Otto-von-Guericke-University Magdeburg

Faculty of Computer Science

Institute for Intelligent Cooperating Systems

Comparison of Real-Time Plane Detection Algorithms on Intel RealSense

Bachelor Thesis

Author:

Lukas Petermann

Examiner:

Prof. Frank Ortmeier

2nd Examiner

M.Sc. Marco Filax

Supervisor:

M.Sc. Maximilian Klockmann

Magdeburg, 32.13.2042

Contents

1 Introduction

2 Background

"Summary", What is to be expected in this chapter In this chapter we will present relevant literature needed to completely understand the proposed concept of chapter 3.

2.1 Intel Realsense

https://www.intelrealsense.com/compare-depth-cameras/

3 Concept

This chapter deals with the realization of analysis. We introduce the definition of real-time and the use case with respect to this work. Plane detection algorithms are selected for comparison, as well as the metrics they will be judged upon.

3.1 Scenario / UseCase

Taking motivation from chapter 1, we focus on indoor environments during this work. This includes the buildings we encounter in our normal lives, wether its the home we live in, the office we work in or a stripped-down version of a building during construction.

3.1.1 Used Sensors

To be able to perform plane detection, we need special hardware that is able to accurately record the surroundings. Numerous different cameras suffice for this task, of course varying in different aspects. For this work, we use the Intel RealSense T256 Tracking Camera, as well as the Intel RealSense D455 RGB-D Camera. Reason for this are the compatibility of the two cameras, since the T256 can be used in combination with any depth camera from the D400 series.

In addition to the cameras, the Intel RealSense software provides a wide variety of usage. We are especially interested in detection of planes in complete environments, it is therefore necessary to be able to build a map out of the data the cameras record continuously.

Realsense-ros internally uses a SLAM(Simultaneous Mapping and Localization) algorithm called RTAB-MAP [3] for map-building. RTAB-MAP is responsible for building a coherent map from a continuous stream of data that is being recorded and published by the two cameras. It is worth noting, that the success of this work does not depend on the specific SLAM algorithm being chosen. We select RTAB-MAP because it is already included in the realsense package and its reported performance suffices for this work, especially since we dont focus on SLAM algorithms in this work.

3.2 Selection Plane Detection Algorithms

First we need to assert the comparability between the algorithms introduced in 2. We report necesary criteria to both shorten the list of algorithms, as well as verify comparability.

Type of Input

Popular representations, which the recorded environment can take the form of, can be grouped into three main categories of input:

- *unorganized* or *unstructured point cloud* (UPC)
- *organized* or *structured point cloud* (OPC)
- *(depth-) image* (D-/I)

As stated before, we focus on the detection of planar structures in the entirety of a scene, rather than just singular segments thereof. In addition, only the first type of input offers a persistent view on the recorded environment.
For that reason, we disregard all algorithms which do not expect an unorganized point cloud as input.

Detected Plane Format

Which specific representation the detected planes take the form of is also important. If no uniform type of output can be determined, consequently no uniform metric for comparison can be found as well. Since the algorithms process point clouds, we choose to stay within the realm of points, i,e. an arbitrary plane should be represented by the set of points included in the plane (inliers). The representation being a list of points enables further processing of the detected planes. This can be useful for some sort of segmentation application,and different kinds of plane representations can be calculated through those points.

Learning based

Learning-based methods, e.g. Deep learning, generally have some varying level of bias, depending on the training data. Another reason against the use of learning-based methods is that we choose not to require a GPU to replicate our findings.

Availability

Lastly, we include the availability of an algorithm into our set of criteria. Each algorithm to be compared needs to run on the same system to exclude the underlying hardware from any experiments.
Furthermore, writing our own implementation for methods for which no implementation is available, or for which the respective publication does not focus on the implementation details, would go beyond the scope of this work.

RSPD - Robust Statistics Approach for Plane Detection

OPS - Oriented Point Sampling

3DKHT - 3-D Kernel-based Hough Transform

With 3D-KHT, as with other octree-based methods, the performance depends, to some degree, on the level of subdivision. In the provided implementation, the octree keeps dividing until either the amount of points in the current node is lower than a set minimum, or the octree has been divided at s_{level} times.
A total of six different parameter presets are included with the official implementation. Since this work does not focus on the evaluation and analysis of a single method, we performed experiments with all presets on a data set, the results thereof can be seen in Figure 3.1

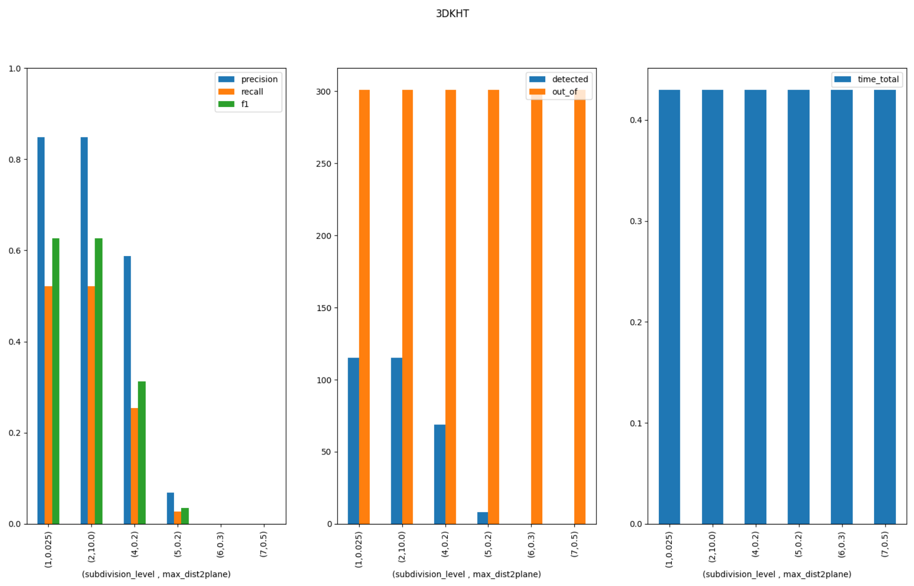

Figure 3.1: Test results of 3D-KHT with different parameters

Because the leftmost two presets, with an s_{level} value of 1 and 2, respectively, seem to yield equal results, but a higher value of subdivision might result in better performance on larger environments, we henceforth perform all calculations of 3D-KHT with an s_{level} value of 2.

OBRG - Octree-based Region Growing

PEAC - Probabilistic Agglomerative Hierarchical Clustering

Summary

hier ist ne tabelle:

	Input Data	Plane format	Learning-Based	Availability
RSPD	UPC	inliers	N	Y
OPS	UPC	inliers	N	Y
3DKHT	UPC	inliers	N	Y
OBRG	UPC	surely inliers	N	N
PEAC	OPC	inliers	N	Y
CAPE	OPC	normal, d	N	Y
SCH RG	OPC	inliers?	N	N
D-KHT	DI	inliers	N	Y
DDFF	DI	indices	N	Y
PlaneNet	I	piecewise planar sth	Y	Y
PLaneRecNet	I	? / -	Y	Y
PlaneRCNN	I	normal + ?	N	Y

Table 3.1: Plane Detection Algorithms

Summary Plane Detection Algorithms

To effectively compare the presented algorithms, the data on which each algorithm performs the plane detection should ideally be the same.

If we furthermore consider algorithms that run on anything other than UPC, it would necessitate finding a data set which includes equivalent point clouds for the structured, as well as the unstructured case. Since these algorithms would disregard the global structure of the point cloud, we deem them not feasable for our use-case and thus exclude them from our evaluation.

For the reasons previously stated in 3.2, we also exclude learning-based methods.

For an even comparison, the detected planes would have to be in the same format because, even for the same plane, representations could very well lead to different results, e.g. a plane in cartesian form compared to the same plane, described by its inliers.

Asserting, comparability, we exclude all methods, which do not offer a plane representation by inliers.

We lastly exclude all methods for which we didnt find any implementations, or necessary formulas.

Finally, we end up with, and thus include the following plane detection algorithms in our evaluation:

- RSPD
- OPS
- 3D-KHT
- OBRG?

3.3 How do we verify "real-time"?

To determine wether or not an algorithm runs in real-time, we have to define the meaning of real-time first.

We have to consider possible hardware limitations, data flow, and simply how often it is needed to perform calculations in correspondence with the given use-case.

The recordings are not directly sent to the plane detection algorithm, but instead given to RTAB-MAP, which then performs calculations to update and publish the map. Therefore, the upper limit is the frequency of how often RTAB-MAP publishes those updates, which by default is once per second. According to this upper limit, we consider an algorithm *real-time applicable*, if it achieves an average frame rate of minimum 1, e.g. the algorithm manages to process the entire point cloud and detect all planes in $1s$.

4 Implementation

To ensure a uniform comparison of the algorithms selected in the concept, the external circumstances must be identical for each algorithm. This chapter will detail the implementation thereof.

4.1 System Setup

It is necessary to perform all experiments on the same machine to ensure a consistent comparison. We implement all algorithms and further architecture on a Lenovo IdeaPad 5 Pro, which runs Linux Ubuntu 20.04.5. The laptop has an AMD Ryzen 7 5800H CPU and 16 GB of RAM.

We install the most recent ROS distribution, *ROS Noetic Ninjemys*, as well as *realsense-ros* with all additional dependencies.

4.2 Plane Detection Algorithms

We implement RSPD and OPS using their respective open source implementations on GitHub[1] [2]. Note that, while the implementation of RSPD is provided by the author, we could not determine whether the user who uploaded his implementation of OPS is affiliated with Sun and Mordohai. Both methods are implemented in C++ and depend on the C++ linear algebra library *Eigen*[3] and the C++ API of the Point-Cloud Library[4], *libpcl-dev*.

The authors of 3D-KHT, provide an implementation, in form of a Visual Studio project, on their website [5]. Since the laptop we use does not run Windows, we use *cmake-converter* [6] to convert the solution to a cmake project we can build using *make*.

[1] `https://github.com/abnerrjo/PlaneDetection`
[2] `https://github.com/victor-amblard/OrientedPointSampling`
[3] https://eigen.tuxfamily.org/index.php
[4] https://pointclouds.org/
[5] `https://www.inf.ufrgs.br/~oliveira/pubs_files/HT3D/HT3D_page.html`
[6] https://cmakeconverter.readthedocs.io/

OBRG

To our knowledge, no open-source implementation is available for the algorithm. We therefore use our own implementation. We implement the method in python, heavily relying on someLib for computation of something.

5 Evaluation

In diesem kapitel werden zuvor ausgewählte algorithmen einheitlich verglichen und die resultierenden ergebnisse ausgewertet.

5.1 Evaluation Protocol

To determine the feasability of performing real-time plane detection, we need to conduct experiments with the selected algorithms.

Another important factor for comparability is the data set, on which the experiments are conducted on. Because each publication from the presented algorithms uses a different data set for their evaluation, we cannot objectively select an algorithm to be the "best". Furthermore, to the best of our knowledge, there is no data set, that contains an incrementally growing and unordered point cloud with corresponding ground truth.

Thus, we first evaluate the algorithms on a dataset while excluding the temporal component. We do this by performing plane detection on whole point clouds, rather than incrementally growing ones.

We then perform experiments, this time including the temporal component, by performing calculations at each time step and evaluating them individually.

Lastly, through comparison, as well as analysis of those different experiments, a statement will be given as to whether and how well plane detection is possible in real-time.

For the evaluation of a given dataset, by comparing the test results with the ground truth, we took inspiration from Araújo and Oliveira, especially their *Results* chapter.

5.1.1 Metrics

To quantitatively evaluate an algorithms performance, we calculate the precision, recall and the f1 score. First we regularize the original point cloud to reduce complexity and furthermore to avoid proximity bias, because of the inverse relationship between distance to sensor and cloud density. This regularization is obtained through voxelization of the pointcloud.
With this voxelgrid we can now calculate corresponding sets of voxels for each point cloud representing a plane. In the next step, we compare our planes from the ground truth with the planes obtained from an algorithm to obtain a list of corresponding pairs of ground truth and found planes.
A grund truth plane p_{gt_i} is marked as *detected*, if any plane from the list of found planes achieves a voxel overlap of $\geq 50\%$. With this list of correspondences, we calculate precision, recall and the f1-score as explained in the following. For a given ground truth plane p_{gt_j} and a corresponding found plane p_{a_k} we can sort a given voxel v_i into the categories *True Positive(TP), False Positive(FP) and False Negative(FN)* as follows.

$$v_i \in p_{gt_j} \wedge v_i \in p_{a_k} \Rightarrow v_i \in TP$$

$$v_i \in p_{gt_j} \wedge v_i \notin p_{a_k} \Rightarrow v_i \in FN$$

$$v_i \notin p_{gt_j} \wedge v_i \in p_{a_k} \Rightarrow v_i \in FP$$

With those four rules, we can calculate the precision, recall and F1-score like this:

$$Precision = \frac{|TP|}{|TP| + |FP|}$$

$$Recall = \frac{|TP|}{|TP| + |FN|}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Aside from the accuracy, we also need to compare the time each algorithm needs to find its respective set of planes. For that, we measure the time spent in the plane detection phase, excluding any preprocessing or postprocessing steps. To measure the detection time, we simply log the exact times before and after calculations and write the differences to a file.

Scene Categories	Area_1	Area_2	Area_3	Area_4	Area_5	Area_6	TOTAL
office	31	14	10	22	42	37	156
conference room	2	1	1	3	3	1	11
auditorium	-	2	-	-	-	-	2
lobby	-	-	-	2	1	-	3
lounge	-	-	2	-	-	1	3
hallway	8	12	6	14	15	6	61
copy room	1	-	-	-	-	1	2
pantry	1	-	-	-	1	1	3
open space	-	-	-	-	-	1	1
storage	-	9	2	4	4	-	19
WC	1	2	2	4	2	-	11
TOTAL	45	39	24	49	55	53	272

Table 5.1: S3DIS Disjoint Space Statistics

5.1.2 Dataset

To evaluate each plane detection algorithm on even grounds, we select the Stanford Large-Scale Indoor Spaces 3D Dataset(S3DIS)[2]. The Dataset had been recorded in three different buildings, which then were divided into six distinct areas. Those six areas include a total of 270 different types of rooms, e.g. offices, hallways, WCs and two auditoriums to name a few.
Each room has a complete unstructured point cloud in form of a list of XYZ values, as well as a list of annotated files representing semantically different objects that can be found in this point cloud of the room. Since our focus lies not on 3D semantic segmentation, we manually select planar regions using CloudCompare[1], obtaining a list of sub-clouds.

5.1.3 Real-Life Test

We record an incrementally growing data set in the Faculty of Computer Science at Otto-von-Guericke University Magdeburg. Running *realsense-ros* and holding our cameras, we walk through different parts of the building, scanning to the best of our ability. We save each incremental update of the map to file for later usage.

We create a corresponding ground truth for only the most recent update, e.g. for the entire recording.

[1]https://cloudcompare.org/

During evaluation of a map at time m_t, we cut off the ground truth with respect to m_t so that we obtain a ground truth for each distinct time gt_t.

5.2 Results

5.2.1 Results Dataset

Hier sind die Ergebnisse:

5.2.2 Results Real-Life Test

Hier sind die Ergebnisse:

6 Conclusion

Possibly redundant with results section in evaluation

7 References

Bibliography

[1] Abner M. C. Araújo and Manuel M. Oliveira. “A robust statistics approach for plane detection in unorganized point clouds”. en. In: *Pattern Recognition* 100 (Apr. 2020), p. 107115. ISSN: 00313203. DOI: 10.1016/j.patcog.2019.107115. URL: https://github.com/abnerrjo/PlaneDetection.

[2] I. Armeni et al. “Joint 2D-3D-Semantic Data for Indoor Scene Understanding”. In: *ArXiv e-prints* (Feb. 2017). arXiv: 1702.01105 [cs.CV].

[3] Mathieu Labbé and François Michaud. “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation: LABBÉ and MICHAUD”. en. In: *Journal of Field Robotics* 36.2 (Mar. 2019), pp. 416–446. ISSN: 15564959. DOI: 10.1002/rob.21831.

[4] Frederico A. Limberger and Manuel M. Oliveira. “Real-time detection of planar regions in unorganized point clouds”. en. In: *Pattern Recognition* 48.6 (June 2015), pp. 2043–2053. ISSN: 00313203. DOI: 10.1016/j.patcog.2014.12.020. URL: https://www.inf.ufrgs.br/~oliveira/pubs_files/HT3D/HT3D_page.html.

[5] Bo Sun and Philippos Mordohai. “Oriented Point Sampling for Plane Detection in Unorganized Point Clouds”. en. In: arXiv:1905.02553 (May 2019). arXiv:1905.02553 [cs]. URL: https://github.com/victor-amblard/OrientedPointSampling.