UNIVERSITÄT OSNABRÜCK

INSTITUTE FOR COMPUTER SCIENCE
KNOWLEDGE-BASED SYSTEMS

Dissertation

Transparent Object Reconstruction and Registration Confidence Measures for 3D Point Clouds based on Data Inconsistency and Viewpoint Analysis

submitted to the Faculty of
Mathematics / Computer Science of Osnabrück University
in fulfillment of the requirements for the degree of
Doctor Rerum Naturalium (Dr. rer. nat.)
by

Sven Albrecht

September 2017

First supervisor:	Prof. Dr. Joachim Hertzberg
Second supervisor:	Prof. Dr. Andreas Nüchter

Preface

In contrast to the guidelines and expectations published from the administration of Osnabrück University for PhD theses, the process, at least according to my experiences, is not always strictly linear and straight forward. One of the requirements, before starting a PhD is to compile an exposé about the thesis and its contents, ideally in combination with a timetable that shows which result is to be achieved at which point of time. In my personal opinion science does not necessarily work in this way. While one should definitely have a goal in mind that one wants to investigate, a priori unexpected results, that pique one's interest, might occur during the investigation progress Not mentioning such an event in the initial outline and not allotting time for it in some table, should not prevent further investigation. At least this is one of the central points of scientific research, from my point of view: if you detect some interesting and unexpected behavior, then try to find a satisfying explanation or a way to handle this problem.

In this regard, I speak from personal experience, because my initial draft of what I wanted to do in my thesis has very little in common with the final result. The topics and techniques that are discussed in the following emerged more or less as by-products of the original topic, but were too fascinating, at least for me, to just leave them alone. I would like to take the opportunity to thank my supervisor Prof. Dr. Joachim Hertzberg for giving me the freedom to stray away from the initial plan and pursue those topics that I deemed more interesting. Apart from this freedom, I am truly grateful for the fruitful discussions that we had about my ideas and everyday working environment in the Knowledge Based Systems research group of Osnabrück University, which was always a nice place to come to work.

Furthermore, I would like to thank Dr. Stephen Marsland and Prof. Dr. Werner Guesgen from Massey University, New Zealand. It was during my stay at Massey that I discovered the properties of highly specular and transparent objects in RGB-D data (with the, at that time, brand new Microsoft Kinect). This observation was a byproduct while recording scenes with cooking pots and frying pans on a stove, originally to be used in the context of smart homes. The discussions with Stephen put me on the track that culminated in the approaches about transparent object reconstruction that are presented in one part of this thesis.

Also in connection with this topic, I am very grateful to Michael Görner, now at University of Hamburg, for his cooperation when conducting part of the experiments for transparent object reconstruction. Without his willingness to share his "curious table exploration", many of the presented results would not have been accomplished in the form they are presented here. In addition, Michael provided valuable feedback, pointing out flaws in earlier attempts to deal with partial observations and multiple recordings from the same pose, in the reconstruction, that lead to the final *viewpoint-based* reconstruction.

Other thanks are due to Dr. Kai Lingemann for his input and patience in explaining some of the more intricate parts of LaTeX to someone with only journeyman insight. Thanks to Jonas Schneider and Jan Hendrik Schoenke for interesting discussions during coffee breaks and at canteen (and on the way), that made work days more enjoyable. Last but not least also thanks to all other members of the KBS research group for their constructive inputs and the sense of companionship.

Many thanks to Juan Carlos Saborío and Sebastian Graetz for proofreading the text and providing advice for rephrasing certain parts. Also many thanks to W. Strunk and E.B. White, authors of *The Elements of Style* [74] for providing some clear rules in my conflict with English punctuation rules (and to Prof. Hertzberg for the recommendation).

Last, but not least, I would like to thank my parents for their encouragement and support through the years. Without them and their belief in me, I would not have accomplished finishing this thesis.

Contents

Chapter 1

Introduction

When working with sensor data in the domain of mobile robotics, one quickly realizes that there is no perfect sensor that can be employed equally well for every task. Terrestrial high-end laser scanners are capable of creating detailed and very precise point clouds of the environment, but are expensive, bulky and the scanning process is too slow to employ them for tasks like obstacle avoidance in dynamic environments. Such tasks can, at close ranges be performed by ultrasonic sensors, due to the shape of their beam spread. However, exactly this property makes ultrasonic sensors rather unsuitable for mapping tasks.

Although each sensor has its strong and weak points, certain types are (currently) very popular in the robotics community, namely, RGB-D cameras, Time of Flight cameras and different types of laser scanners. One big advantage of (regular) cameras is the easy interpretability of the acquired data for a human, since they perceive the environment in a fashion that is very similar to ours. While not all Time of Flight cameras provide color information, they provide reflectance values which produce gray scale images of the environment in addition to the acquired 3D data. The 3D point clouds that all three sensor types produce are also easy to interpret, after little familiarization.

All these sensors have in common that their measurement principle is based on emitting light signals (with differing wavelengths for the types) and employing some mechanism to either perceive the reflected signal or observe its projection in the scene, in order to estimate depth values for individual measurement rays. Unfortunately, that also means that surfaces that are highly reflective (mirrors or polished steel) or transparent (glass, translucent plastic) can reflect or refract the emitted signals. Depending on the sensor and the transparent / specular object, this may either lead to measuring several points at the wrong location, when the distance is wrongly estimated, or to the absence of measurement points, when the signal is reflected in such a way that it is not perceived anymore by the sensor.

Identifying environments where such objects are present is a necessary task for mobile robots, in order to be able to autonomously explore unknown environments. While employing different sensors, based on other measurement principles, like sound, may seem suitable to solve the problem of detecting transparent / specular objects in a given scene, fusing the information of different sensor types is no trivial task, especially in cases where the information of different sensors contradicts each other and both readings appear plausible. Furthermore, keeping in mind that no sensor is suitable for every possible task, finding a way to exploit a sensor's characteristic weakness in order to enrich its measurement capabilities is a welcome benefit and might also help to think differently about other sensor limitations and how to deal with them.

This idea of exploitation of the missing or erroneous data leads to the central point of this thesis, namely, checking for data inconsistencies. Usually consistent data is desired, when trying to model parts of a mobile robot's environment or identify objects in given data. As it turns out, data inconsistencies can also be used to gain knowledge about certain parts of the environment, even if these contain objects that are not directly measurable by the employed sensor. Detecting inconsistencies in the data acquired in a single RGB-D camera frame or laser scan might be enough to determine if a transparent / specular object is likely to be present, but is in general not sufficient to reliably estimate its size and shape.

As will be shown, this can be accomplished, when information of several frames, recorded from different poses, is combined under consideration of the different viewpoints and their constraints. A requirement for this idea is that the data that was acquired from different positions and with different sensor orientations is combined correctly (or consistently). This in turn leads to the second topic of this thesis, namely, automatically evaluating how well two point clouds are aligned, given the transformation between the two sensor poses that were used to record the data. The alignment analysis is based on the same principles as the detection and reconstruction of transparent objects, namely, searching for data inconsistencies under consideration of sensor specific viewing constraints and in-scene occlusions. Setting the detected inconsistencies in relation to the consistent parts of the aligned point clouds will serve as a basis for the evaluation.

1.1 Thesis Outline

Trying to incorporate both, consistencies and inconsistencies, when gathering information from multiple sensor readings is usually a two step process: First a well-known technique that creates a result as consistent as possible is employed and afterwards this result may be checked for special types of inconsistencies.

In the context of this thesis the data that will be investigated consists of 3D point clouds, gathered either from RGB-D cameras or laser scanners. Point clouds gathered from different poses in the observed environment are registered with the widely used ICP algorithm. This corresponds to the first step in the process, namely, fusing the available sensor data consistently. The needed fundamentals in terms of necessary vocabulary and common processing techniques for 3D data, as well as a brief introduction of ICP are presented in chapter 2 to establish a common understanding with the reader about these topics.

This is followed in chapter 3 by demonstrating how the data inconsistencies, caused by the presence of transparent or specular objects in the recorded scene, can be identified in single frames, using a scene analysis that considers viewpoint constraints and in-scene occlusions. Furthermore, two different methods, based on the same principle, will be introduced. These methods can approximate the shape of a transparent object, provided multiple views from different poses of the same scene and a consistent registration of the point clouds.

Subsequently in chapter 4 a novel approach is introduced to automatically evaluate the results obtained by the ICP algorithm, again taking into account potential inconsistencies on the provided point cloud alignment, which goes beyond the regular average error in point-to-point correspondences provided by ICP.

Chapter 5 demonstrates how the approaches detailed in chapters 3 and 4 perform in various experiments. Reconstructions of transparent objects are subject to a quantitative analysis with respect to their shape and size and compared to ground truth instances as well as reconstructions of equal non-transparent object instances. To illustrate the validity of the proposed alignment assessment, the introduced methods are applied to datasets of vastly different properties.

The thesis is completed with a discussion of the presented material and an outlook in chapter 6.

1.2 Scientific Contributions and Goals

This thesis introduces a novel approach that utilizes the absence of expected measurements to approximate the shape and volume of three dimensional transparent objects that are not directly perceivable with the employed sensor setup. In contrast to other methods employed for similar purposes, the presented approach does neither need a special training phase, nor is the shape of the reconstructed objects restricted to a few classes of predefined objects. The desired goals for this reconstruction are as follows:

- A robust detection of transparent / specular objects in the scene that does not miss the presence of such objects nor detects them on a regular basis in scenes that do not contain transparent objects, i.e., the rates for false positives and false negatives should be small
- The reconstruction needs to work in a robust fashion with respect to
  - Varying lighting conditions
  - Sensor noise
  - Imperfect scene alignment (to a certain degree)

  These three demands are necessary requirements to apply the proposed reconstruction method in realistic settings on a mobile platform, that operates outside of a completely controlled lab environment.
- No prior knowledge about the objects in terms of shape or size is required; this also implies that no learning is needed for the reconstruction.
- Approximation of size and shape of the reconstructions needs to be sufficient for at least collision avoidance. If the quality of the reconstructions also allows for object manipulation, this will be a welcome benefit.

While not a scientific contribution in itself, this method illustrates that purposefully searching for specific data inconsistencies in otherwise largely consistent data, can be used to enrich the gathered measurements beyond the sensor's inherent capabilities.

Employing a similar line of thought, i.e., having some expectations concerning the gathered measurements and comparing these with the actually gathered data, the second part of the thesis provides a new heuristic quality measure for the alignment of two (partially) overlapping point clouds, where the alignment might be provided by the ICP algorithm or some other alignment method. Requirements for such an evaluation can be condensed to

- A numerical output that is independent of the size and nature of the aligned point clouds, since this is needed for the evaluation to be scene-independent.
- The output needs to be easier to interpret than the ICP output parameters and suitable for the comparison of different alignments (which is not the case for the ICP parameters).
- No prior scene knowledge needs to be incorporated and only basic knowledge about the sensor parameters that were used to record the data.
- Analysis needs to be performed in such a way that only considers the parts of the scene observed by both sensors. Other parts of the point clouds cannot serve as evidence for good or bad alignment.
- The method needs to be applicable to input data of different scales and different degrees of sensor noise.
- Parameter settings need to be sensor dependent, but independent of the recorded scene; only in this way will it become possible to evaluate unknown scenes. If for each dataset new parameter settings need to be established the benefit of such an evaluation is limited, at best.

Creating an evaluation method that complies with these requirements is a novel endeavor, hence very little related work exists. Common practices often completely ignore numerical evidence, when assessing the alignment quality, but refer to the visual appeal of results to a human observer ("...looks good / better..."). In those cases where a quantitative analysis is performed, some sort of ground truth is (painstakingly) acquired beforehand, against which obtained results are compared. While the first practice cannot be satisfying from a scientific point of view, the second is simply not feasible for all scan matching and mapping scenarios. Furthermore, if ground truth (data) was available for all possible scenarios the original task of scan matching or mapping would become superfluous.

Both topics in this thesis, transparent object reconstruction and alignment evaluation, are heuristic in their nature. Hence, it is possible to create scenarios where both methods will fail to work properly, but in general they will work well. To validate this claim, their performance is tested and evaluated in several experimental settings. Furthermore, both approaches allow for "soft real-time" applications, i.e., they can be performed during normal operations on mobile robots. In the case of the transparent object reconstruction this is directly demonstrated in one of the experiments.

Chapter 2

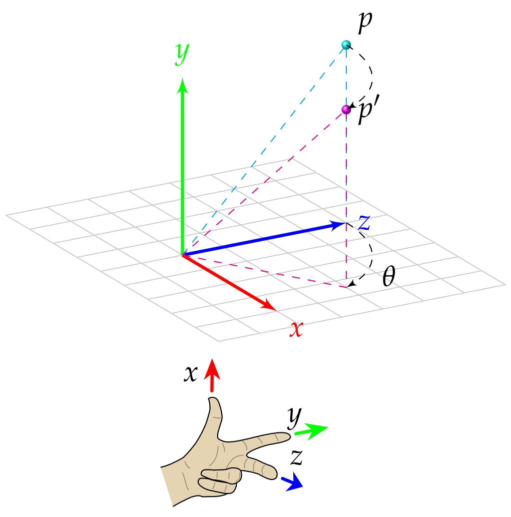
Fundamentals & Definitions

2.1 Coordinate Systems

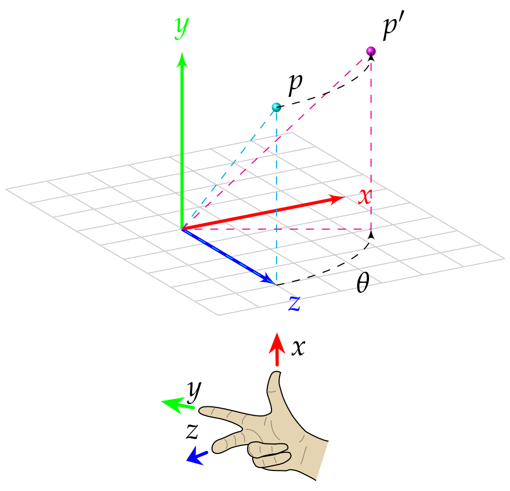
Since this thesis focuses on 3D data (or its absence in specific places) we should first take a brief look at the space into which the 3D data is embedded. Only range sensors were used in the context of this thesis that measure Euclidean distances from the sensor's origin to some obstacle. For such sensors there are two intuitive ways to define coordinate systems that may be used to embed the collected measurements into.

We can either use a *spherical coordinate system* or a 3D *Cartesian coordinate system*. Note that while spherical coordinate systems are inherently three dimensional, Cartesian coordinate systems do not have a fixed dimension: they can be one dimensional (a straight line with a chosen origin), two dimensional (two perpendicular lines, with a fixed ordering, intersecting at the origin), three dimensional or of higher dimension (where the visualization tends to get complicated). In the remainder of the thesis Cartesian coordinate (system) will refer to the three dimensional case, if not explicitly mentioned otherwise. Commonly the three coordinate axes are referred to as x-axis, y-axis and z-axis and thus a three-tuple $(x, y, z)^T$, with $x, y, z \in \mathbb{R}$, unambiguously describes a distinct location in a given Cartesian coordinate system. The location of such a three-tuple coincides with the geometric interpretation of a *point*. Since points in the context of this thesis may also incorporate additional information beside the coordinate values, they are presented more formally later in this chapter (see section 2.2). Until then, point in this text will be used synonymously with the location defined by a coordinate three-tuple. Note however, that later on it becomes convenient to think of Cartesian points as column vectors, mainly in the context of transforming points. Hence, they are, when used in text, marked as transposed.

However, Cartesian coordinate systems can differ in terms of the labeling, interpretation and arrangement of the axes and subsequently also in the way rotations around an axis are specified. The arrangement of the axes is usually split into *left-handed* and *right-handed* coordinate systems. The axes in left-handed coordinate systems are arranged in a way that if you spread thumb and index finger of your left hand in a way that they are perpendicular to each other and also point the middle finger in a direction perpendicular to both, thumb and index finger, these three fingers correspond to the coordinate axes in a left-handed coordinate system:



(a) Left-handed coordinate system



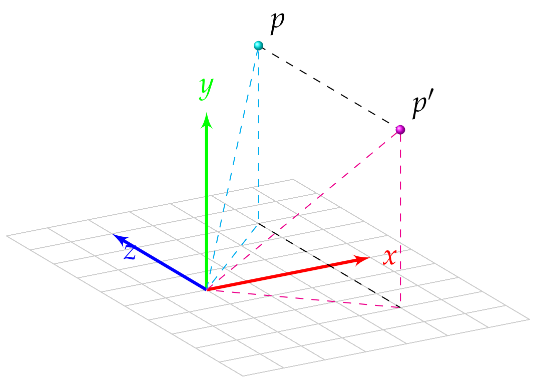
(b) Right-handed coordinate system

Figure 2.1: Illustration of left- and right-handed Cartesian coordinate systems. Both **(a)** and **(b)** display the same point $p = (0, 1, 1)^T$ and p', its rotation by $\theta = 60\,^\circ$ around the y-axis. The hands below **(a)** and **(b)** clarify the naming.

if the thumb is aligned with the x-axis (tip pointing towards the positive direction of the axis), the index finger will point into the direction of the positive y-axis and the middle finger points towards the positive z-axis. An illustration is provided in Figure 2.1.

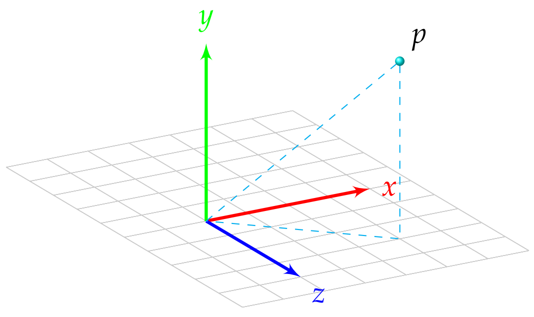
Commonly left-handed coordinate systems define rotations around a coordinate axis clockwise, i.e., in a negative mathematical sense, if the rotation axis points towards the observer (see rotation around y-axis from point p to p' in Figure 2.1a). Colloquially this is also referred to as the "left-hand" rule, since the rotations follows the direction of the bent fingers of the left hand, if the thumb points into the direction of the positive rotation axis. Right-handed coordinate systems on the other hand usually follow the "right-hand" rule, therefore rotations are defined in a counter-clockwise manner around the rotation axes (Figure 2.1b). The interpretation of the axes differs, depending on context / application.

For example the *Robot Operating System* (ROS) [59] uses a right-handed coordinate system, where the x-axis points into the "forward direction" (for example of a mobile robot), the y-axis points to the "left" and the z-axis points upward. The *Point Cloud Library* (pcl) [64, 83] also employs a right handed coordinate system, but here the x-axis points "right", y-axis points "down" and the z-axis corresponds to the viewing direction. With this arrangement the x- and y-axis are aligned in a fashion similar to common image coordinate axes (see [16], for example), while the z-axis points into the direction of measured depth. As a final example of right-handed coordinate systems, in OpenGL the x-axis points to the "right", y-axis "up" and z-axis toward the observer, see [71]. Incidentally this corresponds to the coordinate system depicted in Figure 2.1b. Since all the above mentioned coordinate systems are right-handed, it is fortunately possible to convert them into each other via rotations if we assume the origins to coincide or via a combination of a rotations and a translation otherwise (see section 2.2).



(a) Left-handed coordinate system **(b)** Right-handed coordinate system

Figure 2.2: Left-handed **(a)** and right-handed **(b)** coordinate systems with point $p = (x, y, z)^T$ displayed in cyan. While x- and y-axes coincide in **(a)** and **(b)** the z-axes point into opposite directions. Mirroring p in **(a)** on the xy-plane yields point $p' = (x, y, -z)^T$ (in magenta), which is perceived similar as point p in **(b)**.



Left-handed coordinate systems are also used on occasion, for example in the 3D toolkit (3DTK) [82] where the x-axis points "right", y-axis "upwards" and z-axis away from the observer. Left handed coordinate systems with different arrangements of the axes, e.g., x-axis pointing "left" and z-axis "up", can be converted into other left-handed coordinate systems by rotations (and a translation), just as right-handed coordinate systems. Note however, that rotations (and translation) are not sufficient to convert a left-handed coordinate system into a right-handed coordinate system:

Assuming coinciding origins we can arrange a left and a right handed coordinate system only in such a way that two of the three coordinate axes match. The remaining axis will point in opposite directions (see illustration in Figure 2.2), i.e., points are perceived as mirrored at the plane defined by the origin and the two coinciding axes. Therefore, the coordinate of a point that refers to the non-aligned axis needs to be multiplied by -1 if points from left-handed coordinate systems are to be converted into right-handed coordinate systems or vice versa. For instance if x- and y-axes of a left-handed and a right-handed coordinate system are aligned a point $p = (x, y, z)^T$ in one of them is perceived to be in the same location as point $p' = (x, y, -z)^T$ in the other coordinate system (see Figure 2.2).

The previous paragraphs, exemplary listing some of the possibilities for Cartesian coordinate system orientation and meaning, might illustrate that it is sensible to explicitly specify the coordinate system that will be used in the remainder of this thesis. If not mentioned otherwise, the used coordinate system will correspond to the coordinate system used in pcl (right-handed, z-axis corresponds to "viewing direction"), as illustrated in Figure 2.3. In the following the term *viewing direction* will therefore refer to the z-axis, while "right" and"down" corresponds to the x-axis and y-axis respectively. Furthermore, the *horizontal viewing plane* will refer to the hyperplane created by origin O, x-axis and z-axis, while the *vertical viewing plane* is composed by O, y-axis and z-axis.

While the Cartesian coordinate system is more common, I will briefly introduce spherical

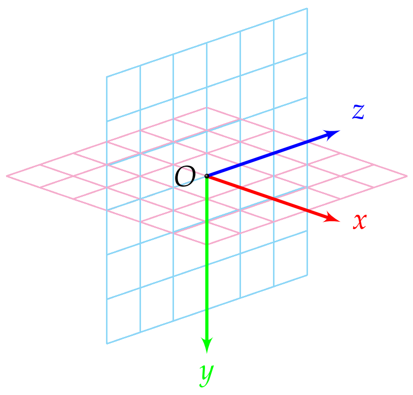


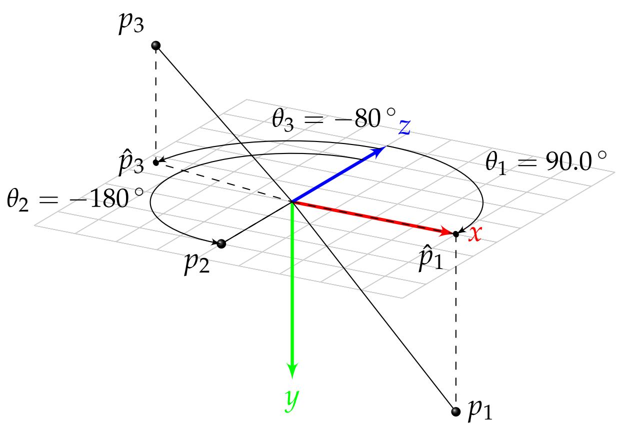
Figure 2.3: Coordinate system employed in this thesis: Origin at O, z-axis points into *viewing direction*, x-axis points right and y-axis down.The xz-plane (magenta) is referred to as the *horizontal viewing plane* while the yz-plane (cyan) is also known as the *vertical viewing plane.*

coordinate systems and illustrate why they are also suitable to represent the measured data. Spherical coordinate systems can be thought of as the extension of polar coordinate systems into three-dimensional space. While 2D polar coordinate systems refer to a point in space via two-tuples (r, θ), with $r \in \mathbb{R}^+$ being the distance to the origin and $\theta \in [-\pi, \pi)$ denoting the *azimuth* angle, spherical coordinates are three-tuples (r, θ, φ), where $\varphi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ describes the *elevation* angle. Angles are measured, as common in right-handed coordinate systems, counter-clockwise. If the need arises to mark the coordinated that belong to a specific spherical point, subscripts will be used to show the correspondence, e.g., $p_j = \left(r_j, \theta_j, \varphi_j\right)^T$. The same notation is also applied in case of Cartesian coordinates.

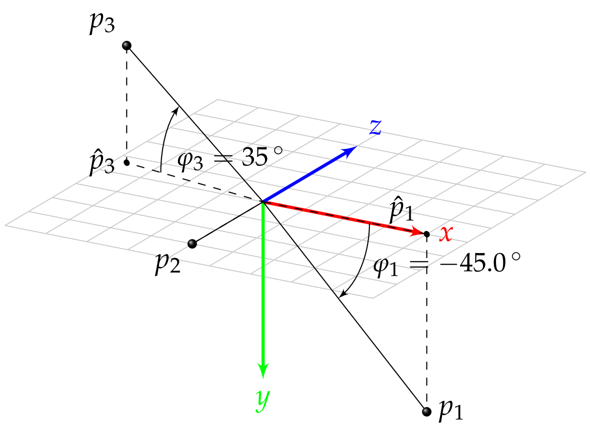
For a given point $p_i = (x_i, y_i, z_i)^T$ in Cartesian coordinates, that we will here also identify with the vector from the origin $O = (0, 0, 0)^T$ to p_i, the azimuth angle is defined as the angle between the viewing direction and the projection $\hat{p}_i$ of p_i onto the horizontal viewing plane. Using the Cartesian coordinate system defined above (see Figure 2.3), for the projection $\hat{p}_i = (x_i, 0, z_i)^T$ holds. If p_i is placed in viewing direction onto the vertical viewing plane (i.e., $x_i = 0, z_i > 0$) the azimuth angle θ_i becomes 0, since $\hat{p}_i$ is aligned with the viewing direction (positive z-axis). Conversely, points directly "behind" the origin ($x_i = 0, z_i < 0$) feature an azimuth angle of $\theta_i = -\pi$, with the minus sign denoting that the rotation is in clockwise fashion around the y-axis. Points located directly to the "right" ($x_i > 0, z_i = 0$) have azimuth $\theta_i = \pi/2$. An illustration of the relations between Cartesian coordinates and the related values for spherical coordinates is provided in Figure 2.4.

In the context of 3D data, obtained by some sort of sensor, spherical coordinates provide a better resemblance of the actual workings of the sensors: most sensors emit some sort of measurement signal in some known direction in relation to the sensor (thus setting azimuth θ and elevation φ for an individual measurement). The received signal is than mapped to a range r, that specifies the distance between sensor and some measured surface in the previously defined direction. Depending on the employed measurement principles of the sensor, the way to map received signal to distance differs.

Elevation is measured as the angle between p_i and $\hat{p}_i$, its projection onto the xz-plane. For



(a) Illustration azimuth angles θ_i



(b) Illustration elevation angles φ_i

Figure 2.4: (a) azimuth angles $\theta_1, \theta_2, \theta_3$ and **(b)** elevation φ_1, φ_3 shown for points p_1, p_2, p_3. Note that φ_2 is not depicted in **(b)** since p_2 lies in the xz-plane and therefor $\varphi_2 = 0$ holds and projection $\hat{p}_2$ coincides with p_2.

points p_i that lie above the horizontal viewing plane ($y_i < 0$) the elevation angle is positive, while the elevation φ_i for points below the horizontal viewing plane ($y_i > 0$) becomes negative. In addition points p_i placed on the positive y-axis ($x_i = z_i = 0$, $y_i \in \mathbb{R}^+$) have an elevation of $-\pi/2$, while points placed on the negative y-axis (note that the negative y-axis corresponds to "up" (see Figure 2.3)) have an elevation of $\varphi_i = \pi/2$.

Note that, since azimuth and elevation can be seen as rotations around coordinate axes, commonly known as *Euler angles*, the order in which the rotations are applied is not commutative. Subsection 2.2.5 will elaborate a bit more on the non-commutative property of Euler angles and provide a visual example in Figure 2.9. For spherical coordinates first the rotation by elevation angle φ_i is applied (as a rotation around the x-axis – since the rotation is done counter-clockwise a positive elevation angle φ_i results in $y_i < 0$) and afterwards the rotation around the y-axis, defined by azimuth angle θ, is applied. Perhaps surprisingly, for a point $p_j = (1, 1, -1)^T$ azimuth and elevation are not equal, but $\theta_j = 45\,°$ and $\varphi_j = 35.25\,°$ turn out to be the correct azimuth and elevation angles.

As with Cartesian coordinate systems, these conventions are not fixed. The azimuth angle is sometimes specified in the range of $[0, 2\pi)$ rather than $[-\pi, \pi)$. In this case an azimuth θ of 0 usually also refers to the viewing direction and increasing azimuth angles denote, assuming our defined coordinate system, rotations around the y-axis only in counter-clockwise direction. Instead of elevation oftentimes the *inclination* or *polar angle* is used with a range of $[0, \pi]$. Inclination is defined as the angle between vector p and the "up" direction.

In the standard coordinate system (Figure 2.3) "up" corresponds to the negative y-axis, so a point $p_j = \left(0, y_j, 0\right)^T$, with $y_j < 0$, that is located on the negative y-axis has an inclination $\varphi_i = 0$, while points on the positive y-axis have inclination $\varphi_i = \pi$. Note that, if elevation φ_i is at its limits (either $-\pi/2$ or $\pi/2$) the azimuth angle is arbitrary in the sense that all three-tuples

$\left(r_i, \theta_i, \pm\frac{\pi}{2}\right)^T$ denote the same location for fixed r_i, independent on the value of θ_i, namely a point on the positive or negative y-axis with distance r_i to origin O. Also if $r_i = 0$ holds, the values for azimuth and elevation become arbitrary, since all three-tuples $(0, \theta_i, \varphi_i)$ coincide with (Cartesian) origin $O = (0,0,0)^T$.

In physics it seems to be common to change the order of azimuth and inclination, so that θ in the three-tuple (r, θ, φ) specifies the inclination angle rather than the azimuth. In this way the ordering in the three-tuple reflects the order in that the rotations are applied (first rotation defined by inclination (in the physics nomenclature θ), afterwards the rotation defined by the azimuth angle (here φ)). Similar to rotations in Cartesian coordinate systems, the direction in which azimuth and elevation / inclination are specified (clockwise as opposed to counter-clockwise) may differ in publications.

A geometrical representation of (x_i, y_i, z_i) as well as $(r_i, \theta_i, \varphi_i)$ for several points p_i is depicted in Figure 2.5. Note that the depicted point p_5 in Figure 2.5b illustrates the lower limit for azimuth angle $\theta \in [-\pi, \pi)$ while point p_6 marks the upper limit for coordinate φ of a spherical point $p = (r, \theta, \varphi)^T$. Furthermore, no points in Figure 2.5b may exist with a negative first coordinate, since $r \in \mathbb{R}^+$ needs to hold.

Cartesian coordinates can be mapped easily to spherical coordinates and vice versa by the following equations, if we use the conventions for both coordinate systems as stated above:

$$f_{\text{spherical}} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \sqrt{x^2+y^2+z^2} \\ \operatorname{atan2}(x, z) \\ -\operatorname{atan2}\left(y, \sqrt{x^2+z^2}\right) \end{pmatrix} = \begin{pmatrix} r \\ \theta \\ \varphi \end{pmatrix} \tag{2.1}$$

$$f_{\text{Cartesian}} \begin{pmatrix} r \\ \theta \\ \varphi \end{pmatrix} = \begin{pmatrix} r \cdot \sin\theta \cdot \cos\varphi \\ r \cdot \sin\varphi \\ r \cdot \cos\theta \cdot \cos\varphi \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.2}$$

where $\operatorname{atan2}(y, x)$ is defined as

$$\operatorname{atan2}(y, x) := \begin{cases} \tan^{-1}\left(\frac{y}{x}\right) & \text{if } x > 0 \\ \tan^{-1}\left(\frac{y}{x}\right) + \pi & \text{if } x < 0, y \geq 0 \\ \tan^{-1}\left(\frac{y}{x}\right) - \pi & \text{if } x < 0, y < 0 \\ +\frac{\pi}{2} & \text{if } x = 0, y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0, y < 0 \\ 0 & \text{if } x = 0, y = 0 \end{cases} \tag{2.3}$$

Note that for spherical coordinate systems that employ inclination rather than elevation equations (2.1) and (2.2) become

$$f_{\text{spherical}}\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}\sqrt{x^2+y^2+z^2}\\ \operatorname{atan2}(x,z)\\ \cos^{-1}\left(\frac{-y}{\sqrt{x^2+y^2+z^2}}\right)\end{pmatrix}=\begin{pmatrix}r\\\theta\\\varphi\end{pmatrix}$$

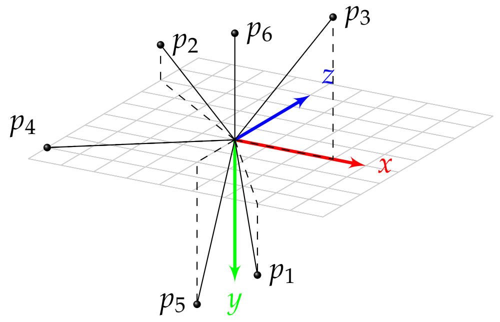
$$f_{\text{Cartesian}}\begin{pmatrix}r\\\theta\\\varphi\end{pmatrix}=\begin{pmatrix}r\cdot\sin\theta\cdot\sin\varphi\\ r\cdot\cos\varphi\\ r\cdot\cos\theta\cdot\sin\varphi\end{pmatrix}=\begin{pmatrix}x\\y\\z\end{pmatrix}$$

instead.

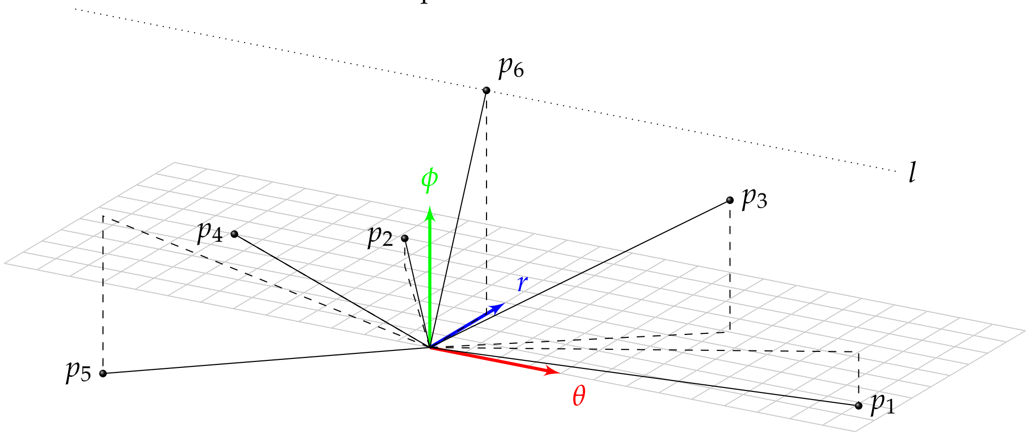
As mentioned earlier azimuth θ becomes arbitrary, for a fixed distance r and an elevation of $\varphi = \pm\pi/2$, and both elevation and azimuth become arbitrary if distance $r = 0$ holds in spherical coordinates. Therefore, equation (2.2) does not define a bijective function: if $r = 0$ holds, all three-tuples $(0, \theta, \varphi)$ are mapped to the Cartesian origin $O = (0, 0, 0)$ and if $\varphi = \pm\pi/2$ holds all tuples (r, θ, φ) are mapped onto $(0, r \cdot \sin\varphi, 0)$. However, for points measured by any (physical) sensor this is rather unimportant. Every sensor has a minimal measurement range and it is safe to assume that $r > 0$ holds – otherwise the sensor would have collided with the object it is supposed to measure and will not yield any information anymore. This removes the first ambiguity, while the second ambiguity just reflects that a physical sensor will usually get multiple sensor measurements for the point directly above (or below) the sensor, if they are in the sensor's field of view.

The main disadvantage of spherical coordinate systems is that data displayed in them is not as easily interpretable for a human observer, since they generally do not resemble the observed scene in a way humans are accustomed to. This is the main advantage of Cartesian coordinate systems which present 3D data in such a way that humans are able to recognize shapes and contours of the recorded environment, since Cartesian coordinates resemble the way we perceive our environments. To exemplify this, please refer to Figure 2.6, where some geometric primitives are plotted in Cartesian and spherical coordinates.

Due to the easier interpretation of data in Cartesian coordinates, the spherical coordinates gathered by a sensor are usually converted into Cartesian ones, sometimes already on the sensor itself. Nevertheless,there are certain problems where spherical coordinates provide an easier solution than Cartesian coordinates and vice versa: if we want to filter points according to their distance to the origin (i.e., usually the sensor position), we just have to compare a single coordinate, namely *range*, as opposed to first compute the distance for each point given in Cartesian coordinates via the first component of equation (2.1). On the other hand, if we want to extract all points that belong to a tabletop, i.e., a planar surface in Cartesian coordinates, from a dataset, doing so on Cartesian data is a more natural solution. This might become more evident, if shape and appearance of the six planar surfaces of the cube are considered in their spherical representation (Figure 2.6b). A common technique to do the latter will be briefly introduced in 2.3.3.



(a) Plot of several points in Cartesian coordinates



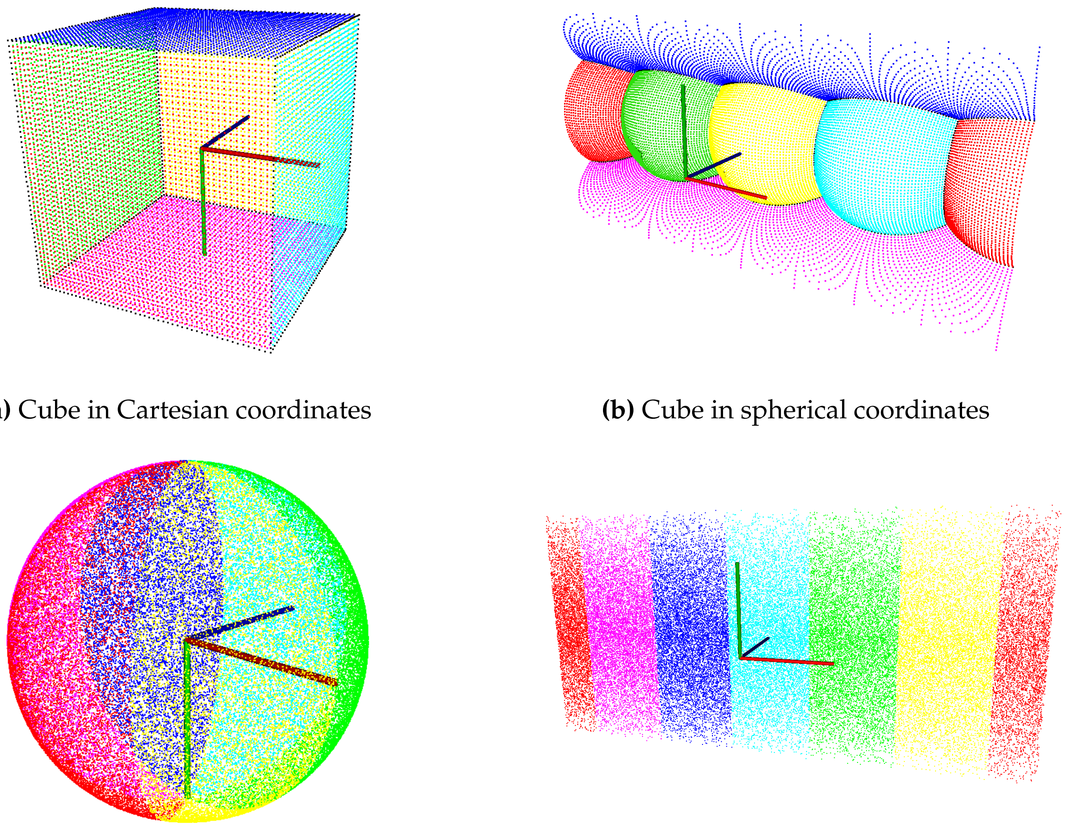
(b) Plot of several points in spherical coordinates

Figure 2.5: Comparison of Cartesian and spherical coordinates: **(a)** displays the Cartesian coordinates for points $p_1, p_2, \ldots, p_6$; **(b)** shows the corresponding spherical coordinates, angles θ_i and φ_i are plotted in radians. Note that the spherical coordinate system is left-handed, so that points with a positive elevation angle are displayed above the $r\theta$-plane, and that the r-axis is only positive. Furthermore, note, that p_6 could, in principle, be placed anywhere on dotted line l, since azimuth angle θ_6 becomes arbitrary for points place on the Cartesian y-axis. Conversion was done according to equations (2.1) and (2.2).

2.2 Points, Point Clouds & Poses

After section 2.1 illustrated differences in coordinate systems and established that generally the coordinate system shown in Figure 2.3 will be used in most contexts of this thesis, we will now take a closer look at the data, that will be embedded into such a coordinate system. Although Cartesian and spherical points have already been used they will be revisited in subsection 2.2.1, in order to describe what additional data might be associated with a single measurement points.

Measurement points are usually not recorded one at a time, but are bundled together into sets, which are commonly referred to as *point clouds* (subsection 2.2.2). The properties of point clouds are, of course, dependent on the individual properties of its point members, but in addition other attributes exist. In this context we will mainly distinguish between *organized* and *unorganized* point clouds, that will be introduced in 2.2.3 and combined point clouds that


(a) Cube in Cartesian coordinates

(b) Cube in spherical coordinates

(c) Sphere in Cartesian coordinates

(d) Sphere in spherical coordinates

Figure 2.6: Visualization of cube and sphere in Cartesian an spherical coordinats: **(a)** and **(c)** display Cartesian coordinates, while **(b)** and **(d)** show the same shapes in spherical coordinates. Cartesian coordinate system corresponds to the system introcuded earlier (see Figure 2.3), conversion from Cartesian to spherical coordinates was done according to equation (2.1). The spherical coordinate system is identical with the one used in Figure 2.5b.

contain the information of multiple sensor measurements (subsection 2.2.4).

A vital aspect when combining data, that is not acquired from the same position and orientation, is the *pose* from which a scene is observed and the relative pose of one measurement to another one. Without the consideration of this pose information, combining data recorded from different locations will, in the general case, result in an inconsistent representation of the recorded scene. A short excursion on this topic is provided by subsection 2.2.5 that brings this section about measurement data, its aggregation and its transformation to a close.

2.2.1 Points

Having established the space into which measurements are embedded into in section 2.1, we will now take a closer look at the actual data. As this thesis focuses on 3D data, or, as will be shown later, on particular absences of such data, the basic unit that will be considered in terms of measurements is a single (measurement) *point* p.

Depending on the employed sensor (see section 2.4) and possible post processing steps a point p can describe various kinds of information: for example, a point can hold information about the color of the surface the measurement was taken from or a local normal, describing the local surface curvature at point p. As mentioned earlier, when we take more than 1 point into consideration, as is usually the case, a subscript will be used to distinguish different points, e.g., p_i and p_j. If not mentioned otherwise, additional information is neglected in the context of this thesis and the point information is reduced to its minimum, namely the position of the point in the 3 dimensional Cartesian coordinate system introduced in section 2.1. Thus we can write a point p_i as a 3-tuple:

$$p_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{2.4}$$

with $x_i, y_i, z_i \in \mathbb{R}$. As mentioned in earlier in section 2.1 such a 3-tuple defines a unique location in $\mathbb{R}^3$, i.e., if two points $p_i = (x_i, y_i, z_i)^T$ and $p_j = \left(x_j, y_j, z_j\right)^T$ refer to the same location $x_i = x_j, y_i = y_j$ and $z_i = z_j$ have to hold. For convenience sake we denote points as column vectors, which allows for compact notation of point transformations via pose information.

As already implicitly conveyed by equation (2.4) point p_i is a point in the mathematical sense, i.e., it has no expanse. This is a simplification of the way actual sensors work – the signal usually spreads more the farther the distance to the measured surface, so that instead of taking measurements at distinct points the sensor gathers data from the surface inside a measurement spot (as a rule of thumb the distance to the closest surface inside such a spot is returned by the sensor).

Some sensor classes, like sonar sensors, are known for a wide spread of their measurement signal, i.e., the measurement spot becomes large quite fast. While this has disadvantages when trying to construct a precise model of the environment, since a sonar sensor tends to underestimate distances, it certainly has benefits in other use-cases: when using a sensor for collision avoidance the wide spread prevents the sensor from overlooking obstacles. Even if the distance is slightly underestimated a "better be safe than sorry" policy is often desirable. While the sensors which are described later in section 2.4 also do not produce distinct measurement points, in reality, the spread is usually smaller than that of sonar sensors, so that the simplification we employ does not distort reality too much.

If we ignore the setting of either a single or a bundle of sonar sensor, used for collision avoidance, single measurements usually do not convey much information. For many applications it is rather desirable to obtain a more or less dense representation of the environment in terms of surface measurements, because these allow for further analysis like segmentation or object recognition, which in turn might form the basis of more complex tasks like interaction via manipulation, path planning in occluded environments and the like. Therefore, a set of measurement points taken at the same time, or a short time interval, become interesting. This leads up to the next subsection which will deal with the aggregation of several measurements into a set.

2.2.2 Point Clouds

Such a set is commonly, and also in this thesis, referred to as a *point cloud*.While it is not guaranteed that a point cloud is a set in the mathematical sense, i.e., no point is present more than once in the point cloud, for most sensors the measurement principle prevents two points with identical coordinates from being captured in a single recording. However, the notion of many points, captured at approximately the same time and from the same position and orientation, aggregated into a point cloud urges us to extend our notation slightly. In the remainder of this thesis a point cloud P refers to a set of points, i.e.

$$P = \{p_0, p_1, \dots, p_n\} \tag{2.5}$$

while $\mid P \mid = n + 1$ denotes, as usual, the cardinality of the set, which in this case is the number of measurement points. An example of a point cloud, taken with a tilting 3D laser scanner is shown in Figure 2.7. The depicted point cloud was taken in a corridor and is visualized in Cartesian coordinates (Figures 2.7a and 2.7a) as well as in spherical coordinates (Figures 2.7c and 2.7c).

Oftentimes there will be more than one point cloud to consider at once, so in order to distinguish different point clouds they will be marked with subscripts (e.g. P_i and P_j). To show that a certain point p_k belongs to a certain point cloud P_i, i.e., $p_k \in P_i$, from now on such a point will also be marked by an additional subscript corresponding to the subscript of the point cloud, in our example we will use $p_{k,i}$, to denote the fact that p_k is an element of point cloud P_i. This notation is extended to the notion of a point and its coordinates from equation (2.4), if we need to make membership explicit for a point and the point cloud it belongs to. Therefore, $p_{k,i} = (x_{k,i}, y_{k,i}, z_{k,i})^T \in P_i$ holds.

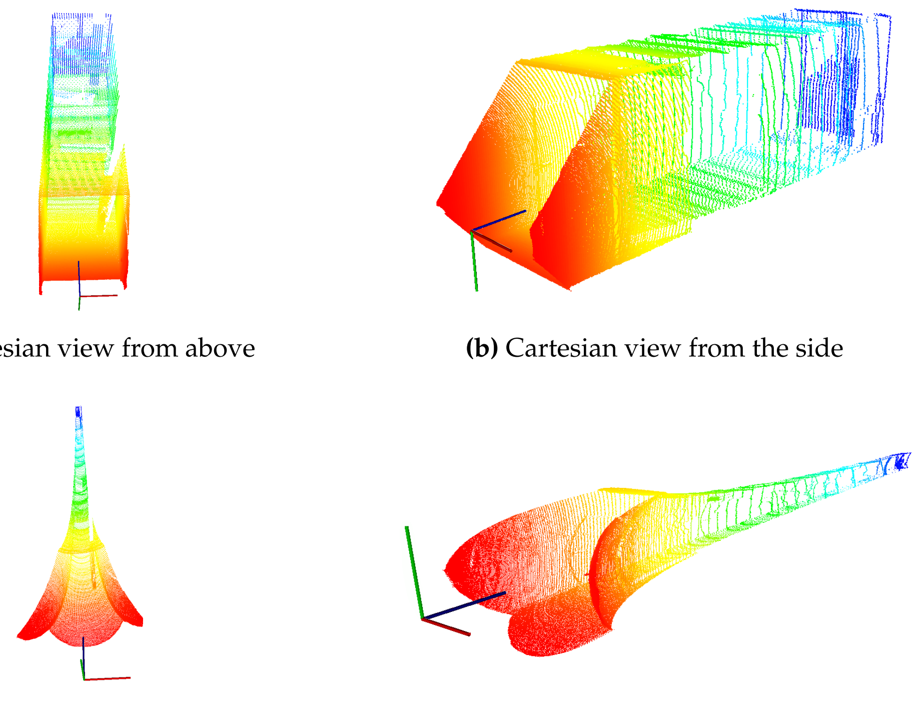
The 3D sensors that are considered in this thesis (see section 2.4), namely *laser scanner* and *RGB-D cameras* usually do not provide single measurement points, but an aggregation of measurement data. In case of RGB-D cameras such an aggregation is usually referred to as *frame*, while in the context of laser scanners it is commonly called *scan*. In the following these two terms are used somewhat interchangeably, if the type of sensor is not important. It is common practice to combine measurement points of a frame or scan into a single point cloud.

Associated with every scan or frame is a *pose* Θ, which describes the position and the orientation from where the data was gathered. Such a pose does not have do be denoted explicitly (and oftentimes is not in a stored point cloud): If no information about a pose is specified it is generally assumed that the position of the sensor was at the origin $O = (0,0,0)^T$ and that the viewing direction of the sensor coincides with the viewing axis of the used coordinate system. More information on poses will follow in subsection 2.2.5.

Having defined point clouds and established their notation the next subsection will introduce the notion of organized point clouds, that have desirable properties for some use-cases, but also their drawbacks.

2.2.3 Organized Point Clouds

Point clouds can furthermore be separated into two groups by a property that we have not discussed so far. There are *organized* and *unorganized* point clouds. In organized point clouds there



(a) Cartesian view from above **(b)** Cartesian view from the side

(c) Spherical view from above **(d)** Spherical view from the side

Figure 2.7: Point cloud resulting of a scan of a corridor using a tilting laser scanner (see section 2.4). Top row with **(a)** and **(b)** displays Cartesian coordinates of the measured data, while bottom row (**(c)** and **(d)**) show the corresponding spherical coordinates. Points are colored according to z-coordinate (**(a)** and **(b)**) and r-coordinate (**(c)** and **(d)**) respectively. Since the z-axis in **(a)** and **(b)** is aligned with the direction of the corridor points in similar colors in Cartesian and spherical view roughly correspond to the same region in the environment. This would be different if the z-axis were not aligned with the corridor.
A photograph of the recorded corridor can be found later in chapter 4, Figure 4.2.

is an underlying structure to the arrangement of points, usually in the form of a 2 dimensional grid. That means that an organized point cloud is not only a set in the mathematical sense, but also the order in which the points are stored in the point cloud becomes important. For data gathered from RGB-D cameras organized point clouds become a natural way to represent the associated 3D data.

Since there is a 1:1 mapping between each pixel in the 2D camera image of a RGB-D camera (see 2.4) and each 3D point, it is often beneficial to apply the 2D structure provided by the image width and height to the associated point cloud. In such an organized point cloud the points are stored row by row. That means if we have an image with a resolution of $w \times h$ pixels, the first point $p_{0,i}$ in the associated organized point cloud P_i corresponds to the upper left corner of the image, point $p_{w-1,i}$ corresponds to the upper right corner, while $p_{w\times(h-1),i}$ and $p_{w\times h-1,i}$ correspond to the lower left and lower right corners, respectively. This mapping allows for each point to be also addressed via 2D (image) coordinates (u, v) with $u \in [0, w-1]$, $v \in [0, h-1]$ instead of its index.

This way of addressing in organized point clouds has some advantages: for example, if we want to search for a nearest neighbor of a certain point $p_{k,i}$, with 2D coordinates (u, v), of organized point cloud P_i, as a heuristic it is very sensible to search not in the whole point cloud but only consider points which have image coordinates in the local neighborhood of (u, v), if the point cloud was recorded by a some projective sensor, i.e., a RGB-D or ToF Camera. This is generally a massive speedup, compared to searching the whole point cloud, but yields only an approximate result. If, for whatever reason, the depth of the point at coordinate (u, v) differs strongly from the points at neighboring coordinates the actually nearest neighbor in terms of Euclidean distance might not be part of the local neighborhood, but generally this heuristic approach yields good results. Other advantages, with respect to the detection of transparent objects, will be shown in more detail in section 3.2.

Also note the geometric information entailed in a point cloud is not influenced by the fact that the point cloud is organized: an unorganized point cloud P_j which contains the same points as organized point cloud P_i, but in a randomized order, looks exactly the same, if displayed in some viewing application. Since its points do not feature a natural, meaningful local neighborhood, a heuristic nearest neighbor search, as sketched above, is not applicable, due to the lack of underlying structure. However, if we apply for example an exact nearest neighbor search (and not some heuristic approximation) in P_j and P_i for the same point (i.e., $(x_{k,i}, y_{k,i}, z_{k,i})^T = (x_{l,j}, y_{l,j}, z_{l,j})^T$) the same point in terms of 3D coordinates will be returned from both point clouds.

While organized point clouds are a natural way to represent point clouds obtained by a RGB-D frame or a ToF camera, from which we also obtain depth images, the recordings of most 3D laser scanners can also be stored in an organized way. If measurements are gathered in a fixed order in a plane that is rotated around one of the axis of the sensor's local coordinate system the data has a similar 2D grid structure as a camera image. For scanners with a horizontal opening angle of 360 $^\circ$ the "grid" gets bent, so that it resembles rather a cylinder instead of a 2D grid. That means the that local neighborhood for points where the first grid coordinate is close to 0 contains points with grid coordinates close to w (if the dimensions are $w \times h$) and vice versa (think about glueing both side edges of the 2D grid together). However, this only works, when the measurement points combined in a single scan are always gathered in the same order – otherwise the 2D grid structure is not obtainable. This also holds true for stored collections of points, when they do not reflect the order in which they were gathered, e.g., when points are stored in ascending distance to the sensor.

The advantages of an organized point cloud come with a certain cost. Each organized point cloud from the same sensor needs to contain the same number of measurement points, regardless of the observed scene. Some scenes might contain many invalid measurements, due to the sensor's range limitations (think of a scene with contains a large portion of sky). While invalid measurements are omitted in the case of unorganized point clouds, an organized point cloud needs to store these points and mark them as invalid (for example by setting their coordinates to `NaN` or some similar value). In the following such points will also be referred to as `NaN`-measurements or `NaN`-points. Also if some filtering is applied to the points in an organized point cloud, for example an outlier removal, and the point clouds is supposed to retain its organized property, "removed" points need to be marked as invalid, instead of being really removed from the point cloud. Therefore, unorganized point clouds are more efficient in terms of used memory, when compared to organized point clouds.

Depending on the desired operation that one wants to apply to a point cloud organized point clouds with their `NaN`-values might also be undesirable, not only from a storage point of view: if several filters are applied sequentially to a point cloud that does not retain its organized property, the size of the point cloud typically decreases, the more filters have been applied. This usually also means that some methods will work faster, since they do not have to consider as many input data. In the case of an organized point cloud, the number of points will remain the same after each filtering step. Even though the number of valid data points, i.e., non `NaN`-points, in the point cloud decreases, this typically needs to be explicitly checked for each point before performing any operation that requires a point coordinate as input, thus the speed-up will be less than in the unorganized case.

Note that other forms of creating an underlying organization of the point cloud data can become useful for search tasks, even if the point cloud was not recorded by a projective sensor. For instance Behley et al. [7] report that their index based organization of data point in octants of an octree provided a significant speedup for nearest neighbor searches.

2.2.4 Combined Point Clouds

While it is common to aggregate the measurements obtained from a single scan into a point cloud, point clouds do not necessarily contain only points from a single scan. Having more than one scan combined into a single point cloud can have several reasons. If the employed sensor provides noisy measurements, then recording multiple frames, without changing the sensor's position or orientation, and storing them into a single point cloud may allow for easier outlier identification. In this case the combination of point clouds is identical to their union, i.e., the combined point cloud of P_1 and P_2 is defined as

$$P_1 \cup P_2 \tag{2.6}$$

Another reason for combining data of several point clouds might be to capture the complete geometric properties of an object. In this case it is usually necessary to record the object in question from different perspectives. However, just performing the union operation on both point clouds will generally not provide the desired results, since the associated data points are embedded in the individual coordinate systems and not into a shared coordinate system. Figure 2.8 illustrates this, with Figure 2.8c showing the effect of combining point clouds according to equation (2.6), even though they were recorded from different perspectives.

With the correct pose information to align the measurements, the complete object can be stored into a combined point cloud, made up from (transformed) measurements of multiple point clouds. If it is important to mark which points in a combined point cloud originated from which scan, points can be easily extended by a label, as an additional attribute. Labeling can be done in a similar fashion as color values or surface normals may be stored to enrich point information. Data points that share the same label are recorded in the same frame. In order to be able to perform data analyses that need viewpoint information, it is sensible to bundle the pose information, used to align the individual point clouds, alongside the combined point cloud.

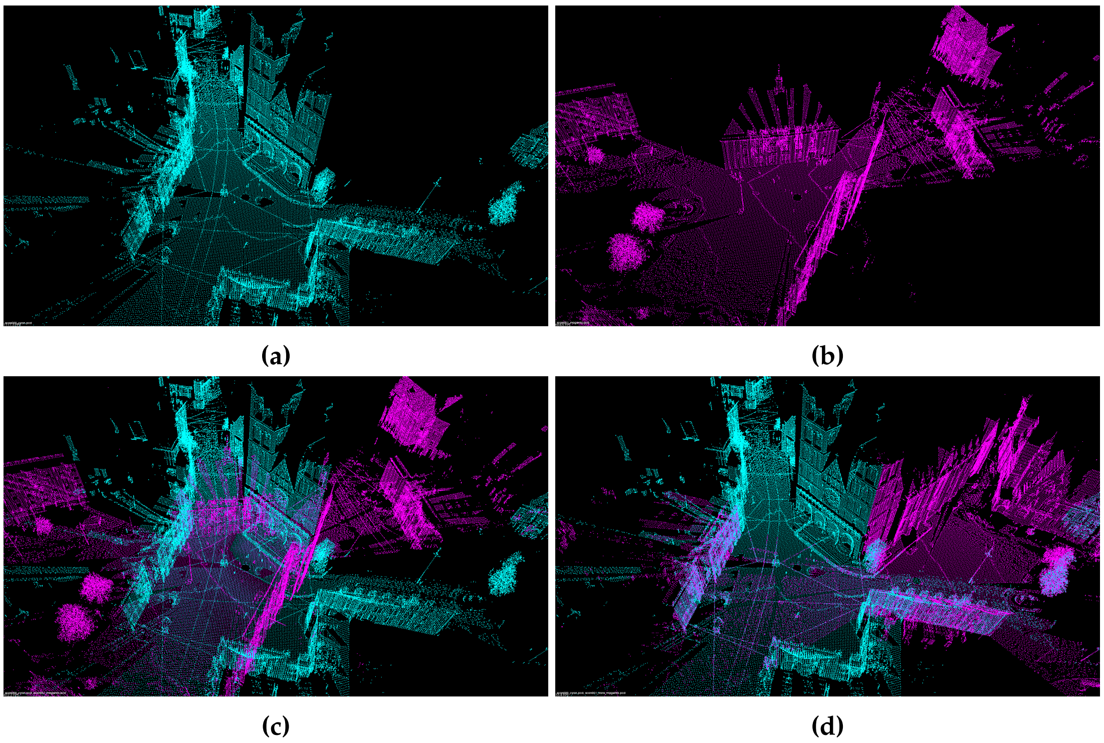


Figure 2.8: Combining point clouds: **(a)** and **(b)** show individual point clouds of a city scene. **(c)** shows the combination of P_1 (cyan) and P_2 (magenta) without embedding the points of P_2 into the coordinate frame of P_1. In **(d)** the pose differences between the sensors are considered before combining the data.

If viewpoint and origin of the recorded data are unnecessary in the application scenario, following the data collection, it is sensible not to store which points were recorded in the same frame and omit the pose information used for point cloud alignment. For instance, when just the geometric properties of a small object are necessary for grasp planning with a robotic arm, in general it is unimportant to know which point on the object's surface was recorded from which sensor pose. Unsurprisingly, once this information is omitted, it becomes in general impossible to infer which points were recorded in the same frame and from which pose these points were measured.

Combining point clouds destroys the organized property, regardless of the input point clouds all being organized or adding labels to mark which points were originally part of the same point cloud: since the alignment of point clouds is done via affine transformations and the aligned point clouds usually overlap, for the resultant combined point cloud the underlying 2D grid structure gets lost.

2.2.5 Poses

In the context of combined point clouds (2.2.4) it was already mentioned that we need to use pose information to align several individual point clouds in a common reference frame and

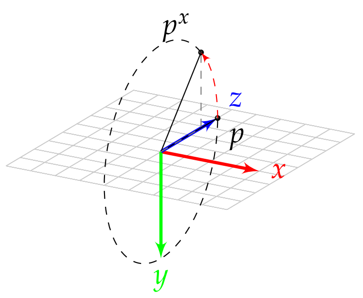
Figure 2.8c illustrated the outcome of combining point clouds while disregarding pose information. Depending on the use-case this reference coordinate system can simply be defined by the sensor for the first captured frame. In other applications, for instance when we think of a mobile robot, featuring multiple sensors, it will often be more convenient to transform all gathered data in a common coordinate frame that is related to center of the robot, so that subsequent tasks like sensor fusion, obstacle avoidance and path planning can take place in this common coordinate system. In this case it is quite common to use the initial position of the robot's center as the origin of the reference coordinate system, which is sometimes referred to as *world coordinate system*.

It is also not uncommon for the reference coordinate system to feature an arrangement of the coordinate axes different from the (local) coordinate systems of individual sensors. ROS, by convention, uses a right-handed coordinate system for its mobile robots, where the x-axis corresponds to "forward", the y-axis points "left" and z-axis "up", while a RGB-D camera in ROS employs the same coordinate system used in this thesis. *Pose information* or *poses* Θ for short are used to transform data points from one coordinate system into another one. This transformation can be applied to individual data points or on complete point clouds. In the latter case it just means that each point $p_{j,i} \in P_i$ is subject to the transformation. If the difference between two sensor poses is known, this knowledge can be used to embed points from the coordinate frame of the associated sensor into the coordinate frame of the other point cloud.

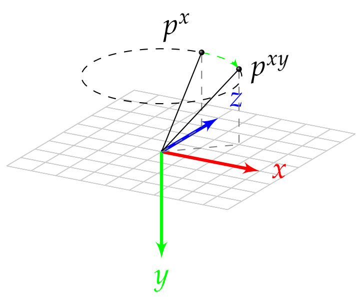
In the case of two frames that observe the same scene from different viewpoints with some overlap, that means that the gathered data in the overlap region should be arranged in such a way that the combined data correctly resembles the environment, e.g., if a building is observed in the first frame from one side and in the second frame a different side is recorded, the desired pose information describes how to position and orient the second frame in relation to the first, so that the walls of the complete real-world building are resembled in the combined measurement data. Figure 2.8d provides an example of such an embedding.

In the case of the mobile robot, pose information is needed to know how the gathered data, which is taken in relation to a sensor's origin, is positioned in relation to the robot. Then it becomes possible to know where obstacles are located relative to the robot and not to the sensor, an essential piece of information even for the most basic navigation tasks. Since this thesis does not focus on how to gather 3D data (mobile robot, tripod-mounted scanners, etc.), the latter use-case for pose information is ignored in the following. Instead the term pose will be used to align several frames in a common reference coordinate system. However, up until now we have not formally defined what a pose actually constitutes and how we can express it in a compact and concise way.

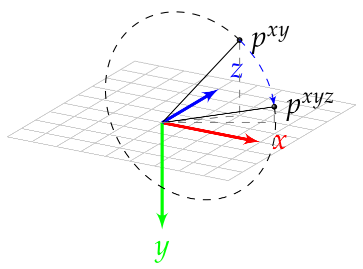
Relative to a given origin of a 3D Cartesian coordinate system there are 6 degrees of freedom (DoF) concerning position and orientation to define the relation of a viewpoint with respect to the origin, namely position (3 DoF) and the orientation (3 DoF). The position with respect to a common reference coordinate system is simply given as a 3D point, which denotes the location of the sensor, when it gathered the data, relative to the origin of the reference coordinate system. The orientation part of the pose information also has 3 DoF, which can be seen as Euler angles, i.e., rotations around the three coordinate axes. Specifying the orientation in Euler angles seems to be intuitive at first glance, but this form of notation has some shortcomings, which will be explained in the following.



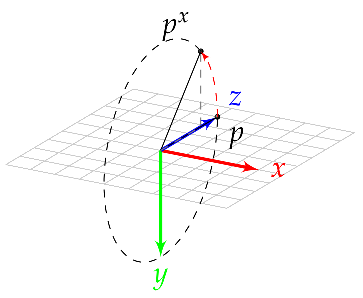
(a) Rotation around x-axis



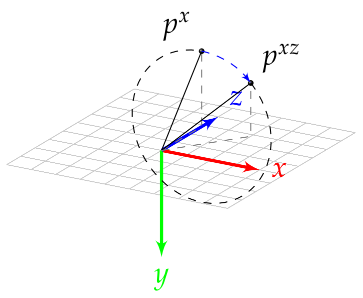
(b) Rotation around y-axis



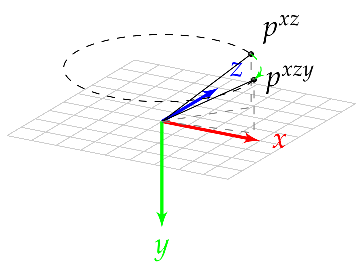
(c) Rotation around z-axis



(d) Rotation around x-axis



(e) Rotation around z-axis



(f) Rotation around y-axis

Figure 2.9: Different outcomes for changed order of rotation. Rotations are 45 ° around each coordinate axis, superscript of points denotes the order in which the rotations were performed on the original point $p = (0, 0, 1)^T$. Top row **(a – c)** shows the result to the rotations applied in order x, y and last z. Bottom row **(d – f)** displays the result if the order of rotation around y and z axis are swapped.

Order of Rotations

As already mentioned in section 2.1 rotations are not commutative, i.e., the order in which they are applied influences the overall outcome. An illustration of this fact is provided by Figure 2.9, where point $(0, 0, 1)^T$ is rotated by 45 ° around each coordinate axis, but rotations are applied in 2 different orders. Since Euler angles are defined as pure rotations around the coordinate axes there needs to be a convention specifying in which order the rotations are applied. For the 3 coordinate axes we arrive at 6 different possibilities (i.e., the permutations) in which order the rotations can be applied. However, a rotation around one axis can also be described as a sequence of (up to three) rotations around the other two axes, so that instead of just looking at the possible permutations involving all three axes, we also have to consider the permutations that can be created by a sequence of three rotations around two different axes (where no subsequent rotations around the same axis are allowed) - which leads to a total of 12 different sequences that are possible to denote a rotation in 3D, given by Euler angles. That in itself is not a large problem – we just have to make sure that we pick one of the 12 possible sequences and apply the rotations every time in the exact same order. However, when collaborating with others and / or using third-party source code, it becomes rather tedious to always check if the same sequence of rotations is used and introduces an additional source of errors.

Denoting rotations in Euler angles can also lead to *Gimbal locks* in certain configurations, where one of the originally 3 DoF is lost. Furthermore, it may be difficult to determine how much change a rotation, denoted in Euler angles, will induce on a point, since there exist combinations where the angles are quite wide, but the resulting rotation is rather small. Taking also into account that there is no meaningful notation for how to combine several rotations, denoted by Euler angles, they are a rather poor choice of representation. One possible way to address these issues is by using quaternions to denote rotations (a very nice overview about their properties in this context can be found in [70]).

Quaternions provide a unique and very compact notation of rotations in 3D, that does not exhibit the inherent problems of Euler angles. However, they are not sufficient to denote a pose, since they only encode the rotational aspect, but not the translational information. A 3×3 rotation matrix retains the desired properties of the quaternion, but is a less compact way to represent the information. However, the matrix representation can be extended to a 4×4 matrix, that incorporates translation information as well. Such a transformation matrix is given as

$$\Theta = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.7}$$

where the 3×3 matrix in the upper left (with entries $r_{1,1} - r_{3,3}$ denotes the 3D rotation and translation information is provided by entries t_x, t_y and t_z.

Extending the Cartesian points, as defined in equation (2.4), to become a four dimensional column vector, where point information is padded with an additional 1 entry, i.e., $(x, y, z)^T$ becomes $(x, y, z, 1)^T$, allows us to apply the effect of a transformation as a simple matrix multiplication. If we denote the transformation of a point $p_{j,i}$ as $\tilde{p}_{j,i}$ the latter is defined as:

$$\tilde{p}_{j,i} = \Theta \cdot p_{j,i} \tag{2.8}$$

If we use the term $\Theta \cdot P_i$ as a shorthand for applying equation (2.8) on all points $p_{j,i} \in P_i$ a transformed point cloud can be defined in an analog fashion as:

$$\tilde{P}_i = \Theta \cdot P_i \tag{2.9}$$

Now we have established how individual points or complete point clouds can be transformed, provided some pose Θ. How to obtain a specific pose $\Theta_{2 \mapsto 1}$, that for instance maps P_2 correctly into the coordinate frame of P_1, has not been discussed so far and is generally a non-trivial problem. One heuristic approach to do so will be introduced as part of the next section.

2.3 Common 3D Data Processing Techniques

This subsection will briefly introduce and discuss a small selection of 3D data processing algorithms that will be used more or less extensively in the later parts of this work. While these algorithms belong to the set of commonly used tools, when working with 3D point clouds a brief description is still worthwhile, since their understanding is beneficial when discussing the workflow of the approaches introduced in chapters 3 and 4.

First subsection 2.3.1 briefly discusses the well known and often applied *ICP Algorithm* and in which scenarios it is applied. This is followed by a short digression about *octrees* and *voxelgrids* that can be used to filter point clouds and allow for rather efficient ray tracing in point cloud data. The final part of this very basic overview of 3D data processing techniques introduces *plane detection* in 3D point clouds, which often serves as one of the first filtering steps to distinguish background from "interesting" objects in recorded 3D data. Examples for this approach can be found in [4, 43, 49, 52, 66].

2.3.1 The ICP Algorithm

When working with point cloud data, recorded from multiple positions, a common problem is the correct alignment of the recorded data. As mentioned earlier, usually a point cloud is associated with the sensor's coordinate system, i.e., the sensor's position coincides with the origin of the coordinate system. Transformation of a point cloud can be done as described by equations (2.8) and (2.9). The problem that needs to be addressed now is how to find the correct transformation to embed one point cloud into the coordinate frame of another one, if we want to create a coherent representation of the recorded environment.

In the following it will be assumed that only two point clouds P_1 and P_2 need to be aligned, but the general considerations can be extended to the alignment of multiple point clouds. Several methods exist to obtain a pose Θ that is as close as possible to "ground truth", meaning correct alignment in this case. One common approach is to distribute several markers in the recorded scene and then manually identify and mark the corresponding markers in both scans. If at least 3 non-collinear correspondences between both point clouds are given a rotation and translation can be specified that map these points, and in connection the whole point clouds, onto each other. While distributing artificial markers in the environment and manually picking the correspondences between scans is a practical solution that is used in a lot of contexts, like surveying, this task becomes very tedious and time consuming, if we consider a series of (several) hundred (small) scans, as obtained in the context of mobile robotics.

Nevertheless, the marker approach serves to illustrate one of the requirements: if scan registration is to be performed successfully, there needs to be sufficient overlap in the recorded data. Markers only serve a purpose when they are recorded in both scans, while markers that are only recorded from one sensor pose do not provide any information about the correct alignment of both scans. If no markers or other means are used to establish known correspondences between points from P_1 and P_2, the problem becomes more challenging, since we now also have to find some heuristic that tells us which points from the two point clouds might be associated.

To tackle this problem the *Iterative Closest Points* algorithm [8], commonly and in the following abbreviated as ICP, has been employed with great success. One requirement that ICP needs is an initial pose estimate Θ_{init} between the two point clouds P_1 and P_2. This initial estimate will serve as an initial alignment that is iteratively refined by the algorithm. Naturally specifying these initial alignments by hand can also be tedious work, so that, when possible, already available information is used to generate these alignments. In the context of mobile robotics the pose estimation obtained by odometry, or its refinements with the help of gyroscopes [73], are readily available and usually provide an initial guess of sufficient quality.

Provided this initial guess Θ_{init}, ICP proceeds to improve the alignment of both point clouds, based on heuristic correspondences, in an iterative fashion, hence the *I* in ICP. Since the mathematical details of ICP are not a prerequisite, but having a notion about its general workings is, for the remainder of this thesis I will omit a formal representation of ICP here. The interested reader will find all details in the original paper by Besl and McKay [8], which is wholeheartedly recommended, and for a step-by-step derivation of how translation and rotation in each iteration can be considered independently, thus simplifying the problem, the explanations in [37] may prove insightful.

Assuming that we have a sufficiently good initial pose estimate Θ_{init} between two scans, the ICP algorithm starts to create correspondences between them. The fixed scan (P_1) is in this case also often referred to as the *model* while P_2 that will be aligned to P_1 is called *data*. During the complete process the model will not be modified, but the pose, starting with Θ_{init}, to embed P_2 (hopefully correctly) into the coordinate system of P_1 will be changed in each iteration, until the algorithm converges. First the initial pose estimation Θ_{init} will be used in order to transform "data" point cloud P_2, i.e., $\tilde{P}_2 = \Theta_{\text{init}} \cdot P_2$ is computed, which constitutes the rough initial alignment of P_2 with P_1.

Automatic detection of unambiguous correspondences between two point clouds P_1 and P_2 is still an open problem, although 3D feature points (like [6, 44, 65, 69] for instance) have made recently progress in this direction. Instead of working with a few, potentially wrong or ambiguous correspondences, ICP proceeds a different way in order to refine the initial alignment Θ_{init}.

ICP tackles the lack of precise knowledge about correspondences between P_1 and P_2 by automatically constructing a large number of possible ad-hoc correspondences, computing the best alignment under the assumption of these correspondences. This process is iterated, and correspondences are reestablished after each alignment step, hence I referred to them "ad-hoc" correspondences.

Correspondences between both point clouds are established on the basis of individual points and as the heuristic to establish them the Euclidean distance is employed. That means that the ICP algorithm determines in each iteration for each point $\tilde{p}_{j,2} \in \tilde{P}_2$ the point $p_{i,1}$ that is closest in point cloud P_1. In the first iteration $\tilde{P}_2$ is composed of $\Theta_{\text{init}} \cdot P_2$, while in following iterations Θ_{init} is replaced by Θ_{curr}, the current embedding of P_2 into the coordinate system of P_1. Note that this mapping from points $\tilde{p}_{j,2}$ to their closest counterpart $p_{i,1}$ is not a $1:1$ mapping, since multiple points $\tilde{p}_{j,2}$ might have the same point $p_{i,1}$ as the closest point in P_1. These point correspondences, based on the closest distance, are responsible for the last two letters of the algorithm's name.

Knowing that the point-to-point correspondences are established according to their Euclidean distance, discarding the correspondences in the next iteration step is a prudent choice: if the alignment, i.e., the translation and rotation that P_2 is subjected to, is changed between two iterations (as it usually is), a different point $p_{k,1}$ might now be closest to a point $\tilde{p}_{j,2}$. Note that the actual set of Cartesian points that $\tilde{p}_{j,2}$ describes will change during the ICP algorithm, since it is dependent on the current pose Θ_{curr}, that changes while the algorithm proceeds.

Once the point-to-point correspondences in an iteration step are established, they are stored in pairs and this information is used to minimize the *average error* e_{avg} between them, by performing a rotation and a translation. The average error is identical with the average distance between the points in a pair – if we assume that they refer to the same point in the scene, this distance can be seen as an error in the alignment. Details on how the minimization if performed can be found in [8, 37].

ICP is guaranteed to converge, but due to its heuristic nature it is not guaranteed to reach a global, but only a local minimum of error e_{avg} and the obtained result is dependent on the quality of the initial guess. The reason for this is easy to see. The simple heuristic for the point correspondences, basically assumes that both point clouds are (almost) correctly aligned. Under this assumption it is sensible to expect that the closest point $p_{i,1} \in P_1$ for a point $\tilde{p}_{j,2} \in P_2$ refers to almost the same scene point. Inaccuracies and errors that are made for some of these expectations are compensated by the mass of the other point pairs and in a lot of cases this simple heuristic performs surprisingly well.

However, depending on the amount of overlap in the recorded data of P_1 and P_2 and the initial guess, the resultant local minimum might be wrong, if for a majority of points wrong correspondences are established (wrong in the sense that the associated measured surfaces in the real environment do not correspond). For example if the scene contains two walls that meet at a small angle, and each point cloud contains only measurement points of a single wall, correspondences between both segments might be established and gradually lead to both wall segments coinciding. While ICP might converge in this local minimum it does not, in this case, reflect the correct alignment of both point clouds. Such cases can oftentimes be avoided when the recorded data features more overlap, i.e., one point cloud contains measurement points from both wall segments.

Until now we have only considered that two point clouds P_1 and P_2 will be registered via ICP. If more point clouds are recorded, the algorithm can be applied in a sequential fashion, i.e., first P_2 is aligned with P_1 and afterwards ICP is executed again, but this time $\tilde{P}_2$ (here denoting the final alignment after the first termination of ICP) is now considered the model, while P_3 constitutes the data that needs to be fit to the model. This process can be repeated, so that a series of point clouds that were recorded one after another can be registered into one large representation of the environment.

In the case of multiple point clouds that need to be aligned, another alternative to the sequential pairwise application of ICP exists, namely matching the latest scan against all previously registered predecessors (or a subset of them), hence the size of the model (in terms of data points) increases after a point cloud is aligned, because it will subsequently be added to the model. While this approach consumes more memory and is also more time consuming, when trying to determine the closest point relationships, the alignments might improve, since more data and therefore also potentially more overlap with regard to the new data are present in the model. This approach is sometimes also referred to as *meta-scan matching*.

2.3.2 Octrees and VoxelGrids

When working with 3D data it is often prudent to filter the data provided from the sensor. Various filtering techniques like median filtering of neighboring measurements can remove outliers from valid measurements. Other techniques can be used to try to compensate for systematic sensor noise.

To a certain extent similar results can be achieved by creating an *octree* or *voxel grid* representation of a gathered 3D point cloud. However, these two related techniques can also be used to compress the point cloud, albeit at a certain loss of precision. Loss of precision, especially when employing high-end 3D laser scanners, is not necessarily as bad as it might initially sound. Oftentimes the recorded point clouds consist of several millions of points, with a very high point density in the close vicinity of the sensor.

Depending on the intended use, such an abundance of measurements, gathered in a small volume, might not be needed for the intended purpose and might actually hinder the workings of some algorithms. If you for example think about the ICP algorithm sketched in 2.3.1, the high density of measurement points near the sensor implicitly increases the importance of this part of the scan in the registration process, due to the large number of point correspondences. Other parts of the point cloud, that contain less points will have a smaller influence of the computed pose correction in each ICP iteration. This may result in pose corrections where the overall alignment of P_1 and P_2 may actually become worse, if the point-to-point distance for those parts of P_2 that have a high point density decreases, resulting in an overall smaller e_{avg}.

An example of the latter case might be a hallway scene, where a wall at the end of the corridor can in principle be used to indicate a correct translation between two point clouds P_1 and P_2. If we assume that both P_1 and P_2 were recorded in relatively close proximity and with a relatively large distance to the end of the corridor both clouds feature a high point density near their origin, while the point density at the end of the corridor is lower. Since the distinct "features" that are available in both point clouds, i.e., the measurement points of the wall at the end of the corridor in P_1 and P_2, constitute only a relatively small number of the overall points, it can happen that ICP translates the estimated position of point cloud P_2 towards the origin of P_1, because this increases the number of point-to-point correspondences. Note that features in this sense are not meant in terms of the ICP algorithm, where all point pairs contribute equally to the error minimization, but instead in the sense that these are prominent regions that a human observer can easily identify when trying to assess if the alignment of the point clouds is sensible.

Assuming a corridor that does not contain other distinct geometric features, this increase of point-to-point correspondences by underestimating the distance between P_1 and P_2 will not necessarily increase e_{avg}, the average error. This is due to scene ambiguity, if we only consider a small region, when moving along the corridor's main direction. If P_1 and P_2 featured a more even point distribution, the influence of the point on the wall at the end of the corridor would gain a greater influence.

Apart from the increased influence of areas with high point density on the outcome of ICP, a reduction of points might also be desired due to purely practical reasons, when running ICP. The most time consuming step in each iteration is to determine which point $p_{j,1} \in P_1$ is closest to point $\tilde{p}_{i,2} \in \tilde{P}_2$. Several techniques exist to improve the time needed to perform this search, but all solutions have one thing in common: the larger the number of points in P_1, the longer it takes to find the nearest neighbor. Hence, by reducing the number of points in P_1, the time for a single search is decreased and by reducing the number of points in $\tilde{P}_2$ the number of queries decreases, thus resulting in an overall speed-up of the algorithm.

In these cases one possible solution comes in the form of voxel grids or octrees, their more memory efficient cousins. Both data structures provide a discretization of the recorded data points via a 3D grid, where the volume of the scene is partitioned into cubic volumes (instead of cubes a partitioning into cuboids is also possible, but rarely performed – therefore in the following we will assume the usage of cubes). The size of these cubes needs to be defined before the construction, usually this is done by the parameter l_L, which specifies the length of one side of such a cube. In the case of octrees, parameter l_L is sometimes also referred to as the *resolution* of an octree. Note that there also exist methods to partition space into primitives that are neither cuboids or cubes, as detailed by Ryde and Brüning in [67], but these approaches are much less employed.

While both, voxel grids and octrees, partition 3D space into small and discrete volumes, the idea behind them is slightly different. Usually a voxel grid is aligned with some previously defined 3D coordinate system, before determining which data point will be represented by which cube. Alignment in this case means that the edges of each cube are parallel to one of the coordinate axes. Furthermore, the voxel grid is a dense structure, i.e., for each cell in the voxel grid exactly 26 neighbors exist (8 cubes surrounding the current cell on its level and 9 cubes above and below, respectively). This arrangement allows referencing a grid cell $v_i^g = \left(x_i^g, y_i^g, z_i^g\right)^T$ by the three integer coordinates $x_i^g, y_i^g, z_i^g \in \mathbb{Z}$, where superscript "g" marks them as grid coordinates.

While in theory the grid may be infinite, in practice it will be restricted to a size that can contain the bounding box associated with the recorded data. As mentioned above, inside this bounding box the voxel grid is dense, i.e., cells exist no matter if they contain measurement points or not. Grid cells that are located on the edges of the constructed grid are exceptions from the general rule, since they will not have the complete set of 26 neighbors. Once the bounding box of a point cloud is known, the appearance of the grid is fixed, given the size of the grid cells, i.e., the number of points and their spatial arrangement in the point cloud has no further influence on the grid structure. Furthermore, note that while the grid structure is independent of the data, after determining the bounding box of the point cloud, the output when using it as a filter, i.e., which cell is occupied and which cell is not, is naturally dependent on the gathered measurement points.

Having defined this partition, each measurement point can be associated with the grid cell in which it is located. When employing the voxel grid as a means to reduce the recorded measurement data, each grid may typically produce up to one point in the reduced point cloud. Whether the contents of a grid cell should be represented by a point or not depends on the number of measurement points contained in the grid cell. If this number is smaller than a given threshold $n_{\min} \in \mathbb{N}$, with $n_{\min} \geq 1$ then the cell is considered as empty and will not be represented by any point in the reduced point cloud. Otherwise a single point will be used to denote the contents of the grid cell. Typical choices for this point are the center point of a grid cell or the center of gravity of all points contained in the grid cell. The former method will provide a very regular output and can be computed faster, while the latter better represents the unreduced point cloud data and will usually produce a less grid-like output.

While octrees also serve as a means of discretization for available data, their construction differs from voxel grids. Voxel grids are, as described above, usually aligned beforehand with some coordinate system, so that the position of each grid cell is fixed, before determining which 3D points belongs to which grid cell. Octrees are, on the other hand, recursive structures that fit themselves to the available data. Having set the size of an octree leaf L_i, i.e. the equivalent to a grid cell in the voxel grid, via parameter l_L, the octree first proceeds to analyze the overall point cloud data and its dimensions. Note that the orientation of these leaves is also aligned with the underlying coordinate system, that contains the point cloud data, but their position is not predetermined. The former property is shared with the grid cells of the voxel grid, while the latter constitutes a difference. When the dimensions of the total point cloud have been determined, i.e., a bounding box that encloses the point cloud as a whole is defined, the algorithm to create an octree proceeds with determining the smallest cube that has an edge length of the form $2^k \cdot l_L$ with some $k \in \mathbb{N}^+$, that is able to contain this bounding box.

This cube constitutes the root node of the octree and its center usually coincides with the center of the enclosed bounding box of the point cloud. When dimension and position of the root node is established, the creation of an octree proceeds by partitioning the cube into 8 cubes that feature a side length of $2^{k-1} \cdot l_L$. Any child node that contains measurement points is then recursively divided into 8 cubes with half their edge length again, until the minimal partitioning size of l_L is reached. Whenever a cube does not contain any measurement points, its subdivision is aborted. This way, volumes that do not contain any data points will not be represented on an (unnecessarily) fine scale, but can be marked as (potentially) large empty areas. Note that this criterion, as when determining if a grid cell in the voxel grid should be represented by a point in the filtered cloud, can be relaxed by specifying parameter $n_{\min}$that provides the minimal number of points that a node in the octree needs to contain in order to be considered non-empty. Instead of aligning the root node of the octree with the center of the bounding box, other possibilities like the center of gravity of the point cloud or some predetermined coordinate exist. In these cases it is possible that the side length of the root node's volume has to be increased from its minimal size $2^k \cdot l_L$, depending on the spatial arrangement of the root node in relation to the point data.

A visual representation of a tree structure for an octree and its volumetric appearance is shown in Figure 2.10. The bottom row (Figures 2.10b – 2.10d) shows how the partitioning of the represented space proceeds, up to the level shown in the tree representation in Figure 2.10a. Note that in the example depicted in Figure 2.10 measurement point can at most be contained in sub-volumes labeled as ② or ⑧. All other nodes on this level (①, ③ – ⑦), cannot contain measurements, since the presence of measurements would lead to further subdivision.

Furthermore, at least one of the children of ② and ⑧ has to contain measurement points – otherwise the subdivisions of the root node, i.e., Figures 2.10c and 2.10d would not have occurred. Depending on the choice of parameter l_L, the nodes on level 3 of the depicted tree in Figure 2.10a, i.e., the nodes labeled $\textcircled{\scriptsize 2.1}$ – $\textcircled{\scriptsize 2.8}$ and $\textcircled{\scriptsize 8.1}$ – $\textcircled{\scriptsize 8.8}$, might either be leaves of the octree of be subdivided again in 8 volumes of equal size.

If we assume, for a moment, that the nodes on level 3 already correspond to the desired resolution of the partitioning, i.e., these nodes are octree leaves, then we can conclude that each of

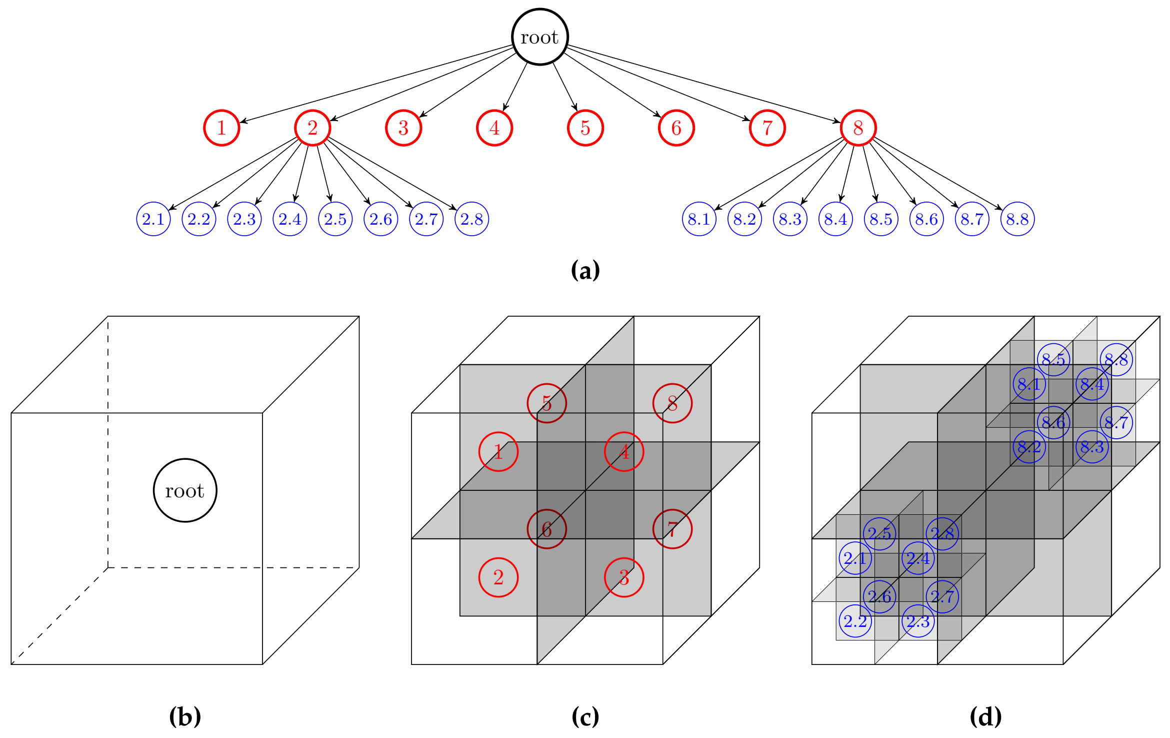


Figure 2.10: Octree visualization: Top row **(a)** shows the octree structure in its tree representation while the bottom row **(b)** – **(d)** depicts the geometric equivalent.

these leaves contains at least 1 measurement point – otherwise the octree algorithm would not have generated a leaf node at this position, but left it empty. When we compare this partition, under the previous assumption, with the equivalent voxel grid, the smaller number of nodes needed to contain the measurement information already becomes evident: in the octree representation a total of 25 nodes is needed (1 root node, 8 nodes on level 2 and 16 nodes on level three). The voxel grid, foregoing the tree approach for structuring the subdivisions does not contain any equivalents to the nodes above level 3, but populates the complete volume with cubes of the same size as an octree leaf, leading to a total of 64 grid cells.

This difference between the needed number of grid cells and the leaves an octree needs to represent the data increases, dependent on two parameters: the resolution l_L, which specifies the granularity of the discretization, and how evenly the acquired data is distributed in the overall volume. If the data is distributed evenly inside the bounding box that contains the whole point cloud, the octree provides no benefit over a voxel grid. However, if the data is distributed unevenly, with large volumes of open space and some volumes that contain the vast majority of measurement points in a dense arrangement, which is usually the case when recording 3D environmental data, the cost of building an octree structure is outweighed by its efficient representation and handling of the data.

From observing Figure 2.10 it is evident that an octree leaf L_i might have a varying number of neighbors, depending on the point cloud data and where it is located inside the octree structure. This is another difference to the voxel grid representation, as mentioned earlier. When using the octree as a means of filtering a dense point cloud each leaf is represented as a single point. In this respect the octree works very similar to a voxel grid; the choices of points to represent a leaf are the same as for a grid cell. An individual octree leaf L_i can also be referenced by its grid coordinates $v^g_{L_i} = \left(x^g_{L_i}, y^g_{L_i}, z^g_{L_i}\right)^T$, just as in the voxel grid. The difference to the latter is, that not every grid coordinate in the limits defined by the octree's size refers to a leaf node, but can also point to a larger empty region.

However, the tree structure of the octree features some nice properties, apart from being more memory efficient than dense voxel grids. Octrees have been successfully employed for (approximate) nearest neighbor search (see [23], for example). Furthermore, they can be used for quick ray tracing and ray intersection queries, where the non-dense structure allows to skip investigating a potentially large number of empty cells that would have to be consulted when performing the same operation in a dense voxel grid. Employing octrees in this fashion dates back to the mid 1980s when Glassner [29] explored their capabilities to speed up computation times, for intersections of 3D objects, that were composed of geometric primitives with different surface properties.

If we want to know if a given ray intersects any surface in a scene that is represented by a 3D point cloud, the advantage of octrees is twofold: not only does it speed up the intersection query, since we do not need to consider any measurement points that are not located inside intersected leaves, but the octree also allows for better results of the query. The chance of a given ray to precisely intersect with a measurement point gathered from a particular surface in the recorded scene is small, since the measurement points only provide a sampling of the surfaces. Assuming that the intersection of a surface is equivalent with intersecting an octree leaf that contains measurement points from this surface, allows for an efficient intersection detection.

This assumption may lead to a slight overestimation of the surface, dependent on parameter l_L, the size of the octree leaves, but this overestimation is in general preferable to a query that (almost) always states that no intersection has occurred. Introducing some parameter ε that allows a maximal distance between ray and measurement point in order to consider the point intersected can be a workaround, when not employing octrees – but this also results in a potential overestimation of surface sizes, without the benefit of the octrees efficiency. Chapter 4 will showcase an example where the ray tracing capabilities of an octree structure will prove beneficial.

2.3.3 RANSAC-Based Plane Detection

With the emergence of low-cost 3D laser scanners, Time of Flight cameras and RGB-D sensors (see also section 2.4), the amount of published work concerning 3D object recognition, reconstruction and manipulation has seen a vast increase. While there are exceptions, for instance the approach by Hinterstoißer et al. [39], many of these approaches assume that objects of interest can be segmented reliably from the other parts of the collected 3D data, i.e., that some common structure is present in the recorded scenes. Since the aforementioned domains of recognition, reconstruction and manipulation are challenging topics in themselves, a common assumption is that the objects which are to be investigated are placed on a flat surface. Examples of this assumption can be found (among many others) in [4, 43, 49, 52, 66] and I will later make the same assumption in chapter 3. Oftentimes this assumption is not as artificial as it might sound at first, since a lot of work with 3D data is currently performed in indoor environments.

In these environments larger planar surfaces are usually found in abundance – just think of the walls, floors and ceilings. What is more, also the furniture that we use in order to organize objects in an orderly fashion in general also features planar surfaces, for example desks, tables or shelves, where objects are usually placed on top of horizontal planar surfaces. If we can detect and subsequently filter the points in a 3D point cloud that belong to such surfaces, we have already accomplished a major step towards successful object segmentation, which then might be the first step towards recognizing objects.

The method described here, namely *RANSAC-Based Plane Detection*, assumes that the majority of points, present in the current point cloud, belong to a planar surface. As described earlier, for indoor environments this is generally a valid assumption. The *RANSAC*-part of the name is short for *random sample consensus* and with the assumptions about the composition of the environment the actual detection of planes becomes rather simple. RANSAC works in an iterative fashion. It picks 3 points of the point cloud at random – hence the "random"-element in its name. If these three points are not collinear they are sufficient to define a unique plane in 3D space, which can be expressed as a normal vector ($\mathbf{n}$) and a single point on the plane (p_0) in the following equation:

$$(p_\mathrm{p} - p_0) \cdot \mathbf{n} = 0 \tag{2.10}$$

where p_p defines a 3D point that is located on the plane and $\cdot$ denotes the dot product. Note that there exist various other methods to denote the same plane, for example by providing a normal vector and a scalar instead of point p_0, that are equivalent to equation (2.10). In the case that the three random points are collinear, a different random triple of points will be drawn from the point cloud.

Once a plane is defined by choosing three random points, equation (2.10) can be employed to determine the distance of any given 3D point to the defined plane. Provided with a distance threshold ξ_max the points of the point cloud can be distinguished into two disjunct subsets, namely plane inliers I_p and outliers O_p, where $P = I_\mathrm{p} \cup O_\mathrm{p}$ holds. The set of inliers I_p is defined as the following:

$$I_\mathrm{p} = \{p_i : |(p_i - p_0) \cdot \mathbf{n}| \leq \xi_\mathrm{max}\} \tag{2.11}$$

All points $p_i \notin I_\mathrm{p}$, i.e., the points that are not part of the defined set, are considered outliers of the current plane. Threshold ξ_max becomes necessary to compensate for sensor noise in the data (and also for the fact, that real tabletops are often not perfectly flat, but contain minuscule crevices and protrusions. Employing a too small threshold ξ_max will cause misclassification of points that belong to a planar surface, so that not all points that are recorded of a tabletop are classified to belong to the same plane, for instance. On the other hand, if ξ_max is chosen too large, a number of points that one would not consider part of a planar surface are aggregated into the latter. This can become evident when observing the contact area between objects and their supporting surface. Oftentimes a small portion of the measurement points of the actual object is classified as points of the planar surface.

Once all inliers for the current plane equation are determined, they are stored along with the plane equation as the best plane detected so far. Subsequently three new random points are drawn to generate a different plane equation. This process is repeated for a previously specified number of iterations and whenever the currently constructed plane contains more inliers than the previous best plane (in terms of inliers), the latter is replaced alongside its inliers. Given a sufficient number of iterations and a scene that contains planar surfaces composed of a larger amount of measurement points (so that the probability to eventually draw three points that are placed on a planar surface is high), RANSAC plane detection can be applied successfully. Apart from the detection of planes in 3D data, the general idea of RANSAC can be applied to different models, if they can be described in a fashion analogue to the plane equation (2.10) and the idea has been published early in the 1980s by Fischler and Bolles [25]. In this case the number of points, that are needed to generate a model hypothesis will usually differ from 3, needed for a plane hypothesis.

If plane extraction was only performed to filter out background from potential object candidates, plane inliers may just be removed. In the case that the planes themselves are to be processed at a subsequent step, it is often sensible to project the inliers onto the extracted plane. If we denote an inlier point as $p_{\tilde{\mathrm{p}}}$ (note the difference in notation from p_{p} that lies on the ideal plane), its projection $\hat{p}_{\tilde{\mathrm{p}}}$ on the plane fulfills the original plane equation (2.10), i.e.,

$$\left|(\hat{p}_{\tilde{\mathrm{p}}} - p_0) \cdot \mathbf{n}\right| = 0$$

To compute the projection $\hat{p}_{\tilde{\mathrm{p}}}$ for a given plane inlier $p_{\tilde{\mathrm{p}}}$ generally two sensible approaches exist: either a perpendicular projection or, if the position of the sensor in the scene is known, a projection that takes the arrangement of the latter in relation to the plane into account. The perpendicular projection of a plane inlier $p_{\tilde{\mathrm{p}}}$ is defined in the following way:

$$\hat{p}_{\tilde{\mathrm{p}}} = p_{\tilde{\mathrm{p}}} - \left((p_{\tilde{\mathrm{p}}} - p_0) \cdot \mathbf{n}\right) \mathbf{n} \tag{2.12}$$

where the term $\left((p_{\tilde{\mathrm{p}}} - p_0) \cdot \mathbf{n}\right)$ denotes the distance of the plane inlier $p_{\tilde{\mathrm{p}}}$ from the specified plane (see equations (2.10) and (2.11)) The projection that takes the relative positions of plane and sensor into account, which will be referred to as *perspective projection* in the following. It provides a slightly more realistic projection, since it does not change the direction of the measurement ray for an individual measurement, but just adapts the distance along this ray in relation to the sensor. Furthermore, it becomes useful later on, when we are interested in the occlusion that a cluster of points causes on its support plane, for instance.

For the perspective projection, we first need to denote the line that is defined by the plane inlier $p_{\tilde{\mathrm{p}}}$ and the sensor's position $S_{\mathrm{pos},k}$. For each point p_{l} on this line the following equation has to hold for some scalar value a:

$$p_{\mathrm{l}} = a\left(p_{\tilde{\mathrm{p}}} - S_{\mathrm{pos},k}\right) + p_{\tilde{\mathrm{p}}} \tag{2.13}$$

The perspective projection is now defined as the intersection of this line with the plane, as defined in equation (2.10). This intersection can be computed by solving

$$a = \frac{(p_0 - p_{\tilde{\mathrm{p}}}) \cdot \mathbf{n}}{(p_{\tilde{\mathrm{p}}} - S_{\mathrm{pos},k}) \cdot \mathbf{n}} \tag{2.14}$$

Note that the planes detected by the RANSAC method are real planes in the sense that they are infinite, i.e., they have no borders and two unconnected portions of a point cloud can be classified to belong to the same plane. Therefore, some clustering after the plane detection is often prudent, when one is interested in the points that belong to a tabletop, for instance. Depending on the application scenario, the plane inliers might be removed from the point cloud or labeled as plane inliers before proceeding with other tasks, like object detection. Also note that, subject to the nature of the scene, several instances of RANSAC plane detection might be run consecutively to filter different planar surfaces, e.g., first filter the floor, then a tabletop and some recorded portion of a wall before proceeding with subsequent steps like object recognition.

2.4 3D Sensors

This section will briefly introduce some properties of the sensors employed in this thesis. Each physical distance sensor has some restrictions about what this sensor is able to observe (some of these restrictions may be lifted for some simulated sensors). These restrictions can be divided into two categories: Range restriction and restrictions of the field of view. The former are given in terms of a minimal and maximal measurement distance.

Independent of the measurement principle used, the sensors, briefly introduced in this section, have one common property: they usually provide only one range measurement for each measurement ray / pixel. Exceptions to this rule are high-end laser scanners that may receive multiple echos, if the signal encountered leaves or other vegetation before encountering a solid surface (Elseberg et al. [22]). However, even these sensors cannot measure through stone, concrete, soil or thick wood, thus only providing surface measurements. This behavior coined the term "2.5D data", which is sometimes used to describe data that was gathered from sensors that only measure the distance to the first encountered obstacle, but this term will not see further use in this thesis. Different sensor classes like radar do not feature this latter restriction, despite still having minimum and maximum range, of course, and thus provide "full" or "true" 3D data.

The restrictions on the field of view, for example due to the sensor's casing, are usually described by a vertical ω_{v} and a horizontal opening angle ω_{h}. As mentioned earlier, sensors come with their own coordinate system and this will need to be transformed, if the sensor's information is to be used consistently in other contexts. The sensor specific coordinate systems differ, depending on the sensor type. For cameras that provide 3D information it is common to associate the measured depth with the z-coordinate, which then coincides with the viewing direction of the camera (see Figure 2.3). In this way the image information is identical to that of a regular image, namely, located on the x-y-plane and the additional depth information is mapped to the added z-dimension. Laser scanners oftentimes use the z-coordinate to indicate if something is positioned "above" or "below" the sensor. The x-axis is often associated with the viewing direction and y points sideways, as depicted in Figure 2.1b.

Two different types of 3D sensors are relevant in this thesis: RGB-D cameras and 3D laser scanners. While both provide 3D point clouds, they differ in size, weight, frequency, measurement range, noise, opening angle, price and measurement principle. Some of these aspects will be compared in this section, starting with the properties of RGB-D cameras.

RGB-D cameras RGB-D cameras became very popular in the mobile robotics community with the advent of the Microsoft Kinect that was introduced in 2009 and available for purchase in November 2010. The main appeal of this sensor type is its low price tag (somewhere around 150 € when it was introduced) and that it also provides regular camera images, which is appealing for people involved in the computer vision community. In addition to the regular digital camera the sensor contains an infrared projector and camera. The projector emits a pattern, which is perceived by the infrared camera and analyzed by a microchip on the sensor. Distortions of the projected pattern allow for depth estimation. Both cameras, regular and infrared, are mapped onto each other, thus allowing a depth measurement for each pixel. Knowledge of the camera parameters allows for the creation of 3D points with additional color information. The resolution of the camera image is 640×480 pixels, so that the resulting (organized) point cloud is composed of 307,200 points and received at frame rates of approximately 30 Hz.

The camera has a vertical opening angle ω_v of $43\,^\circ$ and horizontal opening angle ω_h of $57\,^\circ$. Note that these angles may slightly differ for each individual sensor and do not always have to be completely symmetric, e.g., for a specific Kinect the horizontal opening angle to the left might be slightly larger than the horizontal opening angle to the right. Not all of the points, retrieved in an organized point cloud of the Kinect sensor are valid measurements. Usually around the edges of the field of view depth estimation is not possible, thus leading to `NaN`-measurements. Since the emitted pattern is composed of infrared light, bright sunlight tends to interfere with depth perception and overall data is noisy, when compared to laser scanners. Due to the measurement principle, the Kinect sensor exhibits a loss in depth resolution with increasing distances to the sensor, so that the measurement range for mapping and sensing applications in the robotic context is less than the maximal measurement range of the sensor. The latter can be, depending on lighting conditions and the surface properties in the environment, up to 10 m, but useful data is gathered at 2.5 m or less. The minimal measurement range of the sensor is 0.5 m. Apart from the Kinect, other RGB-D cameras exist, for instance the Asus X-tion. The X-tion provides very similar data, but is slightly more compact and completely powered via its USB connection and thus more appealing for the mobile robotics community. Information concerning differences in depth resolution, measurement accuracy and behavior in case of interference with other sensors for Kinect and X-tion can be found in [35].

Also similar in terms of the generated data, but different in the way that depth measurements are obtained, are *Time of Flight* (ToF) cameras. These also send out an active signal, but in this case it is not a defined pattern, but a momentary illumination of the scene. The camera sensor is then used to record the received signal and via the time difference between emitting and receiving, along with wavelength of the emitted signal, the distance between sensor and measured surface can be estimated. Since the measured time determines the distance, this measurement principle does not suffer from a systematic loss of depth resolution, unlike RGB-D cameras (that basically use a triangulation method to estimate depth). While ToF cameras have been used for quite some time, also in industrial applications, they have recently become much cheaper, probably due to the introduction of the Kinect 2 (released in November 2013) which switched from the depth estimation via a projected pattern to estimation by time of flight.

3D Laser Scanner The other type of sensor that is used in this thesis is called *Laser scanner* or *lidar* and sensors of this type work on one of two different measurement principles. The first one is similar to time of flight cameras, in the sense that a signal is emitted and the time until it returns is measured, thus obtaining distance information. In contrast to ToF cameras, not the whole scene is illuminated and perceived by a digital camera chip, but a single ray is emitted and the time difference for this ray is measured. So for each measurement point a very short laser pulse is emitted and the scanner waits for it to return. The other principle is based of measuring the difference in the phase, when receiving a continuously emitted wave signal. Since this difference does not determine the distance unambiguously, multiple signals with different wavelength are emitted that make it possible to resolve these ambiguities. Laser scanners that measure the time difference generally allow for larger measurement ranges, while the scanners that employ phase difference to measure distances in general can acquire more data points in a given time interval.

In mobile robotics 2D laser scanners, originally build for industry applications, were first used for mapping and localization tasks in 2D maps. In the early 2000s these sensors were used to build the first simple 3D laser scanners in the context of mobile robotics [45,75,79]. The sensors used in these publications have a measurement range of up to 80 m, depending on the reflectivity of the measured surface. Field of view, distribution of points inside a single 3D scan and the time that it takes to acquire a scan is dependent on the mounting of the 2D scanner and how it is moved in order to create a 3D point cloud. A small overview of the properties of the different mountings is shown in [37, chapter 2.5, Table 2.5].

Later on, lidars originally intended for surveying tasks were also employed in mobile robotics or automated scan registration tasks, for instance by research group of Nüchter [22,56]. Measurement ranges of such scanners exceeds the one of industrial monitoring laser scanners several times. For example the Riegl VZ-400i scanner can measure points in up to 800 m, depending on its current configuration. The horizontal opening angle of these terrestrial scanners is typically 360 ° and the scanner rotates during the data acquisition. Vertical opening angles are model dependent, but often somewhere around 100 ° (the elevation angle φ for a measurement ray, thus determining the vertical opening angle are for the VZ-400i is according to the data sheet $\varphi \in [-40°, 60°]$). The number of points inside a single scan is highly dependent on the chosen angular resolutions of the scanner and where a scan was taken (outdoors the sky reduces the number of points significantly). The scanner configuration also largely influences how long it takes to acquire a complete scan, which ranges between several seconds to several minutes. The rate with which measurement points are obtained depends on the aforementioned measurement principle, for the Riegl VZ-400i it is between 42,000 pts./s and 500,000 pts./s On a lower resolution, where the acquisition of a scan takes about 2.5 m in an outdoor environment, a scan contains oftentimes around 15,000,000 points, to give an impression about point cloud size and the time needed to acquire data.

When compared to RGB-D cameras, measurement results contain very little noise and the measurement range is much higher. Unfortunately the cost for such a terrestrial scanner is also much higher, since they are sold starting roughly at 30,000 €, while the more high end systems in this category are much more expensive (the VZ-400i costs approximately 100,000 €). It is easy to conclude that RGB-D cameras and laser scanners differ fundamentally in the properties of the data they produce and the rate with which they are able to acquire it – and they are not equally suited for the same tasks. The reconstruction of transparent objects (chapter 3) strongly focuses on the usage of RGB-D data, although the general principles are also applicable to many laser scanners or ToF cameras. Evaluation of alignment quality introduced in chapter 4, will be demonstrated on both, RGB-D and laser scanner data to show that the ideas are applicable in general on 3D data.

For a more detailed view on the underlying measurement principles of the sensors mentioned here, I would like to recommend some further literature. A good explanation about laser scanners and a short comparison between several terrestrial scanners alongside other common 3D sensors is written in the introduction of Jan Elseberg's PhD thesis [21]. An overview of 3D measurement techniques in general is provided by Blais in [10]. Although this text predates the advent of the Kinect sensor, it covers ToF and the theory behind the sensors that he refers to as *Slit Scanners* is basically the same as for range measurements of today's RGB-D cameras.

Chapter 3

Transparent Objects

3.1 Problem Description

Much work is currently done to enable mobile robots to work in an autonomous fashion. Starting from navigation tasks and path planning in given maps to reach specified location via exploration and map creation to identifying previously specified objects and their manipulation, the field has seen many improvements in recent years. One requirement for almost all of these tasks is to perceive the environment in a suitable fashion. As it turns out no single sensor is deemed perfect for all purposes: RGB-D camera data may suffer in certain lighting conditions and measurements are noisy; 3D laser scanners on the other hand do not provide data that can be used in computer vision applications and provide either a relatively sparse representation of their environment (Velodyne scanners) or need several seconds to complete a scan, which might be insufficient for obstacle avoidance in environments with other agents.

Even though no perfect sensor exists, RGB-D cameras and laser scanners have become prevalent in domain of mobile robotics, the former especially in indoor scenarios. These sensors classes, that are both based on emitting light from a source employing the sent signal to estimate distances, have one drawback in common. Light is refracted and / or reflected from transparent and highly specular surfaces, thus causing measurement errors in those parts of the scene that contain object and surfaces made from these materials. An example is provided in Figure 3.1, for both laser scanner data as well as RGB-D Data. Figure 3.1a shows a photograph of the scene, acquired from a Microsoft Kinect sensor. A drinking glass (on the right) is shown together with a nontransparent blue painted glass of the same type (on the left). In the corresponding point cloud (Figure 3.1b) measurement points of the painted glass are clearly visible, but absent for the unpainted version. Figure 3.1c shows a low resolution laser scan of the same scene, recorded from a slightly different viewpoint. The laser scan is also missing measurement points for the transparent object. In both, Figure 3.1b and 3.1c, points are colored according to their z-value.

In order to act in unknown or changing environments, perceiving such objects becomes an important issue to avoid damages – either caused by the robot (when a manipulator smashes a glass) or to the robot itself (when it drives at high speed into a thick glass wall). This need is emphasized in all research areas that aim to develop robots that can serve as assistants in

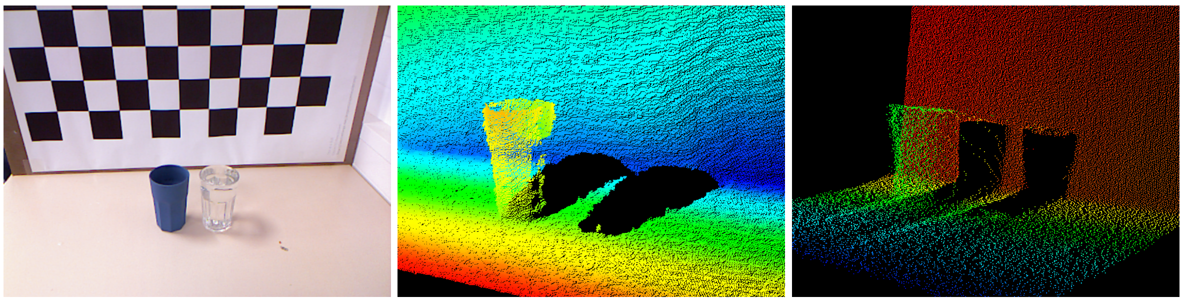


Figure 3.1: Missing / erroneous measurements due to glass objects: **(a)** shows a photograph of the scene (taken from the RGB-D camera) and **(b)** displays the corresponding point cloud. In **(c)** a (low resolution) laser scan of the same scene is shown. Point coloring dependent on their z-coordinate.

human households, since the regular occupants of such households own glass objects and do not always store them in a neat fashion in their cupboards, but may arrange them between other objects on a tabletop for instance.

Perceiving in this context is not only detection, but also estimation about the pose and 3D shape of such an object. If a detector simply tells that a transparent or specular object is present, this does not automatically help to avoidance them. For this, or more sophisticated tasks as manipulation, not only the position, but also the shape need to be estimated, to identify dangerous configurations and potential grasp poses.

Note that for the sake of brevity in the following the term *transparent object* will refer to both problem instances, transparent objects as well as objects with a highly reflective surface, such as mugs made of polished steel. The term *regular object* on the other hand will refer to objects that can be measured accurately by the sensor, taking into account its inherent noise.

3.1.1 State of the Art

As motivated and illustrated in the beginning of this section the detection of transparent objects in a given scene and subsequent estimation of their locations and size is a relevant problem.

The approaches to detect transparent object in a scene vary, depending on the available data, i.e., the employed sensor plays a crucial part. Besides ideas to employ sensors that emit an active signal, like RGB-D cameras and laser scanners, much work has been done with digital cameras, that do not send out a special signal but just register the ambient light as it is reflected from (object) surfaces. Much of the work on transparent objects, especially work that employs single camera images, focuses on the detection of transparent objects only, i.e., identifying the pixels that belong to a transparent object in a given image. Most of these publications will not be mentioned here, since the pure detection is only part of the problem, in my opinion, and should be followed by an estimation of position and shape. Nevertheless, detecting transparent objects in camera images is no trivial problem and different approaches have been used to tackle this task. A comprehensive review of the methods until 2008 was published by Ihrke et al. [40].

Approaches based on cameras usually focus on detecting highlights that appear on the edges of glass objects or at locations of high curvature, where ambient light is reflected in a different way as in the vicinity of such an object. For example, Saito et al. [68] used this method to estimate normals of a transparent plastic hemisphere. Fritz et al. [26] extract local features from these highlights and create a visual bag-of-words to identify transparent objects in images, but their work concentrates on detection, only.

Park and Kak [57] employ an optical-triangulation-based structured-light scanner to generate 3D models of objects placed on a rotating plate. In contrast to regular objects, transparent objects produce more than one peak in the perceived scan line, thus leading to ambiguity when estimating depth by triangulation. By adding some global consistency based constraints they solve this problem, but their sensor setup is not suitable for use on a mobile platform in unknown environments.

A more recent approach for detection is given by Hagg et al. [34] where they combine different sensor modalities, namely `NaN`-measurements and specular reflectance for object recognition tasks. Their paper also contains a recent state of the art section, but unfortunately the claim that previously published results by Stephen Marsland and me [3] require prior full 3D models of the reconstructed object is false, as will be elaborated later.

Common in spirit is the method of Klank et al. [42] where a ToF camera is employed to record the scene. They exploit the property that transparent objects appear darker in the intensity images of the camera than the other parts of the supporting surface and observe the scene from two different viewpoints. Assuming that supporting surfaces are flat, measurement points of a transparent objects are assumed to have a planar surface (more or less perpendicular to the support plane) and the fusion with the information of a second frame proceeds to approximate the shape via triangulation of corresponding points.

While Lysenkov et al. [48, 49] also use a RGB-D camera and the missing data, caused by transparent objects in the resulting point cloud, as cues for object locations in the scene, they demonstrate how machine learning can be applied to this problem. In a training phase 3D models of the transparent objects that can be detected are created by painting equivalent objects and recording them. Using these models, possible silhouettes from different viewpoints are generated. These are then used, after training is complete, to identify the most likely object candidate for a detected region of invalid measurement points and to provide an initial estimate about the object's pose, which is subsequently further refined. The appealing of this approach is that the object's shape and pose can be determined from just a single frame. However, this comes at the cost that only learned objects can be identified and no shape estimation can be done for unknown objects, thus limiting the applicability in unknown environments. Furthermore, painting an instance of each object might not be desired in every scenario, for example if some of the transparent objects in the environment are (expensive) unique pieces.

Very similar in spirit to parts of the work described in this section is the work of Torres-Gómez and Mayol-Cuevas [77] in the domain of augmented reality. Similar to my approach in [3] and detailed here in section 3.3, they observe the scene from multiple viewpoints. Instead of a RGB-D or ToF camera, a regular camera is used to record the scene. A learning approach is used to identify likely regions in the images that belong to transparent object. A set of images, taken from different viewpoints can then be used to approximate the glass shape in a fashion similar to probabilistic space carving [17]. Results look impressive, especially considering that only a regular camera was employed, but their approach requires a precise scene registration and camera poses were obtained using a calibration target. Imprecise pose information might compromise the obtained results, which is also a drawback present in method detailed in subsection 3.3.2, but addressed and improved on in the ideas presented in subsection 3.3.3. Furthermore, the approach might be sensitive to lighting conditions, although the authors do not report anything in this regard.

3.2 Detection in Single Frames / Scans

A prerequisite for a meaningful reconstruction of transparent objects is the detection of said objects. The two detection techniques, introduced in this section, distinguish between two different types of sensor input: if the sensor input consists of *organized point clouds* (see subsection 2.2.3) the detection becomes simpler than in the case of *unorganized point clouds*. As illustrated in section 3.1, the problem with transparent or highly specular objects in 3D point clouds is the absence of correct measurements on the objects surface, either resulting in invalid measurement data or in wrong estimates of the measured distance from sensor to object, resulting in a wrong estimate of the spatial relation between object and sensor, i.e., usually the object is assumed to be farther away from the sensor than it really is.

Let's briefly state the assumptions we make in the scenario for transparent object detection and reconstruction. Firstly, we assume that the captured data is taken from static scenes, i.e., no movement apart from the placement of the sensor in different poses occurs. Secondly, general laws of physics applicable on Earth hold. Combined with the first assumption this means that regular objects need to be in contact with a supporting surface, since they cannot float (due to gravity) and cannot be captured during some motion (falling or other), due to the assumption of a static environment. Thirdly, as often assumed in scene segmentation and object recognition tasks (see [49, 52, 66]), we assume that the objects that we are interested in are placed on top of a planar surface, for example a tabletop. This surface will also be referred to as the *support plane*, even though it usually does not stretch to infinity. In the case of a tabletop, where the boundaries of the support plane may be at least partially visible when observing the scene, we can furthermore define a *region of interest* (`ROI`) that is confined by the table surface or its *convex hull*.

Combining these assumptions with the special measurement properties of transparent objects, we can conclude two mutually exclusive criteria to detect potential locations of transparent objects in the measured data:

1. A cluster of invalid measurement points.
2. A cluster of valid measurement points either hovering above the planar surface or floating beneath it, while regular objects feature a gradual transition between measurement points on the object and the support plane on which they are placed.

For case 1 the reader might already be inclined to realize that such a detection becomes easier in the case of organized point clouds compared to unorganized point clouds. Since organized point clouds always contain a fixed number of points in a grid structure, even invalid measurements must still be included in order to comply with this structure. To distinguish invalid measurements from valid ones, usually special values like `NaN` are used to mark them inside the organized point cloud. As it turns out, the `NaN`-value in combination with the organized property makes the detection of potential transparent object locations rather a 2D than a 3D problem, which will be explained in detail in 3.2.2.

Unorganized point clouds do not feature such an underlying structure, as implied by their name. That means that the number of points, contained in unorganized point clouds which were captured by the same sensor, may vary from frame to frame, i.e., invalid measurements do not result in points with some special coordinate values, but simply in the absence of points. Not all sensors might provide their data with the information needed to construct a organized point cloud from their measurements. If there is an implicit (or explicit) ordering of the measurements, then it is possible to employ this information to construct an ordered point cloud from the measurement data. However, if such an ordering is missing or has been destroyed by filter methods like outlier removal, it is not possible to create an organized point cloud from this data. Last but not least, not all file formats that are used to store point cloud data provide for the storage of organized point clouds: one of the simplest ways to store a point cloud is to write all valid measurements sequentially into a file. Displayed in 3D, the order of the points in the list does not influence the appearance of the point cloud, but if the points in such a list are arranged, for example, in ascending order to the value of their x-coordinate, the grid structure that may originally have been used by a sensor to capture the frame will be lost.

Hence, it is prudent to discuss detection in both types of point clouds and not only concentrate on the most favorable scenario. In the following I will discuss how these differences between organized and unorganized point clouds reflect the techniques used to detect possible locations for transparent objects.

The approaches detailed in the two following subsections differ slightly with their intended use case. The detection in unorganized point clouds (3.2.1) produces, apart from detecting transparent, highly reflective objects and regular objects, the projected outline of the transparent object onto the support surface. This outline, described as a polygon, allows regions of valid measurement points on its inside, for example if the transparent object had a handle or something similar. These regions serve as the foundation of the intersection based reconstruction method, that is detailed in subsection 3.3.2 and experimental results are showcased in subsection 5.1.2, along with its problems concerning sensor noise and imprecise scene alignment.

The detection in organized point clouds (3.2.2) is geared towards the more robust viewpoint-based reconstruction of transparent objects, detailed in subsection 3.3.3. The improved robustness comes at the cost of losing intricate details in the object reconstruction and the corresponding experimental results are found in 5.1.3. Since organized point clouds most commonly originate from RGB-D cameras, nowadays, the description of the approach in 3.2.2 is closely linked to the latter experimental setup and less generic as the detection in unorganized point clouds.

3.2.1 Detection in Unorganized Point Cloud

While unorganized point clouds are more efficient in terms of memory, the detection of transparent objects and their likely locations is less easy, since the individual data point does not provide any hints, as opposed to the `NaN`-values in the organized case. Thus the proposed detection method described here will be a bit more elaborate than its counterpart for organized point clouds, that will be detailed afterwards in 3.2.2. Instead of investigating single points, the relative positioning of points in the recorded frame need to be considered when performing the detection.

First the scene is segmented into planar surfaces and point clusters that might constitute one (ore more) objects. Plane detection is done by applying the RANSAC-based plane detection (see subsection 2.3.3) in an iterative fashion, since a scene can contain more than one large planar surface. Once a plane is detected, its points are removed from the scene and the same RANSAC process is started on the remaining points once more. Termination criteria for this loop are the ratio of plane inliers to the total number of current points, as well as the overall number of inliers that is compared with a threshold specifying how many points a detected surface needs to contain, in order to be regarded as a valid plane.

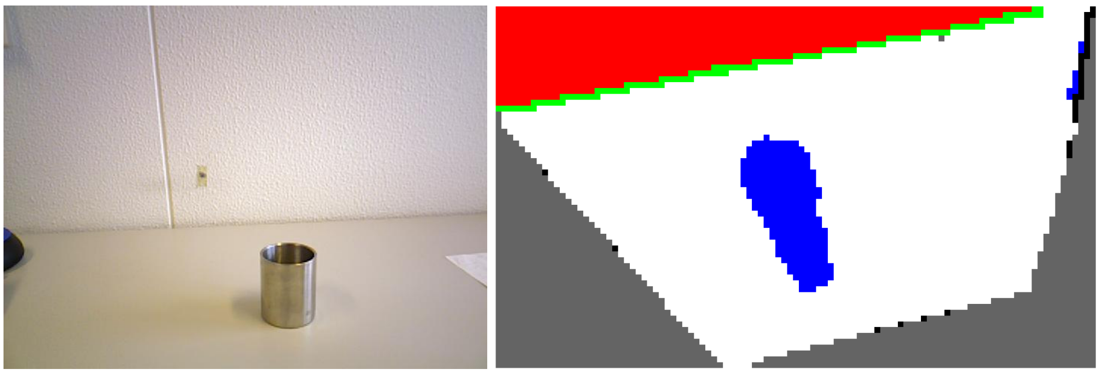
After the plane extraction took place a Euclidean distance based clustering is performed on any points that do not belong to any planar surface. The detected clusters denote potential objects (or a group of multiple objects that are positioned in close proximity to each other). While the objects in themselves are not what we are actually interested in, they will later on help to determine if a transparent object is present in the current scene.

Note that in this step it becomes possible to separate valid points on object surfaces from the (seldom occurring) points mentioned as case 2 previously: according to the assumptions that are made about the scene, an object has to have physical contact with its support plane. The minimal distance from an extracted object cluster to the inliers of its support plane strongly indicates whether the extracted cluster is a regular object or the result of measurement errors on a transparent / specular surface. In the first case the distance will be very small – as detailed in subsection 2.3.3 a threshold $\xi_{\max}$ needs to be specified to determine plane inliers, usually causing some of the object points also being classified as inliers. Hence, there is an almost seamless transition from object points to plane inliers. This is not the case, when a cluster from erroneous measurements is considered. Taking into account not only the smallest distance, but computing the average of the n closest points to the plane inliers increases robustness, of course.

Each extracted plane is converted into a rectangular 2D grid, where each grid cell is either considered occupied (if it contains at least 1 measurement point) or empty, otherwise. Note that for the creation of the 2D grid all plane inliers are projected onto the ideal plane in a perspective fashion, as defined by equations (2.13) and (2.14). After the projection has been performed the convex hull of the planar surface is computed.

If an occupied grid cell is placed on a common boundary of two intersecting planar surfaces, i.e., the distance of a plane inlier to the plane equation of a different plane is less than distance threshold $\xi_{\max}$, this is marked as a special subcategory of occupied grid cells. Furthermore, checks are performed, to determine if an occupied grid cell is located behind any of the other detected planar surfaces, when observed from the sensor's point of view. Such cells may exist, because of the distance threshold $\xi_{\max}$ for plane inliers and the perspective projection, that is used to project them onto the corresponding ideal plane. If a grid cell, either occupied or empty, is located on the boundaries of the 2D grid, or is located on the border of the convex hull of plane inliers, these cells are marked as in these special subcategories.

An example of an occupancy grid created in this fashion is depicted in Figure 3.2. On the left side the scene is depicted from the sensor's point of view, while Figure 3.2b shows the



(a) (b)

Figure 3.2: Planar surface and corresponding occupancy grid: **(a)** depicts the scene, while **(b)** shows corresponding occupancy grid for the tabletop. Grid cells are colored according to their category and colors are explained in the corresponding text.

occupancy grid that was constructed from the tabletop. White points indicate filled grid cells of the extracted plane and empty grid cells are shown either in blue (if they are inside the convex hull of plane inliers) or gray (outside of the convex hull). Black points mark the convex hull of the plane region, whenever it does not coincide with the border defined by the occupied grid cells. The green line shows where the planar surface of the tabletop intersects with the planar surface of the wall, located behind the table (see Figure 3.2a). All points in red belong to cells that are located behind this intersecting plane, when observed from the sensor's position.

All resulting grids, created in this fashion, are examined, to determine if they contain any *hole region*. A hole region, in this context, is defined as a connected region of empty grid cells of at least a certain size, that is located inside the planar region. Connectivity is defined in a recursive fashion, i.e., an empty grid cell is connected to its empty neighbors, and their empty neighbors, where the *von Neumann neighborhood* is used on the 2D grid. For each connected region of empty cells that contains more cells than a given threshold, the borders are investigated. If a large part of the region border is composed from empty cells on boundaries of the 2D grid, this region is discarded, since the sensor data does not support that it is located inside the planar surface. In order to compensate for small amounts of sensor noise or occlusion, a number of empty cells that coincides with the convex hull of the plane inliers is considered like an occupied cell, i.e., they can separate two hole regions. However, if a large fraction of the border of the hole region is composed of such empty cells on the convex hull, they are not considered a proper separation to the empty grid cells around the tabletop. In this fashion empty grid cells that are located in parallel to plane inliers, but inside the convex hull are not considered as hole regions, but discarded. An example of such a region is the small blue region on the right side of Figure 3.2b (although this also is not of sufficient size to be a valid hole region). Grid cells that are placed on two different planes are treated in a similar fashion to occupied grid cells, i.e., the form regular boundaries to hole regions.

This heuristic may sometimes result in discarding valid hole regions, that are located near the border of the currently observed scene, for example an occlusion caused by an object that was only partially observed, due to its position (see the desk trays on the right in Figure 3.3). On the other hand, if such regions were not discarded, empty grid cells that are caused by the embedding of the planar surface into the rectangular 2D grid would be treated as a hole. In Figure 3.2b the large gray areas in the lower left and lower right corner are (correctly) discarded due to this criterion. If we may assume that a (potentially transparent) object is not located near the borders of the field of view, the described approach does not cause any complications.

Each hole region, established in the way described above, is investigated if its existence can be explained by the data gathered in the point cloud. This is done by creating line segments between the center of every empty grid cell of a hole region and the sensor's position and checking if this line intersects any of the measurements that belong to one of the previously extracted Euclidean clusters. For efficiency and usability purposes the Euclidean clusters are inserted into an octree beforehand, and the octree representation is used in the intersection queries. If a line segment intersects any octree leaf, the corresponding cell in the hole region is regarded as explained by the data, otherwise the absence of measurements in this part of the planar surface cannot be explained.

A visual representation of these steps is depicted in Figure 3.3. In Figure 3.3a a photograph of the investigated scene is depicted from the sensor's pose during data acquisition. The table-top serves as the support plane, while the wall behind it is detected as a second major planar surface in the scene. Both detected planar surfaces are colored in gray in Figures 3.3b and 3.3c, respectively. Point clusters that were established after removing plane inliers from the scene are colored blue in Figure 3.3b.

Figure 3.3c illustrates the examination of the established hole regions. The grid cells that belong to a hole region, as defined above, are represented by their center point. The center points are either colored green, if their volume was occluded by some measurement from the sensor's pose, or red, if the absence of measurement data cannot be explained. In the former case, an additional point in blue is shown in Figure 3.3c: These points are the center points of the intersected octree leaves, that were constructed from all object clusters (blue points in Figure 3.3b).

The large occlusion caused by the desk trays on the right in Figure 3.3b is not classified as a hole region. Since the border of this hole region consists to a large part of the border of the 2D grid of the wall plane, it is discarded as not lying inside the planar surface. The same holds true for the occlusion caused by the bottles located on the left side of the scene, when considering the wall surface.

Depending on the ratio of grid cells, that are caused by a detected occlusion in the scene, in relation to the unexplained cells of a hole region, the latter is either assumed to be caused by a regular or a transparent object. If it is determined that the hole was caused by a transparent object a subsequent reconstruction of the object can take place, when the scene is observed from different viewpoints (detailed in section 3.3, experiments in 5.1.2 and 5.1.3).

While transparent objects usually do not provide any surface measurements, objects with a highly reflective surface often feature correct measurement points for parts of their surface. These points mostly occur at positions, where the measurement ray is (almost) perpendicular to the objects surface. An example of such an object along with steps of the detection pipeline are shown in Figure 3.4.

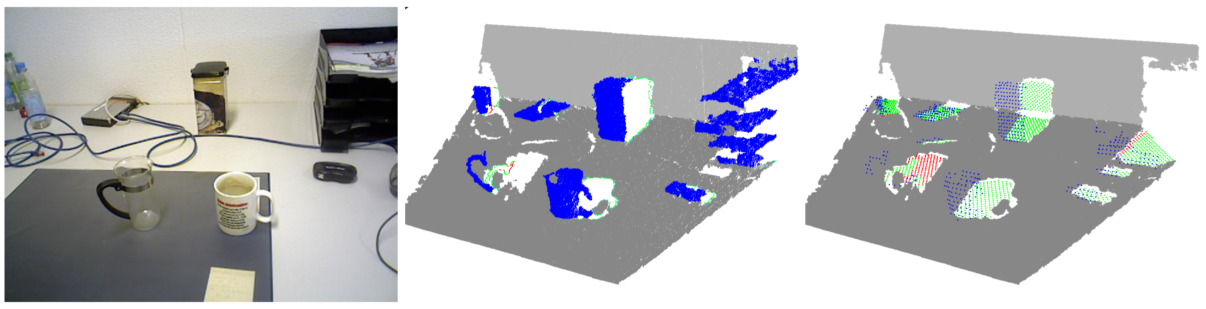

(a) (b) (c)

Figure 3.3: Different stages of transparent object detection pipeline: **(a)** shows a photograph of the recorded scene, from the sensor's point of view. **(b)** displays extracted planes, object clusters and the projection of their outline onto the extracted planes, while **(c)** illustrates the hole classification process.

Sometimes these partial surface measurements are sufficient to account for a large enough fraction of the hole region associated with the specular object – in such cases this object would be treated as a regular object, i.e., it would be assumed that the gathered surface measurements reflect the actual shape and size of the object. However, opposed to regular objects, other parts of highly reflective surfaces do not return the expected measurements, but either no measurements at all or very noisy measurements. An illustration is provided in Figures 3.4a and 3.4b: the (two distinct) clusters representing the metal mug can be clearly seen in the point cloud, where points are colored according to their depth, i.e., the z-coordinate determines their color (Figure 3.4a). Figure 3.4b presents the visual representation of the analysis of the detected hole region, as described above. Red points indicate grid cells that are not explained by occlusion from the measured object cluster, while explained grid cells and the intersected leaf centers are shown in green and blue, respectively. Note that the majority of grid cells of the hole region are explained (green) and therefore the hole might be classified as being caused by a regular object, if only the ratio of explained to unexplained grid cells is taken into consideration.

However, the property, that only parts of the surface of specular objects return measurements, can be utilized to distinguish regular objects from the former and thus prevent a underestimation of an object's size and shape. The latter becomes important, when subsequent tasks like object manipulation are executed, in order to prevent collisions between the manipulator and a wrongly represented object.

In order to distinguish point clusters, detected by Euclidean clustering, that belong to regular objects with clusters that only partially represent the surface of specular objects, the clusters are compared with the occlusion they cause on the support plane. This is done by extracting the contour of the point clusters and computing their perspective projection into the support plane (see Figure 3.4c). In order to project an Euclidean cluster point p_c onto the support plane in a perspective fashion,plane inlier $p_{\tilde{p}}$ in equations (2.13) and (2.14) simply has to be replaced by p_c.

Each projected point of the cluster outline is investigated if it is located closely to the border of the hole in the support plane, that was caused by the occlusion. In a precise fashion, this

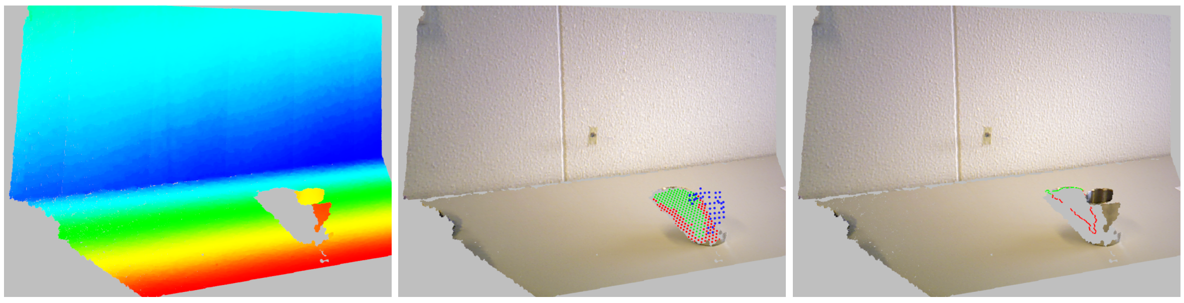


Figure 3.4: Detection of highly reflective object: **(a)** visualizes the recorded point cloud, colored according to the z-coordinate. In **(b)** the result of the hole examination is visualized, while **(c)** illustrates the investigation of the projected Euclidean cluster contour. A photograph of the scene is depicted in Figure 3.2a.

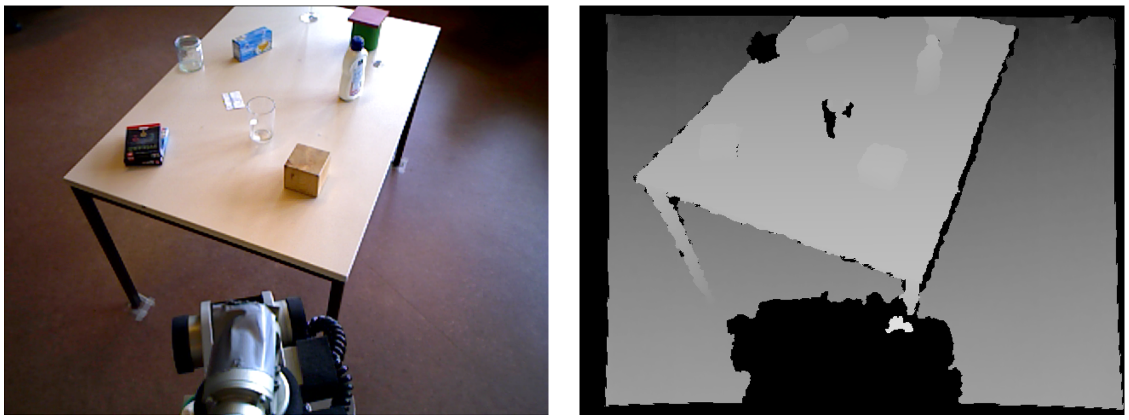
can be done by performing a radius search on the plane inliers. If the radius search returned at least one point in the vicinity of the projected cluster point p'_c it is assumed that this point belongs to an at least locally "regular" outline. However, if there exist a larger number of projected points where the radius search comes up empty, the clusters outline does not fit to the occlusion caused on the support plane. In this case it is concluded that the cluster points belong to an object with a highly reflective surface and a reconstruction, similar to the reconstruction of transparent objects (which will be discussed in subsection 3.3.2) can take place.

The difference in the reconstruction, when compared to completely transparent objects, is that available measurement information is incorporated in the reconstruction. This means that for those parts of the hole region that are explained by gathered point measurements, only the unknown parts, i.e., the volume between the measurements and the support plane are taken into consideration and not the volume between sensor and support plane.

3.2.2 Detection in Organized Point Clouds

As already mentioned in the beginning of this section, detection of potential transparent object locations becomes 2D rather than a 3D problem, when organized point clouds are available. The organized property (see 2.2.3) enables us to view a single frame as a 2D grid where each grid cell contains the distance from the measurement point to the sensor. Section 3.1 illustrated that transparent objects cause erroneous depth measurements – either in the form of completely missing measurement points (`NaN`-points) or by over- or underestimating the measured distance, due to prismatic reflection of the light based measurement signals.

If the used sensor produces `NaN`-points at the location of transparent objects we can employ a region-growing approach to detect potential locations of transparent objects. The general concept idea of region growing for image processing can be found in [30, section 10.4]. An exemplary visualization of a RGB-D camera image pair is displayed in Figure 3.5. The depth image is checked point by point until an invalid measurement (`NaN`-point) is detected. This point is used as a seed for an iterative *region growing*, employing a von Neumann neighborhood.



(a) RGB image **(b)** Depth image

Figure 3.5: Color and corresponding depth image: **(a)** shows the color part of an RGB-D image from a setup used in the experiments (see 5), while **(b)** is the corresponding depth image (brighter means closer). Note that points on the edges of the image and points that are closer than the minimal measurement distance usually do not have valid depth measurements and are thus colored black.

Each visited image coordinate is marked, so that measurement points are only considered once, when constructing a region of invalid measurements. The valid measurements that bound the current `NaN`-region are stored alongside the `NaN`-region, since it will become useful in the reconstruction of the transparent object. If a region of interest in the current frame is known before the transparent object detection, for example the tabletop in 3.5 the points that may initialize a new `NaN`-region can be restricted to points that are inside this region of interest. It should be noted however that in some cases it can be sensible to allow the `NaN`-region (and its border) to grow beyond the limits of the region of interest (see the glass placed near the edge of on the left of the tabletop in Figure 3.5), if a subsequent reconstruction of the transparent objects is desired.

If transparent objects do not cause easily identifiable invalid measurements, i.e., `NaN`-points, but erroneous depth measurements, we can apply a similar strategy to identify potential locations of transparent objects: instead for regions of `NaN`-points we now need to search for regions where the points that are part of the region have similar distance measurements, but distinctly different depth measurements, when compared with the measurement points of the support plane. Note, that regular objects that are placed on the support plane will also consist partially of depth measurements that differ from the average depth of the support plane, especially near the top of these objects, but these depth measurements usually change smoothly and at the base of the objects, where they touch the support plane, correspond with the depth measurements of the support plane. This property can be observed in Figure 3.5b.

In terms of region growing, this means that we can just start a new region with an arbitrary point of the support plane and add the neighbors to the region if the difference in depth to the current point is below a given threshold. Again we need to keep track of which measurements have already been considered and which can still be added to a region. This way one large region will be extracted, namely, the support plane, and due to the smooth transition in depth between regular objects and the support plane these will also be included in this region. Other extracted regions are strong candidates for transparent objects, especially if their average depth is larger than the average depth of the region including the support plane.

Alternatively the support plane can be filtered first, utilizing the actual 3D data instead of the organized 2D grid structure, removing its inliers while retaining the organized property of the point cloud (see 2.2.3), i.e., replacing inliers by `NaN`-points and storing the extracted plane inliers separately. Once the support plane has been removed the perspective is switched back from the 3D view of the data to the organized 2D grid structure. Remaining clusters of measurement points in the 2D grid constitute candidates for either regular objects or transparent objects. These clusters can now be easily extracted from the 2D grid, for example in a region growing fashion. Note that the comparison of depth measurements of the remaining 2D clusters is equivalent to extracting the 3D points that correspond to the 2D cluster and determining how close this collection of 3D points is to the support plane.

To distinguish these two classes from each other, we look at two properties of these clusters, namely where they are located in relation to the support plane, when observed from the sensor's location and the distance to the support plane. As mentioned in the previous paragraph, we can assume a smooth transition in terms of depth measurements between a regular object and the support plane on which it is placed under the general assumptions we made for the observed scene (static and obeying the laws of physics). Therefore, if a remaining 2D cluster contains depth measurements that are very close to the extracted support plane, then this indicates that the cluster of depth measurements is caused by a regular object that was placed on the support surface.

While objects that are placed onto a plane naturally touch this plane, i.e., the actual distance from object to support plane is 0, this is usually not directly reflected in the measured data: as explained in subsection 2.3.3 a distance threshold $\xi_{\max}$ is used when extracting the plane inliers, that causes the lower part of object points to be part of the plane. Hence, when checking the distance from object clusters to the ideal plane via the plane equation it will be larger than 0. If, as an alternative, we do not consider the distance to the ideal plane, but define it as the minimum point-to-point distance between cluster points and $p_{\bar{p}} \in I_{p}$, the plane inliers, this will of course also not be 0, since measurement points are discrete. However, for regular object the distance, independent on the method of computation, should be small.

The second criteria considers the position of the measurement cluster in relation to the support plane: clusters that correspond to regular objects, placed on the support plane, need to be located above the support plane if the scene is observed from the pose of the sensor that collected the measurement data. Transparent objects however, where the measured distance is overestimated, may cause clusters that are located below the support plane. Hence, this can be used as a second criterion to identify clusters that are not caused by regular objects.

It is easier to detect clusters that describe transparent objects in the 2D grid than it is in the corresponding 3D data, once the support plane has been removed: the organization in the 2D grid provides a natural way to determine neighboring measurements, namely the depth measurements in adjacent grid cells. However, the corresponding 3D coordinates (see equation (2.2), since the depth image is a way to convey spherical coordinates) do not necessarily have a small Euclidean distance to each other, if there is a large variance between the measured depth. Since the erroneous depth measurements on the surface of transparent objects typically are subject to stronger noise than measurements taken from regular surfaces, a measurement cluster in the 2D grid does not always lead to a compact cluster in 3D coordinates, even though measurement rays encountered the same object.

Increasing the threshold for Euclidean clustering, so that the remaining 2D clusters will also result in clusters in 3D, is generally not a good option, since this also increases the chance of false cluster associations. This may result in distinct physical objects that are placed close to each other in Euclidean space getting merged into a single large cluster.

Of course, the methods that work on unorganized point clouds see 3.2.1 can in principle also be applied to organized point clouds.

When observing real data, displayed exemplarily in Figure 3.5 it becomes apparent that there are still some issues which should be addressed before assuming that a detected 2D cluster of either `NaN`-points or unusual depth measurements is actually caused by the presence of a transparent object. Therefore, `NaN`-regions (or their counterparts of valid measurement points) will be divided into 4 different classes, that are defined next:

`NaN`.1 Regions that are (at least) partially caused by missing measurements on the boundaries of the underlying organization, i.e., near the borders of the image, need to be identified and filtered.

`NaN`.2 `NaN`-measurements that occur at sharp edges of regular objects need to be distinguished and removed.

`NaN`.3 Small clusters, caused by sensor noise, need to be removed.

`NaN`.4 Valid regions `NaN`-measurements sometimes might need to be fused, due to spurious valid measurements inside a transparent object.

These different types are illustrated in Figure 3.6 and the difficulty of handling them varies: the simplest solution for removing false candidates for transparent object locations is a threshold that sets a minimal size for a valid candidate, thus handling all types that fall into category `NaN`.3, i.e., getting rid of the small cyan `NaN`-regions in Figure 3.6. This becomes necessary especially when employing RGB-D or ToF cameras, since current models are subject to a lot of sensor noise (see [35]) and often produce isolated faulty measurements in a single frame.

There are two general ways to discern the size of the regions. The first one is to simply use the number of pixels / measurement rays that contribute to the region and decide, based on this information, if a specific region belongs into class `NaN`.3. If certain assumptions can be made about the sensor setup (maximal distance for measurements, more or less fixed angle between viewing direction of sensor and support plane), this method is easy to check and provides sufficiently good results. However, if these requirements are not met, this method should not be employed, since there might be a large variance in the physical size of regions that are composed of the same number of 2D coordinates in the organized point cloud.

In this case it is more sensible to compute the actual size, i.e., the area, of these regions. If the sensor returns (faulty) measurements for transparent objects, the way to estimate the size of the region is straight forward. All associated 3D points of the detected 2D cluster are projected, via the perspective projections (equations (2.13) and (2.14)) into the support plane, their outline is

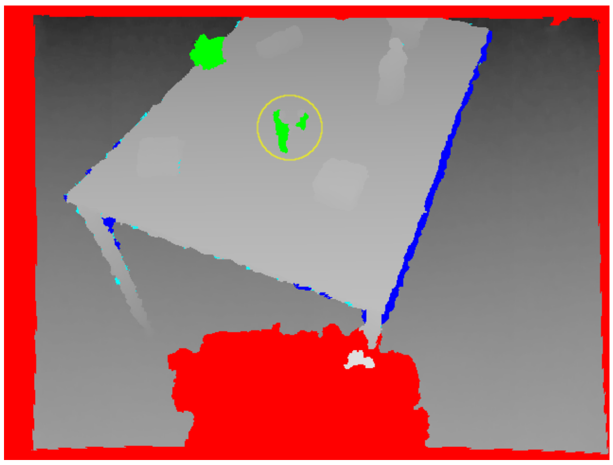


Figure 3.6: Different classes of `NaN`-point clusters in the 2D frame grid. Regular measurement points are displayed in gray, according to their depth, while the colors distinguish the different types of `NaN`-point-clusters: red illustrates a cluster that fall into category `NaN`.1, while blue shows category `NaN`.2 and cyan `NaN`.3. The yellow circle marks two valid clusters from category `NaN`.4 that need to be fused, since the are caused by the same transparent object.

extracted and subsequently the area of this 2D polygon, located in 3D space, can be computed. Projecting the points into the support plane satisfies one of the initial assumptions, that all objects are located on such a plane and thus the projection provides us with the occlusion (or "shadow") the object causes on the support plane.

If measurements on transparent objects cause `NaN`-points, as in Figure 3.6, we cannot project the cluster points directly. However, it it easy to determine the border, i.e., the valid measurements, that are neighbors to a `NaN`-regions, in the grid structure of an organized point cloud. To the set of these neighboring points the perspective projection can than be applied and the area of the resultant polygon can be calculated. Note that usually most of these neighbors will be plane inliers, so that the projection for them might already be available, but if the `NaN`-region overlaps with the border of the support plane, like the upper green `NaN`-region in Figure 3.6, the projection becomes necessary to have a 2D polygon, instead of a non-planar 3D polygon. Furthermore, here it becomes important to use the perspective projection instead of the orthogonal projection (equation (2.12)), since the latter might give a wrong impression of the `NaN`-region's size, if border points are not located on the support plane.

Once the area of the 2 the region on the support plane is computed, it can be compared to a given threshold, to determine if this region is part of class `NaN`.1 or of sufficient size to be a candidate location for a transparent object. Please note that this method also has its limits when trying to determine the size of a `NaN`-region on the support plane: the farther away the `NaN`-region is located from the sensor, the less dense is the point cloud in that part of the scene and the distance between neighbors in the grid structure increases for the corresponding 3D points. Hence, at large distances, using the projected border of a `NaN`-region to establish its size might lead to an overestimation of the area.

Regions that fall in category `NaN`.1 can be detected and handled easily. They occur at the edge of a frame and can thus be detected by checking the 2D coordinates of the grid cells that compose the region. If the width of an organized point cloud is $P_{\text{width}} + 1$, its height $P_{\text{height}} + 1$ and a region contains 2D grid coordinates (u, v) with either $u \in \{0, P_{\text{width}}\}$ or $v \in \{0, P_{\text{height}}\}$ then this region touches the border of the currently observed scene. Thus for the most common sensors (RGB-D cameras) that produce organized point clouds we can assume that the cluster of `NaN`-points was at least partially caused by missing measurements at the aforementioned borders and are thus discarded. The caveat of this approach is the possibility that `NaN`-clusters that contain "valid" `NaN`-points, i.e., points that were actually caused by the presence of a transparent object in the scene, are discarded alongside the missing measurements at the frame borders, if the `NaN`-points, with their different sources, adjoin each other. However, we cannot really distinguish the actual cause of the invalid depth measurements and can thus hardly determine a valid border that encloses the `NaN`-region, which is needed later for the reconstruction of the transparent object (see section 3.3).

The problem of discerning size and shape of a `NaN`-region (or its equivalent of valid measurement points) also exists, if the employed sensor does not cause faulty measurements at the borders of the frame and such a region touches the boundaries of the point cloud. If a transparent object is only partially observed in the current frame, it is prudent to discard this cluster for the subsequent object reconstruction, as opposed to making some (unfounded) assumptions about the extensions of the object. Note however, that in case of a robot with a suitable map representation it is sensible to add the detected location to the map for further investigation in the future.

If the application furthermore allows to define a region of interest in addition to the support plane, we can make some more restrictions when filtering the potential locations for transparent objects. A typical `ROI` can be constituted by the convex hull of a point set belonging to the support plane, for example if we want to investigate (transparent) objects on a tabletop. In this case the support plane is defined as the mathematical plane which contains the majority of the measurement points collected on the tabletop, and afterwards the (2D) convex hull of this set of points (projected into the support plane) can be computed to define a region of support. Depending on how such a region of support is defined it can be transferred from 3D coordinates into the 2D grid of the organized point cloud, which can allow for more efficient and simpler techniques to determine if a given point is inside the region of interest. In the case of the convex hull, extracted in the 3D point cloud, we can transfer it easily into an organized 2D representation of the point cloud, since the vertices of the convex hull are a subset of the original point cloud and therefore a direct mapping between each vertex and a 2D grid coordinate exists and the convex property of the hull is preserved.

Hereafter, the description will be restricted to the following exemplary case: the region of interest is defined as the 3D convex hull of the support plane and the latter is identical to a tabletop. To make the description more compact, it will be assumed that the sensor produces `NaN`-values in the presence of transparent objects. Please note, however that in case of erroneous depth measurements the methods can easily be adapted or become simpler, as was already shown earlier, when determining the size of a potential transparent object location.

Figure 3.7 displays the convex hull of a tabletop extracted in 3D Cartesian coordinates and its transition into the corresponding depth image. The advantage of this transition of the region of interest into the 2D grid representation is twofold. First, it becomes very efficient to

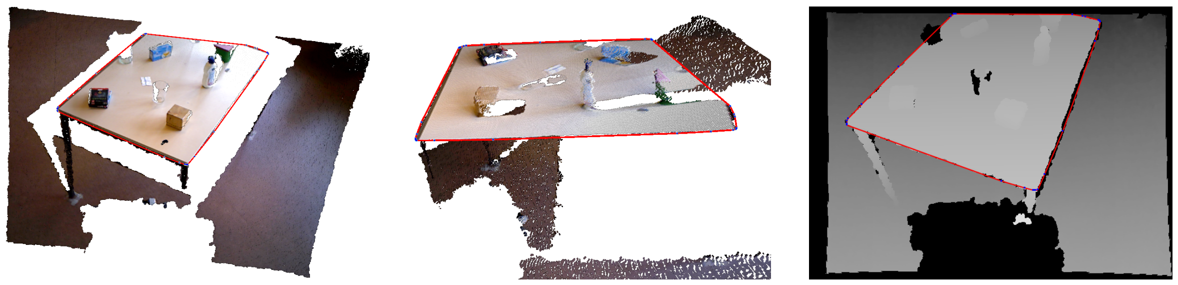


Figure 3.7: Transformation of convex hull from 3D to 2D: **(a)** and **(b)** display the colored 3D point cloud with the detected convex hull (vertices in blue, polygon in red); **(c)** shows the extracted hull in the corresponding 2D depth image.

determine if a given point $p_i \in P_j$ is inside the region of interest, since we now just have to consider if the 2D grid coordinates of p_i, i.e., a 2D point with coordinate values in $\mathbb{Z}^+$, are inside a convex 2D polygon, where the vertices are also composed of positive integer values. If we were to consider this problem in 3D Cartesian space, it would become harder due to the added dimensionality, so in general the 2D case is preferable [36]. While the convex hull itself is still a two dimensional polygon, since we created it from the (projected) inliers of the tabletop, it is embedded in a 3D Cartesian space and in general its normal is not parallel to any of the coordinate axes. In the full 3D case we would need to determine if point p_i lied on the extracted plane and, if this were the case, we would have to investigate if it were indeed inside the convex hull boundaries in 3D.

Of more importance is the second advantage that the 2D representation provides, namely, that it directly allows for checking if `NaN`-points are inside the region of interest, while in the Euclidean 3D representation this can not be done directly (due to the invalid 3D coordinates of `NaN`-points). This is again a feature of the organized property of the point cloud: each point $p_i \in P_j$ has a unique 2D grid coordinate (u_i, v_i) independent of the measured depth or the resultant 3D Cartesian coordinates. Therefore, we do not need to distinguish between valid measurement points and `NaN`-points in order to consider if they are inside the convex hull of the extracted tabletop. This property enables us to easily determine if a cluster of `NaN`-measurements is either completely inside the region of interest, partially inside or completely outside. Since we assumed that all relevant objects are inside the region of interest, e.g., are placed on the tabletop, we can safely discard all regions that fall into the last category. The clusters of `NaN`-points that are only partially inside the region of interest need to be investigated further to determine if they were potentially caused by a transparent object inside the region of interest or not.

Testing how a `NaN`-region is located with respect to the convex hull of the tabletop is not done on the complete set of `NaN`-measurements, but only for its 2D convex hull. This is sufficient to distinguish where the region is located (inside or outside) or if there is overlap. Furthermore, the convex hulls of the `NaN`-regions will prove useful later on. The 3D convex hull of a `NaN`-region is computed from its border of valid measurement points in Cartesian coordinates after they have been projected, according to equation (2.14), into the tabletop plane. The

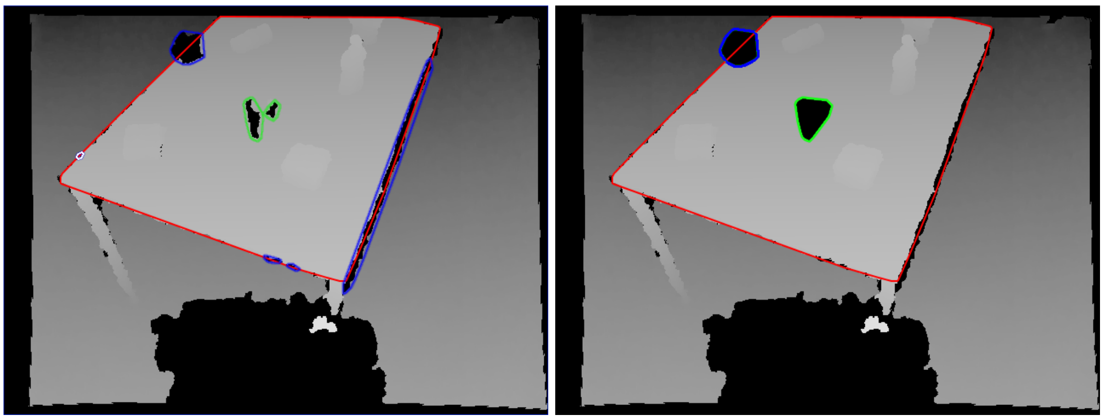

(a) (b)

Figure 3.8: Extracted convex hulls of NaN-clusters and the support plane. Convex hull of support plane (tabletop) is marked in red, convex hull completely inside the tabletop are marked in green, while all hulls that overlap the tabletop hull are marked in blue. **(a)** shows the convex hulls, before removal of category NaN.2 **(b)** shows valid hole regions after fusion and cleaning.

direct mapping from the 3D points to the 2D grid coordinates defines the 2D convex hull of the NaN-region.

If a transparent object is placed near the boundaries of the ROI the NaN-region caused by it, may well extend beyond the convex hull of the tabletop, depending on the point of view (see the glass on the left side of the table in Figure 3.5). On the other hand, if the region of interest features some sharp edges at its border, like a tabletop, the possibility for NaN-regions of category NaN.2 to partially overlap with the convex hull becomes large (see Figure 3.6). In order to distinguish between these two cases we measure how deep a cluster of NaN-points extends into the region of interest, if it is only partially inside the latter. The amount of overlap we want to measure should in turn be determined in Cartesian space; as mentioned earlier, equal distance in terms of grid coordinates does not necessary translate to similar distances in Euclidean space. This can be done efficiently by employing the 2D and 3D convex hulls of a NaN-region: when performing the overlap check, all vertices that are located inside the ROI are stored. For each vertex in this subset the 3D coordinates are used to compute the closest distance to a vertex or line segment of the tabletop's convex hull. The maximum of all these distances is used to determine how much the NaN-region overlaps with the convex hull of the tabletop. Afterwards this maximum is compared to a threshold in order to determine if the region will be retained or discarded.

Examples of the convex hulls of NaN-regions are displayed in Figure 3.8a, where the green convex hulls lie inside the ROI and blue convex hulls overlap. Apart from the large overlapping NaN-region located near the top left of the tabletop, all other overlapping NaN-regions will be discarded in this example and only the convex hulls of NaN-regions are displayed that at least partially overlap with the ROI and where the perspective projection was not discarded due to size (class NaN.3).

The reasoning behind this criterion is that, according to the initial assumptions, (transparent) objects have to be located on the support plane, more specifically inside the ROI, if the latter is provided. Even though the NaN-measurements, caused by a transparent object, may protrude from the convex hull in the projection of the depth image to a large extend, depending on sensor location and the shape of the object, the footprint of the object has to be at least partially located inside the region of interest. The latter statement can be made a bit more precise, since a static environment is assumed and therefore the object needs to be stable, i.e., the orthogonal projection of its center of gravity has to be inside the ROIand inside its own support polygon.

The investigated points of the NaN-region's convex hull, that are located inside the ROI, resemble those parts of the footprint that are facing the sensor (note that a protrusion beyond the convex hull can only happen in viewing direction of the sensor and not opposite its viewing direction in case of depth images or similar data). By defining the aforementioned threshold we employ a heuristic value that determines how much of an object's footprint has to be located inside the region of interest. The advantage of this method lies in its simplicity, but there is, of course, a certain drawback. Objects with a small footprint that are located near the border of the ROI might be discarded, when observed from an unfavorable point of view. Furthermore, certain shapes of objects might be prevented from detection. For instance a highly polished metal ruler that is placed in parallel close to the border of the tabletop's convex hull would also be discarded. However, for more common transparent and specular objects the detailed approach works reasonably well and NaN-regions of class NaN.2, located near the borders of the ROI, are successfully removed (see the transition between Figures 3.8a and 3.8b for an example).

Note that with the proposed method NaN-regions that are caused by sharp edges from objects, located on top of the table are not removed and may lead to false detections. Alternatively to the distance-to-ROI criterion one could investigate the shape of NaN-regions in more detail in order to also discard these regions. However, this will make some assumptions about the shape of transparent objects that may occur in the scene and in this case the aforementioned metal ruler might be discarded, even if it is located in the center of the ROI, depending on the shapes that are identified as to be likely caused by sharp edges.

It is more convenient to determine the plane of support in 3D Cartesian coordinates as opposed to the 2D depth image. Since the depth image closely resembles spherical coordinates (azimuth and elevation of the corresponding measurement ray with respect to the viewing direction can be determined for each pixel) a planar surface will appear differently, as can be observed for the sides of the cube in Figure 2.6b. The observed depth for the support plane will vary, but the magnitude of the gradient and its direction in the image is dependent on the relative pose between sensor and support plane.

Using 3D Cartesian coordinates allows for a quick detection using common RANSAC-based plane fitting algorithms (see section 2.3.3) and automatically yields the plane equation. The plane equation in turn enables us to project plane inliers into the plane and also, which becomes important later in the reconstruction, to (perspectively) project other points onto the plane. Once the inliers of the support plane are determined, the computation of the convex hull can either be performed in 3D or in the depth image.

Furthermore, note that choosing the convex hull as the region of interest, as done in the illustration and in the experiments described in subsection 5.1.3, has several reasons: firstly, it approximates the actual shape of the tabletop surface reasonably well, and secondly, the convex hull is composed of measurement points. This enables its use in both 3D point clouds and depth images. Had the `ROI` been chosen as an axis-aligned bounding box around the tabletop, where the vertices do not coincide with measurement points, the mapping between 3D coordinates and images would be less trivial.

Until now I have presented methods to remove `NaN`-regions that are caused by the depth estimation mechanism of RGB-D cameras and / or exceed the current field of view (class `NaN`.1), regions that may be caused by sharp edges on the near the boundaries of the convex hull of the tabletop (class `NaN`.2) or regions that are of insufficient size (class `NaN`.3). Ideally this should be enough to start with the reconstruction, but the recorded data will sometimes be less ideal than one would hope for. Exemplarily this is illustrated by the two established green convex hulls in Figure 3.8a. Due to their size and location inside the tabletop's convex hull, both are deemed as valid `NaN`-regions, but if we also take the scene composition into account (photograph of the scene in Figure 3.5a) the separate `NaN`-regions were caused by a single glass and in their current forms do hardly resemble the glass's shape.

To address this problem, all remaining `NaN`-regions, or more specifically their convex hulls, are investigated once more. Regions that are located close together, in terms of the minimum Euclidean distance between their convex hulls in the 3D scene, are fused together and after fusion, their convex hull is recomputed. Once this is done for all `NaN`-regions, plane inliers, that are located inside a convex hull of a `NaN`-region, are marked and for the purpose of the subsequent reconstruction also assumed to be `NaN`-values. The result of this final step, that is not quite a detection, but a preparation for reconstruction in the presence of noisy input data, can be seen in the example scene in Figure 3.8b, where 2 `NaN`-regions have remained in the scene.

3.3 Reconstruction

Section 3.2 illustrated how to detect (potential) locations for transparent objects in single frames, taken from RGB-D cameras, laser scanners or similar sensors. This section will describe in detail how to reconstruct the approximate volume of the associated transparent object, without any prior knowledge about its size or shape, given multiple frames of the scene, taken from different viewpoints. Other approaches either make assumptions about the general shape of all transparent objects (Klank et al. [42] with the planar surface assumption for the reconstructed objects) or require a learning phase for all objects that are intended to be recognized (Lysenkov et al. [48, 49]). Therefore, their usefulness is limited in environments where not all encountered transparent objects are initially known or the shape of the objects is not planar.

In particular two different approaches are discussed, namely a precise approach that allows for the reconstruction of non-convex volumes, but requires a precise pose estimation between different frames (subsection 3.3.2). As an alternative to this approach a more robust method with respect to sensor noise and registration errors is introduced in 3.3.3, which comes at the cost of reduced expressiveness, i.e., the volume of the transparent objects will in general be approximated by their 3D convex hull. Both presented reconstructions are based on the same assumptions, which are detailed in 3.3.1.

3.3.1 Occlusion Frusta & Mutual Occlusion

Having detected a `NaN`-region of suitable size in a single frame, that is a valid candidate to be caused by a transparent object, poses the question how to employ this information in a way to reconstruct at least some of the geometrical properties of the object responsible for the `NaN`-region.

If we ignore for a moment that a `NaN`-region could have been caused by sensor errors independently of any transparent objects, we can conclude that the `NaN`-region must have been caused by something placed between the sensor and the supporting plane (how to deal, at least partially, with `NaN`-regions caused by other sensor errors was already addressed partially in subsection 3.2.2). However, this assumption does not yet allow for a precise approximation of the volume, but only for an upper limit, referred to in this work as *occlusion frustum*. The occlusion frustum describes the volume that is constructed if we take the sensor's position as the apex of a (geometrical) pyramid and convex hull, associated with a `NaN`-region, as the pyramid's base. Visual representations of occlusion frusta will appear later in the text, for example in Figures 3.11 or 5.6.

Obviously if the transparent object, that caused the `NaN`-region, would exceed the volume of the occlusion frustum, it would have created a larger "footprint", i.e., the `NaN`-region would have been larger than recorded by the sensor. However, from a single frame we cannot deduce a lower limit of the volume, yet: for example if the point cloud we are investigating is organized, we can only assume that for each measurement ray, associated with the `NaN`-region, something was placed somewhere along the path from sensor to an expected measurement, i.e., the intersection of this particular sensor ray and the support plane. This in turn means that, the albeit unlikely, but possible scenario exists, where each measurement ray could have been "blocked" by some small, individual object, which has no connection with any of the other objects that created the `NaN`-region (think of a small dust cloud between sensor and plane of support that might have been removed by outlier filters). Furthermore, these objects do not necessarily need to have been transparent in order to create a `NaN`-region. Since most sensors have a minimal measurement distance, anything that is closer to the sensor than this distance might cause `NaN`-measurements and furthermore likely to cause a `NaN`-region, because at a relative close distance to the sensor it is likely that multiple measurement rays are affected by even small objects. Again the aforementioned dust cloud might cause such an effect – in this case not even caused by "overzealous" outlier removal. Any meaningful reconstruction needs to be able to cope with these false positives in terms of transparent objects, i.e., objects that cause `NaN`-regions only from certain viewpoints or general measurement noise should not culminate in the reconstruction / creation of a non-existent transparent object. However, if we briefly ignore the various reasons how a `NaN`-region might be caused in a single frame without the presence of a transparent object, the notion of the *occlusion frustum* can be employed in the subsequent reconstruction of the transparent object.

The occlusion frustum behaves similar to the shadow caused by a regular object on a planar surface and a light source. As in the transparent object scenario, the shadow of a regular object alone does not provide enough information to discern the overall shape or size of the object, since it is only a 2 dimensional projection onto a planar surface. The actual size of the object can only be ascertained if additional information, like the distance of the object to the light source, is available, but the outline of the shadow poses an upper limit for the spatial dimension of the object, i.e., it cannot exceed the limits defined by the pyramid with the shadow as its base and the light source at the apex. In order get a better grasp of the overall shape of the object, multiple shadows, caused by differently positioned light sources become necessary, if no additional strict assumptions, like rotational symmetry along the objects upward axis are made.

The shadow metaphor illustrates another important point, namely that the shadow might change size, position and shape, if the light source is moved in relation to the object that casts the shadow. No matter how the light source is moved in relation to the object, the pyramid created by the shadow and the light source always contains the whole object and while a fixed object does not change its position or size, its shadow changes, depending on the light source's position. The same holds true for a transparent object and the `NaN`-region it causes in a frame taken by a sensor:

If the environment is static, as assumed earlier, recording a transparent object from different viewpoints causes differently shaped `NaN`-regions in the frames associated with the viewpoints. Each `NaN`-region in a frame in turn influences, along with the sensor's position, the shape of its corresponding occlusion frustum. If such frames are aligned correctly with respect to each other, it becomes possible to fuse the information gathered from the single frames to discern more about the volume of the transparent object: The *footprint* of the transparent object, i.e., the area where the object is in contact with the region of interest, needs to lie inside all `NaN`-regions and thus be part of their intersection. This observation can be extended from the footprint of the transparent object to its volume, if we look at the intersection of the associated occlusion frusta instead of just the `NaN`-regions and leads to the concept of *Mutual Occlusion*.

The term mutual occlusion defines a volume in 3D space in a common coordinate system of at least two frames, which is occluded from the viewpoints in all frames. Occlusion in this case must not necessarily be caused by transparent objects. For instance the interior volume of a solid object, e.g. a box of cereals standing on a tabletop also fits this description, since the sensors employed in our work are only able to measure the outer boundaries of such a box, while its interior is not observed in any frame. When we consider regular, i.e., non-transparent objects, the mutual occlusion caused by them is generally not of interest, since the exterior that bounds this volume can be measured directly. However, if one is inclined to determine the volumes of mutual occlusion in a scene only featuring regular objects, this is perfectly feasible: one just needs to consider the occlusion frusta, with respect to the sensor poses, bounded by the "shadow" of the objects on the `ROI` and the measurements of the objects surface on the other side, i.e., instead of having an apex at the sensor's location, part of the occlusion frusta is composed by the measured data points. Notice that this is also very similar to detecting if a hole region was caused by a regular object in subsection 3.2.1.

In the case of transparent objects the mutual occlusion becomes much more interesting. Occlusion in this case is caused by the non-measurable surface of the transparent object. Since this cannot be measured by the employed sensors the occlusion frusta are constructed, as sketched above, from the `NaN`-region and the sensor's position as the apex. Still we can conclude, that, assuming a correct alignment of multiple frames, the actual volume of the transparent object needs to be in that volume of 3D space that is part of the mutual occlusion. How these ideas can be implemented and provided with a certain level of robustness with respect to occlusion from regular objects and sensor noise will be shown in detail in subsection 3.3.2, alongside a discussion of known shortcomings. Afterwards, in 3.3.3 a slightly different approach will be introduced, that addresses most of the issues of the method proposed in 3.3.2. Section 5.1 in the experiments chapter will provide ample visualization of occlusion frusta and `NaN`-regions.

3.3.2 Intersection Based Reconstruction

This subsection will illustrate how to actually construct an occlusion frustum in a efficient way, given a frame containing at least one `NaN`-region (see 3.2) that might be caused by a transparent object and how to combine the information of multiple frames of the same scene. Therefore, we assume to have the following inputs:

- The plane equation for the `ROI`
- The pose of the sensor
- The polygon describing the outline of at least one `NaN`-region in terms of 3D point coordinates

One aspect to consider, when implementing the ideas of mutual occlusion and thus determining the overlapping parts of multiple occlusion frusta, is how to represent the volume contained in an occlusion frustum, in a way that its overlap can be determined quickly and at a level of desired precision. A straight-forward solution that comes to mind is to represent a single occlusion frustum as a 3D polyhedron. The polygon describing the `NaN`-region is projected into the plane associated with the `ROI` and therefore already a planar polygon while each line segment of the convex hull in conjunction with the sensor's location forms a triangle, thus fulfilling the requirements of a polyhedron. Note, that if it is more convenient to work with triangular meshes instead of polyhedra, the polygon that constitutes the base of the occlusion frustum can be triangulated, by some tesselation algorithm, as described, for example in [71, chapter 11]. The Computational Geometry Algorithms Library (CGAL [76]) provides for algorithms to represent polyhedra in such a way. These come alongside various operations that are of interest (see [41]), when we want to determine the overlap between multiple occlusion frusta.

The volume that overlaps between two given polyhedra is defined by their intersection. An additional nice property of the intersection operation is that it automatically takes care of erroneous false positives that are only present in individual frames, i.e., as not being part of the (total) intersection of all collected frames, they will simply disappear in the end.

As already indicated, while a representation of the volume occupied by each occlusion frustum as a polyhedron appears natural and is able to provide a precise means of expressing the volume of the mutual occlusion by sequential intersection of the collected polyhedra, this approach might become infeasible in terms of runtime (at least if we consider a soft-realtime application with a robot), with an increasing number of viewpoints. As it turns out this intuition unfortunately turns out to be true, as will be shown in the experimental results in subsection 5.1.2.

In the following a different approach, that is less precise in terms of the reconstructed volume, but more capable to fulfill the desired soft-realtime constraints is introduced. The main idea here is to represent the volume occupied by the mutual occlusion in an efficient and discretized fashion. Discretization of 3D data can be simply done via a voxel grid or if, in relation to the bounding volume, only some portions of the 3D space are occupied with relevant data, more efficiently with an octree (see subsection 2.3.2). Our use case fits the latter: The volume occupied by an occlusion frustum will, in general be much smaller than the volume of its enclosing axis-aligned bounding box. If we combine several occlusion frusta into a common coordinate frame, the ratio between space occupied by one or more occlusion frusta in relation to the enclosing bounding box will usually decrease even more, so a representation via an octree is sensible. In both cases, voxel grids and octrees, a desired resolution has to be set, which determines the edge length l_L of a voxel or octree leaf, respectively. Note, that only octrees with cubic leaves (and not cuboid leaves) are considered.

Unfortunately octrees as well as voxel grids are data structures with the purpose of representing and / or reducing 3D point data, so an occlusion frustum only consisting of the 3D points of its base polygon, describing the `NaN`-region, and the apex (sensor position) does not directly describe the desired volume in neither voxel grid nor octree, but will only produce occupied leaves that approximate the polygon and another lone occupied leaf at the sensor's location. However, by creating a point sampling of the volume occupied of the occlusion frustum it becomes possible to feed the desired information into an octree and thus employ this data structure later to generate an approximation of the desired intersection. Consequently we will investigate how to create a point sampling of an occlusion frustum in a constructive and straight forward fashion, without the need to explicitly check if a sample point is located on the inside of the given occlusion frustum.

One way to create a point sampling of a given occlusion frustum is to randomly generate 3D points that lie inside the enclosing axis aligned bounding box and check if they are also inside the polyhedron defined by the occlusion frustum. This process is repeated until a desired density of sample points in relation to volume is reached. However, given the nature that the occlusion frustum oftentimes occupies much less volume than its enclosing axis aligned bounding box, and testing if a given 3D point is inside a given polyhedron is in linear runtime with respect to the polyhedron's faces, this approach seems rather inefficient. Furthermore, if the thusly generated random sampling will be represented later in a regular data structure (i.e., the octree) I decided that it is more prudent to create a more regular sampling, albeit in a more efficient way.

The first step in construction of the sampling method that I propose consists of sampling the base of the occlusion frustum, i.e., the polygonal outline of a `NaN`-region. This is done by rigidly transforming the polygon (via a pose Θ_i) onto an axis aligned plane, say the x-y-plane, determine its bounding box in said plane, sample the bounding box in a uniform fashion and test for each sample point if it is also inside the polygon, that describes the outline of the `NaN`-region. All points that hold the latter property are subsequently transformed back into the original scene by applying the inverse transformation Θ_i^{-1}.

While this process might sound overly complicated it has some nice properties: first creating a uniform grid sampling for a given rectangle with 2D is trivial, as opposed to an uniform sampling of planar surface in 3D. Second testing if a given 2D point is inside a 2D convex hull is much easier in terms of implementation and computation than the 3D case. Determining a pose Θ_i that transforms a planar polygon from its original 3D position and orientation into the x-y-plane is on the other hand quite simple as is the back-transformation of the generated sample points via Θ_i^{-1}.

Once the NaN-region is sampled uniformly in a grid fashion, as a point cloud $P_{\texttt{NaN}}$, we can proceed to reconstruct the actual volume of the occlusion frustum. For each point $p_{j,\texttt{NaN}} \in P_{\texttt{NaN}}$ the line segment $l_j = \overline{p_{j,\texttt{NaN}} S_{\text{pos}}}$ from $p_{j,\texttt{NaN}}$ to the sensor's position S_{pos} is sampled uniformly with a sample distance smaller or equal to the desired octree resolution, resulting in point cloud P_{l_j}. If we consider all line segments sampled in such a fashion as a single (combined) point cloud $P_{F,i}$, describing the frustum, the sample density is not uniform, but increases the smaller the distance to S_{pos}, since all line segments coincide at this point.

In order to generate a point cloud of uniform point density that describes approximately the same volume as $P_{F,i}$, $P_{F,i}$ can be discretized by a voxelgrid or octree afterwards. The voxelization of $P_{F,i}$ is an just optional step that improves performance (in terms of computation time) as opposed to the point sampling of non-uniform density. Furthermore, note that the presence of multiple NaN-regions in a single frame does not pose any problem for this approach: each NaN-region is individually discretized and the associated occlusion frustum is sampled. Afterwards all sampled frusta for the current frame are combined, thus possibly creating a cloud, consisting of multiple frusta that all coincide at the sensor's potition S_{pos}. However, it is of importance that all sample points that represent the current occlusion frustum, voxelized or not, are assigned with a distinct label, that varies depending on the sensor pose Θ_i from that the scene is observed, i.e., all points in the combined point cloud need to be assigned with the same label, independent on the number of occlusion frusta detected in the current frame.

Let us assume that we have constructed for each frame a point cloud consisting of the sampled occlusion frusta in that frame and the poses Θ_i, $i \in \{0, 1, \ldots, n\}$, that align all $n+1$ frames correctly in some common coordinate frame. Under these conditions the volume of mutual occlusion can be determined by combining all frusta $P_{F,i}$ together in a combined point cloud P_{comb} and feeding P_{comb} into an octree. Once the octree is constructed, we proceed to investigate each leaf, or more precisely the (voxelized) sample points that are bounded by each leaf: any volume that is part of the mutual occlusion needs to contain points from each recorded frame, i.e., if we iterate over all points contained in a leaf we need to detect all labels in order for the leaf to be part of the mutual occlusion. Thus the intersection operation of several occlusion frusta is simplified to iterating over the octree leaves and detecting the individual labels in each volume. If the point clouds in each frame are voxelized, with the same octree resolution as the octree, employed to approximate the intersection, the intersection process can accelerated further. If an individual leaf contains less points than $n+1$, the number of frames, it cannot be part of the intersection. This essentially prevents the investigation of octree leaves that are in the close proximity of the different sensor positions $S_{\text{pos},i}$, where the non-voxelized point density increases (see above) and the algorithm would nevertheless have to iterate over all contained points to check which labels are available in the leaf.

The validity of this approach will be illustrated in the experiment chapter, more specifically in 5.1.2, where the reconstruction of a non-transparent object with its recorded data is compared and Figures of point clouds P_{comb} and the subsequent intersection are shown. In this case an artificial hole region is created by manually editing the captured point clouds and removing the measurement points that belong to a chosen regular object, thus creating the appearance of a transparent object in the scene. Afterwards reconstruction and the actual recorded measurements of the object are compared.

One very convenient property of the intersection operation, that we perform on the registered occlusion frusta, is that we do not explicitly have to model locations, in the case that multiple transparent objects are present in the observed scene. While the point sampling in the first frame forms a single cluster for all present transparent objects, since their individual occlusion frusta meet at $S_{\text{pos},i}$, this will change when additional frames are added to the reconstruction process. Since the occlusion frusta associated with point cloud P_j coincide at $S_{\text{pos},j}$ (and assuming that $S_{\text{pos},i} \neq S_{\text{pos},j}$), the volumes that contain points of both, P_i and P_j, will in general not form a single cluster anymore when multiple transparent objects are present, but several separated clusters. This separation will become sharper with an increase of the number of different viewpoints from which the scene was recorded. Thus object association breaks down to simple clustering after the volume of mutual occlusion is determined via the intersection operation.

If we employ, instead of the polygonal outline, the convex hull of a `NaN`-region, as described in subsection 3.2.2 and ignore valid measurement points located inside of the convex hull, details of the object are lost in the reconstruction. Details, like handles or concave curvatures get lost not only in the 2D projection, if the polygonal outline is substituted by its convex hull, but subsequently also in the intersection of the resulting occlusion frusta. Instead of an approximation of the precise shape of a transparent object its 3D convex hull is approximated in this fashion.

In some cases this loss of precision is not necessarily a disadvantage: if the reconstruction information is to be fed into a occupancy map, that will be used to identify obstacles for trajectory planning of a manipulator, these representations usually do not require miniscule details and in terms of collision avoidance overestimation of the size is preferable to underestimation. Since the convex hull is at least of equal size as the actual entailed `NaN`-region, namely whenever the `NaN`-region's shape already is convex or otherwise overestimates the size of the region its use in such cases is not unfavorable. Furthermore, this overestimation can provide additional robustness to the reconstruction via intersection, since `NaN`-regions oftentimes are smaller than the true perspective projection of the transparent object onto the support surface. Exemplary evidence is provided by comparing size and shape of the drinking glass located at the center of the table in the photograph of Figure 3.5a and the corresponding `NaN`-region(s) in the depth image in Figure 3.5b.

Underestimating the actual size of a transparent object once cannot be recovered from, when using the intersection of occlusion frusta as a means of reconstruction: it is inherent in this method that the volume of the mutual occlusion can successively only shrink or retain its size, starting from the volume of the initial occlusion frustum. Even worse, when in a single frame a transparent object causes two small `NaN`-regions, instead of one well shaped region that resembles its perspective projection (as is incidentally the case in Figure 3.5b), two smaller separate objects will appear in the reconstruction. Using the convex hulls in combination with their possible fusion instead of the actual outline of a `NaN`-region partially addresses this problem (see the resultant convex hull at the tabletop's center in Figure 3.8b).

Even if all issues that result in underestimation of the volume and shape of the transparent object, mentioned above, are addressed by some method or other, there is still an additional problem to consider, which might still result in bad reconstruction results: so far we have always assumed that all frames are registered correctly in some common coordinate system. However, this assumption will in general not hold, especially if we think about a (cheap) robotic platform that produces pose estimation errors in cm range (thus diluting the initial pose estimation for the registration process) and employ a common low-cost sensor like MicroSoft Kinect or Asus X-tion, that also produces measurement errors (for regular objects) that are in cm range.

If we assume that some of the transparent objects we want to reconstruct may also only measure up a few centimeters in one dimension (which is not uncommon for objects placed on a tabletop), several translation errors in the alignment estimation of 1 cm or 2 cm will have a significant impact on the resulting intersection, where parts of the occlusion frusta might (incorrectly) be removed, whereas actually empty space might be considered part of the object. A practical illustration of this problem can also be found in the experiments chapter, subsection 5.1.2. In the following I will introduce a slight modification of the intersection approach that tries to compensate for these problems, by relaxing the intersection criterion.

Relaxation of Intersection

As illustrated above, the intersection provides, based on a point sample of the occlusion frusta, a valid mechanism to reconstruct volume and shape of transparent objects, albeit sensitive to `NaN`-regions that underestimate the transparent objects perspective projection, (not handled) partial occlusions of the `NaN`-regions and registration errors. Now I will introduce an approach that deals with these problems by relaxing the strict intersection operation.

In this approach it is assumed that the decision if a given octree leaf belongs to the mutual occlusion is independent of its neighboring leaves. This was also implicitly the case while employing the (strict) intersection, but in this approach this fact becomes more obvious. In contrast to the intersection, where a octree leaf is either part of the intersection or not, the decision in this approach becomes a less sharp, meaning that voxels that do not contain sampling points from all registered frusta can still be considered part of the volume of mutual occlusion, if this particular volume is occluded in a sufficient number of other frames. This idea allows to handle small registration errors alongside problems caused by (partial) occlusion from regular objects and visibility issues near the borders of the FOV without the need to address them explicitly.

One way to express this concept in a mathematical sound way can be found in *fuzzy theory* [80], or more concise in the *T-norm*, which is defined as a function T:

$$T : [0,1] \times [0,1] \mapsto [0,1] \tag{3.1}$$

that complies with

$$\begin{aligned} T(a,b) &= T(b,a) && (3.2)\\ T(a,b) &\leq T(c,d) \qquad \text{if } a \leq c \text{ and } b \leq d && (3.3)\\ T(a,T(b,c)) &= T(T(a,b),c) && (3.4) \end{aligned}$$

Equations (3.2) – (3.4) constitute the following 3 properties of a T-norm: *commutativity* (3.2), *monotonicity* (3.3) and *associativity* (3.4). The T-norm fits naturally, when we want to be able to express a less strict classification (in terms of being part of the mutual occlusion) as opposed to the intersection. Considering only a single frame, we can still make a crisp decision, if a voxel belongs to the volume of mutual occlusion or not, namely if a voxel is part of the occlusion frusta of this particular frame it belongs to the mutual occlusion, otherwise it does not. Note that the terms *leaf* and *voxel* will be used somewhat interchangeably in this part, generally it can be assumed that a voxel also refers to a volume of a node on the lowest level in an octree. In accordance with the requirements of a general T-norm, the former case will be denoted as $\frac{1}{1}$, while the latter case will be denoted as $\frac{0}{1}$. With this considerations we can define a T-norm, $\mathrm{T_v}$ that represents the degree how much a given voxel is part of the volume of mutual occlusion:

$$\mathrm{T_v}\left(\frac{x}{n}, \frac{y}{m}\right) = \frac{x+y}{n+m} \tag{3.5}$$

Since, according to our initialization for a single frames, the terms $\frac{x}{n}$ and $\frac{y}{m}$ are either $\frac{1}{1}$ or $\frac{0}{1}$, we can easily conclude, that $\frac{x+y}{n+m} \in [0,1]$ holds and thus subsequent applications of $\mathrm{T_v}$ will also produce values in $[0,1]$. Furthermore, since the addition, used in numerator and denominator is associative, commutative and monotone, requirements (3.2), (3.3) and (3.4) are also met.

Comparing the value of $\mathrm{T_v}$ for a given voxel, after all recorded frames are registered and their information has been fused, via $\mathrm{T_v}$, with a threshold $t_{\min} \in [0,1]$ allows us to decide with a varying degree of relaxation which voxels are considered to be part of the volume of mutual occlusion. Note, that the intersection operation is a special case of employing $\mathrm{T_v}$ (with $t_{\min} = 1$) to approximate the mutual occlusion. Furthermore, desired properties of the intersection, e.g., distinguishing between several objects by clustering after determining the mutual occlusion, are retained. Unfortunately the approach employing $\mathrm{T_v}$ also has its drawbacks: reconstructed objects tend to overestimate the actual size of the object, with decreasing $t_{\min}$. As a theoretical extreme one might consider $t_{\min} = 0$, where every voxel is considered part of the mutual occlusion. If we consider how the octree is constructed, we will not actually retrieve all voxels (since empty voxel volumes do not exist in the octree representation), but all leaves that contain at least 1 sampled point of any registered occlusion frusta – thus this would not correspond with the intersection of all occlusion frusta, but with their union. But also with values for $t_{\min}$ that are of more practical use, i.e., $0 \ll t_{\min} < 1$, when trying to reconstruct shape and volume of transparent objects, the reconstruction becomes naturally less sharp, when compared to the intersection. While the parts of the occlusion frusta that are in relatively close proximity to their corresponding sensor position $S_{\mathrm{pos},i}$ are removed correctly, since they typically not overlap in multiple frames, around the actual objects location oftentimes voxels can be found that contain sample points from all but a few select frames, but do not belong the ground truth volume of the object. In this respect the visual appearance is similar to an intermediate stage in the later introduced viewpoint-based reconstruction, so the voxels of cluster `trans_obj02` in Figure 3.13a serve as a good visual impression of the slightly inflated estimation.

Determining the different labels that are present in a cluster extracted after applying the $\mathrm{T_v}$ based reconstruction and applying to this point cloud a subsequent $\mathrm{T_v}$ reconstruction, with increased threshold $t_{\min}$, to "sharpen" the clusters appearance suggests itself as an intuitive impromptu solution. While this can improve the reconstruction result in some cases, it unfortunately does not handle the inherent problem in a substantially different way. For example, if the point cluster, approximating the transparent object is composed of a total of n views, where in k of these n views the object was partially occluded and we assign $t_{\min} \geq \frac{n-k}{n}$ the resulting reconstruction will underestimate the mutual occlusion, similar to the strict intersection.

A different problem might also arise, that is inherent in the T_v, if we choose $t_{\min} < 1$, i.e, we do employ a relaxation compared with the intersection. If multiple frames of the scene are taken from a pose Θ_j that is very similar or identical with a previously used viewpoint Θ_i, i.e., the transformation $\Theta_{i \mapsto j}$ between Θ_i and Θ_j is very small in terms of translation and rotation, the shape of the reconstructed transparent objects will gradually correspond to the occlusion frusta obtained from point cloud P_i. The reason behind this, is that the occlusion frusta detected in point clouds P_j, that fit the previous description, occupy either the same, or very similar volumes in terms of occupied octree leaves. Thus, when applying T_v to extract the volume of mutual occlusion, these leaves will contain point labels from multiple different frames, namely P_i and all those P_j and hence the value of T_v will become larger than $t_{\min}$, with increasing number similar frames.

Of course, this only applies, if multiple similar frames are recorded for one or only a few select poses Θ_i: if n ($n \geq 0, n \in \mathbb{N}$) frames are taken from each pose Θ_i, the resulting reconstruction will be the same as for $n = 1$, since n will cancel itself out in numerator and denominator of T_v. Note that the effect of the mutual occlusion becoming similar to the occlusion frusta obtained from a single pose Θ_i, in general gets weaker, the larger the denominator in T_v, i.e. the larger the number of scenes taken (from different poses) that are incorporated in the reconstruction.

Handling Occlusion from non-transparent Objects

In this part I will illustrate how occlusion, caused by regular, i.e., non-transparent objects, can be handled in a way to not influence the estimation of the mutual occlusion based on the T_v. A prerequisite is a working *object detection* in single frames. In this context *object detection* is not meant as a mechanism to classify objects in the recorded scene into one of several categories (like plates, cereal boxes, books, etc. ...) or even individual instances (i.e., detecting a cereal box of a certain brand, identifying a container of orange juice, ...), but just detecting points that could potentially belong to one object (or a conglomerate of spatially close objects) in the `ROI`, which is a much easier requirement to comply with. Once the `ROI` is determined in a given point cloud (in the example case by plane detection and establishing the convex hull for the table inliers in the current frame), we can easily determine the point clusters, that are positioned spatially close to the `ROI`. While we do not know, without further investigation, what kind of object might be responsible for these measurement points, the surface that is represented by the measured points causes in-scene occlusion and thus has the potential to conceal a transparent object. In order to address this potential cloaking we can create occlusion frusta not only for the `NaN`-regions, that are detected in the current frame, but also for each regular object. Note that in this case objects clusters are not required to be located inside the `ROI`: The backrest of a chair that is place between sensor and tabletop can occlude parts of the tabletop and (transparent) objects that are placed there.

Construction of an occlusion frustum for a regular object in point cloud P_i differs slightly from its creation for a `NaN`-region. In the case of `NaN`-regions first the convex hull is computed an this is afterwards sampled in a uniform fashion (see 3.3.2) in the case of a cluster $C_j \subset P_i$ of regular measurement points the line segments $\overline{p_{k,i}\hat{p}_{k,i}}$ between cluster point $p_{k,i} \in C_j$ and its perspective projection on the tabletop $\hat{p}_{k,i}$ is sampled uniformly with a point distance below the desired octree resolution. As with the occlusion frusta created from `NaN`-regions a voxelization of this sample point cloud can be done and is recommended for a faster computation of the intersection or $\mathrm{T_v}$, respectively. Note, that all points, representing the occlusion caused by regular objects are assigned with the same label, unique for point cloud P_i as the occlusion frusta representing the potential volume of transparent objects in the same frame.

The reason for not considering the convex hull and using a uniform sampling of its area, as in the case of `NaN`-regions, is straight forward: for `NaN`-regions we did not have any information about where to cut off the occlusion frustum, thus each line segment was sampled from the point on the tabletop-plane to sensor position $S_{\mathrm{pos},i}$. However, in the case of regular measurement points $p_{k,i} \in C_j$ of point cluster C_j we can conclude, that nothing is present between the sensor position $S_{\mathrm{pos},i}$ and $p_{k,i}$. Hence, this portion of space should not be included when modelling potential locations that are occluded by transparent or regular objects and consequently only the line segment between $p_{k,i}$ and $\hat{p}_{k,i}$ is sampled. If the occlusion of point cluster C_j were represented by an uniformly sampled 2D convex hull on the `ROI`-plane, it becomes harder to determine the end point of the sampled line segments, since these will usually not go through any of the recorded measurement $p_{k,i} \in C_j$.

However, this method also has an inherent drawback. If for each point cluster in each frame an occlusion frustum is created and added to the occlusion frusta associated with `NaN`-regions, employing the $\mathrm{T_v}$ or the intersection will not only yield clusters that will denote locations of transparent objects. In addition also clusters in the interior of regular objects will emerge, since this portion of space will always be occluded by the objects surface, as already mentioned when the concept of mutual occlusion was introduced.

This problem can be addressed by a post processing step after the reconstruction of all potential transparent object volumes. Voxel clusters obtained by employing $\mathrm{T_v}$ and their 3D convex hull can be tested if they are located inside the volumes that represent a regular object. In this case we can remove these clusters from the list of locations for transparent objects in this context (although it might be true that the interior of some cardboard box contains a glass object). If the convex hull of a reconstructed transparent object does not lie completely inside the volume occupied by a regular object, the degree of overlap needs to be considered when determining if the potential transparent object should be discarded or not. Whenever the overlap involves only a small fraction of the overall volume of regular and potential transparent object and the depth of their intersection in terms of distance measure, i.e., how far one object protrudes into the other, is not too large, the transparent object candidate should not be discarded, since its convex hull might be slightly larger than the actual object and transparent and regular object might just be located close to each other. Therefore, by employing some thresholds the majority of undesired transparent object hypotheses, i.e., the mutual occlusion caused by regular objects, can be filtered. Note that a small overlap in terms of volume can still result in a large depth of intersection and a large overlap in terms of volume can result only in small depth in regard to the protrusion. Hence, both values should be considered in combination.

Unfortunately, this method also has a drawback. If the scene contains objects that feature both, transparent and regular parts, for example a glass bottle with some paper labels, the transparent property might get lost, using this filtering step. In the example of such a bottle both, volume of overlap of the convex hull and the degree of protrusion, in relation to the objects size, will be large, so that the transparent object is likely to be discarded. Depending on the size and placement of the labels, this can result in the loss of valuable information, since labels typically cover bottles only partially. Due to these considerations, this filtering step was not implemented in the results shown in chapter 5 and to simplify the modelling process only valid `NaN`-regions were used to create occlusion frusta and not occlusions form regular objects.

3.3.3 Viewpoint-Based Reconstruction

In the previous subsection the means of approximating the three dimensional shape of a transparent object, using intersection of several occlusion frusta was discussed. As experiments will show (see subsection 5.1.2), this method can create sensible results, but turns out to be sensitive against alignment errors and sensor noise. The main disadvantage of the intersection method is that once a specific volume in the scene is carved away from the volume that constitutes the reconstruction of a transparent object, there is no way to add this volume in a later stage. This is a direct consequence of the intersection operation on volumes, where the first occlusion frustum constitutes the maximal volume that the reconstruction may assume. Subsequent integration of newly acquired data may only decrease this volume or leave it identical. Apart from the sensitivity to noise and imprecise scene alignment, illustrated in the later parts of 5.1.2, this property of the intersection turns out to be problematic, if the object to be reconstructed is only partially visible in certain frames or completely absent, either via occlusion caused by a regular object or being positioned on the edges of the current field of view.

While the first problem, namely occlusion by regular objects, can be approached, as outlined at the end of the previous subsection, the second problem is more complicated to address: a transparent object, that is only partially observable in a scene might, in theory, occupy all volumes outside of the current field of view. Note, that in this context a transparent object is only fully observable, if its `NaN`-region can be distinguished from the `NaN`-region at the boundaries of the depth image that belong to previously introduced class `NaN`.1 (see Figure 3.6). Constricting the volume to something smaller already makes assumptions about shape and size of the object that is to be reconstructed and might result in an incorrect reconstruction in the end. If the information for candidate transparent object locations near the edges of the field of view are to be ignored, it becomes necessary to keep track of individual transparent objects again, since different transparent objects might be fully visible in the frame and subsequently occlusion frusta with the unique frame label will be added to the octree, keeping track of the volumes that are potentially occupied by transparent objects. Hence, one needs to identify at which locations certain labels are to be expected for the individual transparent objects, so that the absence in individual frames does not result in a removal of the object.

By relaxing the intersection operation this problem can be moderated to a certain degree, as discussed earlier, but this problem comes with its own problems, that were outlined. However, also the relaxed version of the intersection operation is not an ideal solution: if threshold t_{min} for T-Norm T_v is chosen too large, transparent objects that are well recorded in only a small number of the total frames will still be discarded. On the other hand, choosing a small value for t_{min} will result in very bloated approximations for all transparent objects that are present in a lot of views.

Keeping track of each individual object could be a way to handle this problem. In this fashion each individual transparent object would get its own octree, so that the number of frames in which one transparent object was present does not have an influence on the reconstruction result of another transparent object. The drawback of this idea is, apart from keeping track of the individual instances, that there are special cases that need to be explicitly handled, like splitting one object into two or the fusion of transparent objects. Splitting may occur, when in the initial frames the perspective projections of two transparent objects overlap, i.e., they create one larger `NaN`-region instead of two separate ones. If in later frames, where the scene is observed from different poses, the projections do not overlap anymore, the combined transparent object will need to be split into two new instances. On the other hand, fusions can also happen, for example it may happen, that one transparent object initially produces two small separate `NaN`-regions, as was illustrated earlier. If these are sufficiently apart from each other in the initial frames, it may happen that they will not be fused and two smaller transparent objects are assumed instead of one. If the object produces a single `NaN`-region, in subsequent frames, this needs to be identified and a fusion of the two transparent objects needs to take place.

A way to deal with both of these problems, i.e., avoid additional tracking potential transparent objects in the scene, while also preventing to give too much weight to information obtained from very similar poses presents itself in the *viewpoint-based reconstruction* (VPBR) that I will introduce next. The basic idea behind this concept is simple: a volume that is part of several occlusion frusta becomes part of a hypothesized transparent object, if it was observed from enough poses that are distinct enough from each other. In this way, viewpoint-based reconstruction differs from the intersection method. In the former the information of the first frame(s) will not be enough to hypothesize a transparent object and approximate its shape (even though the occlusion frusta specify an upper limit to the volume that can be occupied by transparent objects). Reconstructions will only emerge after enough evidence is gathered. In case of the intersection based method the initial occlusion frusta serve as object approximations and these volumes are reduced in subsequent frames (note that in the relaxation with the T_v the approximated volumes may later also increase in size again).

A volume needing confirmation from several different viewpoints, via being part of multiple occlusion frusta has several benefits. Firstly it improves robustness against hole regions that may be the result of sensor noise or similar effects, since these are usually either present only in individual frames, or if caused by external effects like intense sunlight, have only an effect when the scene is observed from certain points of view. Furthermore, this allows for the reconstruction to grow, if evidence from sufficiently different frames is gathered that supports a given volume being part of a transparent object, even though it was not part of all occlusion frusta constructed till that moment. As mentioned above, one consequence of this approach is that initially there will not be an approximation of the shape and size of a transparent object available, but this will only be created, if enough different confirmations are detected. However, an upper limit concerning size and shape is available nevertheless by the union of all aligned occlusion frusta.

In order to measure how distinct two different poses are some external reference concerning the orientation of the sensor in relation to the focus of the scene is needed. If we assume, that transparent and regular objects are located on a tabletop and the sensor is mounted on a mobile

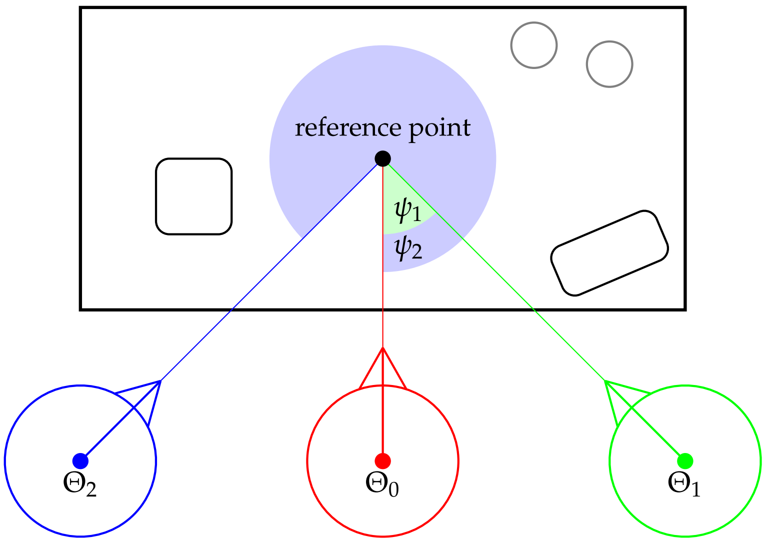


Figure 3.9: Illustration of angle ψ to distinguish between different poses in viewpoint-based reconstruction. Robot poses are represented as colored circles, where the attached edge symbolizes the robot's front.

robot, such a reference may, for instance, be established via the robot localizing itself in a map, that also incorporates the gathered information about the table and the center of the tabletop might serve as the reference point. Assuming that we gather data about the table and its contents by driving to various positions that are located around the table while always recording data, when the robot is positioned towards this center point, for each view an orientation angle ψ in relation to the first view that was gathered can be estimated, hence $\psi \in [0, 2\pi)$. An illustration for ψ is provided in Figure 3.9 This estimation of angle ψ serves as a means to determine how "distinct" the information from different frames is. Note that, different from the illustration in Figure 3.9 the reference point is not required to be directly in front of the robot.

If the robot does not move or moves very little between two gathered frames, the estimation for ψ also does not change, hence, it can be used to denote that the newly acquired data was not collected from a substantially different viewpoint. Instead of using a unique label for the sample points that represent the occlusion frustum in each frame, this label is now used to represent ψ. This means that the labels that are attached to the sample points in two different frames may be the same, if the robot did not move between the recording of both frames.

To utilize this information about the differences in the poses during data acquirement for the reconstruction of transparent objects two additional decisions need to be made: firstly how distinct should two positions be, in order to assume that their respective information does not overlap and, secondly, how much confirmation from distinct poses is needed in order to assume that there is a transparent object at a specific location and we were able to approximate its shape. The first choice defines a parameter w_{vp} with $0 < w_{\text{vp}} \leq 2\pi$ and the second defines a threshold $\mathrm{I}_{\min}$ ($0 < \mathrm{I}_{\min} \leq 2\pi$). When determining, if a given leaf of the octree that aggregates all occlusion frusta is considered part of a transparent object the following steps are performed:

For each label ψ_i that is detected inside the current leaf a corresponding *viewpoint interval* $\mathrm{I}_i = \left[\psi - w_{\mathrm{vp}}, \psi + w_{\mathrm{vp}}\right]_i$ is created (w_{vp} effectively defines the width of the interval). If necessary, these viewpoint intervals are split, so that they are all located inside the range defined for the orientation angle ψ, i.e., $[0, 2\pi)$. Once all viewpoint intervals for the given leaf are created in this way their unions are computed, meaning that from two overlapping intervals I_i and I_j a single new interval is created, that inherits the smaller lower bound and the bigger upper bound. Of all intervals remaining after unification, the sum of their size is computed and checked if it is larger than parameter $\mathrm{I}_{\mathrm{min}}$, i.e., if we denote lower and upper bounds for an interval I_i as $[l_i, u_i]$ and use $\|\mathrm{I}i\| := u_i - l_i$ to denote the width of interval Ii, then the following inequality is considered:

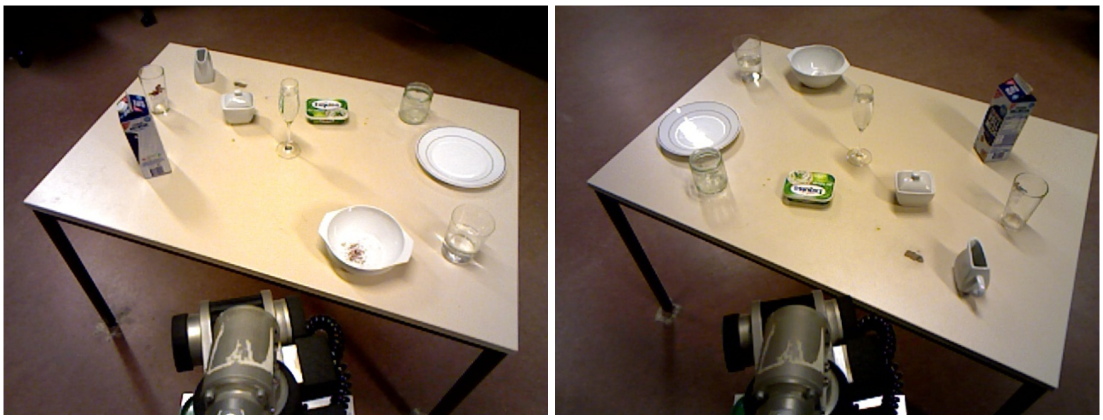
$$\mathrm{I}_{\mathrm{min}} < \sum_i \|\mathrm{I}_i\| \tag{3.6}$$

In this way the contribution of two frames that are similar, hence their viewpoint intervals largely overlap, to the term $\sum_i \|\mathrm{I}_i\|$ is smaller than the contribution of two distinct frames, where the intervals do not overlap. The choice of threshold $\mathrm{I}_{\mathrm{min}}$ in relation to the interval width w_{vp} directly models the minimal required number of views, where the corresponding intervals do not overlap, that are required in order to assume that a given volume is part of a transparent object, i.e., the minimal number of frames needed to give an approximation of the shape of any transparent object in the scene is given by

$$\mathrm{v}_{\mathrm{min}} = \left\lceil \frac{\mathrm{I}_{\mathrm{min}}}{2w_{\mathrm{vp}}} \right\rceil \tag{3.7}$$

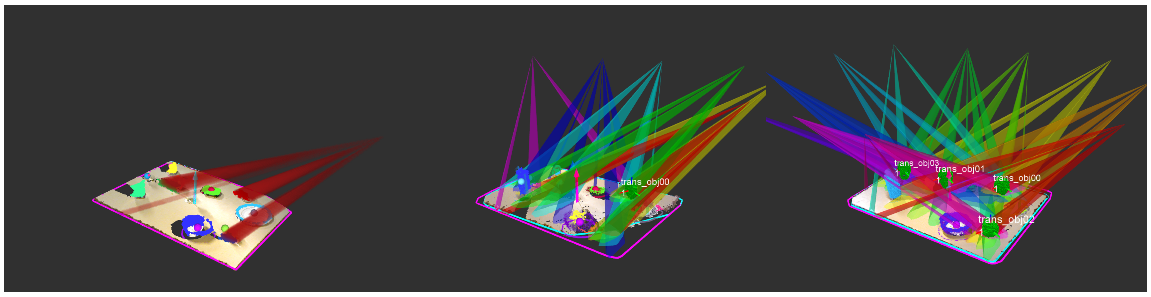
Modelling the reconstruction in this way not only addresses problems with sensor noise, where inequality (3.6) will usually be violated, but also allows to reconstruct transparent objects that were not present in every view, as long as occlusion frusta from some sufficiently different position were acquired of them.

An illustration for the progression of VPBR is provided by Figures 3.11 and 3.12, where a reconstruction of 4 glass object is performed (2 photographs of the actual scene provided in Figure 3.10). In both, Figures 3.11a and 3.12a, the occlusion frusta are represented as octree leaves, where translucent red cubes indicate leaves that do not satisfy inequality (3.6) and therefore are not considered part of a transparent object. In the subsequent images occlusion frusta are visualized as polyhedra, constructed from the convex hulls or detected hole polygons and the sensors location. Note that this is only an additional visual representation and the actual reconstruction is done via investigating the octree leaves, as detailed earlier, but for the visualization of a multitude of occlusion frusta, this representation improves visibility. A magenta colored polygon shows the estimated convex hull of the tabletop, while its cyan colored counterpart displays the convex hull of the table inliers, visible in the current frame. The magenta arrow, perpendicular to the tabletop, marks the middle of the tabletop and serves as reference point for the creation of the viewpoint intervals (see also Figure 3.9). Reconstructed transparent objects are represented by their corresponding octree leaves in green.



(a) **(b)**

Figure 3.10: Photographs of two frames during viewpoint-based reconstruction.

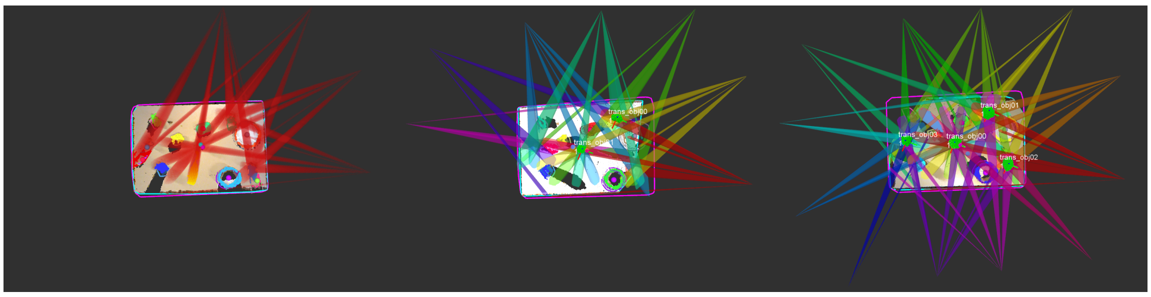


(a) **(b)** **(c)**

Figure 3.11: Progression of viewpoint-based reconstruction: **(a)** shows occlusion frusta in initial frame, **(b)** the frusta and approximation for the first transparent object at frame #6 and **(c)** shows occlusion frusta and reconstructions after 11 frames. Photographs of the scene available in Figure 3.10.

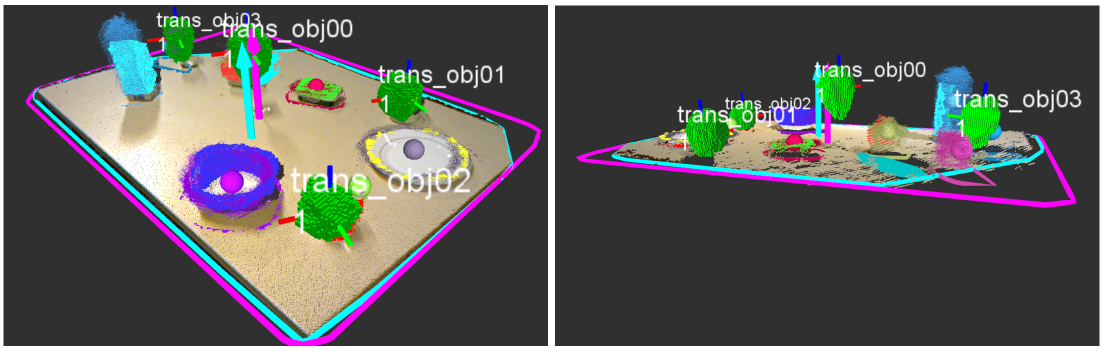
The scene for this example sequence contains 4 transparent objects: Two regular drinking glasses, one Champagne glass and a canning jar (see Figure 3.10). As Figures 3.11 and 3.12 show, not each of these transparent objects need to be observed in every frame. For instance, in Figure 3.11a the drinking glass on the left side of the tabletop shown in Figure 3.10a was not detected, since its `NaN`-region overlapped with the large `NaN`-region at the frames boundaries. Observation of the occlusion, caused by this glass in subsequent frames allows for a reconstruction, as opposed to the strictly intersection based reconstruction approach.

Applying Euclidean clustering on leave centers that are considered to belong to a transparent object, after gathering at least v_{min} frames, has two benefits: first, of all it can tell us how many transparent objects are estimated to be in the scene, given the current observations and specify their locations. This allows to avoid tracking the number of candidates for transparent objects and tracking their location in the scene. Second, in the unlikely event that a false positive occurs, caused by multiple instances of sensor noise in separate frames at a consistent



(a) **(b)** **(c)**

Figure 3.12: Progression of viewpoint-based reconstruction, top view: **(a)** and **(b)** show occlusion frusta after 4 and 7 frames, respectively. **(c)** shows occlusion frusta after collecting 12 frames. In addition **(b)** and **(c)** display reconstructed transparent objects. Photographs of the scene available in Figure 3.10.



(a) **(b)**

Figure 3.13: Results of viewpoint-based reconstruction of the scene depicted in Figure 3.10. Transparent objects are depicted as the octree leaves where viewpoint intervals fulfill inequality (3.6).

location, that satisfies inequality (3.6), a minimal cluster size can likely remove such an instance. Minimal cluster sizes are common, when using Euclidean clustering, and the probability of a false positive of comparable size to actual reconstructions is low.

Transparent objects, reconstructed in this fashion are displayed in Figure 3.13, which corresponds to the scene from Figure 3.10. The depicted results are the accumulated over a total of 12 frames, where the poses to collect data were arranged around the tabletop (see Figure 3.12c). While the 4 transparent objects, present in the scene, have been reconstructed, no false positive was created, even though some where present in individual frames (for example see Figure 3.11a, small region at the edge of the tabletop). Furthermore, one can observe that the reconstruction of a Champagne glass (`trans_obj00` in Figure 3.13) floats above the tabletop. This effect is caused by the thin stem of this type of glass, which often will not produce any `NaN`-points in a depth image. However, with the static scene assumption, we may deduct that something will support the reconstructed part of the glass above the tabletop.

Implicitly this reconstruction method also becomes more robust against error in frame

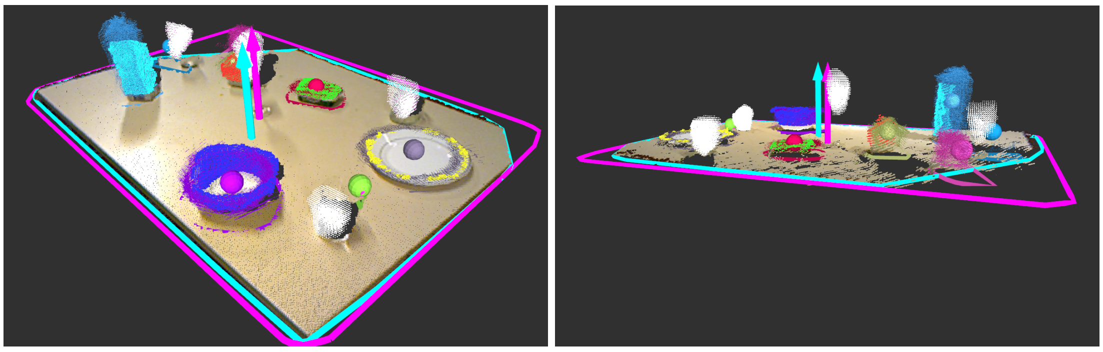


Figure 3.14: Results of viewpoint-based reconstruction after median filtering was applied on the extracted clusters in Figure 3.13

alignment, since one or two misaligned frames are not sufficient to satisfy inequality (3.6) for a given leaf if it was not also included in the occlusion frusta of several other frames. On the other hand, the absence of the viewpoint interval of a few select frames at a true location of transparent volume, will not prevent its reconstruction, if it was observed from other poses (that were correctly aligned).

When compared to the results obtained by the intersection method, if we assume that frame alignment and sensor noise did not interfere with the latter, it is noticeable that the shape of the reconstruction via the viewpoint-based method tends to be less clear-cut, featuring less sharp edges, but inducing roundness in most places of the objects contour. An example of this property can be observed in the foreground of Figure 3.13a with the cluster labeled as `trans_obj02`.

One heuristic approach to slightly moderate this effect is to perform an additional filtering on the individual clusters, again based on the accumulated viewpoint intervals in the octree leaves. First the median value μ_{I} of the sums $\sum_i \|\mathrm{I}_i\|$ of the viewpoint intervals in each Euclidean cluster is computed. Subsequently only those leaves of a cluster are retained, where the sum of the viewpoint intervals exceeds the term $f_{\mu_{\mathrm{I}}} \cdot \mu_{\mathrm{I}}$, where parameter $f_{\mu_{\mathrm{I}}} \in \mathbb{R}, 0 < f_{\mu_{\mathrm{I}}}$.

Results of this post-processing step are shown in Figure 3.14, where the median filtering on viewpoint intervals has been applied to the 4 transparent objects clusters shown in Figure 3.13. Retained octree leaves are symbolized by their center points (shown in white) in Figure 3.14. More reconstruction results of the viewpoint-based method are presented in detail with a quantitative analysis in the experiments chapter, subsection 5.1.3.

The avid reader will have noticed, that the measure of "distinctness" between two robot poses is denoted as a single angle ψ in relation to some reference point. This is only a valid simplification if some additional constraints can be made to the context in which VPBR is used, which have not been mentioned, so far: firstly it assumes that the robot is driving on a planar surface, so that its pose can represented by 2 coordinates in the plane and a single orientation angle. If the surface on which the robot moves is not planar, or the robot has other movement capabilities, this simplification becomes invalid. Furthermore, it also assumes that the sensor is attached at a fixed position on the robot. If the orientation of the sensor or its position in relation to the robots frame could be modified, again the simplification would not be sufficient (for instance the scene could be observed from different heights and with different pitch angles, although ψ remains constant). Lastly it also assumes that the robot stay in roughly the same vicinity to the tabletop – otherwise this would also need to be modeled, since the occlusion frusta caused by transparent objects will change, if a scene is observed from the same angle ψ, but at different distances to the reference point.

If any of there constraints is violated different measures would need to be used in order to distinguish between poses and the simple scheme that utilizes the sum of the viewpoint intervals in order to assume that a given leaf is part of a transparent object would also need to be extended accordingly. Such extensions however were not investigated in the scope of this thesis.

Chapter 4

Confidence Measure for ICP-based Registration

When registering full 3D point clouds, the *ICP* algorithm [8] is highly popular. Applications of ICP range from registration of multiple scans recorded from real world data to object recognition tasks, aligning 3D CAD-models into acquired real world data. While the algorithm is extensively used by many research groups, it is a heuristic approach, that is only guaranteed to converge in a local minimum. Surprisingly, for such a popular method, a concise and automatic method to compare different results, for example when using different parameters for the ICP algorithm, in terms of quality, in the absence of ground truth is still lacking. In this chapter I will introduce an error / confidence measure for the alignment obtained by ICP or similar methods of alignment, thus providing a first step towards reliable automated rejection of unlikely registrations. Similar to the reconstruction of transparent objects from 3D data (see chapter 3) the principle of *Mutual Occlusion* or in this case *Mutual Observability* is used, alongside some general information about the employed sensors, in order to generate a measure that enables meaningful automated comparison of 2 scans aligned by different poses that considers viewpoint restrictions and in-scene occlusions.

Section 4.1 will provide an introduction into the general problem setting, i.e., detail what information about the alignment is available from regular ICP and proceed to illustrate, with the help of suitable examples, why this information is in general not sufficient to decide which alignment to prefer, if faced with alternative alignments or when to decide that a proposed (single) alignment does not comply with the supposedly correct alignment.

In section 4.2 an overview of approaches that are similar in spirit is given. However, this section is rather brief, since not much work about this problem has been published until today.

Afterwards 4.3 explains the concepts of *Mutual Observability* in terms of 2 aligned scans and its relation to the concept of *mutual occlusion* (thus highlighting the similarity to the application of reconstruction of transparent objects), while sections 4.4 and 4.5 proposes two slightly different approaches to employ the concept of Mutual Observability to generate a meaningful confidence measure. Lastly this chapter will be concluded with section 4.6, which discusses the shortcomings of the proposed measures.

4.1 Problem Illustration

As already introduced in subsection 2.3.1, the ICP algorithm aligns 2 point clouds, by reducing an error in point to point correspondences, until a local minimum is reached. In order to simplify the notation in this chapter I will assume that sensor S_1, which recorded point cloud P_1, defines the reference coordinate system (model) and ICP will be used in order to align P_2 (data) with this reference frame.

This implies that the initial pose estimation for S_1 consists of a 4×4 identity matrix and the initial pose estimation Θ_{init} refers to the position and orientation of sensor S_2 in this coordinate system. Note that the assumption that S_1 always defines the reference coordinate frame is not a hard requirement – if point cloud P_1 itself is embedded in a different reference coordinate system, this can be accomplished by applying the appropriate transformation as a post-processing step.

Given 2 point clouds (P_1 and P_2) the output generated by ICP in commonly used free implementations (like 3dtk [82] or pcl [64]) consists of the following three components that are relevant in this context:

Θ_{ICP}	a pose denoting the estimated alignment between P_1 and P_2; if the employed ICP variant returns a pose correction Θ_{corr} to the initial pose estimate Θ_{init} that embeds P_2 into the reference coordinate system, then Θ_{ICP} can be computed as $\Theta_{\text{ICP}} = \Theta_{\text{corr}} \cdot \Theta_{\text{init}}$.
e_{avg}	the average distance between the point-to-point correspondences, when P_1 and P_2 are aligned via Θ_{ICP}. $e_{\text{avg}}$is also seen as the *average error*, since for perfectly aligned identical point clouds it should become 0
$\lvert C_{1,2}\rvert$	the number of point-to-point correspondences, given that P_1 and P_2 are aligned via Θ_{ICP} and the given ICP parameter $d_{\max}$, that determines the maximal distance between points $p_{i,1}$ and $p_{j,2}$ to build a correspondence (refer to section 2.3 for more details)

Alternatively to $|C_{1,2}|$ some implementations instead return $\frac{|C_{1,2}|}{|P_2|}$, i.e, the ratio of points $p_{j,2}$ that are part of a point-to-point correspondence.

Both output parameters e_{avg} and $|C_{1,2}|$ are bounded ($0 \leq e_{\text{avg}} \leq d_{\max}$ and $0 \leq |C_{1,2}| \leq |P_2|$), for a given point cloud P_2 and ICP parameter $d_{\max}$. Given these outputs it might seem strange, at first glance, why they are not suitable to built a measure that tells us about the quality of the proposed alignment Θ_{ICP}, or might tell us, if we have an alternative alignment Θ'_{ICP}, obtained by using different parameters, when executing ICP, which pose might be preferable. Especially if we consider a scenario with Θ_{ICP}, an alternative alignment Θ'_{ICP} and the associated errors and number of correspondences (e_{avg}, e'_{avg}, $|C_{1,2}|$ and $|C_{1,2}|'$) and the following inequalities hold it seems clear which pose to prefer:

$$e_{\text{avg}} \leq e'_{\text{avg}} \tag{4.1}$$

$$|C_{1,2}| \geq |C_{1,2}|' \tag{4.2}$$

Inequality (4.1) states that the average point-to-point error when using the alignment from Θ_{ICP} is smaller compared to the alignment proposed by pose Θ'_{ICP}. In addition (4.2) expresses

	Θ_1	Θ_2	Θ_3
$d_{\max}$	0.1 m	0.25 m	2.5 m
e_{avg}	0.055 m	0.064 m	0.388 m
$\lvert C_{1,2}\rvert$	31518	96489	114250

Table 4.1: ICP output parameters for varying $d_{\max}$ values; $|P_2| = 114488$.

that the number of point-to-point correspondences, when using Θ_{ICP} is larger as compared to Θ'_{ICP}. Therefore, it seems natural to assume that Θ_{ICP} should be preferable to Θ'_{ICP} in terms of correct point cloud alignment. Unfortunately this assumption does generally not hold, which will be shown in the following. Furthermore, if we are confronted with a case where inequality (4.1) holds, but (4.2) does not (or vice versa) it also becomes intuitively unclear which pose estimation might be preferable in terms of "correct" point cloud alignment. The term "correct" alignment in this case refers to the alignment that a (trained) human observer will likely choose to be closer to global minimum of the ICP optimization, i.e., the ground truth alignment.

In addition, if it is not the goal to decide which of two alignments (Θ_{ICP}, Θ'_{ICP}) is preferable, but we are interested how well P_1 and P_2 are aligned, given only one pose, it is not clear how this can be deduced from the output parameters e_{avg} and $|C_{1,2}|$, since these values are also highly dependent on the choice of input parameters $d_{\max}$and if P_1 and P_2 were previously filtered via an octree / voxel grid. In order to illustrate and substantiate the claims I made, I will present two small examples in the following:

Inequalities (4.1) and (4.2) seemingly provide a good criterion, if they hold, that pose Θ_{ICP} is preferable to Θ'_{ICP} in terms of alignment quality. Unfortunately in many real application cases these two inequalities do not always hold simultaneously. This is illustrated by a first example, where two 3D laser scans are registered. Scans were obtained by a tilt mounted SICK LMS-200 laser scanner and subject to post-processing that removed measurement points that belonged to the robot platform that carried the sensor. The ICP algorithm was called three times – in all cases with the same initial pose estimation $\Theta_{\text{init}}$was used and the ICP parameters only varied in the maximal distance $d_{\max}$, the upper limit for point-to-point correspondences between P_1 and P_2. To distinguish the three alignments resultant by varying parameter $d_{\max}$ in the ICP algorithms, these are referred to in the following as Θ_1, Θ_2 and Θ_3 respectively. An overview of the input parameters and the output parameters, provided by ICP, is shown in Table 4.1.

The results in Table 4.1 are a typical case, that highlights that inequalities (4.1) and (4.2) often do not hold simultaneously. Since the upper bound for e_{avg} is defined as $d_{\max}$, increasing the latter will often lead to an increase in the corresponding e_{avg}. However, since a larger threshold $d_{\max}$ allows for more point-to-point correspondences between P_1 and P_2, not surprisingly it often also results in an increase of $|C_{1,2}|$. This behavior is represented in Table 4.1: while the smallest chosen $d_{\max}$ (0.1 m) also produces the smallest average error, the number of point-to-point correspondences is rather small, considering only slightly more than 1/4 of the points in P_2. Increasing $d_{\max}$ to 0.25 m produces the alignment via Θ_2, employing now more than 5/6 of the possible number of correspondences, at a slightly increased average error ($e_{\text{avg}}2 =$

0.064 m). Increasing $d_{\max}$ again (in this case substantially to 2.5 m) now established point-to-point correspondences for almost all available points in P_2, but compared to poses Θ_1 and Θ_2 the average distance between point-to-point correspondences also increased substantially.

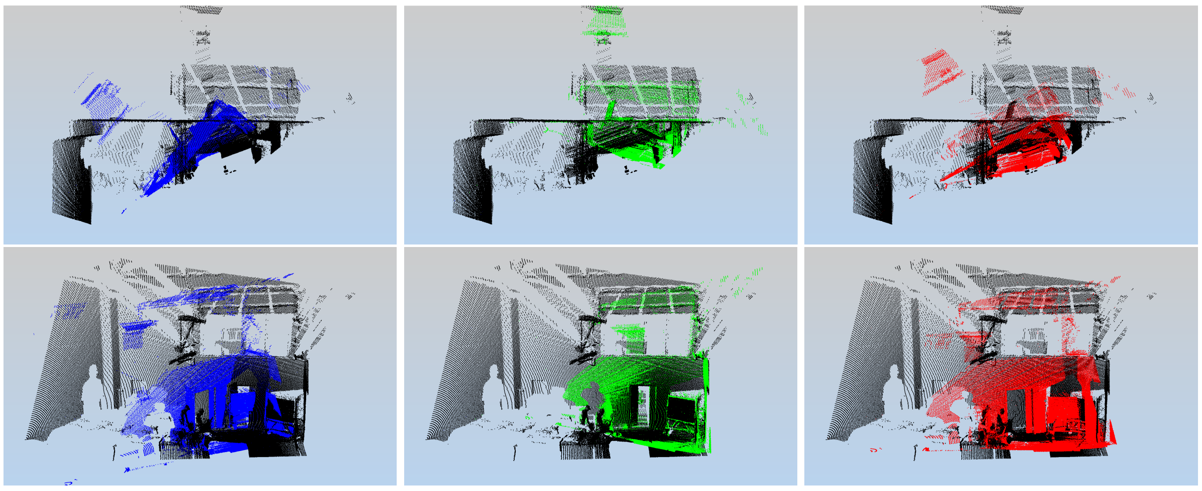
Considering the results displayed in Table 4.1, one might assume that the estimated alignment provided by pose Θ_2 is the most preferable. Expecting that there is a trade-off between e_{avg} and $|C_{1,2}|$ (one extreme would be to set $d_{\max} = 0$ m and subsequently produce a 1 step ICP alignment with $|C_{1,2}| = 0$ and $e_{\text{avg}} = 0$ m, where the final pose corresponds to pose Θ_{init}), Θ_2 seems promising, since the average distance between the corresponding points is only slightly larger than in the alignment provided by Θ_1, but much more correspondences are established. However, the values themselves do not tell us, if this alignment provided by Θ_2 denotes an alignment that a human observer would categorize as “correct” or weather poses Θ_1 and Θ_3 also provide alignments that could still be considered (sufficiently) “correct” or not.

The visual representation of alignments Θ_1, Θ_2 and Θ_3, provided in Figure 4.1 answers these questions much better, than one is able to discern from the entries in Table 4.1. In this case intuition was correct and Θ_2 is indeed the most preferable pose from all given registrations, evident in the much better alignment of the large wall segment (top row of Figure 4.1). It also becomes evident that poses Θ_1 and Θ_3 do not resemble sensible alignment estimations. While this example satisfied the initial expectation, i.e., Θ_2 denotes the best of all three alignments, the question remains if this result generally holds.

In this case it might be possible to define a quality criterion based on some combination of e_{avg} and $|C_{1,2}|$, at least in order to decide, which alignment is the most preferable, when provided with several alternatives. Still it is worth noticing, that the variation of only one parameter $d_{\max}$ can heavily influence the results of the ICP algorithm, not only in terms of output parameters e_{avg} and $|C_{1,2}|$, but also in the resultant alignment. Note furthermore that, depending on the recorded data and initial pose estimate, values for $d_{\max}$ like 0.1 m (as used for Θ_1) is a realistic ICP input parameter that can often produce very sensible results. Unfortunately the next example will demonstrate that combining the values of e_{avg} and $|C_{1,2}|$ for alignment comparison is not sufficient and that, depending on the recorded data and initial pose estimate, not even if both inequalities (4.1) and (4.2) are fulfilled we can conclude that Θ_{ICP} is preferable to Θ'_{ICP}.

In the following example I recorded 2 scans of an office hallway (see 4.2) using a SICK LMS-200 mounted on a rotation unit, i.e., produces scans with a horizontal opening angle of 360 °. The resultant point clouds P_1 and P_2 are shown in Figure 4.3. In Figure 4.3 point cloud P_1 is colored in cyan and P_2 in magenta, which will serve in the following as the default color scheme when 2 point clouds are aligned. The two pillars present in both scans (cf. Figure 4.2) are marked in matching colors (orange and green), to aid correct data association for the reader. A self-filter for the scanning platform was employed, resulting in the empty footprint denoting the sensor's position in the scans. For a better visual presentation of both point clouds the points belonging to the ceiling were removed in both point clouds.

Hallways tend to be challenging environments for scan matching algorithms, since they often lack distinct geometric features, whether these are explicitly represented in the matching algorithm or not, and are especially prone to errors in the estimation of the translation along the corridors main axis.Instead of varying input parameter $d_{\max}$ the initial pose estimate Θ_{init}, another crucial parameter of the ICP algorithm was varied. In one case a roughly correct initial



(a) Final pose Θ_1 **(b)** Final pose Θ_2 **(c)** Final pose Θ_3

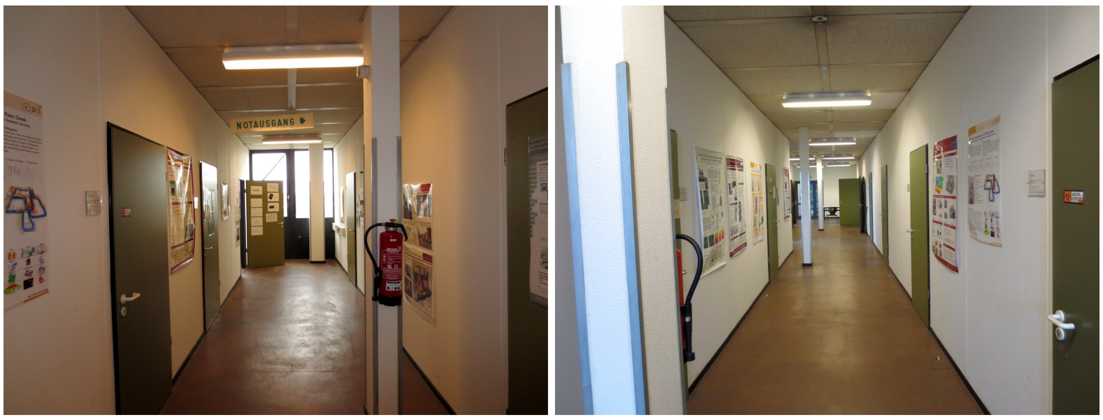
Figure 4.1: Alignments obtained by ICP: Top row provides a topview while bottom row depicts the point clouds from the side. P_1 is always colored in black, while the color of $\Theta_i \cdot P_2, i \in \{1,2,3\}$ is dependent on the estimated final pose. Θ_1, Θ_2, Θ_3 corresponds to blue **(a)**, green **(b)**, red **(c)**, respectively.

pose was given (hand measured, with translation errors in x and z dimension between 0 m and 0.1 m (error in y dimension was negligible, since the floor was parallel to the x-z-plane and the same sensor setup was used for matched scans). The hand estimated location of the second scan position was translated by approximately 0.4 m in x-direction and 4.5 m in z-direction. Although the maximal measurement range of the SICK LMS-200 is specified as up to 80 m, dependent on the reflectivity of the contact surface, meaningful information for scan matching is only recorded to a distance of approximately 8 m in the depicted hallway, mainly due to low point density and the resultant gaps between neighboring measurement points. In regions that are located farther away from the sensor than this distance the point density in the point cloud becomes too low to contain much usable information.

For the first purposefully wrong initial pose estimate $\Theta_{\text{trans\_error}}$ it was assumed that the scanner had not moved between scans, thus resulting in a translation error, along the main axis of the corridor. A second initial pose $\Theta_{\text{rot\_error}}$ was provided, where a rotation error around the z-axis of roughly 180° was induced. To underline the influence that pre-processing can have on the output parameters of ICP the alignment was also performed, using the good initial pose estimation and a down sampled version of P_1 and P_2. Downsampling was done by an octree with a leaf size of 0.05 m.

While both errors in pose estimation might seem a little extreme they underline that ICP will, in some cases, produce estimated alignments that might seem convincing at first glance, although they are actually quite far of from "ground truth". The resulting output from ICP for the unreduced point clouds is shown in Table 4.2 and the aligned point clouds are depicted in Figure 4.4.

Employing the same reasoning as in the beginning of this chapter, i.e., smaller values for e_{avg} and larger values for $|C_{1,2}|$ are desirable, then it follows from the ICP outputs in Table 4.2 that



(a) **(b)**

Figure 4.2: Photographs of the office corridor, recorded for example II, displaying the corridor in opposite directions. The position from which (a) and (b) were taken correspond roughly with the positions were P_1 and P_2 (see Figure 4.3) were recorded.

initial pose	$\Theta_{\text{good\_init}}$	$\Theta_{\text{trans\_error}}$	$\Theta_{\text{rot\_error}}$
e_{avg}	4.73 cm	2.81 cm	3.57 cm
$\|\|C_{1,2}\|\|$	65257	84042	79985

Table 4.2: Comparison of ICP outputs, dependent on initial poses: $\Theta_{\text{good\_init}}$ denotes a good initial pose, while $\Theta_{\text{trans\_error}}$ and $\Theta_{\text{rot\_error}}$ refer to erroneous initial poses.$|P_2| = 150513$.

the alignment obtained from initial pose $\Theta_{\text{trans\_error}}$ is the best available: its average point-to-point distance is smaller when compared to the other alignment estimations and it established more point-to-point correspondences. If we consider the visual representation of the proposed alignments (Figure 4.4) all three proposed alignments seem sensible at first glance, since walls, floor and ceiling are well aligned, so this case is less obvious than the previous example.

However, if we take a closer look at Figure 4.4 a human observer (who is familiar with 3D data) can quickly recognize the areas where the alignment proposed by ICP went wrong: structures like the pillar and the lights mounted under the ceiling are matched pretty well with the good initial pose estimate (see Figures 4.4a and 4.4b), while the measurement points of the same structures from P_1 and P_2 do not coincide, if we look at the registration results starting from the erroneous initial poses. Please refer to Figures 4.4c and 4.4d for results from initial pose $\Theta_{\text{trans\_error}}$ and to Figures 4.4e and 4.4f for the alignment with initial pose $\Theta_{\text{rot\_error}}$, respectively.

In environments that are geometrically very similar, when observed from different poses, as hallways tend to be, if we move a sensor along their main axes, the objects that “stick out” are used by humans as features that help to distinguish between similar, but distinct locations. In the case of the presented corridor example pillars, lights mounted beneath the ceiling and open doors can serve as such features. For the reader’s convenience such features that are visible in

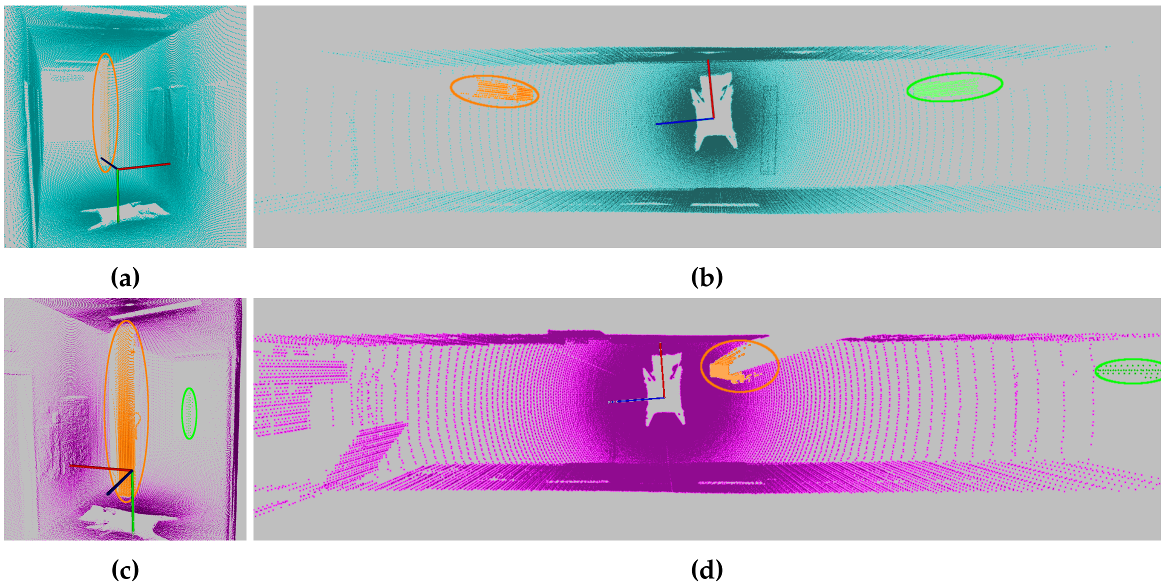


Figure 4.3: Visualization of point clouds of the corridor depicted in Figure 4.2, taken by rotating LMS-200 scanner. Point cloud P_1 is shown in cyan P_2 in magenta. Left column (**(a)** and **(c)**) shows views from inside the corridor. The viewing direction in **(a)** is roughly similar Figure 4.2a and **(c)** to Figure 4.2b. In the right column **(b)** and **(d)** display top-down views of the point clouds.

the depicted scans and correspond to each other in the proposed registration or hint at wrong alignment are marked with ellipses in Figure 4.4.

Unfortunately it is highly specific to the environment which parts of it can be considered distinct features, so the decision (as a human observer) to ignore the majority of points, when comparing the alignments shown in Figure 4.4 already reflects domain specific knowledge and this approach will fail in a lot of other environments. Ideally a confidence measure or error metric should be able to detect false alignments, as induced by initial poses $\Theta_{\text{trans\_error}}$ and $\Theta_{\text{rot\_error}}$, in generic situations, without prior knowledge about domain specific features.

Apart from the fact that the disparity between the "correct" initial pose $\Theta_{\text{good\_init}}$ and its faulty counterparts, $\Theta_{\text{trans\_error}}$ and $\Theta_{\text{rot\_error}}$, is rather large (rotation errors of roughly 180° should not happen on a real robot) one could criticize that the data as used also has an influence on the presented results (i.e., wrong initial poses generating "better" ICP output). For instance if we take a look at the data presented in Figure 4.3 it is clear that the point density in the vicinity of the sensor is much higher when compared with the point density near its maximal measurement range. Such a behaviour is inherent for all sensors that emit discrete measurement signals with a fixed angular resolution from a single source. As such it seems natural that with $\Theta_{\text{trans\_error}}$ as initial guess (where it was assumed that the sensor did not move between scans), a large number of point-to-point correspondences with very little distance between them can be established, since the point density of both P_1 and P_2 is high near the position of sensor S_1 and S_2, respectively. One attempt to reduce the influence of higher point density near sensor positions $S_{\text{pos},1}$ and $S_{\text{pos},2}$ is to employ a voxelization prior to applying ICP.

This is often done via an octree (although a voxel grid is equally valid), as outlined in

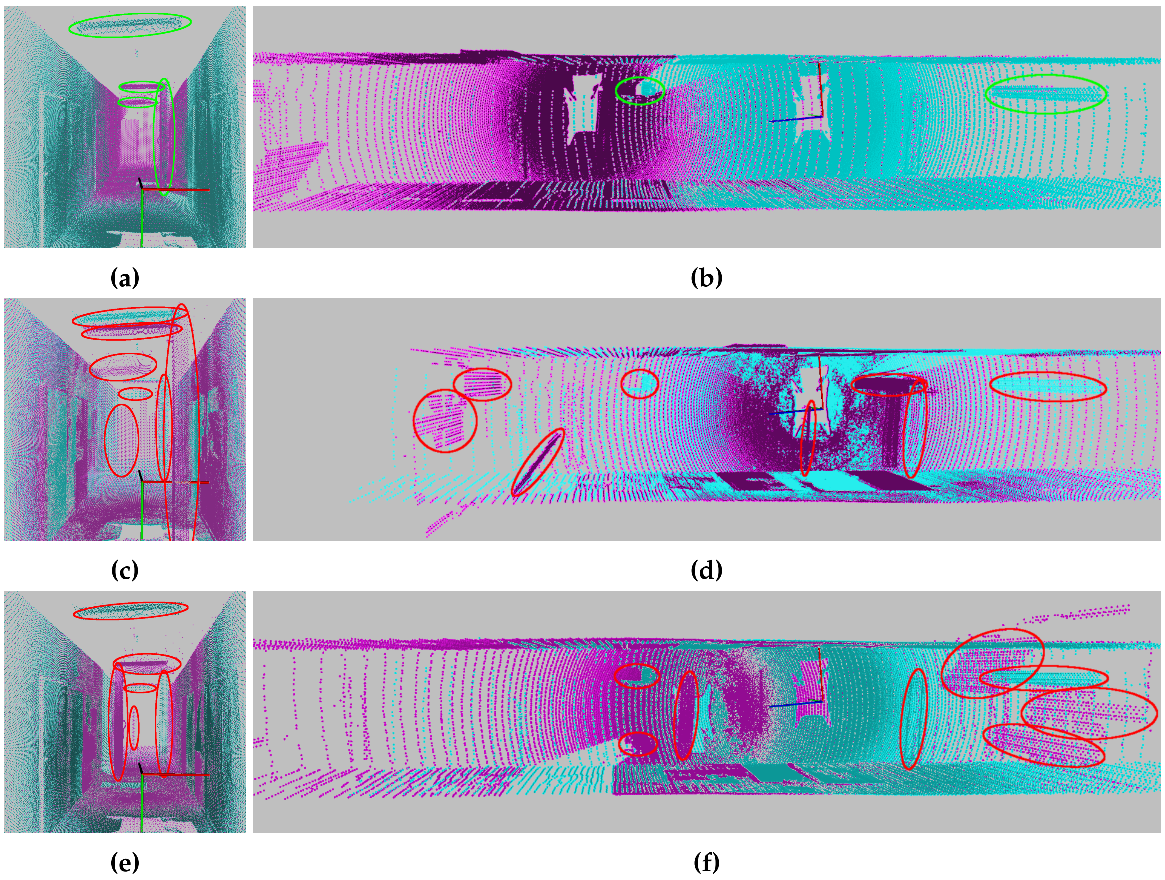


Figure 4.4: Visual comparison of registration results. Top row displays final pose for matching with initial pose $\Theta_{\text{good\_init}}$, middle row with initial pose $\Theta_{\text{trans\_error}}$ and bottom row with pose $\Theta_{\text{rot\_error}}$. Ellipses mark distinct parts of the scan (pillars and lamps) that indicate correct alignment (green) or hint at wrong alignment (red).

subsection 2.3.2, and can have additional benefits, apart from reducing the influence of points (by their sheer number) near $S_{\text{pos},1}$ or $S_{\text{pos},2}$. Given a minimal number of points that need to be present inside an octree leaf / grid cell, the voxelization can also be used to perform outlier removal. Furthermore, when employing a suitable leaf size l_L, which is dependent on scale and detail, the whole matching process becomes faster, since the number of total points gets reduced substantially. Note that the leaf size l_L naturally has an influence on the precision that can be accomplished by any matching algorithm. Also note that results may be different depending on the type of voxelization (center of a leaf / grid cell vs. aggregated mean of all points in the volume, see 2.3.2) that is performed on the data.

To address these points of valid criticism the alignment estimation for both hallway scans was also performed with a point reduction via octrees. Considering the range and resolution of the employed sensor, a leaf size of $l_L = 0.05\,\text{m}$ was chosen and leaves were represented by their center points. The results obtained by ICP on the reduced point clouds are shown in Table 4.3 and Figure 4.5, respectively. Qualitatively the results with the reduced point clouds do not

initial pose	$\Theta_{\text{good\_init}}$	$\Theta_{\text{trans\_error}}$	$\Theta_{\text{rot\_error}}$
	Rotation unit sensor		
e_{avg}	5.06 cm	4.07 cm	4.48 cm
$\|\|C_{1,2}\|\|$	20695	27582	24933

Table 4.3: Comparison of ICP output for reduced input. Point clouds P_1 and P_2 were filtered using and octree with a leaf size of $l_L = 0.05\,\text{m}$. Initial poses were the same as in Table 4.2. $|P_2| = 35819$.

differ from the results obtained by matching the unreduced point clouds, since the numbers in Table 4.3 still seem to suggest that the alignment obtained from initial poses $\Theta_{\text{trans\_error}}$ and $\Theta_{\text{rot\_error}}$ are preferable to the alignment obtained from $\Theta_{\text{good\_init}}$. Also the visual comparison provided in Figure 4.5 conforms with the results showcased for the unreduced scans in Figure 4.4, i.e., the alignment of the domain specific "features" (pillars, open doors, ceiling lights) clearly indicates that the estimated alignment from initial pose $\Theta_{\text{good\_init}}$ is preferable to the alignments obtained from either $\Theta_{\text{trans\_error}}$ or $\Theta_{\text{rot\_error}}$.

While in the chosen example the matching results for voxelized and non-voxelized point clouds are qualitatively the same, such a behaviour is not to be expected in every scenario. Dependent on the recorded data, the initial pose estimation and ICP parameters the estimated final alignment can substantially differ when applying scan matching to reduced and unreduced point clouds. Unfortunately there is no way to predict a priori which will produce the better results and subsequent evaluation by a human is required. However, looking at the numbers provided by Tables 4.2 and 4.3 it becomes obvious that, using ICP output parameters when evaluating the proposed alignment (and not comparing it to a different pose) becomes even more unsuitable, when we try to compare voxelized input cloud with unreduced point clouds. Naturally $|C_{1,2}|$ will become smaller for point clouds preprocessed by an octree / voxelgrid, since the number of points is reduced. In addition, due to the discretization, e_{avg} will usually become larger, as compared to the original point clouds, provided that we consider momentarily the same alignment between P_1 and P_2 whether the data has been preprocessed or not. A good measure for alignment evaluation should therefore be ideally independent on whether point clouds are reduced or not, or at least should have some tolerance, when it comes to voxelization of point clouds.

4.2 State of the Art and Related Work

Quality assessment, if needed, is currently done either by "hand", i.e., a researcher observes the final alignment obtained by some ICP variant or a different matching algorithm and decides if it is sufficiently "good" or the result is "better" than some previously published methods.

This is not only a problem in the domain of 3D data, but also applies to (earlier) work on 2D scan matching, for instance the work of Rodriguez-Losada and Minguez [61]. In their publication they compare the results of different slam methods under the influence of varying noise, based on odometry, ICP scan matching, and two ICP derivatives MbICP [55] and their own development MbICP+IDA. The resultant maps are visually compared and while the re-

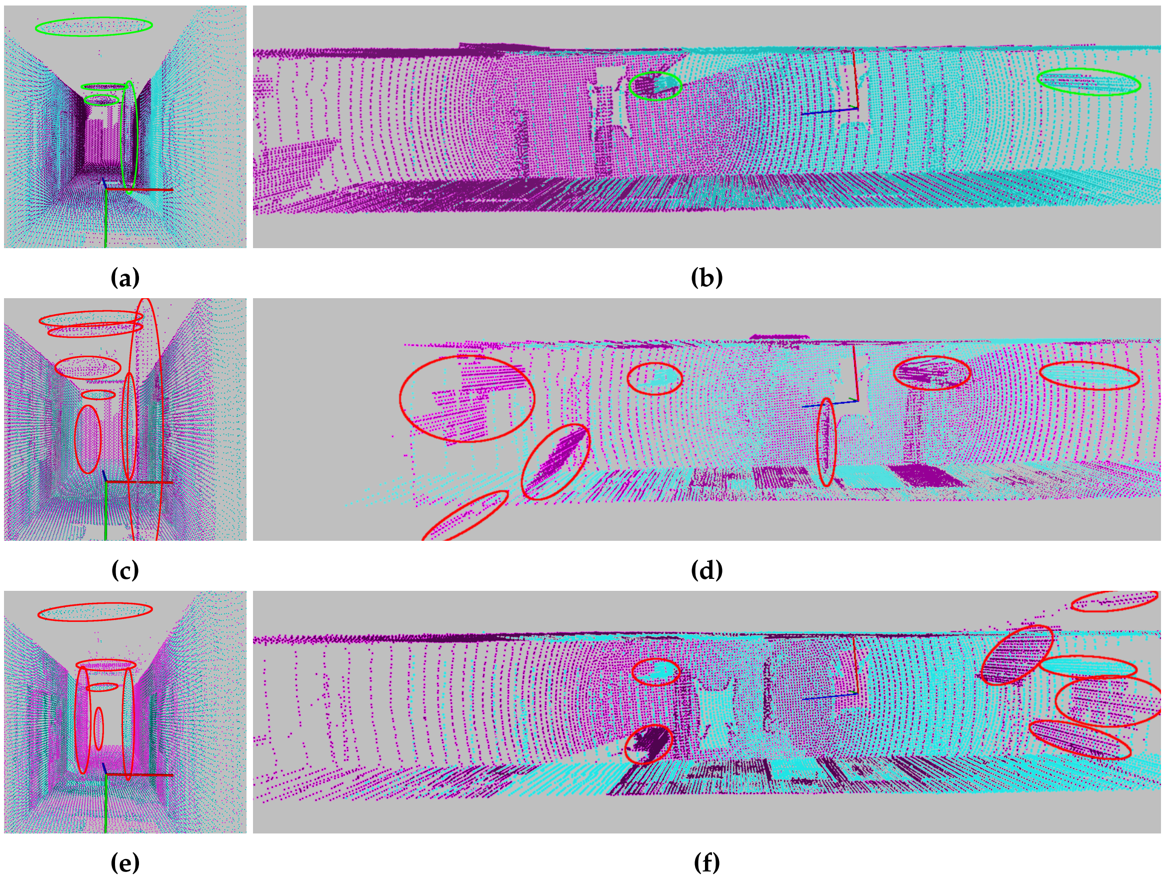


Figure 4.5: Visual comparison of registration results for reduced scans. Top row displays final pose for matching with initial pose $\Theta_{\text{good\_init}}$, middle row with initial pose $\Theta_{\text{trans\_error}}$ and bottom row with pose $\Theta_{\text{rot\_error}}$. Ellipses mark distinct parts of the scan (pillars and lamps) that indicate correct alignment (green) or hint at wrong alignment (red). While the estimated final poses do not differ qualitatively from the ones displayed in Figure 4.4 some parts can be better distinguished due to the point reduction. Therefore, ellipses do not completely correspond to the ones shown in Figure 4.4.

sult obtained by their method was visually the most pleasing of the presented results, there is no hard evidence that it also generated the map that is most consistent with the represented environment.

This is only one of many examples and the work published from the Knowledge-Based Systems Research Group of Osnabrück University is unfortunately not exempt from this flaw: in [51] Magnusson et al. investigate the performance of ICP with the normal distributions transform (NDT [50]) for 3D scan registration from data recorded in the Kvarntorp mine in Sweden. In one conducted experiments a pair of scans was registered by the two different methods in the presence of varying offsets from a reference pose to determine the robustness of both registrations. In the absence of ground truth the reference pose was created by "performing a number of registrations and choosing the mean of the poses that led to visually correct results". While a common practice, for different observers the amount of poses that are still visually correct may vary.

In a different experiment, conducted by Magnusson et al. [51], a loop was closed after registering a series of scans. The loop closure allows to measure how much the estimated pose of the last scan in the loop differs from the first scan (ideally poses should be identical) and was used as means to estimate the overall accuracy of the scan registrations. A weak point of this method is that it does not tell, where during the registration process errors occurred. For example, it is possible that the accumulated error was evenly spread through all performed registrations. However, it is also possible that some scans are aligned perfectly, but others were subject to larger registration errors or, even though highly unlikely, that greater registration errors in the sequential chain even themselves out by chance, so that the error at the loop closure suggests better alignment of the individual scans than there actually is. In fairness, the authors themselves explicitly state these weak points in their publication.

In [32, 33] Günther et al. build 3D semantic maps from RGB-D data, gathered in office and classroom settings. Based on semantic furniture models encoded in an OWL-DL ontology, likely hypotheses for object locations are generated in the acquired data. Using these hypotheses as initial pose estimated CAD models of different pieces of furniture are sampled and ICP is used to align the models in the scene to either confirm or refute the given hypothesis. While Günther et al. present evidence, that not the exact CAD models are needed for sufficient alignment with the data points from the correct furniture category (tables, chairs, racks). The poses obtained after CAD matching was performed were compared to the poses of manually aligned models, that served as "ground truth", but no automatic evaluation of the assumed poses was performed.

These examples are only a small sample of a vast number of publications that touch ICP or scan registration in general in some capacity. These way of evaluating results and the fact that methods are often only applied to a few specific datasets, makes it difficult to compare the results of different approaches in this field. Pomerleau et al. [58] identified this problem and evaluated the performance of several ICP variants on a collection of different datasets. In the evaluation of the alignments an error metric is proposed that provides more information than ICP output parameters e_{avg} and $|C_{1,2}|$ and is easier to interpret, namely a translation error e_t and rotation error e_r for each alignment. However, to compute these values the ground truth poses for each scan are required (and added perturbations to the ground truth poses serve as Θ_{init}), which restricts the use of their error metric to cases where ground truth is available, hence in many scenarios the proposed metric is not applicable, due to the lack of ground truth – if ground truth were readily available there would in general be no need to register scans via ICP or some other method.

If "ground truth" is available, it can be compared with the results of 6D SLAM registrations, for example via applications like CloudCompare [27]. This "ground truth" data may, for instance, consist of a precise 3D blueprint (e.g., in for of CAD models) or by using high-precision scanners, precise pose estimation, artificial scan markers and (semi-) manual alignment, i.e., their acquisition is usually very cost- and work-intensive. Unfortunately, such data needs to be taken with a certain grain of salt. Even if precise 3D Models are available, for example, of the interior of a building, divergences during the construction of the building (within tolerance limits) and small changes over time (due to aging of materials, earthquakes or other external influences) might lead to a state where the "ground truth" model does not precisely confirm with reality. Similarly hand assembled point clouds are of course subject to the measurement errors of the employed sensors and the skill of the person that assembles final result.

Few publications have tried to tackle aspects of what is required for the assessment of scan alignments without much prior knowledge. In [28] Girardeau-Montaut et al. monitor change over time in construction sites in a similar way to CloudCompare. There the earlier recorded point clouds serve as ground truth and are compared with current point cloud data. However, subsequent point clouds are gathered from the same poses as the initial point clouds, thus no pose alignment is performed in this cases, but simply the change in the point clouds is investigated. Similarities to the approaches proposed in sections 4.4 and 4.5 exist in employing octrees as a core component of the analysis and considering visibility constraints in the comparison.

Bosché et al. [15] and Bosché [14] monitor progress of construction sites by comparing the current state, represented in form of a point cloud with the target state, specified via CAD models. Deviations from the target state are determined via aligning point cloud and CAD models, representing individual construction components, via ICP. In contrast to other use cases of ICP the initial pose estimation is already very accurate and thus data association is greatly simplified and the chances for false registration of a component are low and a ground truth (or goal) pose is known via the model of the building.

In [11] Blodow et al. employ a heuristic to hypothesize and validate objects models in 3D scans when only partial information is available. To achieve their goals objects are restricted to be rotational invariant in around their height axis. From the observed data several model hypotheses are generated and represented as 3D meshes. For each model a verification step is performed and the model assigned with the highest score is assumed to be present in the scene. Overlap with this work exists at the model verification step, where the number of vertices from their model that are confirmed in the measurement data is set into relation to the number of vertices that are occluded, given the input data. This is slightly similar to the first proposed evaluation measure in section 4.4.

The work that is, to the best of my knowledge, the closest to what I propose is by Ruhnke et al. [63] in the context of unsupervised learning of 3D models. In their work, they create range images from their point cloud models and, given two viewpoints, they introduce a scoring function for combining two partial models. This scoring function is based on how good a depth image of the combined model fits with respect to the given viewpoints to depth images of the two original model instances. This idea is similar to the basis of the evaluation measure that will be introduced in section 4.4. However, in my approach no range images, that are dependent on predefined viewpoints, are created and the introduction of different weights in my approach allows for different assessments, depending on the distribution of contradicting information.

The use of range images, generated from point cloud data is also done in [5] by Aldoma et al., where they propose a framework for 3D object recognition in cluttered environments. The range images are employed to detect in-scene and self occlusions for object hypothesis. Other inputs that are used for model verification are visual clues and local point features. The latter will explicitly not be used in my proposed method and I will later explain, why I believe these are not suitable in general in this context.

None of the related work actually attempts to generate a measure to evaluate the alignment of two point clouds via a given pose, but single aspects in their work bear similarities.

4.3 Mutual Observability

After introducing the general problem when trying to automatically evaluate proposed alignments obtained from ICP (or some similar method) and illustrating these problems by suitable examples in section 4.1 and discussing currently available related works in 4.2, this section will introduce the general idea of the comparison measures that will be introduced later in this chapter. Furthermore, this section will show more clearly how the proposed idea of scan alignment evaluation is not only a desirable feature for the transparent object reconstruction described in chapter 3 or maybe even a prerequisite, if the transparent object reconstruction is desired to work reliably and completely autonomously, but that it also turns out to be quite related in a conceptual way.

This conceptual connection hinges on the concept of mutual occlusion (see chapter 3) to which, as already briefly mentioned in the introduction to this chapter, a very similar concept, namely *Mutual Observability* exists. The general idea behind the latter and how it can be employed as a means of alignment evaluation will be discussed in the remainder of this section.

As mentioned in the introduction of the general ICP algorithm (see subsection 2.3.1) a vital component for applying ICP is, apart from scenario dependent suitable input parameters and a sufficiently good initial pose estimation, that both taken scans P_1 and P_2 have sufficient overlap, when correctly aligned, so that meaningful point-to-point correspondences can be established. That means that at least some structures and objects in the real world need to be in the field of view of sensor S_1 when recording point cloud P_1 and S_2 when obtaining P_2, respectively. These common structures and their associated measurements are the basis for the earlier mentioned Mutual Observability.

However, Mutual Observability is not a concept that can in general be applied to individual measurement points. Even if we ignore the existence of sensor noise momentarily, we are still employing sensors that are based on measurement rays that are emitted from a singular source. Let us assume in the following that Θ_{GT} will denote the ground truth pose that embed sensor S_2 correctly into the coordinate frame of sensor S_1, i.e., transformed point cloud $\Theta_{GT} \cdot P_2$ is perfectly aligned with point cloud P_1.

Using this notation, it is highly unlikely that measurements for a (real world) surface, recorded by the same sensor from the origin O of the coordinate system (S_1) and from Θ_{GT} (S_2) coincide precisely, even though Θ_{GT} aligns both scans perfectly (with the only exception being $\Theta_{GT} = I$, the identity, i.e., the sensor was not moved between the recording of both point clouds). This can be illustrated, if we assume that pose Θ_{GT} did not induce a change in orientation, but only a translation, for example, perpendicular to some surface measured in P_1. In this case only the distance to the measured surface has changed. Since we assume that the sensor emits measurement rays and the sensor configuration was not modified, i.e., its angular resolution is identical when recording P_1 and P_2, the density of measurement points on the surface is higher in the point cloud where the sensor was positioned closer to the surface.

This fact also becomes clear, when we observe the data in the (approximately correctly) aligned scans depicted in Figure 4.4b (or the octree filtered version in Figure 4.5b) where the number of (magenta colored) measurement points on the pillar marked on the right-hand side from $\Theta_{ICP} \cdot P_2$ is much less than the points from P_1 (cyan). This is mainly due to the aforementioned difference in point density in this part of the environment between P_1 and $\Theta_{ICP} \cdot P_2$.

If we take a bit more time to consider the correctly matched example depicted in Figure 4.4b (or Figure 4.5b) another fact strikes the eye: even if the combined scene is aligned correctly, not all points from P_1 do need to have a corresponding point in $\Theta_{\text{ICP}} \cdot P_2$ or vice versa, independently of point densities in P_1 and $\Theta_{\text{ICP}} \cdot P_2$. This becomes evident in the left-handed pillar (circled in green), since it was observed from different sides. Observed from origin O, from which the cyan colored point cloud P_1 was recorded, the pillar is on to the left of the scanner (the x-axis constitutes its heading), which is depicted in Figure 4.3b. When the scene is recorded from Θ_{GT}, which should roughly correspond to Θ_{ICP} in this case, the pillar is positioned to the right of the sensor (see magenta cloud in Figure 4.3d).

As mentioned in section 2.4, the sensors employed in this thesis are only capable to measure distances up to the first solid obstacle a measurement ray encounters. Hence, P_1 does not contain any measurements of the backside (considered from the origin O) of the pillar. However, this "backside" constitutes the "front" of the pillar, if considered from pose Θ_{GT} and the majority of points belonging to the pillar in P_1 are occluded from this point of view. So while this pillar is certainly in the overlap of both point clouds, we cannot automatically assume that it is represented in the same way in P_1 and P_2 or that $\Theta_{\text{ICP}} \cdot P_2$ should contain points at all locations where measurements of P_1 were recorded and vice versa.

Compactly paraphrased the observations made in the previous paragraphs can be condensed to

- There needs to be overlap between the scenes observed from sensor S_1 located at origin O and sensor S_2 at Θ_{GT} and the associated point clouds P_1, P_2 if ICP is applied (sensibly).
- Detection of this overlap can not be based on direct point-to-point comparisons, even if P_1 and P_2 are aligned perfectly via pose Θ_{GT}.
- Not all points in the overlap in P_1 need to have correspondences in $\Theta_{\text{GT}} \cdot P_2$, or vice versa, due to possible in-scene occlusions.

This begs the question if and how this overlap can be used in order to determine if both point clouds P_1 and P_2 were aligned correctly by a given pose Θ_{ICP}, obtained via ICP or some other scan matching method (in the following I will only mention ICP for the sake of brevity). From the observations, made so far, it is not immediately clear how to directly identify the volume of overlap in both point clouds. Furthermore, we need to distinguish between parts of the environment where measurement points from both sources S_1 and S_2 should occur, albeit usually with different point density, and those parts where it is correct if only measurement points from one source are present. Again we do not have any a-priori information that we can employ to identify which parts of the scene belong to which of these two general classes.

The basic idea is, similar to the basic idea when reconstructing transparent objects (see chapter 3), a simple one: let us assume momentarily that the alignment information, obtained from ICP, is correct, i.e., $\Theta_{\text{ICP}} = \Theta_{\text{GT}}$. As already done above it will be assumed, that the data gathered by sensor S_2 needs to be embedded into the coordinate frame of sensor S_1 via pose Θ_{ICP}, i.e., the combined point cloud with the alignment estimate from ICP can be denoted as $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$. In this case we can employ sensor specific information (i.e., minimal and maximal range and opening angles) to determine which volume in the common coordinate

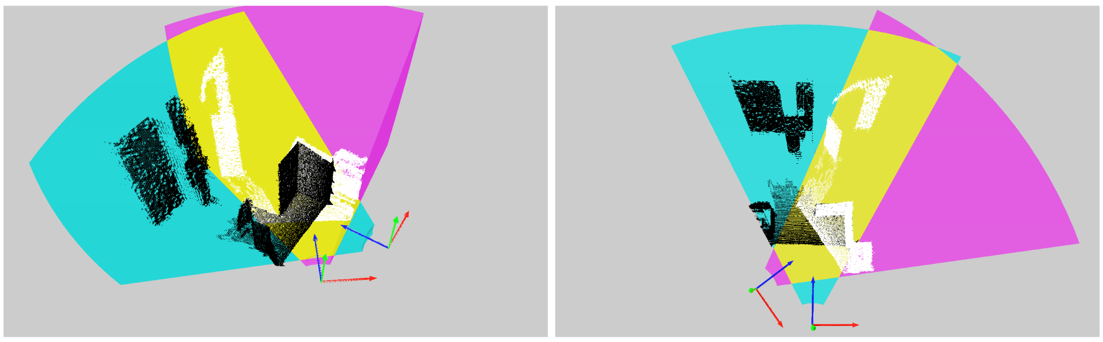


Figure 4.6: Visualization of Region of Overlap and Observable Volume: Side view **(a)** and top view of **(b)** P_1 (black) and P_2 (white). Observable volume OV_1 displayed in cyan, OV_2 in magenta and $ROO_{1,2}$ in yellow.

frame should, in principle, be observable for S_1 at pose Θ_1. This *observable volume* for sensor S_1 will be denoted as OV_1. Likewise OV_2 can be established for S_2 at Θ_{ICP} in the coordinate frame defined by sensor S_1.

Note that OV_i describes the volume, in which measurements of sensor S_i can be located. If sensor S_i is correctly modelled no measurement can be recorded that lies outside of OV_i. It is important to realize that OV_i describes a theoretical volume, in the sense that it does not consider obstacles and occlusions that restrict the actually observable volume in a specific scene. In contrast to the fixed and scene independent volume that OV_i constitutes, the number of points that lie inside of OV_i is scene dependent. If sensor S_i is placed in a way that many measurement rays do not hit any surface within the bounds defined by minimal and maximal measurement range, OV_i will contain less points when compared to a scene where the sensor is placed in a way that every measurement ray hits a surface inside its range limits.

It might be worth considering that, provided correct modelling of the sensor, the number of points inside OV_i should correspond to $|P_i|$, in the case of unorganized point clouds, while this will usually not hold for organized point clouds (see 2.2.3). The intersection of volumes OV_1 and OV_2, i.e., the volume that can potentially be observed by both sensors, is referred to as the *Region of Overlap* ($ROO_{1,2}$). This formalism allows for sensors S_1 and S_2 to be different and accordingly have different properties in terms of opening angles and measurement range. A visualiztion is provided in Figure 4.6 for two aligned point clouds recorded by a Kinect RGB-D camera.

If we want to have a measure for the alignment quality, only those subsets of the measurement points P_1 and $p_{j,2} \in \Theta_{ICP} \cdot P_2$ that lie inside of $ROO_{1,2}$ should be considered, since points that were recorded from only one sensor, but are outside the other sensor's field of view should have no influence when evaluating the quality of alignment. In the following the notations $P_1 \cap ROO_{1,2}$ and $\Theta_{ICP} \cdot P_2 \cap ROO_{1,2}$ will be used to refer to the points of P_1 and aligned point cloud P_2 that are located inside $ROO_{1,2}$.

As stated earlier in this section, an evaluation of the alignment based on individual points is not sensible, therefore I propose discretization of the combined point cloud inside the region of overlap, i.e., $ROO_{1,2} \cap (P_1 \cup \Theta_{\text{ICP}} \cdot P_2)$, via an octree. Subsequently the individual octree volumes will be investigated and on their contents the suggested evaluation measures is based. In sections 4.4 and 4.5 two different approaches will be introduced how to built an evaluation measure, based on the idea of Mutual Observability while taking possible occlusions in the region of overlap into consideration. The approach described in section 4.4 will model $ROO_{1,2}$ explicitly and then proceed to investigate the points inside it. In contrast to this the idea in section 4.5 uses the idea of Mutual Observability in an implicit way to generate a quantitative measure for the alignment of P_1 and P_2.

4.4 Direct Modelling of Mutual Observability

As illustrated in section 4.3 the concept of Mutual Observability can potentially be employed to generate a measure about the quality of point cloud alignment. This section will discuss how these ideas can be implemented in a straight forward fashion to produce the desired quality measure. An alternative approach, which where the modelling is done only in an implicit fashion is introduced in the following in section 4.5.

This section is structured in two subsections: 4.4.1 will discuss how to filter the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ in order to retrieve the points that lie inside $ROO_{1,2}$. Afterwards in 4.4.2 the comparison of the data inside the Region of Overlap while considering Mutual Observability is described in detail.

4.4.1 Region of Overlap

Since only points inside of $ROO_{1,2}$ can serve as positive evidence that the alignment of P_1 and P_2 is sufficiently good (or indicate incorrect alignment), it is sensible to first filter the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ so that only the points inside $ROO_{1,2}$ remain. As stated in 4.3 a measurement point $p_{i,j} \in P_j, j \in \{1,2\}$ recorded by sensor S_1 or S_2 needs to lie inside both OV_1 and OV_2 in order to be part of $ROO_{1,2}$. In the context of an evaluation of the alignment quality between P_1 and $\Theta_{\text{ICP}} \cdot P_2$, the actual volume of the region of overlap (for example in m^3) or its shape (e.g., represented as a polyhedron) are only of minor interest. Relevant is the subset from the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ that are assembled inside $ROO_{1,2}$. Obviously each measurement point $p_{i,j} \in P_j$ with $j \in \{1,2\}$ is inside OV_j, provided the sensor model is accurate. Hence, the points $p_{i,j}$ that are located inside OV_k, with $k \in \{1,2\}, k \neq j$, need to be determined in order to establish the desired subset $ROO_{1,2} \cap (P_1 \cap \Theta_{\text{ICP}} \cdot P_2)$.

If a point $p_{i,j}$ is inside of OV_k depends on the position of $p_{i,j}$ in relation to the pose of sensor S_k and its characteristics. As discussed earlier, it cannot be expected for measurement points obtained from different poses to coincide, even when employing the same sensor. Therefore, I chose to simplify the sensor model as much as possible, so that only a select few properties need to be known about the sensor in order to determine the points that lie inside the $ROO_{1,2}$. A sensor S_i is sufficiently defined for the computation of a region of overlap, by the following four parameters:

- $\rho_{\min,i}$ the minimal measurement range of sensor S_i
- $\rho_{\max,i}$ the maximal measurement range of sensor S_i
- $\omega_{h,i}$ the horizontal opening angle of sensor S_i, with $0 < \omega_{h,i} \leq 2\pi$
- $\omega_{v,i}$ the vertical opening angle of sensor S_i, with $0 < \omega_{v,i} \leq \pi$

Modelling the $ROO_{1,2}$ in this fashion might not be appropriate, if sensors S_1 and S_2 are vastly different in their resolutions, i.e., one produces very dense point clouds, while the other one records only very sparse point clouds. However, usually when combining point clouds via scan registration either the same sensor was used to record a scene from different poses or similar sensors are used, so that the registration of point clouds with vastly different properties is more of a theoretical problem.

In order to simplify the notation used later, when determining if a point belongs to the $ROO_{1,2}$ a formalism will be introduced, namely a point $\tilde{p}_{i,j}$, $j \in \{1,2\}$. Assuming that Θ_{ICP} is the transformation that aligns P_2 with P_1 a point $\tilde{p}_{i,j}$ is defined as the following:

$$\tilde{p}_{i,j} = \begin{cases} \Theta_{\text{ICP}}^{-1} \cdot p_{i,j} & \text{if } j = 1 \\ \Theta_{\text{ICP}} \cdot p_{i,j} & \text{if } j = 2 \end{cases} \tag{4.3}$$

Thus $\tilde{p}_{i,j}$ transforms a point $p_{i,j}$ into the coordinate system of the sensor which did not record it. As a first step when determining which points of the combined cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ belong to $ROO_{1,2}$, the transformed point $\tilde{p}_{i,j}$ is created for every recorded point $p_{i,j} \in P_1 \cup P_2$, culminating in point clouds $\tilde{P}_1$ and $\tilde{P}_2$. Transforming all points $p_{i,2}$ into the coordinate frame of sensor S_1, i.e., $\tilde{p}_{i,2}$ is an operation needed to generate the aligned point cloud in the first place, so no additional computation is needed. Creating the $\tilde{p}_{i,1}$ for all points of point cloud P_1 allows for more convenient checks if a point $p_{i,1}$ is inside of $ROO_{1,2}$, when compared to performing the check for the non-transformed points, since the sensor in this representation is always positioned at the origin of the coordinate frame with a predefined viewing direction.

With the notion of $\tilde{p}_{i,j}$ in place, the first filtering step applied to all points $\tilde{p}_{i,j}$, $j \in \{1,2\}$ is a range check, if they are in the measurement range of the scanner that did not record them. This is the case if the following inequality holds:

$$\rho_{\min,k}^2 \leq \left\|\tilde{p}_{i,j}\right\|^2 \leq \rho_{\max,k}^2 \qquad \text{where } j,k \in \{1,2\}, j \neq k \tag{4.4}$$

Being a necessary requirement and computationally cheap, the range check in inequality (4.4) is applied to reduce the number of points that will be subject to the subsequent checks. All points $\tilde{p}_{i,j}$ that comply with inequality (4.4) are in the measurement range of S_1 and S_2 respectively. In order to be part of $ROO_{1,2}$ additional checks need to be performed to determine if they are also inside of OV_1 or OV_2, i.e., in the field of view of the sensor that did not record them or lie outside of this volume. This will be done by comparing the transformed points $\tilde{p}_{i,j}$ with the opening angles of S_k, namely $\omega_{h,k}$ and $\omega_{v,k}$ (with $j,k \in \{1,2\}, j \neq k$).

The check for the horizontal opening angle $\omega_{\mathrm{h},k}$, each point $\tilde{p}_{i,j}$ is projected into the horizontal viewing plane, i.e., the xz-plane of sensor S_k, assuming the reference coordinate system defined in Figure 2.3, section 2.1. Afterwards we can compute the dot product of the projected point and the viewing direction (z-axis). Comparing the dot product with the cosine of horizontal opening angle $\omega_{\mathrm{h},k}$ tells us if $\tilde{p}_{i,j}$ is within the limits defined by the horizontal opening angle. If we denote the normalized vector in viewing direction as $z_{\mathrm{n}} = (0,0,1)^T$ and the point coordinates of $\tilde{p}_{i,j}$ as $\left(x_{\tilde{p}_{i,j}}, y_{\tilde{p}_{i,j}}, z_{\tilde{p}_{i,j}}\right)^T$, said point $\tilde{p}_{i,j}$ is within the limits defined by horizontal opening angle $\omega_{\mathrm{h},k}$, if

$$\cos\left(\frac{1}{2}\omega_{\mathrm{h},k}\right) \le \frac{\tilde{p}'_{i,j} \cdot z_{\mathrm{n}}}{\left\|\tilde{p}'_{i,j}\right\|} = \frac{z_{\tilde{p}_{i,j}}}{\sqrt{x^2_{\tilde{p}_{i,j}} + z^2_{\tilde{p}_{i,j}}}} \tag{4.5}$$

holds, where $\tilde{p}'_{i,j}$ denotes the orthogonal projection of $\tilde{p}_{i,j}$ into the xz-plane, i.e., $\tilde{p}'_{i,j} = \left(x_{\tilde{p}_{i,j}}, 0, z_{\tilde{p}_{i,j}}\right)^T$.

However, inequality (4.5) is only appropriate for horizontal opening angles $\omega_{\mathrm{h},k} \le \pi$. If sensor S_k features a larger opening angle a different check needs to be performed. In this case if $z_{\tilde{p}_{i,j}} \ge 0$ holds point $\tilde{p}_{i,j}$ is inside the field of view, as far as the horizontal opening angle is concerned (since it is somewhere in front of the sensor). So only points with $z_{\tilde{p}_{i,j}} < 0$ need to be checked with respect to the actual value of the horizontal opening angle $\omega_{\mathrm{h},k}$. This is done with the following inequality instead of (4.5):

$$\cos\left(\frac{1}{2}\left(2\pi - \omega_{\mathrm{h},k}\right)\right) \ge \frac{\tilde{p}'_{i,j} \cdot z_{\mathrm{n}}}{\left\|\tilde{p}'_{i,j}\right\|} = \frac{z_{\tilde{p}_{i,j}}}{\sqrt{x^2_{\tilde{p}_{i,j}} + z^2_{\tilde{p}_{i,j}}}} \tag{4.6}$$

where the term $\cos\left(\frac{1}{2}\left(2\pi - \omega_{\mathrm{h},k}\right)\right)$ describes the "blind spot" behind sensor S_k. Note, that inequalities (4.5) and (4.6) assume a symmetric horizontal opening angle, when considered from the viewing direction. This is also the reason why the term $\frac{1}{2}\omega_{\mathrm{h},k}$ is employed in (4.5) and (4.6), since a horizontal opening angle $\omega_{\mathrm{h},k} = \frac{\pi}{2}$ implies that there is an opening angle of $\frac{\pi}{4}$ between the z-axis and the positive x-axis and an opening angle of $\frac{\pi}{4}$ between z-axis and negative x-axis. In case of an asymmetric horizontal opening angle, which seems to be rather uncommon for 3D sensors, (4.5) and (4.6) would need to be adapted accordingly.

All points $\tilde{p}_{i,j}$ that have passed the checks expressed by either inequalities (4.4) and (4.5) or, dependent on the sensor properties, (4.4) and (4.6) are subject to another examination with respect to the vertical opening angle $\omega_{\mathrm{v},k}$ of the employed sensors. Analog to the check for the horizontal opening angle (inequality (4.5) or (4.6)) it is possible to check if the dot product of a point $\tilde{p}_{i,j}$ and vector y_{n}, the normalized vector in direction of the y-axis, is within the limits defined by vertical opening angle $\omega_{\mathrm{v},k}$. Here I distinguish between points that lie above the xz-plane and below – which also enables easy adaptation for sensors with an asymmetric vertical opening angle. A given point $\tilde{p}_{i,j}$ with $y_{\tilde{p}_{i,j}} < 0$, i.e., a point that lies above the xz-plane, is inside OV_k, the observable region of sensor S_k, if the following inequality holds:

$$\cos\left(\frac{\pi}{2} - \frac{\omega_{\mathrm{v},k}}{2}\right) \ge \frac{\tilde{p}_{i,j} \cdot y_{\mathrm{n}}}{\left\|\tilde{p}_{i,j}\right\|} = \frac{y_{\tilde{p}_{i,j}}}{\sqrt{x^2_{\tilde{p}_{i,j}} + y^2_{\tilde{p}_{i,j}} + z^2_{\tilde{p}_{i,j}}}} \tag{4.7}$$

where y_n denotes the vector $(0,1,0)^T$, i.e., a normalized vector in the direction of the y-axis. For points $\tilde{p}_{i,j}$ that are located below the xz-plane ($y_{\tilde{p}_{i,j}} \geq 0$) the right-hand side of (4.7) only needs to be multiplied by -1 and becomes

$$\cos\left(\frac{\pi}{2} - \frac{\omega_{\text{v},k}}{2}\right) \geq \frac{-\tilde{p}_{i,j} \cdot y_\text{n}}{\|\tilde{p}_{i,j}\|} = \frac{-y_{\tilde{p}_{i,j}}}{\sqrt{x^2_{\tilde{p}_{i,j}} + y^2_{\tilde{p}_{i,j}} + z^2_{\tilde{p}_{i,j}}}} \tag{4.8}$$

Sensors sometimes feature asymmetrical opening angles, e.g., the angle in the direction of the negative y-axis (up) is larger than the field of view in the direction of the y-axis (down). In this case the description of the sensor parameters needs to be slightly changed. Instead of vertical opening angle $\omega_{\text{v},k}$, the opening angle in the direction of the y-axis ($\omega_{\text{v}\downarrow,k}$) and in the negative y-direction ($\omega_{\text{v}\uparrow,k}$) need to be known. It is assumed, as with all opening angles, that $\omega_{\text{v}\uparrow,k} \geq 0$ and $\omega_{\text{v}\downarrow,k} \geq 0$ hold. Subsequently instead of applying inequality (4.7) for points above the xz-plane the following condition must be fulfilled for points $\tilde{p}_{i,j}$ to be inside $ROO_{1,2}$:

$$\cos\left(\frac{\pi}{2} - \omega_{\text{v}\uparrow,k}\right) \geq \frac{\tilde{p}_{i,j} \cdot y_\text{n}}{\|\tilde{p}_{i,j}\|} = \frac{y_{\tilde{p}_{i,j}}}{\sqrt{x^2_{\tilde{p}_{i,j}} + y^2_{\tilde{p}_{i,j}} + z^2_{\tilde{p}_{i,j}}}} \tag{4.9}$$

And conversely inequality (4.8) becomes

$$\cos\left(\frac{\pi}{2} - \omega_{\text{v}\downarrow,k}\right) \geq \frac{-\tilde{p}_{i,j} \cdot y_\text{n}}{\|\tilde{p}_{i,j}\|} = \frac{-y_{\tilde{p}_{i,j}}}{\sqrt{x^2_{\tilde{p}_{i,j}} + y^2_{\tilde{p}_{i,j}} + z^2_{\tilde{p}_{i,j}}}} \tag{4.10}$$

The checks for vertical opening angle $\omega_{\text{v},k}$ assume that $\omega_{\text{v},k} \leq \pi$ holds (or $\omega_{\text{v}\uparrow,k} \leq \frac{\pi}{2}$ and $\omega_{\text{v}\downarrow,k} \leq \frac{\pi}{2}$ respectively). These assumptions about the field of view comply with a large number of popular sensors and I am currently not aware of any sensor that features larger vertical opening angles in its reference coordinate system. However, if needed, inequalities (4.7) and (4.8) or (4.9) and (4.10) respectively, can be adapted analogously to the horizontal opening angle (see transition from (4.5) to (4.6) as reference).

Applying these three filter steps (range filtering, horizontal opening angle and vertical opening angle) to all points $\tilde{p}_{i,j} \in \tilde{P}_1 \cup \tilde{P}_2$ results in $ROO_{1,2}$ in terms its included points. The latter is composed of points $\tilde{p}_{i,2}$, that are retained after filtering and those points $p_{1,1}$, where $\tilde{p}_{i,1}$ passed the filtering. In the following these points will be investigated with regard to Mutual Observability and occlusions in order to asses the quality of the alignment provided by Θ_ICP.

4.4.2 Octree Based Volume Comparison

In subsection 4.4.1 the *Region of Overlap* $ROO_{1,2}$ of the combined point cloud $P_1 \cup \Theta_\text{ICP} \cdot P_2$ was established. The $ROO_{1,2}$ encompasses all the points that are *potentially* visible from both sensors, S_1 and S_2, when they are aligned via pose Θ_ICP. But, up until now, the aspects of occlusion and Mutual Observability were not considered. This problem will be tackled in the following and subsequently lead to a proposed evaluation measure for the alignment obtained by Θ_ICP.

As already explained in section 4.3 a direct point-to-point comparison is not sensible in order to asses alignment quality. Therefore, all points inside the region of overlap, i.e., $ROO_{1,2} \cap (P_1 \cup \Theta_{\text{ICP}} \cdot P_2)$, as determined in the previous subsection, will be discretized into smaller volumes. This volume based representation reduces in the following the influence of different point densities in P_1 and P_2 and can compensate some sensor noise, as opposed to comparisons based on individual point measurements. In this context it is assumed that, apart from their geometric information, points from P_1 and P_2 posses additional information, like a label (see section 2.2), that allows to directly associate a given point $p_{i,j}$ with the sensor S_j that recorded the point.

A natural way to discretize $ROO_{1,2}$ is by subdividing it into uniform smaller volumes, which can be achieved via an octree or voxel grid. Since the former is generally more efficient in terms of memory consumption I opted for that. The discrete volumes, i.e., the octree leaves, are investigated concerning their contents. This way the point cloud $ROO_{1,2} \cap (P_1 \cup \Theta_{\text{ICP}} \cdot P_2)$ can be investigated by iterating over all leaves of the corresponding octree.

Keeping in mind the idea of Mutual Observability, we will assume that a leaf L_i that is occupied by points $p_{j,1} \in P_1$ and points $\tilde{p}_{k,2} \in \Theta_{\text{ICP}} \cdot P_2$ will serve as an indicator for correct alignment. The volume is inside the $ROO_{1,2}$ and contains measurement point from both sources – if P_1 and P_2 are aligned correctly via pose Θ_{ICP} this should occur often. Figures 4.4a and 4.4b illustrate this point: while there are, due occlusions caused by in-scene objects, some particular volumes where only points gathered from one sensor are present, the large majority of leaves from $ROO_{1,2}$ are filled with points $p_{j,1}$ and $\tilde{p}_{k,2}$ recorded from both sensors, albeit often with different point densities.

In contrast to this, scenes with bad alignment (refer to Figures 4.1a and 4.1c as an example) quite a large fraction of the filled leafs inside the $ROO_{1,2}$ are not occupied by points from both P_1 and $\Theta_{\text{ICP}} \cdot P_2$. Although the $ROO_{1,2}$ is not explicitly marked in the Figures, the majority of the colored points can be assumed to be located on the inside of the region of overlap, given the point cloud alignments. These volumes need to be investigated closer in order to make a quality assessment of the registration process, so it becomes possible to distinguish cases caused by scene occlusions and bad alignment. The importance of these volumes becomes also evident if the three different alignment results in Figure 4.4 are considered: due to the nature of the recorded scene, all 3 alignments (the correct one in Figures 4.4a and 4.4b and the two incorrect ones in Figures 4.4c – 4.4f) contain a lot of leaves that are occupied by points from $p_{j,1}$ and $\tilde{p}_{k,2}$. Therefore, the information from the leaves that are only occupied either with points $p_{j,1} \in P_1$ or points $\tilde{p}_{k,2} \in \Theta_{\text{ICP}} \cdot P_2$ become crucial, when deciding if an alignment is suitable or false.

As introduced in subsection 2.3.2 an octree can be viewed as a set $\mathcal{L} = \{L_0, L_1, \ldots, L_n\}$ of its leaves L_i, where each leaf describes a specific volume of the scene and contains at least one measurement point. In the following L_i will also be used to denote a set of measurement points, namely those points of the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ that are encompassed by its volume.

If a volume L_i is encountered, that only contains measurement point from one source, i.e., either $L_i \cap P_1 = \emptyset$ or $L_i \cap (\Theta_{\text{ICP}} \cdot P_2) = \emptyset$, further investigation becomes necessary in order to tell if this volume refutes the notion that point clouds P_1 and P_2 are aligned correctly. In Figure 4.4b the pillar to the left of the origin serves to illustrate that this can happen when scans are aligned correctly, while pillars in Figures 4.4d and 4.4f show that these volumes can also serve as a strong indicator for wrong alignment. However, between the volumes L_i occupied only by points from a single source S_k, with $k \in \{1,2\}$, in correctly aligned Figure 4.4b and the faulty alignment (in 4.4d and 4.4f) a distinct difference exists that can be employed to distinguish between these two general cases. The way the data is aligned in Figure 4.4b contains a justification why some leaves L_i contain only points $p_{k,1} \in P_1$, but not points $\tilde{p}_{k,2} \in \Theta_{\text{ICP}} \cdot P_2$, namely that they are occluded from S_2 when P_1 and P_2 are aligned via Θ_{ICP} and this occlusion is reflected in the data gathered by S_2. Of course the dual can hold true for leaves L_i with $L_i \cap P_1 = \emptyset$ and $L_i \cap (\Theta_{\text{ICP}} \cdot P_2) \neq \emptyset$.

An explanation for the absence of measurement points in a certain leaf L_i can be formalized in the following way: let us, without loss of generality, assume that $L_i \cap \Theta_{\text{ICP}} \cdot P_2 = \emptyset$ and L_i is part of $ROO_{1,2}$. In this case L_i is considered occluded from S_2, if there exists some leaf L_j, that is located between L_i and $\Theta_{\text{ICP}} \cdot S_2$ and where $L_j \cap \Theta_{\text{ICP}} \cdot P_2 \neq \emptyset$ holds. The term $\Theta_{\text{ICP}} \cdot S_2$ denotes here the pose of sensor S_2 in the coordinate frame defined by S_1, although in this context only the position of pose $\Theta_{\text{ICP}} \cdot S_2$ is relevant. The relative rotation between both sensors can be neglected here, since this was already considered in the construction of the region of overlap $ROO_{1,2}$.

A leaf L_j is considered "between" L_i and $\Theta_{\text{ICP}} \cdot S_2$, if the volume associated with L_j is intersected by the line segment defined by the center of L_i and the position of $\Theta_{\text{ICP}} \cdot S_2$. Of course, the same consideration can be applied if $L_i \cap P_1 = \emptyset$ holds. In this case an occlusion is reflected by the data, if a leaf L_j is placed somewhere along the line segment defined by the center of L_i and the origin O (remember that we assume that S_1 defines the reference coordinate system) and $L_j \cap P_1 \neq \emptyset$ holds.

Less formally this can be expressed simply as: a leaf L_i is considered occluded from a sensor S_k if the sensor S_k measured something in front of L_i. These measurements will result in the leaf L_j used in the previous paragraph.

In Figure 4.4b for the majority of leaves L_i with either $L_i \cap P_1 = \emptyset$ or $L_i \cap (\Theta_{\text{ICP}} \cdot P_2) = \emptyset$ a leaf L_j of the previously described properties exists between the concerned sensor and L_i, thus explaining the absence of measurement points in leaf L_i. In contrast such an explanation is absent for most of these leaves in the wrongly aligned case depicted in Figures 4.4d and 4.4f.

Note the possibility of encountering a leaf L_i with $L_i \cap P_1 \neq \emptyset$, while there also exists some leaf L_j located between L_i and S_1 with $L_j \cap P_1 \neq \emptyset$, i.e., there might be a leaf containing points recorded from sensor S_1, although a different leaf, placed between them, also contains points recorded by S_1. This effect can be caused by the discretization, e.g., one particular measurement ray from S_1 can pass through L_j and only hit some obstacle that is located behind L_j in the volume of leaf L_i, while other measurement rays from S_1 already hit some surface inside the volume of L_j.

Therefore, the existence of a leaf L_j with $L_j \cap P_1 \neq \emptyset$ does not necessarily entail that all leaves L_i that are located "behind" L_j are occluded from sensor S_1, but is only a necessary criterion for L_i to be considered occluded. The term "behind" here encompasses all leaves L_i that are intersected by the line segment, defined from the position of S_1 to the center of leaf L_j and have a larger distance to S_1 than L_j. Of course, the same considerations are valid, if they are applied to measurements taken from sensor S_2 instead of S_1. In this case P_1 needs to be replaced by $\Theta_{\text{ICP}} \cdot P_2$ and likewise the position of S_1 with the position $\Theta_{\text{ICP}} \cdot S_2$ in the previous considerations.

Naturally it is important to restrict this analysis to the volume, which defines the $ROO_{1,2}$. Outside of this volume any leaf that contains only points of a single source is no indication against correct alignment, since portions of the scene outside of $ROO_{1,2}$ should only be observable from at most one sensor (either S_1 or S_2 or no sensor at all). If considered from this perspective it might actually also serve as an indicator for wrong alignment if many leaves L_i exist, where $L_i \notin ROO_{1,2}$ and $L_i \cap P_1 \neq \emptyset \wedge L_i \cap \Theta_{\text{ICP}} \cdot P_2 \neq \emptyset$ holds, since such leaves are supposed to exist only inside of the $ROO_{1,2}$.

The considerations above lead to three distinctive cases, when examining the leaves L_i of the filtered combined point cloud $ROO_{1,2} \cap (P_1 \cup \Theta_{\text{ICP}} \cdot P_2)$, which will serve as the basis for the first proposed evaluation measure. Let $\check{P}_i$ be defined as

$$\check{P}_i = \begin{cases} P_i & \text{if } i = 1 \\ \Theta_{\text{ICP}} \cdot P_i & \text{if } i = 2 \end{cases} \tag{4.11}$$

i.e., $\check{P}_i$ refers to point cloud P_i transformed into the reference coordinate system from sensor S_1. If we also define $\check{S}_i$ in a similar fashion as

$$\check{S}_i = \begin{cases} S_i & \text{if } i = 1 \\ \Theta_{\text{ICP}} \cdot S_i & \text{if } i = 2 \end{cases} \tag{4.12}$$

where S_i in this case denotes the position in terms of 3D coordinate of the sensor, so that $\check{S}_i$ refers to the position of S_i in the reference coordinate system, This notation allows for a formalization in the following way:

- The leaf contains points from both sensors, i.e.,

$$L_i \cap P_1 \neq \emptyset \ \wedge \ L_i \cap \Theta_{\text{ICP}} \cdot P_2 \neq \emptyset \tag{4.13}$$

- The leaf contains points from just one sensor, i.e.,

$$\left(L_i \cap \check{P}_k = \emptyset\right) \wedge \left(L_i \cap \check{P}_l \neq \emptyset\right) \tag{4.14}$$

  with $k, l \in \{1, 2\}$ and $k \neq l$.

Leaves L_i where equation (4.13) holds, serve as a confirmation for correct alignment, constitute the first of the aforementioned three distinct cases. In contrast leaves where equation (4.14) holds, require further investigation to classify them either as containing no information about the pose alignment or conveying information that the alignment via Θ_{ICP} is incorrect. Let $\overline{L_i\check{S}_k}$ denote the line segment from L_i to point $\check{S}_k$, then

$$L_j \not\oslash \overline{L_i\check{S}_k} \tag{4.15}$$

symbolizes that the volume of L_j is intersected by line segment $\overline{L_i\check{S}_k}$. Let furthermore $\mathcal{L}$ be the set that contains all octree leaves. With this definitions case (4.14) can be further split into the two remaining distinct cases:

- There exists a leaf L_j that contains measurements from S_k that occluded L_i, i.e.,

$$\exists\, L_j \in \mathcal{L} \;:\; L_j \not\oslash \overline{L_i\check{S}_k} \wedge L_j \cap \check{P}_k \neq \emptyset \tag{4.16}$$

- Such a leaf L_j does not exist, i.e.,

$$\forall\, L_j \in \mathcal{L} \;:\; \neg \left(L_j \not\oslash \overline{L_i\check{S}_k} \right) \vee L_j \cap \check{P}_k = \emptyset \tag{4.17}$$

  meaning that the leaves L_j are either not intersected by line segment $\overline{L_i\check{S}_k}$ or do not contain any points $\check{p}_{l,k} \in \check{P}_k$.

In case that both equations (4.14) and (4.16) simultaneously hold leaf L_i does not directly provide evidence for correct alignment, but also does not contradict the notion that the alignment of P_1 and P_2 via Θ_{ICP} is correct.

However, if equations (4.14) and (4.17) hold for a given leaf L_i, this provides information that the alignment given by pose Θ_{ICP} is not correct.

Note that the octree is created from the combined point cloud that is already restricted to the region of overlap, i.e., $\bigcup_{L_i \in \mathcal{L}} L_i = ROO_{1,2} \cap (P_1 \cup \Theta_{\text{ICP}} \cdot P_2)$. Otherwise in addition to the two cases expressed by equations (4.13) and (4.14) a third case becomes necessary, where $L_i \cap ROO_{1,2} = \emptyset$ is tested. In this case L_i would not contribute to the evaluation matter.

The three defined cases (4.13), (4.14) with (4.16), and (4.14) in combination with (4.17) can now be used to decide if a particular leaf L_i provides evidence for or against proper alignment. This can be employed in a straight forward fashion to define an evaluation measure. Let set $\mathcal{L}^+$ contain all leaves $L_i \in \mathcal{L}$ that indicate correct alignment via Θ_{ICP} and $\mathcal{L}^-$ contain all leaves that contradict the notion of correct alignment. Based on these sets a first error measure for the alignment e_{align} is defined as

$$e_{\text{align}} = \frac{|\mathcal{L}^-|}{|\mathcal{L}^-| + |\mathcal{L}^+|} \tag{4.18}$$

The required sets $\mathcal{L}^+$ and $\mathcal{L}^-$ can be established by iterating over all octree leaves once and investigating their contents. This is described in detail in Algorithm 1. Note that in general $|\mathcal{L}^+| + |\mathcal{L}^-| \neq |\mathcal{L}|$, since leaves that are occluded from either S_1 or S_2 are included into neither $\mathcal{L}^+$ nor $\mathcal{L}^-$. After establishing both sets $\mathcal{L}^+$ and $\mathcal{L}^-$, a post-processing step is oftentimes sensible, since both sets may now include leaves that contain only a single point in their volume. In order to reduce the influence of outliers and other sensor noise removing leaves $L_i \in \mathcal{L}^+ \cup \mathcal{L}^-$ where $|L_i| < n_{\text{min}}$ from the corresponding set is a feasible solution. The threshold n_{min}, that determines when a leaf is discarded from the evaluation, should be set with consideration of the point density near the outer borders of the $ROO_{1,2}$, that should still be considered in the evaluation. However, in case of high quality sensors that provide very accurate measurements and if point clouds P_1 and P_2 were subject to outlier removal and other filtering techniques, this post-processing step may also be omitted, without compromising the results.

By its definition in equation (4.18) e_{align} is bounded in the interval $[0, 1]$ and indicates the error in the proposed alignment, where 0 indicates that no registration error was detected, while 1 would constitute a completely wrong alignment.

If a confidence measure is preferred Instead of an error measure this can be generated from sets $\mathcal{L}^+$ and $\mathcal{L}^-$ in a similar fashion:

$$c_{\text{align}} = \frac{|\mathcal{L}^+|}{|\mathcal{L}^-| + |\mathcal{L}^+|} \tag{4.19}$$

In this case higher values of c_{align} correspond to better alignment of point clouds P_1 and P_2. Naturally c_{align} is also bounded in the interval $[0, 1]$ and $e_{\text{align}} + c_{\text{align}} = 1$ holds, i.e., high confidence is accompanied by low error and vice versa, as one would intuitively expect.

Incidentally iterating over all leaves $L_i \in \mathcal{L}$ in the fashion of Algorithm 1 implicitly also handles all volumes in $ROO_{1,2}$ where neither sensors S_1 nor S_2 recorded any measurements, since these volumes do not constitute any octree leaves. In principle it is also possible to use them as positive evidence for correct alignment, as the dual of the case expressed by (4.13), but in the proposed measure e_{align} we are satisfied with the implicit handling, so that these volumes do not contribute to e_{align}.

Algorithm 1 Creating sets $\mathcal{L}^+$ and $\mathcal{L}^-$ from all points in the *region of overlap* $ROO_{1,2}$. For the definition of $\check{P}_i$ and $\check{S}_i$ refer to (4.11) and (4.12) respectively.

```
Inputs:
    $\mathcal{L}$– the set of octree leaves of filtered point cloud $ROO_{1,2} \cap (P_1 \cup \Theta_{\text{ICP}} \cdot P_2)$
Outputs:
    $\mathcal{L}^+$– the set of leaves that support correct alignment
    $\mathcal{L}^-$– the set of leaves that contradict correct alignment
Initialize:
    $\mathcal{L}^+ \leftarrow \emptyset, \mathcal{L}^- \leftarrow \emptyset$
for all $L_i \in \mathcal{L}$ do
    if $L_i \cap P_1 \neq \emptyset \wedge L_i \cap \Theta_{\text{ICP}} \cdot P_2 \neq \emptyset$ then
        $\mathcal{L}^+ \leftarrow \mathcal{L}^+ \cup L_i$
    else
        if $L_i \cap P_1 = \emptyset$ then
            if $\neg\exists L_j \in \mathcal{L} \ : \ L_j \not\diamond \overline{L_i\check{S}_1} \wedge L_j \cap \check{P}_1 \neq \emptyset$ then
                $\mathcal{L}^- \leftarrow \mathcal{L}^- \cup L_i$
            end if
        else                                                    $\triangleright$ $L_i \cap \Theta_{\text{ICP}} \cdot P_2 = \emptyset$
            if $\neg\exists L_j \in \mathcal{L} \ : \ L_j \not\diamond \overline{L_i\check{S}_2} \wedge L_j \cap \check{P}_2 \neq \emptyset$ then
                $\mathcal{L}^- \leftarrow \mathcal{L}^- \cup L_i$
            end if
        end if
    end if
end for
```

Naturally the size of the octree leaves, parametrized by l_L(the length of an edge of the cubic volume) has a large influence on error measure c_{align}: choosing the volume too small will negate the benefits introduced by the discretization to a coarser scale, namely compensation of

Category	Color	Explanation
Supporting	green	Leaves $L_i \in \mathcal{L}^+$, i.e., case (4.13) applies for leaf L_i
Contradicting	red	Leaves $L_i \in \mathcal{L}^-$, i.e., case (4.14) applies and subsequently case 4.17) also holds
Neutral	yellow	Leaves L_i that are inside $ROO_{1,2}$, but where both cases (4.14) and (4.16) apply
Unconsidered	blue	Octree leaves L_i that are located outside of the region of overlap $ROO_{1,2}$

Table 4.4: Color information for direct modelling visualization, shown in Figure 4.7

different point densities and lowering the influence of sensor noise. If taken to an extreme, for a given point cloud $ROO_{1,2} \cap (P_1 \cup \Theta_{\text{ICP}} \cdot P_2)$ an octree could be fashioned in such a way that each leaf L_i would contain exactly 1 point coordinate (or in rare cases two points, when $p_{l,1} = \Theta_{\text{ICP}} \cdot p_{k,2}$ exactly holds for some points $p_{l,1} \in P_1$ and $p_{k,2} \in P_2$). In this case for (almost) all leaves L_i equation (4.14) holds and subsequently the intersection criteria, i.e., equations (4.16) and (4.17) are employed to determine if L_i was occluded from one sensor. The smaller the volume of the octree leaves the more similar (4.16) becomes to determining if line segment $\overline{L_i \check{S}_k}$ intersects an individual point $\check{p}_{j,k} \in \check{P}_k$. The latter is, of course, very sensitive to sensor noise and minimal errors in the alignment between P_1 and P_2. Thus very small leaves could lead to $|\mathcal{L}^+| \ll |\mathcal{L}^-|$ and therefore to $c_{\text{align}} \approx 0$ even if a human observer might consider the alignment provided by Θ_{ICP} as good. On the opposite side of extremes the complete point cloud $ROO_{1,2} \cap (P_1 \cup \Theta_{\text{ICP}} \cdot P_2)$ could be contained in an octree, where the root node also serves as the only leaf. Consequently confidence measure c_{align} would consider this, independent from Θ_{ICP} as a good alignment, since in this case $|\mathcal{L}^-| = 0$ holds and consequently also $c_{\text{align}} = 1$.

However, the appropriate leaf size seems to be dependent on the sensor specifics, i.e., ρ_{max}, and the sensor noise; not scene dependent, so that, if a good performing leaf size is established for a given sensor, this parameter can be re-used for the alignment evaluation from similar sensors. Subsection 5.2.2 will provide some empirical evidence for this claim.

This property is different to the input parameters of the ICP algorithm, which are not only dependent on the employed sensor, but also depend on the current scene, the quality of the initial pose estimation, the actual overlap between P_1 and $\Theta_{\text{GT}} \cdot P_2$ when aligned via ground truth as well as the distance between sensors S_1 and S_2, when aligned correctly.

A visualization of example of the evaluation is presented in Figure 4.7 (corresponding numerical results are shown in Tables 4.7 and 4.8). The different classifications of the octree $L_i \in \mathcal{L}$ created from the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ are conveyed via the colors of the points that are located inside the individual leaves. The meaning of the different colors is conveyed via Table 4.4.

Now that the meaning of the different colors has been established, let us take a more detailed look at the differences between the correct alignment and the two erroneous arrange-

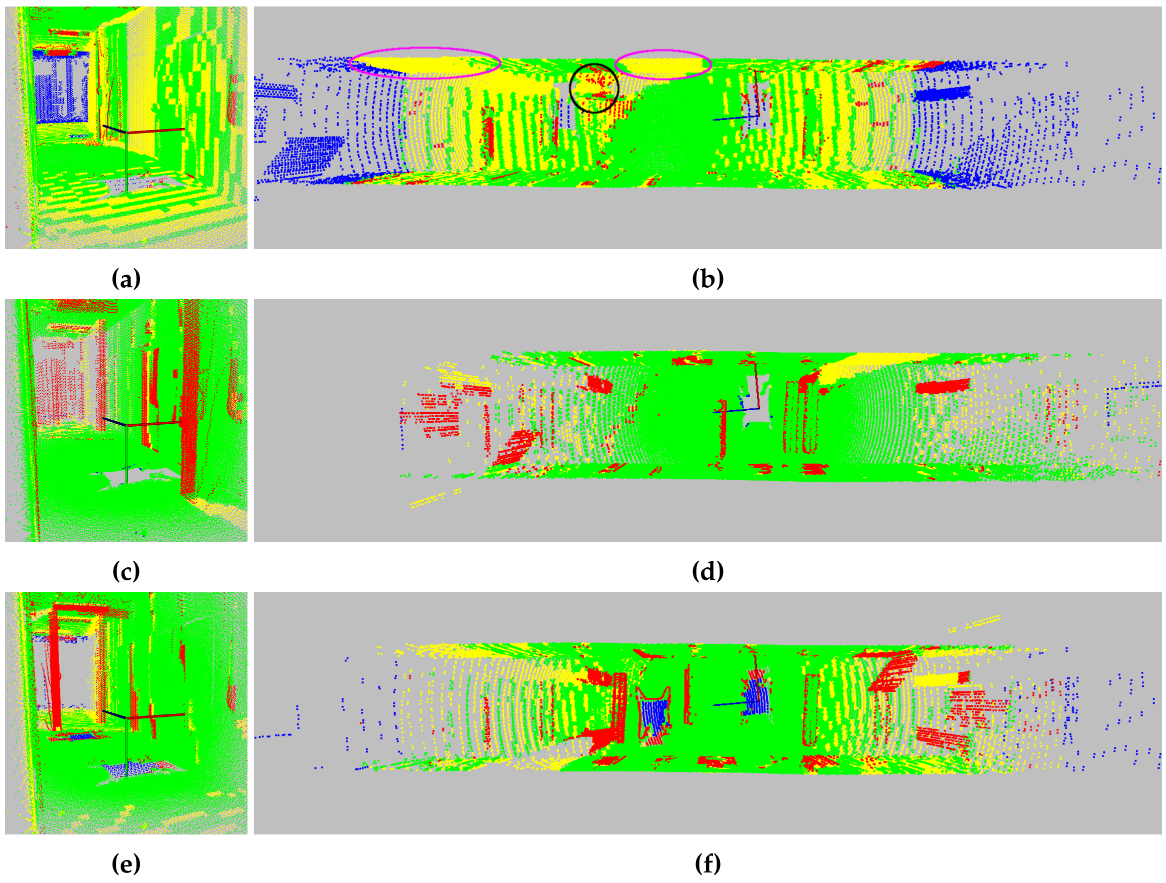


Figure 4.7: Visualization of direct modelling evaluation. Results from top to bottom display correct alignment, translation error and rotation error respectively. Left-hand side displays the point clouds from a viewpoint inside the hallway while right-hand side shows a top view perspective (ceiling points removed). For more details and the meaning of the colors please refer to the accompanying text and Table 4.4.

ments shown in Figure 4.7. When observing all scenes from above (right column in Figure 4.7) it is obvious that the correct alignment in Figure 4.7a contains the most blue colored points, i.e., volumes that are not part of the region of overlap. If we consider the alignment of the sensor positions in the three examples these results should not come as a surprise:

The distance between the sensor positions in the correct alignment is larger than in the two erroneous alignments (see also Figure 4.4 for a better view of the sensor positions). Hence, it is natural that larger portions of the combined point cloud could not have been observed by both sensors, if we assume that the given alignment is correct. This is due to the maximal volume that the region of overlap can attain being identical with the smaller of volumes OV_1 and OV_2. This maximal extent is realized, when sensor positions coincide and the restricition in the field of view are aligned in a suitable fashion (in case of the 360 $^\circ$ rotating scanner only positions and "up"-vector have to coincide).

Furthermore, we can observe that the alignment with the translation error (Figures 4.7c, 4.7d) contains the least amount of yellow colored leaves, i.e., leaves that contain only points from one sensor, but that could be explained by the available point data. The correct alignment in Figure 4.7b and, albeit to a lesser extend, also the alignment with the rotation error in Figure 4.7f contain yellow "curved arcs" on the floor, ceiling and walls. This is caused by the sensor specifics, where the point density is strongly reduced with increasing distance to the sensor's position. This sensor characteristic can be observed in Figures 4.3b and 4.3b where the distance between the scan lines on the floor is strongly increasing with the distance to the sensor. In these cases line segment $\overline{L_i \check{S}_k}$ usually intersects a neighboring leaf on the floor, closer to sensor location $\check{S}_k$, hence the leaf is not counted as evidence against correct alignment. A larger parameter l_L can, to a certain degree, reduce the number of leaves that are grouped into this category, caused by decreasing point density.

In case of the translation error (Figure 4.7d) these scan lines from both sensors are positioned very closely to each other, since the sensor poses almost coincide. Therefore, most octree leaves on the floor, ceiling and walls contain points gathered from both sensors, even when the distance to the sensor positions increases. If, at larger distances from the sensor positions, the gap between scan lines becomes larger, it can also happen that the octree representation contains a gap between the leaves, i.e., there may exist an unoccupied volume positioned between two leaves on the large planar surfaces.

Apart from these somewhat surprising locations, occluded leaves are encountered where they are naturally expected. Exemplary this can be observed in Figure 4.7b: the pillar (marked by a black circle) that is positioned between sensor positions $S_{\text{pos},1}$ and $S_{\text{pos},2}$ causes two occlusions on the wall in the top of the Figure (marked by magenta ellipses). Similar occlusions can be observed in the incorrectly aligned point clouds in Figures 4.7d and 4.7f.

When investigating the volumes that indicate misalignment we can firstly observe that the correct alignment in the top row of Figure 4.7 also contains these. Reasons for this are twofold: while the alignment presented in the top row is the best of the three displayed solutions, it is still not perfect, i.e., there are still some improvements that could be made, if ICP parameters would have been more fine-tuned for this specific scene – a possibility I chose to forego, since overall this would not have conveyed more understanding of the total process. Furthermore, the proposed evaluation methods should work on and be able to distinguish "sufficiently good" aligned scans from worse alignments and not only be able to identify a "perfect" alignment in a collection of bad scan arrangements.

The second reason for the occurrence of sensor noise and measurement artifacts. The latter may occur with the used sensor setup when the sensor observes a distinct edge at a steep angle. An example case is displayed in Figure 4.8, where the point cloud of an emergency exit sign mounted below the ceiling is depicted. In the top row (Figures 4.8a, 4.8b) the points in from point cloud $\Theta_{\text{ICP}} \cdot PC2$ are shown, while the bottom row (Figures 4.8c, 4.8d) depicts the same segment of the scene from the same perspectives, but with points from point cloud P_1 instead. Reference photographs of the actual sign is shown in Figure 4.9.

If we compare the top and the bottom row of Figure 4.8, differences in point density are noticeable: sensor S_1 observed this part of the scene from a larger distance, while sensor S_2 was placed almost below the emergency exit sign. This manifests itself in the number of points in the cutout of the combined point cloud. In total there are 22175 points of $\check{P}_2$ present, while only

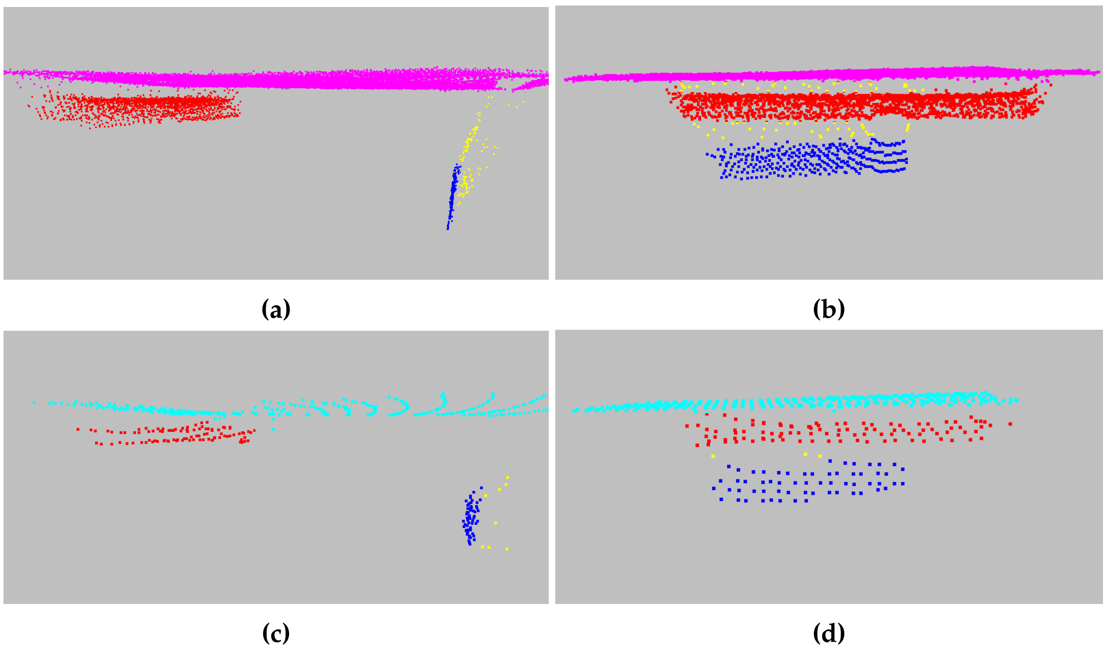


Figure 4.8: Illustration of sensor noise at sharp edges. Top row shows a clipping from the point cloud P_2 (see Figures 4.3c and 4.3d) containing an emergency exit sign, while the bottom row shows the same region from the same perspectives in point cloud P_1 (Figures 4.3a and 4.3b). Left column shows a side view and the right-hand side provides a frontal view. Sensor noise at sharp edges is displayed in yellow. A photograph of the depicted sign can be found in Figure 4.9.

549 points of P_1 are contained in the volume. The positioning of the sensors does not only have an influence on the number of points contained in the cutout, but also on the type of sensor noise that this part of the scene contains.

If we look at the top row of Figure 4.8, where the measurement rays of sensor S_2 encountered the sign (blue points) at a steeper angle, a large number of points can be observed that appear behind the upper and lower edge of the sign, when observed from the sensor's position. These points are marked in yellow in Figure 4.8 and when considering the corresponding photographs, depicted in Figure 4.9, we would not expect any points in these locations. Note that these points are located along measurement rays that (partially) pass along the edges, as evident in Figure 4.8a, where the yellow points lead up to the occlusion of the sign caused on the ceiling (magenta points).

Sensor S_1 observed this part of the common scene from a larger distance, which in turn results in a less steep angle between the measurement rays and the sign surface. This causes much less noise of this specific nature, as is evident in the bottom row of Figure 4.8, where the sign is again colored in blue, the noise caused from the edges is shown in yellow and points belonging to the ceiling are depicted in cyan.

While the recording of the sign in point cloud P_1 contains much less noise caused by a steep measurement angle and edges, it contains more "regular" sensor noise, i.e., the overall accuracy of the measurements decreases. This becomes evident if we compare the two side

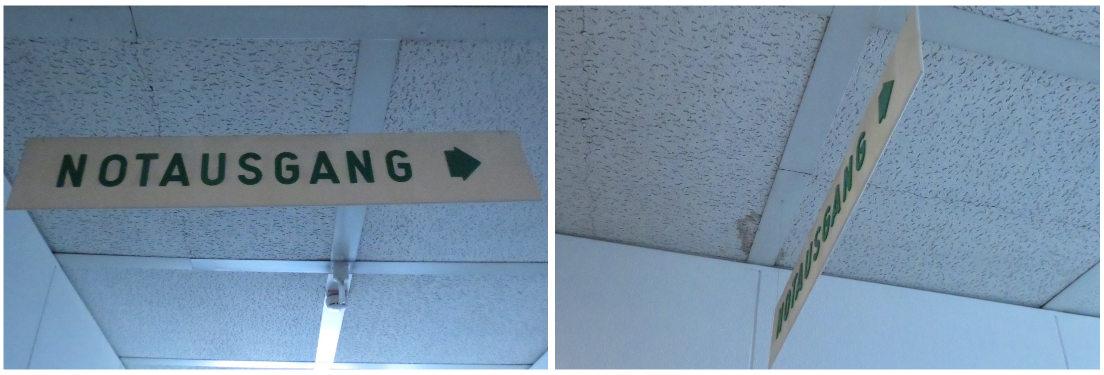


Figure 4.9: Photograph of emergency exit sign, corresponding with the points shown in Figure 4.8. Notice, that, as opposed to what the 3D data suggest, the sign is actually very flat and mounted by two very thin, strings, almost invisible in the photograph.

views Figure 4.8a and 4.8c with each other: the sign in the former is depicted as a almost planar surface, but the corresponding points in Figure 4.8c are much more scattered, causing a wider spread along the z-axis in the scan. The shape of the sign in the frontal view (Figures 4.8b and 4.8d) also appears differently in both point clouds. The points in the frontal view of point cloud $\check{P}_2$ reflect the slightly skewed mounting of the rectangular sign below the ceiling (compare with Figure 4.2a for a general scene overview), it becomes more difficult to interpret the blue points in Figure 4.8d as a rectangular object. If we compare the shape and appearance of the ceiling lamp (red points in Figure 4.8) it is not immediately evident that they belong to the same object, without further information.

Observing this input data it should not be too surprising, that even, if we had “ground truth” available, comparison with the direct modelling approach might perceive some inconsistencies in the data, although the comparison, thanks to the octree, takes place on a more granular scale, as compared to point to point comparisons.

After this, rather detailed, discussion on how to interpret the visualization of the direct modelling approach, it is time to take a first look at the results of the evaluation measure. Table 4.5 offers a direct comparison of the ICP output parameters for the registration of both, full point cloud and octree filtered version, alongside their proposed counterparts c_{align} and e_{align}. From the results shown in Table 4.5 we can already see that the proposed confidence measure c_{align} succeeds in assigning the highest confidence to the alignment closest to a correct “ground truth” alignment, even in a rather challenging scenario as the depicted hallway.

As a side note, please notice that sensor noise of this nature is quite common in low cost laser scanners, RGB-D cameras and 3D Time of Flight measurement systems. Filter techniques to identify likely outliers of such a nature exist, as demonstrated by May et al. [53] in the context of ToF-cameras, but again not too much work was invested in outlier removal and filtering in the presented scenes, since a practical evaluation method should also be able to cope with a certain amount of sensor noise without breaking down. Employing sensors with more precision and higher point density at relatively short measurement ranges can augment these problems

	initial pose	e_{avg}	$\|\|C_{1,2}\|\|$	c_{align}	e_{align}
Full scan	$\Theta_{\text{good\_init}}$	4.73 cm	65257	0.881	0.119
	$\Theta_{\text{trans\_error}}$	2.81 cm	84042	0.849	0.151
	$\Theta_{\text{rot\_error}}$	3.57 cm	79985	0.793	0.207
Reduced scan	$\Theta_{\text{good\_init}}$	5.06 cm	20695	0.918	0.082
	$\Theta_{\text{trans\_error}}$	4.07 cm	27582	0.842	0.158
	$\Theta_{\text{rot\_error}}$	4.48 cm	24933	0.797	0.203

Table 4.5: Comparison of ICP output with direct modelling confidence measure c_{align} and error measure e_{align}. Full scan results refer to the unfiltered point clouds with a maximal number of $|P_2| = 150594$ point correspondences (see Figure 4.4). Below, the results for the scans filtered by an octree (Figure 4.5) are shown, where the maximum of point correspondences is $|P_2| = 35819$.

to a certain degree. As indicated in section 2.4 such sensors unfortunately also feature a much higher price tag, so that these problems are issues that need to be addressed, when dealing with common low-cost sensors, used in mobile robotics, for example.

The previously discussed perceived misalignments, due to noise and measurement characteristics, may, of course, also occur in incorrectly aligned scans. In addition those parts of the combined point cloud that actually do not match are also detected. For example in the incorrectly aligned scenes depicted in Figures 4.7c – 4.7f measurement points on the surfaces of the pillars, open doors, etc. form noticeable red clusters in the visual representation and lead to $|\mathcal{L}^-|$ being larger in relation to set $|\mathcal{L}^+|$, when compared to correct alignment (see Table 4.5).

The observation that detected misalignments are arranged in larger clusters in the incorrect registrations, while they appear more distributed in the alignment resultant from initial pose $\Theta_{\text{good\_init}}$, leads to the new idea, that not only the cardinality of sets $\mathcal{L}^+$, $\mathcal{L}^-$ and their relation to each other could be employed to assess the alignment quality, but the in-scene distribution of the detected misalignments in the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ might yield additional information that should be used. This idea will be elaborated further in section 4.5, where a different method to model mutual observability is introduced and subsequently also extended to the direct modelling approach.

As mentioned before the choice of the octree resolution, i.e., the length of an edge of a leaf is important, as is the threshold n_{min} that determines if a leaf is considered to be occupied or not. For the results presented in Figure 4.7, a leaf size of $l_L = 10$ cm was used and threshold n_{min} was set to 5 and different values for these parameters will lead to different results in terms of appearance and the values of e_{align} and c_{align}, respectively. The influence of these parameters will be illustrated in section 5.2 in the evaluation chapter.

4.5 Ray Tracing Approach

The approach for the introduced measure c_{align} in section 4.4 explicitly models the concept of Mutual Observability, introduced in section 4.3: first, the registered point clouds $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$

are restricted to those points that can potentially have been observed by both sensors S_1 and S_2, by the creation of the region of overlap $ROO_{1,2}$. Subsequently, an octree is generated from the filtered data and all those parts of the scene are determined, that are actually observed from both sensors, culminating in $\mathcal{L}^+$, the set of octree leaves that indicate correct alignment. All leaves that contradict correct alignment are gathered in set $\mathcal{L}^-$. In the construction of $\mathcal{L}^-$ the incorporation of occlusion information is vital, so that only leaves are included if no in-scene occlusion explains why they are not mutually observable. In this section an alternative approach is presented, that will employ Mutual Observability in a less direct modeled fashion, that is based on ray tracing.

Similar to the explicit modelling an octree is used in order to reduce the influence of different point densities and sensor noise and to allow for a sensible occlusion analysis. In contrast to the previously detailed approach, the combined point cloud that is used to create the octree is not restricted to the region of overlap, but will include all measurement data. Therefore, in this section the set of all leaves $\mathcal{L}$ will contain all points $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ instead of its subset $ROO_{1,2} \cap (P_1 \cup \Theta_{\text{ICP}} \cdot P_2)$.

Similar to the direct modelling approach the sensor parameters (see 4.4.1) are needed to create this evaluation measure. While the direct modelling approach employed these parameters exclusively to generate the region of overlap, the ray tracing approach utilizes sensor information for the evaluation of each measurement ray. In addition to the opening angles $\omega_{\text{h},k}$ and $\omega_{\text{v},k}$ for sensor S_k the angle increments $\Delta\omega_{\text{h},k}$ and $\Delta\omega_{\text{v},k}$ between adjacent measurements are needed. When employing angle increments it is explicitly assumed that measurements are arranged in an organized, grid-like structure, where each measurement point, that is not located at the borders of the field of view has exactly four adjacent measurements (above, below, left and right) with respect to the sensor's reference coordinate system. The horizontal angle increment $\Delta\omega_{\text{h},k}$ specifies the angle between the measurement ray belonging to some measurement $p_{l,k}$ and the ray for an adjacent measurement $p_{m,k}$ to the left or right. Conversely $\Delta\omega_{\text{v},k}$ denotes the angle between the measurement of $p_{l,k}$ and the adjacent measurement rays above or below in the grid-like structure.

Instead of the angle increments the resolution of sensor S_k, i.e., the number of measurements points that it records in a single frame denoted by width $\times$ height, along with opening angles $\omega_{\text{h},k}$ and $\omega_{\text{v},k}$ also provides the necessary information, since $\Delta\omega_{\text{h},k}$ and $\Delta\omega_{\text{v},k}$ can be easily derived from this information. Angle increments $\Delta\omega_{\text{h},k}$, $\Delta\omega_{\text{v},k}$ in connection with opening angles $\omega_{\text{h},k}$ and $\omega_{\text{v},k}$ are used to generate the set of all measurement rays for sensor S_k.

These requirements (grid-like arrangement of measurements and resolution / angle increments) are met by most depth cameras or 3D laser range finders. The needed information can be obtained for most common sensors in their data sheet and are a convenient way to express all measurement rays of a single scan in a compact way. If some sensor does not use such an evenly distributed field of measurement rays, but their arrangement is known, the approach introduced in the following can still be applied, with minimal changes. Since ordering and arrangement of the measurement rays are unimportant, as long as a set of all measurement rays for a single scan are known, a set containing (unorganized) rays that model a specific sensors characteristics is as valid as a set of rays generated by parameters $\Delta\omega_{\text{h},k}$, $\Delta\omega_{\text{v},k}$, $\omega_{\text{h},k}$ and $\omega_{\text{v},k}$.

Let R_k denote the set of measurement rays for sensor S_k and $r_{l,k} \in R_k$ denote an individual ray from the set. Coupled with the set of octree leaves $\mathcal{L}$ from the combined point $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$

R_k can be employed to create quality / error measures, similar to c_{align} and e_{align}.

In the following the most basic version of these measures is introduced and later extended to slightly more complex variants. Since the combined point cloud was recorded from two different viewpoints there are two different sets of measurement rays, namely R_1 and R_2. In accordance with the previous conventions the origin of all rays $r_{l,k} \in R_k$ will coincide with the position of the sensor that emitted the measurement ray, namely $\check{S}_k$ (see (4.12)).

For each individual measurement ray $r_{l,k}$ from sets R_k, $k \in \{1, 2\}$ the intersection with the leaf $L_{\min} \in \mathcal{L}$ that is closest to the ray's origin $\check{S}_k$ is investigated. Similar to the intersection of an octree leaf by a line segment (see equation (4.15)), we let

$$L_i \not\diamond r_{l,k} \tag{4.20}$$

denote that leaf L_i was intersected by the ray $r_{l,k}$, originating from $\check{S}_k$.

Employing this notation, $\mathcal{L}^{\not\diamond}_{l,k}$ denotes the set of all octree leaves that are intersected by ray $r_{l,k}$, i.e.,

$$\mathcal{L}^{\not\diamond}_{l,k} = \bigcup_i L_i \in \mathcal{L} \ : \ L_i \not\diamond r_{l,k} \tag{4.21}$$

Thus the first leaf intersected by ray $r_{l,k}$ will be associated with it and denoted as

$$L_{l,k} = \arg\min_i \mathrm{d}\left(L_i, \check{S}_k\right), \ L_i \in \mathcal{L}^{\not\diamond}_{l,k} \tag{4.22}$$

where $\mathrm{d}\left(L_i, \check{S}_k\right)$ denotes the Euclidean distance between the center of octree leaf L_i and $\check{S}_k$, the origin of $r_{l,k}$.

If we consider a single ray $r_{l,k} \in R_k$ that originated from position $\check{S}_k$, there are three distinct cases concerning the first leaf $L_{\min}$ that this ray intersects when travelling from its origin $\check{S}_k$:

- The leaf $L_{l,k}$ contains points from sensor S_k, i.e.,

$$L_{l,k} \cap \check{P}_k \neq \emptyset \tag{4.23}$$

- The leaf $L_{l,k}$ does not contain any point from S_k, i.e.,

$$L_{l,k} \cap \check{P}_k = \emptyset \tag{4.24}$$

- There does not exist any leaf that is intersected by measurement ray $r_{l,k}$, i.e.,

$$\neg \exists \, L_i \in \mathcal{L} \ : \ L_i \not\diamond r_{l,k} \tag{4.25}$$

These three distinct cases have different implications when thinking about the alignment of both point clouds in $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$.

If we consider only a single scan, say P_k and not the combined point cloud, then in the associated set of measurement rays R_k all rays either belong to categories (4.23) or (4.25). The latter will happen if there is no measurable surface within the measurement range of S_k in the direction of ray $r_{l,k}$, while the former case will apply for all valid measurements gathered from sensor S_k. In case of organized point clouds case (4.23) corresponds with regular points $p_n k \in P_k$ while case (4.25) corresponds to the `NaN`-points that are included in the organized point cloud. The remaining case (4.24) cannot occur, when considering single point clouds.

Investigating the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ should almost yield the same results, provided that the alignment via pose Θ_{ICP} is correct: each measurement ray $r_{l,k}$ originating from $\breve{S}_k$ should either not intersect any leaf or the first intersected leaf should be in category (4.23), if it is in measurement range. This restriction becomes necessary if we want to be able to take into account that some object might be too close to sensor $\breve{S}_k$ to be observed, but cause occlusion in other parts of the scene. If we only consider the individual sensor $\breve{S}_k$ and the associated point cloud $\breve{P}_k$, then such a setting will result in measurement rays of category (4.25), as mentioned above. However, if we consider the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ the aforementioned object might be perfectly observable and in measurement range from sensor $\breve{S}_m$ ($k, m \in \{1, 2\}$ with $k \neq m$).

The dual scenario is also possible, i.e., an obstacle is beyond the maximal measurement range $\rho_{\max,k}$ of $\breve{S}_k$ in the combined point cloud, but was inside the range of $\breve{S}_m$. Both these cases may occur even if P_1 and P_2 are registered perfectly and should therefore not be considered as evidence for wrong alignment. Expressed in the terms of section 4.4 such intersected leaves are not part of the region of overlap $ROO_{1,2}$.

The previous considerations show that it is sensible to distinguish between the first leaf $L_{\min}$, intersected by measurement ray $r_{l,k}$, and the first intersected leaf in the sensor's measurement range, if it exists. Let us formalize the latter as

$$L^{\rho}_{l,k} = \arg\min_{i} \mathrm{d}\left(L_i, \breve{S}_k\right), L_i \in \mathcal{L}^{\not\emptyset}_{l,k} \; : \; \rho_{\min,k} \leq \mathrm{d}\left(L_i, \breve{S}_k\right) \leq \rho_{\max,k} \tag{4.26}$$

with $\mathrm{d}\left(L_i, \breve{S}_k\right)$ denoting the Euclidean distance between the center of octree leaf L_i and $\breve{S}_k$, the origin of $r_{l,k}$. Of course, similar to $L_{\min}$ (see equation (4.22) and case (4.25)) a leaf $L^{\rho}_{l,k}$ does not need to exist for a given ray $r_{l,k}$.

If it turns out that the first leaf intersected by ray $r_{l,k}$ is also in measurement range ($L_{l,k} = L^{\rho}_{l,k}$) and case (4.24) applies, i.e., $L^{\rho}_{l,k} \cap \breve{P}_k = \emptyset$ holds, then this ray provides evidence against correct alignment. Since $L^{\rho}_{l,k}$ is also the first leaf intersected by ray $r_{l,k}$ its volume cannot be occluded by any surface placed between the ray's origin $\breve{S}_k$ and $L^{\rho}_{l,k}$. If both point clouds P_1 and P_2 were aligned correctly via pose Θ_{ICP}, $L^{\rho}_{l,k}$ should either only contain points from $\breve{P}_k$ (in this case the leaf might either not be in the region of overlap or occluded from $\breve{S}_m$, $k, m \in \{1, 2\}, k \neq m$) or $L^{\rho}_{l,k}$ should contain points from both P_1 and $\Theta_{\text{ICP}} \cdot P_2$. The fact that measurement ray $r_{l,k}$ first encounters a leaf that is in its measurement range and does not contain any measurement point originating from sensor S_k cannot be explained with the underlying sensor model. Therefore, it is prudent to assume that the alignment provided by Θ_{ICP} is incorrect.

Connecting these cases with the concept of mutual observability, introduced in section 4.3 is straight forward. The last discussed case ($L_{l,k} = L^{\rho}_{l,k} \wedge L^{\rho}_{l,k} \cap \tilde{P}_k = \emptyset$) corresponds to a leaf that is encountered in the region of overlap, not occluded from any sensor, but contains only points gathered by one sensor (namely S_m). According to the sensor modelling this leaf is mutually observable, but this cannot be verified by its data. Rays, where the first encountered leaf is not in measurement range or does not exist, can be considered to go through partitions of space which are not mutually observable: either they are not part of the region of overlap $ROO_{1,2}$, which is the case when a measurement ray $r_{l,k}$ intersects only leaves beyond the maximal measurement range, i.e., $\rho_{\max,k} < \mathrm{d}\left(L_{l,k}, \check{S}_k\right)$ holds. Otherwise the ray may first intersect with a leaf that is below the minimal measurement range ($\mathrm{d}\left(L_{l,k}, \check{S}_k\right) < \rho_{\min,k}$), that effectively occludes all subsequently intersected leaves from position $\check{S}_k$. Lastly, if no leaf was intersected by ray $r_{l,k}$ we can conclude that no measurable object was in sensor range along this ray, independent of the fact if the encountered (empty) volumes might be placed inside the region of overlap of are in theory mutually observable. All these different possibilities have in common that they do not provide us with any information concerning the alignment of P_1 and P_2.

Finally we need to consider those rays that are the equivalent to the mutual observable volumes of the approach detailed in section 4.4, i.e., the equivalent to set $\mathcal{L}^+$. In the ray tracing approach, this is modelled by those measurement rays, where the intuitive expectation concerning the contents of the first intersected leaf for ray $r_{l,k}$ is met, meaning that the first intersected leaf is expected to be in measurement range and contain points recorded by sensor S_k. This can be formally expressed as $L_{l,k} = L^{\rho}_{l,k}$ holds and $L^{\rho}_{l,k} \cap \check{P}_k \neq \emptyset$. This expression still includes leaves that are not mutually observable: leaves that are outside the region of overlap $ROO_{1,2}$, i.e., only in the field of view of a single sensor, are also included, provided that they are not occluded by something else. To ensure that that these leaves are inside the $ROO_{1,2}$, a check concerning potential visibility for sensor S_m needs to be performed. This can be done via inequalities (4.4) – (4.8), when point $\tilde{p}_{i,j}$ needs to be replaced with $\tilde{L}_{l,k}$, the center point of leaf $L_{l,k}$ transformed into the coordinate system of sensor S_m ($k, m \in \{1, 2\}$ and $k \neq m$). In this fashion field of view restrictions need only be checked on a subset of the octree leaves $\mathcal{L}$, when necessary and not on all points in $\Theta^{-1}_{\mathrm{ICP}} \cdot P_1$ and $\Theta_{\mathrm{ICP}} \cdot P_2$.

Without explicitly modelling the concept of mutual observability, investigation of all measurement rays $R_1 \cup R_2$ provides a similar result concerning the alignment assessment of P_1 and P_2. While the approach detailed in section 4.4 investigates each volume of the combined point cloud that is inside the region of overlap, the basic idea in this approach is to check which measurement rays confirm with the expected result, if P_1 and P_2 were aligned correctly via pose Θ_{ICP}.

An inherent advantage of evaluating the measurement rays, when compared to the direct modelling approach introduced in section 4.4, is that it implicitly considers the reduction in measurement point density with increasing distance to the sensor. The correct alignment in Figure 4.7 produced a lot of leaves that were only filled by points collected from one sensor and regarded as occluded from the other sensor (cases (4.14) and (4.16) apply). Oftentimes these classification was due to different point densities of P_1 and $\tilde{P}_2$ at this portion of the combined point cloud. Since the spread between the individual measurement rays $r_{l,k}$ of sensor S_k grows with increasing distance to S_k and these rays are investigated, not every octree leaf is implicitly assumed to contain points of both sensors in order to indicate correct alignment of P_1 and P_2 via pose Θ_{ICP}.

Algorithm 2 details how this translates into the generation of two sets $\mathcal{R}^+$ and $\mathcal{R}^-$ of measurement rays that either support or refute the notion of correct alignment in a very similar fashion to Algorithm 1 that created the leave sets $\mathcal{L}^+$ and $\mathcal{L}^-$. Function is_in_FoV(L_i, S_j) can check if a given leaf L_i is in the field of view of sensor S_j in its current pose.

Algorithm 2 Creating sets $\mathcal{R}^+$ and $\mathcal{R}^-$ from combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ and the sets of measurement rays R_1 and R_2.

```
Inputs:
    $R_1$, $R_2$ – the sets of measurement rays starting from sensor positions $\text{pos}_{S_1}$
    and $\text{pos}_{\breve{S}_2}$
    $\mathcal{L}$ – the set of octree leaves of combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$
Outputs:
    $\mathcal{R}^+$– the set of measurement rays that support correct alignment
    $\mathcal{R}^-$– the set of measurement rays that contradict correct alignment
Initialize:
    $\mathcal{R}^+ \leftarrow \emptyset, \mathcal{R}^- \leftarrow \emptyset$
for all $r_{l,k} \in (R_1 \cup R_2)$ do
    $\mathcal{L}^{\not\phi}_{l,k} \leftarrow \bigcup_i L_i \in \mathcal{L} \ : \ L_i \not\phi \, r_{l,k}$
    if $\mathcal{L}^{\not\phi}_{l,k} \neq \emptyset$ then
        $L_{l,k} \leftarrow \arg\min_i \mathrm{d}\left(L_i, \breve{S}_k\right), \ L_i \in \mathcal{L}^{\not\phi}_{l,k}$
        if $\rho_{\min,k} \leq \mathrm{d}\left(L_{l,k}, \breve{S}_k\right) \leq \rho_{\max,k}$ then          ▷ is $L_{l,k}$ in measurement range?
            if $L_{l,k} \cap \breve{P}_k \neq \emptyset$ then                              ▷ does $L_{l,k} = L^\star_{l,k}$ hold?
                if is_in_FoV$(L_{l,k}, \breve{S}_m)$ then                 ▷ is $L_{l,k}$ observable from sensor $S_m$?
                    $\mathcal{R}^+ \leftarrow \mathcal{R}^+ \cup r_{l,k}$
                end if
            else
                $\mathcal{R}^- \leftarrow \mathcal{R}^- \cup r_{l,k}$
            end if
        end if                                          ▷ else no data point in measurement range
    end if                                                ▷ else no leaf intersected by ray $r_{l,k}$
end for
```

Algorithm 2 gathers all measurement rays that confirm the prior expectation, hence are in support of correct alignment of point clouds P_1 and P_2 via pose Θ_{ICP}, in set $\mathcal{R}^+$ and contradicting rays are collected in set $\mathcal{R}^-$. Once these set are available, they are utilized to serve as a basis for error and alignment measure, respectively. The error measure e_{ray}, based on the results of ray casting on the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ is defined as

$$e_{\text{ray}} = \frac{|\mathcal{R}^-|}{|\mathcal{R}^-| + |\mathcal{R}^+|} \tag{4.27}$$

Its dual, the quality measure c_{ray}, is defined as

$$c_{\text{ray}} = \frac{|\mathcal{R}^+|}{|\mathcal{R}^-| + |\mathcal{R}^+|} \tag{4.28}$$

The same properties that hold for e_{align} and c_{align}, i.e., being bound to interval $[0, 1]$ and $e_{\text{align}} + c_{\text{align}} = 1$ also apply for e_{ray} and c_{ray}. By employing the $\mathcal{R}^+$ and $\mathcal{R}^-$, of measurement rays as the basis for this measures, instead of the sets of octree leaves $\mathcal{L}^+$ and $\mathcal{L}^-$, implicitly adds more weight to those volumes of the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ that feature a higher point density. While each leaf L_i in c_{align} has the same influence, if it contains enough measurement points, i.e., $|L_i| \geq n_{\min}$ holds, the influence of a leaf for measure c_{ray} depends on the number of rays that intersect it. The creation of set R_k guarantees, provided decent modelling of the sensor S_k, that leaves that contain many measurement points from $\breve{P}_k$ are intersected by multiple rays – if these leaves are also the first leaves in measurement range depends solely in the alignment provided by pose Θ_{ICP}.

Now that the basic idea behind the indirect modelling of the Mutual Observability, via comparing the results of measurement ray intersections with prior expectancies, is in place some extension will be discussed in the following.

4.5.1 Adding Weights to Rays

The confidence measures c_{ray} and e_{ray} are both based on the sets of measurement rays $\mathcal{R}^+$ and $\mathcal{R}^-$ that concur or refute correct alignment of point clouds P_1 and P_2. In the basic version, introduced previously, each measurement ray in set $\mathcal{R}^-$ contributes equally to the overall result of c_{ray} and e_{ray}, respectively. I believe however that some of the rays $r_{l,k} \in \mathcal{R}^-$ hint more strongly to wrong alignment than others.

If for a given ray $r_{l,k}$ the closest intersected leaf in measurement range does not contain any points from sensor S_k, i.e., $L^{\rho}_{l,k} \cap \breve{P}_k = \emptyset$, the registration of both point clouds is obviously not perfect, provided that $L_{l,k} = L^{\rho}_{l,k}$ holds. However, there a different levels of quality in terms of "not perfect alignment" for individual measurement rays.

Distinguishing between different grades of wrong alignment allows for a more sophisticated comparison between alignment via several poses. While c_{ray} and e_{ray} inform about the proportion of rays that meet initial expectations in relation to rays that violate the expectations all alignments where these ratios are equal lead to the same assessment. However, if we observe an alignment where all rays that do not meet our initial expectation are caused only by a small registration error, e.g., a very small translation error in the registration, we would usually prefer such an alignment over one where the numbers of $|\mathcal{R}^+|$ and $|\mathcal{R}^-|$ are identical, but all rays $r_{l,k} \in \mathcal{R}^-$ are caused by large errors in alignment. In seldom cases even sensor noise can be responsible for a given ray $r_{l,k}$ being inserted into set $\mathcal{R}^-$: if sensor noise causes sensor S_k to overestimate the distance to some surface, measurement points, that without any noise were located inside $L^{\rho}_{l,k}$, can be placed into the leaf behind $L^{\rho}_{l,k}$, when observed from the position of $\breve{S}_k$. This can lead to $L_{l,k} \neq L^{\rho}_{l,k}$ and thus $r_{l,k}$ would be inserted into set $\mathcal{R}^-$. Provided a reliable mechanism to distinguish for each ray if it indicates only a small registration error or a larger one, it seems prudent to add weights to these rays for the purpose of quantifying the differences in alignment quality that they provide.

In order to identify the differing degree of wrong alignment for a single ray $r_{l,k}$ a new symbol $L^{\star}_{l,k}$ is introduced, that denotes the first leaf, that is intersected by $r_{l,k}$, in measurement range and contains points $\breve{P}_k$:

$$L^{\star}_{l,k} = \arg\min_{i} \mathrm{d}\left(L_i, \check{S}_k\right), L_i \in \mathcal{L}^{\not\varnothing}_{l,k} \; : \; \rho_{\min,k} \leq \mathrm{d}\left(L_i, \check{S}_k\right) \leq \rho_{\max,k} \wedge L_i \cap \check{P}_k \neq \varnothing \tag{4.29}$$

where $\mathcal{L}^{\not\varnothing}_{l,k}$ denotes the set of all leaves $L_i \in \mathcal{L}$, as defined in equation (4.21). As with leaves $L^{\rho}_{l,k}$ and $L_{l,k}$, a leaf $L^{\star}_{l,k}$, that fulfills the requirements of equation (4.29) does not have to exist. Besides, if $L^{\star}_{l,k}$ exists, it does not need to be different from $L^{\rho}_{l,k}$ or $L_{l,k}$. In fact, a ray $r_{l,k}$ meets our expectations, when $L_{l,k} = L^{\rho}_{l,k} = L^{\star}_{l,k}$ hold and is therefore inserted into set $\mathcal{R}^{+}$.

Instead of the scenario, where ray $r_{l,k}$ confirms our expectation, the case where our expectations are not met, demands further investigation. Here those cases are interesting, where all three types of closest leaves $L_{l,k}$, $L^{\rho}_{l,k}$ and $L^{\star}_{l,k}$ exist, but only the first two are equal, i.e., $L^{\rho}_{l,k} \neq L^{\star}_{l,k}$. This means that the first intersected leaf, when observed from location $\check{S}_k$, is in measurement range ($L_{l,k} = L^{\rho}_{l,k}$), does not contain points recorded from sensor S_k. In addition some other leaf $L^{\star}_{l,k}$, that is also in measurement range contains points from $\check{P}_k$ and is, by definition, located behind $L^{\rho}_{l,k}$, i.e., $\mathrm{d}\left(L^{\rho}_{l,k}, \check{S}_k\right) < \mathrm{d}\left(L^{\star}_{l,k}, \check{S}_k\right)$ holds.

If we want to distinguish between cases that might have been caused by either sensor noise or by relatively small alignment errors from cases of obvious misalignment it seems prudent to look at the relation of both leaves $L^{\rho}_{l,k}$ and $L^{\star}_{l,k}$. In case of sensor noise or small alignment errors the distance between $L^{\rho}_{l,k}$ and $L^{\star}_{l,k}$ should be small, for example $L^{\star}_{l,k}$ might be the next octree leaf intersected by $r_{l,k}$, after leaf $L^{\rho}_{l,k}$. The larger the distance between leaves $L^{\star}_{l,k}$ and $L^{\rho}_{l,k}$, the less likely the violation of the a-priori expectation associated with ray $r_{l,k}$ is caused by sensor noise, but the result of a bad alignment of P_1 and $\Theta_{\mathrm{ICP}} \cdot P_2$, at least with respect to this particular measurement ray. This misalignment is even more evident if condition $L^{\rho}_{l,k} \cap \tilde{P}_k = \varnothing$ holds, but no leaf $L^{\star}_{l,k}$ exists.

Taking these considerations into account a desired weight function for a given ray $r_{l,k}$ should have the following properties:

- Have its maximum when $L^{\rho}_{l,k} = L^{\star}_{l,k}$ holds
- Decrease the weight with increasing distance between $L^{\rho}_{l,k}$ and $L^{\star}_{l,k}$
- Handle occlusion, i.e., rays $r_{l,k}$ where $L_{l,k} \neq L^{\rho}_{l,k}$ holds in an appropriate manner

Ideally this weight function should lead to an evaluation / error measure that still retains the properties of e_{align}, c_{align} or e_{ray} and c_{ray}, respectively.

To achieve these properties I introduce a generic weight function, that assigns a $w_{l,k}$ to each ray $r_{l,k}$, which is defined in the following way:

$$w_{l,k} = \begin{cases} 0 & \text{if } \mathcal{L}^{\not\varnothing}_{l,k} = \varnothing \vee L_{k,l} \neq L^{\rho}_{l,k} \\ \delta^{-\mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right)} & \text{otherwise} \end{cases} \tag{4.30}$$

where $\delta \in \mathbb{R}, \delta > 1$ is some constant and $\mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right)$ defines some distance function between the two leaves $L^{\rho}_{l,k}$ and $L^{\star}_{l,k}$, on which will be elaborated a bit more later.

The way that a weight $w_{l,k}$ is defined implies that $w_{l,k} \in [0,1]$ holds, since $\mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right)$ denotes a distance function and hence $\mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) \geq 0$ holds. The maximal weight of 1 is assigned when the distance $\mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) = 0$, which is the case when the first encountered leaf in measurement range also contains points gathered by sensor S_k, i.e., $L^{\rho}_{l,k} = L^{\star}_{l,k}$. Depending on the choice of constant δ the assigned weight either quickly (larger value for δ) or gradually (smaller values) declines. This mechanic enables us to assign smaller, when compared with confirming rays, positive weights to rays $r_{l,k}$ that violate our initial expectation, that can converge quickly to 0 for larger errors, if so desired.

One feature that might stick out, at first, in the definition of the weights $w_{l,k}$ (see 4.30)) is the fact that rays $r_{l,k}$, where the ray either does not intersect any leaf ($\mathcal{L}^{\not\rho}_{l,k} = \emptyset$) or some surface below the minimal measurement range blocked the ray from the scene ($L_{l,k} \neq L^{\rho}_{l,k}$) are assigned the same minimal weight of 0 as rays that clearly indicate wrong alignment. This is done purely for convenience sake, since it allows to define the weighted confidence measure c_{w} as:

$$c_{\mathrm{w}} = \frac{\sum_{k=1}^{2} \sum_{l=0}^{|P_k|-1} w_{l,k}}{|\mathcal{R}^{+}| + |\mathcal{R}^{-}|} \tag{4.31}$$

where the aforementioned rays do not have any influence, since they do not appear in the denominator $|\mathcal{R}^{+}| + |\mathcal{R}^{-}|$. If we would not assign any weight to rays that are either blocked or do not intersect any leaf the term in the numerator would need to become more complicated than just the sum over all available weights.

Since the value of the numerator in equation (4.31) is bound to the interval $[0, |\mathcal{R}^{+}| + |\mathcal{R}^{-}|]$ it follows immediately that $c_{\mathrm{w}} \in [0,1]$ holds. The dual to the confidence measure c_{w}, the error measure e_{w}, can simply be defined as

$$e_{\mathrm{w}} = 1 - c_{\mathrm{w}} \tag{4.32}$$

and thus both measures gain the properties that we desired for comparability with previously introduced measures c_{align}, c_{ray} and e_{align}, e_{ray} respectively.

As mentioned earlier, the choice of the constant δ and the used distance function $\mathrm{d}_{\mathcal{L}}\left(\right)$ have a great influence on the weights $w_{l,k}$ that are assigned to rays $r_{l,k}$ that do contradict correct alignment of point clouds P_1 and P_2 via pose Θ_{ICP}. In fact, if we choose a slightly extreme distance function $\mathrm{d}_{\infty}\left(\right)$ defined as

$$\mathrm{d}_{\infty}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) = \begin{cases} 0 & \text{if } L^{\rho}_{l,k} = L^{\star}_{l,k} \\ \infty & \text{otherwise} \end{cases} \tag{4.33}$$

then c_{w} corresponds to c_{ray}, since all contradicting rays $r_{l,k}$ are assigned with a weight $w_{l,k} = 0$ while all confirming rays are weighted by 1 and the numerator $\sum_{k=1}^{2} \sum_{l=0}^{|P_k|-1} w_{l,k}$ in equation (4.31) becomes $|\mathcal{R}^{+}|$.

While $\mathrm{d}_{\infty}\left(\right)$ formally satisfies the requirements of a distance function, it is only suitable to distinguish between rays that confirm or refute the claim of correct alignment and therefore unsuitable for the desired gradual transition from "correct" via "slightly misaligned" to "clearly misaligned". There are various suitable distance functions that will model the desired properties so that I will just exemplary list some concrete instances for $\mathrm{d}_{\mathcal{L}}\left(\right)$ in the following.

Euclidean Distance Based

This might be the first idea that comes to mind, but it has some drawbacks. The simplest way to define the Euclidean distance between two octree leaves is to employ the distance between their center points, i.e.,

$$\begin{aligned} \mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) &= \mathrm{d}\left(\dot{L}^{\rho}_{l,k}, \dot{L}^{\star}_{l,k}\right) \\ &= \sqrt{\left(x_{\dot{L}^{\rho}_{l,k}} - x_{\dot{L}^{\star}_{l,k}}\right)^2 + \left(y_{\dot{L}^{\rho}_{l,k}} - y_{\dot{L}^{\star}_{l,k}}\right)^2 + \left(z_{\dot{L}^{\rho}_{l,k}} - z_{\dot{L}^{\star}_{l,k}}\right)^2} \end{aligned} \tag{4.34}$$

where $\dot{L}_i$ denotes the center point of octree leaf L_i and $x_{\dot{L}_i}, y_{\dot{L}_i}, z_{\dot{L}_i}$ reference its 3D coordinates.

Since the octree is a regular structure in the sense that every leaf has the same size, using the Euclidean distance in this fashion might be a bit misleading: usually it is associated with a continuous range of distances, but if we measure the distance between the center points of the octree leaves, the distances we can express always contain l_L, the octree's resolution (see subsection 2.3.2), as a factor. This is not necessarily a bad thing, but not confirming the initial expectation one has, when employing the Euclidean distance as a measure.

This effect can be prevented if, instead of using the distance between the centers of $L^{\rho}_{l,k}$ and $L^{\star}_{l,k}$, a distance based on the contents of the leaves is used:

$$\mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) = \underset{i,j}{\arg\min}\ \mathrm{d}\left(\breve{p}_i, \breve{p}_{j,k}\right)\ :\ \breve{p}_i \in L^{\rho}_{l,k} \wedge \breve{p}_{j,k} \in L^{\star}_{l,k} \cap \breve{P}_k \tag{4.35}$$

In equation (4.35) we search for the smallest Euclidean distance between a point $\breve{p}_i \in \breve{P}_k \cup \breve{P}_m$ that is contained in leaf $L^{\rho}_{l,k}$ and a point recorded by sensor S_k that lies inside of $L^{\star}_{l,k}$, with $k, m \in \{1, 2\}, k \neq m$. Through the choice of $\breve{p}_i$ it is ensured that in case of $L^{\rho}_{l,k} = L^{\star}_{l,k}$ the distance function returns 0. In the cases where $L^{\rho}_{l,k} \neq L^{\star}_{l,k}$ holds, we know that $L^{\rho}_{k,l} \cap \breve{P}_k = \emptyset$ holds, so that we do not explicitly need to restrict the choice of point $\breve{p}_i$ in equation (4.35). However, determining the distance between two leaves in this fashion can become computationally expensive, depending on the volume of the leaves and the number of points they contain.

Furthermore, if the Euclidean distance is used in weight function (4.30) the employed measurement units and the scale of the acquired point clouds P_1 and P_2 need to be taken into account. Weights will differ vastly, if the underlying measurement unit are for instance meters or millimeters, potentially causing either too large or too small weights, if not considered in advance.

A way to eliminate the influence of the underlying measurement unit is to divide the obtained Euclidean distance (either via equation (4.34) or (4.35)) by l_L, the length of each edge of $L_i \in \mathcal{L}$, thus shifting the measurement unit of for the distance to the side length of an octree leaf l_L. Thus equations (4.34) and (4.35) become

$$\mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) = \frac{\mathrm{d}\left(\dot{L}^{\rho}_{l,k}, \dot{L}^{\star}_{l,k}\right)}{l_L} \tag{4.36}$$

and

$$\mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) = \frac{\arg\min_{i,j}\ \mathrm{d}\left(\breve{p}_i, \breve{p}_{j,k}\right)}{l_L}\ :\ \breve{p}_i \in L^{\rho}_{l,k} \wedge \breve{p}_{j,k} \in L^{\star}_{l,k} \cap \breve{P}_k \tag{4.37}$$

respectively.

This effectively shifts the measurement unit for the distance function from the underlying measurement unit to considering the distance in terms of the number of edges that fit between two octree leaves. Naturally this is only an improvement, if the edge length l_L is provided in a sensible relation to the underlying measurement unit, but this is a necessary requirement for a sensible pose evaluation in the first place.

Discrete Distance Functions

In the closure of 4.5.1 it was suggested to divide the Euclidean distance between $L^{\rho}_{l,k}$ and $L^{\star}_{l,k}$ by l_L, the length of an edge of an octree leaf $L_i \in \mathcal{L}$, thus effectively measuring the distance in terms of leaves or their side length, respectively. If a (theoretically) continuous distance function is not required, two computationally simpler distance functions that are based on the neighborhood relations inside the octree structure can be employed. This however also implies that the weights, computed according to equation (4.30) will also lose its continuous property.

As mentioned in subsection 2.3.2, each octree leaf $L_i \in \mathcal{L}$ is associated with an internal grid coordinate $v^{\mathrm{g}}_{L_i} = \left(x^{\mathrm{g}}_{L_i}, y^{\mathrm{g}}_{L_i}, z^{\mathrm{g}}_{L_i}\right)^T$, where $x^{\mathrm{g}}_{L_i}, y^{\mathrm{g}}_{L_i}, z^{\mathrm{g}}_{L_i} \in \mathbb{Z}$, that denote the integer coordinates in the octree's 3D grid structure that correspond to this particular leaf L_i. Even though an octree is, as opposed to a voxel grid, a sparse structure, the employed indices correspond to the indices of the dense voxel grid structure. This not only allows for latter addition of points in the octree without reorganizing the existing indices, but also enables us to measure how many cells are located between two given leaves L_i and L_j, even if no measurement points and therefor no other actual leaves are placed between L_i and L_j.

Therefore, these octree coordinates allow for some simple distance functions that fulfill the requirements we ask from distance function $\mathrm{d}_{\mathcal{L}}\left(\right)$.

The first distance function is the well-known *Manhattan Distance* which we will define for the octree leafs as

$$\mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) = \mathrm{d}_M\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) = \left|x^{\mathrm{g}}_{L^{\rho}_{l,k}} - x^{\mathrm{g}}_{L^{\star}_{l,k}}\right| + \left|y^{\mathrm{g}}_{L^{\rho}_{l,k}} - y^{\mathrm{g}}_{L^{\star}_{l,k}}\right| + \left|z^{\mathrm{g}}_{L^{\rho}_{l,k}} - z^{\mathrm{g}}_{L^{\star}_{l,k}}\right| \tag{4.38}$$

so that $\mathrm{d}_M\left(\right)$ corresponds to the sum of coordinate differences of the octree coordinates $v^{\mathrm{g}}_{L^{\rho}_{l,k}}$ and $v^{\mathrm{g}}_{L^{\star}_{l,k}}$ of the two intersected leaves $L^{\rho}_{l,k}$ and $L^{\star}_{l,k}$.

Similarly distance function $\mathrm{d}_{\max}\left(\right)$ is also based on octree coordinates, but considers only the maximal difference over all dimensions, i.e.

$$\mathrm{d}_{\mathcal{L}}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) = \mathrm{d}_{\max}\left(L^{\rho}_{l,k}, L^{\star}_{l,k}\right) = \max\left(\left|x^{\mathrm{g}}_{L^{\rho}_{l,k}} - x^{\mathrm{g}}_{L^{\star}_{l,k}}\right|, \left|y^{\mathrm{g}}_{L^{\rho}_{l,k}} - y^{\mathrm{g}}_{L^{\star}_{l,k}}\right|, \left|z^{\mathrm{g}}_{L^{\rho}_{l,k}} - z^{\mathrm{g}}_{L^{\star}_{l,k}}\right|\right) \tag{4.39}$$

Which of $\mathrm{d}_M\left(\right)$ or $\mathrm{d}_{\max}\left(\right)$ is preferable depends on what one considers as the *direct neighborhood* of an octree leaf L_i. The term direct neighborhood for a given leaf L_i here corresponds to all those leaves L_j where $\mathrm{d}_{\mathcal{L}}\left(L_i, L_j\right) = 1$ holds. The Manhattan Distance $\mathrm{d}_M\left(\right)$ only views up to 6 leaves as direct neighbors of a given leaf L_i, namely those with which the cubic volume of L_i shares a side, i.e., the ones above, below, left, right, before and behind, if those leaves exist. In the maximal difference function $\mathrm{d}_{\max}\left(\right)$ all those leaves that share one of the 8 vertices of L_i, instead of a side, are considered neighbors, increasing the potential size of the neighborhood from 6 to 26 (9 neighbors below, 9 neighbors above and 8 neighbors on the same level).

Apart from the Euclidean based distance functions and $\mathrm{d}_M\left(\right)$ or $\mathrm{d}_{\max}\left(\right)$ other suitable functions exist that can be employed as $\mathrm{d}_{\mathcal{L}}\left(\right)$ in equation (4.30) in order to provide decreasing weights in connection with constant δ, as long as $\mathrm{d}_{\mathcal{L}}\left(\right)$ complies with the conditions to generic distance functions, i.e.,

1. Non-negativity

$$\mathrm{d}_{\mathcal{L}}\left(x, y\right) \geq 0 \tag{4.40}$$

2. Identity of indiscernibles

$$\mathrm{d}_{\mathcal{L}}\left(x, y\right) = 0 \Leftrightarrow x = y \tag{4.41}$$

3. Symmetry

$$\mathrm{d}_{\mathcal{L}}\left(x, y\right) = \mathrm{d}_{\mathcal{L}}\left(y, x\right) \tag{4.42}$$

4. Triangle inequality

$$\mathrm{d}_{\mathcal{L}}\left(x, z\right) \leq \mathrm{d}_{\mathcal{L}}\left(x, y\right) + \mathrm{d}_{\mathcal{L}}\left(y, z\right) \tag{4.43}$$

Especially equation (4.41) is important, since it ensures, in combination with (4.30), that rays $r_{l,k}$ that imply correct alignment, i.e., $L^{\rho}_{l,k} = L^{\star}_{l,k}$, are assigned a weight $w_{l,k} = 1$.

It is noteworthy to mention that, by design, we do not necessarily determine the distance from $L^{\rho}_{l,k}$ to the closest leaf that contains points from $\breve{P}_k$, but the distance the closest leaf that contains points from $\breve{P}_k$ and was intersected by measurement ray $r_{l,k}$, i.e., $L^{\star}_{l,k}$. If $L^{\star}_{l,k}$, provided it exists, is not in the direct neighborhood of $L^{\rho}_{l,k}$, this neighborhood may also contain a leaf L_i with $L_i \cap \breve{P}_k \neq \emptyset$. However, such a leaf does not comply with the assumed sensor model and is not considered when determining weight $w_{l,k}$ for measurement ray $r_{l,k}$, because it does not provide information about the alignment of P_1 and P_2 with regard to $r_{l,k}$.

Punishing Connected Regions of Misalignment

If a valid distance function $\mathrm{d}_{\mathcal{L}}\left(\right)$, that complies with the requirements from equations (4.40) – (4.43), is employed in equation (4.30), then the weighted confidence measure will be larger or equal to the basic measure, based on ray tracing, i.e., inequality $c_{\mathrm{ray}} \leq c_{\mathrm{w}}$ always holds true.

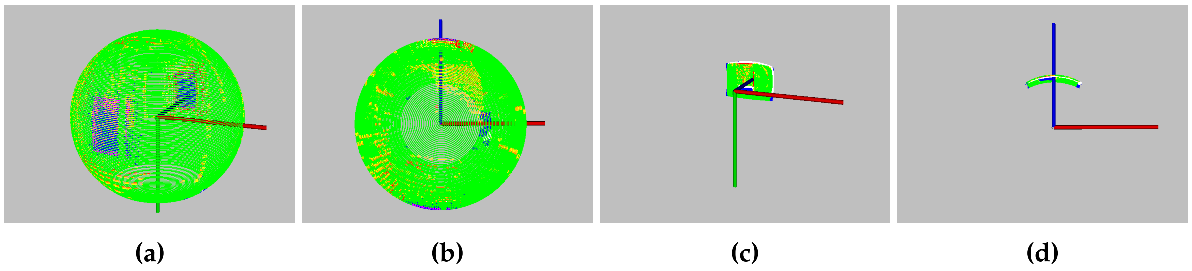


Figure 4.10: 3D visualization of measurement rays. (a) and (b) show side and top view of a rotating scanner with a horizontal opening angle of 360 °. Each depicted point corresponds to the start point of a measurement ray, while the sensor is placed at the origin. Note, that each 'ring' in the resulting sphere corresponds to a row in the 2D projection (see Figures 4.11 – 4.13). In (c) and (d) the starting points of all measurement rays for a Kinect sensor are displayed. For comparison same viewpoints and scale as in (a) and (b) is used. Information concerning the point colors is provided by Table 4.6.

This is obvious, since c_{ray} and c_{w} treat rays in set $\mathcal{R}^+$, that point towards correct alignment, equally, but in c_{w} rays $r_{l,k} \in \mathcal{R}^-$ may be assigned with weights $w_{l,k} > 0$, whereas they effectively are weighted with 0 in case of c_{ray}.

When I introduced weights I argued, that these actually make sense, since there are different grades of misalignment. The idea behind the weights were to distinguish rays in set $\mathcal{R}^-$ that might have been caused by sensor noise or small misalignments from larger alignment errors. However, the relation between c_{ray} and c_{w} points to a caveat, that comes with assigning weights $w_{l,k} \geq 0$ to rays $r_{l,k} \in \mathcal{R}^-$. Depending on the choice of weight function $\mathrm{d}_{\mathcal{L}}\left(\right)$ and constant δ in equation (4.30), the weighted version can have an effect that is rather not desirable: an alignment with a majority of rays pointing towards correct alignment and only few outlier rays with weights of 0 (or very close to 0) might in theory get a worse value for c_{w} than an alignment where all rays are just a little of, i.e., $\mathcal{R}^+ = \emptyset$, but are assigned weights $0 \ll w_{l,k} < 1$.

To further distinguish single outliers in the alignment and the occasional misalignment, due to sensor noise, from misalignments that take place over a larger part of the scene an extension can be made to confidence measure c_{w} and associated error measure e_{w}. A prerequisite for such a modification is a detection of connected regions in the sets of measurement rays R_1 and R_2 that are contradicting correct alignment of P_1 and P_2. This becomes possible if for a given ray $r_{l,k}$ a distinct neighborhood $R_{r_{l,k}} \subset R_k$ can be determined. For example a grid-like arrangement, as discussed at the beginning of this section in the context of the creation of sets R_1 and R_2, allows to apply neighborhoods definitions like the von Neumann neighborhood or Moore neighborhood to the measurement rays of sensor S_k.

To give an impression how the measurement rays for common scanners might appear in an organized fashion, please refer to Figure 4.10. In these visualization each measurement ray is represented by a single 3D point, namely the starting point of the measurement ray. The sensor position in Figure 4.10 coincides with the coordinate systems origin and the distance of each point is dependent on parameter $\rho_{\min,k}$ for the specific sensor. Besides the number of measurement rays and their spacing, also the geometrical shape of the visualized 3D points is dependent on the sensor parameters, since they reflect the field of view. Although the appearance of the starting points in Figures 4.10a and 4.10b differ from those shown in Figures 4.10c and 4.10d, both sets of rays provide an underlying grid structure that can be used to define a desired neighborhood. A 2D depiction of the data provided in Figures 4.10a and 4.10b is provided later in Figure 4.11.

In Figure 4.10 color information is used to convey the evaluation result for a given measurement ray $r_{l,k}$. A brief overview of the meaning of each color is provided by Table 4.6 and followed by a more detailed explanation next: while rays $r_{l,k}$ are grouped into either set $\mathcal{R}^+$, $\mathcal{R}^-$ or not included into any special set (see Algorithm 2) in order to establish measures e_{ray}, e_{w} and c_{ray}, c_{w}, respectively, visual feedback beyond these categories might be helpful, when examining the results of the alignment analysis. Therefore, Table 4.6 also distinguishes those rays that are neither ordered into sets $\mathcal{R}^+$ nor $\mathcal{R}^-$ into subcategories and proceeds in the same fashion with rays that contradict correct alignment of point clouds P_1 and P_2.

Note that pixel and point in the following can and are used somewhat interchangeably: a natural way to directly visualize the 2D neighborhood grid structure is as an image, where each measurement ray $r_{l,k}$ in the grid structure corresponds to a single pixel. On the other hand, the measurement rays and their arrangement can also be represented by 3D points, that are located somewhere on the ray, as is done in Figure 4.10. While it is easier in the flat representation to identify connected regions of misalignment, the 3D visualization can be helpful, to better identify in which regions of the combined point cloud alignment errors occurred, since it retains the correct perspective properties and, if stored as a point cloud, can be visualized alongside combined point cloud.

Points that are associated with rays $r_{l,k} \in \mathcal{R}^+$ that suggest correct alignment are colored in green. A perfectly aligned scene without any sensor noise should only feature three types of pixels, namely green pixels where the measurement rays met their a-priori expectation along blue and cyan pixels, which are described next.

Blue color indicates that a measurement ray did not intersect any octree leaf and according to Algorithm 2 the leaf is contributing neither to $\mathcal{R}^+$ nor $\mathcal{R}^-$. Also colored in this fashion are rays $r_{l,k}$, which are emitted by sensor S_k and first intersect a leaf that only contain points from $\check{P}_m$ ($k, m \in \{1, 2\}$ and $k \neq m$) and is outside of the measurement range of S_k. Formalized this boils down to $\forall L_i \in \mathcal{L}^{\not\emptyset}_{l,k} \ : \ L_i \cap \check{P}_k = \emptyset$ (no leaf intersected that contains points recorded by sensor S_k) in conjunction with $\mathrm{d}\left(L_{l,k}, \check{S}_k\right) < \rho_{\min,k} \vee \rho_{\max,k} < \mathrm{d}\left(L_{l,k}, \check{S}_k\right)$ (first intersected leaf is not in measurement range of S_k).

Pixels in cyan show that ray $r_{l,k}$ confirmed the initial expectation, i.e., $L^{\rho}_{l,k} = L^{\star}_{l,k}$, but the location of leaf $L^{\star}_{l,k}$ is outside the field of view of sensor S_m. Compactly this can be written as $L^{\star}_{l,k} \cap ROO_{1,2} = \emptyset$.

Magenta is used to visualize rays where the intersected leaf was beyond the specified sensor's maximal range. More specifically this means that measurement ray $r_{l,k}$ intersected some leaves ($\mathcal{L}^{\not\emptyset}_{l,k} \neq \emptyset$) and there also exists at least one intersected leaf $L_i \in \mathcal{L}^{\not\emptyset}_{l,k}$ with $L_i \cap \check{P}_k \neq \emptyset$, i.e., an octree leaf that contains measurement points recorded by the sensor S_k that emitted the current ray. However, no leaf $L^{\star}_{l,k}$ (see equation (4.29)) exists, since either the intersected leaf is too close to the sensor ($\mathrm{d}\left(L_i, \check{S}_k\right) < \rho_{\min,k}$) or too far away from the sensor ($\rho_{\max,k} < \mathrm{d}\left(L_i, \check{S}_k\right)$) for all leaves $L_i \in \mathcal{L}^{\not\emptyset}_{l,k} \ : \ L_i \cap \check{P}_k \neq \emptyset$. Single instances of magenta pixels might be caused either by sensor noise or by imperfect filtering of the initial point cloud. If however a large number of measurement rays are colored in magenta this might point towards an inaccurate sensor model

Category	Color	Explanation
Supporting (inserted into $\mathcal{R}^+$)	green	Rays $r_{l,k}$ confirming measurement expectation, i.e., $L^{\rho}_{l,k} = L^{\star}_{l,k}$
	blue	Non-intersecting ray, i.e., $\mathcal{L}^{\not\rho}_{l,k} = \emptyset$
Neutral (neither part of $\mathcal{R}^+$ or $\mathcal{R}^-$)	magenta	Out of range, i.e., first intersected leaf is beyond measurement range (d $\left(L_{l,k}, \tilde{S}_k\right) > \rho_{\max,k}$)
	cyan	Confirming, but not in field of view of other sensor, i.e., $L^{\rho}_{l,k} = L^{\star}_{l,k} \wedge L^{\star}_{l,k} \cap ROO_{1,2} = \emptyset$
Contradicting (inserted into $\mathcal{R}^-$)	red	Ray $r_{l,k}$ indicating misalignment, i.e., $L^{\rho}_{l,k} \neq L^{\star}_{l,k}$, but $L^{\star}_{l,k}$ exists. Decreasing weight $w_{l,k}$ moves color from green towards red in HSV color space.
	white	Unexplained intersection, i.e., $L^{\rho}_{l,k} \cap \check{P}_k = \emptyset$ exists, but $L^{\star}_{l,k}$ does not and $\forall L_i \in \mathcal{L}^{\not\rho}_{l,k} : L_i \cap \check{P}_k = \emptyset$ holds.

Table 4.6: Overview of color coding for measurement ray visualization (for examples please refer to Figures 4.10 and 4.11 – 4.13).

or a not sufficiently calibrated sensor. According to Algorithm 2 rays that are colored in this fashion also do not contribute to the evaluation measures. Incidentally the same holds true for the direct modelling approach, where points that are not in the specified measurement range are not part of the region of overlap $ROO_{1,2}$ and thus are not considered in the computation of e_{align} / c_{align}. In this way decreasing the maximal range $\rho_{\max,k}$ of sensor S_k can be a way to deal with increasingly sparse measurements at larger distances and increasing measurement noise, but it should be noted that a pre-filtering of point clouds P_1 and P_2 is a cleaner solution to address this problem.

Having discussed the supporting and neutral rays listed in Table 4.6 a precise description of contradicting rays is given, i.e., those rays that are bundled into set $\mathcal{R}^-$. In the non-weighted version of the ray tracing based evaluation the vast majority of these rays is colored in red. In the weighted versions (e_{w} / c_{w}) the color of these pixels is dependent on the assigned weight $w_{l,k}$ for the corresponding ray $r_{l,k}$: while rays $r_{l,k}$ with a weight $w_{l,k} = 1.0$ are colored in green and rays where $w_{l,k} = 0.0$ are colored in red, weights between these two extremes are colored as a transition between green and red. Since the weights $w_{l,k}$ are located inside interval $[0, 1]$ the HSV color space can be employed to provide such a gradual transition between green ($H = 120\,^{\circ}$ in HSV color space) and red ($H = 0\,^{\circ}$). Therefore, all yellowish and orange colors also indicate misalignment, albeit not as bad as red colors. Note that the actual colors are of course dependent on the choice of constant δ and the distance function $\mathrm{d}_{\mathcal{L}}\left(\right)$ in equation (4.30).

Finally some rays $r_{l,k}$ may be colored in white in the visualization. These constitute a subcategory of the rays in $\mathcal{R}^{-}$, that indicate misalignment. As opposed to the earlier discussed rays that are colored in red (or in the weighted variant in a color between green and red in HSV color space), for those rays that are colored in white no leaf $L^{\star}_{l,k}$ exists. While the measurement ray $r_{l,k}$ intersects at least one leaf $L^{\rho}_{l,k}$ in sensor range, no intersected leaf contains points recorded by sensor S_k, i.e., $\forall L_i \in \mathcal{L}^{\not\rho}_{l,k} \ : \ L_i \cap \breve{P}_k = \emptyset$. While they are not treated differently from other rays in $\mathcal{R}^{-}$ and are assigned a weight $w_{l,k} = 0$, they are visually highlighted from their red counterparts, since sensor S_m $(k, m \in \{1, 2\}, k \neq m)$ encountered some obstacle that should have also been measured by sensor S_k, under the premise of correct alignment, but S_k did not collect any measurement for this particular ray.

Since both sensor types visualized in Figure 4.10 feature a grid-like arrangement of their measurement rays, they can directly be represented as images, resulting in those displayed in Figures 4.11 – 4.13 for the rotating scanner or Figure 4.19 for the Kinect sensor, respectively. The neighborhood relations of measurement rays come down to the neighborhood of the corresponding pixels in the visualization. In case of sensors with a 360 ° horizontal opening angle ω_{h}, one needs to keep in mind that the left and the right border of the image are also connected. With these considerations neighborhoods like the von Neumann neighborhood or the Moore neighborhood can be applied directly on the image (or rather a 2D array of weights $w_{l,k}$, that encodes the information shown in the image) and then be translated on the associated rays. If, instead of the grid representation of the measurement rays, 3D points, like the start point of each ray $r_{l,k}$, are considered, the Euclidean distance between neighboring points in this case will generally vary for a given sensor. For example the points in the top row of Figure 4.11a are located near the "pole" of the partial sphere around the sensor (see also Figure 4.10b) and therefore closer to each other in terms of Euclidean distance than pixels that represent measurement rays near the "equator" of the sphere.

Visualizations of the measurement rays always come in pairs, since we need to take into account the measurement rays emitted from both sensor S_1 and S_2. When observing Figures 4.11 – 4.13 the top row will always belong to sensor S_1, while the bottom row displays the alignment result from the point of view of S_2. The three Figures show the visual representation of the ray tracing based evaluation approach for the three different alignments of the hallway scene (see Figure 4.4). While the evaluation based on the regular ICP output parameters (Tables 4.2, 4.3) suggest that the wrong alignments are preferable the visual representation of the three different alignments already suggests a different outcome for the proposed evaluation measure.

However, before talking about the differences between the correct alignment in Figure 4.11 and the erroneous alignments in Figure 4.12 (translation error) and Figure 4.13 (rotation error), let us first consider the common features between all three visualizations: all visualizations contain a blue, roughly rectangular shaped region in their middle, their left border and a, somewhat smaller one on the right side. These regions correspond to measurements along the corridor's main axis, where the sensor did not record any obstacle.

In all three Figures only few magenta pixels are present, so few that they are hard to discern without prior knowledge about their location. Both Figures 4.12b and 4.13b feature some

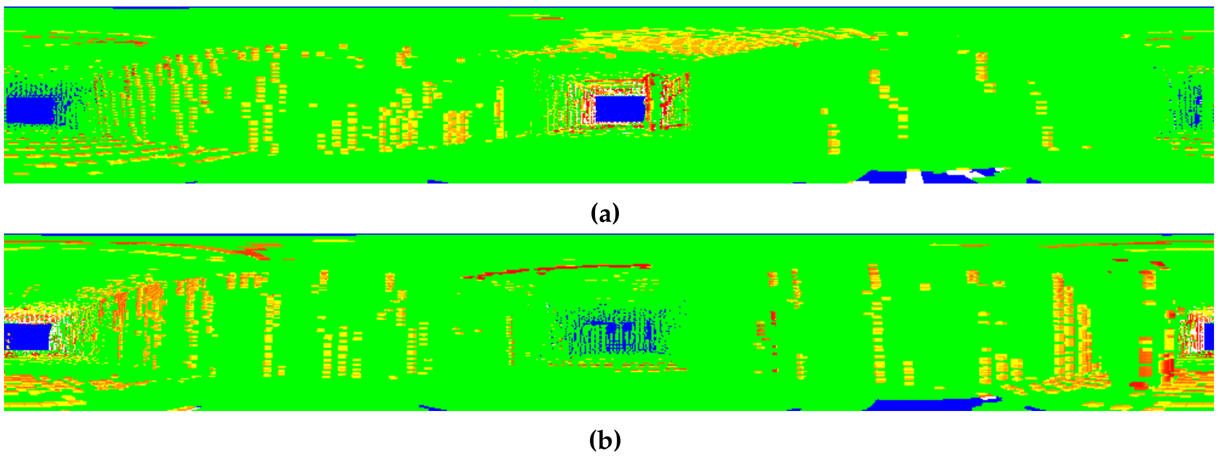


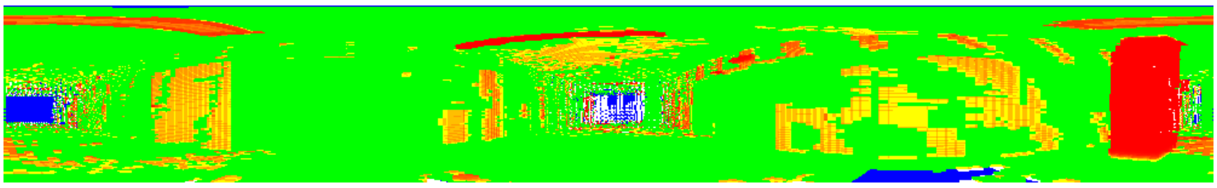
Figure 4.11: Visualization of ray evaluation for correct alignment of hallway scene (see Figure 4.4a, 4.4b). (a) shows the rays originating from sensor S_1 while (b) displays those of sensor S_2. Note that this visualization corresponds to a scan with a horizontal opening angle of $\omega_\mathrm{h} = 360\,°$, so that the left and right edges are actually connected. Furthermore, the points in the top row of (a) almost coincide (and the same holds true for (b), of course). For more details refer to the 3D visualization in Figure 4.10.

magenta pixels inside the blue region (the corridors main axis) on their left. Ideally this type of rays should not occur. In low quantities, as in the displayed Figures, they indicate, that the sensor is not modeled completely accurate, in the example case, some recorded measurements exceeded the set maximal measurement distance $\rho_{\max,k}$. If they appear in larger quantities, the range parameters of the sensor are not valid and need to be refined.

The blue (Figures 4.11, 4.12), respectively white regions (Figure 4.13), slightly to the right of the center and near the lower edge are caused by removing measurement points located on the robot platform from the point clouds. This "self-filtering" is not incorporated into the simple sensor model with its small number of parameters, thus causing a discrepancy between expected measurements and available measurements after filtering. Employing a more complex sensor model that can incorporate such filters could be used to solve this minor issue.

All three alignment visualizations contain a large portion of green – this reflects that large portions of the data in both scans P_1 and P_2 are very similar and these portions are aligned well with each other, whether the correct alignment (Figure 4.11) or the erroneous alignments (Figures 4.12, 4.13) are concerned. The same effect is also evident, when investigating the corresponding point clouds in Figure 4.4. But when considering areas of misalignment marked in the spectrum from yellow to red there are substantial differences between the correct alignment and the misaligned scenes: in the correctly aligned scene the size of connected regions of rays in set $\mathcal{R}^-$ is noticeably smaller than in the wrongly aligned results and mostly colored ranging from yellow to orange, i.e., assigned some positive weight $0 < w_{l,k} < 1$. Some reasons for contradicting information in the correctly aligned point clouds were already discussed, earlier and will not be reiterated here. Furthermore, the alignment obtained from initial pose $\Theta_{\text{good\_init}}$ is not necessarily the best alignment one can achieve for these two point clouds and could probably be more refined when experimenting with different settings for ICP.

The misaligned scenes in Figures 4.12 and 4.13 contain larger connected regions that mainly


(a)

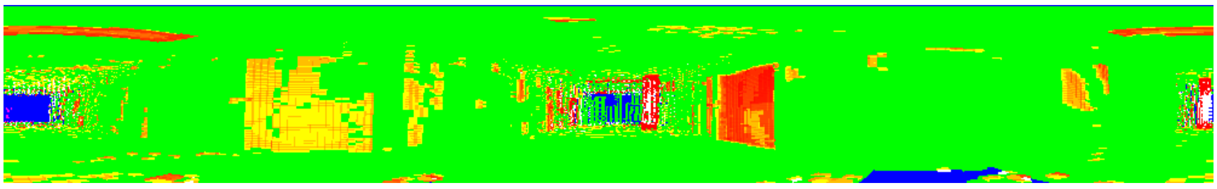

(b)

Figure 4.12: Visualization of ray evaluation for alignment with a translational error of hallway scene (see Figure 4.4c, 4.4d). Top row (a) depicts rays $r_{l,1}$ emitted from sensor S_1 while the bottom row (b) corresponds to S_2.

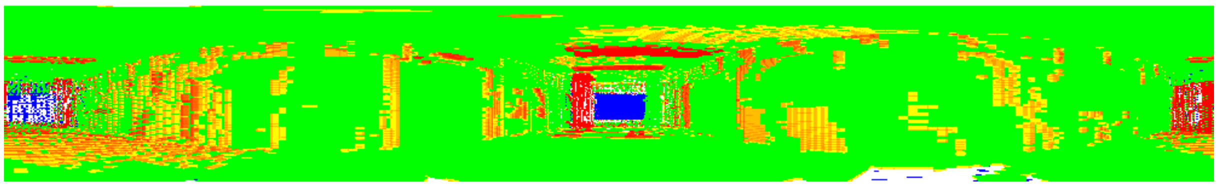
consist of bright red pixels ($w_{l,k} \approx 0$). While pixels in light orange might sometimes be caused by sensor noise a larger connected region of red cannot be explained in such a way. The large red area on the right side of Figure 4.12a corresponds to a pillar formed from points $\breve{p}_{j,2} \in \breve{P}_2$ that do not have any resemblance in P_1 due to the translational error. If we compare the visualized evaluation result with the registered scene, depicted in Figure 4.4d, said area corresponds to the pillar, encircled in red, slightly right of the origin (remember that P_1 is depicted in cyan and P_2 in magenta (see also Figure 4.3)). The white points near the center of Figure 4.12a mark the opened door recorded in P_2 that is not contained in P_1 and therefore an additional indicator that P_1 and P_2 are not registered correctly.

When investigating the rays from the other sensor's position (Figure 4.12b), misalignments are perceived in other regions of the combined scan. The open door, marked in white in Figure 4.12a, appears here as a collection of green measurement rays, since they confirmed the a-priori expectation, i.e., no measurement from P_1 blocked any of the rays. However, other structures from P_1, like a pillar near the center and the emergency exit door do not comply with correct alginment and are colored accordingly.

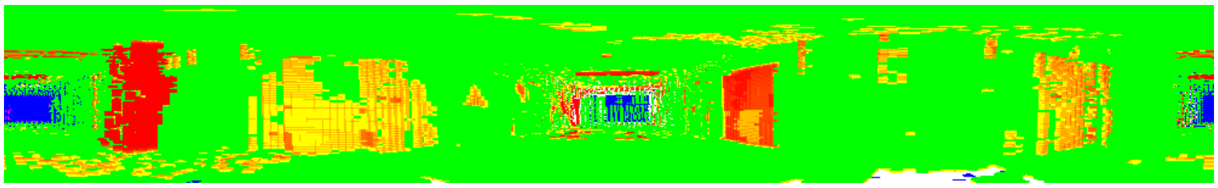
The visualization for the registration with the rotational error (Figure 4.13) also contains larger and more distinct red areas, when compared with the correct registration, albeit for different parts of the environment that do not match and therefore at different locations in the visualization image as in the case of the translation error.

As discussed before and illustrated by the example provided by Figures 4.11 – 4.13 larger connected regions of rays that contradict a-priori expectations seem to be a good indicator to distinguish between multiple isolated misalignments that can be caused by sensor noise, like in Figure 4.11 and rays $r_{l,k} \in \mathcal{R}^-$ that are caused from actual scene misalignment (Figures 4.12, 4.13). Therefore, a short description how to detect these continuous regions of misalignment follows next.

Provided some neighborhood $R_{r_{l,k}}$, for instance the von Neumann neighborhood, for a



(a)



(b)

Figure 4.13: Visualization of ray evaluation for alignment with a rotation error of hallway scene (see Figure 4.4e, 4.4f). Top (a) shows the scene from point of view of sensor S_1 while (b) displays S_2's point of view.

given ray $r_{l,k} \in \mathcal{R}^-$ it becomes easy to start a region growing approach, where neighbors that also point towards wrong alignment, i.e., $r_{m,k} \in R_{r_{l,k}} \cap \mathcal{R}^-$ and their subsequent neighbors are gathered into a new set. Let us denote such set as $R^{\text{c-}}_{p,k}$. Such a set $R^{\text{c-}}_{p,k}$ of connected rays, will always be either a subset of $\mathcal{R}^- \cap R_1$ or $\mathcal{R}^- \cap R_2$, since rays that did not originate from the same sensor cannot be in the same continuous region. The resulting sets, $R^{\text{c-}}_{p,k} \subseteq \mathcal{R}^- \cap R_k$, are independent of the ray $r_{l,k}$ that was used to initialize the region, i.e., using any ray $r_{m,k} \in R^{\text{c-}}_{p,k}$ as a seed for the region growing process results again in the set $R^{\text{c-}}_{p,k}$. A detailed description how to create the continuous regions is shown in Algorithm 3.

In the algorithm, function get_neighborhood$(r_{l,k}, \text{expanded_rays})$ returns either the von Neumann neighborhood, the Moore neighborhood or some other suitable neighborhood for ray $r_{l,k}$, where expanded_rays contains the set of all rays that have been expanded, so far. The second argument in get_neighborhood$(r_{l,k}, \text{expanded_rays})$ ensures that only rays $r_{m,k}$, that have not been expanded yet, i.e., $r_{m,k} \notin$ expanded_rays, are returned. The parameter $w_{\min}$ can be used to pose further restrictions on the rays in $\mathcal{R}^-$ that are included into the connected regions: employing $w_{\min} = 1$ ensures that every ray $r_{l,k} \in \mathcal{R}^-$ will be part of some connected region $R^{\text{c-}}_{l,k}$. If $w_{\min}$ is set to a value smaller than 1, only rays with a corresponding weight will be allowed to initialize or be part of a region $R^{\text{c-}}_{l,k}$. In this way it is possible to only create regions from those rays, where the misalignment cannot be explained by sensor noise.

Once all sets $R^{\text{c-}}_{l,k}$ are established according to Algorithm 3 it is sensible to investigate the size of the connected regions of rays that contradict with correct alignment. While the algorithm guarantees that all rays in a set $R^{\text{c-}}_{l,k}$ are connected in terms of the employed neighborhood, the size of these regions is not considered in the algorithm, in fact, a connected region may consist of only a single ray $r_{l,k} \in \mathcal{R}^-$, if its weight is below threshold $w_{\min}$. Therefore, a post-processing step, that only leaves those regions which contain more measurement rays than a given threshold $|R^{\text{c-}}|_{\min}$, produces the outcome that we would like to gain, i.e., a set of all connected regions with at least a certain size of measurement rays that point towards misalignment. If a region

$R^{\text{c-}}_{l,k}$ passes this filter step, meaning that inequality $|R^{\text{c-}}|_{\min} \leq \left|R^{\text{c-}}_{l,k}\right|$ holds, this is a strong indication for at least local misalignment. With respect to the weights this is expressed by setting all weights $w_{m,k} = -1$ for all rays $r_{m,k} \in R^{\text{c-}}_{l,k}$. Instead of assigning a fixed weight of -1 to all rays $r_{m,k} \in R^{\text{c-}}_{l,k}$ in a given region the existing weights could all be modified by subtracting a fixed value from all of them. This way one could still distinguish between different grades of bad local alignment, but I believe that is a matter of personal preference, since all rays $r_{l,k}$ in a region of local misalignment share a common characteristic, namely $w_{l,k} < w_{\min}$. Hence, they might as well be treated equally with respect to their weights.

An exemplary result of the detection and the post-processing can be observed in Figure 4.14, where results for the alignments displayed in Figures 4.11 – 4.13 are shown. While the color scheme is in general identical to the previous Figures, those pixels that correspond to rays that form a connected region of contradicting measurements are now colored in black. For the results in Figure 4.14 the von Neumann neighborhood was employed in connection with a small threshold ($w_{\min} = 0.1$), and the octree, used to discretize the combined point cloud to enable efficient ray tracing, features leaves with side length $l_L = 5\,\text{cm}$. Connected regions had to encompass at least 100 rays (the total number of rays being 1000×146 per sensor), the function described in equation (4.36) served as a distance function and parameter δ in weight function (4.30) was set to 2. Of course, changing one or more of these parameters may generate different results from those, shown in Figure 4.14.

When observing the results for the correctly aligned scene in the top row of Figure 4.14 it becomes evident that black pixels mainly occur in two regions: first, an overhead light (top center of Figure 4.14b) and, less evident, some smaller regions near the corridors main axis. The overhead lights that were scanned in the scene contain mirror elements to reflect some of the emitted light towards the floor. Mirrors are known to cause problems for the range estimation of laser scanners, so data inconsistencies in this part of the combined point cloud are not surprising and do not point towards a general misalignment. The other connected regions are less easy to explain. One is at least partially caused by some of the aforementioned sensor noise that was caused by the edge of one of the pillars (center of Figure 4.14a), while others may actually hint that the current alignment might still be improved upon.

In contrast to the alignment, obtained when starting from initial pose $\Theta_{\text{good\_init}}$, the visualization of the alignment obtained for initial pose $\Theta_{\text{trans\_error}}$ (Figure 4.14c and 4.14d) and the pose with the rotation error $\Theta_{\text{rot\_error}}$ (Figure 4.14e and 4.14f) feature larger and much more prominent connected areas of misalignment. The data inconsistencies at the locations of the pillars can be clearly seen. In addition areas exist, where the detected regions could have become much larger, if a different threshold $w_{\min}$ would have been employed. One obvious example is on located right to the center in Figure 4.14d, where only some rays on the misaligned emergency exit comply with the chosen threshold parameter.

Apart from the difference in size of the detected regions shown in Figure 4.14, the total number of rays $r_{l,k}$ that are below the specified threshold of $w_{\min} = 0.1$ also differs in the three alignments. In the correctly registered scene for sensor S_1 5 connected regions $R^{\text{c-}}_{l,1}$ with a total of 852 rays were detected and for sensor S_2 the detected 3 regions entailed 571 rays. For the registration with the translation error 6994 rays in 4 regions for sensor S_1 and 2222 rays also organized in 4 regions were detected. In the alignment with the rotation error 5999 rays in 7

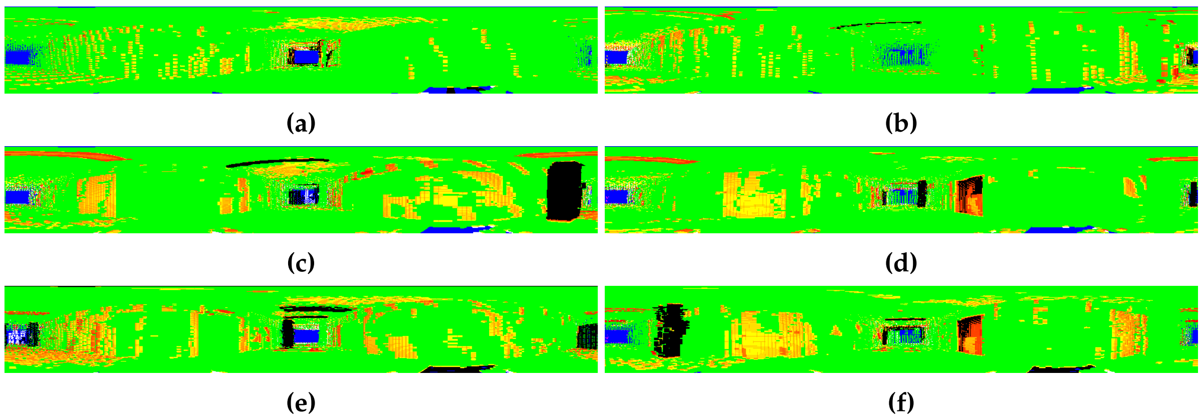


Figure 4.14: Exemplary results of continuous region detection, described in Algorithm 3. Connected regions that contradict correct alignment are colored in black. Displayed from top to bottom: alignment form initial pose $\Theta_{\text{good\_init}}$, alignment from initial pose $\Theta_{\text{trans_error}}$and alignment from initial pose $\Theta_{\text{rot\_error}}$. Left column shows results for sensor S_1 and right column for sensor S_2. Results will vary, dependent on the choice of neighborhood, weight threshold and minimal size for continuous regions.

regions and 5924 rays in 6 regions were detected. Compared with the total number of rays $(2 \cdot 146000)$ all of these numbers are small, but there is a noticeable difference between the correctly aligned scene and its two incorrect counterparts.

The introduced negative weights have an obvious influence on the output value of the originally proposed confidence measure c_{w}. The value, when taking this change into account, is now not bound in $[0, 1]$ anymore, but rather in $[-1, 1]$ (in the extreme every ray $r_{l,k} \in R_1 \cup R_2$ is part of one single connected region of misalignment). Therefore, it is sensible to distinguish between the version without negative weights (c_{w}) and the measure that incorporates the introduced negative weights, which will be denoted as c_{nw} (where the 'n' stands for negative). If one desires to attain compatibility in terms of the valid range of all introduced measures, this can be easily obtained for c_{nw} as well, by introducing a simple mapping from $[-1, 1] \mapsto [0, 1]$ defined as:

$$c_{\text{N}} = \frac{c_{\text{nw}} + 1}{2} \tag{4.44}$$

In general I have omitted to include c_{N} in the displayed results, since it can be easily obtained from c_{nw}. Along the same lines I refrained from providing the dual error measure for each confidence measure in the displayed results.

The same reasoning that lead to negative weights for connected local regions that contradict the hypothesis of correct point cloud alignment can also be employed for the direct modelling approach and its confidence measure c_{align}. Positive weights $0 \leq w_{L_i} < 1$ are assigned to leaves $L_i \in \mathcal{L}^-$ if in their local neighborhood a leaf $L_j \in \mathcal{L}^+$ is detected, where the local neighborhood $\mathcal{L}_i^N$ is defined via octree grid coordinates as

$$\mathcal{L}_i^N = \bigcup_{k,l,m=0}^{n} \left(x_{L_i}^{\text{g}} \pm k, y_{L_i}^{\text{g}} \pm l, z_{L_i}^{\text{g}} \pm m \right)^T \setminus L_i \tag{4.45}$$

where parameter n limits the size of the neighborhood. In contrast to the ray tracing approach the local neighborhood $\mathcal{L}_i^N$ does not consider the direction of a simulated measurement ray, but selects all leaves in the defined vicinity of leaf L_i. For each leaf $L_j \in \mathcal{L}^+ \cap \mathcal{L}_i^N$ the distance to leaf L_i, according to one of the introduced distance functions (equations (4.34) – (4.39)) is computed and the minimal distance is selected to determine the weight of a leaf in a similar fashion as for rays in equation (4.30). Those leaves L_i where $\mathcal{L}^+ \cap \mathcal{L}_i^N = \emptyset$ holds, retain their originally assigned weight of $w_{L_i} = 0$.

To determine the equivalent to set $\mathcal{R}^{c-}$, established in algorithm 3, an Euclidean clustering is performed on all leaves $L_i \in \mathcal{L}^-$ with the corresponding weight $w_{L_i} < w_{\text{min}}$. If a leaf belongs to such a detected cluster its weight is set to -1. The reasoning behind this is again that a spatially dense aggregation of such leaves more strongly points towards misalignment than the same number of leaves, distributed sparsely over the complete scene. Let us denote all leaves that are part of such a contradicting cluster as set $\mathcal{L}_\text{c}^-$ and the sparsely arranged contradicting leaves as the disjunct set $\mathcal{L}_\text{s}^-$, where $\mathcal{L}^- = \mathcal{L}_\text{c}^- \cup \mathcal{L}_\text{s}^-$ holds, we can modify equation (4.19) to incorporate this additional information as follows:

$$c_{\text{nc}} = \frac{|\mathcal{L}^+| + \sum_{L_i \in \mathcal{L}_\text{s}^-} w_{L_i} - |\mathcal{L}_\text{c}^-|}{|\mathcal{L}^+| + |\mathcal{L}^-|} \tag{4.46}$$

where the "nc" stands for "negative clusters". Similarly to confidence measure c_{nw}, the punishment invoked by term $-|\mathcal{L}_\text{c}^-|$ in equation (4.46) causes a change in the range of c_{nc} to $[-1, 1]$. As an extension of Table 4.4 leaves with weights $w_{\text{min}} \leq w_{L_i} < 1$ are displayed in gray.

Up until now, I have proposed different methods how to evaluate the alignment of two scans obtained via ICP (or some similar method): the first discussed idea consisted of directly modelling first the *Region of Overlap* ($ROO_{1,2}$) between the point clouds, followed by determining those parts of the scene that could actually be observed by both sensors (*Mutual Observability*), while taking occlusions into consideration (see section 4.4). As a second idea a rather indirect modelling of these principles, via ray tracing was introduced in the current section, with the possible extensions in form of added individual weights for measurement rays and "punishment" for continuous regions that contradict correct alignment.

Sorely missing, until now, is, how these proposed measures perform on the hallway example that was used to illustrate the shortcomings of the regular ICP output parameters (see Figure 4.4 and Tables 4.2 and 4.3). Tables 4.7 and 4.8 present the results for full and reduced point cloud data, respectively. For direct modelling approach an octree resolution of $l_L = 10\,\text{cm}$ was employed for both the full and the reduced point clouds, since the point density near the maximal sensor range in the full point clouds was not sufficient for a smaller leaf size. For the ray tracing approach the resolution was changed to 5 cm for the non-reduced point clouds (also employing 10 cm for the reduced version), since the intersected leaves need to contain data from only one sensor to provide evidence for a correct or incorrect alignment. All measures that employ non-uniform weights used the distance function proposed in equation (4.36) and set $\delta = 2$ in equation (4.30). The threshold to indicate strong misalignment was set to $w_{\text{min}} = 0.1$.

The confidence values in both, Table 4.7 and 4.8 show, that the alignment obtained from initial pose $\Theta_{\text{good\_init}}$ was evaluated best. When comparing the results between the methods

Algorithm 3 Creation of sets $R^{c\text{-}}_{p,k}$ that denote continuous regions of measurement rays that contradict correct alignment. The function get_neighborhood$(r_{l,k}$, expanded_rays$)$ returns a set containing all neighbors of a given ray, under consideration of the currently already expanded rays.

```
Inputs:
    $R_1$, $R_2$ – the sets of measurement rays starting from sensor positions $\check{S}_1$
    and $\check{S}_2$ $\mathcal{R}^-$– the set of measurement rays that contradict correct alignment
    $w_{\min}$– a weight threshold when to indicate strong misalignment, with
    $0 \leq w_{\min} \leq 1$
Outputs:
    $\mathcal{R}^{c-}$– the set of connected regions of contradicting rays

Initialize:
    $\mathcal{R}^{c-} \leftarrow \emptyset$, expanded_rays $\leftarrow \emptyset$, rays_to_expand $\leftarrow \emptyset$
for all $r_{l,k} \in \mathcal{R}^-$ do
    if $r_{l,k} \notin$ expanded_rays then
        expanded_rays $\leftarrow$ expanded_rays $\cup\, r_{l,k}$
        if $w_{l,k} < w_{\min}$ then
            $R^{c\text{-}}_{l,k} \leftarrow \{r_{l,k}\}$                    ▷ initialize new connected region
            rays_to_expand $\leftarrow$ get_neighborhood$(r_{l,k}$, expanded_rays$)$
            while rays_to_expand $\neq \emptyset$ do
                $r_{m,k} \leftarrow$ pick_element(rays_to_expand)
                rays_to_expand $\leftarrow$ rays_to_expand $\setminus r_{m,k}$
                expanded_rays $\leftarrow$ expanded_rays $\cup\, r_{m,k}$
                if $r_{m,k} \in \mathcal{R}^- \wedge w_{m,k} < w_{\min}$ then
                    $R^{c\text{-}}_{l,k} \leftarrow R^{c\text{-}}_{l,k} \cup r_{m,k}$
                    rays_to_expand $\leftarrow$ rays_to_expand $\cup$ get_neighborhood$(r_{m,k}$, expanded_rays$)$
                end if
            end while
            $\mathcal{R}^{c-} \leftarrow \mathcal{R}^{c-} \cup R^{c\text{-}}_{l,k}$    ▷ Add completed region to set of continuous regions
        end if
    end if
end for
```

Table 4.7: Comparison of introduced confidence measures c_{align}, c_{ray}, c_{w}, c_{nc} and c_{nw} with ICP output for unreduced scans and the poses illustrated in Figure 4.4.

	e_{avg}	$\|\|C_{1,2}\|\|$	c_{align}	c_{ray}	c_{w}	c_{nc}	c_{nw}
$\Theta_{\text{good\_init}}$	4.73 cm	65257	0.8436	0.8822	0.9245	0.9172	0.9194
$\Theta_{\text{trans\_error}}$	2.81 cm	84042	0.7777	0.8001	0.8591	0.7320	0.8263
$\Theta_{\text{rot\_error}}$	3.57 cm	79985	0.7217	0.7803	0.8456	0.6813	0.8038

Table 4.8: Comparison of introduced confidence measures c_{align}, c_{ray}, c_{w}, c_{nc} and c_{nw} with ICP output for filtered scans and the poses illustrated in Figure 4.5. For filtering the point clouds an octree with a leaf size of $l_L = 0.05\,\text{m}$ was used.

	e_{avg}	$\|\|C_{1,2}\|\|$	c_{align}	c_{ray}	c_{w}	c_{nc}	c_{nw}
$\Theta_{\text{good\_init}}$	5.06 cm	20695	0.8507	0.9083	0.9442	0.9192	0.9422
$\Theta_{\text{trans\_error}}$	4.07 cm	27582	0.7650	0.8626	0.9014	0.7240	0.8657
$\Theta_{\text{rot\_error}}$	4.48 cm	24933	0.7093	0.8387	0.8837	0.6579	0.8384

with uniform weights (c_{align} and c_{ray}) with their counterparts c_{nc} and c_{nw}, that allow smaller positive weights for slight misalignment and add a punishment if local regions in the registered scene strongly point towards incorrect alignment, the difference between the confidence values for correct alignment and both incorrect alignments becomes larger. Employing non-uniform weights without allowing negative weights for particular rays (c_{w}) still delivers the highest confidence value for the correct registration, but the difference to both other combined point clouds is smaller. Due to these indications, the experiments discussed in section 5.2 will report the values for confidence measures c_{nc} and c_{nw}, while the unreported alternatives reflect the same results.

Non-Uniform Weights for Confirming Rays

As the reader probably will already have noticed, the weight function (see equation (4.30)) introduces non-uniform weights for rays $r_{l,k} \in \mathcal{R}^-$, but all rays $r_{m,k} \in \mathcal{R}^+$ are treated equally, i.e., their associated weights $w_{m,k}$ are set to 1. This is a conscious choice in the current design, although possible alternatives exist.

For example it is plausible for rays $r_{l,k} \in \mathcal{R}^+$ to closer investigate leaf $L^\star_{l,k}$, the first leaf they intersect. So far we have only checked if this leaf contains measurement points from the sensor that emitted the measurement ray $r_{l,k}$, i.e., $L^\star_{l,k} \cap \tilde{P}_k \neq \emptyset$. While the latter certainly is a requirement for correct alignment of P_1 and P_2, with respect to measurement ray $r_{l,k}$, it leaves room for more refinement. As discussed in the introduction, a comparison based on direct point correspondences between points $p_{i,1} \in P_1$ and $p_{j,2} \in \Theta_{\text{ICP}} \cdot P_2$ is not sensible and naturally the same holds true if a more or less arbitrary volume ($L^\star_{l,k}$) is considered instead of the whole combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$.

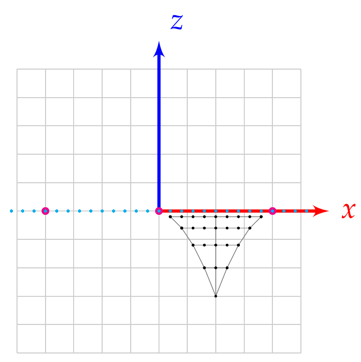
However, it is possible to locally analyze and compare the surface of both P_1 and $\Theta_{\text{ICP}} \cdot P_2$ inside, or in the vicinity of the volume defined by $L^\star_{l,k}$. Ideally such an analysis would take place on a higher level of abstraction, where we could reliably identify corresponding parts of the environment or specific objects in both point clouds and match them onto each other. Unfortunately such mechanisms are currently not available – otherwise we would not need a heuristic approach that I discuss here and the ICP algorithm or other heuristic methods to register scans could be replaced by methods that employ these correct correspondences to guarantee a global registration optimum and not only a local one. However, there exist a variety of approaches that try to locally identify distinctive regions that can be re-identified when the scene is observed from different poses. These local point features can be thought of as an extension of Lowe's SIFT features [47] or similar image features to 3D point clouds. Oftentimes features are located at points that stick out of their vicinity, for example at corners, where the local surface normals undergo a change. The geometric information of measurement points in the vicinity is then employed to provide a (hopefully) unique description of a restricted volume of the point cloud. Some of the more well known local 3D point features are published in [6, 44, 65, 69].

One thing to consider, when thinking about the usage of such 3D feature descriptors, is the data that will be used to compute them. Since the neighborhood of a feature point candidate is incorporated when generating the descriptor (and for the decision if the candidate point is sufficiently distinct or not) it might make a difference if the features are generated from the complete point clouds P_1 and P_2 or from the points that a present inside a given leaf L_i, i.e., $L_i \cap P_1$ and $L_i \cap \Theta_{\text{ICP}} \cdot P_2$, respectively. When computing features on a subset of the available data points the neighborhood that will be taken into consideration for some feature point candidates can become restricted, for example when "artificial" boundaries like the borders of a leaf partition the data. If the local neighborhood of a candidate point is restricted in such a way its descriptor will differ from the one resulting from the complete cloud information, since the descriptor incorporated this local information. Whether is it feasible to compute features only on the data present in a given leaf depends vastly on the size of the leaf: if leaves on average do not contain enough data points to compute local features, these need to be either generated from the complete point clouds or some local regions that encompass several leaves. Furthermore, depending on the employed leaf size, the mere existence of similar features computed from both point clouds in a single leaf does not necessarily point towards good alignment, for instance if the matching features are located on opposing sides inside the leaves volume.

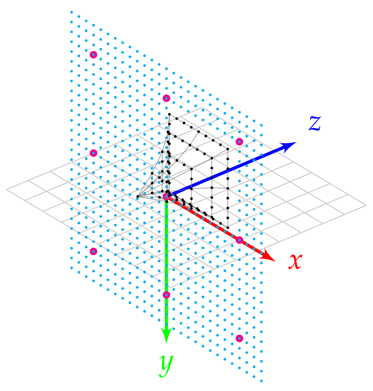
Another problem against which features are susceptible is, that most require similar point densities: if we assume perfect alignment between P_1 and P_2 for a moment and pick a leaf L_i that contains points of both point clouds it can happen that feature analysis would suggest an imperfect alignment. Let us assume that the point density for points from P_1 inside L_i is much higher than the density of points $\Theta_{\text{ICP}} \cdot P_2$, possibly because S_1 is much closer to L_i than S_2. In this case the points obtained from sensor S_1 might capture some detail, that creates a very distinct feature, but which is lost in the much sparser resolution available for the points $L_i \cap \Theta_{\text{ICP}} \cdot P_2$.

This phenomenon is illustrated in a schematic fashion in Figure 4.15. Two point clouds P_1, consisting of the cyan and black points, and $\Theta_{\text{ICP}} \cdot P_2$ (magenta) are depicted. While the tip of the black pyramid is a likely location for most 3D feature point detectors, this detail is lost in $\Theta_{\text{ICP}} \cdot P_2$, due to lower point density. Conversely, computing features on the magenta points of the depicted segment of the environment should not yield a distinctive feature point, since the magenta points form a plane and are locally not distinct from each other. A comparison based on local features between both point clouds P_1 and $\Theta_{\text{ICP}} \cdot P_2$ in the depicted volume will therefore not indicate that these two point clouds resemble each other.

However, if the black points were located in the x-y-plane, like their cyan companions, local features between both P_1 and $\Theta_{\text{ICP}} \cdot P_2$ were likely similar very again, i.e., both clouds would not provide distinct features in the depicted part of the environment. Note, that while the data provided by Figure 4.15 is clearly artificial and schematic in nature, the illustrated problem can also exist in real scenes, when portions of the scene are observed from different poses.



(a) Top view



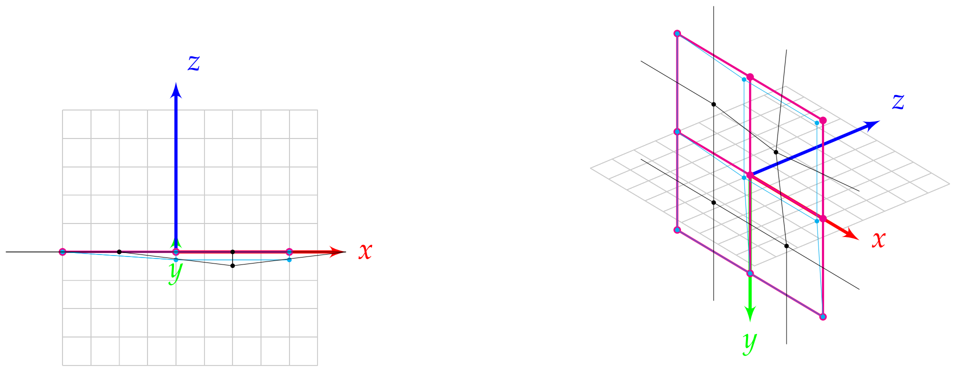
(b) Perspective view

Figure 4.15: Schematic representation how different point densities may influence feature computation and meshing approaches. Cyan and black points (P_1) depict measurements with a high point density, while magenta points ($\Theta_{\text{ICP}} \cdot P_2$) mark measurements from the same scene at a much lower point density. At the low density of the magenta points, distinct details (black points) are lost. Note that the lines, connecting the black points, are only depicted for viewer convenience.

One could argue, that the problems caused for local features can be reduced by filtering point clouds, so that the point density in P_1 and $\Theta_{\text{ICP}} \cdot P_2$ in a given volumes becomes more even. Doing this sensibly will always result in reducing the density of the point cloud that features more points, since regular interpolation between existing measurement points in the sparser point cloud will not generate any more meaningful information (and "hallucinating" arbitrary points also does not seem to provide much insight). Unfortunately downsampling of the denser point cloud has a caveat: reducing the point density and representing multiple measurements by their center of gravity or some other point that aggregates their information will result in a loss of information, especially concerning distinct data points, that will be somewhat evened out by their neighboring points. This is illustrated in Figure 4.16, where the artificial data from Figure 4.15 is reduced in two different ways.

Of course there are different ways how the point data from P_1 in Figure 4.15 can be downsampled. The two different results depicted in Figure 4.16 both generate the similar point density, i.e., the measurement points in the regular downsampled point cloud have the same distances as points in $\Theta_{\text{ICP}} \cdot P_2$. Furthermore, no point of the original cloud $\Theta_{\text{ICP}} \cdot P_2$ is discarded when computing the downsampling. Independent which of the two different results is considered, the representation of the pyramid of the original data in Figure 4.15 is much less distinct in Figure 4.16, thus illustrating the point that downsampling introduces the risk of discarding suitable feature candidates.

A different type of problem, with respect to associating local 3D features, can be caused by the different poses of S_1 and S_2, when recording the scene. If the viewpoints of sensors S_1 and S_2 are sufficiently different, occlusions in the scene can prevent the computation of matching features. If a leaf L_i contains some object and the points $L_i \cap P_1$ are taken from the front side of this object, while $L_i \cap \Theta_{\text{ICP}} \cdot P_2$ represents the back side of the same object, it cannot be expected that features that are computed from the individual points cloud match at this portion of space.



(a) Top view **(b)** Perspective view

Figure 4.16: Effects from point density reduction. The dense point data from Figure 4.15 is downsampled, using different sampling points. Point cloud P_1 was reduced and is represented by either the cyan or the black points, while point cloud $\Theta_{\text{ICP}} \cdot P_2$ (magenta) has retained all points. Cyan sampling points correspond in their x- and y-coordinate with the magenta points of the sparse cloud. One of the black points coincides with the tip of the pyramid in Figure 4.15. Grid lines between point data is added for viewers convenience, to emphasize the 3D nature of the depicted data.

Obviously if, for a given ray $r_{l,k} \in \mathcal{R}^+$, the first intersected leaf $L^\star_{l,k}$ contains some feature points for P_1 and matching features computed from $\Theta_{\text{ICP}} \cdot P_2$ (or the some subsets of the point clouds) the probability that both clouds are aligned correctly, with respect to ray $r_{l,k}$ is higher. In these cases the corresponding weight $w_{l,k}$ could be increased. However, as illustrated above, the absence of matching feature points does not necessarily indicate wrong alignment, but could be caused either by different point densities or occlusions. Should weights be decreased in these cases, to indicate that the matching might not be perfect? Also one needs to take into consideration in which way feature matches will increase weight: if weights are increased too much, a few select rays where the features match really well by chance could compensate for large misalignments in other parts of the combined point cloud. Such a behavior would surely not be desirable.

Instead of local 3D point features, the surface inside the leaves $L^\star_{l,k}$ could be taken into account on a coarser, when thinking about changing the weights for rays $r_{l,k} \in \mathcal{R}^+$. For example an algorithm like Marching Cubes [46] could be used to generate a mesh representation of P_1 and $\Theta_{\text{ICP}} \cdot P_2$, where the cubes size and location should coincide with the octree leaves. After the surface reconstruction a local comparison of the portions of the meshes that are located inside $L^\star_{l,k}$ could be performed and depending on the similarity of the two surfaces higher or lower values could be assigned to weight $w_{l,k}$. However, the problems sketched for the comparison of 3D feature points transition to this comparison method as well. Differences in point density might cause differences in the reconstructed surfaces, if the measurement points with the smaller point density lack some details, present at the higher point density.

This becomes evident if we think about a mesh representation of $\Theta_{\text{ICP}} \cdot P_2$ (magenta) in Figure 4.15: since all 9 measurement points lie in a plane a subdivision via marching cubes, assuming that the cubes are axis aligned, will produce the same type of cube configuration for each cube. If point cloud P_1 (cyan and black) is to be approximated appropriately, the used cube configuration should differ, depending on the location of the cube in the depicted scene. Similar arguments can be made for mesh approximation that do not partition the scene into uniform volumes, since the points in P_1 will need far more triangles to accurately represent the surface than the points in $\Theta_{\text{ICP}} \cdot P_2$. Naturally the problems with in-scene occlusions continue to exist as well as how to distinguish between different local surface approximations caused by misalignment and a correct alignment, where different surfaces from different viewpoints were observed in the same leaf.

4.6 Inherent Shortcomings

As mentioned in the introduction of this chapter, the proposed measures to establish a confidence / error value for the alignment quality of two point clouds P_1 and P_2 via a given pose Θ_{ICP} are heuristic approaches to this problem. I firmly believe that approaches that might address this problem in a non-heuristic way require some mechanism that allows for a reliable association between subsets of P_1 and P_2, that go beyond the representation of purely geometrical properties in some sophisticated way, like 3D point features, but a mapping on a higher level. Better object recognition helps in this case, since it allows to conclude (partial) misalignment if, for instance, points from a chair in P_1 spatially coincide with points from $\Theta_{\text{ICP}} \cdot P_2$ that were identified to belong to table or cabinet. Coinciding in this case implies that a portion of the points need to occupy the same volume in 3D space – the occurrence of a chair in close vicinity of a table should not be considered a misalignment nor should overlapping of the objects bounding boxes be seen as misalignment (think about a chair, that is positioned in a way that its seating surface is located below a tabletop).

Additionally identifying objects in connection with some internal representation of said objects could also help provide positive feedback about the alignment of P_1 and P_2: if a recognized object is recorded from one side in P_1 and, without much or no overlap, from the other side in P_2 and both parts are fitted together in the combined cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ this information should be used to denote that, at least in the spatial vicinity of said object, P_1 and P_2 seem to be aligned well.

Unfortunately object detection alone will not provide enough information. Especially if multiple physically identical instances of an object exist in a scene, some method to distinguish between these instances in P_1 and associate them with the corresponding instances in P_2, provided they are present there as well, are needed. These associations also need not only take into consideration the best individual association for a given object, but also the in-scene arrangement with other objects in the scene, i.e., if P_1 contains a chair located to the “left” of a table and both objects are observable from $\tilde{S}_2$, the position of sensor S_2 in the combined scene, the point cloud $\Theta_{\text{ICP}} \cdot P_2$ would need to also contain points of these two objects at these positions and lack of either table or chair would already point towards some misalignment.

So in addition to object detection, understanding of the scene and the spatial relationships between the detected objects are also needed. Unfortunately this problem is currently not solved in its aggregate, although the last years has seen some impressive results in some of the involved subdomains, like object recognition (for example [12,38,47]) and mechanisms that allow to encode spatial information and reason about it ([19,60]).

If these tools were available, heuristic alignment techniques like ICP or by employing a combination of 3D features would also become obsolete: having reliable information about which parts of P_1 correspond to which parts of P_2, but using it only for the evaluation of the arrangement obtained by some coarser mechanism, seems rather unintuitive.

A characteristic of heuristic approaches is that they usually yield good solutions, but occasionally may fail to provide sensible results. Often, only a heuristic approach makes complicated problems tractable, since their approximate solutions can generally be obtained much more quickly than exact solutions. In case of the evaluation measure this means that in the majority of times the evaluation measure should correctly indicate if an alignment was good or not, but there will occasionally be cases where the alignment is evaluated in a wrong way. However, some specific cases where the two different methods, direct modelling of mutual observability (4.4) and ray-tracing based (4.5), are likely to fail can be described, so that these can be taken into consideration when computing the alignment evaluation.

Both methods need sufficient overlap of point clouds P_1 and P_2 when aligned by the ground truth pose, i.e., when recording data it needs to be taken care that enough portions of the environment are represented in both scans. If we think of an extreme case, where no overlap exists in the final pose estimation between P_1 and P_2 the two introduced approaches fail in similar ways:

If we consider how c_{align} and c_{nc} are computed (see equations (4.19) and (4.46) respectively), these depend on the sets $\mathcal{L}^+$ and $\mathcal{L}^-$. According to Algorithm 1 these sets are computed from the octree that is constructed from those parts of the point cloud that are, at least in theory, observable from both sensors S_1 and S_2. The absence of overlap in the final pose estimation between P_1 and P_2, i.e., $ROO_{1,2} \cap (P_1 \cup \Theta_{\text{ICP}} \cdot P_2) = \emptyset$, causes the octree $\mathcal{L}$ to be generated from an empty point cloud, therefore containing no leaves and subsequently $\mathcal{L}^+ = \mathcal{L}^- = \emptyset$ holds. This effectively prevents a (meaningful) computation of e_{align} and c_{align}, since numerator and denominator in equations (4.19) and (4.46) become 0.

In case of the evaluation measures based on ray tracing (c_{ray}, c_{w} and c_{nw}) such a scenario both sets $\mathcal{R}^+$ and $\mathcal{R}^-$ would also be empty. While for each ray $r_{l,k}$ the first encountered leaf would be in measurement range and contain points from $\check{P}_k$, i.e., $L^{\rho}_{l,k} = L^{\star}_{l,k}$ the check if this leaf is also observable from sensor position $\tilde{S}_m$ would fail and therefore the ray would not be added to either set.

The case that P_1 and P_2 do not overlap at all in the final pose estimation (which also requires no overlap in the initial pose estimation) is an extreme, but it serves to illustrate that the smaller the area of overlap and, in conjunction, the smaller the sets $\mathcal{L}^+$ and $\mathcal{L}^-$or $\mathcal{R}^+$ and $\mathcal{R}^-$respectively, become, the larger the influence of sensor noise and similar effects. If these sets are small, the influence of a few false classification, either by indicating wrong or correct alignment gets much higher. Please note, that sensor noise might, of course, also result in an octree leaf becoming occupied by points from both sensors, that would without noise only contain points gathered by one sensor – thus implying correct alignment.

Fortunately the overlap requirement is not purely a requirement for the alignment evaluation, but also for the registration algorithms. In an automated fashion one could check how much points of the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ overlap and in case of insufficient overlap either produce a warning or abort the evaluation process. The degree of overlap can be

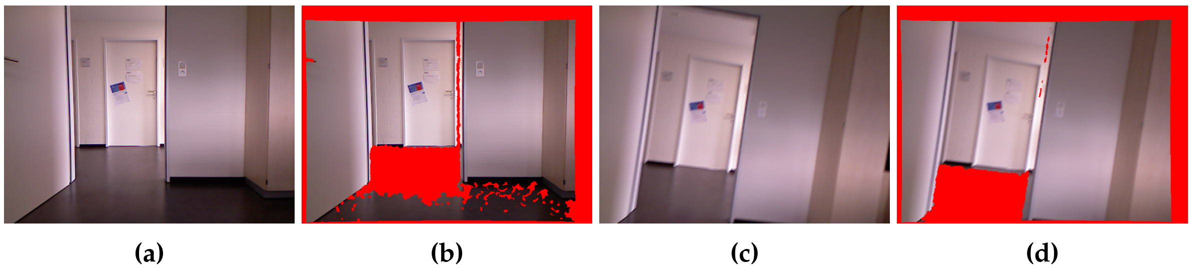


Figure 4.17: Scene photographs of the point clouds depicted in Figure 4.18. Images **(a)** and **(c)** show the regular photographs, while in corresponding images **(b)** and **(d)** all pixels that are associated with a `NaN`-measurement are colored in red.

approximated via the construction of an octree and checking the ratio of leaves that contain points from both sensors versus leaves that contain only points gathered from one sensor.

Apart from alignments where P_1 and $\Theta_{\text{ICP}} \cdot P_2$ share no or very little overlap a certain type of misalignment will cause both evaluation methods to fail. If two scans are falsely aligned in a way that a portion of the combined point cloud, that contains points from both sources, is positioned between the location of S_1 and $\tilde{S}_2$ this will lead to a high value for c_{align}, $c_{\text{ray}}$and c_{w}. An example of such an arrangement of two scans would be, if two scans taken from inside a single room are aligned in a way that points from a wall in P_1 coincide with points from a wall in $\Theta_{\text{ICP}} \cdot P_2$, but the combination of the point clouds now resemble two adjacent rooms instead of a single one. An illustration of such a scenario is depicted in Figure 4.18 and Figure 4.17 provides the corresponding photographs of the recorded scene.

In this case substantial parts of the points from P_1 and $\Theta_{\text{ICP}} \cdot P_2$ exists where their points coincide, if the number of points from both clouds are considered. However, if we compare the volume of the common portion of the combined point cloud it is rather small with respect to its overall dimensions. This can be observed in Figure 4.18c, where only points that belong to the wall right of the doorway in the foreground (see Figure 4.17) coincide, but these constitute to almost 30 % of all possible point associations in the complete scene, i.e., 30 % of the points in P_2 established a point-to-point correspondence during ICP to some point in P_1. If we consider the volume that these points make up of the total volume, occupied by measurements in Figure 4.18c it is much less than 30 %.

Employing the direct modelling approach in such a case will cause the leaves along the wall to be inserted into set $\mathcal{L}^+$, since these leaves contain points of both P_1 and $\Theta_{\text{ICP}} \cdot P_2$. The majority of the other leaves that are inside the region of overlap $ROO_{1,2}$ will only contain points $\breve{p}_{l,k}$ gathered from a single source, but, since the wall is (conveniently) placed between the positions of S_1 and S_2 in the combined point cloud, most of these leaves will be occluded when observed from the position of $\breve{S}_m$ (with $k, m \in \{1, 2\}, m \neq k$). Hence, these leaves will neither be inserted into set $\mathcal{L}^-$ or $\mathcal{L}^+$ (see Algorithm 1 for details) and thus not directly contribute to c_{align}(see equation (4.19)). One can argue that such leaves indirectly contribute to the overall value by reducing the number of leaves inserted into either $\mathcal{L}^-$ or $\mathcal{L}^+$, thus effectively reducing the denominator in equations (4.18) and (4.19). The resulting effect is similar to two point clouds that have very little overlap, i.e., the ratio of leaves that contribute to the confidence

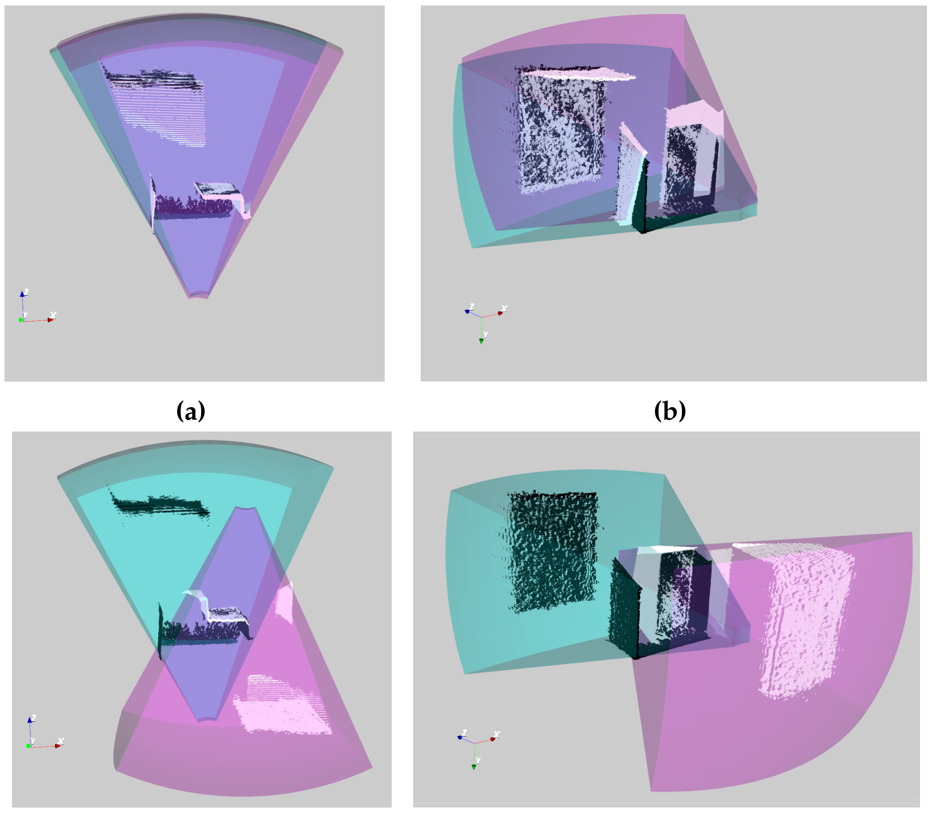
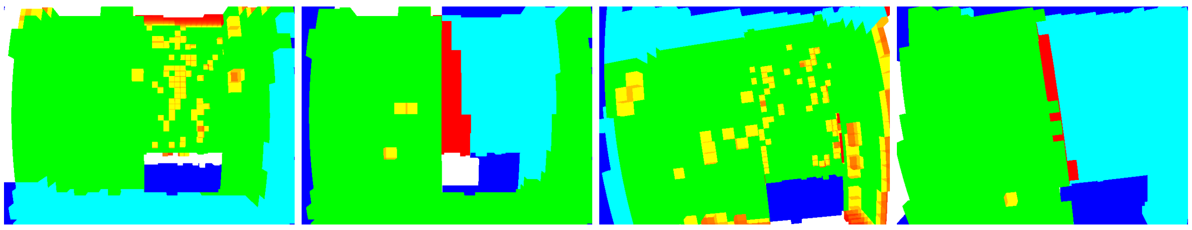


Figure 4.18: Example of problematic alignment: Point clouds in black (P_1) and white (P_2). Corresponding FOVs are shown in the common color scheme. Top row (**(a)** and **(b)**) show good alignment, bottom row (**(c)** and **(d)**) depict wrong alignment that is challenging to detect by the proposed evaluation measures.

measures becomes small.

If we consider the setup sketched in Figures 4.18c and 4.18d with respect to the ray tracing based methods, the outcome will again be similar to the direct modelling approach. For the vast majority of measurement rays $r_{l,k}$ that are directed towards the common wall segment the first intersected leaf will be in range and contain measurement points from the sensor that emitted the ray, i.e., condition $L^{\rho}_{l,k} = L^{\star}_{l,k}$ holds and subsequently these rays are inserted into set $\mathcal{R}^{+}$, since the shared wall segment is observable from both sensors. Furthermore, if the overlap of the fields of view of both sensors in the depicted alignment is considered, almost all measurements that might indicate misalignment are outside the field of view of either S_1 or $\Theta_{\text{ICP}} \cdot S_2$, as evident in Figure 4.18c. Rays $r_{l,k}$intersecting these parts of the combined point cloud are neither inserted into $\mathcal{R}^{-}$, since no contradicting information is gathered, nor into $\mathcal{R}^{+}$, because they are not in the field of view of sensor $\check{S}_m$, with $k, m \in \{1, 2\}, k \neq m$ (for details please refer to Algorithm 2).

Figure 4.19 provides a visualization of the ray tracing based method for the two alignments



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

Figure 4.19: Visualization of ray tracing for alignments from Figure 4.18: **(a)** depicts the results of the ray tracing analysis for correct alignment (Figures 4.18a and 4.18b) from the point of view of sensor S_1 (black and cyan in Figure 4.18) and **(b)** displays the results for wrong alignment (see Figures 4.18c, 4.18d). **(c)** and **(d)** respectively show the results from the point of view of sensor S_2 (white and magenta in Figure 4.18). For information concerning color coding please refer to Table 4.6.

	c_{align}	c_{ray}	c_{w}	c_{nc}	c_{nw}
good alignment	0.7359	0.8227	0.8776	0.7338	0.8261
wrong alignment	0.6246	0.9112	0.9149	0.3575	0.8337

Table 4.9: Comparison of confidence values for problematic alignment (Figure 4.18). Octree resolution was set to $l_L = 10\,\text{cm}$ to cope with Kinect noise at larger distances.

shown in Figure 4.18. While the result for sensor S_1 in the incorrect alignment (Figure 4.19b) contains a distinct connected red area, indicating the misalignment near the door, another detail might be a bit more surprising at first: apart from this single area of wrong alignment the rest of the scene features very little portions where the expectations for correct alignment were not met, i.e., rays $r_{l,k}$ did not first encounter a leaf $L^{\rho}_{l,k}$ in measurement where $L^{\rho}_{l,k} \cap \check{P}_k = \emptyset$. This observation is also true for sensor S_2 (Figure 4.19d). In contrast, the visualization for the correct alignments (Figures 4.19a and 4.19c) do not feature one large connected region that indicates misalignments, but they contain more scattered red and yellow points that mark rays $r_{l,k} \in \mathcal{R}^-$. A large portion of those are gathered on the door (compare with Figure 4.17) and are caused by the decreasing depth resolution and increase in noise at larger distances for the Kinect sensor. In the misaligned scene the majority of these parts of the scene are not located inside of $ROO_{1,2}$ (cyan portions in Figures 4.19b and 4.19d) and therefore do not influence the confidence score for this alignment. These properties (very little misalignment apart from one distinct area) are reflected by the confidence values for the ray tracing-bases approach in Table 4.9.

The direct modelling approach is able to make a better distinction between correct and incorrect alignment, especially when negative weights are added to clusters of contradicting octree leaves (value of c_{nc} in Table 4.9). Figure 4.20 shows a visualization of the direct modelling approach. In the wrongly aligned scene (Figure 4.20b) the amount of points located outside of the $ROO_{1,2}$ (shown in blue) is much larger than in case of correct alignment (Figure 4.20a). Even though confidence measure c_{nc} is able to distinguish between correct and wrong alignments, similar scenarios (for example a wall without a doorway) can cause the evaluation to fail.

In cases where large parts of the combined point cloud $P_1 \cup \Theta_{\text{ICP}} \cdot P_2$ correspond well and

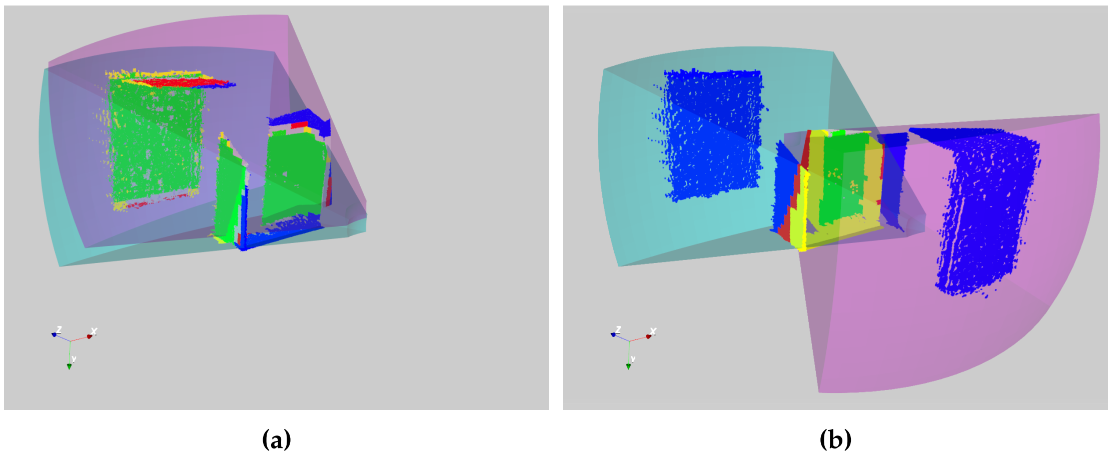


Figure 4.20: Visualization direct modelling approach: **(a)** shows visualization for correct alignment and **(b)** for wrong alignment.

only minor parts of the scene do not match, one should not expect a small confidence values for the proposed confidence measures. An example of such a setup is the hallway example, that served as illustration in most parts of this chapter and the three different investigated alignments (Figure 4.4). While the confidence score for the correct alignment is higher than in the two wrongly aligned cases, both wrongly registered versions were assigned confidence values $0.5 < c < 1$, independent on the specific measure employed. Similarly, if we ignore some of the smaller details in Figures 4.4c – 4.4f the depicted alignments seem good at first glance, even for a human observer. Since the evaluation measures are based on the geometric information such a behaviour is to be expected and arguably the depicted erroneous alignments are "better" than flawed alignments where floor, ceiling and walls of the hallway in P_1 and P_2 do not coincide.

This basis on the geometric properties unfortunately also implies that confidence scores that imply "good" or "sufficient" alignment will generally differ, depending on the employed sensor and its noise characteristics and also, to a certain degree on the recorded scenes. For example, if one of the initial assumptions, like a static scene, are violated then, confidence values for correctly aligned scenes may become lower than for scenes, where the point clouds are registered incorrectly, but still contain large parts that seemingly correspond well, as in the hallway example. For instance, outdoor scenes, that contain plants and are recorded on not completely windless days, usually violate the static scene assumption to at least a certain degree.

Furthermore, incompleteness in the recorded data can lead to lower confidence scores. The term incompleteness in this case means that one of the point clouds $\check{P}_k$, contains measurement points a surface, while these are missing in $\check{P}_m$, although the sensor was (in theory) able to observe this surface. Such incompleteness in the recorded data can be found in RGB-D data, when the surface is glossy and illuminated with a direct light source, for instance. In case of laser scanners, windows sometimes produce measurement points, depending on the angle of the measurement ray that hits the window and its cleanness, and sometimes will not produce measurements. If any of the point clouds P_1 or P_2 misses data in this fashion, this will result in lower confidence scores, because, without interpretation on a higher level, these data inconsistencies cannot be distinguished from inconsistencies causes by misalignment. Of course, if no geometric properties allow to distinguish a correct from an incorrect alignment, for example in a hallway where translation errors can only be detected by employing other sensor information, like color, the proposed methods is also not able to indicate correct or wrong alignment.

In addition sensors that offer a vastly differing degree of precision with respect to the distance of a measured surface also pose a challenge. Common RGB-D cameras tend to fall in this category, since the measurement noise in the sensible measurement range of up to $\approx 2.5\,\mathrm{m}$ is substantially less than at their maximal range $\approx 10\,\mathrm{m}$, due to decreasing depth resolution (for more details please refer to [35]). Choosing a small size for the leaf size via parameter l_L might cause a lot of detected misalignments in those portions of the combined point where points $\breve{p}_{l,k} \in \breve{P}_k$ are present that have a large distance to the associated sensor $\breve{S}_k$, when the sensor noise becomes larger than l_L in these regions. On the other hand, if a larger leaf size is chosen, this might become too coarse for those parts of the scan where noise is rather small, i.e., near the positions of $\breve{S}_k$, and wrongly aligned points might appear to be corresponding correctly on this larger scale view. Depending on the scenario choosing a smaller size of the octree leaves and ignoring measurement points behind a certain range can act as a workaround for this flaw. Laser scanners, where the increase of noise with respect to the measured distance is much smaller, usually do not suffer from this problem.

Chapter 5

Experimental Results

5.1 Transparent Objects

5.1.1 Detection

When determining, if a scene contains transparent objects or objects with a highly reflective surface, several possibilities exist, depending on the type of data that is available. The evaluation on the data presented next employs the more general method of detection in unorganized point clouds, as described in subsection 3.2.1. Though detection in organized point clouds is fundamentally simpler, these are not always available, depending on the sensor that recorded the data and if filters already have been applied on the data, since some filter operations may destroy the organized property of point clouds. This is also true for the scenes that were investigated here, since the performed outlier removal also removed all `NaN`-measurements, thus destroying the originally organized property of the input data. In total 30 scenes were recorded by a Kinect sensor, where each scene contained at least one planar surface – either as a support plane or arranged behind the objects. The objects that were present in the scenes belong to one of three distinct categories, namely *Regular*, *Specular* and *Transparent*. Descriptions of these categories are listed in Table 5.1

Each scene contained at least one object from these categories, while the majority of scenes featured multiple objects. A sample of the scenes, from the sensor's point of view, can be seen in Figure 5.1. Performance of the detection on this dataset are shown in Table 5.2. Apart from a good detection rate, i.e., a high recall value, a usable detector should also not create too many false positives in relation to the number of true positives (precision value). The scenes contained a total of 32 regular objects, 7 specular objects and 9 objects that were transparent. Of the 30 collected scenes 16 frames contained at least one non-regular object, i.e., transparent and/or specular object. The presence of the latter were successfully detected in 14 of the 16 scenes, as listed in the bottom line of Table 5.2. While first three lines in Table report the detection rates for the three distinct classes, the detector in the bottom line groups transparent and specular objects into a single category and distinguishes between them and regular objects.

As is evident from the first row in Table 5.2 the detection results for regular objects are also not perfect, with 4 false positives. These are caused by setting the maximal distance for the Euclidean cluster extraction to 2 cm, i.e., if two points are separated by more than 2 cm and no

Category	Description
Regular	Objects that can be measured directly from the sensor.
Specular	Objects that feature highly reflective or specular surfaces. Scarce measurements may occur if the surface is arranged in an (almost) perpendicular fashion to incoming measurement rays (see Figure 3.4a).
Transparent	Objects made from transparent materials, that provide either no measurements at all, or very few, typically near the support plane with a lot of sensor noise.

Table 5.1: Object categories for transparent object detection in experiment for the detection in unorganized point clouds.

	true positives	false positives	false negatives	precision	recall	F_1 score
Regular	32	4	0	88.9 %	100.0 %	94.1 %
Specular	5	0	2	100.0 %	71.4 %	83.3 %
Transparent	8	4	1	66.7 %	88.9 %	76.2 %
Specular and/or Transparent	14	0	2	100.0 %	87.5 %	93.3 %

Table 5.2: Detection results for regular, transparent and specular objects.

other points are located between them, they are not grouped in the same cluster. This can cause some objects to be split into two distinct clusters, especially if part of the object is occluded or they are located at larger distances to the sensor where the gaps in the depth estimation start to grow for triangulation based sensors (see 2.4). On the other hand, using larger thresholds during the cluster extraction will cause distinct objects to be merged into a single cluster. In order to not detect too many transparent objects, the ratio of unexplained grid cells for a hole region was set to 0.5, i.e., a hole region is only assumed to be caused by a transparent object if at least 50 % of its grid cells cannot be attributed to occlusion by some recorded measurements.

Depending on the scene a single misclassification might result in three negative entries

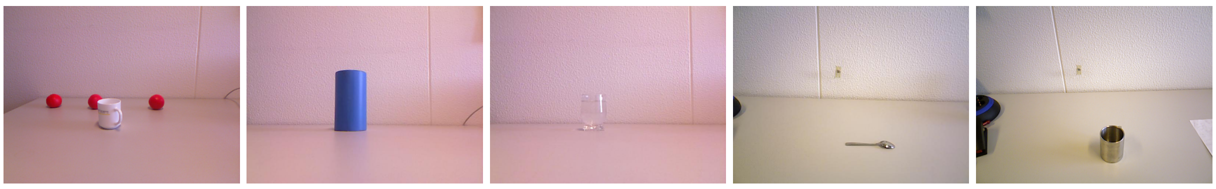
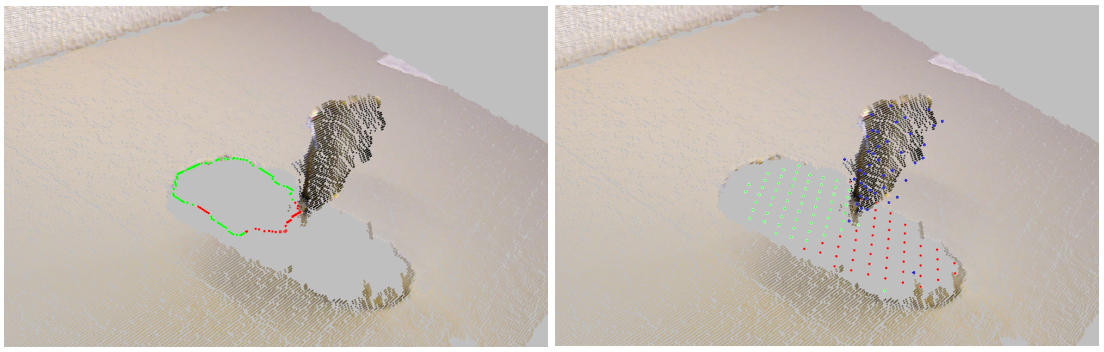


Figure 5.1: Subset of the scenes for detection of specular / transparent objects.



(a) **(b)**

Figure 5.2: Example of misclassification of specular object: **(a)** visualizes the analysis of the outline projection of the depicted cluster while **(b)** shows the result of the hole analysis. Displayed data corresponds to the scene depicted in Figure 5.1 on the far right.

in Table 5.2: if the projected outline of a specular object cluster fits the shape of the caused occlusion on the tabletop well for a large part, this object will be labeled as a regular object, hence creating one false positive (in the regular object category) and a false negative, since the specular object was not recognized. In addition, if the analysis of the corresponding hole region features a large portion of unexplained empty grid cells, this same object might furthermore result in the detected presence of a transparent object, therefore causing an additional false positive in the transparent object category.

One example of misclassification is provided in Figure 5.2, where a specular object is wrongly classified as regular. The point cloud corresponds to the scene with the metal mug on the far right of Figure 5.1 and the recorded measurement points of the mug are located on its interior, while the complete front side was not recorded. The majority of the projected outline of the cluster fits the shape of the hole (green points in Figure 5.2a), so that the criterion for the cluster belonging to a highly reflective object is not fulfilled. Fortunately in this case a little less than 50 % of the hole region is left unexplained, so that this particular object is responsible for just 2 wrong entries in Table 5.2.

The threshold necessary for a successful plane detection, when applied on the rather noisy data provided by RGB-D cameras combined with the grid cell approach to detect hole regions of suitable size for further investigation may, of course, result in missing some rather small or delicate objects. An example is given in Figure 5.3 where recorded data, containing a metal spoon (see Figure 5.1, second from the right) is shown. Since the spoon in this setup is a rather flat object (Figure 5.3a), almost all of the few surface measurements are classified as planar inliers of the tabletop. When these inliers are projected into the plane and the occupancy grid is constructed, the spoon results only in a very small hole region. This is evident in the spacing of the blue points in Figure 5.3b that visualize the reference points of occupied grid cells in the planar surface, where no hole of sufficient size is caused by the metal spoon. Hence, in the detection this scene was classified as containing no object, apart from the two detected planar surfaces.

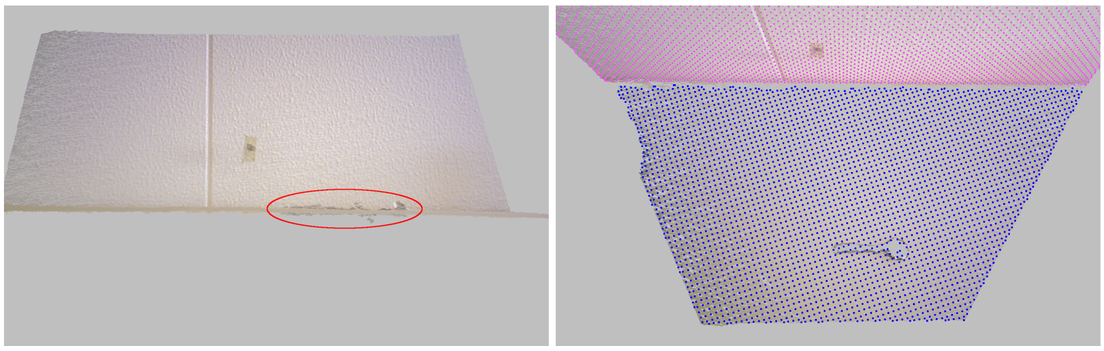

(a) (b)

Figure 5.3: Example for problems with delicate objects: **(a)** and **(b)** show the point cloud recorded for a spoon placed onto a table (see Figure 5.1, second from the right). In addition to the recorded measurements the top view in **(b)** also superimposes the resulting grid representation of the extracted planes.

Employing a smaller plane inlier threshold or a less coarse occupancy grid will result in detecting many false positives, if we consider how noisy the recorded data is in Figure 5.3 and how many gaps are in between measurements. Note that the occupancy grid, employed here, returns the center of gravity of all points contained in a cell as its reference. Therefore, the reference points do usually not coincide with the center of a grid cell, but resemble the measured data in a slightly more accurate way.

While the results in Table 5.2 are not perfect, they were produced without much fine-tuning of the involved parameters (plane inlier threshold, Euclidean clustering threshold, sample grid discretization size, ratio of explain vs unexplained points, etc.). Furthermore, results could probably also be improved if the analysis of the clusters projected outline was directly linked with the results of the detected hole regions, which was not done in the reported results. However, since already this very basic detection in unorganized point clouds performs reasonably well and also employs some of the principle ideas involved in the proposed reconstruction, i.e., consideration of occlusion and viewpoint, the detection method was not further refined.

The simpler detection idea for organized point clouds (see 3.2.2) is not reported explicitly here, but can be seen implicitly in the experiments conducted later in 5.1.3. These experiments will also demonstrate that by employing the more robust reconstruction scheme, as proposed in subsection 3.3.3, is able to still successfully reconstruct a transparent object, even if it was not detected in a subset of available views, i.e., a very precise detection becomes less important than in the reconstruction based on direct intersection (see subsection 3.3.2).

Note however, that the detection whether there is a non-regular object in the scene (bottom line of Table 5.2) already provides good results and might be used as a primer to investigate if a process like VPBR should be started.

5.1.2 Intersection-Based Reconstruction

As subsection 5.1.1 indicated, it is possible to detect the occurrence of transparent or specular objects in single, unorganized 3D point cloud, this section proceeds to present some results of the (subsequent) actual reconstruction. Reconstruction in this part is done via the intersection method (see subsection 3.3.2) and done in two separate steps. First a preliminary evaluation of the general idea under simplified assumptions and conditions is presented to illustrate that a reconstruction in the fashion as proposed in section 3.3 is indeed possible. This first proof of concept is followed by the second step, namely a demonstration on actual transparent objects, but also under some simplified conditions.

The conclusion of this subsection points out some disadvantages of the reconstruction by intersection and highlights these by some examples. These are followed by the non-simplified use case in subsection 5.1.3, where the obtained reconstructions are compared in a qualitative and quantitative way with the best possible reconstruction under the experimental conditions as well as a measure of "ground truth", as far as that is possible. However, reconstruction is then not performed via intersection, but by viewpoint-based reconstruction.

Proof of Concept

As a proof of concept that it is possible to approximate shape and volume of an object based on the missing measurement points of the support plane, I first conducted a preliminary experiment with the following simplified conditions:

1. Point clouds are aligned by hand
2. The reconstructed object is present in all point clouds
3. The pure intersection-based reconstruction (see section 3.3.2) is used
4. Instead of a transparent object a regular object was used and its points were manually removed from the point cloud before the actual reconstruction.

All these relaxations are somewhat interlinked with each other and I will briefly argue why they are sensible simplifications in order to demonstrate the principle applicability of the proposed reconstruction method. As stated in section 3.3, it is vital for the reconstruction to align the multiple point clouds in a concise fashion. Therefore, it is prudent, when trying to evaluate the principle workings of the reconstruction method, to remove this potential source of errors. In this way it becomes possible to conclude, in the case of unsatisfactory reconstruction results, that the proposed method in itself is insufficient and the error is not induced by some external source, i.e., the wrong alignment of the point clouds. Closely connected to this are relaxations 2 and 3: if we can assume that all scans are registered correctly and that the object is present in all point clouds, the methods (see 3.3.3) to cope with these types of noise, albeit at the cost of a less precise reconstruction can be momentarily ignored.

Lastly in order to be able to get a meaningful idea of how precise the reconstruction can be under optimal conditions, the occlusion caused by a regular object, instead of a transparent one, were used to reconstruct the object's shape (simplification 4). This provides for an opportunity to directly compare the reconstruction with the shape of the actual object – which is not available, if it is not possible to measure the object directly. Note that before the reconstruction was performed, all points belonging to the object to be reconstructed (in this specific case a mug), were manually removed from the point cloud, which causes a connected region of no measurements in the support plane, just like a transparent object.

Since the alignments between the individual point clouds are known, they can be applied to the manually removed object points and thus a (complete) representation of the actual object becomes available. As opposed to measure a transparent object with a different kind of sensor, say some sensor based on ultrasonic measurements, this approach has the advantage, that different sensors do not need to be calibrated against each other and all measurements are subject to the same sensor noise, i.e., if there is a noise measurement on the surface of the object, this is also reflected in the "shadow" this measurement causes on the support plane.

In addition the occlusion caused on the support plane by regular objects tend to be much clearer than those of actually transparent objects, so that their approximation by the use of the convex hull can be omitted and the actual outline can be used. This property also allows to consider small groups of measurements of the support plane that are embedded into a larger region of non-measurements, for example caused by the handle of a mug. When (slightly) imprecise alignment and more sensor noise needs to be considered, as in the general application scenario that will be discussed in subsection 5.1.3, it is prudent to remove stray measurements that are embedded into `NaN`-regions, but when we want to explore how good a reconstruction can be in principle, it should prove interesting to also investigate these points and determine if they could also be used in the unsimplified reconstruction.

With these simplifications in place, the actual reconstruction was comprised of the views displayed in Figure 5.4. In all scenes the mug is recorded from a somewhat frontal position and an elevated viewpoint. While this is not ideal for the reconstruction results, as will be shown in a moment, and therefore slightly contradictory to the previously introduced simplifications, it is a realistic restriction for many application scenarios: oftentimes it might not be possible to observe some objects from all possible sides, due to in-scene constraints, like the wall behind the tabletop in Figure 5.4. Furthermore, not all robotic platforms, our own used in the experiments detailed in subsection 5.1.3 included, do not have the actuator capabilities to change the elevation and / or angle of their sensors in order to make their setup more simple and robust and less error prone when integrating sensor information into their world model. Therefore, it is desirable that such constraints do not prevent a reconstruction per se. However, it should also be clear that a richer source of inputs in terms of sensor elevation and position around the object, that is incorporated into the reconstruction, will provide better results, under the assumption that all scenes are registered in a consistent way.

After the scene, observed from the four different poses depicted in Figure 5.4, was aligned by hand, the construction of the occlusion frusta (see subsection 3.3.1) took place. To better illustrate the reconstruction process, the pipeline is showcased on the scene depicted in Figure 5.4b: first the raw 3D data was classified via a RANSAC based algorithm (see subsection 2.3.3) in larger planar surfaces. Afterwards Euclidean distance-based clustering was performed on the remaining points to generate possible object candidates. The results of these steps are displayed in Figure 5.5b. Main support plane and back plane are colored in cyan and

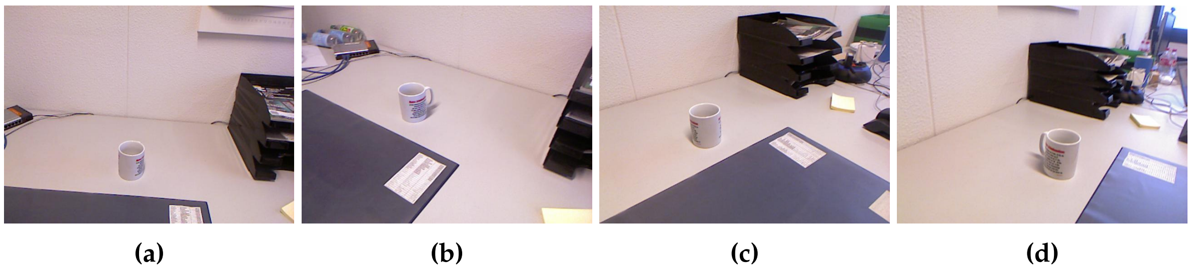


Figure 5.4: Viewpoints of the scenes used for the mug reconstruction example. Note that in **(b)** and **(d)** the handle is partially visible. However, only frame **(b)** contains valid tabletop measurements that are enclosed by the hole region.

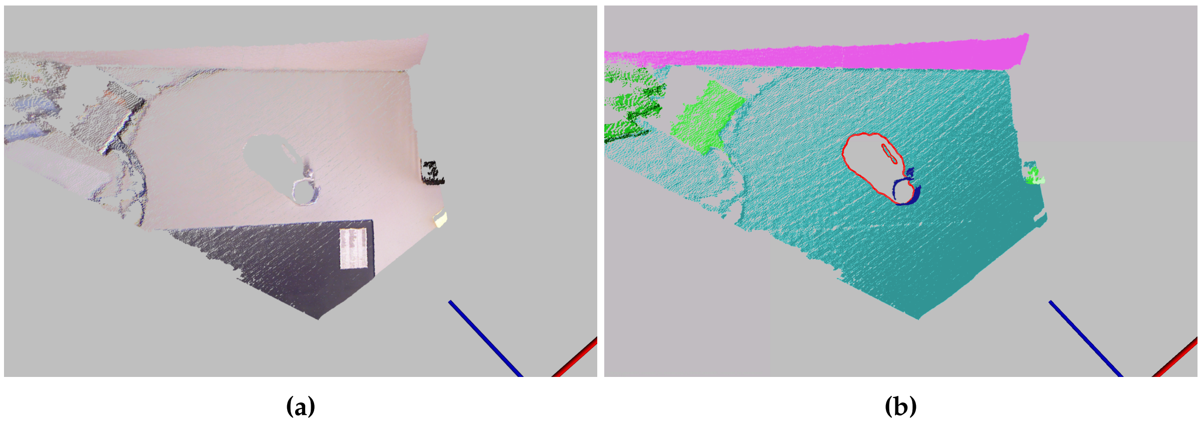


Figure 5.5: Top view of the point cloud obtained from view 5.4b: Left-hand side **(a)** shows the unclassified scene in color, while **(b)** displays a scene where planar surfaces are shown in cyan and magenta, respectively. Potential objects are shown in shades of green, while the mug is colored blue. Red points indicate the location of the polygon, representing the hole region.

magenta, respectively, extracted object clusters, apart from the mug are shown in green and the mug is colored in blue. The polygonal representation of the hole caused by the mug on the main support plane is indicated in red. Note that in this particular case the hole region has a "hole" of its own, i.e., a small embedded region of valid tabletop measurements.

Figure 5.6 displays, from two different points of view, part of the combined point cloud that was obtained by manual alignment of the scenes depicted in Figure 5.4. Frusta are color-coded, i.e., the blue frustum in Figure 5.6a corresponds to the blue one in Figure 5.6b and each color is associated with one of the views in Figure 5.4. As is evident in Figure 5.6a the height of the sensor, when recording the scene did not vary too much.

The results of the frusta intersection is compared in Figure 5.7 with the combined point clusters that were actually obtained from the mug. To compare these two point clouds is sensible, since the actual measurements of the mug (Figures 5.7a and 5.7b) are subject to the same alignment errors as the reconstruction and reflect the measurement noise that influences the

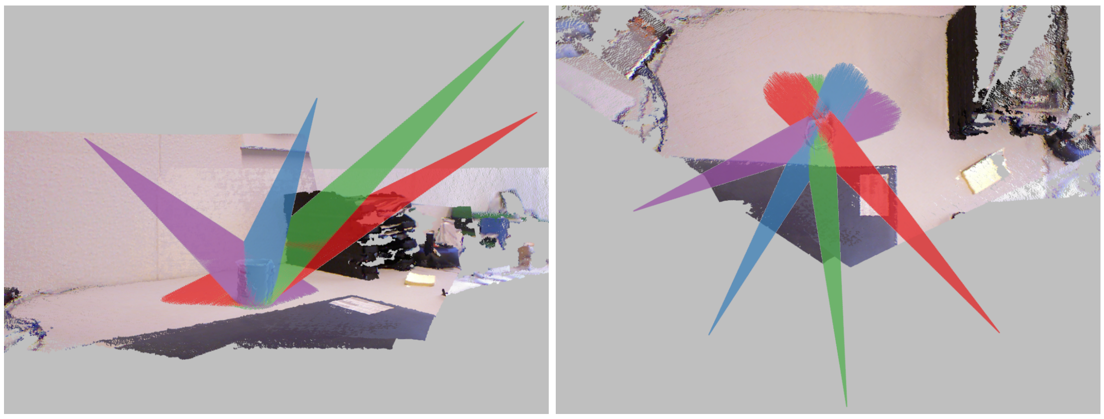


Figure 5.6: Visualization of occlusion frusta for solid object. **(a)** and **(b)** both depict a cut-out of the combined point clouds gathered from the scenes in Figure 5.4, only from different viewpoints. Occlusion frusta indicate sizes of detected holes and sensor positions in relation to the scene.

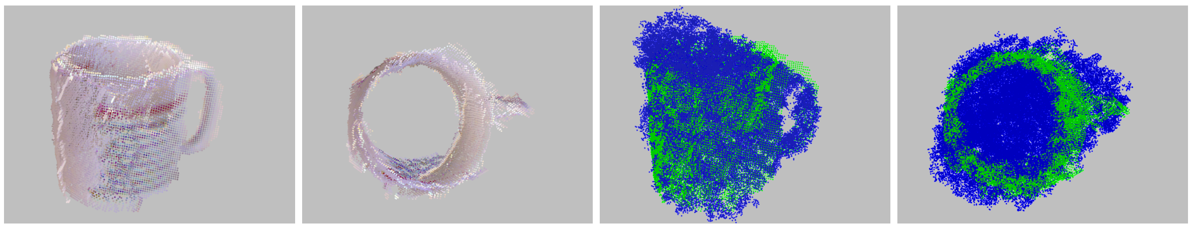


Figure 5.7: Comparison of reconstruction result with recorded data. Side view **(a)** and top view **(b)** of the acquired and aligned mug data. In **(c)** and **(d)** these points are depicted in green and displayed together with the result of the reconstruction (blue points).

occlusion caused by the mug.

While the reconstruction approximates the overall shape and volume of the measured mug pretty well, some room for improvement is evident in this first attempt: near the handle (upper right in Figure 5.7d) the reconstruction is less precise than on the front of the mug (lower left), where measurements and estimation only diverge slightly. If we instead consider the side view (Figure 5.7c) the reconstruction overestimates the height of the actual mug on the side opposed to its handle. These results are not too surprising, when we take the poses of the sensor during data acquisition into account. The overestimation of the height can be explained by the arrangement and position of the occlusion frusta at this volume in space, where all 4 frusta are also rising upward, towards the sensor positions they are associated with. If the scene would have been recorded from an additional viewpoint located roughly opposite of one of the existing poses, this problem would be addressed. The same holds true for the reconstruction near the mug's handle: one more occlusion frustum that provides a better estimation of the "footprint" of the mug on the table at this position would have solved this problem. Still one should acknowledge that the reconstruction is already reasonably good and even the hole in the handle is evident in Figure 5.7c.

The visualization for the reconstruction of the mug in Figures 5.7c and 5.7d is composed of all sample points that were located inside those octree leaves that were occupied with points from all 4 frusta. For the displayed data, each hole region was represented by a distribution of random point samples with a density of 10 sample points per $1\,\text{cm}^2$ and a leaf size of $l_L = 1\,\text{cm}$ was chosen. The random sampling is responsible for the non-uniform distribution and slightly rugged outline of the blue sample points that form the reconstruction.

Sampling with a higher density and choosing a size $l_L < 1\,\text{cm}$ improves the appearance of the reconstruction further, i.e., the estimated shape becomes less "blocky". This improvement in terms of reconstruction precision comes at the cost of creating more sample points. With the parameters employed in this experiment the point cloud that contains the sample points of all 4 occlusion frusta contains 1.752.256 points (as a comparison, the recorded scenes combine to a total of 1.021.174 points).

The amount of points that is needed to represent and compute the current estimation of a transparent / specular object becomes important, when the reconstruction becomes one of many applications in a larger robot control architecture, where its results need to be published frequently in order to communicate with other modules. Also note that increasing the density of the point sampling, without changing l_L will only change the visual appearance of the reconstruction in the sense that each octree leaf is filled with more sample points. Therefore, later implementation omits the random sampling of the hole region and works directly on the underlying octree. This will be shown in more detail in subsection 5.1.3 when reconstruction is performed with the viewpoint-based method.

Instead of basing the reconstruction on an octree that approximates the intersection by means of its leaves, i.e., small cubes, another representation naturally comes to mind. Since we have extracted a polygonal description of the outline of a hole region, a single occlusion frustum can easily be expresses by a polyhedron. Each line segment together with the sensor's position forms a triangle. Employing the OpenGL tesselator (see [71, chapter 11]) on the polygonal outline of the hole region, this results in a closed polyhedron representation of the occlusion frustum.

Hole regions in subsequently gathered frames are processed the same way, i.e., first the polygon describing the outline of the hole region is extracted and then used, with the sensor's position as the apex to define an enclosed polyhedron. In the aligned scene the intersection can now be performed on the different polyhedra and the reconstruction corresponds to the volume that is located on the inside of all polyhedra.

In order to compute what is inside of a polyhedron, the enclosed property plays an important role. Therefore, if we want to allow modelling enclosed regions of planar inliers inside the hole region, as for the mug presented in Figure 5.5b, the construction of the polyhedron has to be modified slightly: since the enclosed polygon, caused by valid measurements inside the hole region serves as a cut-out from the polyhedron, the sensor's position cannot be used to construct the triangles that describe this part of the polyhedron surface. Doing so would create a point, namely the apex, that belongs to the inside of the polyhedron, via the triangles of the outer polygon, as well as to the outside. In order to retain the enclosed property a new point,

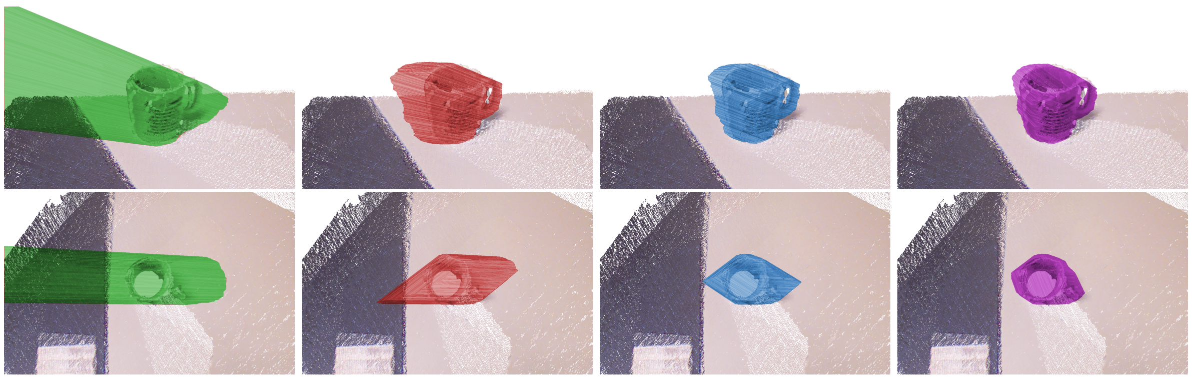


Figure 5.8: Progression of polyhedra intersection for mug reconstruction: Top row shows a side view, while bottom row displays a top down view of the scene. The result of the polyhedra intersection is superimposed on the combined point cloud in translucent color.

slightly below the sensor's position will be used to form the triangular faces of any enclosed polygon.

Figure 5.8 shows a visualization of the progression of the intersection process when incorporating the information of frames 1 – 4, depicted in Figure 5.4. The color of the reconstruction in Figure 5.8 correspond to the colors of the occlusion frusta in Figure 5.6 and indicate which occlusion frustum was used last in the sequential intersection process. The order of the intersections corresponds to the sequence of scenes as shown in Figure 5.4.

Although the intersection computation of the polyhedron representation of the occlusion frusta was performed in an efficient way with the help of the CGAL library [76], this turned out to get slow for the data presented in Figure 5.8. This behavior is caused by the initially already jagged outline of the polygons, describing the individual hole regions, which will result in an increasing complexity of the polyhedra that represent the current state of the intersection. A compact overview of this phenomenon is shown in Table 5.3. All values displayed in Table 5.3 are given in a consecutive manner, i.e., the column for the first frame displays the properties of the occlusion frustum for this frame, hence no time for the intersection computation is given. All other columns list the results of the intersection of the frustum of the specified frame with the previous input. The computation of the intersection of all 4 occlusion frusta takes on average a total of 138.245 s, i.e., the sum of the bottom column in Table 5.3, which would increase even more if more viewpoints were specified.

The computation of the intersection, when occlusion frusta are represented via polyhedra in this way, is not only time-consuming, but this fine grained modelling is also unsuitable for the rather inaccurate Kinect sensor. Since its measurements from the tabletop, which define the shape of the hole region polygon, are rather inaccurate, not modelling every miniscule detail seems sensible (notice that the polygon for the hole region in the first frame is composed from 199 vertices, resulting in an occlusion frustum represented by 200 vertices (polygon + apex)).

In order to reduce the computational load when computing the polyhedra intersection, the originally extracted polygons were simplified, employing the Douglas-Peucker-Algorithm [20] provided by the psimpl library [18], while still trying to retain their overall shape and infor-

# frame	1	2	3	4
# vertices	200	974	1703	1948
# faces	396	1948	3406	3896
intersection computation time	–	7.465 s	32.53 s	98.249 s

Table 5.3: Increasing complexity of polyhedron representation for the occlusion frusta intersection. Intersection times were averaged over 100 runs each.

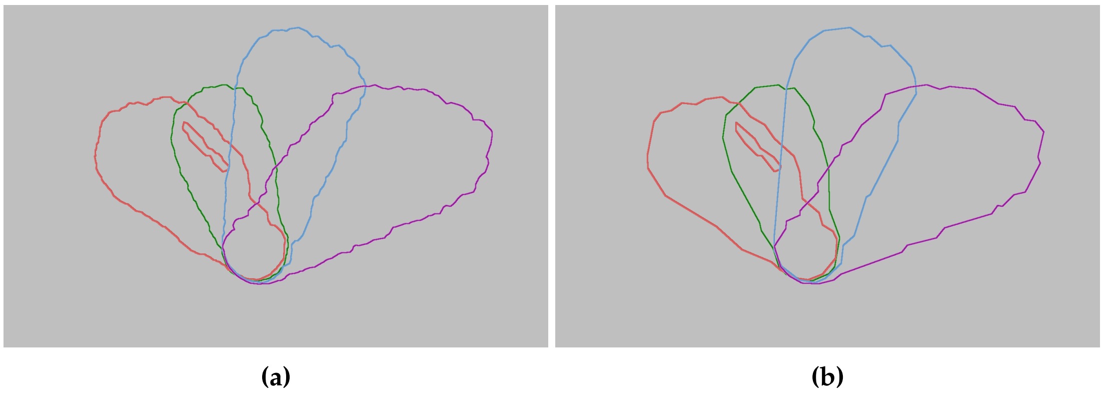


Figure 5.9: Comparison of original and simplified hole polygons: **(a)** shows the polygonal representation hole representation extracted from the outlines of the hole regions, while **(b)** displays their simplified counterparts. Colors of the polygons in **(a)** and **(b)** indicate their association to the occlusion frusta in Figure 5.6.

mation. The tolerance threshold for the Douglas-Peucker-Algorithm was set to 2.5 mm for the simplification of the original polygons. A comparison of the polygons which describe each hole region is depicted in Figure 5.9, where the non-simplified polygons (Figure 5.9a) are shown together with their simplifications (Figure 5.9b). In both cases the polygons are arranged according to scene alignment and shown in a top down view (keep in mind that the polygons consist of 3D points, since they are located on a 3D tabletop). While Figure 5.9 illustrates that the overall shape of the hole regions is still retained after simplifying the polygons, Table 5.4 summarizes their reduced complexity.

The less complex polygonal representation of the different hole regions in turn yields polyhedra that contain fewer and larger faces. However, not only is the simplified polygonal representation of the hole regions similar, but also the results of the intersection computation, when performed on the simplified polyhedra. Figure 5.10 provides a visualization of the reconstruction results, which differs only in very small details from the visualization of the non-simplified intersection in Figure 5.8. Associated Table 5.5 shows the resulting reduced complexity in terms of vertices, faces and computation time. With the average time needed for the computation of the three intersections down to 3.475 s from 138.245 s, this representation becomes more suit-

	# frame	1	2	3	4
Original hole polygons	# vertices	199	276	226	292
	# faces	197	276	224	290
	area	206.38 cm^2	278.59 cm^2	276.73 cm^2	335.17 cm^2
Simplified hole polygons	# vertices	24	43	25	42
	# faces	22	43	23	40
	area	205.09 cm^2	277.69 cm^2	276.86 cm^2	333.96 cm^2

Table 5.4: Differences in polygonal hole representation between original and simplified versions. Note that the decreased complexity only has a minor influence on the enclosed area and the shape of the outline (see Figure 5.9).

able for usage on a mobile robot.

However, notice that the increase in computation time for subsequent intersections is a trait of this representation: the more complex one of the polyhedra involved in the intersection operation becomes, the longer its computation will take. This is fundamentally different from the octree representation. Firstly, the intersection of more than two occlusion frusta does not need to be performed in pairs of two, but can be done in a single pass. If all sampled frusta are added into a single octree, then each leaf needs to be investigated only once in order to determine if it belongs to the intersection or not. On the other hand, if the intersection computation in the octree representation is performed in a sequential fashion, the computation in later stages actually becomes faster than in the earlier stages: since the intersection can at most be identical to the preceeding approximation in terms of octree leaves, only those leaves that were part of the previous intersection need to be investigated again, i.e., the number of leaves that need to checked to compute a new intersection stays either equal or decreases with each consecutive intersection computation.

Although the intersection computation based on the simplified polyhedra is a definite speed-up when compared to the non-simplified version, if compared with the octree representation it is still somewhat slower. The determination of the intersection as shown in Figure 5.7 from the 4 occlusion frusta that comprised a total 1.752.256 points took on average 0.038 s, averaging runtime over 100 runs as for the other reported times. If the time needed to create the octree representation of the point cloud is also included computation time for the octree-based intersection rises to an average of 0.682 s – still faster than the polyhedron based intersection, where the creation of the polyhedra in the appropriate CGAL data structures was not accounted for in the time measurements.

Reconstruction of Transparent Objects

Since the results for reconstruction of the solid mug looked promising, the same method was employed next to a series of actually transparent objects. In the following the results for two

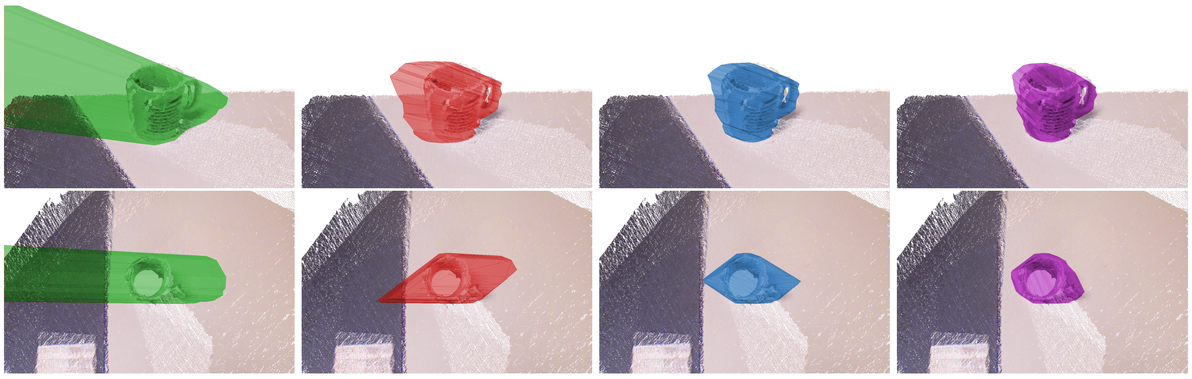


Figure 5.10: Progression of simplified polyhedra intersection for mug reconstruction: Top row shows a side view, while bottom row displays a top down view of the scene. The result of the polyhedra intersection is superimposed on the combined point cloud in translucent color.

# frame	1	2	3	4
# vertices	25	138	208	265
# faces	46	276	416	530
intersection computation time	–	0.278 s	0.959 s	2.238 s

Table 5.5: Increasing complexity of simplified polyhedron representation for the occlusion frusta intersection. Intersection times were averaged over 100 runs each.

example objects will be presented, namely, a glass and a French press that is mainly made out of glass, but features a non-transparent handle. For these two objects, the reconstruction was based on 6 different viewpoints, resulting in a more precise reconstruction of the transparent objects. The poses from where the point clouds were recorded are shown in Figures 5.11 and 5.12, respectively.

Naturally, for the transparent objects a comparison as depicted in Figure 5.7 for the reconstruction of the regular mug is not possible. However, the scenes with the glass and the French press both also feature a mug placed on the tabletop. Alongside with the available color information, these enable the viewer to get a qualitative notion about the size of the reconstructed objects.

The occlusion frusta that were generated for the glasses reconstruction can be seen in Figure 5.13: in 5.13a the polyhedron representations of the hole polygons are shown from a side view and Figure 5.13b displays the point sample counter part from a top view. The different colors are used to distinguish between the different frames and they are reused in Figure 5.13c to indicate correspondences between the polygons and the occlusion frusta. The polygons displayed in Figure 5.13c are already simplified, using the Douglas-Peucker-Algorithm with the same parametrization as before. However, when compared to the polygons in Figure 5.9b their outlines appear less smooth (the purple polygon in Figure 5.13c features some larger protru-

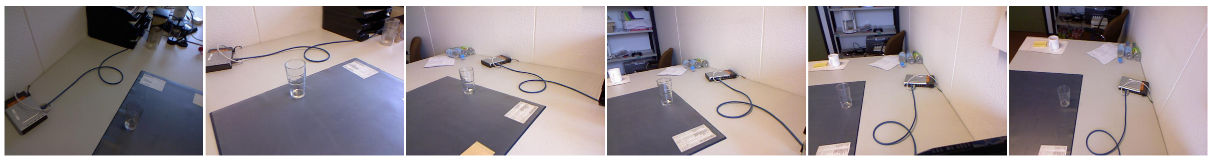


Figure 5.11: The six different viewpoints used to reconstruct the glass on top the table.

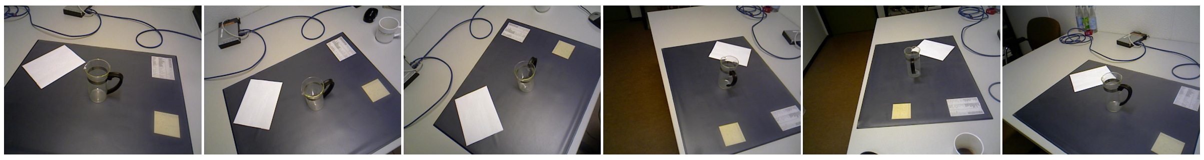


Figure 5.12: The six different viewpoints that were used to reconstruct the French press. Note that the French press also contains a non-transparent plastic handle, that needs to be taken into account, during reconstruction.

sions, for example). As additional information the area that lies inside all the hole polygons is marked in green. This information is interesting, since it corresponds with the footprint of the reconstructed object on the support plane. In the case of the drinking glass, this should ideally have the shape of a circle.

Transparent objects unfortunately feature a less precise and clear-cut occlusion on a support plane when compared to non-transparent counterparts, hence the less smooth appearance of the corresponding hole regions and their outlines. This property is partially reflected in the reconstruction results shown in Figure 5.14. Both images show the reconstruction result embedded into the aligned point clouds that were collected from the views shown in Figure 5.11. Visual comparison of location and size of the reconstruction in Figure 5.14 fit to the scene impressions that can be obtained from the photographs of the scene. As a reference object the mug present in the scene is highlighted by a cyan circle in both Figures 5.14a and 5.14b. Although

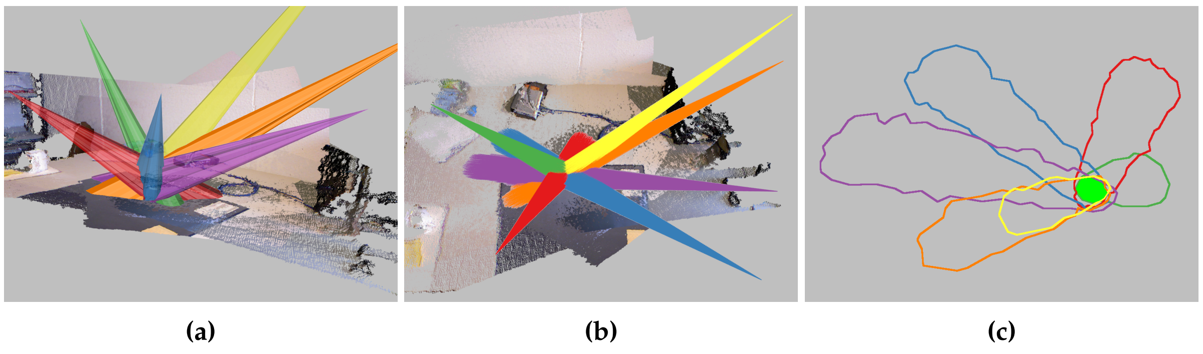


Figure 5.13: Intersection reconstruction result for drinking glass dataset shown in Figure 5.11: **(a)** and **(b)** show the occlusion frusta while **(c)** visualizes the simplified polygons of each hole region.

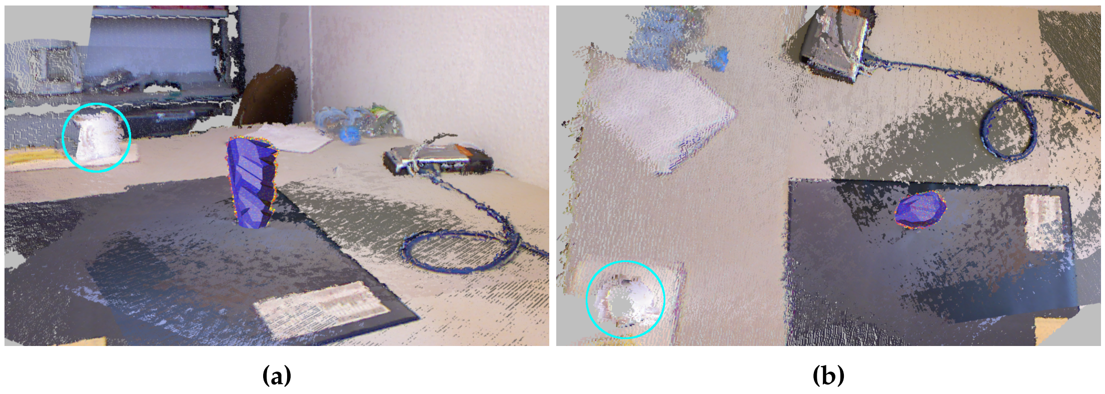
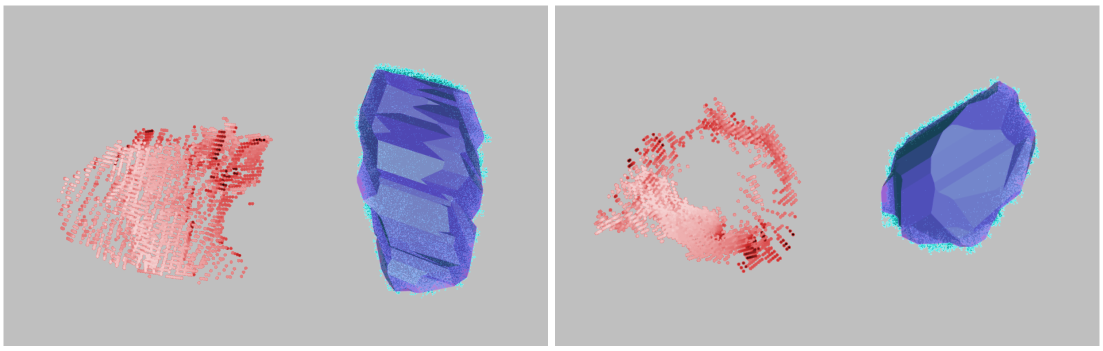


Figure 5.14: Reconstruction results for drinking glass embedded into the scene. The intersection result of the octree representation is shown as a dense point cloud, composed of points in the same colors as the occlusion frusta in Figure 5.13. The result of the polyhedra intersection is overlaid in a semi-transparent fashion in blue. **(a)** provides a perspective view, while **(b)** provides a top view.

Figure 5.14 contains both, reconstruction by octree volumes and the polyhedra intersection, only the latter one is clearly discernible, due to their strong similarity.

For a better comparison of the glass's size and shape, Figure 5.15 provides a side by side view of the reconstructed glass and the measurements obtained from the mug's surface in the 6 frames, so that both objects are displayed in the same scale (note that the mug in Figure 5.14a is placed in the background and hence displayed on a smaller scale than the drinking glass, due to the perspective projection). To increase visibility of the originally beige mug, the corresponding points are visualized in red. The point cloud modelling the transparent object obtained via the octree method is colored in cyan, while the polyhedron intersection is displayed in magenta in a semi-transparent fashion, thus resulting in a blue color where both reconstruction methods match. Note that the points measurements of the mug contain already quite some measurement noise at distances between approximately 1.32 m and 1.46 m from the kinect sensor. Furthermore, note that the mug in the side view shown in Figure 5.15a is in reality slightly higher than displayed in the Figure, but some of its points were classified as tabletop during the RANSAC plane extraction. Comparing the result of the reconstruction alongside the result of a simple cluster fusion for the non-transparent mug shows that the reconstruction provides satisfying results, given the inherent noise of the employed sensor.

Table 5.6 presents a numerical overview of the intersection progression. It shows, the general trend that the higher the number of vertices and faces of the polyhedra that are involved in the intersection computation, the longer the intersection operation takes. The last column however demonstrates that this is not an automatic correlation, but also depends on the shape of the involved polyhedra, since the intersection of the 6th frame with the prior intersection takes much less time than the two previous computations. Added up, the whole intersection process for the drinking glass takes on average a total of 7.849 s. The same operation, on the point cloud sample based method takes on average 1.142 s, including octree creation. In the case of the drinking glass reconstruction the point cloud modelling all occlusion frusta is made



(a) **(b)**

Figure 5.15: Comparison of glass reconstruction with registered point clusters of in-scene mug. **(a)** shows a comparison from the side and **(b)** displays a top-down view.

# frame	1	2	3	4	5	6
			Individual frames			
# vertices	18	44	39	50	32	18
# faces	32	84	74	96	60	32
			Intersection progress			
# vertices	18	119	174	226	265	269
# faces	32	234	344	448	526	534
computation time	–	0.206 s	1.471 s	2.492 s	2.294 s	1.386 s

Table 5.6: Complexity and runtime progression of drinking glass reconstruction. Intersection times were averaged over 100 runs each.

up of 3.135.441 points, where the occlusion frusta that were created from less elevated sensor positions (blue and violet in Figure 5.13) are responsible for more than half of the sample points, due to their large hole regions (Figure 5.13c).

As a second example of a transparent object reconstruction, using the intersection of occlusion frusta, the results for the French press (see Figure 5.12 for photographs from the 6 poses that are used for reconstruction) are shown in a similar manner: a visualization of the occlusion frusta as polyhedra is provided in Figure 5.16a, while a representation as a labeled dense point cloud from above is displayed in Figure 5.16b. The outlines of the detected hole regions that serve as a basis for both representations are displayed in Figure 5.16c.

When comparing these with their counterparts of the glass (Figure 5.13c) or the regular mug (Figure 5.9), their outlines are much less smooth, being distinctly concave in some cases (yellow and green outline). To a certain part, this concavity is to be expected, since the handle of the

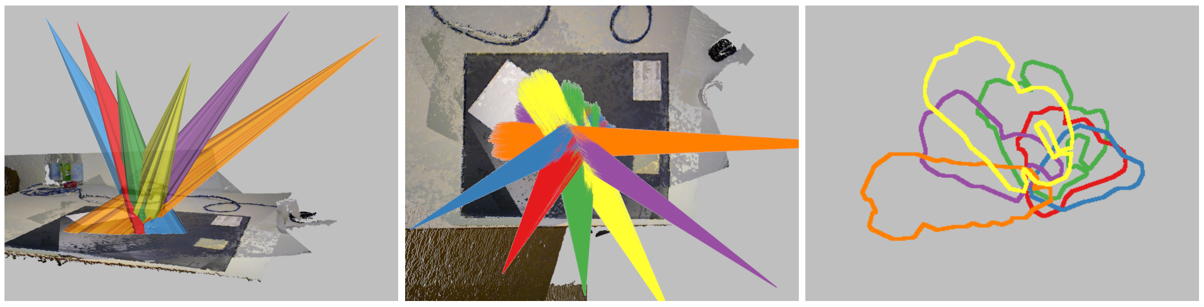


Figure 5.16: Intersection reconstruction result for French press data shown in Figure 5.12: **(a)** and **(b)** show the occlusion frusta while **(c)** visualizes the simplified polygons of each hole region.

French press should, from certain viewpoints, create a concave occlusion on a support surface. However, it can also be observed that the polygons in Figure 5.16c do not provide an obvious part that belongs into the intersection as in both previous examples. When observing the hole polygons for both, mug and drinking glass, each polygon contains a rounded portion at the section that was located in the direction of the sensor. These result in a more or less circular portion that lies inside of all hole polygons and roughly resembles the footprint of the objects on the tabletop. The polygons for the French press do not contain such a clear cut region, but near the footprint's location the outlines are jumbled and intersecting each other.

This property of the hole polygons is unfortunately a direct consequence of the recorded point clouds, where the sensor sometimes was able to measure the tabletop through the glass surface of the French press near its footprint location. Examples for such rather "unexpected" measurement points, when dealing with transparent objects, are displayed in Figure 5.17, where they are highlighted in cyan. Unfortunately, these measurements do not diverge to a large extent from regular measurements of the tabletop surface, i.e., filtering methods on geometric properties have only slim chances to succeed: decreasing the distance threshold for RANSAC plane fitting will remove some of these points, but in turn result in additional holes in the tabletop, since regular measurements in some regions of the tabletop diverge to a comparable extent from the parametrized plane. Analyzing the contents of each grid cell of the occupancy grid and checking and comparing features like the mean distance or standard deviation from all points in the cell to the plane, also did not yield results that removed a noticeable portion of these points without removing valid measurements as well.

Basing the reconstruction from the intersection of the occlusion frusta as presented in Figure 5.16 will naturally result in an estimation exhibiting a smaller and slightly deformed footprint, when compared with the actual object. An in-scene display of the reconstruction is shown in Figure 5.18, while a side-by-side comparison with the measurements collected from the regular mug that was present in the scenes is shown in Figure 5.19. The color schemes for the reconstruction objects is the same as used in Figures 5.14 and 5.15.

The results provided in Figure 5.18 show that while the shape of the reconstruction does not completely correspond to the desired result, i.e., a roughly cylindrical representation for

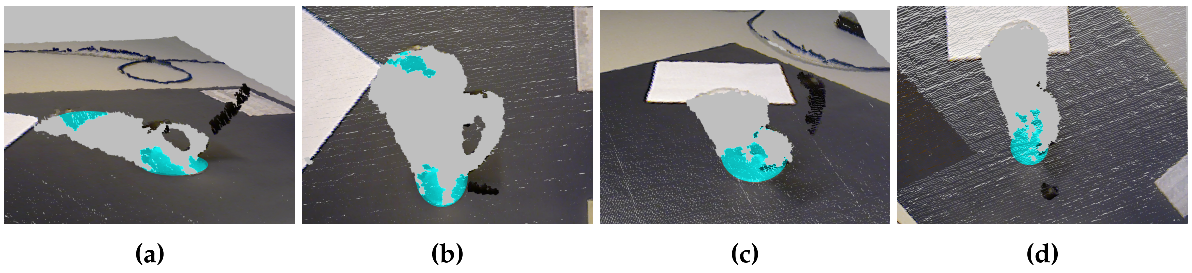


Figure 5.17: Unexpected measurements in French press dataset: **(a)** and **(b)** show the points cloud from the first frame (green occlusion frustum) of the French press dataset; **(c)** and **(d)** display the point cloud of the fourth frame (violet). Highlighted points belong to the tabletop and are located behind the French press made out of glass.

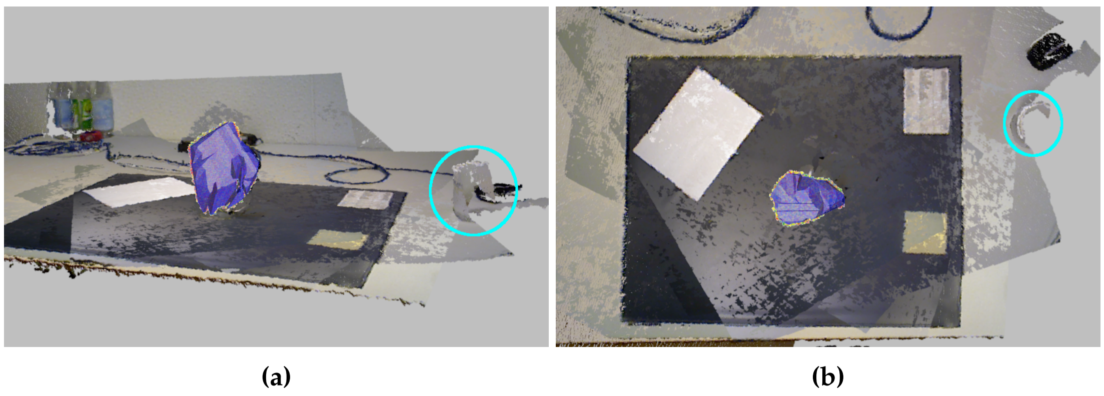
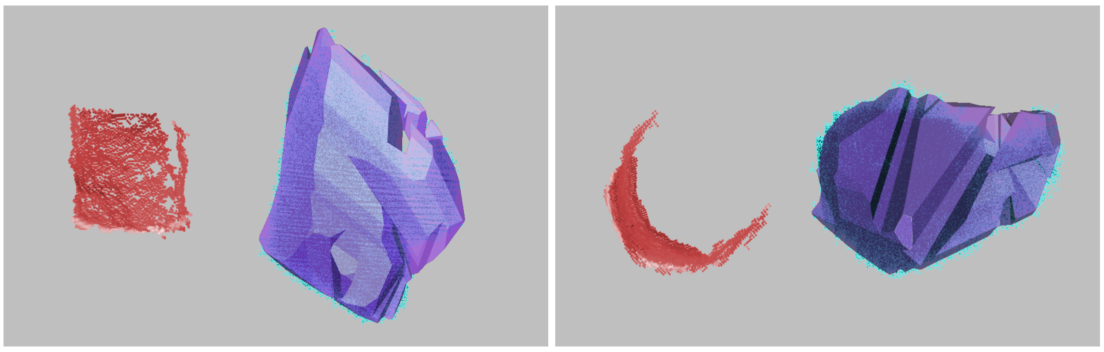


Figure 5.18: Reconstruction results for French press embedded into the scene. The intersection result of the octree representation is shown as a dense point cloud, composed of points in the same colors as the occlusion frusta in Figure 5.16. Result of the polyhedra intersection is overlaid in a semi-transparent fashion in blue. **(a)** provides a perspective view, while **(b)** provides a top down view.

the French press with an attached handle, the general size estimation is close. When observed from the top (Figure 5.18b and 5.19b), the round outline of the glass component can be seen, however the side views reveal that this is not a cylindrical shape. Also the height of the French press seems to be overestimated; probably an additional view of the scene that observed it from a side perspective might have improved the reconstruction in this area, since the corresponding occlusion frusta would have cut off some parts of the top.

Even though the quality of the reconstruction of the French press is worse than the result obtained for the drinking glass, it is sufficient for avoiding this object during manipulation tasks on a tabletop. If we compare it with the actual measurement points of the mug in Figure 5.15a that should have been arranged in a cylindrical fashion, the bad quality of the reconstruction result is put into perspective again.

Looking at the progression of the polyhedra intersection for the French press reconstruction in Table 5.7, we can see that the overall computation takes longer, when compared with



(a) (b)

Figure 5.19: Comparison of French press reconstruction with registered point clusters of in-scene mug. **(a)** shows a comparison from the side and **(b)** displays a top-down view.

# frame	1	2	3	4	5	6
			Individual frames			
# vertices	40	27	18	32	41	39
# faces	76	50	32	60	78	74
			Intersection progress			
# vertices	40	154	193	215	271	347
# faces	76	304	382	426	538	686
computation time	–	0.275 s	0.908 s	1.792 s	3.778 s	3.154 s

Table 5.7: Complexity and runtime progression of French press reconstruction. Intersection times were averaged over 100 runs each.

the drinking glass (9.907 s for French press and 7.849 s for the drinking glass on average). While the sum of vertices and faces for all 6 individual polyhedra does not differ much between the two scenarios (201 vertices for the drinking glass frusta compared to 195 vertices for the French press and 378 faces compared to 370, respectively), the complexity of the final intersection differs considerably: while the drinking glass is represented by 269 vertices that define 534 faces, the French press is modeled from 347 vertices that constitute 686 faces. From these numbers we can deduct, unsurprisingly, that the complexity of the intersection operation, when modelled via polyhedra, is not a direct function of the number of vertices and faces of the involved polyhedra, but also needs to take into account the shape of the individual polyhedra and their alignments with respect to each other.

The increased cost, when computing the intersection-based on the polyhedra representation is not reflected, when the occlusion frusta are modelled via sample points and an octree filtering is used to determine the intersection. In the latter case the intersection operation, including the creation of the octree, took place in 0.68916 s on average (drinking glass 1.142 s). All occlusion frusta were represented by a total of 1.811.339 sample points, where the drinking glass was represented by 3.135.441.

The smaller number of points needed to represent the occlusion frusta for the French press is due to that the latter was recorded from slightly more elevated poses that did not create such a large footprint on the tabletop, hence creating frusta that do not encompass such a large volume as in the case of the drinking glass. This property is directly reflected in the amount of leaves that the octree contains in each case. For the drinking glass the octree was composed of a total of 156.068 leaves, while the sample points of the French press's occlusion frusta occupied 110.511 leaves (leave size l_L= 0.5 cm in both cases).

Since the intersection operation for the sample point based representation boils down to iterating once over all leaves, the shape of an individual frustum does not change the complexity of the investigation, as long as the frustum encompasses the same volume. Note however that the shape of the frusta in combination with their alignment has an influence, since this determines the number of leaves in the resulting octree: if we add a frustum, albeit with different labels, into an initial octree structure that contains just this frustum, the number of occupied leaves can stay the same (if the second frusta are added in the same pose as the first), can be doubled (if there is no overlap between both frusta) or can be some number in between (if both frusta partially overlap).

Problems of Reconstruction via Intersection

Provided a precise alignment of the individual frames and an occlusion on the support plane that closely reflects the appearance of the transparent object that is to be reconstructed, intersection of the occlusion frusta offers a feasible solution. Unfortunately, these requirements are not always satisfied.

Figure 5.20 provides an example of the influence of the scene alignment. The displayed reconstruction is the result of applying some artificial noise to the alignments used for the mug reconstruction shown earlier (Figure 5.7). The final alignments were subject to random noise where each translation (x, y and z) was modified by a Gaussian normal distribution with parameters $\mu = 0$ and $\sigma = 1.5$ cm. Orientation noise was simulated by a rotation using Euler angles with a second Gaussian ($\mu = 0, \sigma = 1\,^{\circ}$) for each angle. Figure 5.20a shows the occlusion frusta embedded in the scene, from the same point of view used in Figure 5.6a.

Due to the small amount of noise that was applied, both images appear very similar to a human observer, but differences become more evident, when we consider the result of the intersection operation: Figures 5.20b and 5.20c show the equivalents to Figures 5.7c and 5.7d, with identical color coding and from the same points of view. The most obvious differences between the two results is the change in location of the noise-induced reconstruction, which does not overlap with the left part of the mug's actual position, when observed in Figure 5.20c. Unsurprisingly, the overall shape approximation also suffered under the influence of noise.

As was already evident in the reconstruction of the French press measurements of the support surface that exist via "measuring through" the transparent object also constitute a problem, when approximating size and shape of the latter. To further illustrate this claim, let us

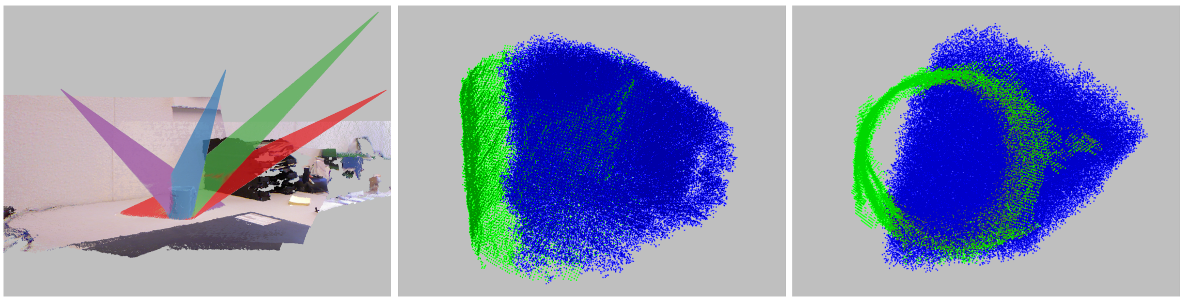

(a) (b) (c)

Figure 5.20: Influence of alignment on intersection reconstruction: **(a)** visualizes the slightly misaligned occlusion frusta while **(b)** and **(c)** show the reconstruction results for the solid mug in comparison with the direct measurements in the acquired point clouds.

take a look at the result of the drinking glass reconstruction, where we incorporate the information of two additional frames. Gaining more knowledge, i.e., recording more point clouds of the scene, should enable us to reconstruct the shape of an object more precisely, provided that scenes are aligned correctly. However, this intuition is only correct, if the gathered data behaves "as expected", which means in this case that there are no measurement points inside a hole region, if we momentarily ignore special cases like handles observed from certain angles.

A counterexample is visualized in Figure 5.21, where a reconstruction for the drinking glass is presented that incorporated information from two additional frames. Reconstructions in Figures 5.21a and 5.21b are displayed from the same viewpoints as used for the comparison with the solid mug, shown in Figure 5.15, and the reconstruction displayed in cyan is identical to the polyhedron representations presented there. Naturally, an intersection with additional occlusion frusta results in a reconstruction that is either identical with the previous reconstruction or encompasses a smaller volume. This property is evident in the new reconstruction (magenta in Figure 5.21), however the rather prominent indentation in the new reconstruction and the missing corner at the object's footprint are rather disappointing, since they do not reflect the true shape of the glass one would assume from the photographs (Figure 5.11).

When observing the point clouds that were recorded from the 2 additional poses, the cause for the indentation and the strongly tapered footprint become more obvious. The first additional frame (Figure 5.22a) contains some unexpected measurements, near the upper left corner of the occlusion, which the glass left on the support surface and which are responsible for the indentation in both, hole region and reconstruction. In the second additional frame, shown in Figure 5.22b, also unexpected measurements occur, but this time near the footprint of the drinking glass, effectively cutting of part of the glass's base. The resultant polygonal representations together with those of the other 6 frames are displayed in Figure 5.22c. Comparison with Figure 5.13c shows the decrease in the area that is shared by all polygons (marked in bright green), thus resulting in the tapered appearance of the reconstructed glass, near the support plane. In addition, the shape of the shared area deviates more from the ideal shape of a circle than the area without the 2 additional frames in Figure 5.13c

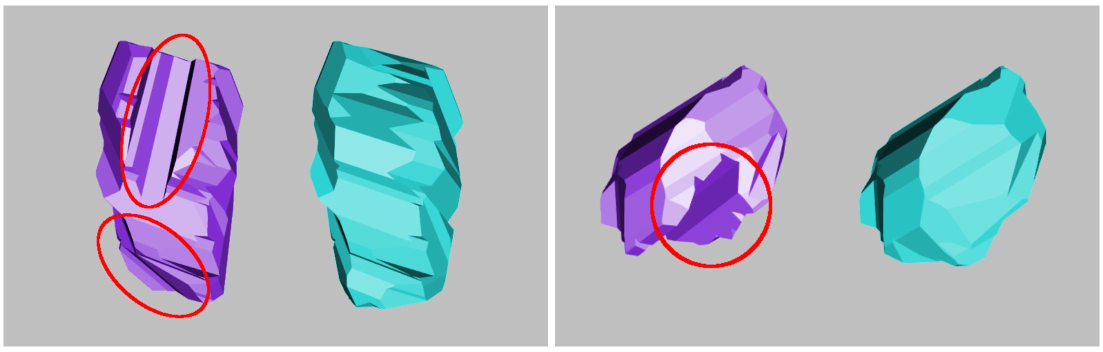


Figure 5.21: Impact of unexpected measurements on reconstruction of drinking glass: cyan shows intersection from 6 frames, while magenta polyhedron incorporates 2 additional frames. **(a)** and **(b)** show both reconstructions from the side and top, respectively. Red circles indicate regions of the reconstruction that poorly model the object's actual shape.

The results of these two experiments (sensitivity to imperfect alignment and measurement noise near the occluded areas on the support plane) show that the intersection-based reconstruction contains some issues that should be addressed in order to increase robustness and performance on mobile robotic platforms and similar scenarios, where precise scene alignment and crisp occlusion outlines that reflect the shape of the object cannot be guaranteed at all times. In order to address these problems, I proposed a reconstruction based on the difference in viewpoints of the collected frames (see subsection 3.3.3), and its results will be presented next.

5.1.3 Viewpoint-Based Reconstruction

Before presenting the results of the experiments conducted by employing VPBR (cf. 3.3.3) some remarks concerning the setup of the conducted experiments are in order. When experimenting with the intersection-based reconstruction, two alternative representations of the intersection volume were used, namely, a reconstruction made from a dense point sampling of each occlusion frustum and subsequent octree filtering and a reconstruction based on the intersection of polyhedra. For the viewpoint-based reconstruction a modified version of the point sample method is used for the following reasons:

As the results of the previous subsection suggest, the results of both reconstruction methods differ mainly in miniscule details. While the appearance of the octree-based approach tends to be "blocky" in its appearance, due to the octree leaves being its building blocks, the polyhedra intersection also results in a rather ragged surface representation, in consequence of the usually ragged outline of the hole regions in the support plane. This becomes evident in particular when considering Figures 5.15 and 5.19 and comparing the appearance of the octree-based reconstruction (cyan points) with its polyhedron counterpart (overlay in magenta).

The rough time measurements conducted on both reconstruction methods (see Tables 5.5,

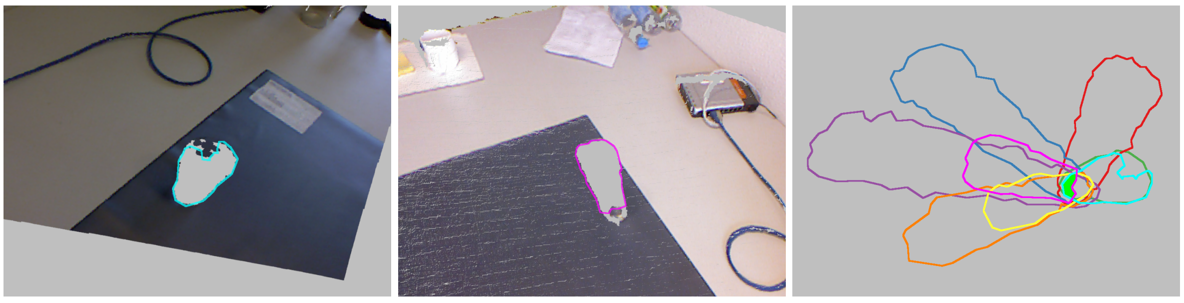


Figure 5.22: Problematic frames for intersection-based reconstruction: **(a)** and **(b)** show "unexpected measurements" of the support surface behind the transparent object and the deformations they cause on the associated hole regions. **(c)** displays all hole polygons and their shared area.

5.6, 5.7 and accompanying text) suggest that the polyhedron intersection is the more time consuming of the two operations – although this is no real complexity analysis of both approaches, by any means. Since overall runtime of both approaches will generally increase as more frames are incorporated into the reconstruction, the octree representation seems to have an edge in this respect.

However, this is not the main reason for dropping the polyhedron representation, when considering VPBR. If you recall the idea behind VPBR as presented in 3.3.3, the general idea is to partition space and determine for each volume whether it was occluded from a sufficient number of different viewpoints to consider it part of a transparent object. While the octree lends itself to model this general idea in a natural way, the polyhedron is no suitable representation to model this approach.

Accomplishing something similar in a representation employing polyhedra would involve determining all possible combinations of recorded frames, that according to the associated viewpoint intervals are sufficient to reconstruct a transparent object. Of all these combinations the intersections would need to be computed and subsequently all resulting volumes would need to be unified. Just considering the amount of possible subsets makes this idea rather infeasible.

As an example, think of 10 recorded frames, where (as a simplification) the intersection of any combination of 5 frames (or more) is considered to be observed from diverse enough viewpoints to be part of a transparent object. In this example, the number of intersections that would be needed to compute is equal to $\binom{10}{5} = 252$. Assuming in this example that the computation of one of these takes about 5 s each (intersection for 6 frames in the previously presented experiments took 7.849 s and 9.907 s, respectively), the computation of the potentially relevant intersections would sum up to $1260\,\text{s} = 21\,\text{m}$, which is rather impractical. As additional overhead, the computation of the union of these intersections would also needed to be considered.

Hence, the viewpoint-based reconstruction is based on creating a point sample representation of the occlusion frusta, that are added into a given octree. Since the implementation of VPBR is done in ROS [59], where frequent inter-node communication is desirable, represent-

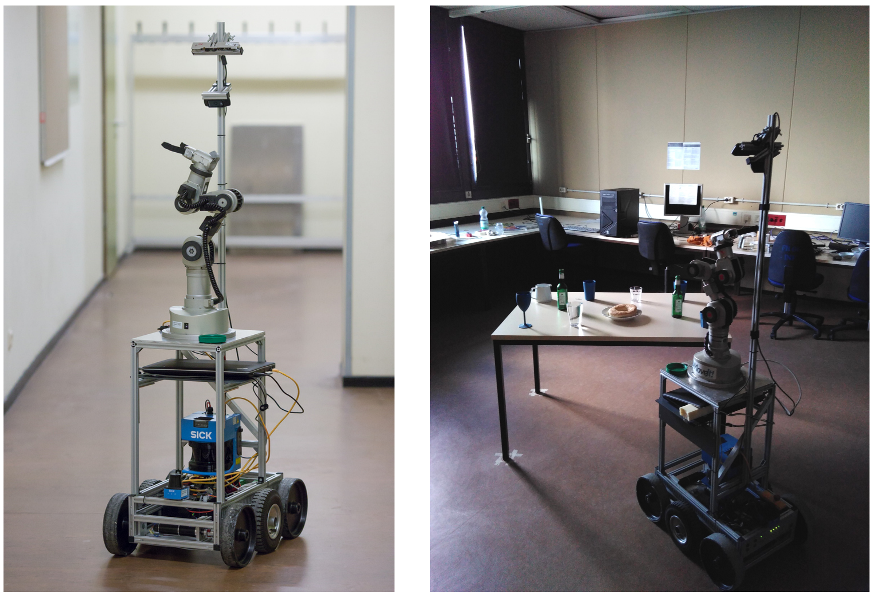


Figure 5.23: Robot platform Calvin: **(a)** shows the robot exploring a corridor and **(b)** displays it during one of the runs for viewpoint-based reconstruction. Image **(a)** used with courtesy of Michael Görner (`http://wiki.ros.org/calvin_robot`)

ing an occlusion frustum as a dense point sample is infeasible. Since the intersection employs the octree leaves as their building blocks, i.e., the side dimension l_L of a leaf basically determines the possible precision of the reconstruction, the number of sample points can be reduced accordingly, so that occupancy of an octree leaf for a recorded frame is denoted by a single point inside the leaf. This way, a compact, in terms of storage, representation of the occlusion frusta is provided that can be easily published to other nodes, that might integrate this information. In case that a polyhedron representation is needed, the 3D convex hull of the leaves that approximate the transparent object's shape, is provided as an additional output.

Since the main motivation behind the VPBR method was increased robustness that would allow for use on a mobile robot, consequently the experiments were conducted on such a platform, namely, the mobile robot "Calvin" of the Knowledge-Based Systems research group of Osnabrück University (Figure 5.23). As can be seen from the image, an Asus X-tion sensor is mounted at a height of approximately 1.75 m to enable observing a tabletop and objects placed on it. The sensor mounting is fixed, i.e., the pitch angle cannot be changed and the sensor elevation is also not variable.

The basic setup for the reconstruction experiments were adopted from Michael Görner's Master Thesis [31], where (regular) objects, placed on a tabletop were detected from multiple viewpoints and their properties learned for re-identification. Object detection was performed by observing a table from different viewpoints and fusing the information acquired in the individual frames. In case of a mobile platform like Calvin, where the employed sensor is mounted in a fixed position, generating several different viewpoints in practice means choosing several positions that are located around the table and observing the tabletop from these poses. In each recorded frame, the tabletop is detected, via RANSAC plane fitting and Euclidean clustering is performed on those points that are located above the convex hull of the detected tabletop.

For aligning subsequent frames, the robot's localization in a previously generated 2D map of the environment serves as an initial pose. These initial poses are used with a variant of ICP, that assumes that the robot's orientation can only change around its yaw axis, since it moves on a horizontal plane. After clusters are extracted in these frames, their information is fused with the already existing object clusters. Note that due to the setup of the Calvin platform, the tabletop is oftentimes only partially recorded, so that the number of objects on a table may increase (when a previously unseen object appears in the field of view of the latest acquired frame) or decrease, for instance, if two distinct object clusters that were located at the edges of the field of view are fused by newly acquired point information.

Adding the viewpoint-based reconstruction of transparent objects, as described in subsection 3.3.3, to this process proves convenient in ROS, since both methods share their initial steps: the extracted points of the detected tabletop along with their convex hull, the plane equation and the sensor's position are a requisite for the robust detection of hole regions and a ROS node that performs these steps simply needs to subscribe to an already existing node inside the existing table exploration of Michael Görner [31]. Since the provided point cloud input is organized and `NaN`-points are retained, hole detection proceeds, as detailed in 3.2.2. The frame alignment needed for the coherent aggregation of objects points in context of the Master thesis is also needed for the reconstruction of the transparent objects and, provided a valid registration, the differences between the viewpoint of two acquired frames can also be computed. Of course, correct alignment of acquired frames was also a requirement in the previous experiments, detailed in 5.1.2, but, as it was described there, it was performed manually for those experiments.

The results presented next were conducted together with Michael Görner, and the following general experimental setup was chosen: one transparent object was placed onto the table along a collection of five non-transparent objects. For one such given configuration the reconstruction was performed, i.e., the robot drove once around the table, observed it from different points of view and VPBR was performed. Before the next run, the objects on the table were exchanged, in particular the transparent object. Drinking glasses of different types, so that they provided variety in terms of shape and size, were used as transparent objects. As regular objects, identical glasses were painted with a blue chalk-based color, so that they lost their transparency and could be measured directly.

For each table, set in the described fashion, the robot drives around once and stops at a total of 12 different positions, roughly placed along an ellipsoid around the table and records it from the current position, i.e., a total of 12 frames are registered. With the viewpoint restrictions imposed by the sensor, the occlusion caused by the glass was completely visible in an average of 10 frames. Unsurprisingly, the `NaN`-regions of higher glasses, for instance the Champagne flute, were more often not completely located inside the convex table hull, so that less information for their reconstruction was available (e.g. 7 frames in case of the Champagne flute). A `NaN`-region had to consist of at least 15 connected pixels (von Neumann neighborhood) in order to be a hypothesis for a transparent object.

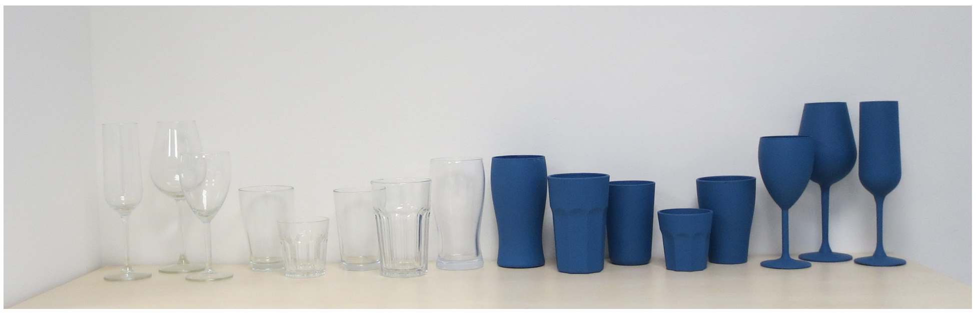


Figure 5.24: Collection of drinking glasses used in viewpoint-based reconstruction; painted versions included for better visibility. For later referencing each glass is assigned a unique identifier. Starting with the painted beer glass in the center, glasses are denoted from left to right as `glass1` – `glass8`.

For simplicity viewpoint angles ψ were rounded to the next integer value in $[0\,°, 360\,°)$ and parameter w_{vp} was set to $10\,°$, thus resulting in individual viewpoint intervals of width $21\,°$. Threshold parameter I_{min} was set to $120\,°$, i.e., at least the information of 6 views, where the viewpoint intervals do not overlap are needed in order to assume a transparent object at any location (see equation (3.6)). While this is a rather conservative approach, it reliably prevents the detection of false positives and proves flexible enough to allow for the reconstructed object to not be observed or fully visible in several frames. In the subsequent median filtering step to refine the approximated volumes the factor f_{μ_I} was set to 1, i.e., all leaves where the sum of the accumulated viewpoint intervals is below the median for that particular cluster are removed in the refined approximation.

Taking a less conservative approach for the estimation of transparent objects, i.e., increasing the width w_{vp} of single intervals while retaining I_{min} will result in a quicker appearance of estimated transparent objects, however at some cost: the likelihood of false positive detections also rises and the overestimation of the reconstructed convex hull in general increases. The detection of false positives could be prevented by assigning minimal dimensions for valid transparent object clusters, since most of such false positives are caused by small `NaN`-regions near or at the table edges or by stray highlights in certain viewpoints. The latter disadvantage can be lessened by experimenting with larger values for factor f_{μ_I}, so that the size of the established clusters is reduced in a post-processing step.

Figure 5.24 shows the different drinking glasses and their painted counterparts. One of the five non-transparent objects in each scene corresponded to the transparent object that was placed on the tabletop We believe that this last part of the setup provides us with a meaningful way to evaluate the performance of VPBR.

The aggregated point clouds of the painted version of one of the transparent objects is subject to the same systematic measurement noise of the sensor as well as to possibly occurring alignment errors. While the glass reconstructed from the directly measured points does not constitute ground truth, a comparison with it will yield insight, since it can be considered as a best case scenario, what the employed sensor (Asus X-tion) in its setup on the robot platform and the given scene alignment can achieve. However, both reconstructions, transparent and directly measured object, include their individual noises, so if we compare both to each other and see how well the viewpoint-based reconstruction fits to it opaque equivalent, the former will be punished for not emulating noise present in the latter.

Therefore, it would be nice to have some form of "ground truth" that can be used for the comparison with both reconstructions. Unfortunately CAD models are not directly available from the manufacturer of the glasses. However, using a Riegl VZ-400i terrestrial laser scanner, the painted version of the glass objects can be measured much more accurately than with a RGB-D camera. After manually removing measurement outliers, the data obtained by the laser scanner is used to create a precise point cloud representation for each of the glasses shown in Figure 5.24. While these models still do not constitute real ground truth they are treated as such, in the remainder of this experiment. In this role, these "ground truth" models will be used in the following to evaluate and relate both reconstructions, VPBR for the transparent version and the ICP based alignment (for details please refer to [31]) for their painted equivalents. For convenience the quotation marks around "ground truth" will be omitted in the following, even though I am well aware that these models are not real ground truth.

For this comparison, all instances of a given glass, i.e., ground truth, viewpoint-based reconstruction and directly obtained point cloud are aligned together in a common reference frame. This alignment is done in a semi-automatic fashion: the orientation of both reconstructions is the same, since they were created in a common coordinate system and placed on the same planar surface (the tabletop) and the ground truth object was created to fit to this orientation. Therefore, only translations are needed for alignment. As a first step, all instances are translated in such a fashion that their center of gravity coincides with origin of the common coordinate system. Afterwards the reconstructions are translated along the height axis to fit the ground truth data as good as possible, when observed from the side. Following this line of thought, VPBR is aligned at the bottom with the ground truth, since for all glasses without a stem this is the area that meets the tabletop and hence should correspond to the footprint of the ground truth, provided a good reconstruction. The reconstructions created from the painted glasses are aligned at the top with the ground truth object. This is done as they are created from point clusters after the tabletop extraction, which may result in missing points near the footprint of the glass, that were attributed to the planar surface of the tabletop. Note that the alignment of the reconstructions with the ground truth can be done this way, since all glasses are rotationally symmetrical around their height axis – otherwise at least one orientation parameter would have to be taken into consideration, when aligning reconstructions with ground truth.

This manual alignment is afterwards refined once more, using the fine registration tool in CloudCompare [27], the software that is used for comparing both reconstructions with the ground truth data. In the fine registration step, no translation along the height axis of the reconstructed glasses is allowed, since this constitutes external knowledge about the alignments. For obtaining minimal average errors, as is the aim of the fine registration, translations along this axis would usually happen without this constraint in place.

Finding a meaningful, quantitative and systematic way to evaluate the results of the reconstructions unfortunately turns out to be a hard problem: comparing the volume of reconstruction, whether in its representation as octree leaves or the volume contained in the 3D convex hull, with the points of the painted version is in itself no sufficient information, since it does not take into account the spatial arrangement of the octree leaves or shape of the 3D convex hull, respectively.

Several different approaches have been created over the years, mainly in the context of object recognition to compute some distance measure between two given 3D shapes, for instance [1,2,9,72].

Local 3D features (see [6,44,65,69]) could also be used as building blocks to create such a distance measure, when combining several features into a descriptor for a given object, which can be compared against the descriptors created for different objects. However, in this case it turns out that just checking those features individually against each other is not sufficient – since in this case the spatial arrangement of the individual features to each other in a single object are not considered: for example, if local features are computed for a mug and afterwards its height is doubled by glueing a cylinder of the same radius on its top, the vast majority of the local features detected in the original mug will still be retained by the elongated version, but we would consider both shapes as rather dissimilar in this specific context, since one is roughly half the size of the other.

A similarity measure for 3D shapes in this specific context has two main requirements, namely, no scale invariance and the ability to deal with partial information, if an object was either never fully observable or some parts were not successfully reconstructed. Instead of opting to develop another 3D similarity measure that fits the needs of this use-case, that could probably constitute a thesis on its own, a simpler, but less elaborate solution to this problem was chosen.

Since the viewpoint-based reconstruction provides a 3D volume that is very similar to the 3D convex hull of an object, a comparison of the latter is part of the reconstruction evaluation. First in terms of pure volume, where the volume obtained by VPBR is set in relation to the volume of the convex hull of the reconstruction of the non-transparent version. Afterwards both volumes are set in relation to the volume of the 3D convex hull of the ground truth objects.

In order to refer to the glasses in Figure 5.24 in a concise manner, they will in the following be labeled as `glass1` – `glass8`, with `glass1` denoting the left-most painted glass (beer glass) in Figure 5.24 and then proceeding to the right till `glass8` (Champagne flute).

In case of the glasses featuring a stem, an additional distinction is made: since the stems are so thin that they do not cause `NaN`-regions on the tabletop, the resultant viewpoint-based reconstruction is composed of just the upper part of the glass that "floats" above the tabletop. Therefore, the volume of the 3D convex hull of VPBR, which is missing the stem, will differ largely from the volume of both, ground truth and non-transparent reconstruction, that include the stem. For a better comparison of the parts that were successfully reconstructed via VPBR, an additional evaluation is performed for all glasses featureing a stem. In this comparison the stem is manually removed from ground truth model and non-transparent reconstruction. To indicate whenever the viewpoint-based reconstruction is compared with its counterparts that had their stem removed, they are marked with a star, i.e., `glass6`*, `glass7`* and `glass8`*. The results of this comparison are displayed in Table 5.8, which first provides the absolute volumes and then proceeds to display the ratios, if volumes are compared to each other.

As mentioned earlier, the volumetric comparison is just one indicator, but not sufficient to claim a good reconstruction. Therefore, in addition a comparison based on the distances between ground truth model and reconstruction was performed as well, which is the reason for the previously mentioned alignment of both reconstructions with the ground truth model.

	volume convex hull			comparison		
Object	trans. ①	solid ②	"ground truth" ③	①/②	①/③	②/③
glass1	1432.21 cm^3	1220.1 cm^3	741.92 cm^3	117.38 %	193.04 %	164.45 %
glass2	918.80 cm^3	1048.72 cm^3	597.39 cm^3	87.61 %	153.80 %	175.55 %
glass3	692.11 cm^3	690.95 cm^3	398.05 cm^3	100.17 %	173.88 %	173.58 %
glass4	409.29 cm^3	440.81 cm^3	272.99 cm^3	92.85 %	149.93 %	161.47 %
glass5	753.52 cm^3	891.22 cm^3	535.98 cm^3	84.55 %	140.59 %	166.29 %
glass6	538.25 cm^3	855.04 cm^3	741.36 cm^3	62.95 %	72.60 %	115.33 %
glass6*	538.25 cm^3	592.76 cm^3	298.49 cm^3	90.80 %	180.32 %	198.59 %
glass7	852.59 cm^3	1265.21 cm^3	1140.32 cm^3	67.42 %	74.77 %	110.95 %
glass7*	852.59 cm^3	862.74 cm^3	475.52 cm^3	98.82 %	179.30 %	181.43 %
glass8	557.71 cm^3	881.53 cm^3	641.63 cm^3	63.27 %	86.92 %	137.39 %
glass8*	557.71 cm^3	662.04 cm^3	263.66 cm^3	84.27 %	211.53 %	251.10 %

Table 5.8: Comparison of computed convex hull volumes: first three columns specify the volumes of the convex hull for the different glasses, followed by their ratios.

First a comparison on the point clouds is performed. That means that for each point in both reconstructions, transparent and painted glass, the closest point of the ground truth model is determined. Since the ground truth model features a much higher point density than both reconstructions, the differences in point density between reconstruction and ground truth can be retained. This evaluation was done in CloudCompare [27] and the results of this point cloud based comparison are shown in Table 5.9.

Besides the point cloud comparison, the convex hulls, namely, their mesh representations, are also compared via CloudCompare in a similar fashion, i.e., for each point that defines the convex hull of one of the reconstructions the distance to the convex hull of the ground truth model is computed. Since the individual faces of the convex hull of the ground truth model define a normal (that points outwards), distances are assigned a sign ($\pm$) to indicate on which side of a face the point is located. The results of this comparison are compiled in Table 5.10.

To better convey the meanings of the numbers accumulated in Tables 5.8, 5.9 and 5.10, visualizations of the comparisons are provided next. The reconstructed objects are displayed alongside their "ground truth" counterpart from three different points of view (top, side, perspective) and in two different modalities: the point cloud representation refers to Table 5.9 and a visualization using meshes shows the results for Table 5.10. Points are colored according to their distance to the ground truth model, where green indicates distances at or very close to 0 cm. Colors from green, via yellow, to red mark increasing distances between reconstruction and ground truth where every distance that exceeds 3 cm is colored in pure bright red. If the divergence between reconstruction and ground truth was smaller than the fixed scale, this is shown as gray areas in the displayed error distributions. When comparing convex hulls in their mesh representation, a signed distance function becomes available, depending on which side a point is located in relation to the face of a mesh. Negative distances, i.e., for those parts of a

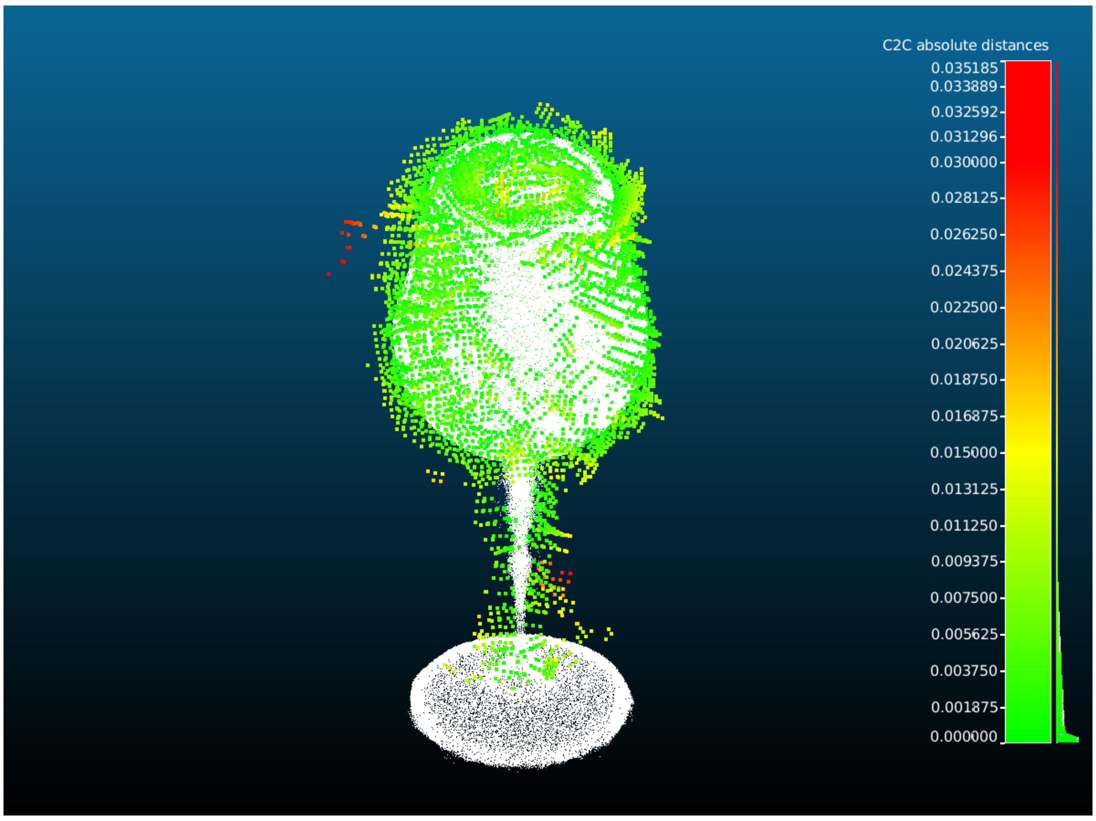
	min. dist.		max. dist		avg. dist.		σ	
Object	trans.	solid	trans.	solid	trans.	solid	trans.	solid
`glass1`	0.010 cm	0.001 cm	5.210 cm	3.628 cm	1.245 cm	0.323 cm	0.746 cm	0.328 cm
`glass2`	0.011 cm	0.001 cm	5.357 cm	2.618 cm	0.906 cm	0.254 cm	0.766 cm	0.279 cm
`glass3`	0.005 cm	0.001 cm	4.630 cm	2.649 cm	0.958 cm	0.275 cm	0.782 cm	0.302 cm
`glass4`	0.007 cm	0.002 cm	4.361 cm	2.572 cm	0.779 cm	0.278 cm	0.731 cm	0.282 cm
`glass5`	0.004 cm	0.002 cm	4.897 cm	3.422 cm	0.838 cm	0.369 cm	0.820 cm	0.354 cm
`glass6`	0.015 cm	0.002 cm	4.004 cm	2.855 cm	0.876 cm	0.295 cm	0.615 cm	0.325 cm
`glass6`*	0.015 cm	0.001 cm	4.004 cm	2.845 cm	0.878 cm	0.278 cm	0.614 cm	0.315 cm
`glass7`	0.009 cm	0.002 cm	3.689 cm	3.518 cm	0.973 cm	0.347 cm	0.583 cm	0.329 cm
`glass7`*	0.009 cm	0.002 cm	3.689 cm	3.472 cm	0.992 cm	0.353 cm	0.577 cm	0.331 cm
`glass8`	0.010 cm	0.001 cm	3.429 cm	3.120 cm	0.984 cm	0.320 cm	0.477 cm	0.327 cm
`glass8`*	0.010 cm	0.002 cm	3.429 cm	3.120 cm	0.987 cm	0.322 cm	0.476 cm	0.337 cm

Table 5.9: Point cloud based comparison of reconstruction errors.

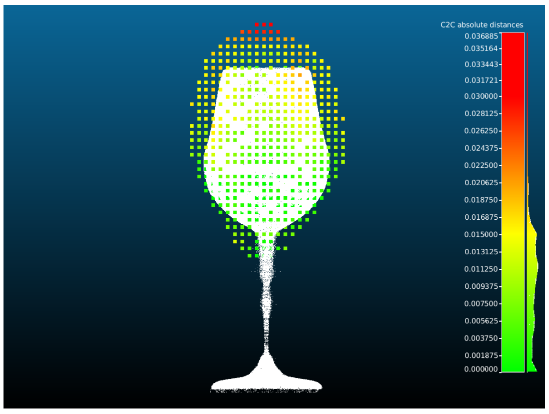
reconstructed convex hull that are inside the convex hull of the ground truth object are colored from green to blue, where pure blue indicates distances of -0.35 cm and below.

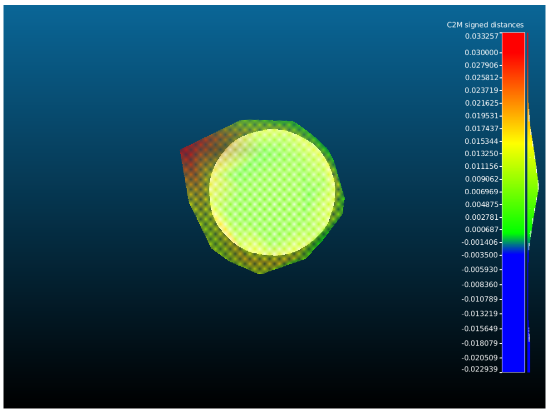
Figure 5.25 shows an exemplary visualization in a large scale for a subset of the results obtained for `glass7`. These images are complemented by a medium scale depiction of the complete set for glasses `glass2` and `glass6` in Figures 5.26 and 5.27. The delicate stem of the wineglass `glass6` is not successfully reconstructed, since it does not repeatedly produce `NaN`-measurements. To give a better feel of the quality of the parts that were actually reconstructed, Figure A.1 in Appendix A shows the reconstruction results alongside modified versions of ground truth and solid object, where the stem has been removed. The remaining results, including the complete comparison set for `glass7` are also shown in the Appendix A, but in a smaller scale in Figures A.2 and A.3. In accordance to the results displayed in Tables 5.9 and 5.10 glasses that feature a stem, i.e., `glass6` – `glass8`, are compared to two instances of the ground truth object: one that contains the stem and one where it was removed. In the latter case, the point cloud of the directly measured reconstruction is adapted accordingly.

Concerning the results in Tables 5.8, 5.9, 5.10 and the corresponding Figures 5.25 – 5.27 and Figures A.1 – A.3, some general observations can be made. When considering the values of the generated 3D convex hulls (see Table 5.8) transparent and directly measured reconstruction are not too different, granted that we ignore `glass6`, `glass7`, `glass8` and instead consider the results for `glass6`* – `glass8`*. Setting the volumes of the convex hulls in relation with the ground truth convex hull volume, somewhat surprisingly VPBR is oftentimes closer to the actual volume, with the notable exception of `glass1`. In all cases the volume estimated by both reconstructions exceeds the volume obtained for ground truth, again ignoring `glass6` – `glass8`. This effect is caused by the larger amount of noise in the RGB-D data that tends to extend the 3D convex hull of a combined point cloud, and possibly due to small alignment errors during the automated registration of the individual frames.



(a)





(b) **(c)**

Figure 5.25: Visualization of viewpoint-based reconstruction of `glass7`, detailed view: **(a)** shows the point cloud comparison of directly recorded measurement points against ground truth, from the perspective point of view, **(b)** displays the side view of the point cloud comparison of the transparent reconstruction with ground truth and **(c)** depicts a top view of the comparison of the convex hulls (direct measure vs. ground truth). The depicted c2c distances are given in m.

	min. dist.		max. dist		avg. dist.		σ	
Object	trans.	solid	trans.	solid	trans.	solid	trans.	solid
`glass1`	-0.005 cm	-0.472 cm	3.887 cm	3.243 cm	1.375 cm	0.989 cm	0.812 cm	0.612 cm
`glass2`	-0.880 cm	-0.716 cm	3.836 cm	2.708 cm	0.949 cm	1.107 cm	0.854 cm	0.757 cm
`glass3`	-0.362 cm	-0.510 cm	3.655 cm	2.648 cm	1.026 cm	0.810 cm	0.871 cm	0.660 cm
`glass4`	-0.605 cm	-1.009 cm	3.223 cm	2.520 cm	0.718 cm	0.714 cm	0.671 cm	0.745 cm
`glass5`	-0.667 cm	-0.764 cm	3.626 cm	3.388 cm	0.801 cm	0.940 cm	0.957 cm	0.896 cm
`glass6`	-3.404 cm	-1.359 cm	2.898 cm	2.305 cm	0.526 cm	0.610 cm	1.293 cm	0.955 cm
`glass6`*	-0.122 cm	0.024 cm	2.898 cm	2.317 cm	1.028 cm	1.142 cm	0.594 cm	0.432 cm
`glass7`	-3.703 cm	-2.294 cm	2.959 cm	3.326 cm	0.691 cm	0.516 cm	1.348 cm	1.117 cm
`glass7`*	0.152 cm	0.180 cm	2.959 cm	3.383 cm	1.191 cm	1.130 cm	0.567 cm	0.574 cm
`glass8`	-2.510 cm	-1.613 cm	2.979 cm	3.016 cm	0.697 cm	0.700 cm	1.100 cm	1.076 cm
`glass8`*	-0.032 cm	0.392 cm	2.979 cm	3.075 cm	1.102 cm	1.412 cm	0.530 cm	0.437 cm

Table 5.10: Reconstruction comparison based on convex hulls.

If we take a closer look at the point-based comparison between both reconstructions and ground truth, the viewpoint-based reconstruction performs much worse than the reconstructed objects obtained from the direct measurements. This result should not be too surprising: using the convex hull of the hole regions in each frame as the building block for VPBR prevents it from modelling the non-convex regions of objects in detail. However, some non-convexity can be induced into the object's approximation via parameters $\mathrm{I}_{\mathrm{min}}$ and $f_{\mu_{\mathrm{I}}}$, that are used to refine the approximated volumes. Furthermore, we should notice that the viewpoint-based reconstruction results in a closed surface of points, while in the directly obtained reconstruction and in the ground truth data the glasses feature a rim and an opening (refer to Figures 5.26 and 5.27, right column, top two rows for a visualization). For those points in the transparent reconstruction above the object (and also below) no close counterparts exist in the ground truth objects, hence increasing the maximal and average distances in Table 5.9.

This is also reflected, when observing the results for the convex hull comparisons, where the numerical differences of both reconstructions from the ground truth are much more similar. The computation of the convex hull for ground truth objects and directly measured reconstruction results in a closed property for the resultant mesh, while it is naturally retained for the viewpoint-based reconstruction, when computing the convex hull of the estimated volume. Therefore, corresponding surfaces for parts of the transparent reconstruction that are located above the ground truth object are closer than in the point cloud comparison.

When considering the visual appearance of the reconstructions, as shown in Figures 5.25 – 5.27 and in the Appendix in Figures A.1 – A.3, tasks like collision avoidance or object manipulation appear feasible for the transparent reconstructions, if we assume that the reconstructions obtained from the solid measurements can be employed in these tasks. Unfortunately object manipulation could not be performed in the experimental setup, since the manipulator in the Calvin platform (see Figure 5.23) is rather unreliable, even when attempting to manipulate

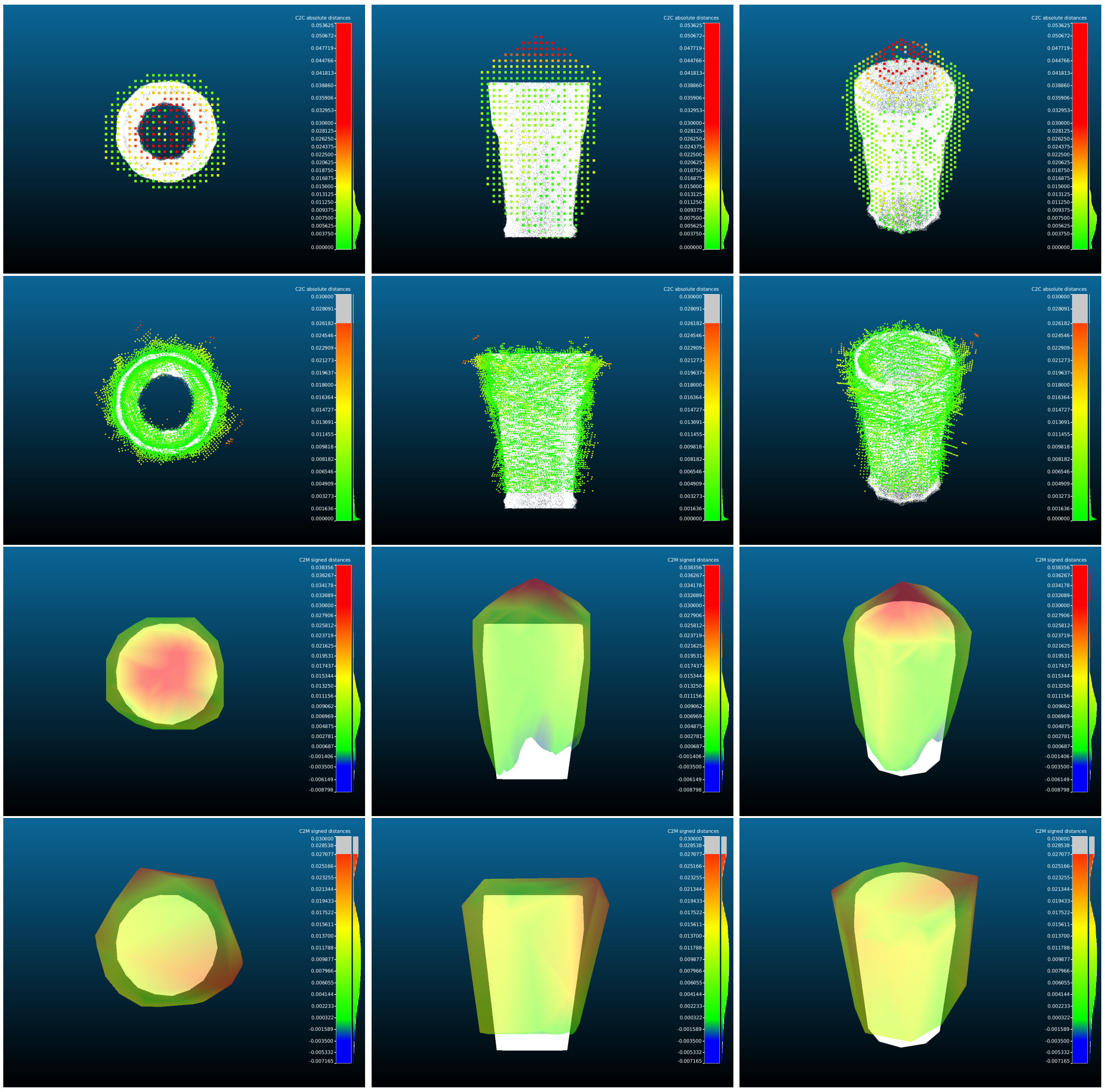


Figure 5.26: Comparison viewpoint-based reconstruction `glass2`. From left to right: top view; side view; perspective view. From top to bottom: point cloud viewpoint-based reconstruction; point cloud direct measurements; convex hull viewpoint-based reconstruction; convex hull direct measurements.

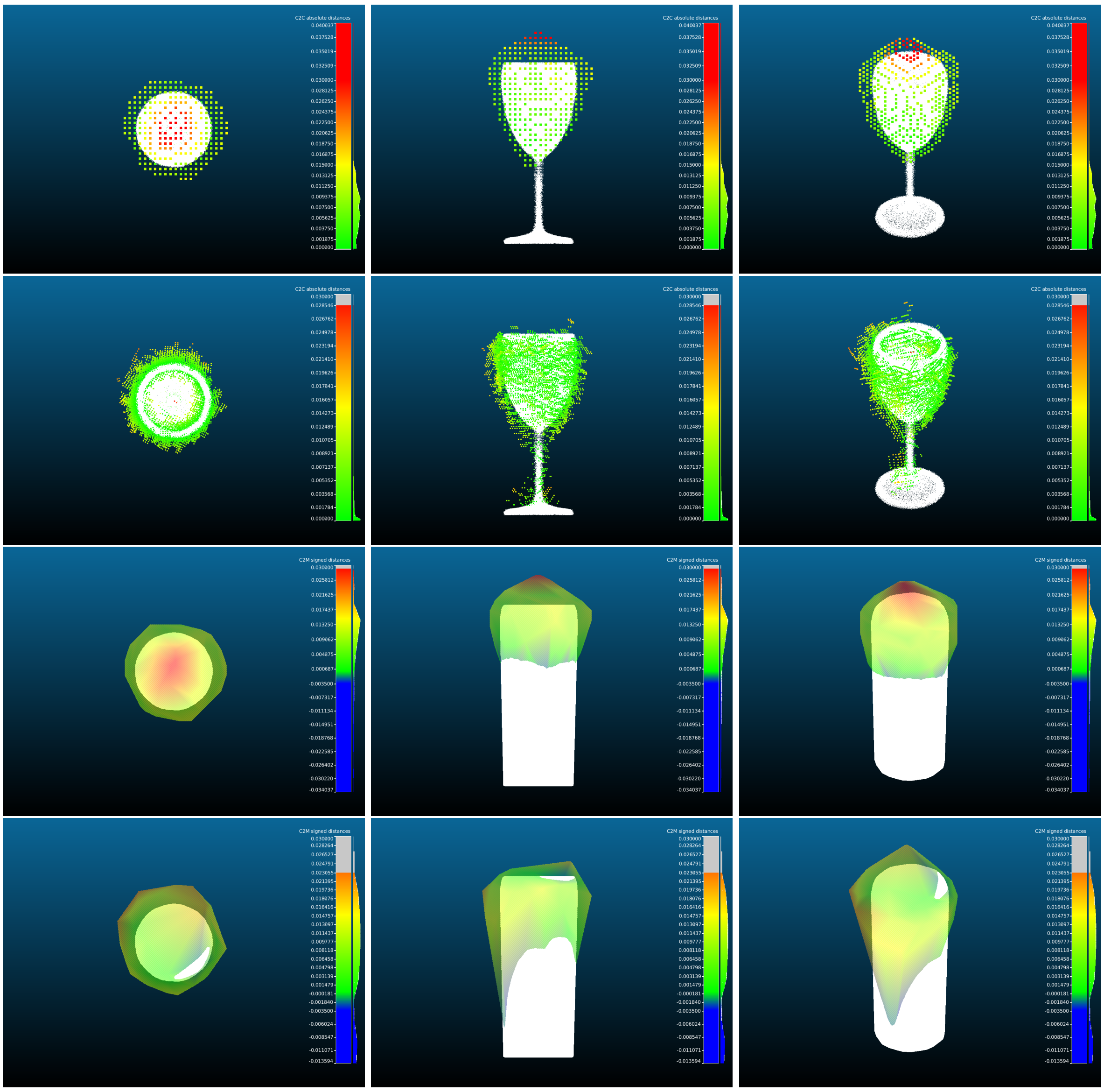


Figure 5.27: Comparison viewpoint-based reconstruction `glass6`. From left to right: top view; side view; perspective view. From top to bottom: point cloud viewpoint-based reconstruction; point cloud direct measurements; convex hull viewpoint-based reconstruction; convex hull direct measurements.

regular objects. Therefore, the ratio of successful grasps to non-successes would more reflect the arm's inaccuracy than the suitability of the transparent reconstruction and its estimated location.

Some of the drawbacks of the used evaluation method were pointed out above and some measure specifically designed to compare 3D shapes might provide some more insight. Yet the chosen evaluation approach already goes beyond the evaluations done in other approaches that try to reconstruct transparent objects:

Klank et al. [42] report a detection rate of 55.24 % and successful grasping of 41.38 % of the detected transparent objects as proof for the quality of their reconstruction method, without providing any quantitative information about the objects volume in relation to some baseline or elaborating on how the grasping operation of the transparent objects was implemented.

In [77] Torres-Gómez and Mayol-Cuevas report classification rates of their reconstruction approach (80 % true positives, with 13 % false positives on their evaluation set of 65 scenes), they do not compare their reconstructions with any form of ground truth.

Lysenkov's [48, 49] approach differs, since he does not attempt to reconstruct unknown transparent objects in a given scene, but tries to identify learned transparent objects in recorded Kinect frames. Nevertheless, he investigated how good his identification and pose estimation for the transparent objects performed. For this experiment he grouped transparent objects into 5 different classes, namely, "bank", "bucket", "glass", "bottle" and "wine glass". The results concerning the identification varied strongly, depending on the examined object. For instance, for three different objects (bank, bucket and glass) the rate of correct identification is reported as 93 %, but the bottle is often confused with the glass (correctly identified in 57 %) and the wine glass is never correctly identified, due to its thin stem.

With respect to the pose estimation, the experimental setup, reported in [48], placed all (rotational symmetrical) objects upright on a tabletop, so that the rotational part of the pose estimation could be ignored. Whenever the translation error was less than 2 cm, the pose estimate was considered a success. This threshold helps to puts the reported results in Tables 5.9 and 5.10 into perspective: the average distance between both reconstructions, transparent and regular, and the ground truth is always less than the 2 cm employed by Lysenkov. Except for comparison of the convex hulls for `glass1` and `glass7` the sum of average distance and σ also does not exceed this threshold. Even though a confusion matrix is reported for the different glass objects in [48], surprisingly no detection rates are provided, i.e., the ratio of true positives to false positives is not reported.

While the number of investigated scenes and transparent object samples is too small for doing a serious statistical analysis of the performance of reconstruction and, connected to this, detection of the transparent objects with the discussed method, some things should still be mentioned here. In all scenes, containing a single transparent object from the collection of `glass1` – `glass8`, this object was successfully detected. This holds also true for the later briefly mentioned scenes that were less structured and contained more than 1 single transparent object. This would constitute a recall score of 1.0. Concerning the number of detected false positives in the recorded scenes, these are either 1 or 0, depending on the interpretation of the data: one false positive transparent object was detected on the stem of the painted version of `glass6`. However, when closely inspecting the stem at that particular location, it turned out that there was actually some color missing, i.e., a small part of the stem was indeed transparent. If we count this as a false positive and we end up with a precision value of 0.93 (one false positive and 14 true positives in all scenes). These two values lead to a F1-score of 0.966, which is much better than any other method that actually provides numerical evaluations (see [26, 42, 77]). Note however, that these results are not obtained by single frames, but by the aggregated information from multiple frames.

When observing the results displayed in Tables 5.9, 5.10 alongside Figures 5.26, 5.27 and Figures A.1 – A.3, it is evident that the largest difference, in terms of distance between VPBR and ground truth is located at the apex of the reconstruction. One reason for this, when considering the point cloud comparison, has already been mentioned, namely, that VPBR provides a closed approximation of the volume, while the ground truth point cloud contains the opening of the drinking glass. However, there is also a second reason for the systematic overestimation of the height in the transparent reconstructions: the sensor setup of robot platform Calvin (Figure 5.23), used to conduct these experiments. On Calvin, the employed RGB-D camera is located in a fixed position, observing the tabletop from a relatively high location (ca. 1.75 m above ground, so roughly 75 cm above a tabletop) with a relatively steep angle between its viewing direction and the tabletop.

As seen earlier in 5.1.2, recording the scene from such poses can cause a systematic overestimation of the height of transparent objects, when employing the intersection method. The same holds true for the viewpoint-based reconstruction, where a somewhat cone-shaped volume above the actual ground truth object is part of the majority of the individual occlusion frusta that are used to compute the viewpoint-based reconstruction. If the sensor's position on the robot platform could have been changed, so that the tabletop could additionally be observed in a side view fashion, this type of overestimation could have been prevented. Detection of valid `NaN`-regions should have been no problem, as the experiments done in subsections 5.1.1 and in 5.1.2 both suggest, if observations from different angles were possible when employing the Calvin platform.

To provide a rough idea how the reconstruction on a robotic platform with such capabilities might perform, two glasses were picked and part of the overestimation was manually removed. A visual representation if presented in Figure 5.28, where the modified results of VPBR for `glass2` and `glass6`* are shown. While points near the apex were manually removed, the alignment of the viewpoint-based reconstruction to the ground truth models have not been modified. The change in the displayed point-to-point distances are therefore only the result of the point removal and not caused by improving the alignment. The gray area in upper part of the fixed scale that displays the distribution of the point-to-point distances reflects this improvement. After the points were removed, convex hulls for both glasses, `glass2` and `glass6`*, were recomputed and compared with their ground truth counterparts.

Needless to say that this manual editing improves the comparison results with ground truth on all three evaluated scales (volume of convex hull, distances point clouds and distances convex hulls). However, since the removal was performed without simulating a sensor, but in a way that the results look plausible, the computation of the numerical values, that directly depend on the this decision, was omitted. The main purpose of Figure 5.28 is to illustrate that part of the inaccurate reconstruction is not inherent in the approach, but caused by the limitations of the available robotic platform, in particular, the rigid mount of the RGB-D sensor.

In fairness to the results of the reconstruction from the painted glasses, one should men-

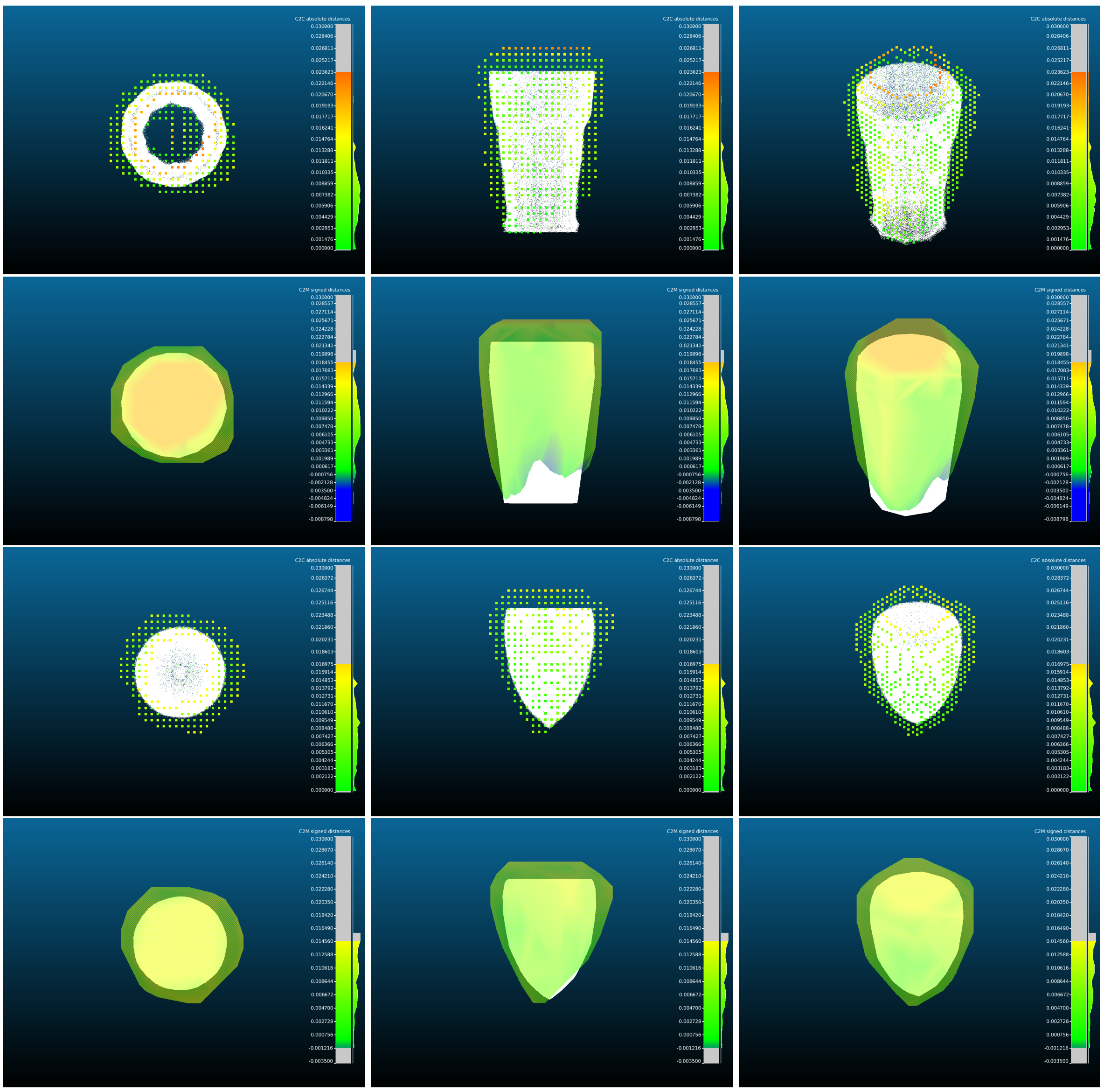


Figure 5.28: Reconstruction with manually removed apex points. Top two rows show the modified results for `glass2`, displayed in Figure 5.26, and the two bottom rows feature the modified version of `glass6`* (Figure A.1).

tion that they could possibly be improved by additional filtering steps: for instance, once the clusters are determined, additional outlier removal in relation to the average point distances of the point clusters could be performed, so that some erroneous measurements could still be removed. When objects and tabletop are colored in easily distinguishable colors, as is the case here, one could also associate tabletop inliers with near objects clusters, on a color base. This idea, however, requires a precise calibration between color information and 3D information.

As detailed in subsection 3.3.3, the octree leaves that fulfill inequality (3.6), i.e., provide the initial estimation of the transparent objects, are subject to median filtering based in the viewpoint intervals in each individual cluster. Figures 3.13 and 3.14 illustrate the difference in the approximated shapes. The same technique has been applied to the reconstruction of the glasses in this experiment and the reported values in Tables 5.8 – 5.10 already incorporate this filtering step. The effect on the estimated shape of this heuristic filtering step is visualized in Figure 5.29 for the example of `glass2`. Each depicted point marks the center of an octree leaf that is assumed to be part of the transparent object, and the convex hull of the tabletop along a vector pointing upwards at the table's center are shown to get a better impression of the different points of view. Evidently, the refinement achieved by the median filter (green point clusters) improves on the initial estimation shown in the top row of Figure 5.29 (blue clusters). Considering the results in Tables 5.8 – 5.10, where the reconstruction overestimates ground truth, even after the filtering step was applied, the median filtering is not just visually more appealing, but also improves the results in a quantitative manner. Changing parameter f_{μ_I} (set to 1 in the presented results), to remove / retain more octree leaves in each detected cluster, will change both, 3D shape estimation and numerical results, of the reconstruction.

Reconstruction in less structured Scenes

For the purpose of a quantitative evaluation, the results presented up until now contained only a single transparent object together with regular objects, including its painted counterpart, on the table. The reconstruction mechanism in itself however is not restricted to single objects in the complete scene, but can deal with multiple transparent objects, without needing to change any of the parameter settings. In order to demonstrate this ability, the table was set in a more "natural" way, where objects were not that evenly arranged on the table and more than one transparent object is present in the setting. A depiction of two of such scenes is provided by Figure 5.30.

The increased difficulty with respect to the reconstruction of transparent objects is not only due to the fact that more than one transparent object is present, but also in the arrangement of all objects on the tabletop. Compared to the scene for the quantitative analysis, the placement of the objects is less regular and evenly spaced. This implies a greater likelihood of (partial) occlusions of the transparent objects themselves or of the `NaN`-regions that they cause on the tabletop. Therefore, it may be possible that a `NaN`-region is sufficiently located inside the convex hull of the tabletop, but does underestimates the actual size of the associated transparent object. On the other hand, some `NaN`-regions may be larger than expected, namely, if two transparent objects overlap, from the sensor's point of view and produce one large `NaN`-region instead of two distinct, smaller ones. An example can be seen in 5.30c where a beer bottle overlaps with a glass (red ellipse), thus creating one large convex hull for the enclosed `NaN`-region.

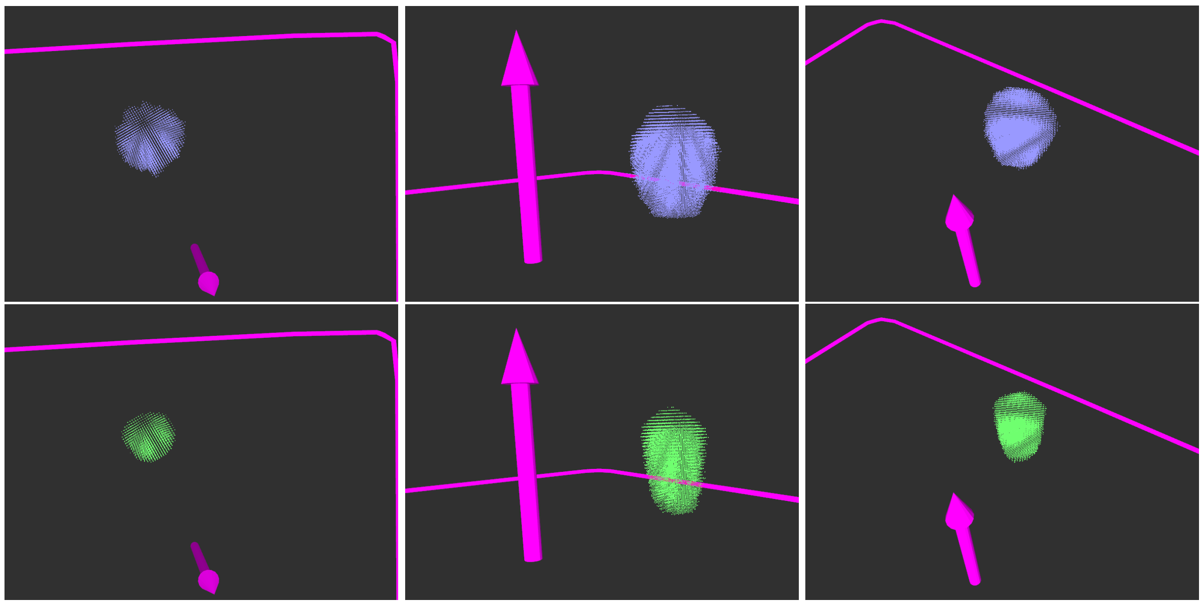


Figure 5.29: Effect of median filtering on estimated transparent object shape. Top row: Three different views (top, side, above and side) of the estimation for `glass2` (see Figure 5.24) without median filtering. Bottom row: Refinement of estimation after median filtering was applied. The outline of the convex hull of the tabletop and its up vector at the center point of the table are included as references.

The two beer bottles in Figure 5.30c and 5.30d are objects that feature inhomogeneity with respect to transparency: the glass part of the bottle produces `NaN`-values, hence with respect to the point cloud obtained by the RGB-D sensor it can be seen as a transparent object. The paper labels, on the other hand, are regular surfaces that can be measured directly. If the reconstruction method can deal with these types of objects and scenes, there is strong evidence for its robustness.

Provided that the estimated volumes roughly correspond to the transparent objects present in the scene, the distinction between them is fairly uncomplicated, since this constitutes a simple Euclidean clustering, performed on the point cloud that represent transparent leaves in the underlying octree. In case of transparent objects that are placed very close to each other, Euclidean clustering might not be able to distinguish the two objects, but assumes a single large transparent object – however this also holds true for clustering performed on points obtained from regular objects.

Figures 5.31 and 5.32 visualize the reconstruction for the breakfast table and the beer bottle setting, respectively. In both figures the tabletop is shown after gathering information from 12 individual frames and the tabletop is displayed from the same points of view. The magenta line in all images shows the convex hull of the tabletop that was accumulated during the sequence. Hence, it extends the displayed points of the tabletop, which only show the last sensor input of the 12 total frames. Note that the convex hull in Figure 5.31 slightly differs from the one in Figure 5.32 (most noticeable in the top views in the bottom left). This is caused by the combination of different sensor noise and imperfect frame alignment in both settings. Uni-

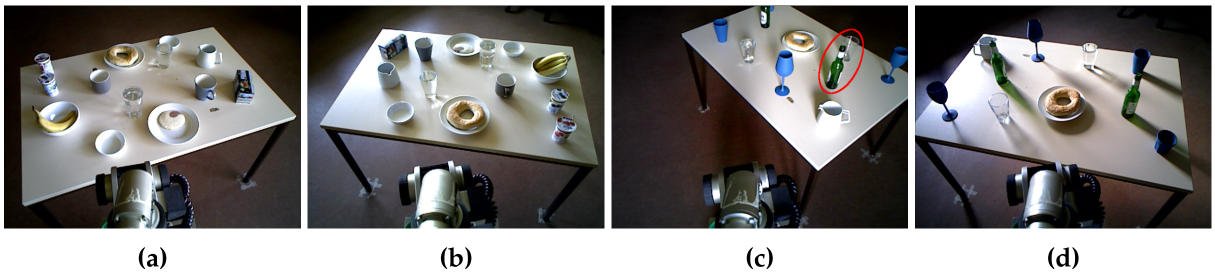


Figure 5.30: Scenes for reconstruction of multiple transparent objects, as observed by the robot: **(a)** and **(b)** show a "breakfast" setting while the scene in **(c)** and **(d)** depict a scene that contains two beer bottles along the drinking glasses. The view in **(c)** highlights two overlapping transparent objects. Notice the different lighting conditions between the individual frames.

formly colored point clouds show the regular objects in both scenes, and a colored solid sphere represents their associated centers of gravity. Reconstructed transparent objects are marked by translucent convex hulls and text; their centers of gravity are indicated by a small coordinate system origin.

The results displayed in Figure 5.31 and 5.32 indicate that the proposed reconstruction is robust enough to also work in more cluttered scenes, when compared to the setup used for the quantitative analysis. With respect to the quality of the reconstruction, the estimation of the objects' convex hulls does not seem fundamentally worse when compared to the reconstruction results achieved for the regular objects in the scene. Also the beer bottles that combine properties of transparent with regular objects are reconstructed adequately. In case of one bottle (`trans_obj02` in Figure 5.32) the location of the transparent reconstruction overlaps with a regular object reconstructed by Görner's method [31]. The points of this object are made up by the paper labels and the sleeve near the top of the bottle, indicating that neither location estimation nor size of the estimated beer bottle is too far off in this case.

When considering the obtained results it makes sense to also inspect the raw data that is used to obtain these results. As described in subsection 3.3.3, the convex hulls of `NaN`-regions, detected in the fashion as detailed in 3.2.2, form the building blocks for the reconstruction of transparent objects. Under the varying lighting conditions, depending on the position of the robot in relation to the tabletop (see Figure 5.30), the occurrence and shape of `NaN`-measurements in the presence of transparent objects also varies. Also occasionally highlights on regular objects may cause additional `NaN`-regions. To give an impression on the data that was used to reconstruct the two beer bottles and glasses, as shown in Figure 5.32, please refer to the images in Figure 5.33:

Pixels that are colored in bright red in the photographs indicate `NaN`-measurements that might be relevant for the reconstruction, i.e., the "frame" of `NaN`-measurements located near the edges of the images (see 3.6 on page 50 as reference), the `NaN`-measurements caused by the occlusion of the robot's arm and regions that are directly connected with the former are removed. The cyan line marks the convex hull of the tabletop in the current frame. Keep in mind that this is computed on the 3D point cloud and subsequently projected back into the image. Therefore, the detected convex hull does not encompass those parts of the tabletop that

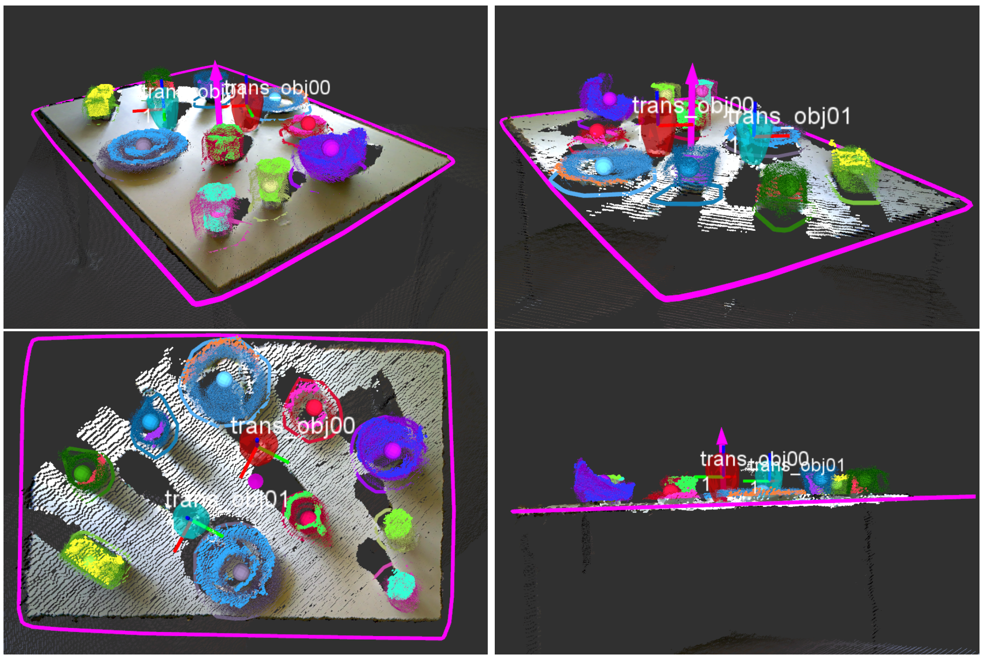


Figure 5.31: Reconstruction results "breakfast table".

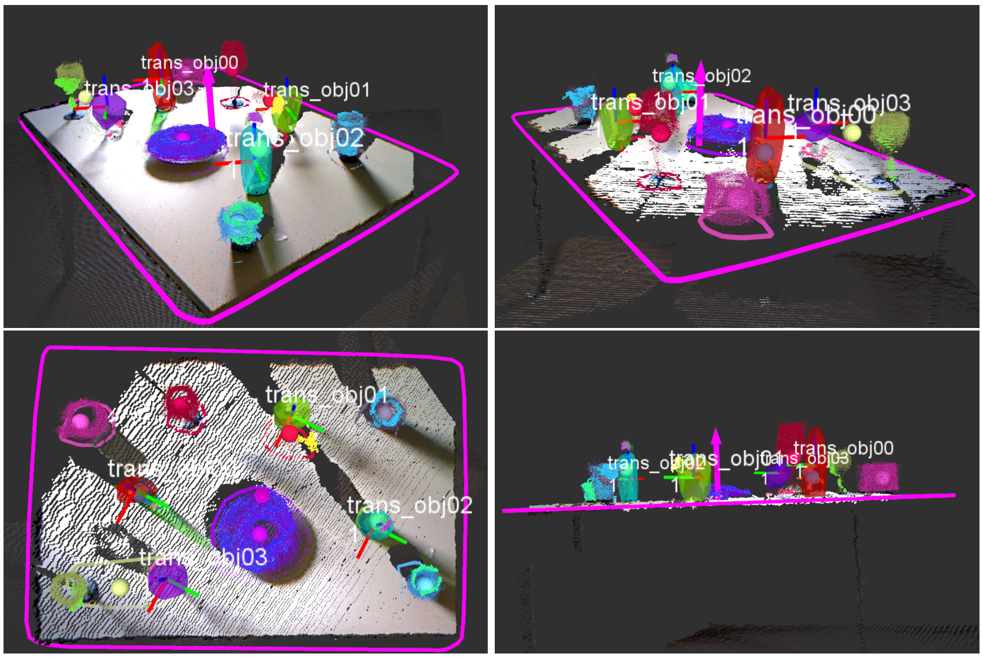


Figure 5.32: Reconstruction results "scene with beer bottles".

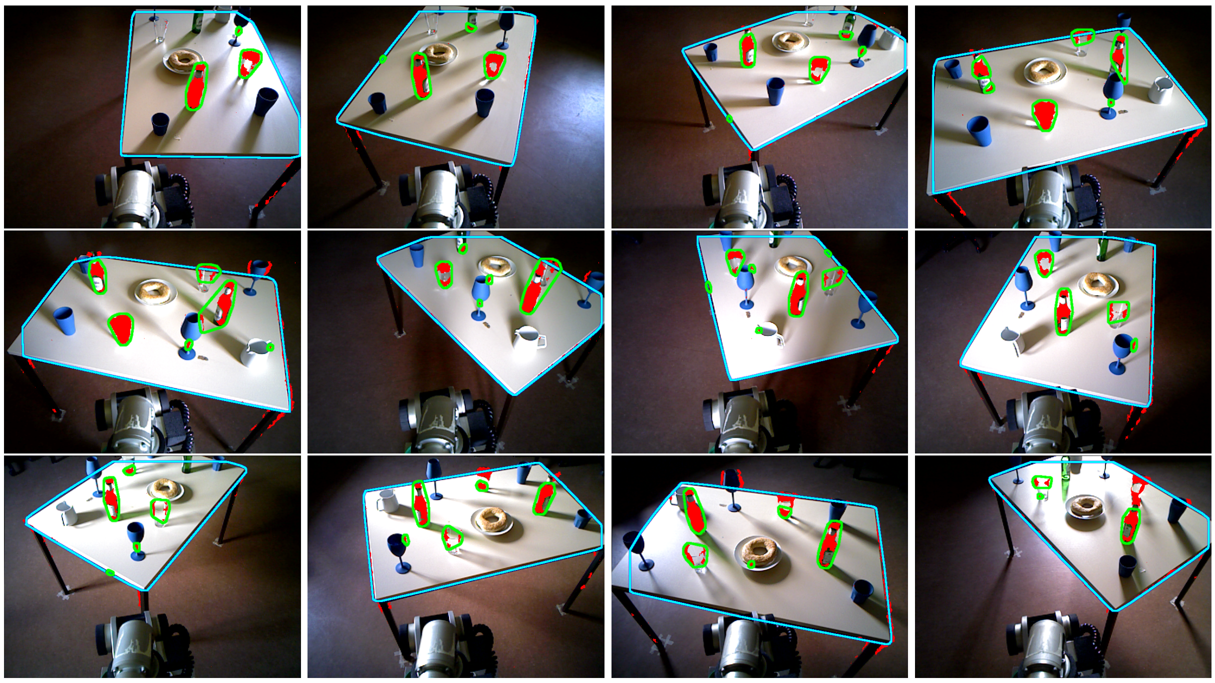


Figure 5.33: Individual frames with `NaN`-regions for "beer bottle" sequence. Bright red indicates `NaN`-measurements, the cyan line depicts the convex hull of the tabletop in the current frame and green lines show the convex hull of (fused) `NaN`-regions that are sufficiently inside the convex hull of the tabletop.

are close to the edges of the image, since no 3D information is available for those pixels. Green lines display the convex hulls that are considered of `NaN`-regions that are potentially caused by transparent objects, which are located sufficiently inside the convex hull of the tabletop.

Ideally, no large enough `NaN`-region would be located outside of any actual transparent object and all transparent objects would produce a dense region that resembles their shape in the image. Unfortunately, this is obviously not the case with real data. Considering the available information from the `NaN`-measurements, the reconstructions shown in 5.32 are more than satisfactory. Furthermore, also note, that while individual frames contain false positives of transparent object locations, no false positive was reconstructed and the obtained approximations are placed at the correct locations on the tabletop, as far as a visual comparison between the photographs in Figure 5.33 and the data shown in Figure 5.32 is concerned.

Concluding Remarks concerning Transparent Object Reconstruction

As a concluding remark of this section, I would like to point out that the parameter settings, employed to achieve the presented results, were not specially adapted to the recorded data, but are meant to work reasonably well in a wide range of scenes. By specifically tuning the parameters for the given sequences, it is likely that reconstructions are possible that correspond more to the "ground truth" objects than the results presented in this section. Modifying parameters in this fashion would, naturally, constitute as overfitting, but it should also be mentioned that finding the "ideal" parameter settings for a given data set is not trivial, since some changing some parameters might have similar effects, but with different characteristics.

For instance, if we look at parameters $\mathrm{I}_{\mathrm{min}}$ and $f_{\mu_{\mathrm{I}}}$, both can be used to determine which leaves in the underlying octree will be assumed to be part of a transparent object. Increasing $\mathrm{I}_{\mathrm{min}}$ will reduce the number of of leaves that pass the criterion defined by inequality (3.6) while $f_{\mu_{\mathrm{I}}}$ influences the median filtering that is performed on the set of leaves that confirm with inequality (3.6). A priori, i.e., without testing the results for different settings of these parameters, it is hard to tell which combination of $\mathrm{I}_{\mathrm{min}}$ and $f_{\mu_{\mathrm{I}}}$ will provide a result were the approximated volume does not overestimate the actual object or too many octree leaves are discarded, due to parameter settings.

If it is desired to approximate location and shape of transparent objects after obtaining less information, as opposed to the conservative parameter combination of $w_{\mathrm{vp}} = 10\,^{\circ}$ and $\mathrm{I}_{\mathrm{min}} = 120\,^{\circ}$, these can of course be changed, but probably at the expense of the detection of some false positives, if we take into account the quality of the raw data displayed in Figure 5.33. To a certain degree, the resulting number of false positives might be reduced again by increasing the minimal number of `NaN`-measurements that constitute a potential transparent object location in the scene (15 connected pixels in the employed settings), but increasing this number might cause some `NaN`-measurements, caused by transparent objects, to be discarded, too. This in turn might lead to some false negatives in the reconstruction, i.e., some transparent objects in the scene might be missed. As an example of valid locations that might be missed, when increasing the number of `NaN`-measurements, please refer to the bottom row of Figure 5.33: the drinking glass in the foreground, located between the painted wine glass, the beer bottle and the plate, is created by considering the information of rather small `NaN`-regions (that are combined into a single convex hull).

All code used for the viewpoint-based reconstruction of transparent objects is freely available at `https://github.com/uos/transparent_object_reconstruction`. The ROS bagfiles containing the recorded sensor data of the experiments are available in the release version of the code on github. Together with Michael Görner's "curious table exploration" (`https://github.com/uos/curious_table_explorer`) it can be used to replicate the reported experimental results.

5.2 Pose Evaluation

Similar to the evaluation of the transparent object reconstruction, presented in section 5.1, one challenge when trying to assess the quality of the proposed evaluation measure for the alignment of 2 frames is the measure of comparison. In the case of the transparent objects a reference model was constructed with the help of a laser scanner (see 5.1.3) that served as "ground truth". For the alignment of two given frames, such a ground truth usually does not exist and, since no similar attempts to estimate the alignment quality have been published, no reference data sets exist that could be used to compare the methods proposed in chapter 4 with existing attempts.

Therefore, the results presented next will contain a certain subjectivity, namely when determining what (still) might be considered a suitable alignment of 2 frames and which alignments need to be considered erroneous. Experiments were performed on two main datasets that differ in their nature. The first dataset (subsection 5.2.1) consists of a large number of point clouds, grouped into several scenes, that were registered iteratively, using different registration methods (both ICP variants). Data was recorded by a low cost RGB-D camera (Asus X-tion) and filtered before the registration process, only leaving potential object clusters, i.e., the point clouds are neither organized nor dense. Due to the nature of the sensor and the experimental setup in which this data was recorded, measurements were obtained only at close distances, usually less than 2 m away from the sensor.

The second dataset consists of much fewer point clouds, but these were obtained outdoors with a high quality 3D laser scanner (Riegl VZ-400). While the point clouds are not stored in an organized fashion, they are unfiltered and contain a relative high point density on all measured surfaces. The size of the point clouds is on a different scale, when compared to the first dataset, not only in the number of points (more than 14.6 million points per scan vs. much less than 300,000 points per frame), but also in the dimensions, since the recorded measurement range for this dataset is between 1.4 m and 458 m. The parameter settings that were established for this dataset were afterwards re-used on a different dataset, that was recorded with a very similar sensor (Riegl VZ-400i), but in a different setting. Results for the alignment of these high quality scans and the demonstration of the general applicability of established parameter settings are detailed in subsection 5.2.2. The experiments will show that the proposed alignment evaluation methods can be applied to both kinds of data.

5.2.1 Tabletop Exploration Dataset

A first illustration of the different proposed variants of the confidence / error measures have already been shown in chapter 4, namely, for ICP results obtained by matching 2 hallway scans from different initial pose estimations (see Tables 4.7 and 4.8). In the following the confidence measures are computed for a series of tabletop scenes that were recorded by Michael Görner during his master's thesis [31]. In his thesis, Görner noted that employing full 6D ICP on the point clusters located above the tabletop sometimes resulted in a noticeably poorer alignment, when compared to the results of what he called *Plane ICP*. The latter assumes that the robot is driving on a planar surface and hence does not perform a full 6D alignment between the two scenes, but does not allow translations along the z-axis that is perpendicular to the plane on which the robot is driving. This setup also simplifies the rotational part of the scene alignment, since now only rotations around the z-axis can be applied, without violating the planar assumption. For details on *Plane ICP* please refer to [31], specifically pages 23 – 27.

From the experiments of his thesis Michael Görner provided point clouds for 17 different table settings, alongside the alignment of individual frames, obtained by both matching methods, full 6D ICP and Plane ICP. The accumulated data encompasses a total of 180 views for the matching performed by ICP, while the Plane ICP version contains 6 additional frames, which apparently were skipped when 6D ICP was performed. In conjunction with the 17 different table settings, this dataset contains 163 alignments between consecutive frames for 6D ICP and 169 poses obtained by Plane ICP, respectively.

Since both, full ICP and Plane ICP, were used in an iterative or sequential fashion in Görner's experiments, i.e., a scan is only matched with its direct predecessor, the alignment measures detailed in chapter 4 can be applied directly on the results. No ground truth data about the alignment of consecutive frames is available, hence it is not possible to obtain a quantitative difference between the results of the two different registration methods and a correct alignment. However, it is possible to evaluate each pair of consecutive frames in a qualitative fashion.

Such a qualitative rating will, to a certain degree, be subjective with the person(s) performing the assessment of each alignment. The more fine grained such a qualitative scale becomes, the more likely disagreements between observers, when determining the correct category for an alignment, will arise. Of course, when a human observer evaluates the alignment of two frames, scene knowledge is employed. For example, this means that an alignment where the object edges from both point clouds coincide is preferable to an alignment where they are not aligned with each other, but e_{avg}, the average correspondence error of ICP, is smaller.

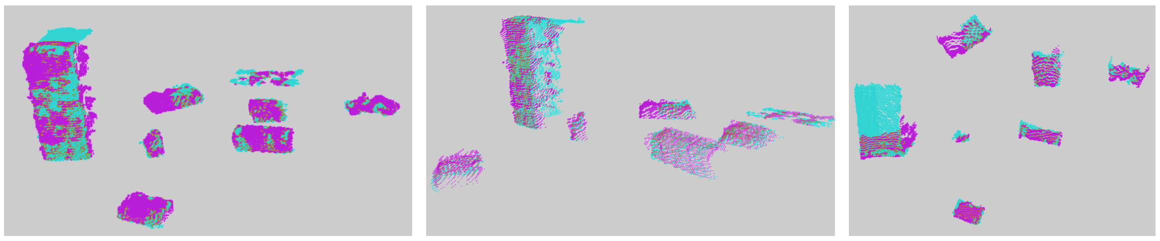
Therefore, I will distinguish only between three different qualitative categories, concerning the alignment quality in this dataset, namely

$+$ This category encompasses "good" and "sufficient" alignments. Considering the nature of the sensor data (RGB-D data) and its inherent noise and decreasing depth resolution, points of two point clouds will not (almost) coincide in all volumes of the environment that were recorded from both sensors, even if they were aligned perfectly. This nature makes it hard to distinguish between scenes that are very well aligned and might feature a larger degree of noise or scenes where the data is less noisy, but that are aligned in a slightly less concise way. While not all measurements in the overlap of both point clouds will coincide, differences are small and usually arranged in such a fashion that no evident translation and / or rotation exists that would decrease these misalignments.

$-$ Alignments that are assigned with this label contain more inconsistencies than could be explained by mere sensor noise. However, these errors might not be immediately evident, when just briefly observing the alignment. Alignments that appear to be "off" by a simple transformation, for instance a translation applied to the second frame would cause object borders to coincide, are grouped in this category.

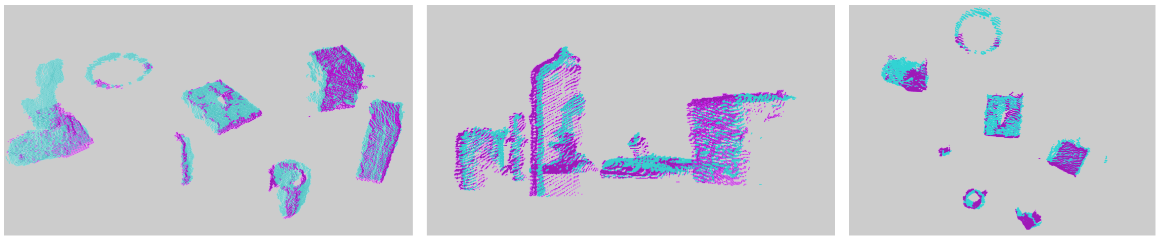
$\lightning$ Lastly there are alignments that are obviously wrong and the misalignment can be identified immediately.

To get a better feeling for these categories, a small set of examples is displayed in Figures 5.34 – 5.36.



(a) (b) (c)

Figure 5.34: Example for good alignment (class $+$): scene depicted from slighly elevated front **(a)**, side **(b)** and top **(c)**, where point colors indicate frame correspondence. Note that in those areas where points from both point clouds are present, they resemble each other decently well.



(a) (b) (c)

Figure 5.35: Example for slightly bad alignment (class $-$): scene depicted from slighly elevated front **(a)**, side **(b)** and top **(c)**, where point colors indicate frame correspondence. Though not a terrible fit, data does not align as good as the scene depicted in Figure 5.34. In the perspectives shown in **(a)** and **(b)** magenta points seem to be systematically shifted slightly to the left from their cyan counterparts, which suggests, that some translation / rotation combination might actually still improve the alignment.

The data shown in Figure 5.34 is an example of class $+$, i.e., a good or at least decent alignment. While the alignment is not perfect, this is also, in part, due to the available data: As the Figure clearly shows, not all common parts of point clouds P_1 and P_2 closely match each other and at some locations measurements are missing, from one source, that cannot be explained by field of view limitations.

If we compare the previous scenario with the alignment depicted in Figure 5.35 the latter leaves more room for improvement: while the alignment still seems sensible at first glance, a closer inspection suggests that a large portion of the magenta points in the side view (Figure 5.35b) are shifted slightly with respect to their cyan counterparts. The magenta points for the two objects in the center front are shifted to the left, while for the objects on the left and in the center back they are positioned to the right of the cyan points measured from the same objects. This suggests that a small rotation could actually improve on the depicted alignment, hence the alignment is classified as $-$.

The last example of the three different classes of alignment is shown in Figure 5.36. This is a clear case of bad alignment between both point clouds. While membership of classes $+$ or $-$ might sometimes be ambiguous, all alignments in class $\lightning$ clearly indicate that something

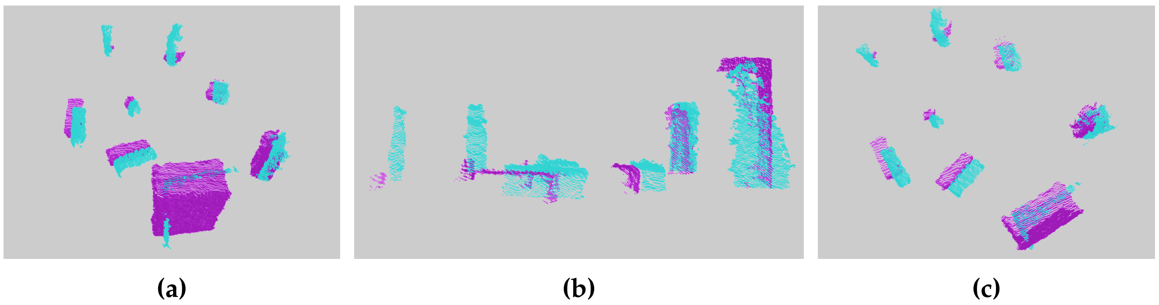


Figure 5.36: Example for bad alignment (class ↯): scene depicted from slighly elevated front **(a)**, side **(b)** and top **(c)**, where point colors indicate frame correspondence. This is an evidently bad alignment between the cyan and the magenta points.

	$+$		$-$		↯	
alignment method	# align.	%	# align.	%	# align.	%
6D ICP	67	41.10 %	69	42.33 %	27	16.56 %
Plane ICP	87	51.48 %	68	40.24 %	14	8.28 %
All frames	154	46.39 %	137	41.27 %	41	12.35 %

Table 5.11: Categorization of table object alignments: total numbers and percentages of each category is listed, dependent on the used registration method.

went wrong during alignment. One should, however, keep in mind that alignments, such as the one depicted in Figure 5.36 are the result of ICP, i.e., constitute a local minimum in the error function, albeit one that does not convince a human observer.

Even with these 3 simple categories, there are borderline cases, especially between $+$ and $-$. The results of the manual evaluations of all 332 alignments is shown in Table 5.11. The frame assessment displayed in Table 5.11 matches the one of Görner reported in [31], in that the shape of his learned objects, when employing 6D ICP, on average seemed to correspond worse to reality than the results obtained by Plane ICP. Most important in this regard is the column for obviously wrong alignment (category ↯), where the percentage, when using 6D ICP is twice as high as for Plane ICP.

The proposed evaluation measures should be able to reflect this categorization, i.e., in general alignments in category $+$ should be assigned a higher confidence score than alignments in category $-$. The latter should in turn receive higher scores than those in category ↯. Results are displayed in Tables 5.12 and 5.13 for 6D ICP and Plane ICP alignment, respectively. In addition Table 5.14 shows the results, if all available alignments in the different classes, independently from the alignment method, are combined. Both tables display the results for the measures c_{nc} that adds negative weights for contradicting clusters and c_{nw}, where continuous regions of contradicting rays add negative weights to the confidence measure.

This way the influence of punishing connected regions that contradict correct alignment can

measure & parameters	class	min	max	mean	MAD	precision	recall	F1-score
c_{nc},	$+$	0.8931	0.9750	0.9503	0.0120	0.7910	0.7465	0.7681
l_L= 15 mm,	$-$	0.5262	0.9560	0.9110	0.0301	0.6377	0.6470	0.6423
$n_{\min} = 10$	$\lightning$	0.3782	0.9217	0.8089	0.0825	0.7778	0.6364	0.700
c_{nw},	$+$	0.8787	0.9647	0.9350	0.0143	0.8060	0.8571	0.8308
l_L= 10 mm,	$-$	0.6427	0.9462	0.8768	0.0364	0.6087	0.7636	0.6774
$n_{\min} = 10$	$\lightning$	0.4211	0.8906	0.7562	0.0853	0.8519	0.6571	0.7419

Table 5.12: Values for different confidence measures for tabletop scenes aligned via 6D ICP. For details on the computation of precision, recall and F1-score, please refer to the accompanying text.

be illustrated. Apart from displaying minimum, maximum and mean for each measure in each class, the mean absolute deviation (MAD) is also displayed, which is defined in the following:

$$\mathrm{MAD} = \frac{1}{n}\sum_{i=1}^{n} |x_i - \bar{x}| \tag{5.1}$$

where $\bar{x}$ denotes the mean for variables $x_1, \ldots, x_n$. The MAD in connection with the mean values are employed as an ad-hoc classifier that associates an alignment with a class, depending on its confidence score. Using MAD rather than the standard deviation σ is due to the latter being much more sensitive to outliers, which do exist in the available data, as will be illustrated in a moment. If we use the class sign ($+$, $-$, $\lightning$) as a subscript to denote mean and MAD in the respective class, the ad-hoc classifier for a confidence value x is defined as

$$\text{class}\,(x_i) = \begin{cases} + & \text{if } x_i \geq \bar{x}_+ - \mathrm{MAD}_+ \\ \lightning & \text{if } x_i \leq \bar{x}_\lightning + \mathrm{MAD}_\lightning \\ - & \text{if } \bar{x}_- + \mathrm{MAD}_- \geq x_i \geq \bar{x}_- - \mathrm{MAD}_- \end{cases} \tag{5.2}$$

This classification can now be employed to compute precision, recall and F1-scores for the respective classes, which are also displayed in Tables 5.12, 5.13 and 5.14.

When assessing the results displayed in Tables 5.12 – 5.14 some things should be taken into consideration: the mean value of each class reflects the initial estimate that one would like to see for a confidence measure for the three given classes, namely $\bar{x}_+ > \bar{x}_- > \bar{x}_\lightning$, independently of employed alignment method or confidence measure. The reported precision, recall and F1-score values illustrate that, even though a really simple ad-hoc classification is used (see equation (5.2)), the generated confidence measures already provide a certain level of distinctiveness. For comparison how good (or bad) these results are please refer to Table 5.15, where the results for 2 naive baseline classifications are shown. The upper three rows in Table 5.15 show precision, recall and F1-score, when a class is chosen according to a uniform distribution, i.e., each class has a probability of 1/3. This is followed by randomly selecting a class, according

measure & parameters	class	min	max	mean	MAD	precision	recall	F1-score
c_{nc}, l_L= 15 mm, $n_{\text{min}} = 10$	$+$	0.8850	0.9791	0.9464	0.0136	0.8506	0.7327	0.7872
	$-$	0.7974	0.9516	0.9191	0.0215	0.6324	0.6232	0.6277
	↯	0.6244	0.9162	0.8158	0.0584	0.7857	0.6111	0.6875
c_{nw}, l_L= 10 mm, $n_{\text{min}} = 10$	$+$	0.7631	0.9622	0.9265	0.0195	0.8391	0.7766	0.8066
	$-$	0.7304	0.9379	0.8837	0.0321	0.6471	0.6984	0.6718
	↯	0.5428	0.8508	0.7573	0.0645	0.6429	0.5625	0.600

Table 5.13: Values for different confidence measures for tabletop scenes aligned via Plane ICP. For details on the computation of precision, recall and F1-score, please refer to the accompanying text.

measure & parameters	class	min	max	mean	MAD	precision	recall	F1-score
c_{nc}, l_L= 15 mm, $n_{\text{min}} = 10$	$+$	0.8850	0.9791	0.9481	0.0129	0.8052	0.7425	0.7726
	$-$	0.5262	0.9560	0.9150	0.0259	0.6423	0.6423	0.6423
	↯	0.3782	0.9217	0.8113	0.0743	0.8049	0.6471	0.7173
c_{nw}, l_L= 10 mm, $n_{\text{min}} = 10$	$+$	0.7631	0.9647	0.9302	0.0174	0.8247	0.8089	0.8167
	$-$	0.6427	0.9462	0.8802	0.0344	0.6496	0.7295	0.6873
	↯	0.4178	0.8905	0.7562	0.0784	0.8537	0.6481	0.7368

Table 5.14: Values for different confidence measures for both alignment methods combined. For details on the computation of precision, recall and F1-score, please refer to the accompanying text.

to the class distribution in the dataset (see bottom row of Table 5.11). Since the ad-hoc classifier performs much better than these baseline approaches, there is evidence that the results of the confidence measures do convey information about the alignment quality.

If we also take into account that transitions between classes, especially between $+$ and $-$ for the relatively noisy RGB-D data are fuzzy in some instances, it might be worthwhile to look at how many misclassifications occurred between the extreme ends of the spectrum, namely class $+$ and ↯. This type of misclassification is very rare, as evident by the results in Table 5.16.

Another point that should be taken in consideration when observing the results is type of data that the proposed confidence measures are applied to. Frames and their alignments are provided from a different experiment, containing the noise typical for RGB-D data and originally not created for the purpose of alignment evaluation, i.e., they do not constitute a dataset that plays to the strong points of the methods that are applied in this case. This was a deliberate decision, since I expected that evidence obtained from this kind of data provides more insight into the usefulness of the proposed methods than tailor-made datasets. Since the topic of automatic alignment evaluation has been neglected until now, no benchmark datasets exist that could be used for the evaluation.

probability	class	precision	recall	F1-score
1/3	$+$	0.4639	0.3333	0.3879
1/3	$-$	0.4127	0.3333	0.3688
1/3	↯	0.1235	0.3333	0.1802
154/332	$+$	0.4639	0.4639	0.4639
137/332	$-$	0.4127	0.4127	0.4127
41/332	↯	0.1235	0.1235	0.1235

Table 5.15: Results of baseline classification for tabletop scenes. First three results assign class randomly with uniform distribution to each scene. Bottom results are obtained by choosing class randomly according to their relative occurence in the dataset.

measure	error type	6D ICP	Plane ICP	combined
c_{nc}, l_L= 15 mm, $n_{\min} = 10$	$+$ as ↯	0	0	1
	↯ as $+$	0	0	0
c_{nw}, l_L= 10 mm, $n_{\min} = 10$	$+$ as ↯	0	1	1
	↯ as $+$	0	0	0

Table 5.16: Statistic of misclassifications between classes $+$ and ↯.

This "non-tailored" data also has some drawbacks, with respect to the evaluation. Firstly, as can be seen in Figures 5.34 – 5.36, the available data consists solely of object clusters that are originally located on a tabletop and only those points were used, when computing the alignment (for more details please refer to [31]). When employing the ray tracing-based approach (section 4.5) to this sparse kind of data, a large number of rays will not intersect any octree leaf and subsequently not contribute to the evaluation. If the full data were available, the filtered out parts of the scene would also be taken into consideration. In the case of alignment errors, where the two frames are aligned at wrong heights, planar surfaces like the tabletops would no longer coincide. Such an arrangement would be reflected in the confidence measures.

Furthermore, one of the underlying assumptions of the alignment evaluation is violated in some instances, namely, that one can expect, measurement points from sensors S_1 and S_2, if both sensors could observe the same surface. Depending on the relative poses of both sensors to the surface, the point density on the surface may vary in $\breve{P}_1$ and $\breve{P}_2$, but the surface should be resembled in both point clouds. Unfortunately this is not always the case with RGB-D data, provided by sensors like Asus X-tion or MicroSoft Kinect under realistic lighting conditions. An example is given in Figures 5.37 and 5.38. As Figure 5.37 illustrates, the overall alignment is satisfactory, given the quality of the data. However, since measurements at different locations are missing on the book in the foreground (details in Figure 5.38), the resulting confi-

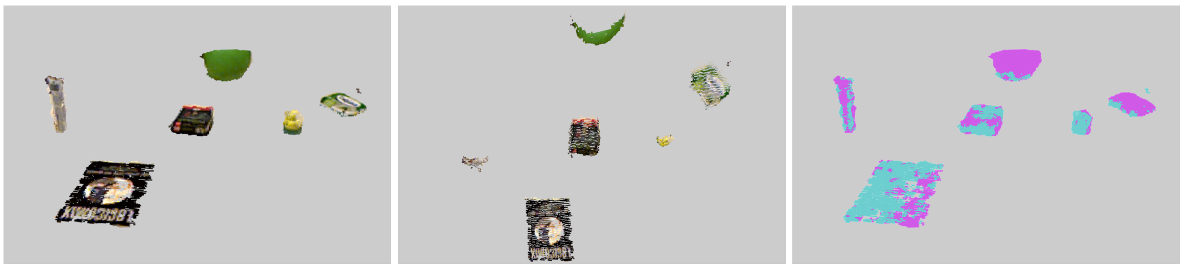


Figure 5.37: Violation of data assumption, overview: display of the complete scene, first in perspective and top down view with color information. Points in the image on the right are colored according to the original point cloud. All three images suggest good alignment, similar to Figure 5.34.

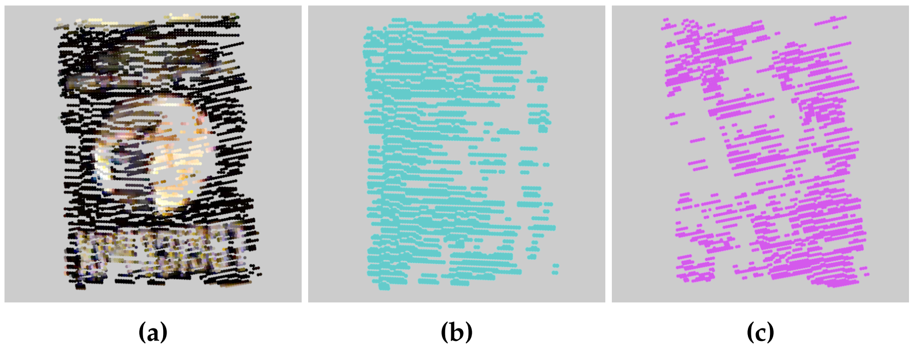


Figure 5.38: Violation of data assumption, detailed view of the book in the foreground of Figure 5.37: **(a)** shows the fused information from two frames in color, while **(b)** and **(c)** show the geometric information of the individual frames. As is evident **(b)** and **(c)** do contain gaps in the measurements at different positions, although there was no occlusion between sensor and object at the time of the measurement. This type of missing data will lead to a lower confidence score.

dence score is lower than expected, since such missing data is treated similar in the evaluation as data discrepancies caused by misalignment. Similar occurrences of missing data appear regularly in the dataset. Another example can be observed in Figure 5.34a with the object in the background, for instance.

As demonstrated by the tabletop object dataset, even under unfavorable conditions the proposed measures are able to distinguish between really good and bad alignments (classes + and ϟ in this case), as evident by Table 5.16. More sophisticated classifiers than the base version defined by equation (5.2) might improve the existing results further and chances are good that such classifiers will work reasonably well in indoor settings that employ RGB-D cameras, since the collected data in such experiments usually is observed from similar measurement ranges – if it proves to be less noisy and contains less missing data points, the performance should only improve.

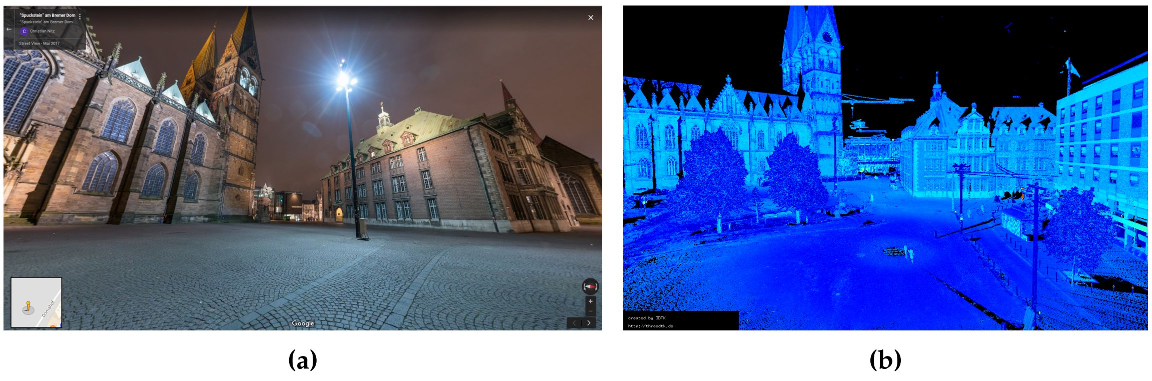


Figure 5.39: Photograph and point cloud of Bremen city scene. **(a)**: a photograph of the plaza behind the Bremen cathedral at night; **(b)** a portion of combined point cloud $P_{02} \cup \tilde{P}_{03}$ of the Bremen city dataset. Photograph taken by Christian Nitz and published on `https://www.google.de/maps`, use in accordance to Google's permission guidelines and "Fair Use" policy.

5.2.2 Bremen City Dataset

But how do the proposed methods perform on different types of data in terms of quality and measurement range? To investigate this question, let us take a look at the "Bremen city" dataset (freely available at [81], dataset 12) recorded with a Riegl VZ-400 sensor, that is on the other end of the available spectrum, when set into relation the low-cost RGB-D sensors. The dataset consists of a total of 13 3D scans, taken in the city center of Bremen. A visualization of part of the point cloud data is shown in Figure 5.39b, where the color for the point cloud data is derived from the reflectance values of the measurement points. After testing the proposed measures on the Bremen City datasets, the same parameter settings are applied to a different dataset that was recorded with a Riegl VZ-400i (the successor model of the VZ-400), but at a different location, containing more flora than the Bremen City dataset. This was done to showcase that parameters for given sensors generalize reasonably well and are not specific to one single dataset.

Initial alignment of the scans was obtained by using reflecting markers as corresponding point pairs. The quality of the marker registration when observing consecutive scans has a varying degree. While some scans are aligned very well, three scan pairs stand out in particular, where the marker registrations proved to be inaccurate, namely, the alignment between scan pairs `scan001` and `scan002`, `scan002` and `scan003` and pair `scan005` and `scan006`. While large portions of the recorded points in the associated combined point clouds coincide, there also exist larger regions that contradict the notion of sound alignment.

To improve on the alignment provided by using markers, the authors of the dataset proceeded to use these poses as initial pose estimates for pairwise ICP registration with a subsequent global relaxation [13]. Sequential registration and subsequent global optimization via the 6 DoF extension of Lu & Milios were performed on reduced version of the point clouds, employing octrees with $l_L = 0.1\,\text{m}$ for reducing the point cloud data. More detailed information concerning the matching parameters is available in the dataset at [81]. When observing the

					other scenes		
alignment	measure	$P_{01} \cup \tilde{P}_{02}$	$P_{02} \cup \tilde{P}_{03}$	$P_{05} \cup \tilde{P}_{06}$	mean	min	max chg
		$l_L = 0.1\,\mathrm{m}$					
marker-based	c_{nc}	0.0284	0.0589	0.0534	0.9016	0.6343	0.6343
	c_{nw}	0.7692	0.8727	0.7300	0.9769	0.9609	0.9769
ICP & global optimization	c_{nc}	0.9401	0.9569	0.9371	0.8945	0.7695	0.8370
	c_{nw}	0.9778	0.9782	0.9543	0.9651	0.9174	0.9174
		$l_L = 0.2\,\mathrm{m}$					
marker-based	c_{nc}	0.4443	0.3028	0.4599	0.9467	0.8758	0.8948
	c_{nw}	0.6158	0.8026	0.7191	0.9703	0.9423	0.9673
ICP & global optimization	c_{nc}	0.9696	0.9716	0.9656	0.9453	0.8617	0.9455
	c_{nw}	0.8949	0.9392	0.8681	0.9678	0.9281	0.9281

Table 5.17: Evaluation results on the Bremen city dataset. Threshold n_{min} to consider octree leaves in the evaluation was set to 5.

obtained overall result, the registration is noticeably improved compared to the result obtained from the markers. If the proposed confidence measures are sensible, they should also reflect this change, when comparing initial and final poses. Especially for the aforementioned three scan pairs confidence results should improve, while they should not vary much for the other point cloud pairs, since those were initially already very well aligned. Notice that due to the global optimization step that is performed after sequential ICP alignment, alignment between two subsequent scans may also slightly disimprove, since in this step not only the local information of these two scans is considered, but also the information of other scans. Table 5.17 shows the results of the evaluation for the Bremen City dataset.

For the results shown in Table 5.17, the ray tracing approach (c_{nw}) with octree resolutions of $l_L = 0.1\,\mathrm{m}$ and $l_L = 0.2\,\mathrm{m}$ were chosen and the threshold for considering a leaf to be occupied was set to $n_{\mathrm{min}} = 5$. The relatively small octree resolution, when set into relation with the measurement range of the sensor (maximal measured distance in the Bremen city dataset is 457.89 m), ensures that a relatively high level of detail is retained. Simultaneously only considering octree leaves that contain at least 5 data points removes the individual stray measurements from the evaluation. The same parameters were also applied for the direct modelling approach that punished clusters indicating misalignment (c_{nc}) and worked reasonably well, as Table 5.17 indicates. In addition to the alignment between the specifically aforementioned scans, Table 5.17 also conveys the change in the confidence measures between initial alignment via markers and the combined ICP and slam6D refinement. This done by the last three columns in the table: The first of these reports the average confidence measures of all other scenes, employing the specified alignment. This is followed by the minimal confidence value of all other scenes. Note that the scans to which this refers to, might not be the same for initial and final

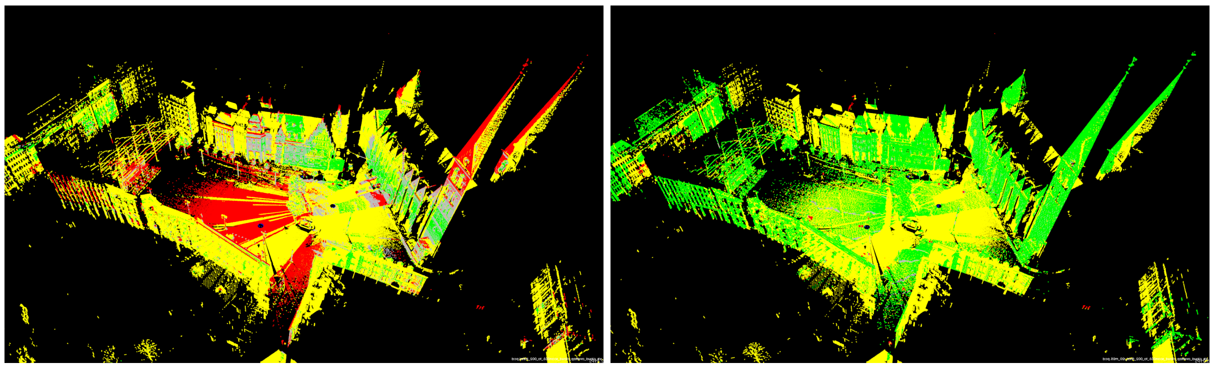


Figure 5.40: Visualization of c_{nc} for $P_{02} \cup \tilde{P}_{03}$ of Bremen city dataset: initial alignment shown in **(a)**, while **(b)** depicts the final registration obtained by ICP. Details concerning the color information are provided in Table 4.4.

alignment. The last column states the confidence value of the other scenes where the largest change occurred between initial registration and final alignment (apart from $P_{01} \cup \tilde{P}_{02}$, $P_{02} \cup \tilde{P}_{03}$ and $P_{05} \cup \tilde{P}_{06}$). This additional information helps to put the reported confidence values of the three explicitly reported alignments into perspective and illustrate that these actually stick out, like the quality of the corresponding initial alignments.

The perceived improvement of the alignment are clearly reflected by the confidence values in Table 5.17. While the average values for all other pairwise alignments did not change much between the marker registration and the final result obtained after ICP and slam6D, a strong increase in the confidence values for the initially suboptimally aligned combined point clouds is evident. Not only are the shown initial confidence values for these three alignments substantially smaller than for the other alignments, but also the change between initial and final alignment is much larger than for the other point clouds.

Figure 5.40 visualizes the results for c_{nc} with parameters $l_L = 0.2\,\mathrm{m}$ and $n_{\mathrm{min}} = 5$. On the left side (Figure 5.40a) the result for the initial alignment is shown. The two bell towers on the right side of the depicted point cloud correspond to the cathedral shown in Figure 5.39. According to the color information (see Table 4.4 for details), the evaluation detected large areas of misalignments at the bell towers and at parts of the plaza. For the bell towers the imperfect alignment is clearly visible (the layer of yellow points behind the red points). When observing the result after the ICP improvement (Figure 5.40b) the large areas of misalignment have disappeared.

A visualization for ray tracing based confidence measure c_{N} with parameters $l_L = 0.2\,\mathrm{m}$ and $n_{\mathrm{min}} = 5$ is provided in Figure 5.41. The displayed data shows combined the point cloud $P_{02} \cup \tilde{P}_{03}$ in marker alignment (top row) and refinement (bottom row) from the point of view of both sensors. It can be clearly seen, that in the initial alignment parts of the ground do not seem to align well, when observed from sensor S_1, since oftentimes measurements of sensor S_2 are placed on top of them (Figure 5.41a). Also one building in the background, right of the center does not seem to be aligned well. From the perspective of sensor S_2 mainly the visible tower of

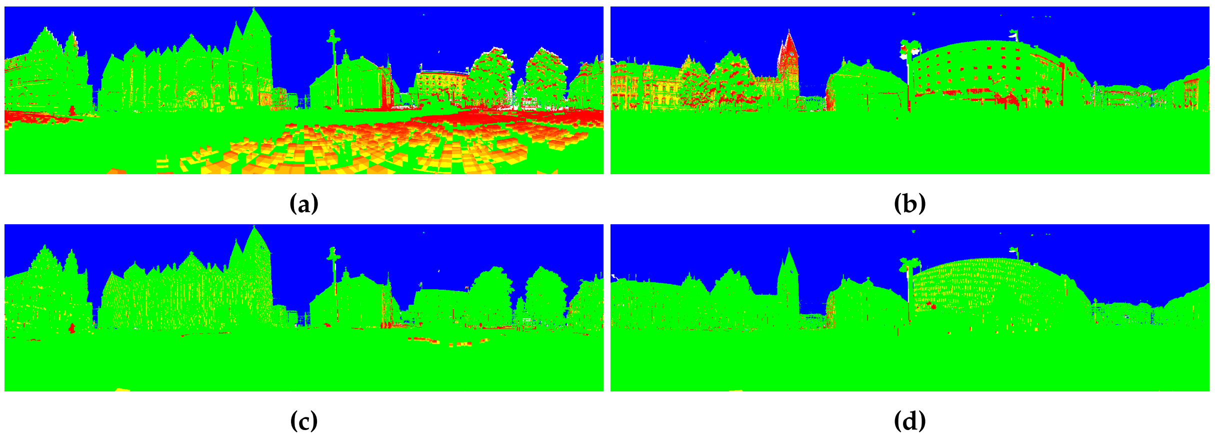


Figure 5.41: Visualization of c_{nw} for $P_{02} \cup \tilde{P}_{03}$ of Bremen city dataset: **(a)** and **(b)** show result for initial alignment for sensor S_1 and S_2, respectively. In **(c)** and **(d)** the results after ICP and slam6D alignment are displayed.

the cathedral in the background to the left is clearly aligned imperfectly (Figure 5.41b).

These large areas of detected misalignment disappear in the bottom row, but some small areas of detected imperfection remain. Partially these can be explained by a minor violation of the initial assumptions for the alignment evaluation, namely, that the scene is static. The scans of the Bremen city dataset were recorded during day, so moving objects (people, cars, tram) were present in the scene. These can, dependent on their relative position towards the sensor that did not record them, be detected as data inconsistencies that would suggest some sort of misalignment. Furthermore, while the data in the Bremen city dataset is of much higher quality, when compared to the tabletop scenes, it still contains some invalid points, mainly caused by refraction of the laser beams and producing multiple echoes. The occurrence of such refractions is dependent on the relative position of the scanner to a glass surface, for example a large window-front, and therefore these points are usually also only present in individual point clouds. For the meaning encoded in the colors of Figure 5.41, please refer to Table 4.6.

While Table 5.17 contains the information for the dense point clouds (containing on average 16.59 million points), the refinement of the alignment for the Bremen city dataset was performed on reduced versions of the point clouds, resulting in an average of 857 thousand points per cloud. Table 5.18 lists the confidence values that were computed on the reduced point clouds. However, when computing the confidence measures for the reduced versions of the point clouds, the parameter settings should be adapted: maintaining the octree resolution at $l_L = 0.1\,\mathrm{m}$ and discarding all leaves from the evaluation that contain less than 5 points is no longer sensible, when the original data was reduced by an octree that had its resolution set to the same value (so that in a good alignment a maximum of 2 points per $0.001\,\mathrm{m}^3$ should be present). To account for the reduced point density, parameters for the confidence measures were changed to $l_L = 0.2\,\mathrm{m}$ and $n_{\min} = 2$. These parameter settings ensure that in regions that contained sufficient measurements, i.e., at least two measurements of the reduced combined point cloud were present in the $0.2\,\mathrm{m} \times 0.2\,\mathrm{m} \times 0.2\,\mathrm{m}$ volume, are considered without automat-

					other scenes		
alignment	measure	$P_{01} \cup \tilde{P}_{02}$	$P_{02} \cup \tilde{P}_{03}$	$P_{05} \cup \tilde{P}_{06}$	mean	min	max chg
marker-based	c_{nc}	0.4501	0.3542	0.4834	0.9275	0.8378	0.8378
	c_{nw}	0.5593	0.7317	0.7049	0.9370	0.8880	0.9402
ICP & global optimization	c_{nc}	0.9418	0.9570	0.9371	0.9363	0.9165	0.9207
	c_{nw}	0.8949	0.9392	0.8681	0.9406	0.8746	0.9625

Table 5.18: Evaluation results on the reduced Bremen city dataset. For the displayed values parameters $l_L = 0.2\,\text{m}$ and $n_{\text{min}} = 2$ were used.

ically ignoring leaves that only contain measurements from one source. The latter, for instance, would occur, if parameters were set to $l_L = 0.1\,\text{m}$ and $n_{\text{min}} = 2$, when investigating the reduced point clouds.

Similar to the results for the unreduced point clouds, the results in Table 5.18 show a clear distinction between the three alignments that were previously identified as improvable and the remaining alignments. While the confidence measures for the alignments of $P_{01} \cup \tilde{P}_{02}$, $P_{02} \cup \tilde{P}_{03}$ and $P_{05} \cup \tilde{P}_{06}$ increased noticeably after the pose refinement was applied (minimum increase of 0.4537 and 0.1632 for c_{nc} and c_{nw}, respectively), the evaluation for the other alignments much less (maximum of 0.0829 and 0.0223, respectively).

Evaluation results are visualized in Figures 5.42 and 5.43, respectively. Both are qualitatively very similar to the results on the unreduced data shown in Figures 5.40 and 5.41. Since only those parts of the point clouds are visualized in Figures 5.40 and 5.42, that are considered occupied, according to threshold parameter n_{min}, and the former shows the unreduced point clouds, while the latter displays a reduced version, minor dissimilarities in the shown data are to be expected. The slightly increased red areas in Figure 5.43c when compared to Figure 5.41c might be due to different positioning of the octree and which leaves are considered to be occupied.

The results obtained on the Bremen city dataset suggest that the proposed confidence measures are not restricted to small close-scene scenarios, but can also be successfully applied to larger scale data, if parameters are chosen sensibly. Furthermore, the less noisy data of the laser scanner improves the ability of the confidence measures to discern between good and suboptimal alignments. The differences between the initial alignments of $P_{01} \cup \tilde{P}_{02}$, $P_{02} \cup \tilde{P}_{03}$ and $P_{05} \cup \tilde{P}_{06}$ and their final registrations were on a clearly different scale than the changes that occurred between the other alignments

In this way the confidence measures on both, full and reduced data, reflect the impression of a human observer that the 3 explicitly listed alignments were significantly improved while the other alignments were not modified substantially (when observed in the sequential pairwise context).

Judging from the computed confidence values for the Bremen city dataset the confidence measures were capable to distinguish between good and rather not so good alignments with the reported parameter values. However, how well do these parameters generalize when ap-

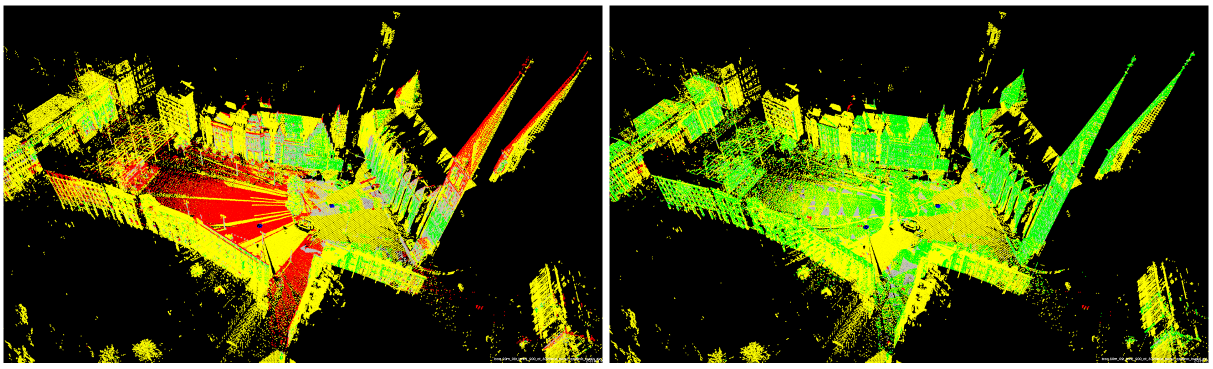


Figure 5.42: Visualization of c_{nc} on reduced data of $P_{02} \cup \tilde{P}_{03}$: **(a)** displays initial alignment; final registration results shown in **(b)**

plied to similar settings? Do they also produce sensible results or is it necessary to establish suitable parameters for each scenario anew, which would vastly diminsih a meaningful applicability of these measures? To exemplarily address the issue of re-usability of parameters, another set of scans was investigated, using the same settings for confidence measures as in the Bremen city dataset.

The scans featured in this dataset were taken from a historic building used as a headframe for a former coal mine (Haseschacht near Osnabrück). Recording and registration of the data was part of the joint research project 3DinOS [1] with the institute for cultural heritage of Osnabrück University (`https://www.kunstgeschichte.uni-osnabrueck.de/`) about the digitalization of historic buildings that are relevant in the context of cultural heritage. Technical aspects of the joint project are discussed and introduced in [78], albeit in German. Photographs of the side of the building that were subject to the evaluation are shown in Figure 5.44.

The scans were obtained, using a Riegl VZ-400i, hence a sensor that is very similar to the one that recorded the Bremen city dataset, and measurements were taken outdoors. Apart from the building on top of the mineshaft the scene consisted mainly of the forest area surrounding the building. Matching was this time only performed sequentially, but in a two step process: from their initial poses scans were registered using a reduced version of the point cloud (reduction via octree with $l_L = 0.1\,\mathrm{m}$) and the maximal distance for point-to-point correspondences was set to 1 m. Since the scene contained a lot of trees at larger distances, that cannot be assumed as completely static, all measurements that were farther away than 40 m from the scanner's position were ignored during the matching process. The result of this step was followed by a further refinement, where point data was not reduced, the maximal point-to-point distance was decreased to 0.1 m and no measurements farther away than 25 m were considered during matching. For the first two scans the initial pose was specified, relatively decently, by hand, while all other initial alignments were obtained by GPS information.

A visual inspection revealed that scans were already well aligned after the first ICP step

[1]BMBF-funded, grant no. 01UG1641X

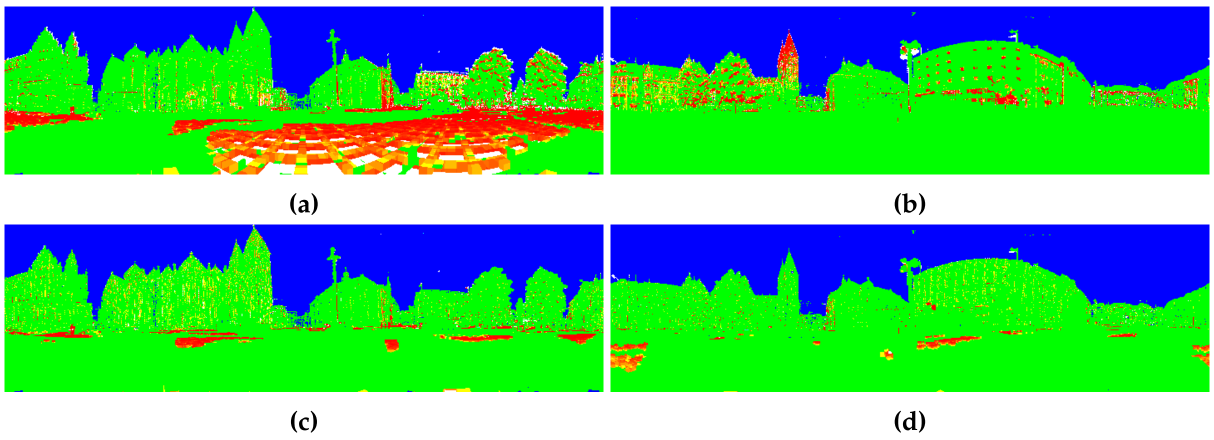


Figure 5.43: Visualization of c_{nw} on reduced data of $P_{02} \cup \tilde{P}_{03}$: top row with **(a)** and **(b)** displays results of marker based alignment while **(c)** and **(d)** in the bottom row show the results after ICP and slam6D improvement.

was performed, but the following refinement still improved the registration a bit further, evident at some locations like window sills in the scans. Since no global optimization took place, one expects an increase in the confidence values after each ICP alignment is completed. The first improvement should be rather substantially, since the GPS estimation was not too accurate and ideally this will be followed by a rather small improvement. An exemplary visualization of the scan alignment is provided in Figure 5.45 for point clouds P_{01}, shown in cyan, and P_{02} (magenta). While the change from Figure 5.45a to 5.45b is immediately noticeable and an improvement, differences between 5.45b and 5.45c are more subtle. Vertical walls, for instance, are slightly better aligned in the latter, visible by their less uniform coloring.

Table 5.19 shows that these improvements are also reflected in the confidence values for all scans, when using the same parameters for l_L and n_{min} as in the Bremen city dataset. Values, independently from the type of measure increase after each registration step is performed, also the better manual initial alignment of scans P_{00} and P_{01} is clearly distinct from the other initial alignments. As expected, the general behaviour, that could be observed for the Bremen city scans, when increasing parameter l_L from 0.1 m to 0.2 m, namely an increase in the confidence values, is replicated for this dataset. Even tough, due to the coarser scale on which the alignments are evaluated with the increased leaf size, alignments are assigned a more generous score, the computed confidence values are still substantially different for the initial GPS alignments ($P_{01} \cup \tilde{P}_{02} - P_{04} \cup \tilde{P}_{05}$) when compared to all other alignments. However, the distinction between the initial alignment by hand and the subsequent improvements become less noticeable. Interestingly these are the most visible in ray tracing based measure c_{nw}, while in the other cases the change from the initial alignment for measure c_{nc} is much larger.

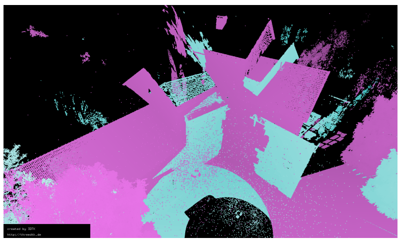
Also similar to the Bremen city dataset, confidence measures for the reduced versions of the scans (see Table 5.20) report the same findings as the unreduced original data, i.e., an improvement occurred after each ICP step.

Figures 5.46 and 5.47 provide visualizations of measures c_{nc} and c_{nw}, respectively. In the

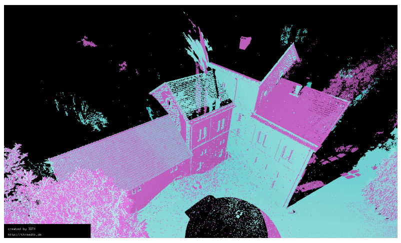
(a)

(b)

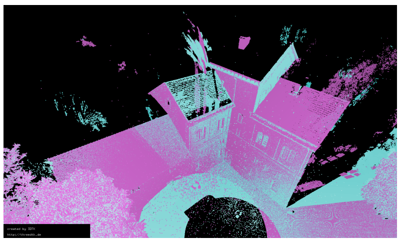
Figure 5.44: Photograph of the Haseschacht headframe building **(a)** and of scan acquisation **(b)**. Photograph **(a)** taken by Christian Bergstedt and published on `https://www.google.de/maps`, use in accordance to Google's permission guidelines and "Fair Use" policy.



(a)



(b)



(c)

Figure 5.45: Different alignments of P_{01} and P_{02} from Haseschacht dataset: **(a)** shows initial alignment via GPS information, **(b)** the result of the first ICP improvement and **(c)** the final result.

initial alignment point clouds P_{01} and P_{02} are poorly arranged, which is evident by the large red (contradicting) and yellow areas (occluded) in Figure 5.46a. Apart from a small segment on one of the walls of the building only very little evidence for correct alignment is detected. The visualization of the first ICP alignment (Figure 5.46b) illustrates the improvement in the confidence values in Table 5.19: most red regions have disappeared and the walls of the building are mainly colored green, although they still contain some indications of imperfect alignment (stray yellow areas on the vertical walls). One region that still indicates misalignment is composed of measurement points that were produced by smoke from the chimney – however, these points violated the static scene assumption. The improvement from Figure 5.46b to 5.46c are much more subtle. Some of the yellow regions on the walls of the building have vanished and the yellow stripes on the ground in the lower right corner of Figure 5.46b have also disappeared, visualizing the improved alignment obtains by the additional refinement step.

alignment	measure	$P_{00} \cup \tilde{P}_{01}$	$P_{01} \cup \tilde{P}_{02}$	$P_{02} \cup \tilde{P}_{03}$	$P_{03} \cup \tilde{P}_{04}$	$P_{04} \cup \tilde{P}_{05}$
		$l_L = 0.1\,\text{m}$				
initial GPS	c_{nc}	0.9691*	-0.3942	-0.2042	-0.0073	0.5361
	c_{nw}	0.8881*	0.3549	0.3980	0.5088	0.5254
first ICP step	c_{nc}	0.9905	0.9833	0.9759	0.9114	0.9795
	c_{nw}	0.9574	0.9299	0.9090	0.8998	0.8969
second ICP step	c_{nc}	0.9954	0.9958	0.9964	0.9928	0.9972
	c_{nw}	0.9771	0.9748	0.9808	0.9647	0.9761
		$l_L = 0.2\,\text{m}$				
initial GPS	c_{nc}	0.9966*	0.2551	0.5063	0.3955	0.8269
	c_{nw}	0.9426*	0.7205	0.7229	0.6451	0.6661
first ICP step	c_{nc}	0.9988	0.9981	0.9972	0.9867	0.9958
	c_{nw}	0.9660	0.9702	0.9663	0.9344	0.9430
second ICP step	c_{nc}	0.9994	0.9992	0.9996	0.9978	0.9993
	c_{nw}	0.9735	0.9813	0.9864	0.9711	0.9813

Table 5.19: Evaluation results on the Haseschacht building dataset. Threshold n_{min} to consider octree leaves in the evaluation was set to 5. The marked initial alignment for scans P_{00} and P_{01} was not obtained by GPS, but by hand.

Figure 5.47 shows the visualization for the ray tracing-based approach. The left column displays the view from sensor S_1 that recorded P_{01}, while the right side shows the results from the point of view of S_2. From top to bottom initial alignment, alignment after first ICP improvement and final alignment are shown. The visualization of the ray tracing based approach is less easy to interpret, when compared to the Bremen City dataset (Figure 5.41). This is mainly due to the less structured nature of the scene. As opposed to the Bremen City data, that contained many approximately flat surfaces like house fronts, the Haseschacht scans contain only measurements from a single building and the remainder of the measurements is taken from trees, bushes and the like. In P_{01} (Figures 5.47a, 5.47c and 5.47e) the building is located slightly right of the center and the left side of the images consist of trees. During the acquisition of P_{02} the sensor was positioned in a way that a part of the building is located left side of the images (Figures 5.47b, 5.47d and 5.47f) and on the right side of the images, while the majority of the center is occupied by trees (both borders are connected due to horizontal opening angle $\omega_{\text{h}} = 360\,^{\circ}$ for these scans). Improvements from top to bottom are clearly visible: the amount of connected red regions is reduced in each step and the second ICP step (transition from middle row to bottom row) removes some scattered yellow regions in near the center (left column) and on the right side (right column), respectively. Small green clusters in the sky in both point clouds are likely caused by bright sunlight during data acquisition (see also Figure 5.44b) and other influences like chimney smoke, which can also be observed in the point clouds in Figure 5.45.

alignment	measure	$P_{00} \cup \tilde{P}_{01}$	$P_{01} \cup \tilde{P}_{02}$	$P_{02} \cup \tilde{P}_{03}$	$P_{03} \cup \tilde{P}_{04}$	$P_{04} \cup \tilde{P}_{05}$
initial GPS	c_{nc}	0.9952*	0.1244	0.4517	0.2610	0.7784
	c_{nw}	0.8535*	0.7329	0.7463	0.6614	0.6696
first ICP step	c_{nc}	0.9984	0.9977	0.9971	0.9817	0.9937
	c_{nw}	0.8758	0.9368	0.9478	0.9184	0.9188
second ICP step	c_{nc}	0.9989	0.9985	0.9989	0.9934	0.9982
	c_{nw}	0.8848	0.9440	0.9598	0.9503	0.9632

Table 5.20: Evaluation results on the reduced Haseschacht building dataset. Threshold $n_{\min}$ to consider octree leaves in the evaluation was set to 2 and octree resolution $l_L = 0.2\,\text{m}$. The marked initial alignment for scans P_{00} and P_{01} was not obtained by GPS, but by hand.

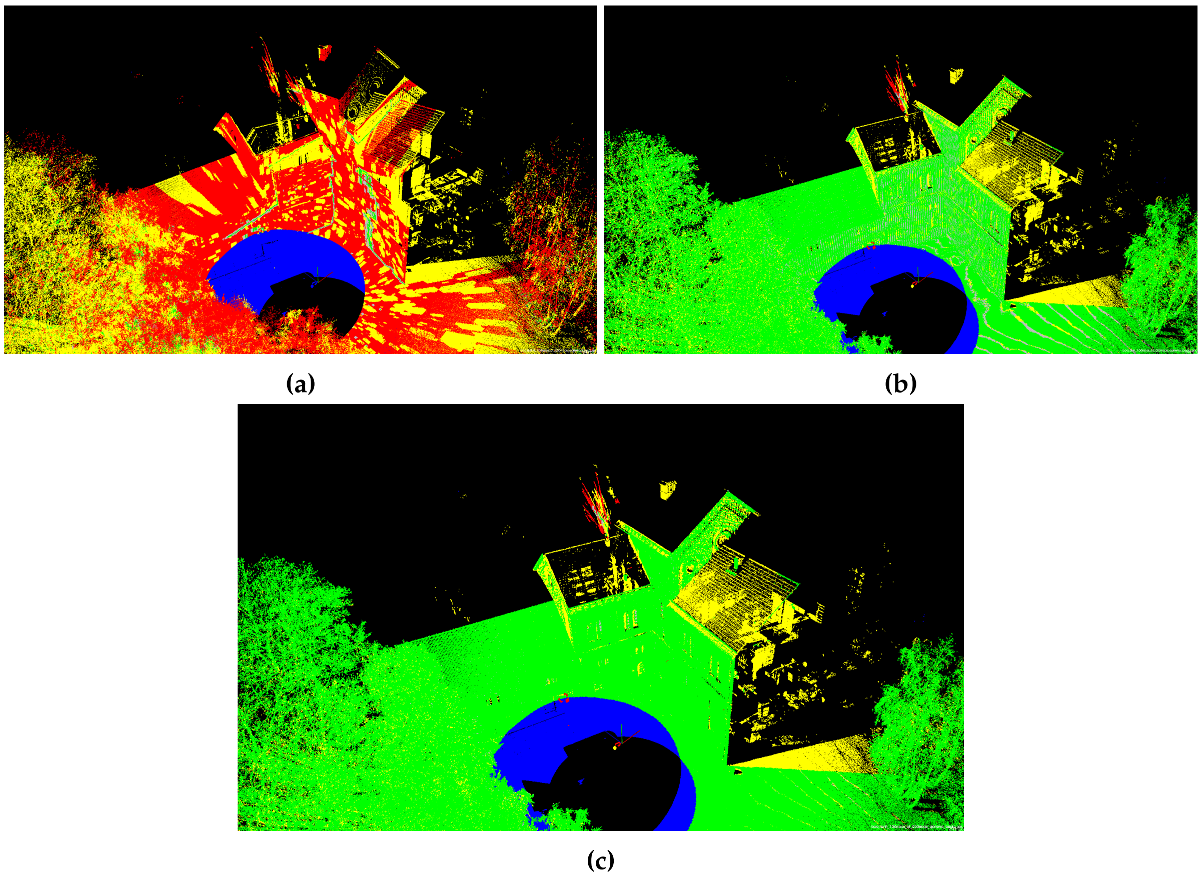


Figure 5.46: Visualization of c_{nc} of $P_{01} \cup \tilde{P}_{02}$ of Haseschacht dataset. **(a)** shows initial alignment via GPS, **(b)** alignment after first run of ICP and **(c)** the result obtained by further refinement. Threshold $n_{\min}$ was set to 5 and octree resolution to $l_L = 0.1\,\text{m}$.

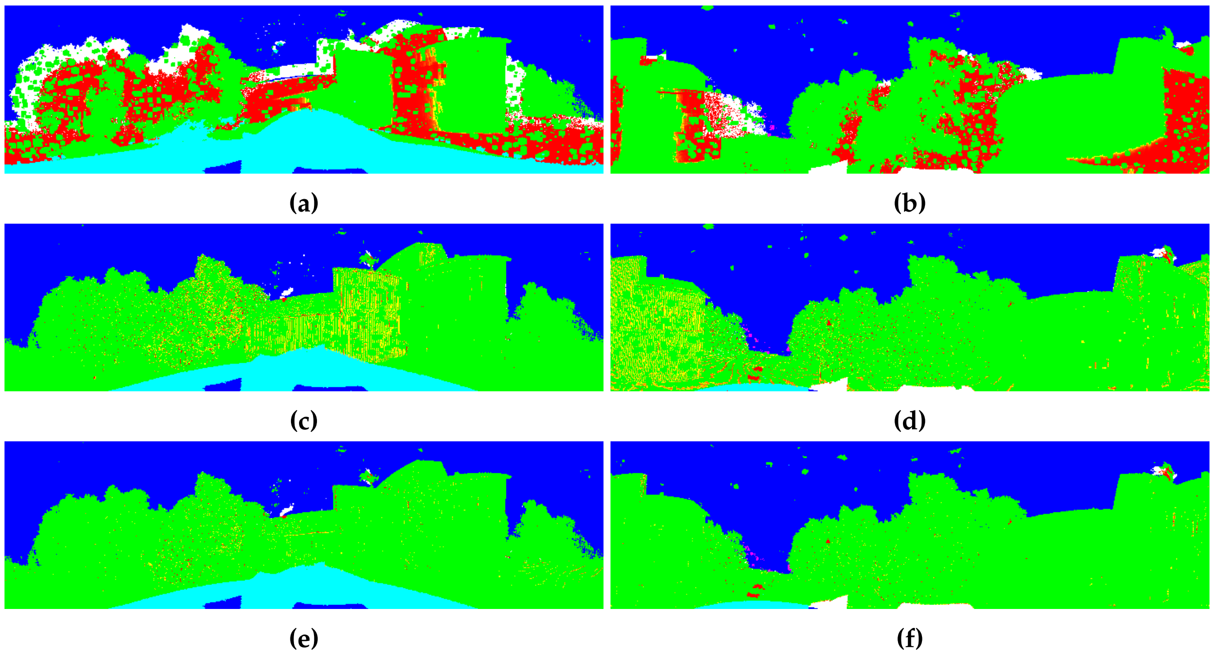


Figure 5.47: Visualization of c_{nw} for $P_{01} \cup \tilde{P}_{02}$ of Haseschacht dataset. From top to bottom: initial alignment, alignment after first ICP registration, alignment after subsequent ICP refinement. Left column shows results of the analysis for S_1 (that recorded P_{01}); right column the visualization for S_2 ($\tilde{P}_{02}$). Parameters l_L and $n_{\min}$ were set to 0.1 m and 5, respectively.

Between the initial alignment and the first refinement, P_{02} was lowered quite a bit, in relation to P_{01}. The transition of the magenta point cloud in Figures 5.45a and 5.45b shows this clearly, most evident at the location of the two distinct gables. This can also be observed in the change from Figure 5.47a to 5.47c. The large cyan area at the bottom is decreased in size between both images. According to Table 4.6 this color indicates that a confirmed ray was not in the field of view of the other sensor, in this case they are located in the blind spot underneath sensor S_2 in its initial pose. The lower position of sensor S_2 after the ICP improvement decreases this blind spot, hence the cyan area is reduced.

Figure 5.48 shows the visualization of the direct modelling approach (c_{nc}) of the reduced point clouds. The results are very similar to the unreduced case (Figure 5.46), therefore displayed only in a smaller scale. However, the visualization of the ray tracing-based approach for the reduced point clouds (Figure 5.49) differs from the unreduced version, shown in Figure 5.47. Correspondences between pixels in the visualization and the parts of the environment that were intersected by the associated rays are much harder to identify than in the unreduced case. Especially in the upper part of the images in Figure 5.49 a large number of green clusters are located in the blue area, that represents open sky in the scene (see Figures 5.45 and 5.47). As mentioned already for the unreduced visualization some erroneous measurements were caused by the bright sunlight. Since these are mostly scattered in the open space, they were rarely visible in the analysis of the unreduced point clouds, since one octree leaf with an edge length of 0.1 m needed to contain at least 5 of such measurements to be considered occupied

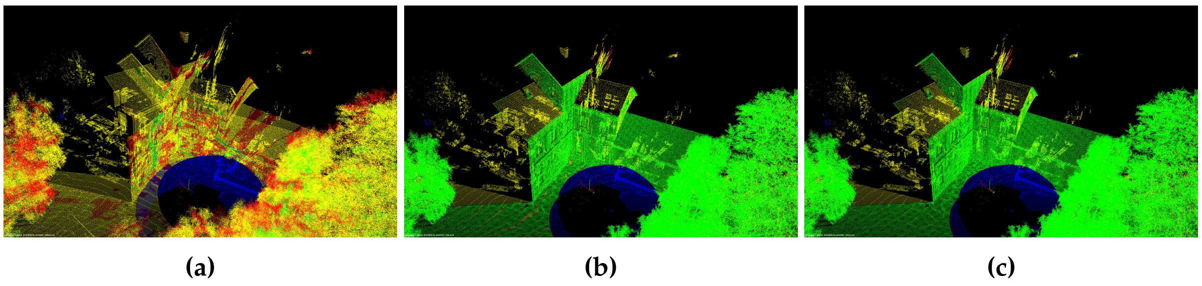
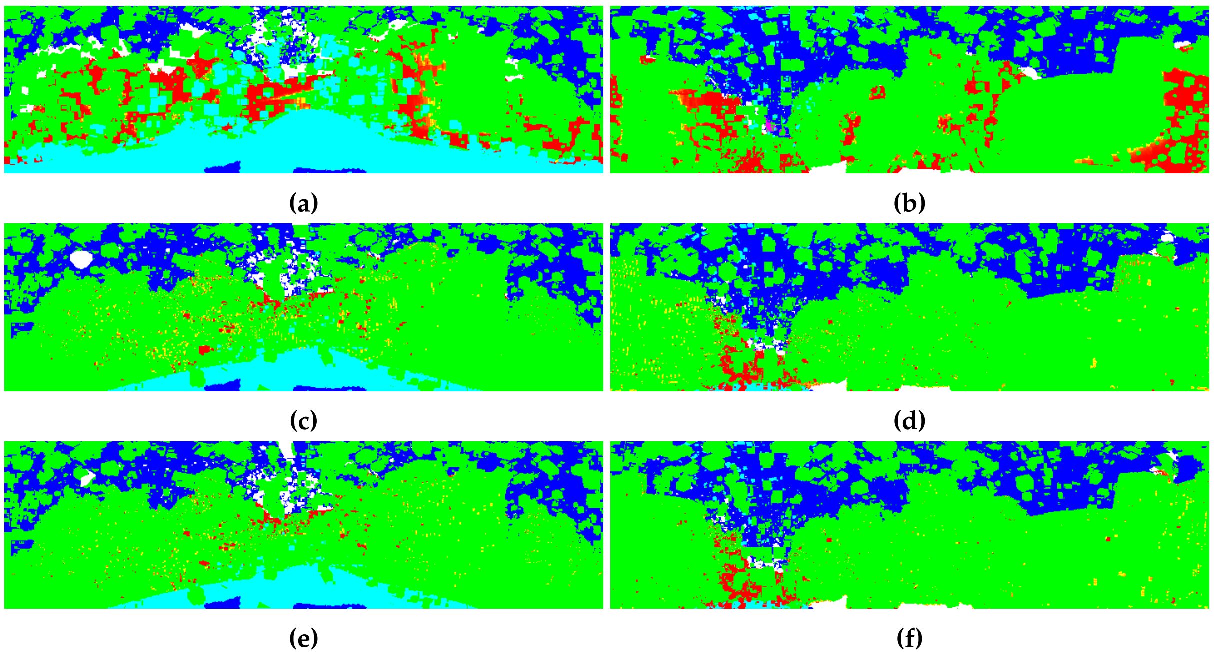


Figure 5.48: Visualization of c_{nc} for reduced version of $P_{01} \cup \tilde{P}_{02}$ of Haseschacht dataset. **(a)** shows initial alignment via GPS, **(b)** alignment after first run of ICP and **(c)** the result obtained by further refinement. Threshold n_{min} was set to 2 and octree resolution to $l_L = 0.2\,\mathrm{m}$.

for the analysis. In case of the reduced point clouds parameter l_L was increased to 0.2 m while n_{min} was reduced from 5 to 2. Therefore, the probability of erroneous measurements to occupy an octree leaf in this case was increased, leading to the somewhat confusing visualization in Figure 5.49.

This behavior can be prevented to a certain degree, if a parameter similar to n_{min} is employed during the reduction of the point cloud. Only those parts of the point cloud are represented in the reduced version, where the octree of the original point cloud contains at least a given specified number of measurement points. On the other hand, such an approach might remove desired parts of the point clouds that feature a low point density. Despite the appearance of visualization in Figure 5.49 in general the same observations can be made as in the unreduced case: the size of red (and white) clusters is decreased clearly from the top row to the middle row and slightly when comparing the middle row with the bottom row. Furthermore, the change in the pose estimation for point cloud P_{02} is also evident in the reduction of the cyan area that marks parts outside the field of view of sensor S_2 from Figure 5.49a to 5.49c.



(a) (b) (c) (d) (e) (f)

Figure 5.49: Visualization of c_{nw} for reduced version of $P_{01} \cup \tilde{P}_{02}$ of Haseschacht dataset. From top to bottom: initial alignment, alignment after first ICP registration, alignment after subsequent ICP refinement. Left column shows results of the analysis for S_1 (that recorded P_{01}); right column the visualization for S_2 ($\tilde{P}_{02}$). Parameters l_L and n_{min} were set to 0.1 m and 5, respectively.

Chapter 6

Conclusion & Outlook

6.1 Conclusion

This thesis introduced approaches to two different, at first, unrelated topics, namely the detection and reconstruction of transparent objects and a measure evaluating alignment quality of two registered point clouds. When these two issues are investigated as proposed here, they turn out to be closely related: in both cases the relative viewpoint inherent information, while observing the scene turns out to be crucial, as well as the consideration of in-scene occlusion. Last, but not least, both approaches actively utilize perceived inconsistencies in the recorded data, as opposed to the probably more intuitive way to use data consistency as the basis of some method. Let us briefly revisit both topics under consideration of these common points.

In case of the detection and reconstruction of transparent objects, the `NaN`-regions inside the convex hull of the support plane constitute to these data inconsistencies (or more generally, connected regions that contain no measurements and are not caused by in-scene occlusion), as detailed in section 3.2. Subsection 3.3.1 shows how the information of a single viewpoint inside a single frame enables the creation of occlusion frusta, one for each transparent object in the scene, that have their apex point at the sensor's location and introduces the concept of mutual occlusion. The latter allows to approximate the shape of the transparent objects, when the scene is observed from multiple viewpoints and the information is registered consistently. This is done via the intersection of the occlusion frusta (see 3.3.2) and subsections 5.1.2 and 5.1.2 demonstrated the validity of this approach under the assumption of very well-aligned frames.

In order to gain a more robust reconstruction method that will be able to cope with less precise scene alignment, only partial observance of the relevant scene, occlusion, "false" `NaN`-regions (not caused by transparent objects) and (partially) missing `NaN`-regions for the existing transparent objects, the reconstruction based on *viewpoint intervals* was introduced in 3.3.3. This method was tested in a series of experiments in 5.1.3, where the reconstructed transparent objects were compared to the reconstructions obtained by painted versions and measures with the same sensor. To put both reconstructions into perspective, a "ground truth" instance of each object was also created from the painted instance and measurements obtained from a much more precise sensor. Comparison was not only performed in a qualitative manner, but also quantitatively, which is a novelty for reconstructing of transparent objects from RGB-D data.

After this brief synopsis of the transparent object reconstruction, let me return to the pose evaluation to emphasize the commonalities between both topics. The different evaluation methods for the registration of two point clouds, introduced in chapter 4 all have in common that they initially assume that the given registration is correct. Subsequently, they search for evidence that refutes this initial assumption via data inconsistencies. Connected to the inconsistencies is the concept of *mutual observability*, the dual to the mutual occlusion of the transparent object reconstruction. These data inconsistencies manifest in two ways, dependent on the modelling approach. In case of the *direct modelling*, described in section 4.4, an inconsistency is present, if a volume of the combined scene contains only sensor measurement from a single sensor. This criterion, however, would not be sufficient on its own and needs to be enriched by information concerning the viewpoint of each sensor. This is realized by the *region of overlap* that incorporates sensor specific field of view constraints and their relative poses to each other. Analogously to the detection of potential locations of transparent objects in unorganized point clouds (see 3.2.1), in-scene occlusion also needs to be regarded, to determine if there actually is an inconsistency in the data.

When considering the *Ray Tracing Approach* (section 4.5), data inconsistencies are discovered, when a ray cast from the location of a sensor does not first encounter measurement from the same sensor. The rays that are emitted from a sensor are, of course, dependent on the field of view of the sensor and the pose of the sensor in the combined point cloud. In this way, the viewpoint information is needed to cast the sensor rays in a sensible fashion. In-scene occlusion is implicitly handled in this method, since rays are only investigated till they encounter the first measurement that is taken from the same sensor – parts of the scene that are located farther away from the sensor and would subsequently be occluded by the encountered measurement, are not considered in this approach. On both of the general modelling ideas several measures were proposed and implemented and evaluated in section 5.2 on fundamentally different types of data. While not directly modelled as such, the idea behind the occlusion frusta is very similar to the ray casting performed in the pose evaluation. Instead of casting rays, according to the sensor's properties, into the scene, line segments from the border points of `NaN`-regions are used to create line segments together with the sensor's location – which is inverse ray casting, so to speak.

Further similarities between transparent object reconstruction and pose evaluation emerge, when the implementation is considered. For several reasons the octree serves as the underlying data structure in both cases. It allows for a fast approximation of the volumes that are part of a transparent object, either via the intersection method or viewpoint-based reconstruction. Similarly both, direct modelling and ray tracing, employ the octree to obtain their results. In all cases the octree does not only provide benefits in terms of speed of the performed search actions, but also provides additional robustness towards noise, by considering aggregated data in its leaves instead of single data points. As it turns out, reconstruction of transparent objects (from data where they cannot be measured directly) and evaluating the results of ICP or similar registration methods of point clouds have quite a lot in common – at least in the way these two problems are addressed here.

Next I will briefly summarize the results that were obtained on both of the investigated issues. The detection of transparent objects in single frames (of unorganized point cloud data) was incidentally done before actually thinking about fusing information of several viewpoints and actually thinking about reconstruction. The obtained results, while not perfect, were encouraging enough to further pursue this topic and therefore lead to the intersection-based reconstruction. The proof of concept (5.1.2) and the test done on actual transparent objects suggest that precise reconstructions of non-convex shapes are possible, as long as some constraints can be guaranteed: the poses from which the transparent object is observed need to differ sufficiently, in terms of azimuth θ, elevation φ and distance d if we think of the transparent object at the center around which the sensor positions are arranged. Furthermore, the registration between the recorded frames needs to be precise, as was shown by the impact of artificial noise on the poses obtained by hand-registration (see Figures 5.7 and 5.20). Lastly, in order to approximate more intricate details, such as a handle on a glass mug, it is important that `NaN`-regions or hole regions, dependent on the organized property of the processed point cloud, do actually resemble the "shadow" of the transparent object precisely, since violations of this constraint may seriously hamper the shape approximation, as illustrated in Figures 5.21 and 5.22.

In order to retain as much of the properties of the reconstruction via occlusion frusta intersection while adding robustness to allow the method to work on mobile robots, VPBR was created. Accordingly, the experimental setup was not specifically created for the reconstruction of transparent objects, but experiments were based on the tabletop exploration developed in [31] and adapted in terms of the objects that were place on the tabletop. Results of the approximation were compared to ground truth and reconstructions created by painted equivalent objects that were placed in the same scene. Results between reconstructed and directly measured objects were not too different, especially when compared with the convex hull of the latter (see Table 5.10) and were reported in a quantitative manner, from which other publications have refrained, so far. The quality of the results strongly suggest that tasks like object manipulation and collision avoidance, when manipulating other objects in the vicinity of a transparent object are very feasible, but have not been subject to practical evaluation, due to the limitations of the available robotic platform. Proving that will be part of future work.

Lastly it was shown that the reconstruction method is able to cope well with scenes that contain multiple transparent objects, or objects that are only partially transparent (Figure 5.32). These results are especially impressive, since the scene did not only include overlap of `NaN`-regions belonging to different transparent objects, but also many frames that only contain very sparse raw data for the reconstruction, due to varying lighting conditions, as shown in Figure 5.33. In terms of robustness the proposed method was able to successfully detect all transparent objects that were present in the scene and only added a single false positive at a location where a painted version of a transparent object was not fully covered. Although a small sample size, this results in a recall score of 1.0, precision of 0.933 and F1-score of 0.966.

Before summarizing the conducted experiments and their results for alignment evaluation, let me point out once more that this is indeed a novel idea. The need for such a mechanism is evident, for example, to enable more automated matching processes and being able to automatically identify, when something went wrong during sequential scan registration. This information can be used to abort the registration process, when needed, or to (partially) restart it with different parameter settings. Also for comparison of different alignment methods quantifiable evidence is much preferable to the often read argument "looks better". While the limited experiments provided here do not allow to claim that the problem is conclusively solved with the proposed evaluation measures, I strongly believe that it is a step in the right direction and can serve as a basis for more refined methods in the future.

The data used for the pose evaluation experiments was also not specifically created for the purpose of testing a pose evaluation measure. The first dataset was a byproduct of [31] where clusters of tabletop objects were registered against each other, employing two different ICP variants. Alignments were grouped into three general classes ($+$, $-$ and $\lightning$) and for these, mean and mean absolute deviation (MAD) were computed. These two simple features were uses to define an ad-hoc classifier (5.2) and compare its output with the manual labelling. Even this simple classification already shows promising results when assessing the alignment quality, i.e., in this case classifying two aligned frames into one of the three proposed classes. Performance of the classification might be hampered in this particular dataset, due to two main reasons: sometimes alignment is ambiguous between classes, mainly between $+$ and $-$. Secondly, the data is noisy and in some locations data points are missing, which may lead to lower confidence scores than the actual alignment would suggest, as illustrated by Figures 5.37 and 5.38. However, even with the imperfect data and the unrefined classification strategy, the evaluation measures were able to successfully distinguish between good alignments ($+$) and those registrations that were obviously wrong ($\lightning$), as reported in Table 5.16.

Performance on the "Bremen City" dataset, which substantially differs in the provided data, shows a much clearer result: differences between three alignments that were initially identified as improvable and the score of the other alignments were very noticeable, independent on the used confidence measure or if applied to the original data or a reduced version (see Tables 5.17 and 5.18). Also the amount of change between initial alignment via marker registration and the refinement provided by ICP and subsequent global optimization was clearly different between the three improvable alignments and those that were already well aligned using marker registration. Furthermore, the confidence scores obtained for the final alignments are very similar for all scenes, suggesting that the confidence measures concur with human observers, that the end result are very well aligned. Incidentally, the visualizations of the confidence measures for the final alignments, identified those parts of the point cloud that recorded moving objects (violation of the assumption that the scenes are static), positively as information contradicting correct alignment. Pedestrians can be observed as red point clusters in the scene. Figure 6.1 provides a detailed view of the plaza (since the views provided by Figure 5.40 do not show the needed details). To demonstrate that established parameter settings are not unique to a single dataset, but rather dependent on the sensor characteristics, a second dataset was analyzed, using the same setting. The Evaluation on the "Haseschacht" dataset was as convincing as for the Bremen City dataset.

The evaluation on both, tabletop exploration and Bremen city datasets, showcases that the proposed evaluation methods are applicable in scenarios of different scale, point size and noise. Similarly to a human observer the alignment assessment on the more precise Bremen city dataset is much less ambiguous, but feasibility has been demonstrated in both cases. It might be noteworthy to point out, apart from the fact that the evaluation was performed on data recorded in different context and not created to showcase the ability of some alignment measure, that the evaluated registrations were all the result of some registration method, either via markers (initial alignment Bremen city) or by some ICP variant. Even though some of the final alignments in the tabletop dataset are obviously wrong for the human observer, for ICP

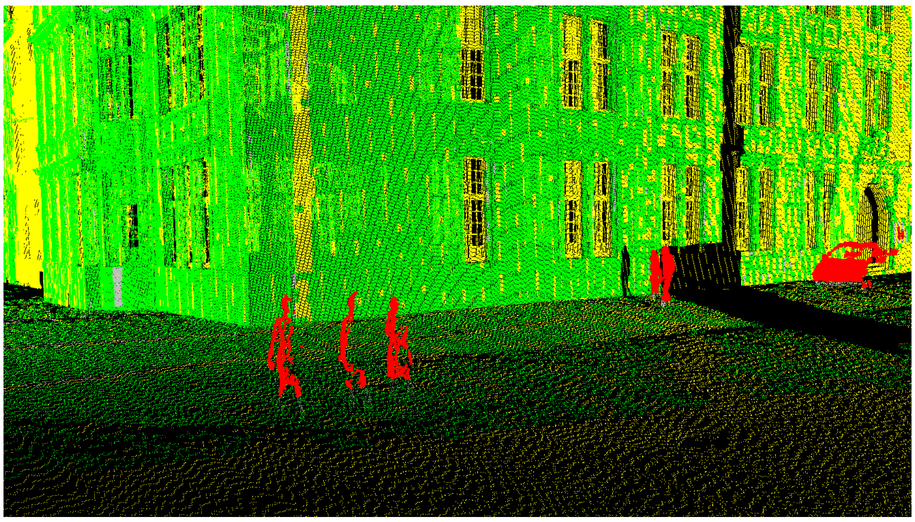


Figure 6.1: Detail pose evaluation, Bremen City dataset. Red clusters correspond to pedestrians and a car that violate the static scene assumption.

they constituted a valid local minimum, thus illustrating the need for some evaluation beyond ICP information when judging its results.

Revisiting the goals set in section 1.2 shows that they are fulfilled. In case of transparent object reconstruction the results of the viewpoint-based reconstruction (subsection 5.1.3) demonstrate, that the method is able to cope with changing lighting conditions, the inherent sensor noise of RGB-D cameras and the not always perfect alignment of frames, when using a mobile robot. If we take into account precision, recall and, most important, F1-score of this experiment the method certainly is robust and reliable in detecting correct locations of transparent objects in a given scene, without any prior knowledge about the objects. The visual appearance of the reconstructed objects support the notion that they provide sufficient information for collision avoidance and manipulation tasks also seem feasible. The latter claim is emphasized, if we take into account the similarities, in terms of quality, between the reconstruction of the transparent objects and their painted instances in the same scene. Assuming that object manipulation is possible with the extracted objects from the directly measured points supports the idea that it will also be possible for the transparent reconstructions.

With concern to the pose evaluation analysis, the desired requirements are also accomplished: the numerical output of the confidence / error measures is by design always inside a defined interval, hence independent on scene specific attributes. The results on all three examined datasets strongly support that the confidence measures can be employed to provide a relative ranking between different proposed alignments. In case of the Haseschacht dataset the confidence measures were also able to correctly identify the rather small improvements made by additional refinement step. This dataset also serves as evidence that established parameter settings for a given sensor can be applied successfully, independent on the contents of the scene, since the nature of the recorded data differed between Bremen City and Haseschacht datasets. Employing concepts like $ROO_{1,2}$ and considering in-scene occlusions ensures that only those parts of the combined point cloud are used in the assessment that can provide evidence for or against correct alignment – which is a feature not yet available in tools like CloudCompare [27]. The only goal that needs to be investigated further, in some aspects, is the easier interpretability of the confidence measures. While there is evidence that the classifiers can identify which alignments are preferable to other ones, the question which score constitutes to a "sufficiently good" alignment is not yet answered. Judging from the results on the 3 different datasets, this value is likely to be sensor dependent and influenced by the noise in its measurements. However, this needs to be investigated further in the future.

6.2 Outlook

Although the feasibility and usefulness of both, transparent object reconstruction and pose evaluation have been demonstrated in chapter 5, there is still room for improvement and future work.

With concern to the reconstruction of transparent objects, a method that retains the robustness of VPBR while also being able to approximate non-convex shapes would be highly desirable. Unfortunately, these goals are in a certain conflict, since using the 2D convex hull of a `NaN`-region instead of its (usually non-convex) outline is one of the mechanisms that allow robust and sensible estimation of the 3D convex hull for transparent objects, even though `NaN`-regions repeatedly do not properly reflect the objects shape (see Figures 5.32 and 5.33).

In its current state, the reconstruction only regards geometric information, i.e., only x, y and z of the recorded data are considered. However, if the scenes are recorded with an RGB-D camera or some sensor that offers reflectance values, additional information may be used in the reconstruction process. Considering the work that was done on detection and recognition of transparent objects in image data, alongside the more recent approach in AR [77] to estimate transparent shapes from multiple images, potential for further improvement is definitely available, if this additional information can be meaningfully fused with the existing approach. Unfortunately, many computer vision techniques are very sensitive with respect to the lighting conditions, so their applicability alongside the viewpoint-based reconstruction might be hampered on input data in the form that has been used so far.

One possible extension would be to incorporate "GrabCut" [62] into the reconstruction, in case of colored and organized input data, i.e., an image representation of the point cloud data is also available. GrabCut is originally intended to be a method to interactively separate an object of interest from the background in images, where a user provides input as to what constitutes the interesting part of given image. This is done first by defining a rectangle around the object a first segmentation process is triggered and the user can inspect its result. Refinements can be made by explicitly selecting points inside the selected rectangle to either be foreground or background and this information is then fed into the segmentation process. This technique could potentially be used to refine the appearance of `NaN`-regions.

The rectangular region of interest could be automatically selected to contain the convex hull of an extracted `NaN`-region and all `NaN`-measurements inside this rectangle are explicitly defined as foreground, hence correspond to the input usually delivered by user interaction. GrabCut could then, at least in theory, proceed to use some clues inside the image, like highlights, caused by the transparent object, to classify additional pixels as belonging to the foreground. Via the mapping between pixels and point cloud data, subsequently valid measurement points could also be classified as to be part of a transparent object, thus identifying measurement points where the support surface was measures behind a transparent object (see Figures 5.22a and 5.22b).

The success of such an approach would, like most image processing techniques, be very sensitive to lighting conditions: if insufficient visual clues of the transparent object appear in the image, GrabCut will not be able to correctly separate pixels that depict the glass from the background. This constitutes a general problem, when fusing information obtained through different sources: how to identify if one of the used approaches has failed to extract the desired information (in this case detecting the shape of the transparent object in the image data and finding the "shadow" of the object in the point cloud data) and decide to discard it in order to not corrupt the extracted information from the other source? It is also, a priori, unclear how to deal with situations, when both detection methods seem to work well (however that might be tested), but the extracted information contradicts itself. So while using the additional information available in the color information of RGB-D data can potentially offer better results, this addition also leads to some new, as off yet, unanswered problems.

Furthermore, it might be worthwhile to investigate if integrating multiple measurements from the same pose might improve performance, mainly in the appearance of the `NaN`-regions. So instead of capturing and using a single frame, taken at a given pose and using its information in the viewpoint-based reconstruction, several frames could be gathered from the same pose and from them some kind of "meta" point cloud could be generated. Approaches in this direction have been published [24, 54], but they do naturally concentrate on the improvement of regular data points, where the results of the RGB-D sensors can be compared to ground truth values. With the intended use case of transparent object reconstruction in mind, it will be interesting to investigate, if the reported improvements can be extended to the `NaN`-regions, that also appear unstable, when observed over time.

Lastly, the mentioned, but omitted, implementation of adding the occluded space behind regular objects to the occlusion frusta in each scene should be investigated. This additional information might improve reconstruction results in cluttered scenes, where in the majority of frames the transparent object is at least partially occluded by some regular object. While the addition of the new volumes to the existing occlusion frusta for a given frame is not complicated, one needs to consider the implications of this modelling. As already stated in the conclusion of subsection 3.3.2 this modelling approach will result in assuming a transparent object inside of each regular object. Removing these transparent object hypotheses, while retaining reconstructions of hybrid objects like the beer bottles in Figure 5.32 and transparent objects that are spatially close to regular objects is currently an open challenge.

For the evaluation of scan alignments, there are several areas where the existing work can be build upon. Firstly one could try to construct a more sophisticated classification method than the ad-hoc classifier used for the tabletop dataset. Also it might be worthwhile to investigate if classification performance increases, when the information of several measures, for instance c_{nc} and c_{nw}, is combined and if the relative difference between a confidence value without negative influence of contradicting clusters to its counterpart with the punishment term, i.e., $c_{\text{align}} - c_{\text{nc}}$ or $c_{\text{w}} - c_{\text{nw}}$, can be utilized to provide better results on datasets that are as noisy as the tabletop data.

In this context it will also be interesting to investigate, how much performance differs on a dataset that is similar in terms of quality to the tabletop dataset, but does contain the full point cloud information instead of only the extracted clusters, since this will reduce the influence of the object clusters on the outcome of the evaluation.

As for the application to large scale point clouds, an investigation of the parameter settings might prove useful. Even though the same parameters were successfully applied on both the Bremen city dataset and the Haseschacht building, this constitutes a rather small sample size and the question remains if, in general, these parameters tend to work well, when evaluating datasets obtained by sensors with similar measurement ranges and resolutions.

The datasets used in the experimental chapter of this thesis were chosen to demonstrate the feasibility and usefulness of the proposed methods on "real" data, i.e., data that was recorded in different contexts. Experiments on an artificial dataset, where ground truth is available, might prove useful when systematically investigating the influence of parameter settings and how good the measures can distinguish between different levels of misalignment. Such an artificial dataset would also allow an analysis about the amount of noise that the data itself may contain in order to successfully apply alignment analysis, since the tabletop dataset showed that it may become hard to (numerically) distinguish between good and insufficient, albeit not catastrophic, alignment, if the input data is very noisy.

One point that one might criticize in the foundation of both evaluation measures is the fact that in both, the direct modelling approach and the measures based on ray tracing, the investigation is stopped on the level of the octree leaves. This was a deliberate decision, but one might argue that too much information might be lost, when proceeding in this fashion: for example, an octree leaf that contains 100 measurements from sensor S_1 and a single measurement from sensor S_2 will be counted as evidence towards correct alignment. Dependent on the chosen octree leaf and the relative positioning of the sensors with respect to each other, this might actually happen, when point clouds are perfectly aligned, for instance when the leaf is close to one sensor, but so far away from the other sensor, that the point density decreases to the exemplary single measurement point. However, in other cases, when the leaf has roughly the same distance to both sensors, such a ratio should in the general case not occur, if the alignment is correct.

In a subset of leaves, where the point density of both sensors can be expected to be roughly equal and sufficiently high, local point features might be computed to gain better information about local point cloud correspondence. In order to do this, suitable leaves have to be identified first, since in general the computation of local features in octree leaves is not meaningful, as discussed at the end of subsection 4.5.1. Extending the idea further that octree leaves might be treated differently, depending on their relative position to the sensors, some parameters in the algorithms could be adapted accordingly. For instance, parameter n_{min}, the threshold that determines the minimal number of data points inside a leaf in order to include it in the evaluation, is currently constant for all leaves. If from the position of a leaf, relatively to a sensor, and the sensor parameters, an expectation about the number of measurement points could be generated, parameter n_{min} could be adapted accordingly. Thus in regions where the point density should be low, due to their distance to the sensors, no leaves would be discarded (since n_{min} would be set accordingly) while in close proximity to a sensor a leaf with the same small amount of data points could be discarded as noise. Such expectations about contained measurement points would also be able to address the previously raised issue if the ratio of measurement points from both sensors indicates good alignment or not. However, determining an expectation about the number of data points in a given leaf is by no means trivial, since this depends on what part of the actual scene is placed inside the volume of a given leaf: relative angle between measured surface and sensor and surface continuity inside a leaf influence the number of data points heavily.

Lastly, one limitation in the current form of the alignment measures is the fact that they only incorporate information concerning two point clouds that were recorded from two sensors, i.e., they can be applied well for sequences of point cloud registrations, where each scan is registered exclusively to its predecessor. However, ICP does not necessarily have to be done in a sequential fashion, but can also be performed by registering the latest scan with the combination of all predecessors, i.e., P_i is registered with $P_0 \cup \tilde{P}_1 \cup \cdots \cup \tilde{P}_{i-1}$ or a subset of these point clouds, sometimes referred to as meta-scan matching. Furthermore, as in the Bremen city dataset, some global optimization can be performed after ICP registration that also considers the information of more than two scans, when improving their alignment. This may, in some cases, actually slightly deteriorate the alignment of two subsequent point clouds, if this improves their embedding in the global context. Hence, an extension of the measures to handle the registration of one point cloud against a set of multiple point clouds is desirable, so that also alignments obtained in this fashion can be evaluated.

Implementing such an extension should be relatively simple. In case of the direct modelling approach, the region of overlap needs to be extended to not only encompass the overlap between two point clouds P_1 and P_2, but to contain the overlap of P_i with each of its predecessors. When investigating the individual octree leaves, evidence for alignment is provided in case that a leaf contains points from $\tilde{P}_i$ and measurement points of at least one other point cloud. Checking for occlusion from the combined point cloud becomes slightly more complicated, since all previous sensor positions need to be considered, additional checks concerning their field of view will become necessary. When determining if the acquired data supports in-scene occlusion of a given leaf L_i from some sensor S_j only points recorded from sensor S_j are allowed to be considered, not points taken from other sensors in the combined point cloud.

Also the extension of the ray tracing-based modelling to handle meta-scan matching seems feasible. In this case, also a combined point cloud, containing all scans that were involved in the latest registration, needs to be considered. Ray casting for the newly aligned point cloud $\tilde{P}_i$ can be performed in the same way as in the current setup. When investigating the alignment of $\tilde{P}_i$ with respect to the meta-scan things might turn out to be more complicated: rays emitted from the position of some sensor S_j ($j < i$) should be confirmed as evidence when they first encounter a leaf that contains any point of the meta-scan $P_0 \cup \tilde{P}_1 \cup \cdots \cup \tilde{P}_{i-1}$; otherwise, the quality of the previous alignments will directly influence the score for the currently investigated alignment. The way ray casting is performed from sensor poses contained in the meta-scan needs to be considered carefully.

Intuitively ray tracing should only be performed for a (previously aligned) sensor $\tilde{S}_j$ ($j < i$), if their respective region of overlap. $ROO_{i,j}$ is not empty, meaning that both sensors could, at least in theory, have observed some common surfaces in the scene. However, this check does not seem to be sufficient, since the pose of sensor $\tilde{S}_j$ may be inside the region of overlap, i.e., potentially visible from S_i, but completely occluded by measurement information gathered from different sensors that is part of the meta-scan. Handling such cases will involve a more extensive visibility analysis that the setting with only 2 scans allowed to omit. While such an analysis seems to be solvable, it will probably be no trivial problem to address. Furthermore, one needs to handle how to combine the ray tracing results for all relevant poses of the meta-scan. Simply combining all confirming and contradicting rays into sets as is done in the current setup (see equation (4.28)), entails the danger of reducing the influence that current scan $\tilde{P}_i$ has on the overall outcome, since the cardinality of cast rays in the meta-scan with its multiple sensors can become much higher.

The common parts of both approaches, namely, employing information about the viewpoint and comparing recorded data with some expectancy, can be employed for other purposes, for example CAD-matching [14, 33, 43]. One technique in this context is to create point sampling of a CAD model that one tries to locate in a given point cloud and use ICP to determine if the model is indeed present in the scene, given a sufficiently good initial guess about its position.

Instead of matching two static point clouds (scan of the environment and complete surface point sample of the CAD model) via ICP, a better approach could consider visibility constraints to determine which points of the sampled CAD model should be used in the current iteration of ICP to establish point-to-point correspondences. Depending on the initial pose estimate for the CAD model in the scene, firstly all sensors (and their associated poses) that created the combined point cloud of the environment would need to be identified. Given this information, an expectancy of the measurement points that a sensor should record, if the component, represented by the CAD model, is located at the provided pose, can be generated. These expectancies for each sensor could be fused and then used to refine the pose of the CAD model. In the next iteration, due to the change of pose, new expected measurements for each involved sensor would need to be generated and therefore the set of points that is employed in this iteration may differ from the one of the previous iteration.

The determination of the expected measurements could be performed in two different ways: firstly, if one models the used sensor precisely, ray casting from the sensor can be performed and the first intersections with the surfaces of the CAD model can be computed. As a faster, but less precise approximation, one could create a complete surface sampling of the CAD model and create an octree of the resulting point cloud. This octree can subsequently be used to determine which leaves are visible, i.e., not occluded from the provided sensor positions. Expected measurements in this case would be represented by all surface samples that are located inside those leaves that are considered visible. In order to avoid transforming the octree structure after changing the pose of its associated CAD model or creating a new octree, one could consider store the relative poses of the sensors, in relation to the (fixed) octree, instead. To perform the next visibility analysis it would then become possible to simply apply the inverse transformation to the sensors' poses and hence keep the octree structure unchanged.

Similarly, once the CAD model has been fitted into the point cloud, determining if the point cloud contains the component represented by the CAD model at the estimated pose, visibility constraints with respect to the sensor poses should be considered, so that points that could not have been observed from the sensors do not contribute in a negative or positive fashion.

Appendices

Appendix A

Images for Transparent Object Reconstruction

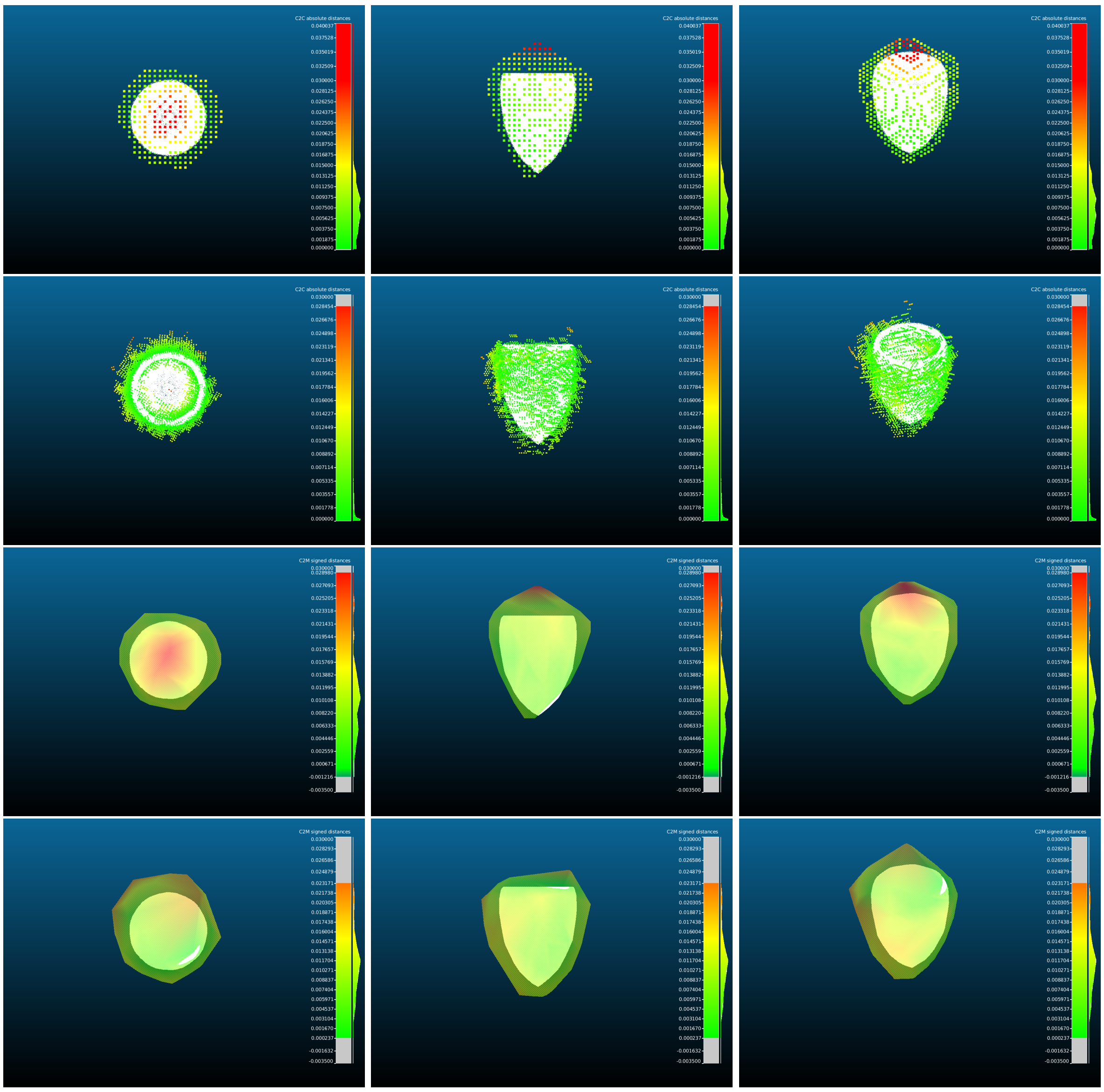


Figure A.1: Comparison viewpoint-based reconstruction `glass6`* without stem. For this comparison the stem was manually removed from the “ground truth” point cloud and the point cloud obtained by direct measurements, shown in Figure 5.27. From left to right: Top view; side view; perspective view. From top to bottom: Point cloud viewpoint-based reconstruction; point cloud direct measurements; convex hull viewpoint-based reconstruction; convex hull direct measurements.

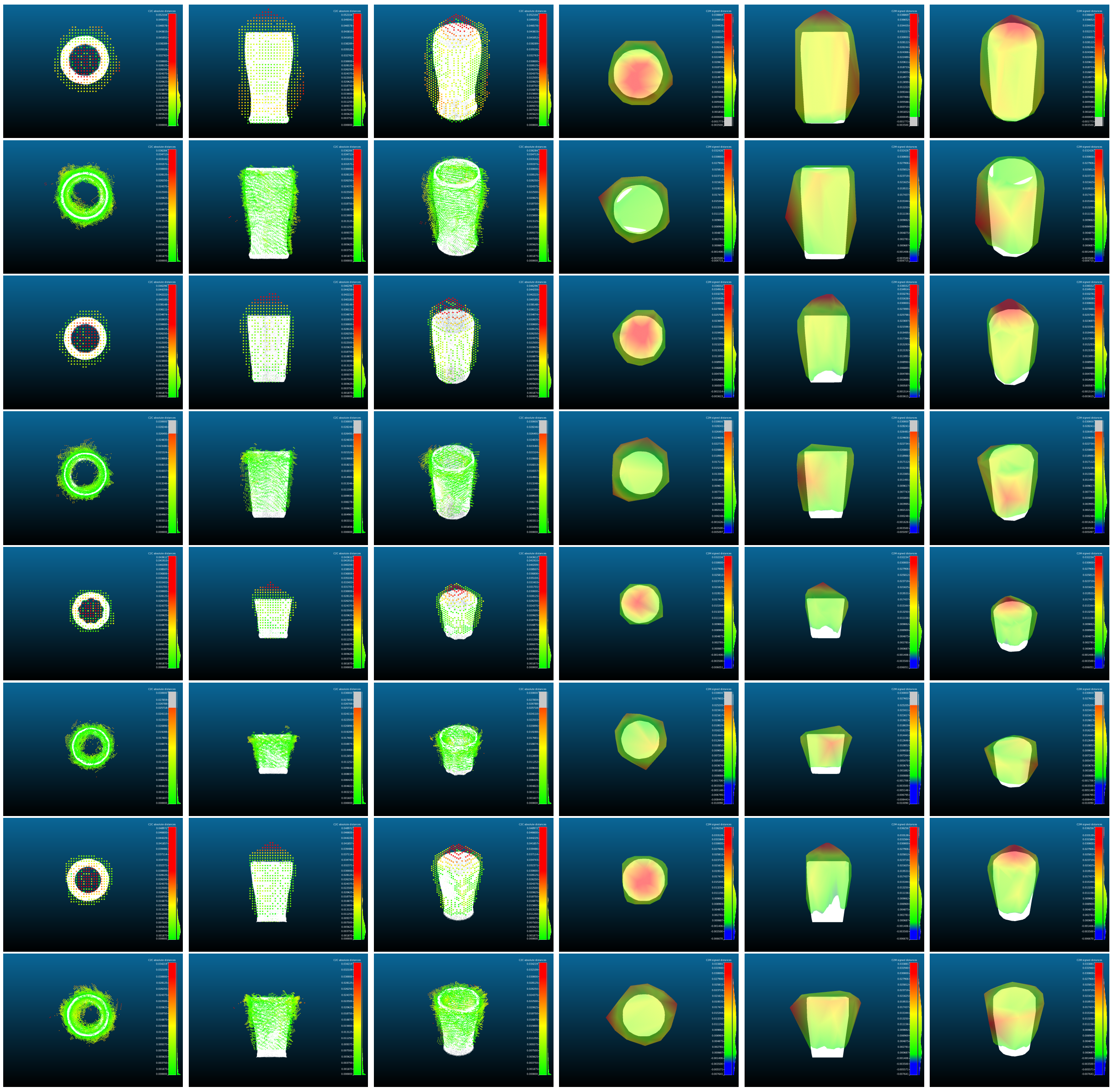


Figure A.2: Comparison viewpoint-based reconstruction `glass1`, `glass3` – `glass5`. Each row from left to right: Top view point cloud comparison; side view, point cloud; perspective view, point cloud; top view convex hull mesh; side view convex hull; perspective view convex hull. From top to bottom: Rows come in pairs, with the first row displaying the results of the viewpoint-based reconstruction and the second the reconstruction result from the painted counterpart in the scene. From top to bottom: `glass1`, `glass3`, `glass4` and `glass5`.

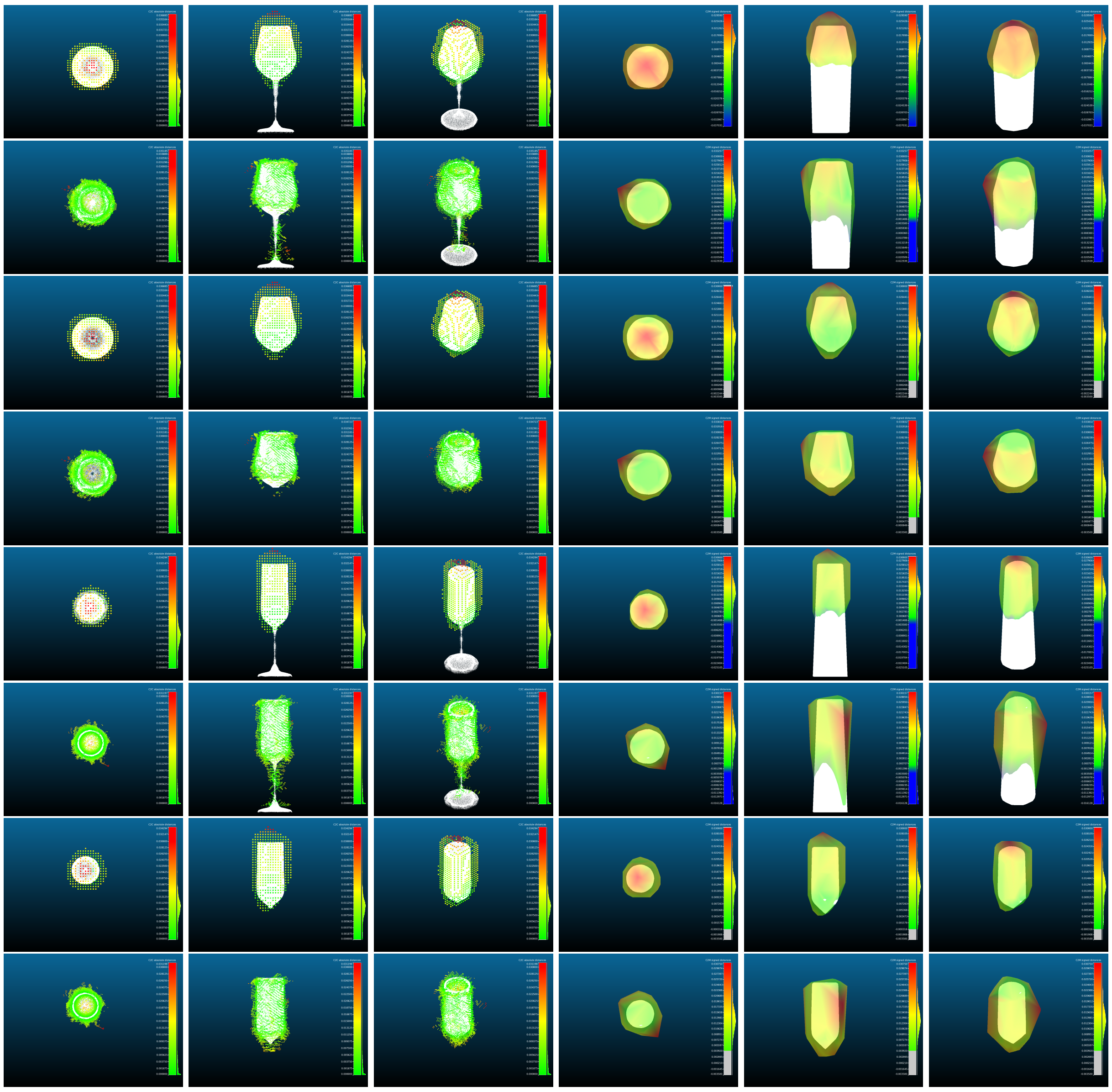


Figure A.3: Comparison of viewpoint-based reconstruction of glasses with a stem, i.e., `glass7` and `glass8`. Each row from left to right: Top view point cloud comparison; side view, point cloud; perspective view, point cloud; top view convex hull mesh; side view convex hull; perspective view convex hull. From top to bottom: Rows come in pairs, with the first row displaying the results of the viewpoint-based reconstruction and the second the reconstruction result from the painted counterpart in the scene. From top to bottom: `glass7`, `glass8`, `glass7`* and `glass8`* without stems.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Adán, A. and Adán, M. A Flexible Similarity Measure for 3D Shapes Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(11):pages 1507–1520, 2004.

[2] Adán, A. and Adán, M. Extracting Understandable 3D Object Groups with Multiple Similarity Metrics. In *Proc. CIARP 2012*, pages 179–186. 2012.

[3] Albrecht, S. and Marsland, S. Seeing the Unseen: Simple Reconstruction of Transparent Objects from Point Cloud Data. In *Proc. 2nd RSS Workshop on Robots in Clutter*. 2013.

[4] Albrecht, S., Wiemann, T., Hertzberg, J., Guesgen, H.W. and Marsland, S. From Object Recognition to Activity Interpretation and Back, Based on Point Cloud Data. *KI*, 27(2):pages 161–167, 2013.

[5] Aldoma, A., Tombari, F., di Stefano, L. and Vincze, M. A Global Hypothesis Verification Framework for 3D Object Recognition in Clutter. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(7):pages 1383–1396, 2016.

[6] Aldoma, A., Tombari, F., Rusu, R.B. and Vincze, M. OUR-CVFH - Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation. In *Proc. Pattern Recognition - Joint 34th DAGM and 36th OAGM Symposium*, pages 113–122. 2012.

[7] Behley, J., Steinhage, V. and Cremers, A.B. Efficient radius neighbor search in three-dimensional point clouds. In *Proc. ICRA 2015*, pages 3625–3630. 2015.

[8] Besl, P.J. and McKay, N.D. A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):pages 239–256, 1992.

[9] Biasotti, S., Cerri, A., Bronstein, A.M. and Bronstein, M.M. Quantifying 3D Shape Similarity Using Maps: Recent Trends, Applications and Perspectives. In *Eurographics 2014 - State of the Art Reports*, pages 135–159. 2014.

[10] Blais, F. Review of 20 years of range sensor development. *Journal of Electronic Imaging*, 13(1):pages 231–243, 2004.

[11] Blodow, N., Rusu, R.B., Marton, Z.C. and Beetz, M. Partial view modeling and validation in 3D laser scans for grasping. In *Proc. Humanoids 2009*, pages 459–464. 2009.

[12] Bo, L., Ren, X. and Fox, D. Unsupervised Feature Learning for RGB-D Based Object Recognition. In *Proc. ISER 2012*, pages 387–402. 2012.

[13] Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A. and Hertzberg, J. Globally consistent 3D mapping with scan matching. *Robotics and Autonomous Systems*, 56(2):pages 130–142, 2008.

[14] Bosché, F. Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction. *Advanced Engineering Informatics*, 24(1):pages 107–118, 2010.

[15] Bosche, F., Haas, C.T. and Akinci, B. Automated recognition of 3D CAD objects in site laser scans for project 3D status visualization and performance control. *Journal of Computing in Civil Engineering*, 23(6):pages 311–318, 2009.

[16] Bradski, G. and Kaehler, A. *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., 2nd edition, 2013. ISBN 1449314651, 9781449314651.

[17] Broadhurst, A., Drummond, T. and Cipolla, R. A Probabilistic Framework for Space Carving. In *Proc. ICCV 2001*, pages 388–393. 2001.

[18] de Koning, E. psimpl: generic n-dimensional poyline simplification, 2017 (access date).

[19] Deeken, H., Wiemann, T., Lingemann, K. and Hertzberg, J. SEMAP - a semantic environment mapping framework. In *Proc. ECMR 2015*, pages 1–6. 2015.

[20] Douglas, D.H. and Peucker, T.K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):pages 112–122, 1973.

[21] Elseberg, J. *Algorithms for Improving Mobile Laser Scanning*. Ph.D. thesis, Jacobs University Bremen, November 2013.

[22] Elseberg, J., Borrmann, D. and Nüchter, A. Full Wave Analysis in 3D laser scans for vegetation detection in urban environments. In *Proc. ICAT 2011*, pages 1–7. 2011.

[23] Elseberg, J., Borrmann, D. and Nüchter, A. One billion points in the cloud–an octree for efficient processing of 3D laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 76:pages 76–88, 2013.

[24] Essmaeel, K., Gallo, L., Damiani, E., Pietro, G.D. and Dipanda, A. Temporal Denoising of Kinect Depth Data. In *Proc. SITIS 2012*, pages 47–52. 2012.

[25] Fischler, M.A. and Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):pages 381–395, 1981.

[26] Fritz, M., Black, M.J., Bradski, G.R., Karayev, S. and Darrell, T. An Additive Latent Feature Model for Transparent Object Recognition. In *Proc. NIPS 2009*, pages 558–566. 2009.

[27] Girardeau-Montaut, D. CloudCompare, 2017 (access date).

[28] Girardeau-Montaut, D., Roux, M., Marc, R. and Thibault, G. Change detection on points cloud data acquired with a ground laser scanner. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(part 3):pages 30–35, 2005.

[29] Glassner, A.S. Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications*, 4(10):pages 15–24, 1984.

[30] González, R.C. and Woods, R.E. *Digital image processing, 3rd Edition*. Pearson Education, 2008. ISBN 9780135052679.

[31] Görner, M. *Autonomous Tabletop Object Learning*. Master's thesis, Osnabrück University, August 2015.

[32] Günther, M., Wiemann, T., Albrecht, S. and Hertzberg, J. Building semantic object maps from sparse and noisy 3D data. In *Proc. IROS 2013*, pages 2228–2233. 2013.

[33] Günther, M., Wiemann, T., Albrecht, S. and Hertzberg, J. Model-based furniture recognition for building semantic object maps. *Artificial Intelligence*, 247:pages 336–351, 2017.

[34] Hagg, A., Hegger, F. and Plöger, P.G. On Recognizing Transparent Objects in Domestic Environments Using Fusion of Multiple Sensor Modalities. *Computing Research Repository*, abs/1606.01001, 2016.

[35] Haggag, H., Hossny, M., Filippidis, D., Creighton, D.C., Nahavandi, S. and Puri, V. Measuring depth accuracy in RGBD cameras. In *Proc. ICSPCS 2013*, pages 1–7. 2013.

[36] Haines, E. Point in Polygon Strategies. In *Graphics Gems IV*, chapter Polygons and Polyhedra, pages 24–46. Academic Press Professional, Inc., San Diego, CA, USA, 1994. ISBN 0-12-336155-9.

[37] Hertzberg, J., Lingemann, K. and Nüchter, A. *Mobile Roboter: Eine Einführung aus Sicht der Informatik*. Springer-Verlag GmbH, 2012. ISBN 9783642017254.

[38] Hinterstoißer, S. *Real-time detection and pose estimation of low-textured and texture-less objects*. Ph.D. thesis, Technical University Munich, 2011.

[39] Hinterstoißer, S., Cagniart, C., Ilic, S., Sturm, P.F., Navab, N., Fua, P. and Lepetit, V. Gradient Response Maps for Real-Time Detection of Textureless Objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(5):pages 876–888, 2012.

[40] Ihrke, I., Kutulakos, K.N., Lensch, H.P.A., Magnor, M.A. and Heidrich, W. Transparent and Specular Object Reconstruction. In *Eurographics 2008 - State of the Art Reports*, pages 81–102. 2008.

[41] Kettner, L. 3D Polyhedral Surface. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.7 edition, 2015.

[42] Klank, U., Carton, D. and Beetz, M. Transparent object detection and reconstruction on a mobile platform. In *Proc. ICRA 2011*, pages 5971–5978. 2011.

[43] Klank, U., Pangercic, D., Rusu, R.B. and Beetz, M. Real-time CAD model matching for mobile manipulation and grasping. In *Proc. Humanoids 2009*, pages 290–296. 2009.

[44] Knopp, J., Prasad, M., Willems, G., Timofte, R. and Gool, L.J.V. Hough Transform and 3D SURF for Robust Three Dimensional Classification. In *Proc. ECCV 2010*, pages 589–602. 2010.

[45] Kohlhepp, P., Walther, M. and Steinhaus, P. Schritthaltende 3D-Kartierung und Lokalisierung für mobile Inspektionsroboter. In *Proc. AMS 2003*, pages 223–233. 2003.

[46] Lorensen, W.E. and Cline, H.E. Marching cubes: A high resolution 3D surface construction algorithm. In *Proc. SIGGRAPH 1987*, pages 163–169. 1987.

[47] Lowe, D.G. Object Recognition from Local Scale-Invariant Features. In *Proc. ICCV 1999*, pages 1150–1157. 1999.

[48] Lysenkov, I., Eruhimov, V. and Bradski, G.R. Recognition and Pose Estimation of Rigid Transparent Objects with a Kinect Sensor. In *Proc. RSS 2012*. 2012.

[49] Lysenkov, I. and Rabaud, V. Pose estimation of rigid transparent objects in transparent clutter. In *Proc. ICRA 2013*, pages 162–169. 2013.

[50] Magnusson, M., Lilienthal, A.J. and Duckett, T. Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10):pages 803–827, 2007.

[51] Magnusson, M., Nüchter, A., Lörken, C., Lilienthal, A.J. and Hertzberg, J. Evaluation of 3D registration reliability and speed - A comparison of ICP and NDT. In *Proc. ICRA 2009*, pages 3907–3912. 2009.

[52] Mason, J., Marthi, B. and Parr, R. Unsupervised discovery of object classes with a mobile robot. In *Proc. ICRA 2014*, pages 3074–3081. 2014.

[53] May, S., Droeschel, D., Holz, D., Fuchs, S., Malis, E., Nüchter, A. and Hertzberg, J. Three-dimensional mapping with time-of-flight cameras. *Journal of Field Robotics*, 26(11-12):pages 934–965, 2009.

[54] Milani, S. and Calvagno, G. Correction and Interpolation of Depth Maps from Structured Light Infrared Sensors. *Signal Processing: Image Communication*, 41(C):pages 28–39, February 2016.

[55] Minguez, J., Montesano, L. and Lamiraux, F. Metric-based iterative closest point scan matching for sensor displacement estimation. *IEEE Transactions on Robotics*, 22(5):pages 1047–1054, 2006.

[56] Nüchter, A., Gutev, S., Borrmann, D. and Elseberg, J. Skyline-based registration of 3D laser scans. *Geo-Spatial Information Science*, 14(2):pages 85–90, June 2011.

[57] Park, J. and Kak, A.C. 3D Modeling of Optically Challenging Objects. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):pages 246–262, 2008.

[58] Pomerleau, F., Colas, F., Siegwart, R. and Magnenat, S. Comparing ICP variants on real-world data sets - Open-source library and experimental protocol. *Autonomous Robots*, 34(3):pages 133–148, 2013.

[59] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T.B., Leibs, J., Wheeler, R. and Ng, A.Y. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*. 2009.

[60] Renz, J. and Nebel, B. Qualitative Spatial Reasoning Using Constraint Calculi. In *Handbook of Spatial Logics*, pages 161–215. Springer, 2007. ISBN 978-1-4020-5586-7.

[61] Rodríguez-Losada, D. and Minguez, J. Improved Data Association for ICP-based Scan Matching in Noisy and Dynamic Environments. In *Proc. ICRA 2007*, pages 3161–3166. 2007.

[62] Rother, C., Kolmogorov, V. and Blake, A. "GrabCut": Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Transactions on Graphics*, 23(3):pages 309–314, 2004.

[63] Ruhnke, M., Steder, B., Grisetti, G. and Burgard, W. Unsupervised learning of 3D object models from partial views. In *Proc. ICRA 2009*, pages 801–806. 2009.

[64] Rusu, R.B. and Cousins, S. 3D is here: Point Cloud Library (PCL). In *Proc. ICRA 2011*, pages 1–4. 2011.

[65] Rusu, R.B., Blodow, N. and Beetz, M. Fast Point Feature Histograms (FPFH) for 3D registration. In *Proc. ICRA 2009*, pages 3212–3217. 2009.

[66] Rusu, R.B., Blodow, N., Marton, Z.C. and Beetz, M. Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments. In *Proc. IROS 2009*, pages 1–6. 2009.

[67] Ryde, J. and Brünig, M. Non-cubic occupied voxel lists for robot maps. In *Proc. IROS 2009*, pages 4771–4776. 2009.

[68] Saito, M., Sato, Y., Ikeuchi, K. and Kashiwagi, H. Measurement of surface orientations of transparent objects using polarization in highlight. *Systems and Computers in Japan*, 32(5):pages 64–71, 2001.

[69] Salti, S., Tombari, F. and di Stefano, L. SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:pages 251–264, 2014.

[70] Shoemake, K. Animating rotation with quaternion curves. In *Proc. SIGGRAPH 1985*, pages 245–254. ACM Press, New York, NY, USA, 1985.

[71] Shreiner, D., Sellers, G., Kessenich, J.M. and Licea-Kane, B.M. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*. Addison-Wesley Professional, 8th edition, 2013. ISBN 9780321773036.

[72] Shum, H., Hebert, M. and Ikeuchi, K. On 3D Shape Similarity. Technical Report CMU-CS-95-212, Carnegie Mellon University, 1995.

[73] Solda, E., Worst, R. and Hertzberg, J. Poor-Man's Gyro-Based Localization. In *Proc. IAV 2004)*, pages 412–417. 2004.

[74] Strunk, W. and White, E.B. *The Elements of style*. Allyn and Bacon : Longman, Boston (USA), 1999. ISBN 9780205313426.

[75] Surmann, H., Lingemann, K., Nüchter, A. and Hertzberg, J. A 3D laser range finder for autonomous mobile robots. In *Proc. ISR 2001*, pages 153–158. 2001.

[76] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.7 edition, 2015.

[77] Torres-Gomez, A. and Mayol-Cuevas, W.W. Recognition and reconstruction of transparent objects for augmented reality. In *Proc. ISMAR 2014*, pages 129–134. 2014.

[78] Wiemann, T., Albrecht, S. and Hertzberg, J. Konzeptentwicklung zur automationsgestützten 3D-Digitalisierung kulturhistorisch relevanter Bauten. In *Photogrammetrie, Laserscanning, Optische 3D-Messtechnik : Beiträge der Oldenburger 3D-Tage 2017*, pages 248–253. 2 2017.

[79] Wulf, O., Arras, K.O., Christensen, H.I. and Wagner, B. 2D Mapping of Cluttered Indoor Environments by Means of 3D Perception. In *Proc. ICRA 2004*, pages 4204–4209. 2004.

[80] Zadeh, L.A. Fuzzy Sets. *Information and Control*, 8(3):pages 338–353, 1965.

[81] Robotic 3D Scan Repository. http://kos.informatik.uni-osnabrueck.de/3Dscans, 2017 (access date).

[82] 3DTK – The 3D Toolkit. http://slam6d.sourceforge.net/, 2017 (access date).

[83] Point Cloud Library (PCL) API Documentation. http://docs.pointclouds.org/trunk/, 2017 (access date).