1 Concept

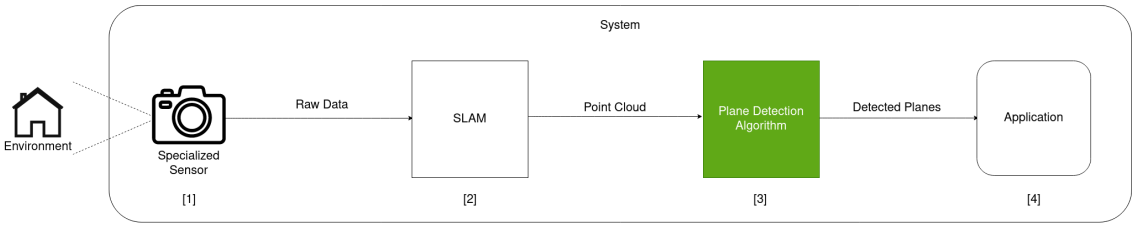


Figure 1.1: The procedure of the plane detection process. The specialized sensor records data ([1]), which is passed to a SLAM algorithm ([2]). After map assembly, a point cloud is handed to a plane detection algorithm ([3]). The detected planes are given to a use-case-specific application ([4]).

Internal Introduction Many AR and VR Systems integrate plane detection into their software, some use it only to calculate the ground floor while others use plane detection to build a smaller model of the environment. Die genauen Anwengunsmöglicheiten sind hierbei jedoch endlos. Figure 1.1 shows a generic block diagram of such a VR/AR system including plane detection. In general, the environment is continuously recorded by a specialized sensor which is usually a camera([1]). A SLAM algorithm then integrates the new data into its already existing map([2]). The map, in form of a point cloud, is subsequently passed to a plane detection algorithm([3]). The algorithm performs the necessary steps to detect all planes inside the current map and passes the planes to the application([4]). The application would then further process those planes, e.g., by creating a live visualization of them or by assisting the movement of visually impaired people [3].

Beim erstellen eines solchen systems ist die wahl des PDA natürlich von großer bedeutung. Das Problem daran ist, dass die meisten veröffentlichten algorithmen nicht von natur aus vergleichbar sind. Oft werden verschiedene Datensätze oder metriken benutzt, was einen vergleich durch quantifizierung ausschliesst. Alternativ sind algorithmen auf der internen funktionsweise nicht vergleichbar, da viele methoden verschiedene inputs voraussetzen und sich dementsprechend das format der ebenen genauso unterscheidet. Alles in allem ist es nicht möglich einen eindeutig "besten" algorithmus allein basierend auf den präsentierten Metriken auszuwählen.

Um die frage zu beantworten, welche algorithmus am besten geeignet ist, und ob dieser echtzeitfähig ist, machen wir einen einheitlichen vergleich von PDAs. Um diese evaluierung durchführen zu können brauchen wir die folgenden dinge:

1. Appropriate plane detection algorithms,
2. a useful dataset *and*
3. a definition of *real-time*.

Die folgenden sections kümmern sich darum.

1.1 Selection of Plane Detection Algorithms

Da, wie schon angemerkt die meisten algorithmen sich in bestimmten aspekten unterscheiden, ist es nicht möglich alle einheitlich zu vergleichen. Ferner sind nicht alle algorithmen aus der selben motivation entstanden und legen daher den fokus auf verschiedene dinge. Es wäre beispielsweise sinnlos einen algorithmus wie *Underwater SLAM* [5] auf die performanz in kleinen indoor umgebungen zu testen. Es ist somit nötig, zuerst objektive kriterien zu definieren woran oberflächlich beurteilt werden kann, welcher algorithmus in dem kontext dieser arbeit von nutzen sein könnte.

1.1.1 Criteria

In den folgenden paragraphen definieren und umreissen wir angemessene kriterien für die objektive beurteilung von PDAs.

Type of Input The first criterion is the type of input expected by a plane detection algorithm. Usually, the data representation of the recorded environment falls into one of three categories:

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

The fundamental difference between UPC and OPC is their format. Each point cloud has a *width* and a *height* parameter. An unorganized point cloud c is generally equal to an unordered 1D array of points, i.e., $width = |c|$ and $height = 1$. In contrast, the memory layout of an organized point cloud is a 2D array, where the width and height depend on the resolution of the used sensor. Taking the maximum resolution

of the T265(see Table **??**) as an example, the OPC would have a *width* and *height* of 1280 and 720, respectively. Intuitively, the value at index (0,0) would be in the top-left corner, and the value at index (*width,height*) would be in the bottom-right corner.

Depth images are inherently similar to organized point clouds, given their resolution and two-dimensional structure. The primary difference is that the values stored in the array are distances to the sensor instead of 3D coordinates.

Consequently, since unorganized point clouds are not limited in their dimension, they are more suitable for capturing entire environments.

As stated before, we focus on detecting planar structures in the entire environment rather than just distinct segments. In addition, only the unorganized/unstructured point clouds offer a complete view of the recorded environment.

Detected Plane Format Which specific representation the detected planes take the form of is also essential. If no uniform output type can be determined, consequently, no uniform metric for comparison can also be found. 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. A list of points would, in contrast to some plane equation, enable us to detect holes in planes, e.g., an open door or window, which can be helpful in any use case involving remodeling architectural elements. It also allows further filtering of planes based on a density value that we can calculate over the bounding box and the number of points, e.g., removing planes with a density lower than a certain threshold.

Learning based Some methods are learning-based, e.g., they use deep learning to detect planes in point clouds or images. Their ability to generalize depends on the choice of training data. If an algorithm is perfectly trained on a dataset that consists of only tables, the algorithm would find all table tops but might fail to detect planes that do not have a certain number of legs attached to them.

1.1.2 Plane Detection Algorithms

A list of state-of-the-art algorithms is compiled through comprehensive research of the current literature on plane detection (see Table 1.1). explanation of table

Im folgenden werden aus den zuvor aufgestellten kriterien die für diese arbeit sinnvollsten werte(?"ich nehme UPC aus UPC, OPC, DI... idk wie ich das nennen soll)2 ausgewählt und anhand dessen unpassende algorithmen von der evaluierung ausgeschlossen.

Plane Detection Algorithm	Input Data	Plane Format	Learning-Based
RSPD [1]	UPC	inliers	N
OPS [12]	UPC	inliers	N
3DKHT [6]	UPC	inliers	N
OBRG [14]	UPC	inliers	N
PEAC [4]	OPC	inliers	N
CAPE [10]	OPC	normal, d	N
SCH-RG [9]	OPC	inliers	N
D-KHT [13]	DI	inliers	N
DDFF [11]	DI	inliers	N
PlaneNet [7]	I	normal, d	Y
PlaneRecNet [15]	I	reconstructed scene	Y
PlaneRCNN [8]	I	normal, d	Y

Table 1.1: Plane Detection Algorithms

Addressing the criterion of input type, we are only interested in performing plane detection in complete environments. We hereby consider organized point clouds or images inappropriate because they do not offer a complete view on a scene.
We hereby exclude *PEAC, CAPE, SCH-RG, D-KHT, DDFF, PlaneNet, PlaneRecNet* and *PlaneRCNN* from our evaluation.

Secondly, the detected planes need to be in the same format because, even for the same plane, different representations could very well lead to different results. Assume a plane in cartesian form and a plane represented by its inliers. The calculated metrics may differ significantly because the plane in cartesian form is infinitely dense. In contrast, the plane described by its inliers allows for holes and non-rectangular shapes, e.g., doorways or a round table. We thereby determine *inliers* as the preferred plane format and exclude all methods which do not comply, namely *CAPE, PlaneNet, PlaneRecNet*, and *PlaneRCNN*.

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

- **RSPD**
- **OPS**
- **3D-KHT**
- **OBRG**

1.2 Datasets

@marco Sollte hier dann nicht der anfang der evaluation hin? also bzgl warum wir das aufteilen in temporal und nicht temporal? dann wäre aber halt die intro der eval quasi leer

1.2.1 S3DIS Experiment

This subsection deals with the selection of the dataset without the temporal component.

In Section 1.1, we determine unorganized point clouds (UPC) as input for the selected algorithms. Furthermore, this work focuses on real, indoor environments. Additionally, a ground truth is necessary for the evaluation. We select the Stanford Large-Scale Indoor Spaces 3D Dataset(S3DIS)[2] from the list of datasets to evaluate each plane detection algorithm on even grounds. Since our focus is not on 3D semantic segmentation and the provided ground truth is focused on semantic segmentation rather than planes, we manually select planar regions using CloudCompare[1]. S3DIS was recorded in three different buildings and divided into six distinct areas, including 272 different scenes. A detailed statistic of the included scene types can be found in Table 1.2. An individual scene has a complete unstructured point cloud and a list of annotated files representing semantically different objects that can be found therein.

Furthermore, one could argue that an uneven distribution of scene types introduces a particular bias. While it is true that the distribution is quite uneven, the dataset nevertheless reflects a realistic distribution of scene types since it is not realistic if a building contains only lecture halls. Inversely, it is appropriate to assume that an office complex contains a substantial amount of hallways needed to connect all offices.

[1] https://cloudcompare.org/

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 1.2: S3DIS Disjoint Space Statistics

1.2.2 FIN Experiment

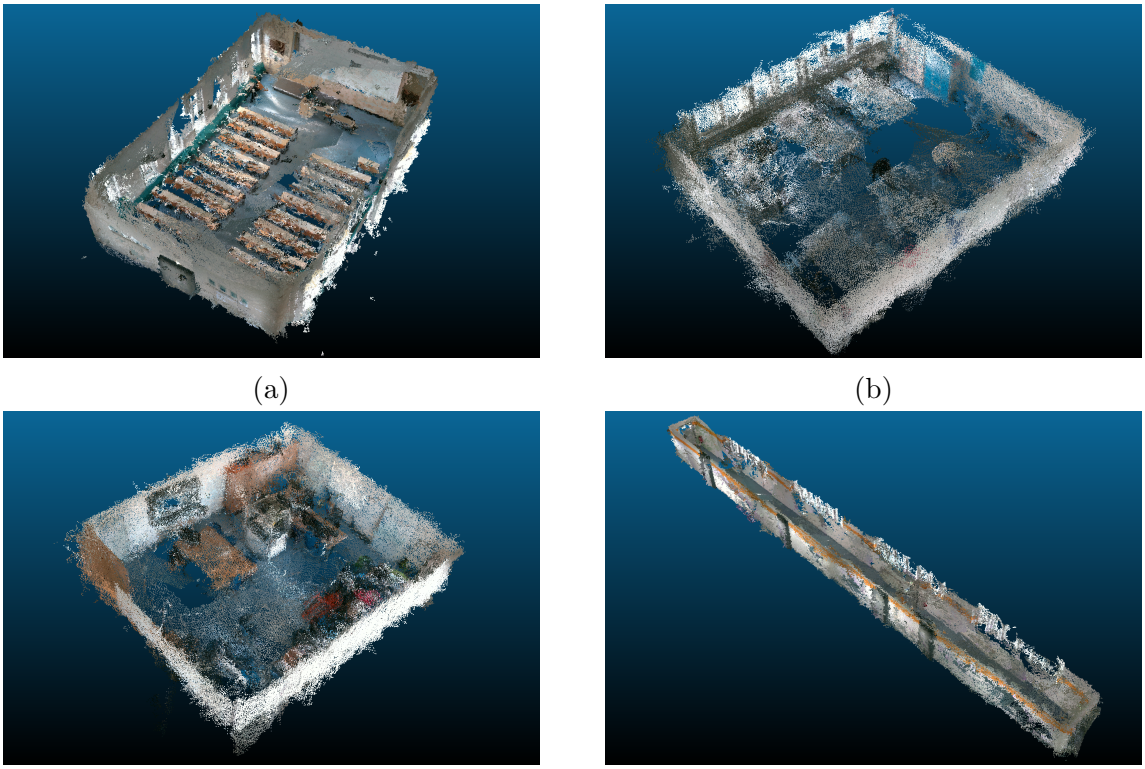
We record an incrementally growing dataset in the Faculty of Computer Science at Otto-von-Guericke University Magdeburg. To perform a thorough comparison between the FIN and S3DIS, we record a scene for each of the following scene types:

- office
- conference room
- auditorium
- hallway

We focus on these four scene types because they are the most common in a real environment.

The recorded point clouds can be seen in Figure 1.2.

Running *realsense-ros* and holding our cameras, we walk through the aforementioned parts of the building while scanning to the best of our ability. We save each incremental map update to a file for later usage.



(a) (b) (c) (d)

Figure 1.2: The recordings for each scene type: (a)auditorium, (b) conference room, (c) office and (d) hallway.

Since no ground truth exists for a novel dataset like this, we create a set of ground truth planes gt_{end} for only the most recent update of each scene, e.g., for the entire recording. To prepare for the evaluation of a map m_t at a given time t, we crop all planes in gt_{end} by removing all points that are not present in m_t, as shown in Figure 1.3. We speed up this expensive process by employing a KD-Tree neighbor search with a small search radius since we only need to know whether a certain point is present or not.

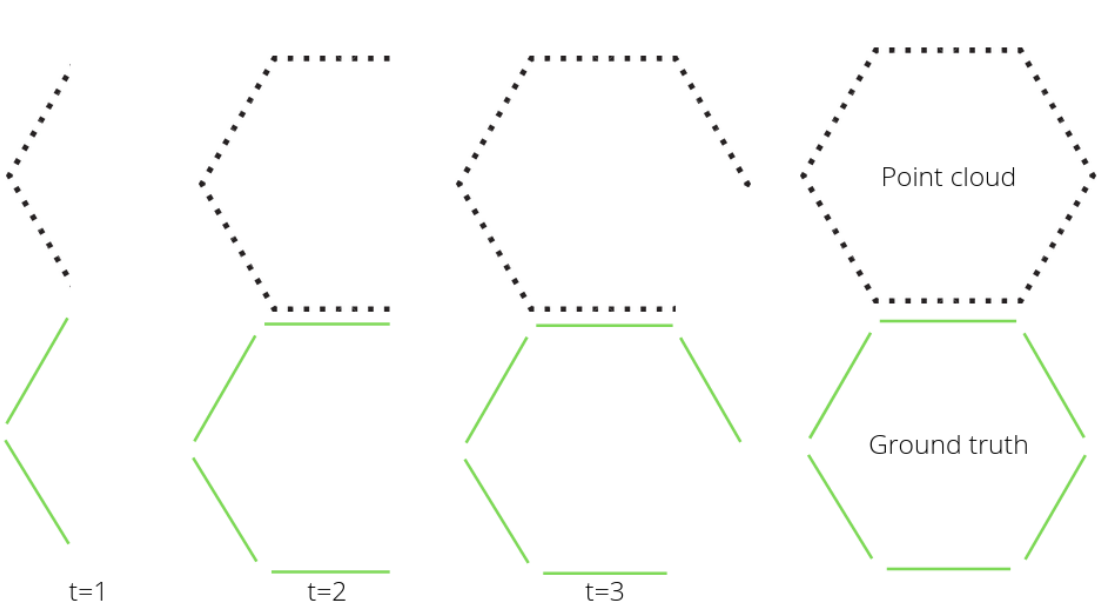


Figure 1.3: Dynamic ground truth generation. All planes that are included in *Ground Truth* are cropped depending on the available point cloud at each time t

1.3 Definition Real-Time

To determine whether or not an algorithm runs in real-time, we must first define the meaning of real-time.

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 recorded raw data is 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 within one second.

1.4 Summary

Many applications have constraints in the form of a temporal component. Augmented or Virtual Reality applications that include plane detection are no exception. In addition to time constraints, good quality is usually tightly coupled to expensive sensors. To evaluate to what extent it is possible to perform precise plane detection with a real-time constraint on off the shelf hardware, we compare selected algorithms.