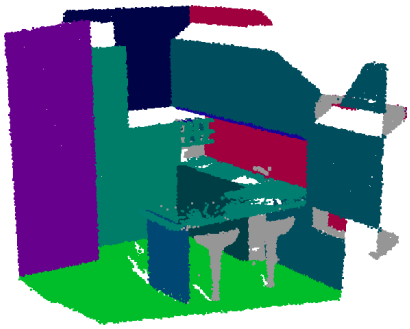
(b) UFRANSAC



(c) RHT

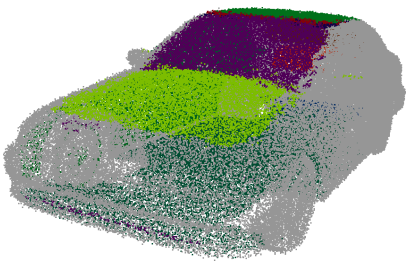


(d) FDHT

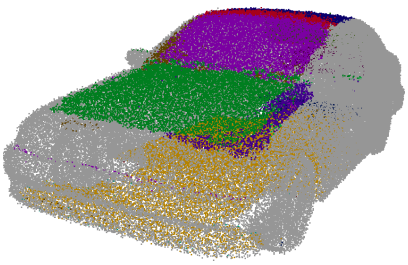


(e) Proposed

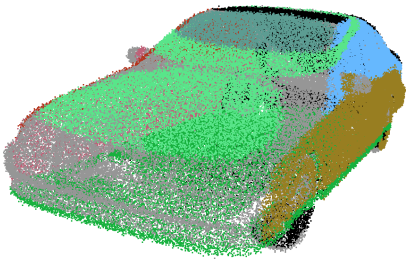
Figure 4.15: Kitchen graphical results



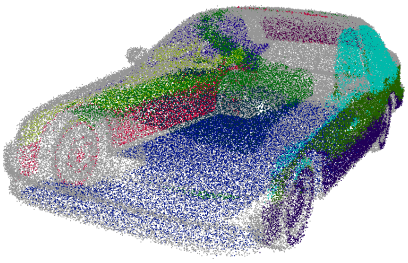
(a) CFRANSAC



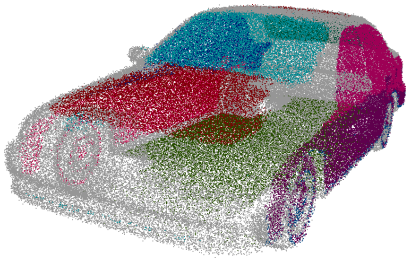
(b) UFRANSAC



(c) RHT

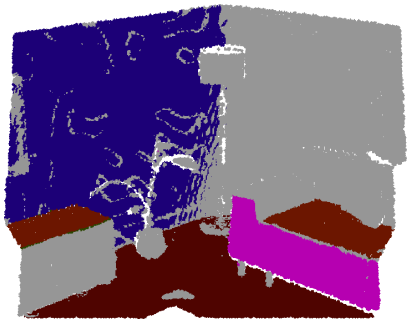


(d) FDHT

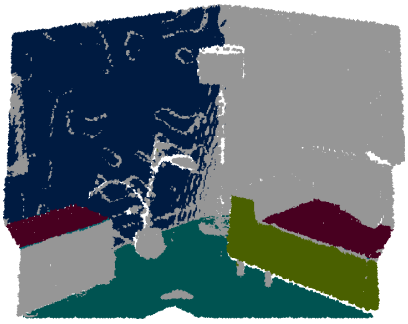


(e) Proposed

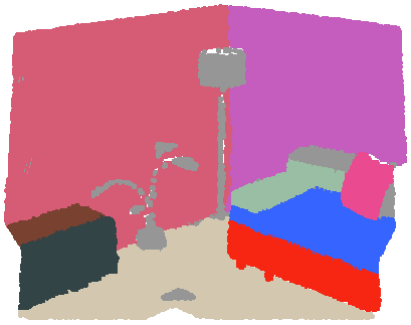
Figure 4.16: Car graphical results



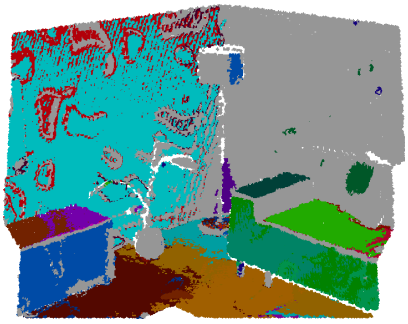
(a) CFRANSAC



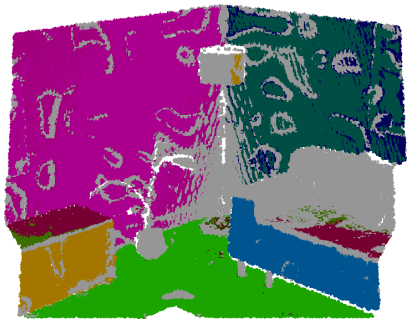
(b) UFRANSAC



(c) RHT

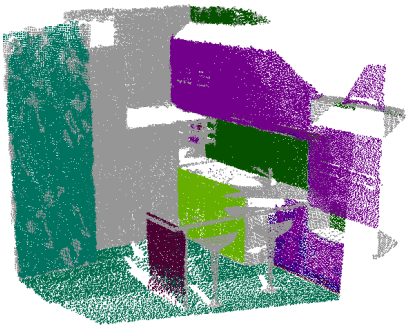


(d) FDHT

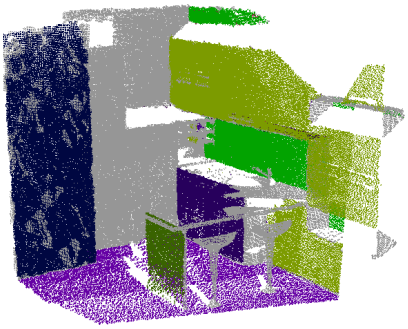


(e) Proposed

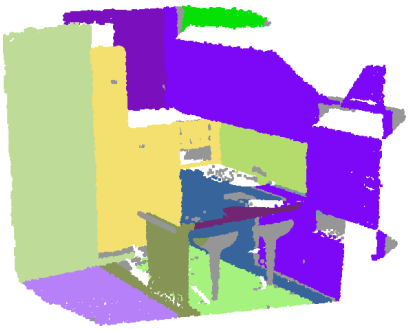
Figure 4.17: Room (noisy) graphical results



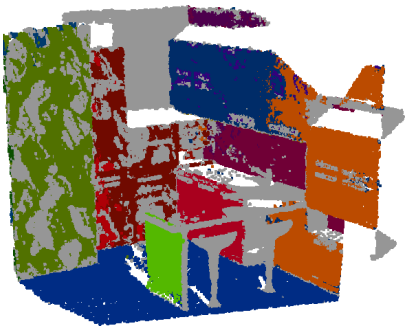
(a) CFRANSAC



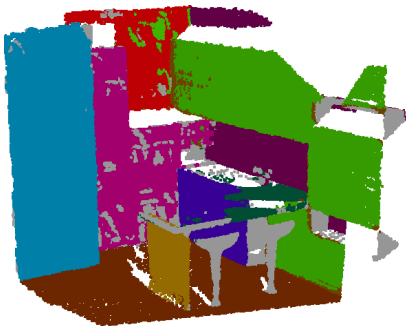
(b) UFRANSAC



(c) RHT

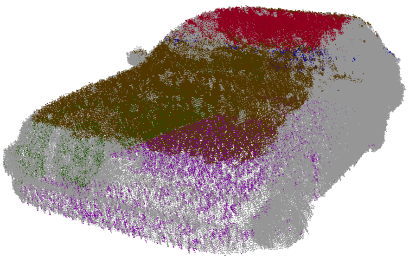


(d) FDHT

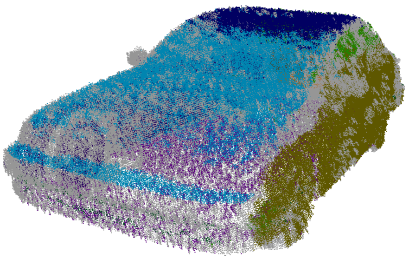


(e) Proposed

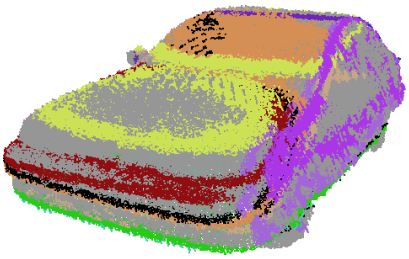
Figure 4.18: Kitchen (noisy) graphical results



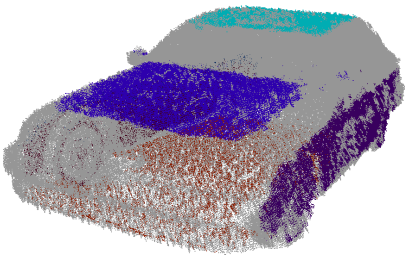
(a) CFRANSAC



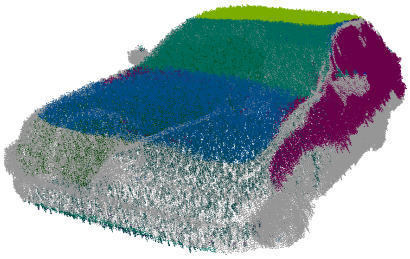
(b) UFRANSAC



(c) RHT



(d) FDHT



(e) Proposed

Figure 4.19: Car (noisy) graphical results

5 SLIDING VOXELS FOR PLANE DETECTION

5.1 INTRODUCTION

In point clouds processing, high variations of sensors noise patterns[2, 55] are challenging problems when dealing with point cloud data. Conventional algorithms tend to fail to detect planar surfaces from commodity sensors. Moreover, relying on random sampling, they need a vast amount of model inliers to perform correctly.

Therefore, in this work we contribute with a drastically faster approach: a sliding voxel–based algorithm. It does not need pre-computed normal vectors; instead, it directly works with the points distribution of overlapping 3D voxels to acquire surface information. We analyze the scattering of each sliding voxel to locate co-planar regions and detect hypothetical planes. Then, planes are extracted from the validation of hypothetical planes against an enhanced subset of the point cloud.

Since ground truth planes data can be obtained from 3D models, we provided experiments with realistic point cloud simulations generated from these models as well as ground truth data. To verify the performance of the proposed method, we measured its efficiency and error against the most popular algorithms: two RANSAC variations[73, 83] and the Randomized Hough Transform[5] for planes detection.

5.2 PROPOSED METHOD

We focused on two main problems of conventional plane detection methods: speed and robustness. The major speed problem of conventional methods is that they need to compute point-wise normals on overlapping neighborhoods [31].

Even though planar surfaces can be described locally, it is more precise to describe them globally. Nonetheless, algorithms that work on the whole set of points are very slow, e.g., the Standard Hough Transform.

Quantized noise from RGB-D and structured light sensors makes it even harder to describe locally planar surfaces, forcing algorithms to increase their search radius, threshold or voxel size; thus preventing to detect smaller objects.

Speed achieved by downsampling can also vanish small surfaces and extremely deviate their normal vectors, as the neighborhood has to be expanded way farther the downsampling region to compute them.

Therefore, we propose a method that efficiently detects planes via sliding voxels. Based on the Sliding Window in images, the proposed method uses a 3D Sliding Window implemented with octree voxels. It

travels through occupied voxels of a point cloud and calculates geometric information about the points distribution using neighbor voxels.

Figure 5.1 outlines the proposed method algorithm. First, we build an octree with voxel size V_s and calculate tangent planes. This local plane fitting provides us with curvature information; therefore, we use sliding voxels to estimate the degree of coplanarity of each voxel.

Since coplanar voxels are more likely to be part of a prominent plane, we sort and mark them as hypothetical planes. Finally, planes are extracted from the validation of hypothetical planes against a geometrically enriched subset of the whole point cloud.

5.2.1 *Hypothetical plane extraction from coplanar voxels*

For each voxel, its centroid $\mathbf{c}$, unit normal vector $\hat{\mathbf{n}}$ and a planarity value P are calculated using the eigenvalues $\lambda_1 \leq \lambda_2 \leq \lambda_3$ and their corresponding eigenvectors $\mathbf{v}_i$ of the covariance matrix, where:

$$\hat{\mathbf{n}} = \frac{\mathbf{v}_1}{|\mathbf{v}_1|} \tag{5.1}$$

and

$$P = \frac{\lambda_1}{\lambda_2}. \tag{5.2}$$

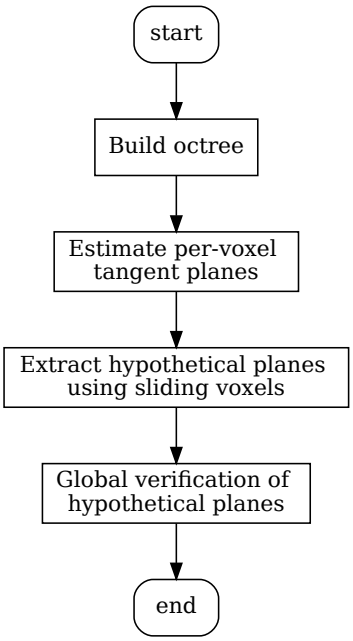


Figure 5.1: Flowchart of the proposed method

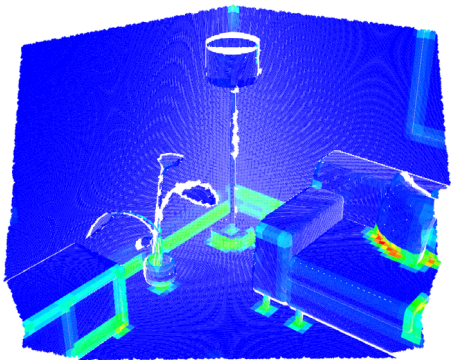


Figure 5.2: Point cloud of a room model, color represents a heatmap of the scores $\mathcal{S}_r$ of each voxel

After the voxel information is processed, the Sliding Voxel walks through the occupied voxels and calculates an overall score $\mathcal{S}_r$ for the whole neighborhood, i.e. 26 neighbors plus the current voxel:

$$\mathcal{S}_r = \sum_{i=1}^{n} \mathcal{S}_i : i \in \mathbb{Z}^+_{\leq 27}, \tag{5.3}$$

where $\mathcal{S}_i$ is the planarity P of the i-th neighbor voxel. As this score gets bigger, the less planar is the neighborhood. Therefore, a low score means the current voxel is coplanar. For a more straightforward parametrization, this planarity measure is normalized using the maximum and minimum observed values:

$$\mathcal{S}_r^* = \frac{\mathcal{S}_r - \min\limits_{i=1}^{n} \mathcal{S}_{r_i}}{\max\limits_{i=1}^{n} \mathcal{S}_{r_i}}, \tag{5.4}$$

where n is the number of occupied voxels.

Figure 5.2 illustrates in warmer colors how this score can approximate regions that have a low probability of being coplanar.

Coplanar voxels are further filtered and sorted in ascending order by their score $\mathcal{S}_r^*$. The resulting coplanar subset $\mathcal{V}_p$ is used to find planes in a centroids point cloud $\mathcal{C}$, that was enriched with their normal vector to form an approximate representation of non-overlapping tangential planes.

5.2.2 *Global verification*

At this point, we can map a voxel centroid to their voxel information such as its score $\mathcal{S}_r^*$, and the total number of points inside the voxel. Simultaneously, its normal vector $\hat{\mathbf{n}}$ and its centroid $\mathbf{c}$ are used to approximate the tangential plane $\mathbf{p}_v$ at $\mathbf{c}$, i.e., a geometrically enriched version of the original point cloud: $\mathcal{C}$.

The purpose of this point cloud is to provide efficient representation to validate hypothetical planes in a global sense. For all coplanar voxels $\mathcal{V}_p$, their tangential plane $\mathbf{p}_v$ is validated against $\mathbf{c} \in \mathcal{C}$ using a decomposed

plane-to-plane distance: the Euclidean distance to each $\mathbf{c}$ and the angular deviation between their normal vectors.

Ideally, the magnitude of the inner product between n normal vectors of a planar surface tend to be 1, i.e. $\frac{1}{n}\sum_{i=0,j=0}^{n} |\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j| \approx 1$ where $i \neq j$. Therefore, inliers are selected by thresholding their Euclidean distance and the normalized angular distance between the hypothetical plane normal vector $\hat{\mathbf{n}}_v^h$, and the corresponding tangent plane normal vector $\hat{\mathbf{n}}_v^t$. This metric is defined by the positive cosine distance $\cos_d^+$ calculated from its similarity $\cos_s$ as follows

$$\cos_s = \frac{\hat{\mathbf{n}}_v^h \cdot \hat{\mathbf{n}}_v^t}{|\hat{\mathbf{n}}_v^h|\,|\hat{\mathbf{n}}_v^t|} \mapsto [-1, 1], \tag{5.5}$$

$$\cos_d^+ = \frac{2}{\pi}\cos^{-1}(|\cos_s|) \mapsto [0, 1]. \tag{5.6}$$

When $\cos_d^+$ takes the value of 1, the planes are orthogonal, and if it is 0, they are parallel. In the proposed method, it is used as a parameter to decide whether an inlier will be rejected or not.

5.2.3 *Plane extraction*

Inliers of the global point cloud are sometimes too disperse and that can lead to false positives. To avoid this issue, a fast 1-cluster euclidean clustering is performed from the refined inliers using a distance threshold defined by the octree voxel size $Th = 4V_s$, which is twice the maximum possible distance between centroids.

From the resulting inliers, a cluster is constructed and its plane is calculated from these via Principal Component Analysis. Since it is impossible to calculate a plane if the number of inliers is less than 3, the plane coefficients are copied from the tangent plane of the voxel with the lowest score in the cluster.

If the cluster does not map enough points on the original point cloud, or if its planarity P (see Equation (5.2)) is not low enough, then the cluster is rejected. Otherwise, it is added to a list of detected clusters which includes its plane coefficients and inliers.

After all hypothetical planes are processed, they are sorted by their number of inliers. This allows us to reuse $\mathcal{C}$ and remove inliers progressively from the most to the least prominent plane.

Once the algorithm removes all the centroids inliers of the hypothetical planes, the process finishes and the list of clustered hypothetical planes becomes the detected planes list. A summary of the proposed method parameters is shown in Table 5.1.

Table 5.1: Proposed method parameters

Parameter	Description
Voxel size[m]	Size of the octree leaves
Planarity threshold	Maximum value for $\mathcal{S}_r^*$
Inliers threshold[m]	Max. euclidean distance to plane
Max cosine distance	Max. $\cos_d^+$ of plane and inliers
Min plane size[#]	Min. support of planes

Table 5.2: Dataset information

Name	BBDD[m]	Points[#]	Planes[#]
$\mathcal{R}$	6.09	295,144	9
$\mathcal{R}$ noisy	6.09	295,144	
$\mathcal{K}$	12.19	249,348	14
$\mathcal{K}$ noisy	12.19	503,398	

5.3 EXPERIMENTAL SETUP

5.3.1 *Datasets and evaluation method*

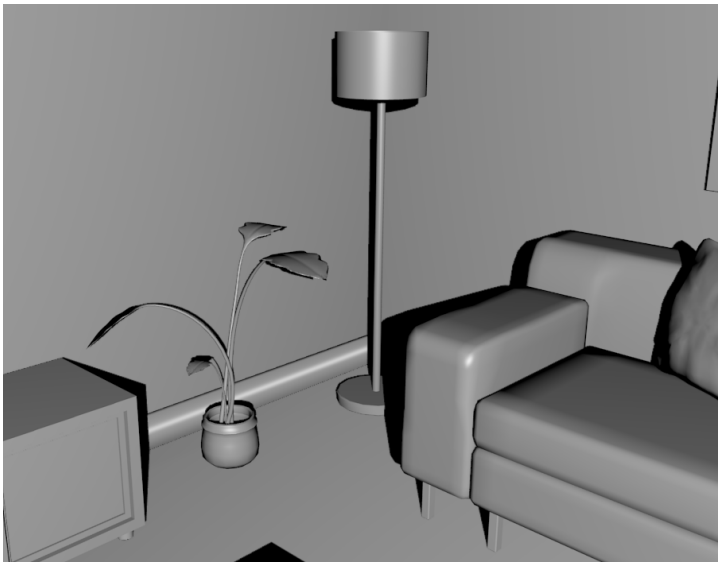
The datasets and ground truth planes used in the experiments can be seen in Figure 5.3 and Figure 5.4. Point clouds from Kinect V1 sensor simulation were built using Blensor 1.0.18 [23].

A room model[26] $\mathcal{R}$ was used to generate noisy and noiseless simultaneous scans. To create a more complicated scenario, a kitchen model[49] $\mathcal{K}$ was scanned by translating and rotating 10 times the sensor in the simulation software. All scans were performed in world coordinates to correctly register the points via concatenation. The resulting point cloud has a wide variety of noise patterns which makes difficult even for humans to detect small planar surfaces in some locations. In addition to the registered noisy scan, we also simulated a clean registration which does not have noise or quantization artifacts. Due to registration, both $\mathcal{K}$ point clouds are incredibly dense. Hence, voxel grid filter of leaf size 0.01[m] was applied to them.

Numerical information of the datasets can be seen in Table 5.2, where BBDD stands for Bounding Box Diagonal Distance, i.e. the length of a diagonal line that crosses the bounding box of the point cloud.

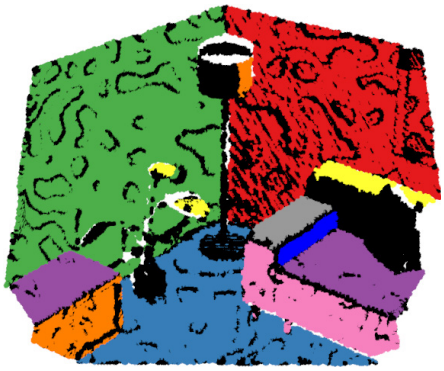
For both $\mathcal{K}$ and $\mathcal{R}$, plane models were extracted directly from the polygons inside Blensor. For each planar surface, plane coefficients were generated from a polygon over each plane by using its normal vector and barycenter. This ground truth data allows us to measure numerically and precisely the accuracy of each plane detection method.

Regarding the processing time T_w, we used wall time since its clock has more resolution. We note that we excluded the time it takes to load a point cloud from disk, also, we included the time for the required



(a) $\mathcal{R}$ model

(b) $\mathcal{R}$ ground truth



(c) $\mathcal{R}$ noisy ground truth

Figure 5.3: Room $\mathcal{R}$ datasets models

preprocessing of each algorithm. For CFRANSAC and EFRANSAC we included the calculation of normal vectors inside T_w since they rely heavily on them. For the proposed method, we also included the neighborhood-aware octree preparation time.

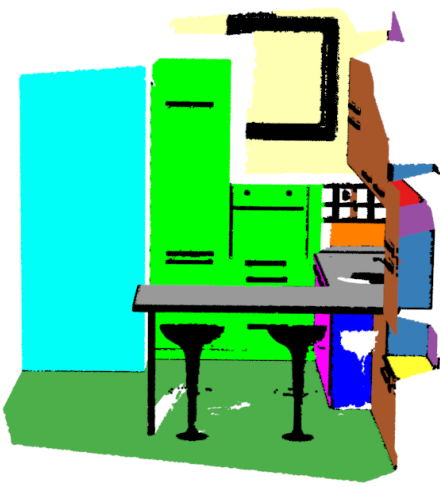
For each run, a list of ground truth planes $\mathbf{P}^{gt}$ and detected planes $\mathbf{P}^{det}$ coefficients is prepared. Then, we defined 2 error metrics. First, the angular error ω (in degrees) between the normal vector of an i-th ground truth plane $\hat{\mathbf{n}}_i^{gt} \subset \mathbf{P}_i^{gt}$ and the normal vector of a j-th detected plane $\hat{\mathbf{n}}_j^{det} \subset \mathbf{P}_j^{det}$, is defined as

$$\omega = 180 \cos_d^+(\hat{\mathbf{n}}_i^{gt}, \hat{\mathbf{n}}_j^{det}) \mapsto [0, 180]. \tag{5.7}$$

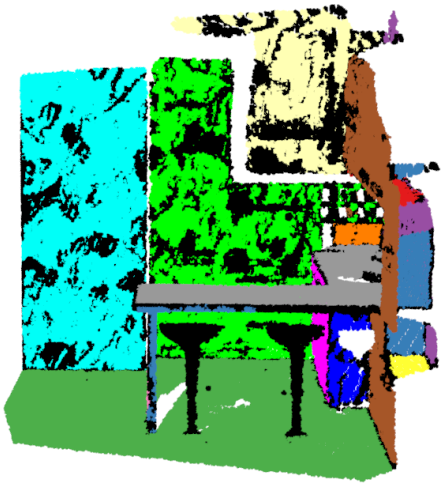
Second, the offset difference δ between the i-th ground truth plane $d_i^{gt} \subset \mathbf{P}_i^{gt}$ and the j-th detected plane $d_j^{det} \subset \mathbf{P}_j^{det}$ is

$$\delta = \left| |d_i^{gt}| - |d_j^{det}| \right|. \tag{5.8}$$

(a) $\mathcal{K}$ model



(b) $\mathcal{K}$ ground truth



(c) $\mathcal{K}$ noisy ground truth

Figure 5.4: Kitchen $\mathcal{K}$ datasets models

For a ground truth plane $\mathbf{p}_i^{gt} \in \mathbf{P}^{gt}$, the best detection match is a plane from $\mathbf{p}_j^{det} \in \mathbf{P}^{det}$ that minimizes their positive cosine distance $\cos_d^+$ as vectors in $\mathbb{R}^4$, then similar to Equation (5.6)

$$\cos_d^+ = \frac{2}{\pi} \cos^{-1} \left(\left| \frac{\mathbf{p}_i^{gt} \cdot \mathbf{p}_j^{det}}{\left|\mathbf{p}_i^{gt}\right| \left|\mathbf{p}_j^{det}\right|} \right| \right). \tag{5.9}$$

Also, a match is rejected if their $\omega > 15$[deg] and $\delta > 20$[cm]. Thus, its result will be a list of matches $\mathbf{K}$ of ground truth planes associated with their best detected plane match satisfying the above conditions.

Let $\mathbf{K}_i = \{\mathbf{P}_i^{gt}, \mathbf{P}_i^{det}\}$, where $i = \{1, 2, \ldots, M\}$ and M is the total number of matched planes (true positives), then we can define the precision γ as

$$\gamma = \frac{M}{|\mathbf{P}^{det}|}, \tag{5.10}$$

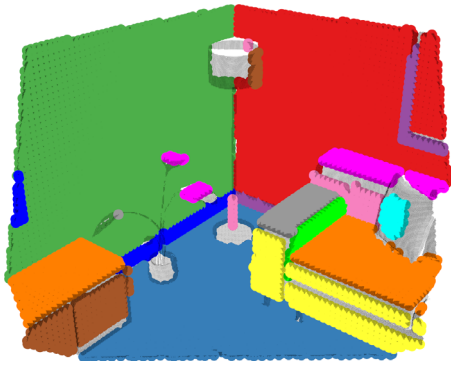


Figure 5.5: Detection example of the proposed method

and recall ζ of the detection as

$$\zeta = \frac{M}{|\mathbf{P}^{gt}|}, \tag{5.11}$$

where $|\mathbf{P}^{det}|$ is the number of detected planes and $|\mathbf{P}^{gt}|$ is the number of ground truth planes. Therefore, the harmonic mean between precision and recall, i.e. the F_1 score[48] is defined as

$$F_1 = \frac{2}{\gamma^{-1} + \zeta^{-1}}. \tag{5.12}$$

The range of the above metrics is $[0, 1]$ where higher values mean better results.

On the other hand, the (true) efficiency E_{ff} measures how fast the evaluated methods detected correct planes, i.e., the number of true positives M, over the processing time T_w

$$E_{ff} = \frac{M}{T_w}. \tag{5.13}$$

Note that since the conventional methods are non–deterministic, we used the average and standard deviation of 50 runs.

Qualitative results are evaluated by segmenting the output planes list $\mathbf{P}^{det}$ of each method. Although inlier points can be obtained from every method natively, their patterns can confuse the reader and generate an unfair comparison. For instance, RHT results may look cleaner while it mistakenly selects protruding noise patterns as part of a planar surface, producing more significant errors in the coefficients of the resulting planes.

A native output of the proposed method is shown in Figure 5.5, for every plane in the detection result, the centroids inliers are shown, then for each plane, a different color was chosen according to the Glasbey lookup table[21], otherwise, points are kept black. Therefore, we segmented the results only by looking at the standard output of the evaluated algorithms: the list of planes coefficients $\mathbf{P}^{det}$.

As a prerequisite, point-wise normal vectors are calculated within a support of k-neighbors via local plane fitting. The segmentation

Table 5.3: Parameters of the proposed method in the evaluation experiments

Parameter	$\mathcal{K}$	$\mathcal{K}$ noisy	$\mathcal{R}$	$\mathcal{R}$ noisy
Voxel size[m]	0.14	0.16	0.12	0.12
Planarity threshold	0.001	0.1	0.02	0.02
Inliers threshold[m]	0.04	0.07	0.07	0.07
Max cosine distance	0.04	0.2	0.05	0.05
Min plane size[#]	500	2500	1300	1000

Table 5.4: Parameters values of the segmentation algorithm

Parameter	$\mathcal{K}$	$\mathcal{K}$ noisy	$\mathcal{R}$	$\mathcal{R}$ noisy
k[#]	50	100	50	100
d_ϵ[m]	0.03	0.05	0.025	0.05
θ_ϵ[deg.]	45	45	45	45

algorithm share similarities with the last step of the proposed method. For each plane $\mathbf{p}_j^{det} \in \mathbf{P}^{det}$, plane-to-plane inliers are selected within a distance threshold of d_ϵ and an angular threshold θ_ϵ. Planes are sorted in descending order by their amount of inliers. In that order, we extract and remove the inliers of each plane from the point cloud. This ensures the most prominent planes are segmented correctly and the segmentation of false positives is minimum.

The parameters of the proposed method are shown in Table 5.3 and for the segmentation are shown in Table 5.4.

5.3.2 *Experiments results and discussion*

The computer used to run the experiments has a CPU Intel Core i7-6700K with 32GB of RAM, it runs on Ubuntu 18.04.2 with PCL 1.8.1 [67], CGAL 4.11 [80], and clang++ 3.8.0. The proposed method was implemented using routines of the PCL library with O3 compiler optimizations.

Figure 5.6 and Figure 5.7 show the visual results of executing the segmentation algorithm over the ground truth and the resulting planes of the evaluated methods.

Figure 5.6 shows the visual results of the noiseless point clouds. Figure 5.6(a) and Figure 5.6(f) depict the segmented ground truth planes of $\mathcal{R}$ and $\mathcal{K}$ respectively. In Figure 5.6(a), we can visualize the ground truth planes selection for the $\mathcal{R}$ model, avoiding slightly curved surfaces such as the backrest of the sofa and the pillow. Additionally, we avoided selecting parallel planes that are not far from each other, because it would be sporadic to detect those structures in noisy point clouds.

The upper row of Figure 5.6 shows the results of the $\mathcal{R}$ point cloud. Noticeably, the proposed method has zero false positives in Figure 5.6(b).

RHT detected spurious planes in the bookshelf as well as several planes over the sofa backrest in Figure 5.6(c). CFRANSAC erroneously detected the pillow and the sofa backrest as planar in Figure 5.6(d). Although EFRANSAC detected some planes with good precision in Figure 5.6(e), it tends to detect several spurious planes while it failed to detect even a prominent planar structure such as the wall on the left side.

The second row of Figure 5.6 shows the results of the $\mathcal{K}$ point cloud. Because of its lack of noise and higher density, most methods performed accurately except for RHT, which could not detect the dining table as seen in Figure 5.6(h). EFRANSAC performed fairly good in Figure 5.6(j) because the higher density of $\mathcal{K}$ allows its random subsets to be more descriptive. However, its precision and recall are lower than the proposed method (as described numerically later).

Figure 5.7 shows the results of executing the evaluated methods over the noisy datasets. Figure 5.7(a) and Figure 5.7(f) depict the segmented ground truth planes similar to Figure 5.6.

The first row of Figure 5.7 illustrates the segmentation results on the $\mathcal{R}$ noisy point cloud. In Figure 5.7(b), the proposed method detected most of the planar structures with high accuracy and no spurious planes, showing similar segmentation patterns when comparing its results with the ground truth. Noticeably, CFRANSAC detected several spurious planes in the noisiest region of the point cloud in Figure 5.7(d), whereas EFRANSAC detected less spurious planes than CFRANSAC in Figure 5.7(e).

The second row of Figure 5.7 shows the segmentation results on the $\mathcal{K}$ noisy point cloud. Figure 5.7(g) shows that the proposed method detected most of the planes while having no false positives. In Figure 5.7(h), illustrates that RHT was able to identify the most prominent planes; nonetheless, it detected fewer planes than the proposed method. CFRANSAC detected more false positives in Figure 5.7(i) while it failed to detect several planar structures. In Figure 5.7(j) EFRANSAC detected slightly more planes than the proposed method because it got benefited with the higher density of the $\mathcal{K}$ noisy point cloud. However, its precision is inferior as described numerically later.

While we confirmed the robustness of the proposed method visually, now we show objective assessment. Processing time, precision and recall are shown in Figure 5.8. Each bar represents the result of executing an evaluated method over a point cloud of the dataset, where smaller bars denote better results. For the conventional methods we used the average of 50 executions and show their standard deviation as error bars.

Figure 5.8(a) shows the processing time T_w in logarithmic scale. There we can confirm that the proposed method is drastically faster than the conventional methods in every case.

The variations in terms of the mean angular and offset error of the evaluated methods are negligible; therefore, we show their assessment based on more standard metrics used in binary classification tasks: precision γ, recall ζ and F_1 score.

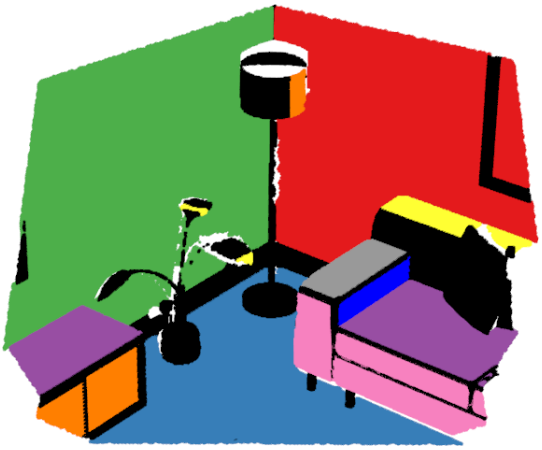
Figure 5.8(b) shows the precision γ of the evaluated methods (see Equation (5.10)). This metric measures how relevant were the detection

results. Noticeably, the proposed method precisely detected appropriate planes in all tested datasets.

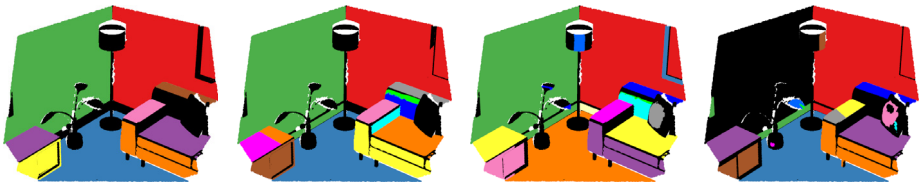
In Figure 5.8(c), we show the recall ζ as defined in Equation (5.11). This metric tells us how many of the ground truth planes were detected by the evaluated methods. The proposed method most of the time detected more ground truth planes than the evaluated methods. Only in the $\mathcal{R}$ noisy point cloud, its recall is paired with EFRANSAC; noticeably EFRANSAC had a very low precision (<0.6) as shown in Figure 5.8(b).

In addition to the above metrics, we evaluated the F_1 score and efficiency E_{ff} of the evaluated methods as defined in Equation (5.12) and Equation (5.13) respectively. Figure 5.9 shows the results of applying these metrics on the experiment results. Figure 5.9(a) shows the overall precision of the evaluated methods. The proposed method shows a superior precision in every case; it had better scores than the best execution of the conventional methods. Figure 5.9(b) shows the efficiency of the evaluated methods in logarithmic scale. Here, we confirm the proposed method detects planes more accurately in a drastically more efficient way.

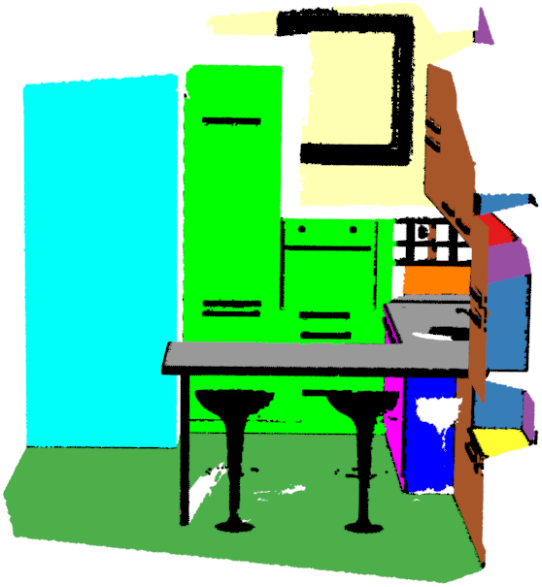
Furthermore, it should be noted that the parameters set of the proposed method is smaller and easier to configure. It has only 5 parameters while EFRANSAC, CFRANSAC, and RHT have 6, 12, and 16 parameters respectively.



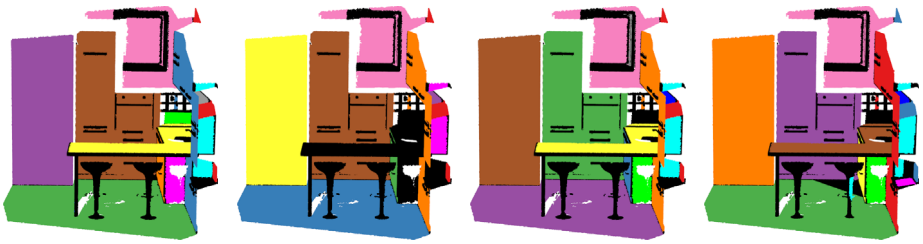
(a) $\mathcal{R}$: Ground truth



(b) $\mathcal{R}$: Proposed (c) $\mathcal{R}$: RHT (d) $\mathcal{R}$: CFRANSAC (e) $\mathcal{R}$: EFRANSAC

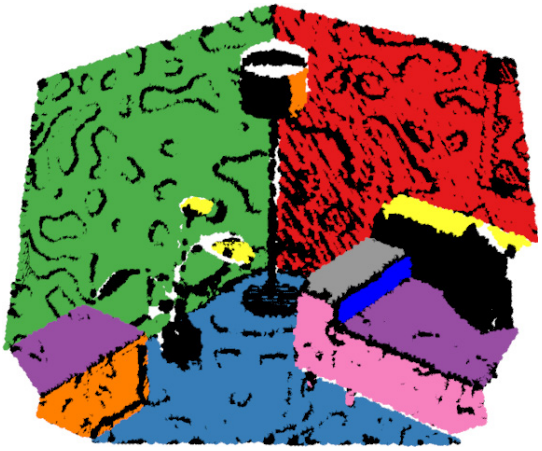


(f) $\mathcal{K}$: Ground truth

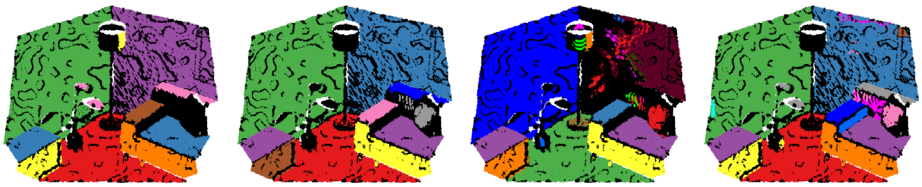


(g) $\mathcal{K}$: Proposed (h) $\mathcal{K}$: RHT (i) $\mathcal{K}$: CFRANSAC (j) $\mathcal{K}$: EFRANSAC

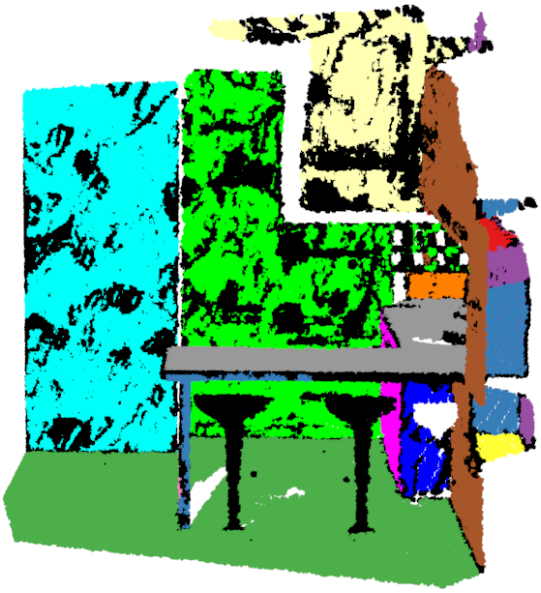
Figure 5.6: Plane detection inliers using the noiseless dataset



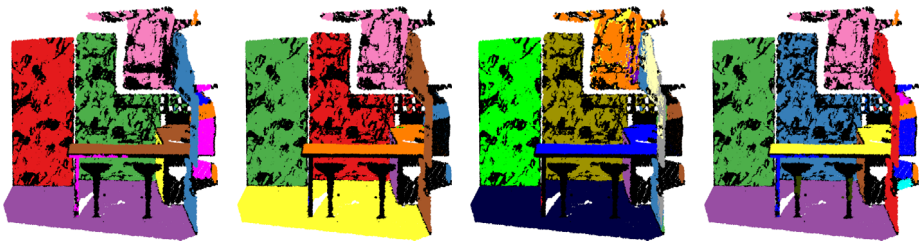
(a) $\mathcal{R}$: Ground truth



(b) $\mathcal{R}$: Proposed (c) $\mathcal{R}$: RHT (d) $\mathcal{R}$: CFRANSAC (e) $\mathcal{R}$: EFRANSAC

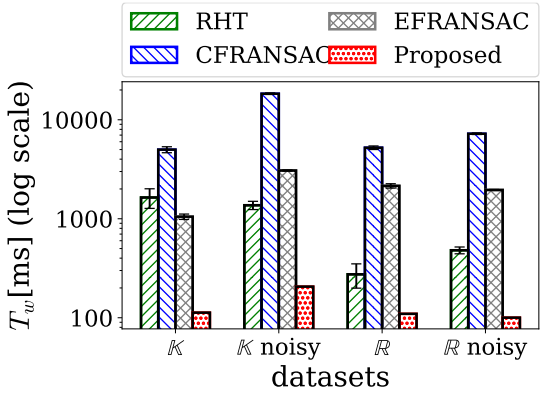


(f) $\mathcal{K}$: Ground truth

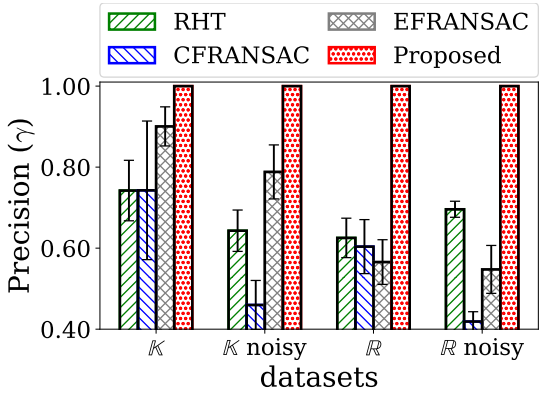


(g) $\mathcal{K}$: Proposed (h) $\mathcal{K}$: RHT (i) $\mathcal{K}$: CFRANSAC (j) $\mathcal{K}$: EFRANSAC

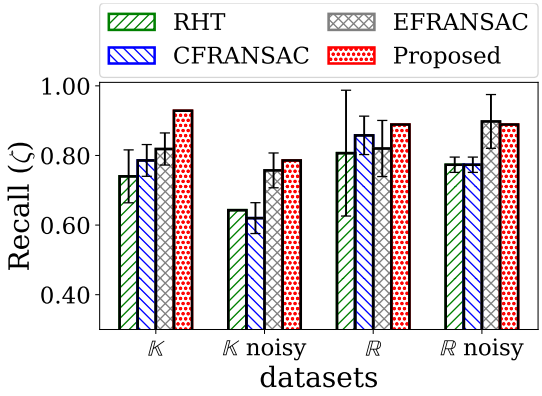
Figure 5.7: Plane detection inliers using the noisy dataset



(a) Processing time T_w (lower is better)

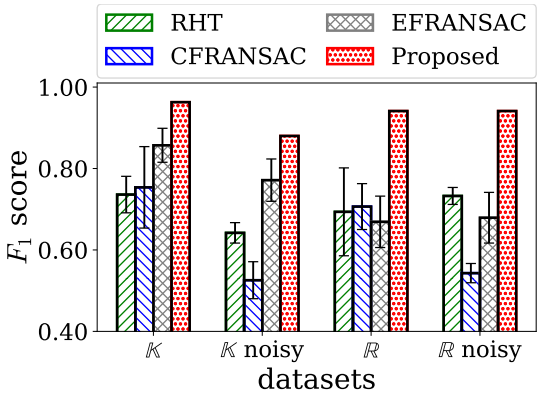


(b) Precision γ (higher is better)

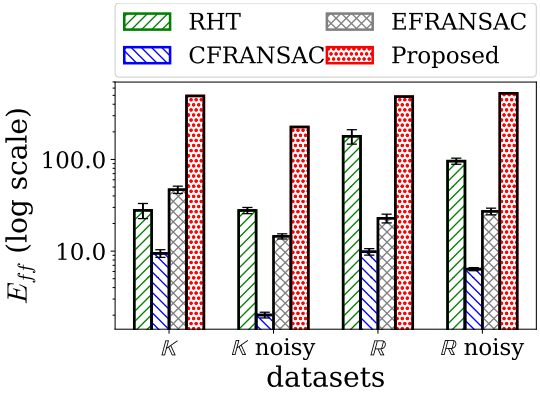


(c) Recall ζ (higher is better)

Figure 5.8: Processing time, precision and recall of the evaluated methods



(a) F_1 score (higher is better)



(b) Efficiency E_{ff} (higher is better)

Figure 5.9: Accuracy and F_1 score of the proposed method compared against the conventional methods

6 SLIDING VOXELS FOR SPHERE DETECTION

6.1 INTRODUCTION

Point clouds are sets of points in an $\mathbb{R}^3$ space that resemble the surface of objects. They can be obtained by a wide variety of laser-based sensing techniques and photogrammetry. However, because of these diverse ways of obtaining them, we face several problems such as huge variations in points density, sensing patterns, sensor artifacts, and noise.

Point clouds can be categorized as organized and unorganized. Organized point clouds can be arranged into an image-like 2D matrix, in which each pixel is associated with a 3D point. These point clouds come from range/depth sensors or stereovision. On the other hand, unorganized point clouds have no specified order and are just a list of 3D points, which are a more general form than the organized counterparts. These come from sensors that change their coordinate system while scanning, like rotating-head 3D LIDAR sensors or dense point clouds from photogrammetry. However, because unorganized point clouds have no order, basic techniques such as nearest neighbor searching become increasingly difficult and time-consuming in unorganized point clouds. Therefore, in this paper, we focus on developing algorithms for unorganized point clouds, which are in high demand because of their numerous applications.

Man-made objects can be approximated by geometric primitives such as spheres, planes, and cylinders[9, 73]. These geometric primitives can act as proxy entities for other real-world objects, for instance, the trunk of a tree and human arms and legs can be approximated by cylinders, also, the head of animals and humans can be approximated as spheres.

Sphere detection is an important technique in 3D computer vision with applications in broad areas such as materials engineering[37], measuring[53], medicine[20] among others. As opposed to other parametric shapes such as planes or other quadric surfaces, spheres have a viewpoint-independent geometry. This property allows us to calculate its coefficients with partial views such as those obtained from 3D sensors. Therefore, they are preferred as targets in in point clouds registration [17, 86, 92] where it is crucial to detect their coefficients with the highest accuracy.

However, the conventional sphere detection methods [1, 7, 56, 73, 76] fail to detect spheres when the inliers ratio becomes too small due to noise, and the range of the point cloud.

Moreover, the processing time can increase exponentially as some algorithms depend on point-wise normal vectors, and their accuracy is highly dependent on their estimation. Also, normal vectors are calculated via Principal Component Analysis (PCA), a method known to be very susceptible to noise and outliers.

Therefore, in this paper, we propose a novel, highly accurate, robust, and drastically more efficient (high speed) method for sphere detection based on sliding voxels. Instead of random sampling, it uses an efficient octree subdivision to detect robustly hypothetical spheres deterministically. Then, the most prominent spheres are detected via Hough voting [3, 32, 33] . Lastly, its coefficients are refined and pruned by their completeness. Experiments with synthetic and real point cloud data from Terrestrial Laser Scanning (TLS) confirm the superior performance of the proposed method.

6.2 PREVIOUS WORK

6.2.1 *Literature overview*

Several approaches to detect spheres using spatial i-nformation have been developed in recent years. A survey work [38] summarizes the conventional methods for geometric primitive detection from 3D data. Region growing clusters similar regions of the point cloud at random locations iteratively. Although computationally expensive, it can be used to later fit geometric primitives in segmented regions.

Hough Transform (HT) [3, 32] based algorithms for parametric shape detection have been proposed in the past. As spheres become perfect circles when projected into a plane, the circular HT was applied to detect spheres as circles in images [39] . Unfortunately, unorganized point clouds do not have a defined projection to an image array. Moreover, when using images to detect spheres, the radius of circles vary depending on the distance from the sensor.

The parameter space for the sphere has four dimensions. Therefore, applying the Standard Hough Transform (SHT) [3] is unfeasible both in computational complexity and in memory usage of the accumulator array used to store votes and detect shapes. Ogundana et al. [56] proposed a fast HT detector by fixing the radius, casting a single vote for each point, and detecting spheres using a sparse accumulator. Nonetheless, they require computationally expensive point-wise normals estimation. Also, fixing the radius limits its applications in real-world scenarios where it is usually unknown.

Abuzaina et al. [1] also proposed a HT for sphere detection using a sparse accumulator. Their approach uses a polar representation of the parameter space and a more exhaustive per-point voting. To overcome this, they fixed the radius and reduced the number of input points based on the points' density of a Kinect sensor limited at a certain range.

In the past, strategies based on random sampling were proposed to reduce the computational complexity of the voting phase in Hough transform methods. The Probabilistic Hough Transform (PHT) [40] reduces the number of evaluated points by selecting a random subset. The Combined Multi-Point Hough Transform (CMPHT) [7] follows the PHT approach to reduce computational complexity, and evaluates several Hough transforms for sphere detection. A single-point HT with

coarse accumulator quantization serves as an efficient coarse approximation that identifies regions-of-interest (ROI) where spheres are more likely to be found. Then, a 4-point Hough transform is chosen as a detection refinement over the ROI. Experiments with real point clouds with a Kinect sensor showed that the computational complexity and the success rate of CMPHT are highly affected by the inliers ratio

$$\phi_r = \frac{N_{\text{in}}}{N}, \tag{6.1}$$

where N_{in} is the number of inliers of the point cloud and N is the number of total points, which means that lower ϕ_r contains more outliers induced by noise or other non-spherical surfaces.

Reducing the dimensions of the parameter space was an unexplored approach for primitive detection. A recent work [76] shows a study of a multi-shape and multi-model detector based on Point-Pair Features (PPF). However, it expects an input cloud with normal vectors, and the PPF is highly dependent on the correct estimation of its normal orientation. Although it outperforms state-of-the-art primitive detection in their experiments, the dataset they used were CAD models and point clouds with removed background and geometric primitives present in the foreground where points are denser and less noisy. Theoretically, if we remove the computational complexity of normal vectors estimation, its processing time is still heavily dependent on the total number of points since it needs to compute PPFs for every unique combination of point-pairs with normals. Therefore, the authors restricted the number of points to compute to 2048 random points, which is unacceptable for long-range (long distance between the sensor and its farthest point) point clouds such as those used in TLS or large-scale photogrammetry.

6.2.2 *RANSAC-based methods*

RANSAC tries to fit a model into a point cloud by random sampling iteratively. The number of trials k is defined by

$$k = \frac{\log(1-z)}{\log(1-b)}, \tag{6.2}$$

where z is the probability that at least one of the data points is error-free, and b is the probability that any set of selected data points is within the error tolerance to the model. Finally, it selects the model that has above a defined number of inliers within a threshold distance. As b can be calculated from the sampling, z is a parameter defined by the user.

RANSAC and its variations, such as MSAC [81] are implemented alongside geometric models of planes, spheres, and cylinders. However, the PCL implementation follows the original RANSAC [16], which is a single-instance and single-model fitting algorithm. When detecting more than one sphere using this approach, we would need to iteratively detect and remove inliers each time we find a good model. Hence, decreasing

ϕ_r in each successful detection making it increasingly difficult to detect more shapes, and to decide finishing conditions.

Wang et al. combine a RANSAC-like sampling strategy with energy minimization to detect spheres in Kinect point clouds [85]. It starts by drawing a small set of hypothetical sphere models from random samples and then use energy minimization to label spherical points. Although it is not dependent on distance thresholds, its accuracy depends on the initial sphere models and weighing terms of the energy minimization function, which have to be guessed by the user depending on the outliers rate.

Spheres are a subtype of quadric surfaces (quadrics); therefore, they have nine Degrees-of-Freedom (DoF) parameters. A study [4] provides a quadric detector algorithm that overcomes current limitations by using RANSAC to search for sets of points (basis) that are likely to be on a quadric surface. These bases provide a coarse identification of a quadric surface to be further refined. Nonetheless, they proposed to fit linearly, a non-linear (quadric) surface. Hence, their fitting results are biased, and the effect of this bias in the final model coefficients of their method is unknown since it was out of the scope of their study [4] . Therefore, this method is not suitable for applications where the accuracy of the resulting sphere coefficients is crucial [17, 86, 92]. Furthermore, selecting an appropriate basis out of a long-range point cloud becomes complex task and highly depends on the inliers ratio of the point cloud (ϕ_r) since it inherits RANSAC disadvantages.

6.2.3 *Efficient RANSAC*

Efficient RANSAC (EFRANSAC) [73] uses octrees for a more efficient localized sampling. For spheres, it uses two random sampled points with their normal vectors to generate hypothetical spheres. Then, iteratively executes RANSAC over disjoint random subsets of the point cloud to validate the generated model.

Each sphere is refined by thresholding the expected curvature at each point N_{th}. Also, they map sphere inliers within a threshold ϵ to a low-distortion bitmap that resembles the surface of the sphere. This bitmap of bin size C_ϵ allows EFRANSAC to select the biggest connected component as the final inliers of each hypothetical sphere. The final coefficients are refined using non-linear least squares[74].

EFRANSAC finishing condition takes into account the octree level of the samples and is parameterized similarly to RANSAC. At a lower z, these algorithms will increase their determinism at the cost of increasing their iterations. According to the EFRANSAC implementation in the CGAL library[80], z is thresholded against

$$\text{stop}_p = \left(1 - \frac{|L_c|}{4|P|O_{\text{depth}}}\right)^{|C|}, \tag{6.3}$$

where L_c is the largest candidate size (in number of points), and $|P|$ is the number of available points that are not part of the selected shape

Table 6.1: Summary of conventional methods

Method	Detection approach	Efficiency strategy	Drawbacks
Fast HT [56]	Single-vote HT	Fixing radius, sparse accumulator	Fixed radius
Kinect HT [1]	SHT [3]	Background points removal	Point-wise search
CMPHT [7]	PHT [40]	Coarse-to-fine voting, random subsampling	Weak to low ϕ_r
PPF Hough voting [76]	PPF [14]	Random subsampling (up to 2048 points)	High combinatorial complexity, weak to low ϕ_r
Wang et al. [85]	RANSAC[16], energy minimization	Random sampling	Weak to low ϕ_r
Birdal et al. [4]	RANSAC[16], quadric voting	Random sampling	Weak to low ϕ_r, biased
EFRANSAC [73]	RANSAC[16], energy minimization	Random sampling, octree	Weak to low ϕ_r, biased

candidates. O_{depth} is the depth of the octree, and $|C|$ is the number of candidates drawn so far.

6.2.4 *Drawbacks of conventional methods*

Table 6.1 lists the conventional methods, their approach for sphere detection, efficiency strategies, and their drawbacks. Conventional methods are inefficient because they depend on point-wise voting or normal vectors estimation to generate hypothetical spheres.

Methods based on the Hough transform (and Hough voting), due to point-wise exhaustive search, they resort to fix the radius of the spheres they can detect[56], or to limit the points they process[1, 7, 76]; thus, diminishing their accuracy and narrowing their applicability. To avoid doing exhaustive search, RANSAC-based methods [4, 73, 85] resort to random sampling; making them weak to the inliers ratio ϕ_r of point clouds. As noise or outliers in point clouds increase, these methods fail to obtain valid hypothetical spheres; leading to misdetections. Moreover, their accuracy highly depends on the correct estimation of point-wise normal vectors.

Furthermore, 3D LIDAR point clouds possess a long-range, and the detection of relatively small and non-invasive sphere targets [17, 86, 92] is an extremely difficult task for all the conventional methods proposed so far.

In addition, most of the conventional methods lack of public implementations, only EFRANSAC [73] is implemented in the CGAL Library [80]. Therefore, in this work, we provide comparative assessments with EFRANSAC and the proposed method.

6.3 PROPOSED METHOD

6.3.1 *Main features and superiority*

In order to solve the aforementioned drawbacks of the conventional methods, in this paper, we propose an efficient and deterministic method to detect spheres in unorganized point clouds. The proposed method has two main features; (i) it employs a 3D space subdivision called sliding voxels that generates hypothetical spheres for Hough voting

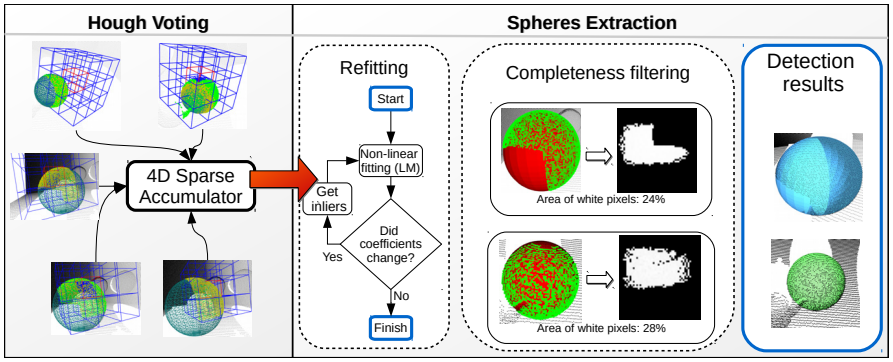


Figure 6.1: Overview of the proposed method

without discarding any point. In other words, the proposed method is capable of analyzing the whole point cloud without resorting to naive random sampling for hypothesis generation. Therefore, the sliding voxel technique contributes to achieving superior accuracy and robustness in sphere detection even in point clouds with low ϕ_r. (ii) Also, the proposed method transforms voxelized regions of the point cloud into local planes, which efficiently reduces the number of computations for Hough voting. That is, as opposed to conventional subsampling strategies prone to noise and outliers, the proposed method can achieve highly efficient Hough voting by employing sliding voxels, which contributes to reducing the entire processing time drastically without deteriorating its accuracy. Moreover, these superiorities allow us to extend the applicability of the proposed method to the case of processing a huge amount of point clouds captured by TLS in real-world situations.

6.3.2 *Process flow*

Figure 6.1 shows a graphical overview of the proposed method. We proposed an efficient octree-based point cloud subdivision to robustly estimate hypothetical spheres with our novel sphere fitting algorithm. To globally detect those spheres, we performed Hough voting with a memory-efficient accumulator based on nested tree structures. Finally, the spheres are pruned by a completeness score and refitted using the connected components of the projecting bitmap.

The proposed method starts by dividing the point cloud space in a 3D grid using an octree. Figure 6.2 shows how the 3D space of the point cloud is enclosed by an Axis Aligned Bounding Box (AABB) and subdivision occurs recursively until it complies with a voxel size V_s. The bounding box is expanded accordingly such that the leaves of the octree match the desired voxel size.

In each node of the octree, we save information about the local geometry of its points: centroid and normal vector. The normal vector was computed using PCA and corresponds to the eigenvector associated with the smallest eigenvalue. Therefore, the number of points to process becomes the number of occupied voxels with three or more points.

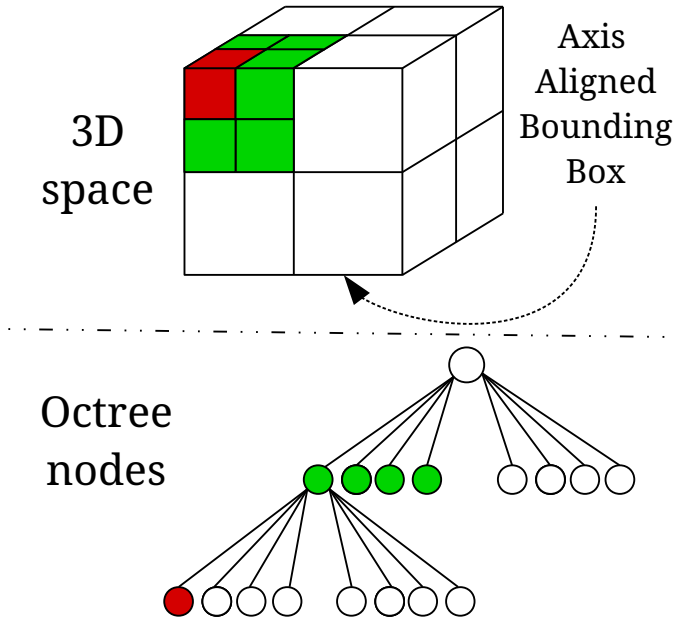


Figure 6.2: 3D space subdivision using an octree

For each leaf, we are going to select their 26-neighbors leaves, such that contiguous cells overlap and share geometrical properties, i.e., a 3D $(3 \times 3 \times 3)$ sliding window with a stride of 1. We call this structure a sliding voxel, and it helps us to robustly and efficiently identify spherical-like regions in the point cloud. Then we take advantage of this structure to generate a hypothetical sphere for every sliding voxel in a point cloud.

6.3.3 *Hypothetical spheres generation*

Several local sphere fitting methods exists, among them, algebraic fitting uses linear least squares to fit a sphere in a point set. It works by rearranging the sphere equation

$$(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2 = r^2 \tag{6.4}$$

to the following

$$2xC_x + 2yC_y + 2zC_z + \alpha = x^2 + y^2 + z^2, \tag{6.5}$$

where $\{C_x, C_y, C_z, r\}$ are the parameters of the sphere and $\alpha = r^2 - C_x^2 - C_y^2 - C_z^2$.

Then, we can obtain the sphere parameters using the least squares normal equation of its matrix representation

$$A^T Ax = A^T b. \tag{6.6}$$

As $A^T A$ is symmetric and positive-definite, it can be solved with Cholesky factorization to get the sphere parameters from x.

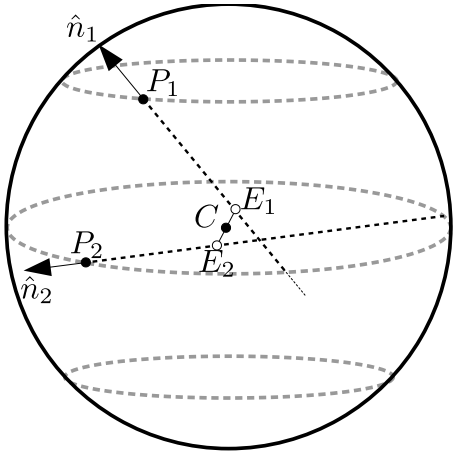


Figure 6.3: Sphere estimated with two points and their normal vectors

A second alternative is to use a non-linear least-squares approach. The Levenberg-Marquardt(LM)[50] method uses gradient-based optimization to find the sphere coefficients that minimize the error between a sphere model and a point set. In the proposed method we used the error function

$$\arg\min_{C,r} \sum_{i=1}^{n} (P_i - C)^T (P_i - C) - r^2 \tag{6.7}$$

where P_i is a point of a set of n number of points, C is the center of the sphere and r is the radius. The Jacobian J of Equation (6.8) given a point P_i is

$$J = [-2(P_{ix} - C_x), -2(P_{iy} - C_y), -2(P_{iz} - C_z), -2r]^T \tag{6.8}$$

where $\{P_{ix}, P_{iy}, P_{iz}\}$ are the point coordinates.

Since its correct convergence depends on an initial hypothesis, it is not clear which value is best for each case. When a sliding voxel falls in a planar surface we would expect to get a sphere with an arbitrarily far center and a big radius. However, both linear and non-linear least-squares approaches produce unpredictable results on planar point sets and are greatly affected by their number of outliers.

Therefore, we introduce a novel local fitting algorithm that provides a robust estimate of the best fit sphere of a point set. Given a set of N points with normal vectors Ψ of every sliding voxel, for all combinations of its items $\binom{N}{2}$, we estimate a sphere using the model generation method described by Schnabel, et al. [73] which uses two points and their normal vectors to estimate the parameters of one hypothetical sphere at a time. However, they do not indicate the specific method they used to estimate the sphere center. Therefore, I will describe the method used to generate hypothetical spheres from two oriented points.

6.3.3.1 *Sphere estimation from two oriented points*

It starts by detecting the shortest segment between two lines defined by two oriented points P_1 and P_2 using Sunday's approach[13]. Fig-

ure 6.3 illustrates how the center $C = \{C_x, C_y, C_z\}$ of the sphere can be calculated from the line segments $L_1 = \overline{P_1\hat{n_1}}$ and $L_2 = \overline{P_2\hat{n_2}}$. C is the midpoint between the shortest line segment $L^* = \overline{E_1E_2}$ between L_1 and L_2. L^* is the shortest line segment if and only if it is orthogonal to L_1 and L_2 simultaneously. To estimate the endpoints E_1 and E_2, we need to parameterize L_1 and L_2 as follows,

$$\begin{aligned} E_1(s) &= P_1 + s\hat{n}_1, \text{ where } s \in \mathbb{R}, \\ E_2(t) &= P_2 + t\hat{n}_2, \text{ where } t \in \mathbb{R}. \end{aligned} \tag{6.9}$$

Then, we need to estimate s^* and t^* such that the resulting line segment

$$\begin{aligned} L^*(s^*, t^*) &= E_1(s^*) - E_2(t^*) \\ L^*(s^*, t^*) &= P_1 - P_2 + s^*\hat{n}_1 - t^*\hat{n}_2 \end{aligned} \tag{6.10}$$

is parallel to both L_1 and L_2.

The angle between L^* and $\{L_1, L_2\}$ can be derived from the equations

$$\begin{aligned} \hat{n}_1 \cdot L^*(s^*, t^*) &= \|\hat{n}_1\|_2 \, \|L^*(s^*, t^*)\|_2 \cos\theta \text{ and} \\ \hat{n}_2 \cdot L^*(s^*, t^*) &= \|\hat{n}_2\|_2 \, \|L^*(s^*, t^*)\|_2 \cos\phi. \end{aligned} \tag{6.11}$$

Since we need to fix $\theta = \phi = 90[deg]$, then substituting $\cos(90[deg]) = 0$ in Equation (6.11) results in the following equations

$$\hat{n}_1 \cdot L^*(s^*, t^*) = 0, \tag{6.12}$$

$$\hat{n}_2 \cdot L^*(s^*, t^*) = 0. \tag{6.13}$$

In other words, given the parameterized points $E_1(s)$ and $E_2(t)$ from Equation (6.9), we need to find s and t such that the conditions of Equation (6.12) and Equation (6.13) apply simultaneously and will name them s^* and t^*.

By substituting Equation (6.9) in Equation (6.12) and Equation (6.13), we get

$$s(\hat{n}_1 \cdot \hat{n}_1) - t(\hat{n}_1 \cdot \hat{n}_2) = -\hat{n}_1(P_1 - P_2) \text{ and} \tag{6.14}$$

$$s(\hat{n}_1 \cdot \hat{n}_2) - t(\hat{n}_2 \cdot \hat{n}_2) = -\hat{n}_2(P_1 - P_2). \tag{6.15}$$

For the sake of simplicity we are going to apply the following substitutions to Equation (6.14) and Equation (6.15)

$$\begin{aligned} a &= \hat{n}_1 \cdot \hat{n}_1 \\ b &= \hat{n}_1 \cdot \hat{n}_2 \\ c &= \hat{n}_2 \cdot \hat{n}_2 \\ d &= \hat{n}_1(P_1 - P_2) \\ e &= \hat{n}_2(P_1 - P_2). \end{aligned} \tag{6.16}$$

Then Equation (6.14) and Equation (6.15) become

$$sa - tb = -d \tag{6.17}$$

$$sb - tc = -e. \tag{6.18}$$

To solve s and t we apply Cramer's rule and obtain

$$s = \frac{cd - be}{b^2 - ac} \tag{6.19}$$

$$t = \frac{db - ae}{b^2 - ac}. \tag{6.20}$$

Noticeably, if $\hat{n}_1$ and $\hat{n}_2$ are nearly parallel, the denominator $b^2 - ac$ tends to 0, and the solution is undefined. In this case, we ignore it if $b^2 - ac$ is close to the machine epsilon.

After we estimate appropriate values of s and t, the center of the sphere C is

$$C = \frac{1}{2}(E_1 + E_2), \tag{6.21}$$

and its radius

$$r = \frac{1}{2}(\|P_1 - C\|_2 + \|P_2 - C\|_2). \tag{6.22}$$

6.3.3.2 *Robust sphere estimation from sliding voxel*

By doing this for every pair of points and normals inside each sliding voxel, we are obtaining a set of all possible spheres $S \in \mathbb{R}^4$. As the sliding voxel has a maximum of 27 occupied voxels, the maximum cardinality of S is 351. Ideally, if the surface is perfectly spherical, all the spheres estimated will be the same, and S would have 0-variance in all its four dimensions. If we face with noise or outliers, its variance will fluctuate and so its mean and median.

To robustly get the most probable sphere, we select the median sphere $S_m \in S$ such that

$$\arg\min_i \|S_i - S_\mu\|_2, \;\; i \in [1, |S|] \tag{6.23}$$

where $S_i \in S$ and $|S|$ is the cardinality of the set S and S_μ is the mean sphere in S.

Given the mean sphere S_μ and S_m, we also calculate the spherical likelihood

$$\delta\mu = \|S_m - S_\mu\|_2 \tag{6.24}$$

where $\|\cdot\|_2$ denotes the Euclidean norm. If the sliding voxel falls into a perfect sphere, $\delta\mu$ will tend to 0 and will represent the deviation of the median with respect to the mean of the set of all possible spheres. Therefore, $\delta\mu$ is thresholded with the parameter $Th_{\delta\mu}$ to avoid regions that do not possess a spherical geometry.

Figure 6.4 shows how a portion of the point cloud in the sliding voxel can define the underlying spherical geometry by using our fitting method.

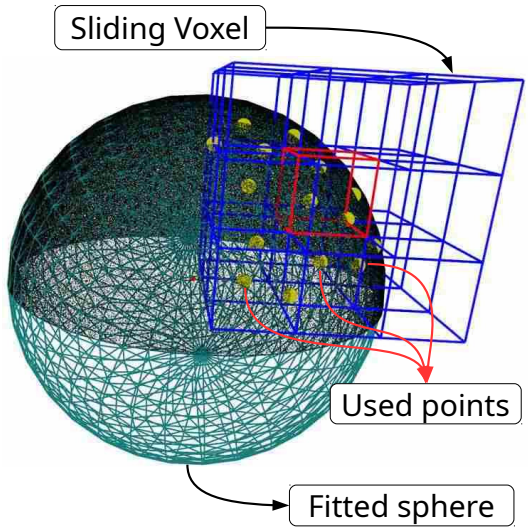


Figure 6.4: Sphere fitting with sliding voxels

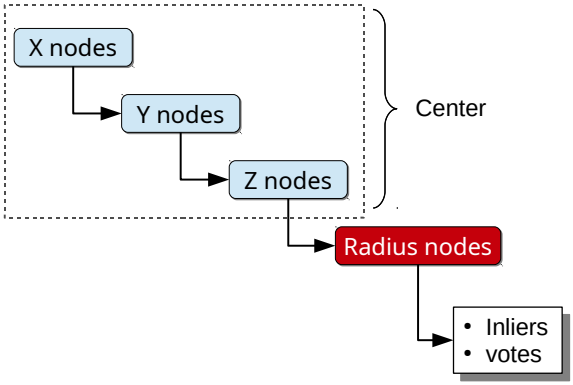


Figure 6.5: Spheres accumulator structure

6.3.4 *Hypothesis verification*

Once all our hypothetical spheres are estimated, we make them converge into a Hough accumulator since the spheres accumulator has four dimensions, we chose a memory-efficient nested tree structure shown in Figure 6.5. It is common in the literature to find the accumulator discretization defined by the number of bins of each dimension. However, since all sphere dimensions are expressed using the same metric, we can parameterize it using the accumulator bin size Acc_{res} instead of the bins number.

After the voting finished, the most prominent spheres are extracted by accumulating the votes and performing peak detection in a $3 \times 3 \times 3 \times 3$ sliding window over the accumulator. A sphere will only be extracted if it has a minimum number of votes V_{min}.

As a result of the discretization, noise and outliers, the extracted spheres are not the final spheres but an approximation of what they

should be. Therefore, we have to refit and prune them, as shown in Figure 6.1. Voting took place with sliding voxel centroids, but refitting will take place with the actual points. Therefore, we start to search for inliers and refit iteratively until the sphere coefficients changes are negligible.

To avoid the negative influence of outliers, we map the inlier points of the refitting into a plane

$$\begin{aligned} x &= \frac{1}{\pi} \arccos\left(\frac{P_z}{\|P - C\|_2}\right) \\ y &= \frac{1}{2\pi} \text{atan2}(P_y, P_x) + \frac{1}{2} \end{aligned} \tag{6.25}$$

such that it results in a squared points distribution. Then a bitmap is generated by deciding the number of the discretization bins B_{bins}. From this bitmap, we perform two things:

- To select the biggest cluster using 8-neighbors clustering.
- To estimate a completeness measure.

After getting the biggest cluster, completeness K is measured by the ratio

$$K = \frac{S_O}{S_T} \tag{6.26}$$

where S_O is the number of occupied pixels, and S_T is the total number of pixels. Although there is no mapping between a sphere and a plane without deformations, we find this ratio to be a close approximation of the sphere completeness. Particularly in the visible quadrants, deformations are negligible.

This process of refitting and pruning is applied for every sphere extracted from the accumulator. After all the extracted spheres are processed, the algorithm finishes.

6.4 EXPERIMENTS AND DISCUSSION

6.4.1 *Datasets*

We evaluated EFRANSAC and the proposed method with experiments against synthetic and real data.

The synthetic dataset $\mathcal{M}$, shown in Figure 6.6, was generated using sensor simulation in Blensor 1.0.18-RC10[23] and a model with six spheres, among other objects recreating realistic scenarios of occlusion and loss of the spheres' surface. Although the simulation provides organized point clouds, we treat them as unorganized. In the lower part, we can observe a render of the model used to simulate a realistic time-of-flight (TOF) sensor point clouds. The three spheres on the back have the same radius, but we varied their levels of potential visibility from the sensor: 100%, 50% and 25%. Furthermore, as we are aiming

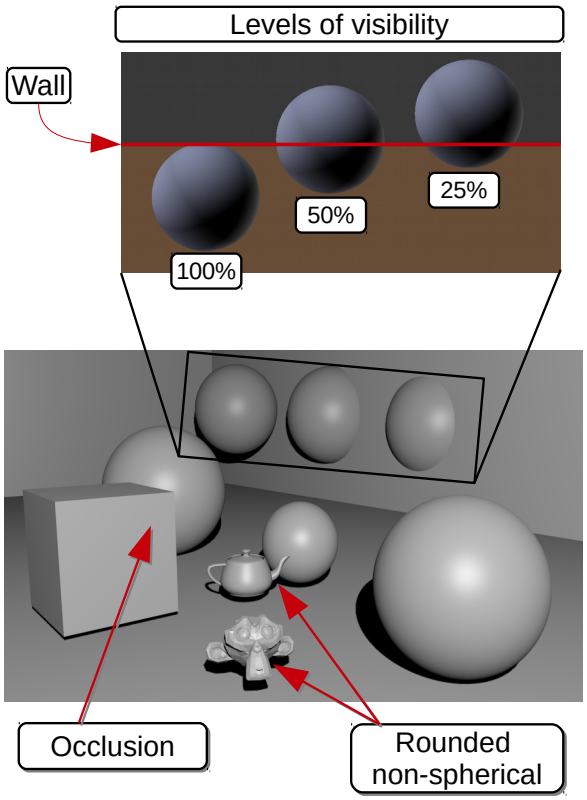


Figure 6.6: Synthetic dataset $\mathcal{M}$

to simulate a real sensor as close as possible, we set the parameters of the sensor simulation software to match those of the KinectV2.

The simulations occurred at the same sensor position but with variations in the level of noise, with a 0-mean Gaussian noise, and its variance σ set to $[0.004, 0.008, 0.012, \ldots, 0.04]$. These variations are used to modify the coordinates of the sensed points. Since it is a KinectV2 simulation its number of points is 217,088, its Bounding Box Diagonal Distance (BBDD) is around 9.00[m] and its resolution is about 0.014[m]. Where BBDD is the vector length of the extreme points of the bounding box.

Figure 6.7 shows a synthetic point cloud of $\mathcal{M}$ that corresponds to the lowest level of Gaussian noise ($\sigma = 0.004$). The ground truth spheres are shown in green as 3D models, and their index is shown in the numbers displayed above.

The NDAJ dataset $\mathcal{N}$ consists of 34 point clouds obtained from a FARO® 3D LiDAR scanner, they are highly dense and cover a broad area.

The scanning target is a building of the Engineering campus of the National Defense Academy of Japan (NDAJ). Figure 6.8 shows a render of the registration of all the point clouds of the dataset. This render was generated by processing all the registered point clouds with a voxel size of 0.05[m] and applying normals estimation with a radius of 0.075[m], then we activated the EDL shader from the qEDL plugin in Cloud Compare.

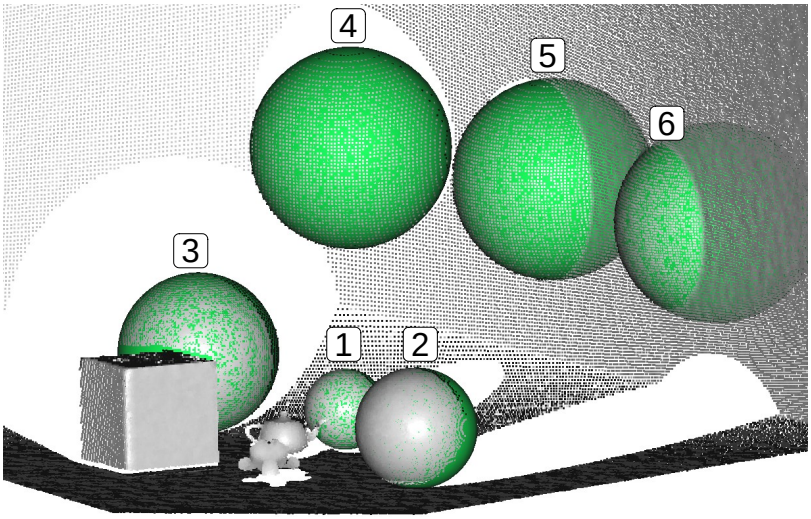


Figure 6.7: Point cloud example from the $\mathcal{M}$ dataset with ground truth spheres

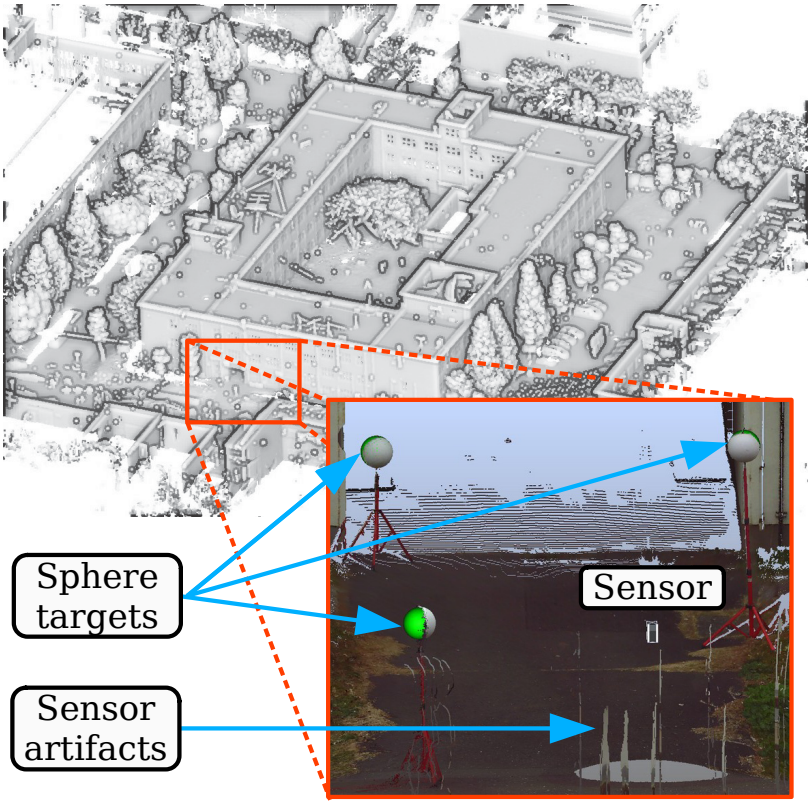


Figure 6.8: Rendering of all subsets of the $\mathcal{N}$ dataset

Figure 6.8 also shows a close up of a point cloud of $\mathcal{N}$ (G_2) and its location inside the rendering. Three sphere targets that were physically placed near the sensor are shown alongside its ground truth spheres shown in green. These are part of the ground truth spheres set. Additionally, sensor artifacts produced by moving objects in the scene are shown.

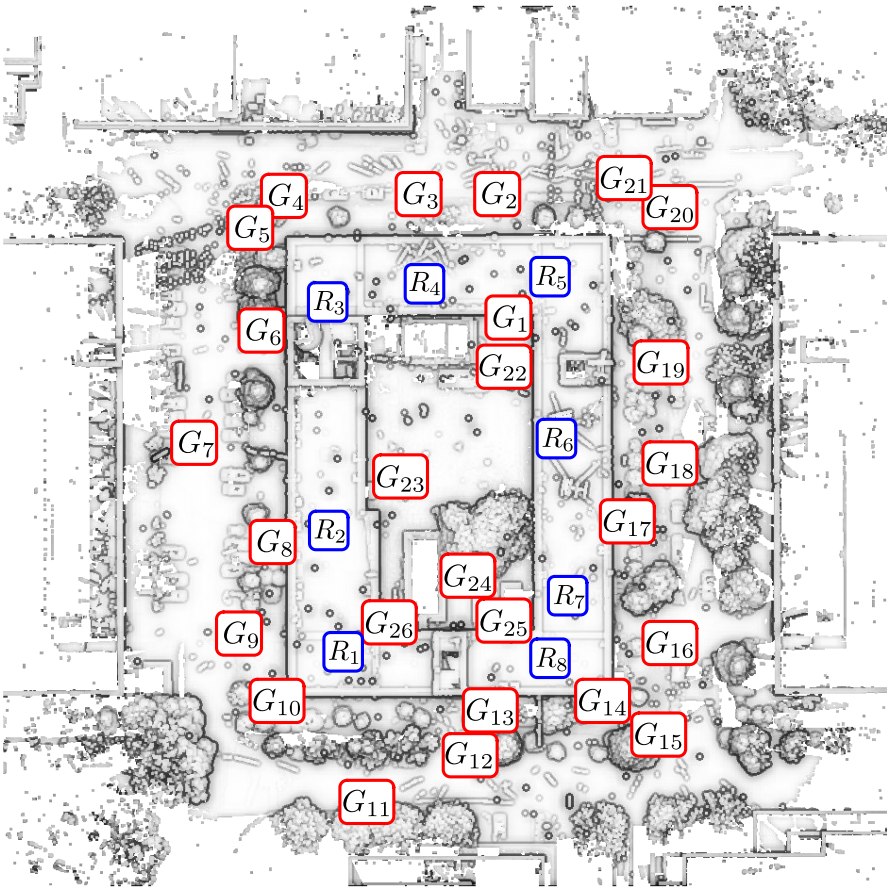


Figure 6.9: Sensor locations of the $\mathcal{N}$ dataset

The ground truth spheres were generated by using the scanner's software in a user-guided process. The ground truth data spheres of the real dataset are highly precise since it was used to register the point clouds by manually labeling the spheres in the FARO Scene® point cloud processing software.

The location of each categorized scan can be seen in Figure 6.9. 26 of the point clouds were taken from ground level, numbered from G_1 to G_{26}, and marked with red boxes in Figure 6.9. Eight point clouds were taken from the roof of the building, numbered from R_1 to R_8, and marked with blue boxes in Figure 6.9.

The ground truth spheres of $\mathcal{N}$ have fixed positions, but depending on the distance from the sensor, the density of their points and their noise can vary enormously. Figure 6.10 shows the effect of a sphere sensed from 6.15[m] and 31[m], there is a big difference in the density of their points when they are near or far from the sensor. Therefore, a challenging feature of these long-range point clouds, is their variations in density, which makes it difficult to validate hypothetical spheres by only counting their inliers.

In Table 6.2, we summarize statistical information of both datasets. The synthetic dataset consists of 10 point clouds that resemble the ones gathered from low-cost sensors. The table shows the data from the

(a) Nearest sphere: 6.15[m], 4,958 points

(b) Farthest sphere: 31[m], 169 points

Figure 6.10: Example of sensed spheres and their respective ground truth 3D models of the R_4 (LIDAR) point cloud

Table 6.2: Datasets information

Dataset	**Points[#]**	**BBDD[m]**	**CR[cm]**
$\mathcal{M}_L$	217,088	8.77	1.00
$\mathcal{M}_M$	217,088	8.88	1.24
$\mathcal{M}_H$	217,088	8.99	1.47
$\mathcal{N}_G$	25,543,762.73 ±3,625,816.05	299.70±60.27	0.44±0.07
$\mathcal{N}_R$	18,261,275.62 ±2,615,343.03	344.15±18.81	0.30±0.04

synthetic clouds with low ($\mathcal{M}_L$), mid ($\mathcal{M}_M$), and high ($\mathcal{M}_H$) levels of Gaussian noise when $\mu = 0$ and σ become [0.004, 0.02, 0.04], respectively. Also, we divided the real point clouds $\mathcal{N}$ dataset into two parts, ground ($\mathcal{N}_G$) and roof ($\mathcal{N}_R$), which separates the point clouds that were scanned at the ground level and over the building. CR stands for *Cloud Resolution*, which is a measure of the average density of the point cloud and is defined as the mean distance between each point and its nearest neighbor.

6.4.2 *Experimental setup*

To evaluate the accuracy, we performed matching of the ground truth spheres S_{gt} and the detected spheres S_{det} coefficients. A match occurs if a sphere in S_{det} is found in S_{gt} by thresholding its Euclidean distance in $\mathbb{R}^4$,

$$\left| S_{det_i} - S_{gt_j} \right| < 0.1[\text{m}]. \tag{6.27}$$

This threshold provides an average maximum error tolerance of 0.025[m] for the four parameters of a sphere model. To decide this threshold, we ran EFRANSAC on the noisiest point cloud of $\mathcal{M}$ and adjusted it such that a visible bias coming from noise, and affecting the position and radius of spheres do not alter the true positives count TP when most of its inliers are close to the sphere (shown later in visual assessment).

Table 6.3: EFRANSAC parameters

Name	ϵ	C_ϵ	N_{th}	Min. support	z	Normals radius
$\mathcal{M}_L$	0.016	0.80	0.95	700	0.00010	0.04
$\mathcal{M}_M$	0.020	0.80	0.95	1100	0.00005	0.14
$\mathcal{M}_H$	0.040	0.80	0.95	1100	0.00005	0.20
$\mathcal{N}_G$	0.020	0.02	0.10	200	0.00005	0.05
$\mathcal{N}_R$	0.020	0.02	0.10	200	0.00005	0.05

Table 6.4: Proposed method parameters

Name	V_s	V_{min}	Acc_{res}	ϵ	$Th_{\delta\mu}$	K	B_{bins}
$\mathcal{M}_L$	0.16	20	0.05	0.020	0.4	0.1	50
$\mathcal{M}_M$	0.16	20	0.05	0.020	0.4	0.1	50
$\mathcal{M}_H$	0.18	30	0.05	0.020	0.4	0.1	50
$\mathcal{N}_G$	0.05	50	0.01	0.002	0.3	0.1	25
$\mathcal{N}_R$	0.05	50	0.01	0.002	0.3	0.1	25

All the detected spheres that satisfy Equation (6.27) are TP, then we can define the following metrics from information retrieval[48] precision γ, recall ζ, and F_1 score η,

$$\gamma = \frac{TP}{\#S_{det}} \tag{6.28}$$

$$\zeta = \frac{TP}{\#S_{gt}} \tag{6.29}$$

$$\eta = \frac{2TP}{\#S_{det} + \#S_{gt}} \tag{6.30}$$

where # denotes set cardinality. Precision will max at 1 when we detected only correct spheres, while recall will max at 1 when all the ground truth spheres were detected. η is the harmonic mean between γ and ζ.

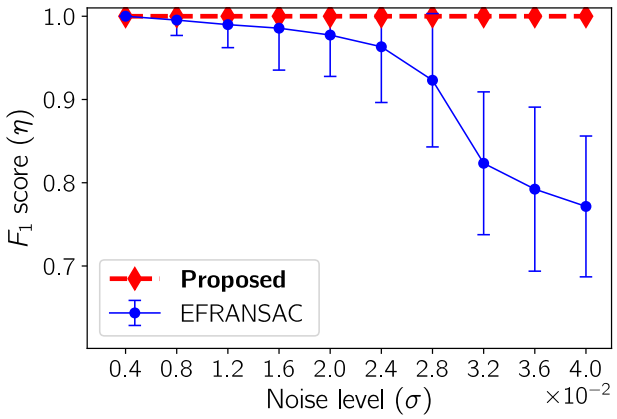
Both algorithms implementations are in C++, compiled with gcc 7.5 and O3 optimizations. We used the EFRANSAC implementation from the CGAL library 4.11[80], while the proposed method was implemented mostly using routines from the PCL Library 1.9.1[67]. Since EFRANSAC is nondeterministic, we executed the experiments 50 times and measured both mean and standard deviation.

We adjusted the parameters of the evaluated methods based on preliminary experiments to get the most accurate results, following to adjustments in improving processing time without deteriorating their accuracy. Table 6.3 and Table 6.4 show EFRANSAC and the proposed method parameters, respectively.

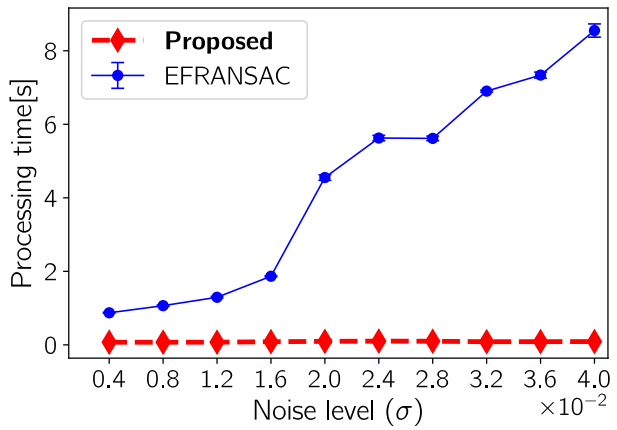
6.4.3 *Experiments results and discussion*

For each dataset, we provide results of both visual and numerical assessment. For the synthetic datasets, we calculated η for each level of noise when using EFRANSAC and the proposed method. Figure 6.11(a) shows the accuracy of the evaluated methods. A value of one means the proposed method detected all the ground truth spheres without false positives, η goes down when either false positives or false negatives drop. The EFRANSAC accuracy drops as the noise worsen, even when we increased its iterations and the radius of normal vectors. On the other hand, the proposed method detected successfully all the spheres with high accuracy. Figure 6.11(b) is showing that the proposed method is drastically more efficient than EFRANSAC, which is requiring more processing power as noise increases due to the necessity of expanding the support radius of normal vectors estimation and its iterations.

Table 6.5 shows the results of per-ground truth sphere in the dataset $\mathcal{M}$. The index of the sphere is shown in the left-most column. The two following columns show their mean error with respect to their closest



(a) η experiment results



(b) Processing time[s] experiment results

Figure 6.11: η and processing time experiment results of the $\mathcal{M}$ dataset

match of detected spheres in $\mathbb{R}^4$ by matching the ground truth spheres with the one that minimizes the left term of Equation (6.27). Also, we highlighted in bold the ones that showed the best results.

EFRANSAC has a slightly better detection error than the proposed method only when its detection rate is competitive. Hence, in this step, EFRANSAC is filtering points that do not agree with the normal vector of the hypothetical model. However, the heavy computational costs associated with computing point-wise normal vectors (see Fig. (b)) are not worth a few millimeters of better precision.

On the other hand, in the detection rate of Table 6.5, we can notice that EFRANSAC fails to detect the sphere with the least visibility. This happens because the normal vectors near the border of the plane wall and the sphere are imprecise. Hence, further reducing the available surface to detect the sphere number six.

Figure 6.12 shows the η results of the evaluated methods for each of the point cloud groups of the $\mathcal{N}$ dataset. The y-axis of the plot is divided into two because of the poor results of EFRANSAC since it had drastically lower precision and negatively affected its η. EFRANSAC validates a model by thresholding its inliers count without considering their geometry and completeness. Therefore, the inliers ratio ϕ_r of those point clouds is quite low, and near the machine epsilon, it is obvious

Table 6.5: Detailed results per ground truth sphere of experiments with synthetic data of $\mathcal{M}$.

Sphere	Mean error in $\mathbb{R}^4$[m] (EFRANSAC)	Mean error in $\mathbb{R}^4$[m] (proposed)	Detection rate[%] (EFRANSAC)	Detection rate[%] (proposed)	Note
1	0.1105	**0.0321**	89.8	**100.0**	Small radius (0.2[m])
2	**0.0196**	0.0233	**100.0**	**100.0**	Nearest
3	**0.0155**	0.0173	**100.0**	**100.0**	Occluded
4	0.0219	**0.0138**	99.8	**100.0**	100% Visibility
5	0.0389	**0.0172**	98.0	**100.0**	50% Visibility
6	0.0886	**0.0292**	73.2	**100.0**	25% Visibility

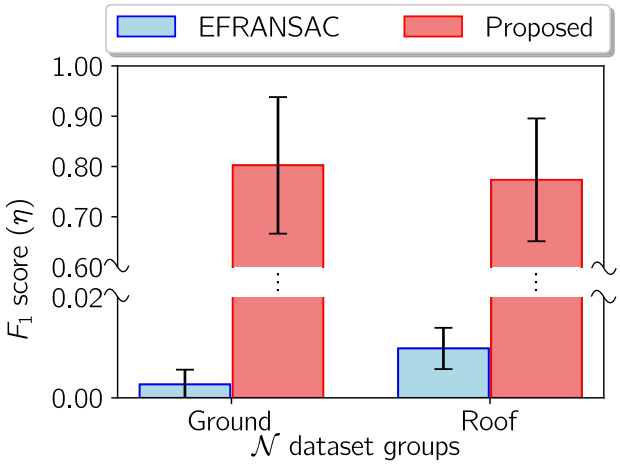


Figure 6.12: η results using the $\mathcal{N}$ dataset, higher is better

that a method that relies only on random sampling like EFRANSAC is failing very badly in the results.

Moreover, a sphere of the dataset usually contains a few hundred of points while the whole point cloud accounts for dozens of millions. This situation forces EFRANSAC to set a lower value of inliers count that supports a sphere candidate, thus detecting numerous false positives and diminishing its precision. The standard approach to address this issue is to increase RANSAC iterations by modifying its probability parameter described in Equation (6.2). However, this value becomes dangerously near the machine epsilon for single float precision and thus numerically unstable. Furthermore, its performance decreases in several orders of magnitude.

On the other hand, the proposed method opts for a smarter strategy by robustly measuring the likelihood of regions to be spherical, converging hypothetical spheres from highly spherical regions into an accumulator and filtering by their completeness. It demonstrated superior accuracy even in massive point clouds with very low ϕ_r. We roughly estimated that this ratio for the $\mathcal{N}$ dataset is less than 0.00005.

Figure 6.13 illustrates the processing time of the evaluation methods for each point cloud of the $\mathcal{N}$ dataset. The y-axis is divided into two sections with different scales to visualize the processing time results. For better visualization, and unlike the experiments with synthetic data, we are not including the normals vectors estimation time in these results. If we included them, it would be extremely difficult to visualize and compare the high efficiency of the proposed method, since it is astronomically better than EFRANSAC.

Figure 6.14, Figure 6.15, and Figure 6.16 show the detailed results of EFRANSAC and the proposed method against the $\mathcal{N}$ dataset. Figure 6.14 shows the proposed method has superior accuracy due to the low precision of EFRANSAC shown in Figure 6.15 because EFRANSAC detected numerous false positives in the $\mathcal{N}$ dataset. Figure 6.16 is showing how many of the ground truth spheres were detected without consider-

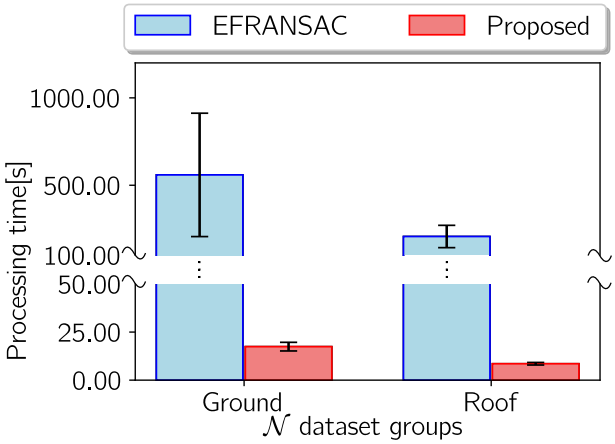


Figure 6.13: Processing time[s] evaluation results using the $\mathcal{N}$ dataset, lower is better

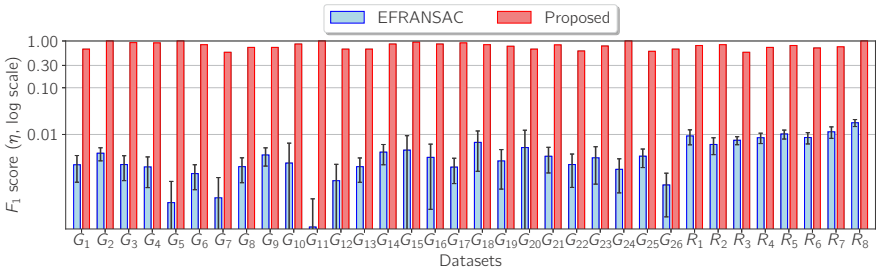


Figure 6.14: F_1 score (η) results using the $\mathcal{N}$ dataset, higher is better

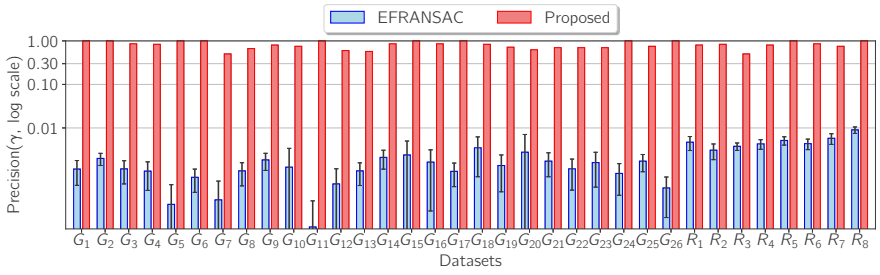


Figure 6.15: Precision (γ) results using the $\mathcal{N}$ dataset, higher is better

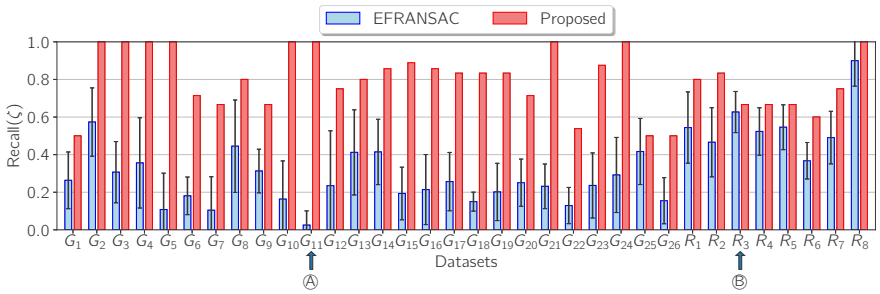


Figure 6.16: Recall (ζ) results using the $\mathcal{N}$ dataset, higher is better

ing false positives. The proposed method showed a superior recall in all scenarios. Noticeably, at the mark Ⓐ in Figure 6.16, EFRANSAC recall was extremely low, and the proposed method recall was remarkably high. EFRANSAC failed because the ground truth spheres of the point cloud G_{11} are relatively far from the sensor and has a lower density, all the spherical points account for 0.0024% of the points, and a ground truth sphere was partially occluded.

At the mark Ⓑ in Figure 6.16, EFRANSAC recall was very close to the proposed method. Figure 6.17(a) shows the R_3 point cloud from $\mathcal{N}$. The green triangle represents the sensor location, and the blue circles the ground truth spheres. One of the ground truth spheres is extremely far from the sensor, its distance is represented by the red dashed line and is about 38.33[m]. Both methods failed to detect this sphere since it is barely recognizable due to its extremely low density.

We also show a visual assessment of the experiment results. In Figure 6.18, we can observe the results of executing the evaluated methods on the synthetic point cloud with the least ($\sigma = 0.004$) and most noise

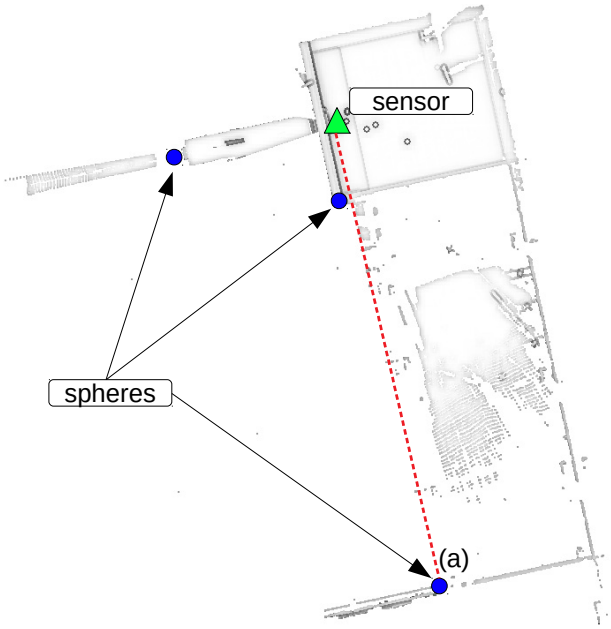


Figure 6.17: R_3 point cloud from the $\mathcal{N}$ dataset

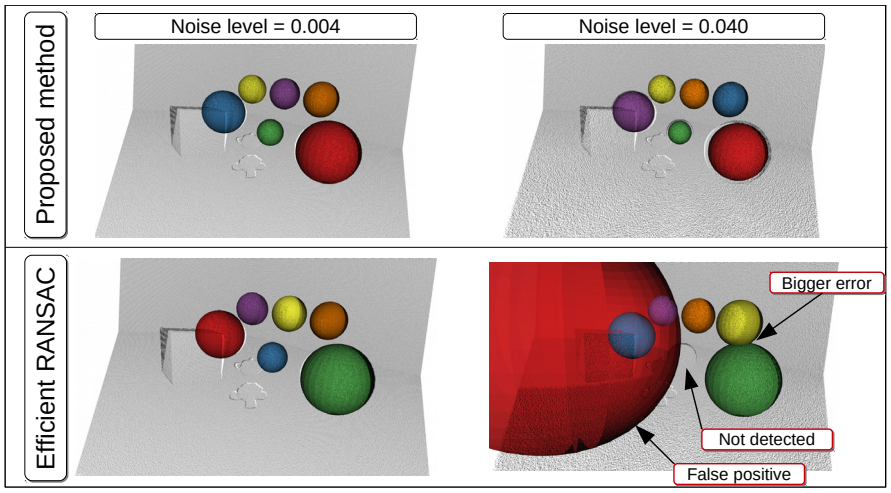
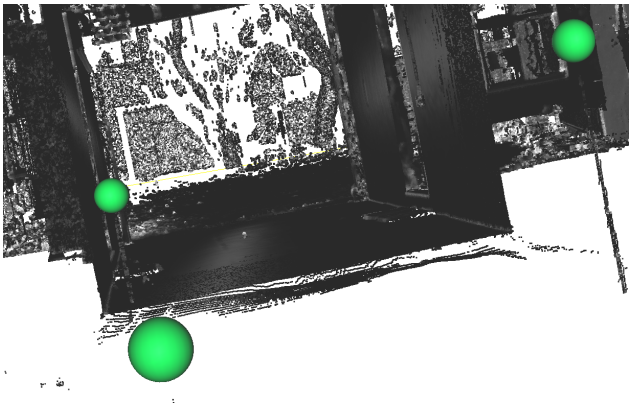


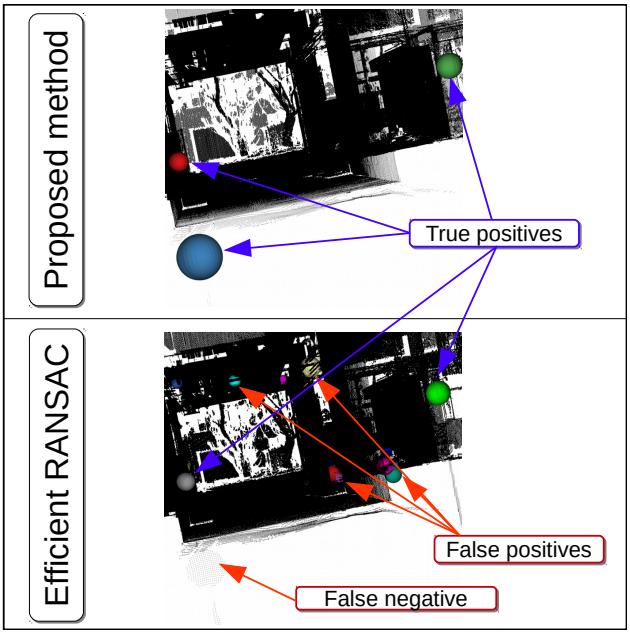
Figure 6.18: Graphical comparison: $\mathcal{M}$ dataset

($\sigma = 0.04$). We assigned a different color to each detected sphere in the order they are given by the evaluated methods. The proposed method detected the spheres flawlessly with high accuracy. EFRANSAC results are very good with low noise scenarios, but as we test for higher levels of noise the number of misdetections and false positives arise. Moreover, as we need to increase thresholds and the support radius for normals estimation, the smallest spheres tend to be undetected, and bigger errors in the spheres coefficients become visible.

Figure 6.19 shows the ground truth and results of spheres detection of the point cloud G_1. Three ground truth spheres are not visible in Figure 6.19(a), which were placed in the central part of the building with a radius of 0.0381[m], neither the proposed method nor EFRANSAC



(a) Ground truth spheres render of G_1



(b) Detection results

Figure 6.19: Graphical comparison: $\mathcal{N}$ dataset (G_1)

were able to detect. Figure 6.19(a) shows the detected spheres of both the proposed method and EFRANSAC. Due to the numerous false positives and for better visualization, we omitted EFRANSAC spheres larger than 0.3[m]. Even though the extra filtering we can notice EFRANSAC failed to detect one target sphere and detected several false positives in the background.

6.5 CONCLUSIONS AND FUTURE WORKS

Sphere detection is a core technique in point cloud processing with applications in computer vision, reconstruction, modeling, among others. However, existing algorithms work with various drawbacks such as fixed radius, low efficiency, and poor accuracy in noisy data with a numerous

amount of outliers. To solve these problems, in this paper, we proposed a new sphere detection method based on sliding voxels and Hough voting.

Through experiments we found that the proposed method achieved 50 times faster processing time and 1.08 times more accurate than EFRANSAC for the synthetic $\mathcal{M}$. In the real 3D LIDAR dataset $\mathcal{N}$, the proposed method achieved 31 times faster processing time and 183 times more accurate (F-score) than EFRANSAC without including normals estimation processing time. This is due to the proposed method can analyze all the points by employing sliding voxels even for big point clouds with numerous amount of points.

As our future works, we should investigate adaptive settings of the voxel size for various density levels in massive point clouds to further improve its accuracy. Also, we need more comprehensive experiments to analyze the relationship among processing time, total number of points and size of the spheres to give us better insight on the performance and scalability of the sphere detection methods. Furthermore, as promising extensions of this work for real-world applications, we are considering to detect nearly spherical shapes, such as human heads, bone junction, and so on.

7 CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

In this thesis, the problem of geometric primitive fitting in point clouds is investigated. Conventional methods face many problems: high computational complexity, and low accuracy. This is due to the numerous noise and artifact patterns of point clouds, and their inliers/outliers ratio of high-range clouds. Therefore, methods for the detection of planes and spheres are proposed.

PLANE DETECTION

Plane detection is a highly demanded task for many applications. However, conventional methods fail extensively while trying to detect multiple planes in unorganized point clouds. Their computational complexity becomes a problem when plane detection is a part of the pipeline of a more complex task. This lack of efficiency and robustness comes from the way conventional algorithms were designed. Methods that work in other types of data such as images or correspondences were adapted to point clouds. In this thesis, we considered from the beginning the nature and main characteristics of point clouds; sensor artifacts, noise, and variable density. Therefore, drastically more efficient and robust methods were proposed for plane detection.

- **Fast and Deterministic Hough Transform with SDoN Filtering (FDHT).**
  An efficient scheme for plane detection using Hough voting was proposed. Just after the estimation of tangent planes for each voxel, this method pre-filters the voxels that contain hypothetical planes with higher quality. Hough voting using a sparse memory model for the accumulator allowed to detect planes with finer accuracy without running out of memory. Therefore, achieving excellent performance when compared with conventional methods. However, it still was detecting spurious planes and failed to detect some planar structures.
- **Sliding Voxels for Plane Detection.**
  To increase precision and maintain efficiency a method based on sliding voxels was proposed. This paradigm does not limit the search for planes for single voxels, it exploits the overlapping nature of the sliding voxels to generate hypothetical planes with exceptionally high precision. With the sliding voxels, we can reduce noise, the number of points, and enrich a point cloud with normal vectors to robustly detect planes. The proposed sliding voxel planes detector outperforms the

state-of-the-art plane detectors as it is several orders of magnitude faster, more robust, and deterministic.

SPHERE DETECTION

- **Sliding Voxels for Planes Detection.**
  Sphere detection is another important task in point cloud processing. Spheres have geometric features that turn them into excellent targets for point cloud registration among other applications. However, its detection is unfeasible using conventional Hough transform techniques. Moreover, RANSAC and its variations, fail extremely in massive point clouds that come from high-range sensors that are commonly used for Terrestrial Laser Scanning. Therefore, in this thesis, the sliding voxel paradigm was extended for sphere detection and demonstrated superior efficiency and precision both in synthetic and real 3D LIDAR point clouds. The key of its efficiency is the increasingly reduced number of computations needed as we analyze the points distribution of each sliding voxel for spherical features. Its superior robustness is due to the local sphere fitting approach that generates robust hypothetical spheres, and its filtering by completeness.

I can say that the objectives of improving both efficiency and accuracy of geometric primitive detection were accomplished in this work. Although there are future challenges that derive from the results of this work and I consider are very important to address.

7.2 FUTURE CHALLENGES

MULTI-RESOLUTION, ADAPTIVE-RESOLUTION SLIDING VOXELS
Until now, the sliding voxel paradigm worked exceptionally good for point clouds. However, further improvements in accuracy can be made by analyzing adaptively regions defined not only by the sliding voxel, but from the path upwards the octree. This opens a door for adaptive-resolution or multiple-resolution approaches to detect shapes of any size and not constrained by their voxel size. Moreover, this can lead us to the estimation of the best threshold and voxel size depending on the points distribution of each sliding voxel.

MORE GEOMETRIC SHAPES The proposed sliding voxel paradigm was applied to both sphere and plane detection, which are basic geometric primitives with numerous applications. A generalization that comes directly from the sphere detection approach can be used for other parametric shapes such as cylinders, ellipsoids, and torus. Thus, its applications in the approximation of complex scenes into geometric shapes and compression can be achieved very efficiently and with increased robustness.

ARBITRARY SHAPE DETECTION The detection of arbitrary shapes is a complex but demanding application of point clouds. Given a 3D model of the detection target, we must find all its instances inside a point cloud. Sliding voxels can be used to efficiently match surfaces between the target model and parts of the point cloud.

BIBLIOGRAPHY

[1] Anas Abuzaina, Mark S Nixon, and John N Carter. "Sphere detection in kinect point clouds via the 3d hough transform." In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2013, pp. 290–297.

[2] Gerardo Atanacio-Jiménez, José-Joel González-Barbosa, Juan B Hurtado-Ramos, Francisco J Ornelas-Rodríguez, Hugo Jiménez-Hernández, Teresa García-Ramirez, and Ricardo González-Barbosa. "Lidar velodyne hdl-64e calibration using pattern planes." In: *International Journal of Advanced Robotic Systems* 8.5 (2011), p. 59.

[3] Dana H Ballard. "Generalizing the Hough transform to detect arbitrary shapes." In: *Readings in computer vision*. Elsevier, 1987, pp. 714–725.

[4] Tolga Birdal, Benjamin Busam, Nassir Navab, Slobodan Ilic, and Peter Sturm. "A minimalist approach to type-agnostic detection of quadrics in point clouds." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3530–3540.

[5] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. "The 3d hough transform for plane detection in point clouds: A review and a new accumulator design." In: *3D Research* 2.2 (2011), p. 3.

[6] Thomas Butkiewicz. "Low-cost coastal mapping using Kinect v2 time-of-flight cameras." In: *2014 Oceans - St. John's, OCEANS 2014* (Jan. 2015). DOI: 10.1109/OCEANS.2014.7003084.

[7] Marco Camurri, Roberto Vezzani, and Rita Cucchiara. "3D Hough transform for sphere recognition on point clouds." In: *Machine vision and applications* 25.7 (2014), pp. 1877–1891.

[8] Jiawei Chen, Cheng Zhang, and Pingbo Tang. "Geometry-based optimized point cloud compression methodology for construction and infrastructure management." In: *Computing in Civil Engineering 2017*. 2017, pp. 377–385.

[9] Jyun-Yuan Chen, Hung-Jui Lai, and Chao-Hung Lin. "Point cloud modeling using algebraic template." In: *International Journal of Innovative Computing, Information and Control* 7.4 (2011), pp. 1521–1532.

[10] Benjamin Choo, Michael Landau, Michael DeVore, and Peter A Beling. "Statistical analysis-based error models for the microsoft kinecttm depth sensor." In: *Sensors* 14.9 (2014), pp. 17430–17450.

[11] Jacky CK Chow and Derek D Lichti. “Photogrammetric bundle adjustment with self-calibration of the PrimeSense 3D camera technology: Microsoft Kinect.” In: *IEEE Access* 1 (2013), pp. 465–474.

[12] Ondrej Chum and Jiri Matas. “Matching with PROSAC-progressive sample consensus.” In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 220–226.

[13] *Distance Between 3D Lines and Segments*. "http://geomalgorithms.com/a07-_distance.html". "[Online; accessed Jul-2020]". 2020.

[14] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. “Model globally, match locally: Efficient and robust 3D object recognition.” In: *2010 IEEE computer society conference on computer vision and pattern recognition*. Ieee. 2010, pp. 998–1005.

[15] Chen Feng, Yuichi Taguchi, and Vineet R Kamat. “Fast plane extraction in organized point clouds using agglomerative hierarchical clustering.” In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 6218–6225.

[16] Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography.” In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[17] Marek Franaszek, Geraldine S Cheok, and Christoph Witzgall. “Fast automatic registration of range images from 3D imaging systems using sphere targets.” In: *Automation in Construction* 18.3 (2009), pp. 265–274.

[18] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. “Vision meets Robotics: The KITTI Dataset.” In: *International Journal of Robotics Research (IJRR)* (2013).

[19] Marjolein van der Glas, Frans M Vos, Charl P Botha, and Albert M Vossepoel. “Determination of position and radius of ball joints.” In: *Medical Imaging 2002: Image Processing*. Vol. 4684. International Society for Optics and Photonics. 2002, pp. 1571–1577.

[20] Marjolein van der Glas, Frans M Vos, Charl P Botha, and Albert M Vossepoel. “Determination of position and radius of ball joints.” In: *Medical Imaging 2002: Image Processing*. Vol. 4684. International Society for Optics and Photonics. 2002, pp. 1571–1577.

[21] Chris Glasbey, Gerie van der Heijden, Vivian FK Toh, and Alision Gray. “Colour displays for categorical images.” In: *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur* 32.4 (2007), pp. 304–309.

[22] Gaile Gordon, Mark Billinghurst, Melanie Bell, John Woodfill, Bill Kowalik, Alex Erendi, and Janet Tilander. "The use of dense stereo range data in augmented reality." In: *Proceedings. International Symposium on Mixed and Augmented Reality*. IEEE. 2002, pp. 14–23.

[23] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. "BlenSor: Blender sensor simulation toolbox." In: *International Symposium on Visual Computing*. Springer. 2011, pp. 199–208.

[24] David S Hall. *High definition lidar system*. US Patent 7,969,558. 2011.

[25] David S Hall. *Color lidar scanner*. US Patent 8,675,181. 2014.

[26] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J Davison. "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM." In: *2014 IEEE international conference on Robotics and automation (ICRA)*. IEEE. 2014, pp. 1524–1531.

[27] T. Hayata and M. Iwakiri. "3d point cloud feature extraction with the difference of centers of gravity." In: *Proc. of IPSJ Interaction*. 2016, pp. 1085–1087.

[28] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments." In: *The International Journal of Robotics Research* 31.5 (2012), pp. 647–663.

[29] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. "Real-time plane segmentation using RGB-D cameras." In: *Robot Soccer World Cup*. Springer. 2011, pp. 306–317.

[30] Stefan Holzer, Radu Bogdan Rusu, Michael Dixon, Suat Gedikli, and Nassir Navab. "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 2684–2689.

[31] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. "Surface reconstruction from unorganized points." In: *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. 1992, pp. 71–78.

[32] Paul VC Hough. *Method and means for recognizing complex patterns*. US Patent 3,069,654. Dec. 1962.

[33] John Illingworth and Josef Kittler. "A survey of the Hough transform." In: *Computer vision, graphics, and image processing* 44.1 (1988), pp. 87–116.

[34] Ulfat Imdad, Muhammad Asif, Mirza Tahir Ahmad, Osama Sohaib, Muhammad Kashif Hanif, and Muhammad Hasanain Chaudary. "Three Dimensional Point Cloud Compression and Decompression Using Polynomials of Degree One." In: *Symmetry* 11.2 (2019), p. 209.

[35] Y. Ioannou, B. Taati, R. Harrap, and M. Greenspan. "Difference of Normals as a Multi-scale Operator in Unorganized Point Clouds." In: *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*. 2012, pp. 501–508.

[36] Chris L Jackins and Steven L Tanimoto. "Oct-trees and their use in representing three-dimensional objects." In: *Computer Graphics and Image Processing* 14.3 (1980), pp. 249–270.

[37] Charles F Jekel. "Obtaining non-linear orthotropic material models for PVC-coated polyester via inverse bubble inflation." PhD thesis. Stellenbosch: Stellenbosch University, 2016.

[38] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. "A survey of simple geometric primitives detection methods for captured 3d data." In: *Computer Graphics Forum*. Vol. 38. 1. Wiley Online Library. 2019, pp. 167–196.

[39] M Kharbat, Nabil Aouf, Antonios Tsourdos, and B White. "Sphere detection and tracking for a space capturing operation." In: *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*. IEEE. 2007, pp. 182–187.

[40] Nahum Kiryati, Yuval Eldar, and Alfred M Bruckstein. "A probabilistic Hough transform." In: *Pattern recognition* 24.4 (1991), pp. 303–316.

[41] Andreas Kolb, Erhardt Barth, Reinhard Koch, and Rasmus Larsen. "Time-of-flight cameras in computer graphics." In: *Computer Graphics Forum*. Vol. 29. 1. Wiley Online Library. 2010, pp. 141–159.

[42] Kruno Lenac, Andrej Kitanov, Robert Cupec, and Ivan Petrović. "Fast planar surface 3D SLAM using LIDAR." In: *Robotics and Autonomous Systems* 92 (2017), pp. 197–220.

[43] Marc Levoy, J Gerth, B Curless, and K Pull. "The Stanford 3D scanning repository." In: *URL http:// graphics.stanford.edu /data /3Dscanrep* (2005).

[44] Frederico A Limberger and Manuel M Oliveira. "Real-time detection of planar regions in unorganized point clouds." In: *Pattern Recognition* 48.6 (2015), pp. 2043–2053.

[45] Dahua Lin, Sanja Fidler, and Raquel Urtasun. "Holistic scene understanding for 3d object detection with rgbd cameras." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1417–1424.

[46] H Christopher Longuet-Higgins. "A computer algorithm for reconstructing a scene from two projections." In: *Nature* 293.5828 (1981), pp. 133–135.

[47] Mark Maimone, Yang Cheng, and Larry Matthies. "Two years of visual odometry on the mars exploration rovers." In: *Journal of Field Robotics* 24.3 (2007), pp. 169–186.

[48] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

[49] *Marela kitchen red&white*. "https://3dwarehouse.sketchup.com". "[Online; accessed Dec-2019]". 2019.

[50] Donald W Marquardt. "An algorithm for least-squares estimation of nonlinear parameters." In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[51] Donald Meagher. "Geometric modeling using octree encoding." In: *Computer Graphics and Image Processing* 19.2 (1982), pp. 129 –147. ISSN: 0146-664X. DOI: https://doi.org/10.1016/0146-664X(82)90104-6. URL: http://www.sciencedirect.com/science/article/pii/0146664X82901046.

[52] Vicente Morell, Sergio Orts, Miguel Cazorla, and Jose Garcia-Rodriguez. "Geometric 3D point cloud compression." In: *Pattern Recognition Letters* 50 (2014). Depth Image Analysis, pp. 55 –62. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2014.05.016. URL: http://www.sciencedirect.com/science/article/pii/S016786551400172X.

[53] Bala Muralikrishnan, Prem Rachakonda, Vincent Lee, Meghan Shilling, Daniel Sawyer, Geraldine Cheok, and Luc Cournoyer. "Relative range error evaluation of terrestrial laser scanners using a plate, a sphere, and a novel dual-sphere-plate target." In: *Measurement* 111 (2017), pp. 60–68.

[54] Helmy Mustafa, Toh Yen Pang, Thierry Perret-Ellena, and Aleksandar Subic. "Finite element analysis of user-centred bicycle helmet design." In: *ARPN Journal of Engineering and Applied Sciences* 11 (2015).

[55] Chuong V Nguyen, Shahram Izadi, and David Lovell. "Modeling kinect sensor noise for improved 3d reconstruction and tracking." In: *2012 second international conference on 3D imaging, modeling, processing, visualization & transmission*. IEEE. 2012, pp. 524–530.

[56] Tokunbo Ogundana, Charles Russell Coggrave, Richard Burguete, and Jonathan Mark Huntley. "Fast Hough transform for automated detection of spheres in three-dimensional point clouds." In: *Optical Engineering* 46.5 (2007), p. 051002.

[57] Kei Okada, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. "Plane segment finder: algorithm, implementation and applications." In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. Vol. 2. IEEE. 2001, pp. 2120–2125.

[58] D Oram. "Rectification for any epipolar geometry." In: *Proceedings of the British Machine Vision Conference*. doi:10.5244/C.15.67. BMVA Press, 2001, pp. 67.1–67.10. ISBN: 1-901725-16-2.

[59] Stefan Oßwald, Jens-Steffen Gutmann, Armin Hornung, and Maren Bennewitz. "From 3D point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids." In: *2011 11th IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2011, pp. 93–98.

[60] Panagiotis Papadakis. "The canonically posed 3D objects dataset." In: 2014.

[61] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius, Max Pfingsthorn, Sören Schwertfeger, and Jann Poppinga. "Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation." In: *Journal of Field Robotics* 27.1 (2010), pp. 52–84.

[62] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius, and Jann Poppinga. "Fast registration based on noisy planes with unknown correspondences for 3-D mapping." In: *IEEE Transactions on Robotics* 26.3 (2010), pp. 424–441.

[63] Marek Pierzchała, Philippe Giguère, and Rasmus Astrup. "Mapping forests using an unmanned ground vehicle with 3D LiDAR and graph-SLAM." In: *Computers and Electronics in Agriculture* 145 (2018), pp. 217 –225. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2017.12.034. URL: http://www.sciencedirect.com/science/article/pii/S0168169917301631.

[64] Yulia Ponomareva. "Scanners Help Keep Naval Ships in Perfect Condition: Combining 3D scanning and reverse engineering makes it possible to quickly repair or replace critically important parts." In: *Optik & Photonik* 12.4 (2017), pp. 48–49.

[65] Zoltan Pusztai and Levente Hajder. "Accurate calibration of LiDAR-camera systems using ordinary boxes." In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 394–402.

[66] Grzegorz Redlarski, Marek Krawczuk, and Aleksander Palkowski. "Application of 3D whole body scanning in research on human body surface area." In: *Book of Abstracts 3DBODY. TECH 2017 8th International Conference and Exhibition on 3D Body Scanning and Processing Technologies, Montreal, Canada*. 2017, pp. 11–12.

[67] Radu Bogdan Rusu and Steve Cousins. "3d is here: Point cloud library (pcl)." In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 1–4.

[68] Jaime SANDOVAL, Kazuma UENISHI, Munetoshi IWAKIRI, and Kiyoshi TANAKA. "Robust 3D Planes Detection under Noisy Conditions Using Scaled Difference of Normals." In: *IIEEJ Transactions on Image Electronics and Visual Computing* 5.2 (2017), pp. 60–73.

[69] Jaime SANDOVAL, Kazuma UENISHI, Munetoshi IWAKIRI, and Kiyoshi TANAKA. "Robust, Efficient and Deterministic Planes Detection in Unorganized Point Clouds Based on Sliding Voxels." In: *IIEEJ Transactions on Image Electronics and Visual Computing* 7.2 (2019), pp. 67–77.

[70] Jaime SANDOVAL, Kazuma UENISHI, Munetoshi IWAKIRI, and Kiyoshi TANAKA. "Robust Sphere Detection in Unorganized 3D Point Clouds Using an Efficient Hough Voting Scheme based on Sliding Voxels." In: *IIEEJ Transactions on Image Electronics and Visual Computing* 8.2 (2020).

[71] Hamed Sarbolandi, Damien Lefloch, and Andreas Kolb. "Kinect range sensing: Structured-light versus Time-of-Flight Kinect." In: *Computer Vision and Image Understanding* 139 (2015), pp. 1–20. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2015.05.006. URL: http://www.sciencedirect.com/science/article/pii/S1077314215001071.

[72] Ruwen Schnabel and Reinhard Klein. "Octree-based Point-Cloud Compression." In: *Spbg* 6 (2006), pp. 111–120.

[73] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. "Efficient RANSAC for point-cloud shape detection." In: *Computer graphics forum*. Vol. 26. 2. Wiley Online Library. 2007, pp. 214–226.

[74] Craig M Shakarji. "Least-squares fitting algorithms of the NIST algorithm testing system." In: *Journal of research of the National Institute of Standards and Technology* 103.6 (1998), p. 633.

[75] Jason Smith, G Petrova, and Scott Schaefer. "Progressive encoding and compression of surfaces generated from point cloud data." In: *Computers & Graphics* 36.5 (2012), pp. 341–348.

[76] Christiane Sommer, Yumin Sun, Erik Bylow, and Daniel Cremers. "PrimiTect: Fast Continuous Hough Voting for Primitive Detection." In: *arXiv preprint arXiv:2005.07457* (2020).

[77] Charles V Stewart. "Bias in robust estimation caused by discontinuities and multiple structures." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.8 (1997), pp. 818–833.

[78] Xuebin Sun, Han Ma, Yuxiang Sun, and Ming Liu. "A novel point cloud compression algorithm based on clustering." In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2132–2139.

[79] Fayez Tarsha-Kurdi, Tania Landes, and Pierre Grussenmeyer. "Hough-Transform and Extended RANSAC Algorithms for Automatic Detection of 3D Building Roof Planes from Lidar Data." In: *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*. Vol. XXXVI. Espoo, Finland, Sept. 2007, pp. 407–412. URL: https://halshs.archives-ouvertes.fr/halshs-00264843.

[80] The CGAL Project. *CGAL User and Reference Manual*. 5.0.2. CGAL Editorial Board, 2020. URL: https://doc.cgal.org/5.0.2/Manual/packages.html.

[81] Philip HS Torr and Andrew Zisserman. "MLESAC: A new robust estimator with application to estimating image geometry." In: *Computer vision and image understanding* 78.1 (2000), pp. 138–156.

[82] Alexander JB Trevor, Suat Gedikli, Radu B Rusu, and Henrik I Christensen. "Efficient organized point cloud segmentation with connected components." In: *Semantic Perception Mapping and Exploration (SPME)* (2013).

[83] Kazuma Uenishi, Munetoshi Iwakiri, and Kiyoshi Tanaka. "VKOP: 3D Virtual Keypoint Detector adapted to geometric structures and its feature descriptor." In: *The journal of the Institute of Image Electronics Engineers of Japan: visual computing, devices & communications* 46.2 ((In Japanese) 2017), pp. 283–297.

[84] Martin Vel'as, Michal Španěl, Zdeněk Materna, and Adam Herout. "Calibration of rgb camera with velodyne lidar." In: (2014).

[85] Liang Wang, Chao Shen, Fuqing Duan, and Ke Lu. "Energy-based automatic recognition of multiple spheres in three-dimensional point cloud." In: *Pattern Recognition Letters* 83 (2016), pp. 287–293.

[86] Yanmin Wang, Hongbin Shi, Yanyan Zhang, and Dongmei Zhang. "Automatic registration of laser point cloud using precisely located sphere targets." In: *Journal of applied remote sensing* 8.1 (2014), p. 083588.

[87] Oliver Wasenmüller and Didier Stricker. "Comparison of kinect v1 and v2 depth images in terms of accuracy and precision." In: *Asian Conference on Computer Vision*. Springer. 2016, pp. 34–45.

[88] Jan Weingarten and Roland Siegwart. "3D SLAM using planar segments." In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 3062–3067.

[89] Lei Xu, Erkki Oja, and Pekka Kultanen. "A new curve detection method: randomized Hough transform (RHT)." In: *Pattern recognition letters* 11.5 (1990), pp. 331–338.

[90] Tomoaki Yoshida, Keiji Nagatani, Satoshi Tadokoro, Takeshi Nishimura, and Eiji Koyanagi. "Improvements to the rescue robot quince toward future indoor surveillance missions in the Fukushima Daiichi nuclear power plant." In: *Field and service robotics*. Springer. 2014, pp. 19–32.

[91] Yizhou Yu. "Surface reconstruction from unorganized points using self-organizing neural networks." In: *Proc. of IEEE visualization*. Vol. 99. 1999, pp. 61–64.

[92] Dongho Yun, Sunghan Kim, Heeyoung Heo, and Kwang Hee Ko. "Automated registration of multi-view point clouds using sphere targets." In: *Advanced Engineering Informatics* 29.4 (2015). Collective Intelligence Modeling, Analysis, and Synthesis for Innovative Engineering Decision Making Special Issue of the 1st International

Conference on Civil and Building Engineering Informatics, pp. 930–939. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2015.09.008. URL: http://www.sciencedirect.com/science/article/pii/S1474034615001032.

[93] Yizhong Zhang, Weiwei Xu, Yiying Tong, and Kun Zhou. "Online structure analysis for real-time indoor scene reconstruction." In: *ACM Transactions on Graphics (TOG)* 34.5 (2015), pp. 1–13.