

Master Thesis



Czech  
Technical  
University  
in Prague

**F3**

Faculty of Electrical Engineering  
Department of Computer Graphics and Interaction

## Indoor localization utilizing dense point cloud

Anna Zderadičková

Supervisor: Ing. Michal Polic  
May 2022



## I. Personal and study details

Student's name: **Zderadi ková Anna** Personal ID number: **408058**  
Faculty / Institute: **Faculty of Electrical Engineering**  
Department / Institute: **Department of Computer Graphics and Interaction**  
Study program: **Open Informatics**  
Specialisation: **Computer Graphics**

## II. Master's thesis details

Master's thesis title in English:

**Indoor localization utilizing dense pointcloud**

Master's thesis title in Czech:

**Využití hustého mra na bod pro lokalizaci ve vnitřních prostorách**

Guidelines:

- 0) Read related literature and get familiar with the topic of localization from RGB-D images
- 1) Capture a dataset of RGB-D images from laboratory B-315 by HoloLens
- 2) Align the camera poses from HoloLens to 3D model of B-315 laboratory created by Matterport
- 3) Run the state-of-the-art localization pipeline HLOC (CVPR2019) using SuperGlue matching (CVPR2020) on the captured dataset
- 4) Fine-tune the camera pose by state-of-the-art dense pointcloud alignment methods R-ICP (PAMI2021), PREDATOR (CVPR2021), FCGF (CVPR2019), TEASER (IEEE 2020)
- 5) Create the GUI (using Meshroom interface) for fine-tuning the parameters of the localization pipeline
- 6) Evaluate the accuracy of localization with proposed improvements

Bibliography / sources:

- 1) Sarlin, Paul-Edouard, et al. 'From coarse to fine: Robust hierarchical localization at large scale.' Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
- 2) Sarlin, Paul-Edouard, et al. 'Superglue: Learning feature matching with graph neural networks.' Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.
- 3) Zhang, Juyong, Yuxin Yao, and Bailin Deng. 'Fast and Robust Iterative Closest Point.' IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).
- 4) Huang, Shengyu, et al. 'PREDATOR: Registration of 3D Point Clouds with Low Overlap.' Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.
- 5) Choy, Christopher, Jaesik Park, and Vladlen Koltun. 'Fully convolutional geometric features.' Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.
- 6) Yang, Heng, Jingnan Shi, and Luca Carlone. 'Teaser: Fast and certifiable point cloud registration.' IEEE Transactions on Robotics 37.2 (2020): 314-333.

Name and workplace of master's thesis supervisor:

**Ing. Michal Polic Applied Algebra and Geometry CIIRC**

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **01.03.2022** Deadline for master's thesis submission: **20.05.2022**

Assignment valid until: **19.02.2024**

Ing. Michal Polic  
Supervisor's signature

Head of department's signature

prof. Mgr. Petr Páta, Ph.D.  
Dean's signature

### III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

\_\_\_\_\_  
Date of assignment receipt

\_\_\_\_\_  
Student's signature

## Acknowledgements

I would like to thank my supervisor, Ing. Michal Polic, for supervising my work and introducing me into this interesting topic.

## Declaration

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, May 20, 2022

Prohlašuji, že jsem předloženou práci vypracovala samostatně, a že jsem uvedla veškerou použitou literaturu.

V Praze, 20. května 2022

## Abstract

Accurate and robust localization is crucial in various fields such as augmented reality or robot navigation. State-of-the-art methods based on image localization give pose estimation which are rather accurate. However accurate pose estimation might be challenging in plain (textureless) or visually repeating environments. This thesis proposes an approach to combine state-of-the-art localization method with dense point cloud alignment to obtain robust and accurate camera poses. It also presents a process of obtaining a ground truth dataset for RGB-D localization using Microsoft HoloLens, Matterport Pro2 and Vicon. The proposed point cloud alignment methods are then evaluated on the created ground truth data.

**Keywords:** Microsoft HoloLens, point cloud alignment, AR, augmented reality, localization, indoor localization

**Supervisor:** Ing. Michal Polic  
Český institut informatiky, robotiky a kybernetiky, ČVUT v Praze,  
Jugoslávských partyzánů 1580/3,  
160 00 Dejvice

## Abstrakt

Přesná a robustní lokalizace je důležitá v mnoha aplikacích jako například rozšířená realita nebo navigace robotů. Nejmodernější přístupy založené na lokalizaci pomocí fotografií poskytují odhad pozice, který je přesný. Avšak najít přesný odhad pozice je náročné v prostředích bez textur či v prostředích, kde se vizuální prvky opakují. Tato práce navrhuje postup využití metody lokalizace z fotek společně se zarovnáváním hustých mračen bodů k zrobustnění a zpřesnění pozice kamery. Také je představen postup pro získání přesné validační sady dat pro lokalizaci RGB-D snímků za pomoci technologií Microsoft HoloLens, Matterport Pro2 a Vicon. Na těchto datech jsou porovnány čtyři metody pro zarovnávání mračen bodů.

**Klíčová slova:** Microsoft HoloLens, zarovnání mračen bodů, AR, rozšířená realita, lokalizace, lokalizace vnitřních prostor

**Překlad názvu:** Využití hustého mračen bodů pro lokalizaci ve vnitřních prostorech

# Contents

<b>1 Introduction</b>	<b>1</b>
<b>2 Goal</b>	<b>3</b>
<b>3 Related work</b>	<b>5</b>
3.1 Localization . . . . .	5
3.2 Microsoft HoloLens Localization .	6
3.3 Dense point cloud alignment . . . .	9
<b>4 Dataset acquisition</b>	<b>11</b>
4.1 Matterport map . . . . .	11
4.2 HoloLens data acquisition . . . . .	13
4.3 HoloLens data alignment to Vicon	14
4.3.1 Approximate R-ICP alignment	15
4.3.2 Outlier filtering . . . . .	16
4.3.3 Objective function . . . . .	16
4.3.4 Three-step optimization . . . .	18
4.3.5 Mean error correction . . . . .	19
4.3.6 Point cloud alignment . . . . .	20
<b>5 GUI</b>	<b>25</b>
5.1 Meshroom . . . . .	25
5.2 HLoc localization pipeline . . . . .	26
5.3 Point cloud alignment pipeline .	26
<b>6 Localization improvement</b>	<b>29</b>
6.1 The Hierarchical Localization method . . . . .	29
6.2 Point cloud alignment methods .	30
6.2.1 Testing alignment methods ..	31
6.2.2 Simulated data . . . . .	31
6.2.3 Alignment methods evaluation	35
6.2.4 R-ICP . . . . .	35
6.2.5 FCGF + TEASER++ . . . . .	39
6.2.6 FCGF + RANSAC . . . . .	42
6.2.7 Overlap PREDATOR . . . . .	43
6.2.8 Localization fine-tuning . . . .	45
<b>7 Conclusion</b>	<b>51</b>
<b>Bibliography</b>	<b>53</b>
<b>A Dataset acquisition additional materials</b>	<b>59</b>
<b>B Full evaluation results</b>	<b>61</b>
B.1 R-ICP . . . . .	61
B.2 FCGF + RANSAC . . . . .	70
B.3 FCGF + TEASER++ . . . . .	79
B.4 Overlap PREDATOR . . . . .	88

## Figures

<p>3.1 HoloLens sensors, their placing, and view described by [31]. . . . . 7</p> <p>3.2 Closed trajectory from [31] showing the error caused by accumulated drift on HoloLens. . . . . 8</p> <p>3.3 Example of alignment of point clouds with low overlap by PREDATOR shown in [10]. . . . . 10</p> <p>4.1 Matterport markers and colored point cloud converted to Vicon coordinate system. . . . . 12</p> <p>4.2 Matterport to Vicon coordinate system alignment errors in mm when gradually leaving out 1, 2, 3, 4 points out of 8 correspondences used for the alignment. . . . . 12</p> <p>4.3 Observed errors of Matterport to Vicon coordinate system alignment. 12</p> <p>4.4 Vicon markers attached to HoloLens. . . . . 14</p> <p>4.5 Scheme of the alignment pipeline. First, we get a roughly aligned HoloLens poses to Vicon coordinate system by R-ICP, then we filter out outliers. Then we minimize the objective function for fine alignment. Lastly, final fine-tuning options are explored (mean error correction and R-ICP alignment on point clouds). The color frames are representing the color of the residuals in following figures 4.10, 4.11. . . . . 15</p> <p>4.6 Timestamps of the depth camera and the PV camera. . . . . 16</p> <p>4.7 Graphical interpretation of eqn. (4.5), (4.6), i.e. the description of the transformation from HoloLens coordinate system to Vicon coordinate system. In practise, because of a irregular HoloLens framerate, the timestamp is substituted by <math>\tau</math>. Which means that we use time steps obtained from subtracting first timestamp of HoloLens photo-video camera from all HoloLens timestamps. . . . . 18</p>	<p>4.8 Graph of the objective function values dependent on <math>\beta</math>. . . . . 19</p> <p>4.9 Example of HoloLens PV camera and depth (long throw depth) sensor aligned with Vicon. Aligned HoloLens cameras are purple, unaligned HoloLens are yellow, Vicon is green, and blue are the Vicon markers shifted by <math>t_i</math> to HoloLens. Red lines represents the errors between aligned HoloLens and Vicon. . . . . 20</p> <p>4.10 Distances between all aligned HoloLens camera centres and Vicon marker origins. The histogram can be seen in appendix B, stair graph was synoptical. . . . . 21</p> <p>4.11 Distances between aligned HoloLens point clouds and Matterport point cloud. (Note that the number of points in point clouds differ as point clouds are being cut accordingly to the rough alignment. Thus the histograms are normalized by the number of points aligned. Histogram can be seen in appendix A. . . . . 22</p> <p>4.12 Examples of point clouds incorrectly aligned by R-ICP. . . . . 23</p> <p>5.1 Example of a Meshroom pipeline 25</p> <p>5.2 Example of a Meshroom node definition. . . . . 27</p> <p>5.3 HLoc localization pipeline. . . . . 28</p> <p>5.4 Dense point cloud aligner node.. 28</p> <p>6.1 The Hloc localization accuracy. More than 95% of the camera poses were localized within 20 cm and 5 degree distance threshold. . . . . 30</p> <p>6.2 Overview of the Hloc localization for each camera. . . . . 31</p>
--	---




6.3 Matterport (upper) and depthmap (lower) maps cutout example. Purple point cloud is the map, green is the cutout of the map and blue is ground truth depthmap from HoloLens assuming the camera pose. . . . .	34	6.13 Alignment recall for FCGF + Teaser++ over all translation noised point clouds on the depthmap map. . . . .	42
6.4 The alignment errors for R-ICP over all HoloLens point clouds constructed with approximate camera poses on the Matterport map. . . . .	35	6.14 Alignment errors for FCGF + RANSAC over all translation noised point clouds on the Matterport map. . . . .	43
6.5 Alignment recall for R-ICP over all HoloLens point clouds constructed with approximate camera poses on the Matterport map. . . . .	36	6.15 Alignment recall for FCGF + RANSAC over all translation noised point clouds on the Matterport map. . . . .	44
6.6 Alignment errors for R-ICP alignment over all translation noised point clouds on a map composed from HoloLens depthmaps. . . . .	37	6.16 Alignment errors for FCGF + RANSAC over all translation noised point clouds on the depthmap map. . . . .	44
6.7 Alignment recall for R-ICP alignment over all translation noised point clouds on a map composed from HoloLens depthmaps. . . . .	37	6.17 Alignment recall for FCGF + RANSAC over all translation noised point clouds on the depthmap map. . . . .	45
6.8 Examples of high error alignment by R-ICP, the initial state of point clouds is on the left and on the right is the resulting alignment. In the first row, the alignment fails because there are some objects on the map cutout that are missing in the depthmap. The second and third row shows, that point cloud with little to no objects in it is very likely to be aligned incorrectly. . . . .	38	6.18 Results of HoloLens and Matterport point clouds alignment by PREDATOR several times with same pair of point clouds and same parameters, each time the alignment gave a different result. . . . .	47
6.9 Examples of only mutually nearest feature descriptors from FCGF, features considered for alignment are red and green. . . . .	39	6.19 Alignment errors for PREDATOR over all translation noised point clouds on the Matterport map. . . . .	48
6.10 Alignment errors for FCGF + Teaser++ over all translation noised point clouds on the Matterport map. . . . .	40	6.20 Alignment recall for PREDATOR over all translation noised point clouds on the Matterport map. . . . .	48
6.11 Alignment recall for FCGF + Teaser++ over all translation noised point clouds on the Matterport map. . . . .	40	6.21 Alignment errors for PREDATOR alignment over all translation noised point clouds on a map composed from HoloLens depthmaps. . . . .	49
6.12 Alignment errors for FCGF + Teaser++ over all translation noised point clouds on the depthmap map. . . . .	41	6.22 Alignment recall for PREDATOR alignment over all translation noised point clouds on a map composed from HoloLens depthmaps. . . . .	49
		6.23 Localization recall before (left) and after (right) fine-tuning by point cloud alignment. . . . .	50
		A.1 Distances between all aligned HoloLens camera centres and Vicon marker centres. . . . .	59

A.2 Distances between aligned HoloLens point clouds and Matterport point cloud. (Note that the number of points in point clouds differ as point clouds are being cut accordingly to alignment, thus leading to normalizing the histograms (the value of the bin equals to occurrence/sum(occurrences))) ...	60
B.1 Translation error for R-ICP alignment over all translation and rotation noised point clouds on the Matterport map.....	62
B.2 Rotation error for R-ICP alignment over all translation and rotation noised point clouds on the Matterport map.....	63
B.3 Translation recall for R-ICP alignment over all translation and rotation noised point clouds on the Matterport map.....	64
B.4 Rotation recall for R-ICP alignment over all translation and rotation noised point clouds on the Matterport map.....	65
B.5 Translation error for R-ICP alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. ....	66
B.6 Rotation error for R-ICP alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. ....	67
B.7 Translation recall for R-ICP alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. ....	68
B.8 Rotation recall for R-ICP alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. ....	69
B.9 Translation error for FCGF + RANSAC alignment over all translation and rotation noised point clouds on the Matterport map. ...	71
B.10 Rotation error for FCGF + RANSAC alignment over all translation and rotation noised point clouds on the Matterport map. ...	72
B.11 Translation recall for FCGF + RANSAC alignment over all translation and rotation noised point clouds on the Matterport map. ...	73
B.12 Rotation recall for FCGF + RANSAC alignment over all translation and rotation noised point clouds on the Matterport map. ...	74
B.13 Translation error for FCGF + RANSAC alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. ....	75
B.14 Rotation error for FCGF + RANSAC alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. ....	76
B.15 Translation recall for FCGF + RANSAC alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. ....	77
B.16 Rotation recall for FCGF + RANSAC alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. ....	78
B.17 Translation error for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on the Matterport map. ...	80
B.18 Rotation error for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on the Matterport map. ...	81

B.19 Translation recall for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on the Matterport map. . . . .	82	B.29 Translation error for PREDATOR alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. . . . .	93
B.20 Rotation recall for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on the Matterport map. . . . .	83	B.30 Rotation error for PREDATOR alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. . . . .	94
B.21 Translation error for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. . . . .	84	B.31 Translation recall for PREDATOR alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. . . . .	95
B.22 Rotation error for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. . . . .	85	B.32 Rotation recall for PREDATOR alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. . . . .	96
B.23 Translation recall for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. . . . .	86		
B.24 Rotation recall for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps. . . . .	87		
B.25 Translation error for PREDATOR alignment over all translation and rotation noised point clouds on the Matterport map. . . . .	89		
B.26 Rotation error for PREDATOR alignment over all translation and rotation noised point clouds on the Matterport map. . . . .	90		
B.27 Translation recall for PREDATOR alignment over all translation and rotation noised point clouds on the Matterport map. . . . .	91		
B.28 Rotation recall for PREDATOR alignment over all translation and rotation noised point clouds on the Matterport map. . . . .	92		

## Tables

6.1 Alignment .....	32
---------------------	----



# Chapter 1

## Introduction

The problem of precise localization is becoming increasingly important as more and more applications rely on accurate pose estimation (e.g., augmented reality (AR) or navigation). Most smartphones prefer fast *signal-based localization* (global navigation satellite system, GNSS). It is real-time, however, the precision of the calculated pose is unstable [1, 2, 3, 4]. Other AR-focused devices, like Microsoft HoloLens, localize based on information from the environment (images, 3D points, etc.). This might be challenging when users are using two or more different devices for the localization and map update, while these devices' localization approaches and accuracy differ.

Image based localization can estimate a position from which the image was taken based on a previously built image database with known camera poses. In this work, we want to evaluate the HLoc method [5]. However, localizing from images alone can be challenging when there is only a wall or other texture poor object. This is where point cloud alignment methods can be used to improve the localization. When the image is featureless, the depthmap of the same view can be much more relevant to establish the location. Point cloud alignment methods that are explored in this work are: R-ICP [6], FCGF [7] + RANSAC [8], FCGF + TEASER++ [9], and Overlap PREDATOR [10].

We focus on indoor localization of Microsoft HoloLens. However, all proposed methods can be applied to any device that can capture RGB-D frames. First, a pose estimate is obtained by HLoc [5], and then it is refined by using a depthmap to environment map alignment. We compare four methods for point cloud alignment and show whether and when they improve the localization accuracy.

The structure of this thesis is as follows. The first section focuses on the analysis of current localization solutions and state-of-the-art methods. Next, we discuss the approach to obtain datasets that are later used for evaluating several alignment methods. This dataset is composed of aligned recordings obtained by Matterport Pro2 scanner [11], Motion Capture system Vicon [12] and Microsoft HoloLens. Then, the process of creating ground truth data from individual recordings is explained. The following chapter describes the implementation of the graphical user interface for the utilized methods and build evaluation pipelines. Lastly, the evaluation of several point cloud





## Chapter 2

### Goal

This thesis investigates if and how estimated poses from images can be improved by dense point cloud alignment. It evaluates state-of-the-art methods and explores several of their attributes. We summarize the goal of this work as follows.

At first, **review the literature about localization from RGB-D images**. Get familiar with localization from images only and recent dense point cloud alignment methods.

To evaluate the accuracy of the localization and proposed fine-tuning, we require an RGB-D indoor dataset with known camera poses captured by Microsoft HoloLens. This is motivated by the EU H2020 grant ARTwin and desired localization of AR devices (HoloLens, iPad, etc.) at the construction site and factory environment. There exists a number of high quality public datasets for pose estimation from images, such as KITTI [13] or Aachen Day-Night [14]. However, these datasets consist of RGB-D data and camera poses but focus mostly on outdoor environments. As far as we know, there is little public RGB-D indoor datasets that could be used to evaluate the accuracy of localization. For example, the datasets First-Person Hand Action dataset [15], EgoDexter [16], or HandNet [17] are used for hand and gesture recognition. InLoc [18] dataset is indoor, provides dense point cloud as an environment map, but the localization queries consist of RGB images only, missing the depthmap. SUN3D [19] dataset fits the criteria, but the environment maps are created using SfM leading to holes in inaccessible areas. In this work, we **create the dataset consisting of RGB-D images, ground truth camera poses, and a dense point cloud map of the environment**. This dataset can be used for other projects, experiments, or localization methods, and it is publicly available at <https://bit.ly/3ID9puj>.

The technology used in this thesis to obtain the ground truth camera poses is Microsoft HoloLens and Motion Capture system Vicon [12]. The 3D map is built by the Matterport Pro2 scanner [11].

Our next goal is to **run the localization pipeline HLoc [5]** on the created ground truth dataset. The HLoc method is selected because it is robust and performs well even on challenging benchmarks [20]. The localization

runs in tens of milliseconds and thus can be used for real-time localization. We want to verify if the estimated poses by HLoc can be improved by the utilization of additional depth sensor available on many recent AR devices.

We want to check if **fine-tuning of camera poses by depthmap alignment** to a 3D map of the environment helps to improve the pose estimation accuracy if RGB images are poor on textures and distinct points. Challenging are also environments with repeated 3D structures (for example, hallways or labs). In general, all the environments where it is difficult to localize just based on an RGB image alone, e.g., construction site, corridors, etc.

Localization algorithms have different parameters, and the accuracy of the localization highly depends on their setup. Running these methods manually would mean setting these parameters over and over again for each experiment which is both time consuming and prone to human mistakes. Therefore we want to simplify the evaluation of methods by **running the pipeline using a graphical user interface (GUI)**, where all parameters are clearly arranged and easy to set. We were looking for an open-source solution that allows the quick, easy, and well-manageable realization of pipelines. We chose Meshroom because it is an open-source 3D reconstruction pipeline. This software represents an easy and comfortable way to create and run pipelines where methods are implemented in the form of nodes connected together by edges if data is transferred between them. This means that all nodes implemented in this work can be used in any other pipeline and in any future project. The codes are publicly available at <https://bit.ly/3ID9puj>.



## Chapter 3

### Related work

This chapter summarizes state-of-the-art methods for localization, point cloud alignment, and localization properties of the Microsoft HoloLens. The first section discusses several widely used methods for the localization of devices. Then the description of HoloLens follows. The last section focuses on several well known and recently published point cloud alignment methods.

### 3.1 Localization

The thesis mainly focuses on the localization of AR devices. Most of the modern are equipped with sensors that allow them to be utilized as AR devices as well. Therefore, we discuss all relevant ways of localization (GNSS, IMU, markers, etc.).

The most common way of localization on mobile devices is based on global navigation satellite systems (GNSS) like GPS [1] or Galileo [2]. These systems have accuracy for smartphones in the order of meters (within 4.9 m for GPS and within 4 m for Galileo [21, 22]). There is also a high-end, high-accuracy solution for both of the mentioned systems, and their accuracy is in the order of tens of centimeters (within 1.82 m for GPS and within 20 cm for Galileo [21, 22]). The localization is fast, real-time, and it works well in open areas. The GNSS rarely performs well in an indoor environment as it needs a signal from satellites.

A similar method for the localization of mobile devices is the triangulation based on the signal strength of wireless networks [23], Wi-Fi [4], Bluetooth [3], etc. These methods require network coverage, and the device gets lost when the user gets out of reach. These methods' accuracy varies depending on the distance from the source of the signal. Wi-Fi based indoor localization can be precise up to 5 cm with the approach proposed by [24]. For Bluetooth, the accuracy was observed to be 0.79-2.28 m [25].

Some mobile devices can track their pose using an inertial measurement unit (IMU). IMUs tend to be sensitive to drifting errors and often need to be corrected by some external approach [26]. Paper [27] explores tracking of a human using 3 IMUs, the accumulated error over a 3.6 m trajectory is 7.6 cm.

The next approach is marker-based localization. There are two ways to

use markers for localization. The markers (ArUco, for example) are either placed in the environment, and the device equipped with a camera is localized with respect to them, or the markers are attached to the device and then observed by an external Motion Capture system. The first approach requires the placement of markers, knowledge about their location, and access to the device's camera. On the other hand, in the second approach, the device must be seen by the tracking system and can only move in a determined space. Authors of [28] explored the accuracy of AprilTags markers for localization and observed that the localization error is in the order of centimeters, mostly varying from 5 cm up to 40 cm in translation and below 0.002 degrees in rotation. The motion capture accuracy depends mainly on the hardware of the system. For example accuracy of Vicon has been evaluated to be under 2 mm [29].

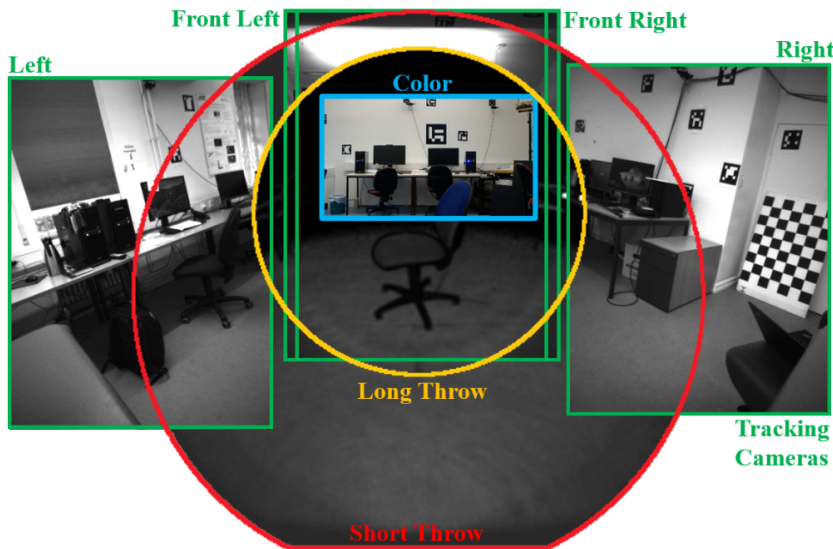
The last approach mentioned is based on visual feature detection. Its idea is to localize the device from an image, i.e., its distinctive local points and their correspondences in a 3D map or database of features. It can be used as well for outdoor as for the indoor localization. The InLoc method [18] shows that we can localize under 1 m and 10 degrees error in 69.9% of cases. Another method (used in this work), the Hierarchical Localization (HLoc) [5], focuses on large scale localization with significant changes in the environment. It has been observed that this method localizes within 25 cm and 2 degrees in around 70% of cases during the day and around 40% at night. This evaluation was performed on Aachen Day-Night, RobotCar Seasons, and CMU Seasons datasets, consisting of approximately 80,000 query images.

## ■ 3.2 Microsoft HoloLens Localization

HoloLens is a mobile headset augmented reality (AR) device. This work uses data obtained from the first generation HoloLens. In default mode, the headset only provides access to the main RGB photo-video camera. Upon activating the research mode, users gain access to sensor streams and sensor poses [30]:

- 4 visible light tracking cameras, gray-scale (used for localization and building of the map of the environment)
- 2 depth streams - short-throw depth stream is used for tracking hands and hand gestures (viewing distance is 0-0.8 m [31]), while the long-throw depth stream is used for obtaining depth data from the environment (0.8-3.5 m [31]) (utilized in spatial mapping)
- 2 infra-red-reflectivity streams - similar to depth sensors, two versions of the sensor are: short-throw and long-throw

The overview of sensor field-of-view and their overlap is described in [31]. The visualization is in fig. 3.1.

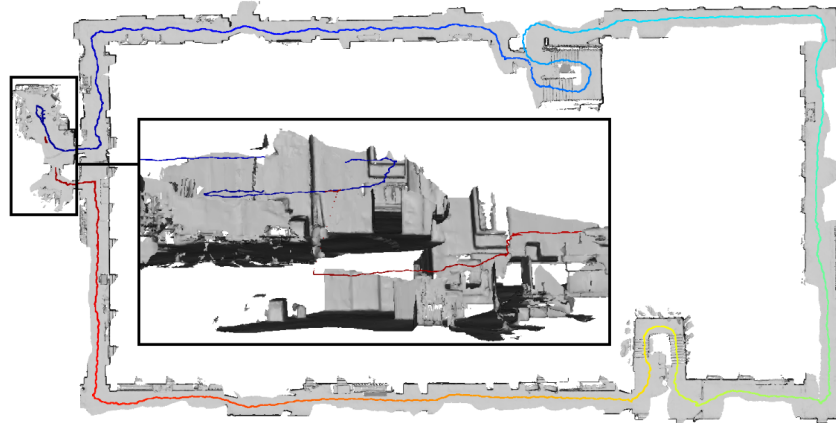


**Figure 3.1:** HoloLens sensors, their placing, and view described by [31].

However, this mode can be used for research purposes only, and applications using it cannot be deployed into the Windows Store as it can be misused.

HoloLens tracking performs well in ideal conditions, however, some environmental conditions need to be taken into consideration [32]. Since the device uses visible light cameras to track location, light plays an important role. When it is too light or too dark, cameras cannot see anything, and the device may get lost. A similar situation may occur when there is a change in lighting (coming from bright light to dim light or going in the same location during different seasons), it may confuse the device. HoloLens' localization performance is also influenced by the appearance of the environment. The device uses features (unique landmarks) to locate, so when the user is in a featureless environment or some same features are repeated (same images on a wall in a hall or the same furniture in offices), the device tends to get lost or thinks it is in a different location. Even when there are a lot of features in the environment, HoloLens cannot localize if they constantly move or change since there are no stable points to locate against. Objects that are too close to the device (15 cm and less) or even blocking one or more of the tracking cameras, as well as highly reflective surfaces, may lead to problems in localization. Lastly, the device links the map data with a Wi-Fi fingerprint, so without Wi-Fi, the headset may take longer to localize initially, or it may not recognize the place if the fingerprint has changed.

In [31], the authors focused on evaluating HoloLens localization indoors in several different aspects. First, they examined the noise and accuracy of the depth data. During this experiment, the device was lying on a still stand facing a white wall. Gradually, they tested whether the device's temperature, distance from the wall, and inclination impacted the depth data accuracy. The first experiment showed that the depth value changed by 6 mm while warming up when running. The next test revealed that for the distance



**Figure 3.2:** Closed trajectory from [31] showing the error caused by accumulated drift on HoloLens.

of the wall lower than 2.5 m, the sensor noise stays below 5 mm, from 2.5 m upwards, the noise increases up to 1 cm. The inclination changes cause noise below 5 mm for angles below 20 degrees and up to 1 cm noise at 80 degrees. Next, the authors looked into HoloLens tracking ability. They used a motion capture system (OptiTrack Prime 17W) to track the device and get ground truth data. They evaluated the accuracy of the tracking on one room scale and on a large (one floor) scale. HoloLens tracking can have up to approximately  $1.6 \pm 0.02$  cm and  $2.2 \pm 0.3$  degrees drift per second. This drift can lead to large errors, for example, during the evaluation of tracking for a trajectory long 287 m, where the accumulated error caused by drifting equals 2.39 m (fig. 3.2).

Paper [33] explores other ways of localization using HoloLens, ARUco marker based localization. Their method is used to overlay the indoor environment with a virtual room-scale model, thus augmenting the room. The accuracy is then measured by placing other “control” markers in the room and adding squares in the virtual environment where they should be located. During the testing, they took 15 images from 7 points of view (105 images in total) in a room of the dimensions of 8 m×5 m×3 m. The error is then defined as a spatial deviation between the corners of the marker and the virtual square. The mean error is 2.3 cm, while the distance values range from 2.0 cm to 2.5 cm. However, the poses of markers have to be known in order to localize, and it is not suitable for a large-scale localization.

The last related work to be mentioned is [34]. This paper explores indoor localization for AR devices, using point clouds and building information modeling (BIM). BIM is a digital representation of a building (3D model or a point cloud usually), and if it is accurate enough, it could be used for localization. The authors reduce the point cloud data dimensionality to 2D and use a floor-plan matching algorithm. The estimated error over point clouds from this method is about 10-20 cm in the horizontal (xy) plane and 1-5 cm in the ground estimation for HoloLens. The overall localization error

was observed to be smaller than 0.5 m and 5 degrees.

### 3.3 Dense point cloud alignment

There are many point cloud alignment methods available and explored. This section will look at some of the widely used methods as well as recently published methods used later in this work.

The first method to mention is **Procrustes analysis** [35]. Procrustes analysis compares two sets of points and tries to align one to the other. The alignment is done by calculating centroids, i.e., mean and variance of both point clouds and align them using translation, rotation, and scale.

Another approach is a robust model estimator utilized for aligning point clouds. It is called **Random Sample Consensus (RANSAC)** [8]. It is based on uniformly and randomly subsampling point feature descriptor correspondences. Then, for each sample, the transformation parameters are computed. After that, resulting alignments are evaluated on full data for all computed transformations. The transformation that fits the best results on specified criteria (like inlier count) is chosen as the best estimate. RANSAC is widely used, and new alignment methods based on it are often introduced. Some recent methods to mention are MAGSAC [36] and NAPSAC [37] with its recent modification - Progressive NAPSAC [38].

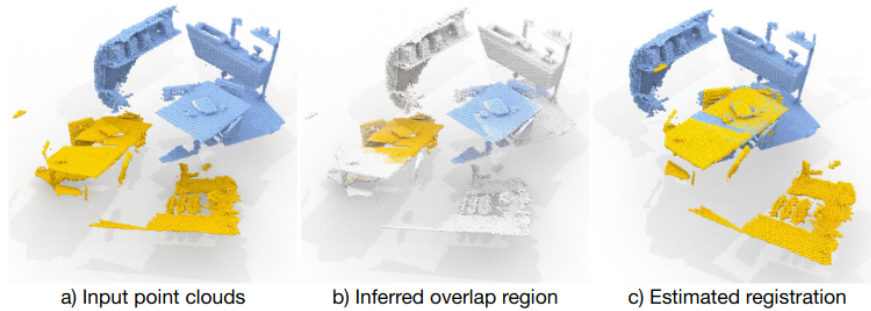
**Iterative Closest Point (ICP)** [39] is a method for fitting two point clouds based on an iterative least-squares approach. For each point, the algorithm finds its nearest neighbor from the second point cloud. For this set of pairs, it then determines their transformation, transforms the point cloud, and repeats this process until it converges to a desired result. This method converges to a locally optimal alignment, which may not be ideal when considering possible noise or low point cloud overlap. Robust ICP (R-ICP) [6] is an approach focusing on this problem. It is a robust implementation of the Iterative Closest Point algorithm. Authors propose a new robust error metric that makes ICP perform well on challenging datasets which are noised or overlap only partially.

**TEASER++** [9] stands for "truncated least squares estimation and semi definite relaxation". TEASER++ is used to align point clouds. It requires point correspondences on input. The method decouples scale, rotation, and translation using invariant measurements and uses truncated least squares to solve them. The scale is estimated using adaptive voting, rotation is estimated by semi definite relaxation, and translation is again estimated by adaptive voting. The authors report observed average error of 7 cm and 4 degrees on pose estimation on the 3DMatch dataset.

**Fully Convolutional Geometric Features (FCGF)** [7] is a method used to calculate point cloud features. FCGF uses a 3D fully-convolutional network to estimate feature vectors describing the local neighborhood of each

point. The network needs either an existing pre-trained model or training on specific data. The features obtained by FCGF can be used to find similar points, i.e. correspondences, in multiple point clouds. The RANSAC-based model estimator can then estimate the Euclidean transformation from these correspondences. Thus this method is later used in this work paired with two other feature matching methods (RANSAC, as it has been used in the paper, and TEASER++). It has been observed that approximately in 95% of cases, point clouds align within 10 cm and 5 degrees error on the 3DMatch dataset.

**Overlap PREDATOR** [10] is a model for pairwise point cloud registration specialized in point clouds with low overlap, i.e.,  $\leq 30\%$  overlap. Interest points of the point clouds are sampled, and their local neighborhood is described by feature descriptors. PREDATOR utilizes FCGF descriptors that are matched to establish correspondences. The rotation and translation are then estimated by running RANSAC estimator. As PREDATOR focuses on low overlap point clouds, it focuses on sampling interest points mostly in overlapping areas. It has been shown that PREDATOR aligns point clouds from the 3DMatch dataset with an average error of 8 cm in translation and 5 degrees in rotation.



**Figure 3.3:** Example of alignment of point clouds with low overlap by PREDATOR shown in [10].

## Chapter 4

### Dataset acquisition

As far as we know, there is no suitable dataset for localization of Microsoft HoloLens. This chapter describes the software and interface for recording RGB-D data. It summarizes and evaluates the format of a recording and aligning of its camera poses to camera poses recorded by Vicon motion capture system [12]. Since the Vicon marker poses are obtained for both, HoloLens data and Matterport model, both coordinate systems are converted to Vicon coordinate system. In the following section, we describe the coordinate systems, used transformations and its accuracy.

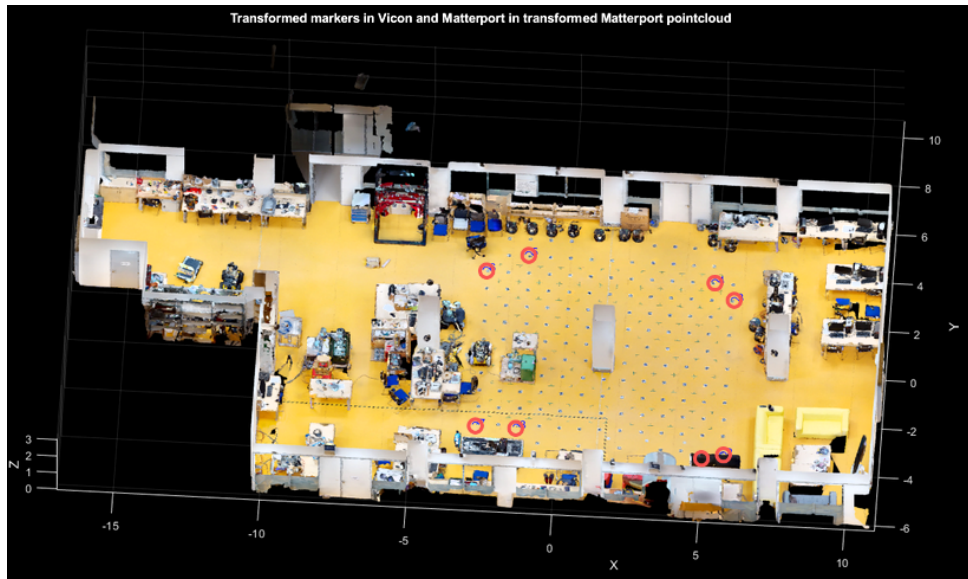
#### 4.1 Matterport map

The Matterport scan can be obtained by several devices. We employed the Matterport Pro2 scanner [11]. The scan is represented as a colored point cloud.

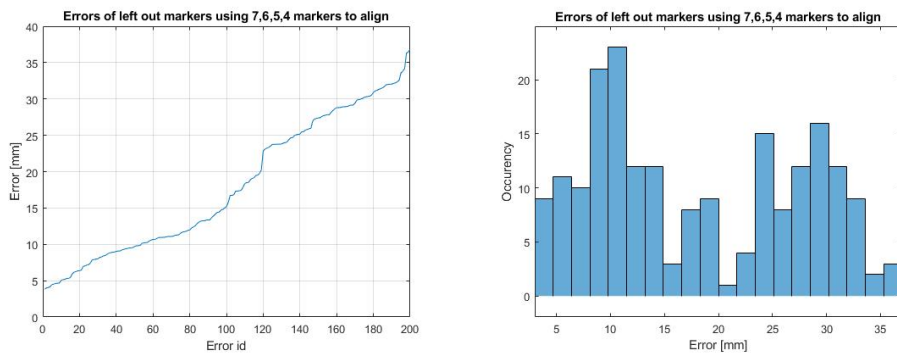
The scanned room (laboratory B-315) is equipped by ArUco markers attached on the floor (see fig. 4.3). We aligned the Matterport coordinate system by manually measuring 8 of these markers in both coordinate systems (Vicon and Matterport) fig. 4.1. The corresponding measurements were aligned by the procrustes method.

To validate its accuracy, we used the cross-validation gradually leaving out 1 to 4 points. The omitted points were used to measure the error (i.e. the Euclidean distance to corresponding points) and the rest for the alignment itself. Results of this experiment can be seen in fig. 4.2.

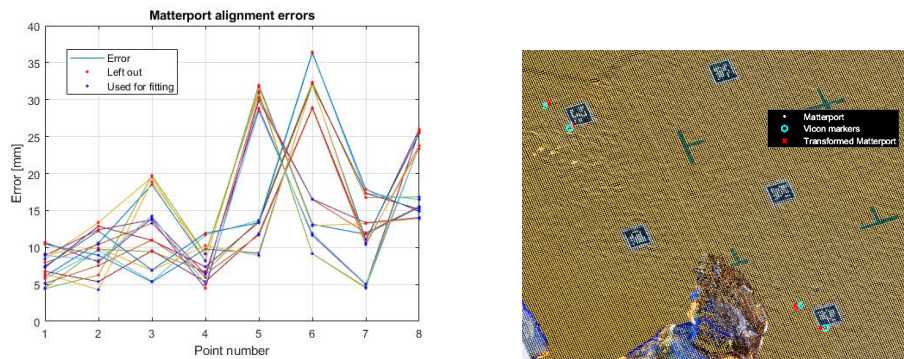
Note that we can observe in the histogram fig. 4.2 that there are two main clusters of errors. To further inspect this observation, the errors for 4 points alignment over all point combinations were plotted. The points used for alignment are highlighted, see fig. 4.3. These results imply a large relational error between 5th and 6th point.



**Figure 4.1:** Matterport markers and colored point cloud converted to Vicon coordinate system.



**Figure 4.2:** Matterport to Vicon coordinate system alignment errors in mm when gradually leaving out 1, 2, 3, 4 points out of 8 correspondences used for the alignment.



**(a)** : Alignment errors using 4 points to align.

**(b)** : Error of 5th and 6th points.

**Figure 4.3:** Observed errors of Matterport to Vicon coordinate system alignment.



In conclusion the Matterport scan is transformed to Vicon coordinate system with the accuracy between approximately 5-35 mm.

## 4.2 HoloLens data acquisition

We employed the first generation of HoloLens in this work. This AR device is equipped with four gray-scale “environment understanding” cameras (also called “*visible light*” cameras), one time-of-flight camera (capturing *short throw* and *long throw* depth and reflectivity frames) and a RGB camera called the “*photo-video*” camera. Streams from these sensors can be obtained by the HoloLensForCV example project [40]. We modified HoloLensForCV so that the photo-video camera captures images in the  $1344 \times 756$  resolution, 15 fps, and 67 degrees field of view. The visible light cameras are synchronised with the photo-video camera. The resolution of visible light cameras is  $448 \times 450$ . The depth sensor uses the same resolution as visible light cameras and its frequency is 1-3 fps.

Each recorded stream of data includes a file (.csv file) containing camera poses, timestamps, frame names and matrices for conversion from camera coordinate system to world coordinate system.

The photo-video camera frames are obtained in a .ppm file format, i.e. as a software bitmap in raw BGRX pixel mode. We evaluated the conversion of them to PGM pixel format on the HoloLens device. However, it caused a significant bottleneck and thus the conversion was left to the post-recording step. This is resolved offline using python Pillow library for image manipulation. The images are converted to .jpg files.

The visible light camera frame is saved as a .pgm file. It has one gray-scale channel of data, so the saved raw data is displayed correctly so the image does not required any conversion. However the frames acquired are rotated by 90 degrees. The rotation is fixed again by using Pillow python library.

The depthmap is obtained in .pgm format with values  $g(u, v) \in \mathbb{R}$  in range from 0 to 65535, where each pixel is realized by a tuple  $(u, v)$ . The Microsoft API provides an undistortion hashing table  $h(u, v) \in \mathbb{R}^3$  that realizes the transformation from distorted pixels to undistorted ones  $[u', v', 1]^T$ . Values from a depthmap can be then transformed to points in camera coordinate system by:

$$\mathbf{X}(u, v) = \frac{h(u, v)g(u, v)}{\|h(u, v)\|_2} \quad (4.1)$$

To transform HoloLens coordinate system to Vicon coordinate system, Vicon markers were attached to HoloLens headset (fig. 4.4). The recording session was started on HoloLens and Vicon at approximately the same time. Using this approach we acquire HoloLens camera poses with timestamps at frequency 15 fps and Vicon poses with timestamps at 100 fps. Vicon poses are then considered ground truth which is used later to obtain HoloLens ground truth poses. To obtain these poses, we discuss the time synchronization, Vicon to HoloLens translation and drift optimization in the next sections.



**Figure 4.4:** Vicon markers attached to HoloLens.

To use HoloLens camera poses for building a database for HLoc. The database is build upon an SfM 3D reconstruction, so HoloLens poses are converted to COLMAP camera model definition. Rotation  $\mathbf{Q}_{k,i}$  of  $i$ -th camera is transformed as follows. Where  $i$  realizes the id of the camera and  $k$  represents the  $k$ -th image taken.

The photo-video camera rotation equals:

$$\mathbf{Q}_{k,1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{Q}'_{k,1} \quad (4.2)$$

where  $i = 1$  stands for the photo-video camera. All the other  $j$ -th camera rotation matrices are:

$$\mathbf{Q}_{k,j} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{Q}'_{k,j} \quad \text{where } j = \{2, 3, 4, 5, 6\} \quad (4.3)$$

and  $j$  stands of visible light and depth cameras. The matrix  $\mathbf{Q}'_{k,i}$  is calculated as:

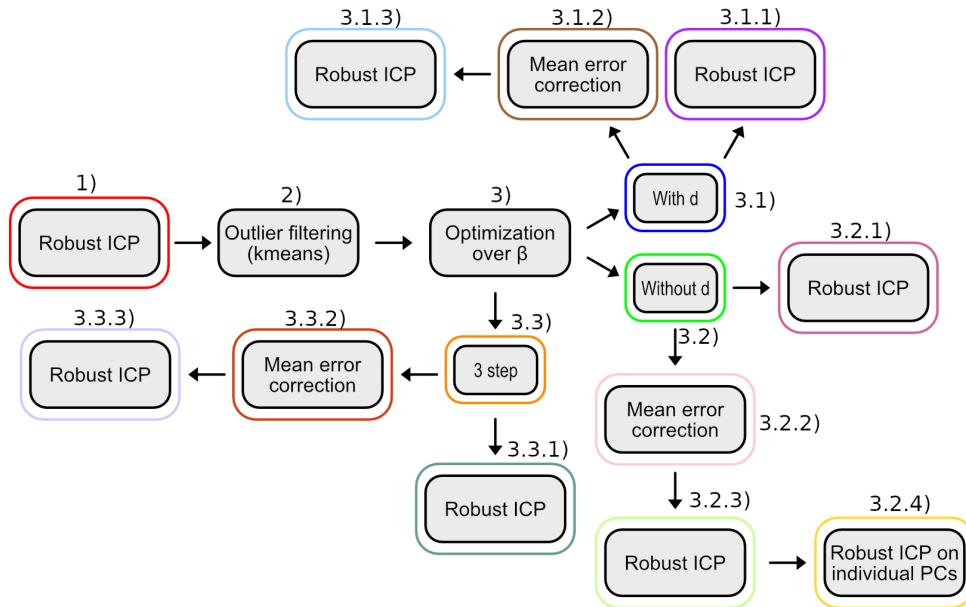
$$\mathbf{Q}'_{k,i} = \mathbf{A}_{D2O}^{(k,i)} \mathbf{A}_{C2D}^{(k,i)} \quad (4.4)$$

Both matrices,  $\mathbf{A}_{D2O}^{(k,i)}$  and  $\mathbf{A}_{C2D}^{(k,i)}$ , are further notated without the  $(k, i)$  index if clear from context. They are stored in the .csv recording file provided by the HoloLens API.

### 4.3 HoloLens data alignment to Vicon

This section describes the transformation from HoloLens coordinate system to Vicon coordinate system. HoloLens cameras and Vicon tracking are

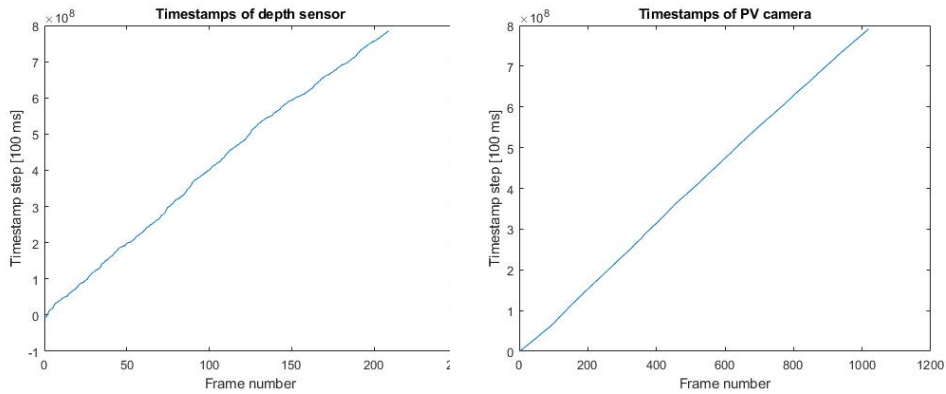
not synchronized in time which leads to a requirement of a post-processing synchronization task. It is also known, that HoloLens is prone to localization drifts (2.39 m per 287 m trajectory). Moreover, Vicon tracking system sometimes confuses the marker ids and switch them, causing a tracking error. The following text describes several methods, that solve individual challenges and we use them for HoloLens alignment to Vicon coordinate system. The whole process is visualized in fig. 4.5. Robust ICP [6] (later called R-ICP) is used to roughly align HoloLens camera poses with Vicon poses. Then the outliers in Vicon tracks are filtered out. Finally, an objective function is optimized in order to find accurately transformation parameters aligning HoloLens to Vicon coordinate system. Three objective functions of the optimization are explored. We also evaluated fine-tuning of camera poses by using R-ICP [6] on depthmaps to Matterport point cloud alignment.



**Figure 4.5:** Scheme of the alignment pipeline. First, we get a roughly aligned HoloLens poses to Vicon coordinate system by R-ICP, then we filter out outliers. Then we minimize the objective function for fine alignment. Lastly, final fine-tuning options are explored (mean error correction and R-ICP alignment on point clouds). The color frames are representing the color of the residuals in following figures 4.10, 4.11.

### 4.3.1 Approximate R-ICP alignment

The HoloLens camera poses and Vicon marker poses are temporally and spatially unaligned “at the beginning”. We want to roughly align HoloLens and Vicon trajectory both, temporally and spatially (step 1) in fig. 4.5). Camera poses from HoloLens photo-video camera, visible-light cameras and the depth camera are taken into consideration. We are looking for a global optimum of the alignment and R-ICP does not guarantee to converge to it,



**Figure 4.6:** Timestamps of the depth camera and the PV camera.

as it may converge to a local optimum. For this reason, all camera poses are randomly rotated 100 times before running R-ICP to robustify the alignment estimate. The Euclidean transform with the lowest nearest neighbour distance between transformed camera centers and Vicon markers is chosen. This is the approximate alignment, because the Vicon outliers and translation between centers of Vicon markers and HoloLens cameras are not considered here.

### 4.3.2 Outlier filtering

As it has been mentioned before, Vicon is prone to switching markers, when they get too close together. To consider this possibility happening in our tracks, we are filtering out possible errors before fine-tuned alignment (step 2) in fig. 4.5). We have found the outliers by using k-means on distances between Vicon markers and HoloLens camera centers. We select the cluster with smaller Euclidean distance. The distance threshold was found and used to filter out the presumed outliers. Then we ran the approximate R-ICP alignment again on poses without outliers.

### 4.3.3 Objective function

We want to achieve even more accurate camera poses by aligning HoloLens depthmaps to Matterport map in Vicon coordinate system. Since the camera poses and Vicon trajectory are roughly aligned, we can focus on local surrounding of depthmaps in Vicon coordinate system and optimize the distance between point cloud from Matterport and from HoloLens (step 3) in fig. 4.5). This can be achieved by R-ICP. However applying this approach to photo-video camera poses may impose further pose deviations because the depth sensor is not synchronized with other cameras, see fig. 4.6.

The framerate of HoloLens sensors is irregular. The most significant variance of the framerate has the depth sensor. This leads to the challenge that constant framerate cannot be assumed in any alignment calculations. Instead we use time steps obtained from subtracting first timestamp of HoloLens photo-video camera from all HoloLens timestamps. The objective

function transforming HoloLens poses to Vicon coordinate system can be described as follows.

We assume:

- $\tau \in \mathbb{R}$  - is the difference between the current frame and the first frame timestamp. We use directly the relative time of HoloLens capture w.r.t. the first timestamp instead of the multiple of 1/framerate.
- $\beta \in \mathbb{R}$  - time shift that realizes the synchronization offset of timestamps w.r.t. of Vicon to HoloLens timestamps
- $\gamma \in \mathbb{R}$  - we denote  $\gamma = \tau + \beta$  for brevity
- $\mathbf{v}_{\gamma,i} \in \mathbb{R}^3$  - realizes Vicon markers shifted into HoloLens camera positions in Vicon coordinate system
- $\mathbf{h}_{k,i} \in \mathbb{R}^3$  - stands for HoloLens camera positions transformed to Vicon coordinate system
- $\mathbf{c}_\gamma \in \mathbb{R}^3$  - origin of Vicon marker in Vicon coordinate system
- $\mathbf{R}_\gamma \in \mathbb{R}^{3 \times 3}$  - the rotation of Vicon marker in Vicon coordinate system captured in time  $\gamma$
- $\mathbf{t}_i \in \mathbb{R}^3$  - translation between Vicon marker and position of  $i$ -th HoloLens camera in Vicon coordinate system
- $\rho \in \mathbb{R}$  - is a scalar that realizes the scale w.r.t. the Vicon coordinate system to HoloLens coordinate system
- $\mathbf{S} \in \mathbb{R}^{3 \times 3}$  - stands for the rotation of the HoloLens coordinate system to the Vicon coordinate system
- $\mathbf{e}_{k,i} \in \mathbb{R}^3$  - realize the  $k$ -th center of  $i$ -th camera in HoloLens coordinate system
- $\mathbf{d} \in \mathbb{R}^3$  - is the origin of HoloLens coordinate system in Vicon coordinate system.

Known variables are:  $\mathbf{c}_\gamma, \mathbf{R}_\gamma, \mathbf{e}_{k,i}$  and these parameters are unknown and to be optimized:  $\mathbf{t}_i, \rho, \mathbf{S}, \mathbf{d}$ .

Vicon to ground truth camera pose transformation is:

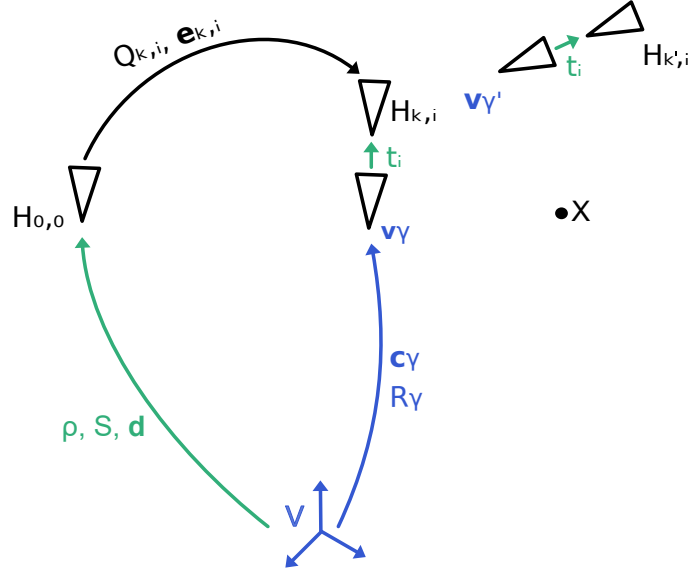
$$\mathbf{v}_{\gamma,i} = \mathbf{c}_\gamma + \mathbf{R}_\gamma^\top \mathbf{t}_i \quad (4.5)$$

HoloLens to Vicon camera position transformation equals:

$$\mathbf{h}_{k,i} = \frac{1}{\rho} \mathbf{S}_{k,i}^\top \mathbf{e}_{k,i} - \mathbf{d}, \forall k = 1, \dots, K \quad (4.6)$$

where  $K$  is the number of HoloLens camera positions. In ideal conditions, the difference between these position equals zero. Thus, we are optimizing the function:

$$f_{\gamma,i} = \mathbf{v}_{\gamma,i} - \mathbf{h}_{k,i}. \quad (4.7)$$



**Figure 4.7:** Graphical interpretation of eqn. (4.5), (4.6), i.e. the description of the transformation from HoloLens coordinate system to Vicon coordinate system. In practise, because of an irregular HoloLens framerate, the timestamp is substituted by  $\tau$ . Which means that we use time steps obtained from subtracting first timestamp of HoloLens photo-video camera from all HoloLens timestamps.

In the fig. 4.8 both function responses, i.e. the initial and optimized are shown. We are minimizing the following function:

$$\operatorname{argmin}_{t_i, \rho, S, d, \beta} \left( \sum (\|f_{\gamma, i}\|^2) + 10(\|t_i\| + \|d\|) \right) \quad (4.8)$$

We expect that the values of  $t$  and  $d$  are small, because they represent a translation in the order of centimeters (from Vicon marker origin to HoloLens camera center).

The function value is optimized over  $\beta$  from 0 to 100 to determine the best time shift. The best  $\beta$  is determined to be 57. In Figure 4.9 are presented examples of aligned cameras with HoloLens after tuning, using the best  $\beta$  and the parameters optimized for this  $\beta$ . To determine wherever it is beneficial to use  $d$  in the objective function, the optimization is evaluated for the objective function with and without  $d$  (steps 3.1) and 3.2) in fig. 4.5). Thus the redefined objective function:

$$\sum (\|f'_{\gamma, i}\|^2) + 10\|t_i\|. \quad (4.9)$$

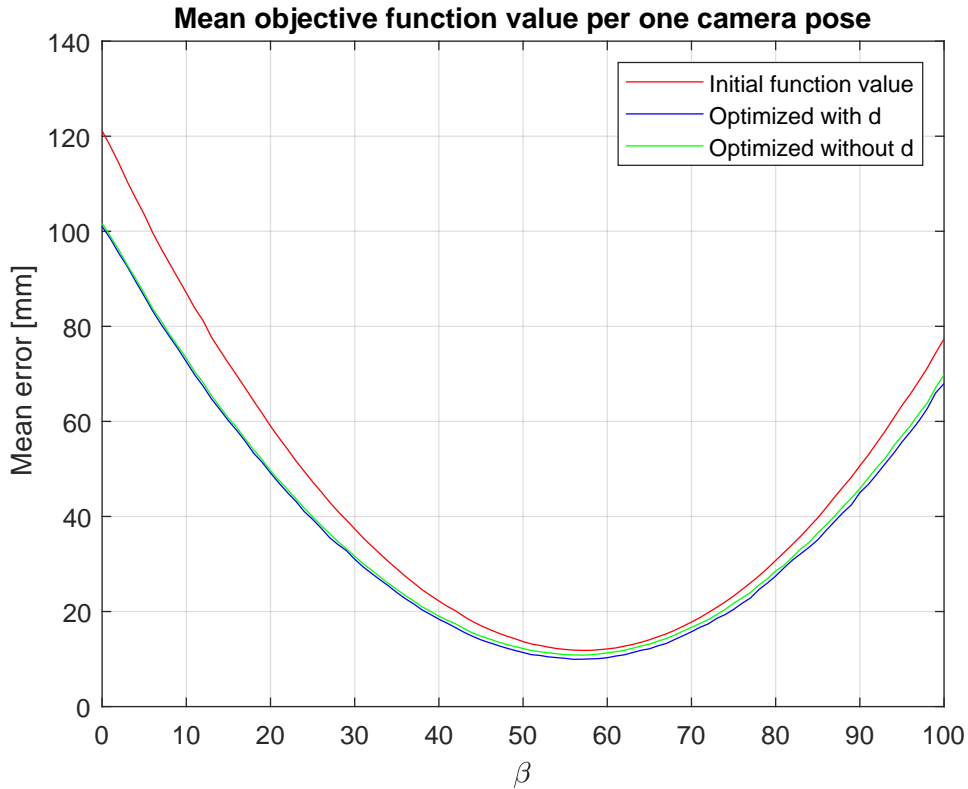
where

$$f'_{\gamma, i} = c_{\gamma} + R_{\gamma}^{\top} t_i - \frac{1}{\rho} S_{k, i}^{\top} e_{k, i} \quad (4.10)$$

#### 4.3.4 Three-step optimization

We know that HoloLens have a drift error. To remove HoloLens shift and rotation drift a “three-step” optimization is examined (step 3.3) in fig. 4.5).

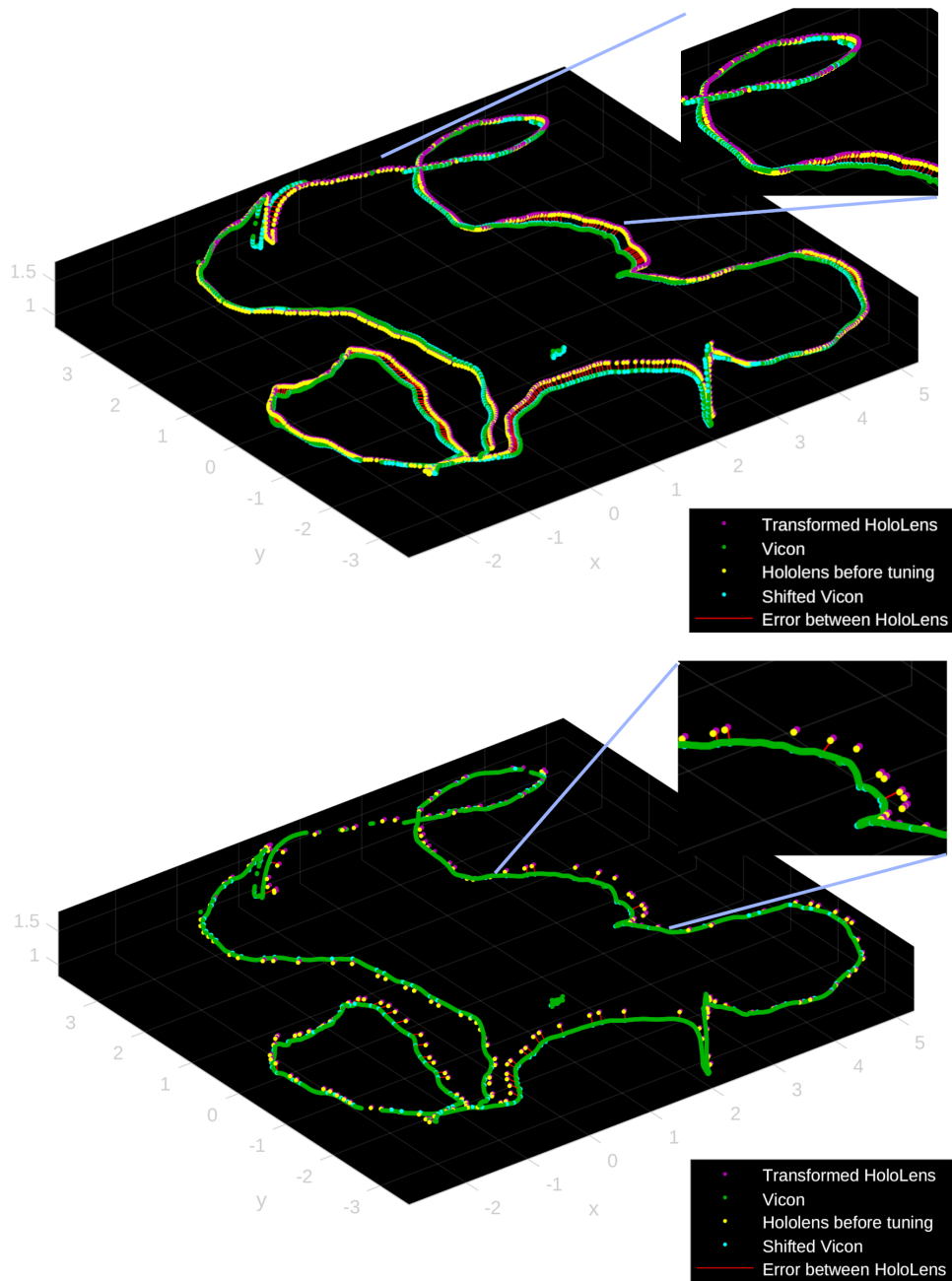
In first step, the following parameters are optimized:  $\mathbf{t}_i, \rho, \mathbf{S}$ . Then we fix these parameters and optimize the objective function  $\mathbf{f}_{\gamma,i}$  (in eqn. 4.10) with HoloLens camera poses  $\mathbf{e}_{k,i}$  as parameters. Lastly, the same parameters as in the first step are optimized. We assume the values of optimized  $\mathbf{t}_i$  are approximately equal to those in first pass if the right  $\beta$  was found. Moreover the drift of HoloLens cameras should follow Gaussian distribution. The three-step optimization is implemented in MATLAB. However, we found out that the calculations are running slowly (approximately a day long calculation for one  $\beta$ ), so a full evaluation was not performed.



**Figure 4.8:** Graph of the objective function values dependent on  $\beta$ .

#### 4.3.5 Mean error correction

Taking into consideration that HoloLens have a tracking error, we tested improvement of our results by using a *mean error correction*  $\epsilon$  (steps 3.1.2), 3.2.2) and 3.3.2 in fig. 4.5). Distances between aligned HoloLens camera poses from Vicon poses were calculated and then used the mean of these distances in the calculations (by transforming HoloLens poses by this mean distance, eqn. 4.11).



**Figure 4.9:** Example of HoloLens PV camera and depth (long throw depth) sensor aligned with Vicon. Aligned HoloLens cameras are purple, unaligned HoloLens are yellow, Vicon is green, and blue are the Vicon markers shifted by  $t_i$  to HoloLens. Red lines represents the errors between aligned HoloLens and Vicon.

### 4.3.6 Point cloud alignment

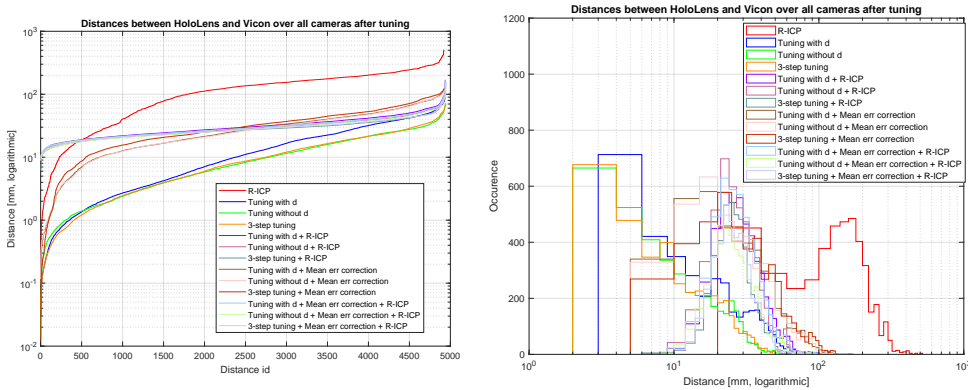
Finally, the HoloLens point clouds from depthmaps are transformed by the same transformation as HoloLens camera poses to Vicon markers origin.



For each point cloud obtained from k-th depthmap  $\mathbf{X}_k$  the recording, i.e. file, contains two matrices, i.e.  $\mathbf{A}_{D2O}$  and  $\mathbf{A}_{C2D}$ , describing the recorded transformation to the world coordinate system. The point cloud transformed to Vicon is:

$$\mathbf{X}_{V_k} = \frac{1}{\rho} \mathbf{P} \mathbf{S}^\top \mathbf{R}_{ii}^\top \mathbf{A}_{D2O} \mathbf{A}_{C2D} \mathbf{X}_k - \mathbf{d}' + \mathbf{R}_V \mathbf{t} + \epsilon \quad (4.11)$$

Where  $\rho$  is a factor of the Vicon vectors to be mapped onto HoloLens vectors,  $\mathbf{P}$  is transformation matrix obtained from Robust ICP [6], i.e. rough transformation of HoloLens to Vicon coordinate system,  $\mathbf{S}$  is rotation from Vicon to HoloLens coordinate system,  $\mathbf{R}_{ii}$  is the random rotation used for better convergence of R-ICP,  $\mathbf{d}'$  is  $\mathbf{d}$  when considering  $\mathbf{d}_V$  for optimization or a zero vector otherwise ( $\mathbf{d}$  is the origin of HoloLens coordinate system in Vicon coordinate system),  $\mathbf{R}_V$  is rotation from Vicon coordinate system to Vicon markers coordinate system,  $\mathbf{t}_i$  represents the translation between Vicon marker and i-th camera centers of HoloLens expressed in Vicon coordinate system. It differs depending on each camera (as they have a different position on the headset). If we would consider a rig of cameras, there would be the same  $\mathbf{t}_i$  for all sensor streams. However since the streams are not all synchronized and differ in framerate (namely the depth sensor), we do not consider it as a rig of cameras. The correction  $\epsilon$  represents the shift between HoloLens camera centers and the Vicon markers (the errors in figure 4.10).

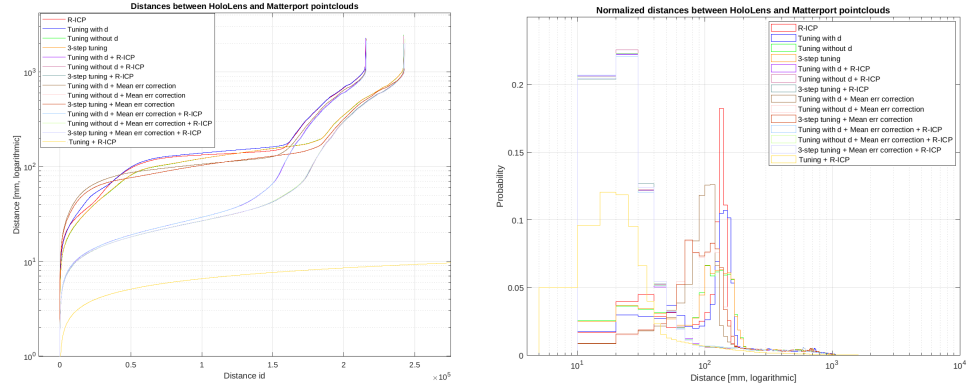


**Figure 4.10:** Distances between all aligned HoloLens camera centres and Vicon marker origins. The histogram can be seen in appendix B, stair graph was synoptical.

Each resulting point cloud  $\mathbf{X}_{V_k}$  is then aligned to Matterport scan by R-ICP (steps 3.1.3), 3.2.3), 3.3.1) and 3.3.3) in fig. 4.5). To measure distances between both point clouds the mean of nearest neighbors is used, all distances can be seen in fig. 4.11. The best result seems to be the best result of the tuning process (tuning without  $\mathbf{d}$  + mean error correction + R-ICP), where each point cloud was aligned by R-ICP individually.

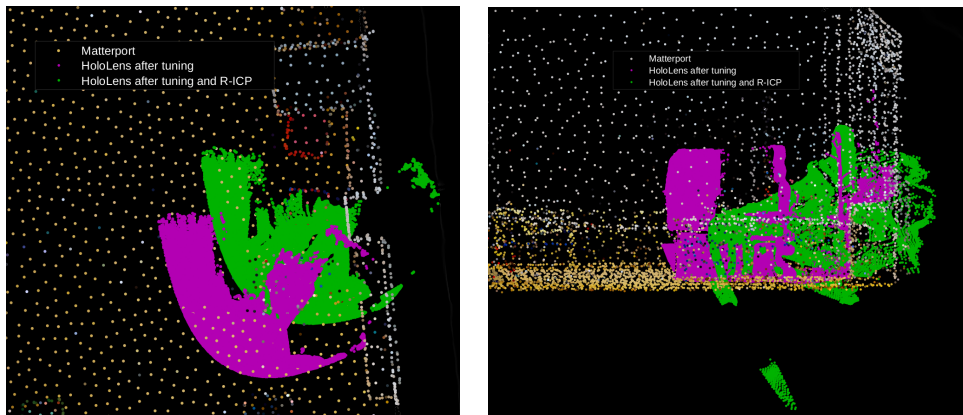
For point cloud alignment all transformed depthmaps from HoloLens are merged into one and then hashed (we multiply all points by 100, round them, take only the unique ones and divide them by 100). We can not align the

HoloLens depthmap to the whole Matterport model. We make a rough cutout of the Matterport model around the roughly aligned HoloLens depthmap. The Matterport point cloud is subsampled as well. This leads to different number of points in point clouds for each of previously mentioned optimization methods. To give the most objective results, all resulting nearest neighbour distance graphs (fig. 4.11) are normalized by using the value of the bin equals to occurrence/sum(occurrences).



**Figure 4.11:** Distances between aligned HoloLens point clouds and Matterport point cloud. (Note that the number of points in point clouds differ as point clouds are being cut accordingly to the rough alignment. Thus the histograms are normalized by the number of points aligned. Histogram can be seen in appendix A.

However, after detailed inspection some point clouds converged to a local minimum that is not ideal, see fig. 4.12. This state occurred most likely because there are dynamic objects in the environment that changed between Matterport map and HoloLens depth data acquisition. The next best result in fig. 4.11 amongst all presented distances is when using tuning without  $d$  with mean error correction and R-ICP on the whole sequence. Thus, several approaches lead to similar accuracy (expressed as a distance between nearest neighbours of the reconstructed point clouds). Therefore, we assume found camera poses by fine-tuning without  $d$  and with mean error correction and R-ICP (the path ending in step 3.2.3) in fig. 4.5) as the ground truth.



**Figure 4.12:** Examples of point clouds incorrectly aligned by R-ICP.



# Chapter 5

## GUI

This chapter focuses on the implementation of graphical user interface (GUI). The localization (HLoc) and alignment methods (R-ICP, FCGF + TEASER++, PREDATOR, FCGF + RANSAC) mentioned in this thesis are implemented as a node in Meshroom [41]. All the codes are available in the scope of the GIT repository <https://bit.ly/3ID9puj>.

### 5.1 Meshroom

Meshroom is an open-source software focused on photogrammetry and 3D reconstructions. Meshroom is published under the MPL2 license, i.e., the codes are fully available and free to extend. Moreover, it allows checking of the results and mid-results, building clear “graph-based” pipelines of connected nodes (individual methods), storing the pipelines, setting the parameters of individual methods and visualization of the pipeline in GUI. Meshroom has been selected as a “primary” library for implementation of GUI of all pipelines mentioned in this thesis.

Meshroom pipeline is represented as a directed oriented graph where nodes represent methods. Inputs are arguments (or outputs from previous nodes), see fig. 5.1. When running the pipeline, nodes are processed in sequential order according to their connections. Nodes that are independent on each other get processed in parallel. Pipeline is described by a JSON file containing all nodes, parameters and connections.

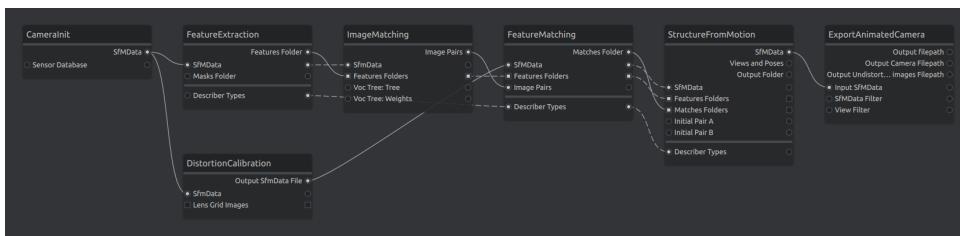


Figure 5.1: Example of a Meshroom pipeline

Each node is implemented in Python as a class inheriting from the Meshroom Node class. For full documentation and list of all possible parameters

see [42]. An example, the fig. 5.2 shows an `ExampleNode` implementation. The `category` item describes under which list of nodes `ExampleNode` can be found. The list of nodes is available when adding a new one in the GUI pipeline. Our example node has two input parameters: one is a path to file and the second is a choice parameter from 4 possible choices. The `uid` variable for each parameter (output parameters included) describes whether the value of this parameter plays a role while generating the hash name for cache folder. The `output file` parameter can be defined by the user or as the cache folder. The `processChunk` method contains code that is running during the processing of the node.

In the scope of this thesis, we created these nodes:

- `HlocQueryComposer` - to compose an HLoc localization query
- `HlocLocalizer`- to run HLoc localization
- `DensePointCloudAligner` - to align point clouds

## 5.2 HLoc localization pipeline

To ensure that all required dependencies for each localization and alignment method are installed, every method is wrapped in a virtual container (Singularity on Linux, Docker on Windows). All these containers are build by an initialization script `init.sh`.

There are two nodes focusing on HLoc localization: `HlocQueryComposer` and `HlocLocalizer`. The `HlocQueryComposer` node prepares a query file required for HLoc localization. It's input is a checkbox whether to use all images in a directory. If so, the path to the directory is required as the input. In the second case there is an array parameter for image paths. The next parameter is a checker box to indicate whether all images are taken by the same camera. This is used to either say that all images are from a camera with the same intrinsic parameters or that each image was captured by a different camera device. The camera intrinsics are in the form of an array of parameters structure. For each intrinsics entry there are four fields: `camera model`, `width`, `height` and `parameters`. The camera models are defined as COLMAP format [43]. Last parameter is the path to the HLoc map. The `HlocQueryComposer` node then composes the query file consisting of image paths, and camera intrinsics. This file is stored in the cache folder. The cache folder is then sent as a parameter to `HlocLocalizer` together with a path to HLoc map. The `HlocLocalizer` node runs a localization script in the container. The results are again stored in the cache folder of `HlocLocalizer` in a text file (containing image names and camera poses).

## 5.3 Point cloud alignment pipeline

The node for aligning point clouds is called: `DensePointCloudAligner`. The first parameter is an array of point clouds to be aligned. Next, we can

```

class ExampleNode(desc.Node):

    category = "NodeCategory"
    documentation = "Node documentation"

    # Input parameters
    inputs = [
        desc.File(
            name="fileInput",
            label="File input",
            description="Path to file input",
            value="./file.txt",
            uid=[0], # is used to generate cache folder path
        ),
        desc.ChoiceParam(
            name="choiceParameter",
            label="Choice parameter",
            description="Multiple choice parameter",
            value="A",
            values=["A", "B", "C", "D"],
            exclusive=True,
            uid=[], # not used to generate cache folder path
        ),
    ]

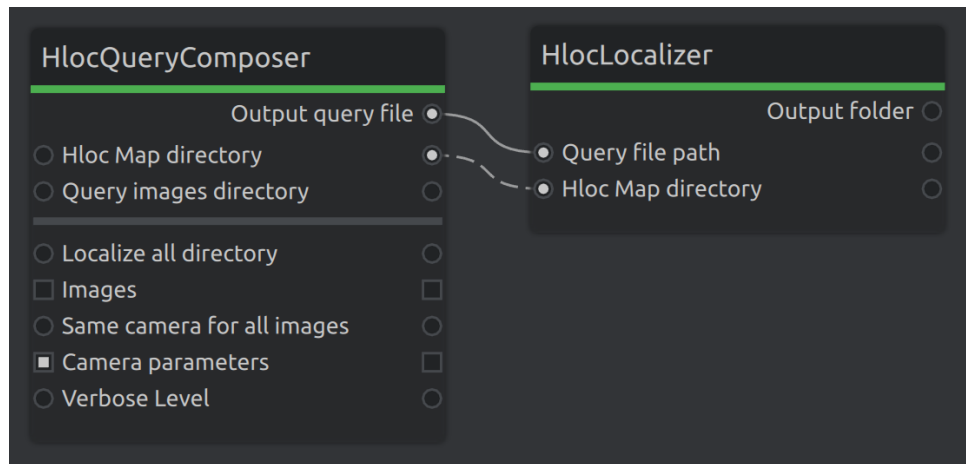
    # Output parameters
    outputs = [
        desc.File(
            name="output",
            label="Output folder",
            description="",
            value=desc.Node.internalFolder,
            uid=[],
        ),
    ]

    # Code that gets executed while running the node
    def processChunk(self, chunk):
        pass

```

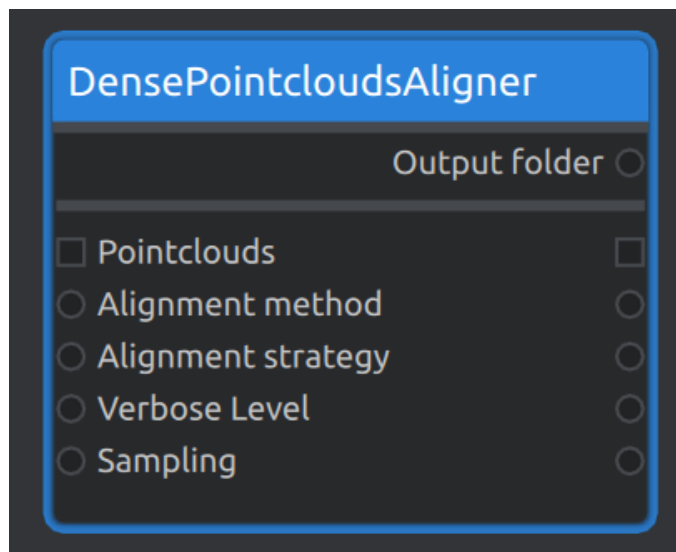
**Figure 5.2:** Example of a Meshroom node definition.

choose the method used to align point clouds (Concatenation, R-ICP, Overlap PREDATOR, FCGF + RANSAC, and FCGF + TEASER++). Then follows the alignment strategy, i.e. the order of point clouds alignments when



**Figure 5.3:** HLoc localization pipeline.

there is more than two point clouds. The only implemented strategy so far is the sequential one, where point clouds are sequentially aligned in their input order, i.e., the current point cloud is aligned to the result of the previous alignments. Another considered strategy would be a hierarchical approach. However, it is not implemented in the scope of this work. Parameter `verbose level` describes the amount of informational output in the terminal. Lastly, `sampling` determines if and how point clouds should be sub-sampled.



**Figure 5.4:** Dense point cloud aligner node.

The implementation has been uploaded to a publicly available Git repository. Nodes and pipelines were developed on Ubuntu 18.04.5 LTS, using Python 3.7.12.



## Chapter 6

### Localization improvement

This chapter evaluates the state of the localization from images. The results are further used as an initialization to methods improving the localization and the approaches for the pose refinement using dense point cloud alignment.

#### 6.1 The Hierarchical Localization method

For the evaluation of localization from images provided by Hierarchical localization (HLoc) [5] the Matterport scan of a B-315 laboratory is used. Note that this scan was performed earlier than the acquisition of the HoloLens depth data. The scanner was not available later on again. This leads to a possible source of errors in the localization as there is a number of dynamic objects (boxes, chairs, robots, etc.) that changed its pose. The HoloLens depth data is therefore not fully consistent with the Matterport scan.

The first map for localization is composed of dense point cloud from Matterport scan and 800 images from each image sensor (i.e., photo-video camera, and visible light cameras). Then 200 images for each sensor were used to localize. As can be seen in fig. 6.1, more than 90% of images localized with an error under 20 cm in translation and 5 degrees in rotation. Based on these results the point cloud alignment methods will be evaluated on synthetic data with errors from 0 to 20 centimeters in translation and 0 to 5 degrees in rotation at first. The translation error is calculated as a distance between ground truth camera pose and the aligned “*approximate*” camera pose. By approximate camera pose, we assume the shifted and rotated ground truth camera pose. The rotation error is calculated:

$$e = \arccos(0.5(\Delta_{\mathbf{R}} - 1)) \quad (6.1)$$

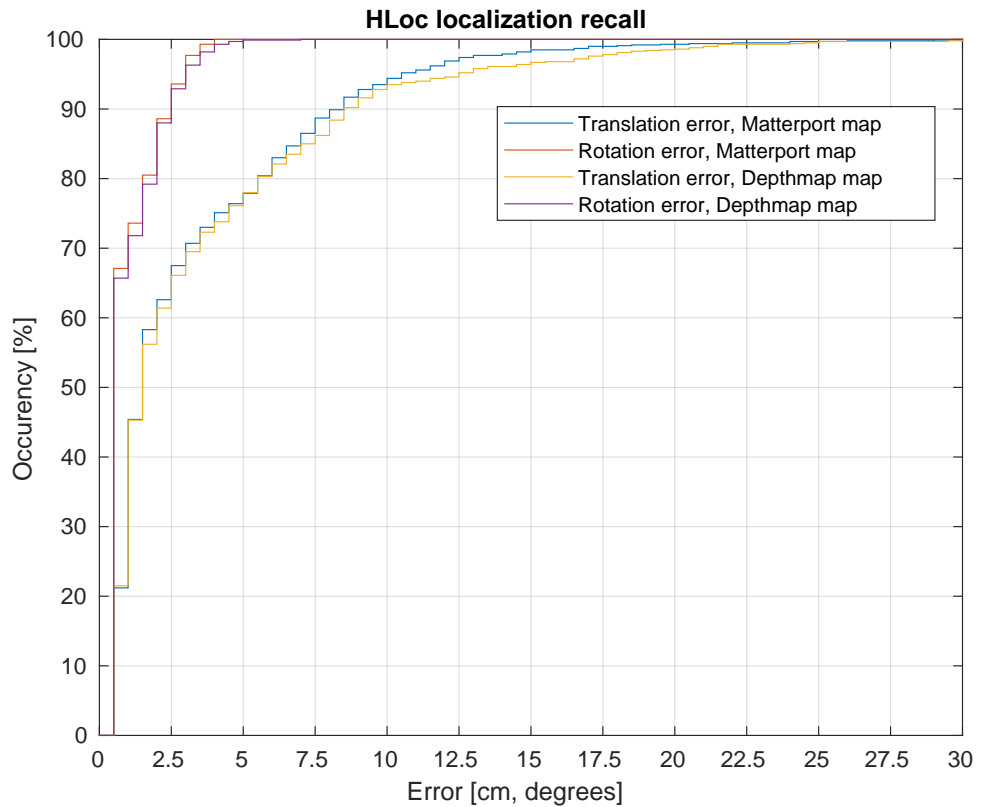
where

$$\Delta_{\mathbf{R}} = \mathbf{R}_{GT}^{\top} \mathbf{R}_a \quad (6.2)$$

The  $\mathbf{R}_{GT}$  is the rotation matrix of ground truth camera pose and  $\mathbf{R}_a$  is the rotation matrix of the approximately camera pose.

The second map used for evaluation is realized by merging depthmaps of HoloLens leading to a map of the laboratory without any dynamic objects.

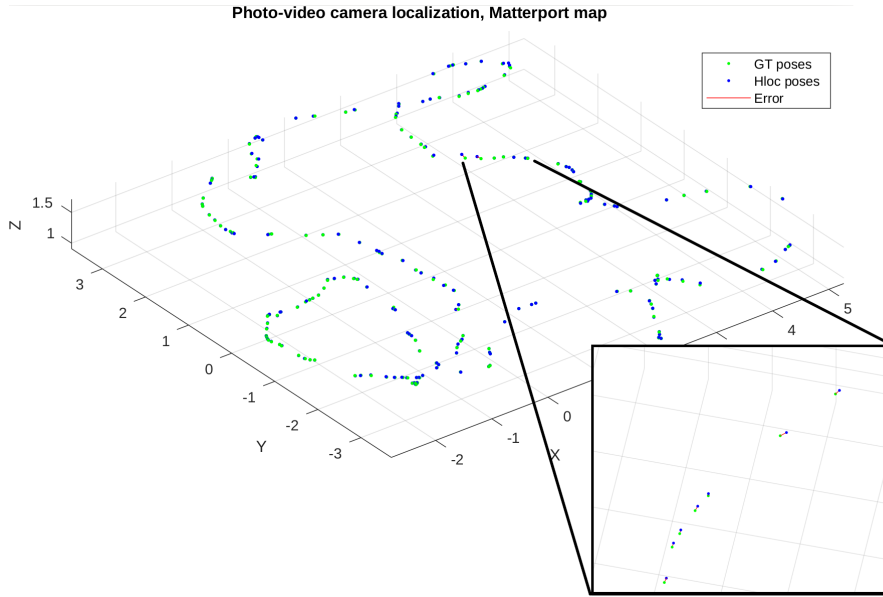
This map is used to test the dense point cloud alignment methods on the map composed from the same source of data, to see how much the source of data influences the alignment. The HLoc map was calculated again using 800 images for each camera. The 200 images were employed for the localization. Results (fig. 6.1) are close to the HLoc localization on Matterport map, which leads to the same approach when evaluating point cloud alignment, i.e., the testing shift of ground truth camera position is between 0-20 cm and 0-5 degrees.



**Figure 6.1:** The Hloc localization accuracy. More than 95% of the camera poses were localized within 20 cm and 5 degree distance threshold.

## 6.2 Point cloud alignment methods

This part is focused on evaluating the localization accuracy improvement obtained by several methods. First, all alignment methods will be compared on simulated data (distorted ground truth) according to the accuracy of HLoc observed in fig. 6.1. The best performing method is further evaluated on real data.



**Figure 6.2:** Overview of the Hloc localization for each camera.

### 6.2.1 Testing alignment methods

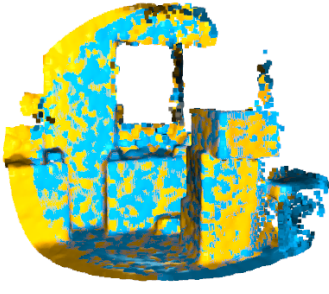

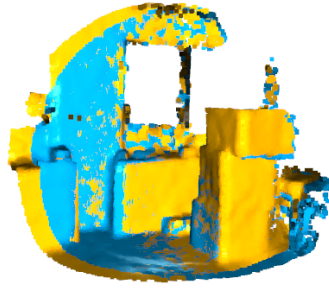
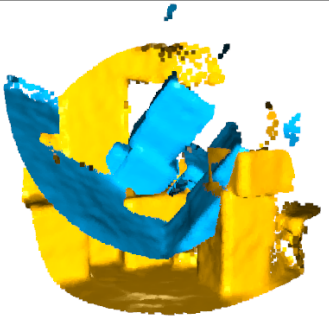
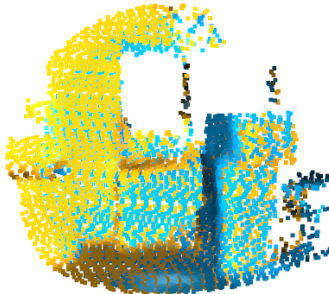

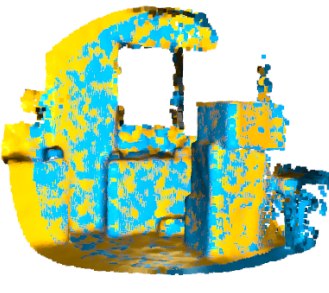
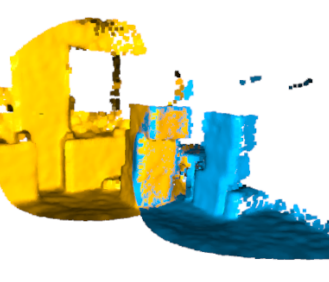
To ensure that all alignment methods are suited for the goal of this work and that they are working well on our acquired data, we run an alignment test on several sequential depthmaps from HoloLens.

All methods were running on the same sequence, where the point clouds are overlapping and no dynamic objects are present. The alignment was performed on the first point cloud of the sequence, i.e., we align the  $(i + 1)$ -th depthmap to the  $i$ -th depthmap. The results can be seen in table 6.1.

In our example, R-ICP can align the sequence up to the fifth point cloud, FCGF + RANSAC method fails on the fifth one as well as FCGF + TEASER++. However, this may be caused by subsampling point clouds due to memory consumption of the TEASER++ method. The last method (Overlap PREDATOR) aligned the depthmaps up to the seventh one, failing on the eighth one. This evaluation is only a check that all methods are working well and that they are integrated into Meshroom correctly.

### 6.2.2 Simulated data

The *simulated experiment* uses the ground truth depthmaps from HoloLens. The dataset consists of 193 point clouds of the B-315 laboratory. These point clouds are moved with translation (from 0 up to 20 cm using 4 cm step) and the rotation around random axis (from 0 up to 5 degrees with 1 degree step). These approximate camera poses are aligned to the cutout of the Matterport map and a map consisting of HoloLens depthmaps transformed into the point cloud. Cutouts are cropped from the map by projecting the camera view with increased field of view (by 5 degrees in all direction) to consider the

Method	Point clouds 0 and 4	Point clouds 0 and 6
R-ICP		
FCGF + RANSAC		
FCGF + TEASER++		
Overlap PREDATOR		

**Table 6.1:** Alignment results on a depthmap sequence out of the HoloLens recording. We have shown that all the alignment methods work well if the source of data is the same and they have a significant overlap.

possible error in rotation and with 1m longer viewing depth to consider the translation error, see fig. 6.3.

The depth camera pose is calculated as mentioned before except the camera

rotations are rotated around the X axis by 180 degrees since the HoloLens cameras look in the negative Z axis direction. The depth frame field-of-view is 60 x 54 degrees, for the alignment, we assume the field-of-view about 5 degrees larger in all directions. The desired field-of-view for point cloud cutouts from Matterport map is 70 x 64 degrees, which is approximately 1.18 times larger. This coefficient (called  $k$ ) is then used to calculate the desired focal length and principal point for camera calibration matrix. The focal length is then calculated depending on the horizontal and vertical field of view ( $H_{FOV}$ ,  $W_{FOV}$ ) and frame resolution ( $W \times H$ ):

$$f_x = \frac{W}{2} * tg^{-1}\left(\frac{H_{FOV}}{2}\right) \quad (6.3)$$

$$f_y = \frac{H}{2} * tg^{-1}\left(\frac{W_{FOV}}{2}\right) \quad (6.4)$$

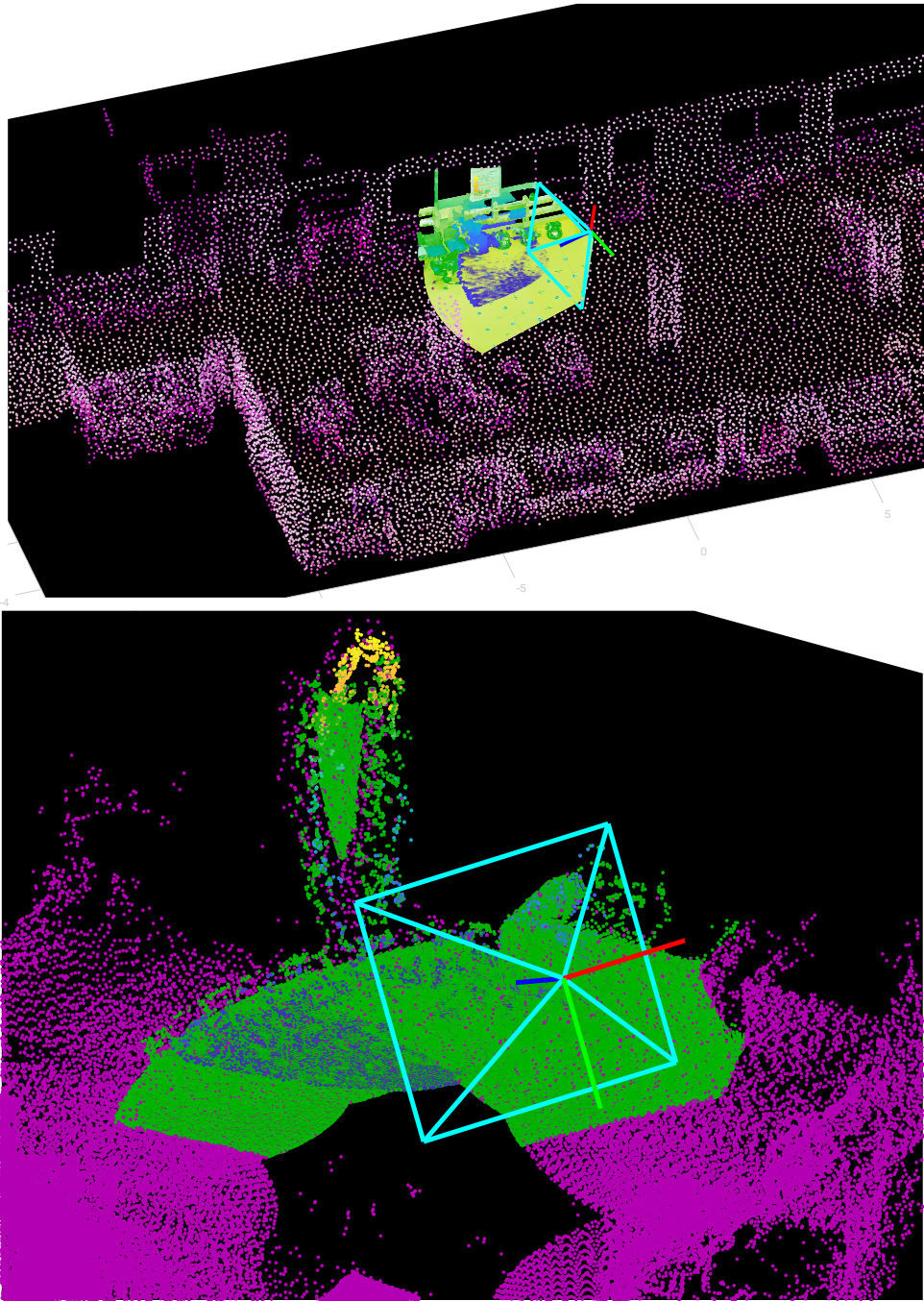
The calibration matrix is then construed:

$$K = \begin{bmatrix} \frac{f_x}{k} & 0 & \frac{W}{2} * k \\ 0 & \frac{f_y}{k} & \frac{H}{2} * k \\ 0 & 0 & 1 \end{bmatrix} \quad (6.5)$$

The camera defining *pyramid*, i.e, the corner points of the camera frame projected into the world coordinate system that represent the plains of the view frustum, is then transformed to world coordinate system. For all planes of the view frustum (consisting of camera centre  $\mathbf{c}$ , and two neighboring end points of the frustum  $\mathbf{a}$ ,  $\mathbf{b}$ ). For all points ( $\mathbf{x}$ ) from a point cloud we check that the determinant ( $|\dots|$  notes determinant) of the following 3x3 matrix is positive (i.e, if points lie in the camera view frustum):

$$\left| \begin{bmatrix} a_1 - c_1 & b_1 - c_1 & x_1 - c_1 \\ a_2 - c_2 & b_2 - c_2 & x_2 - c_2 \\ a_3 - c_3 & b_3 - c_3 & x_3 - c_3 \end{bmatrix} \right| > 0 \quad (6.6)$$

The depth sensor viewing distance is about 3.5 m [31]. To include the possible error in localization the cut's viewing distance is set to 4.5 m and all points above this distance from the camera are ignored.



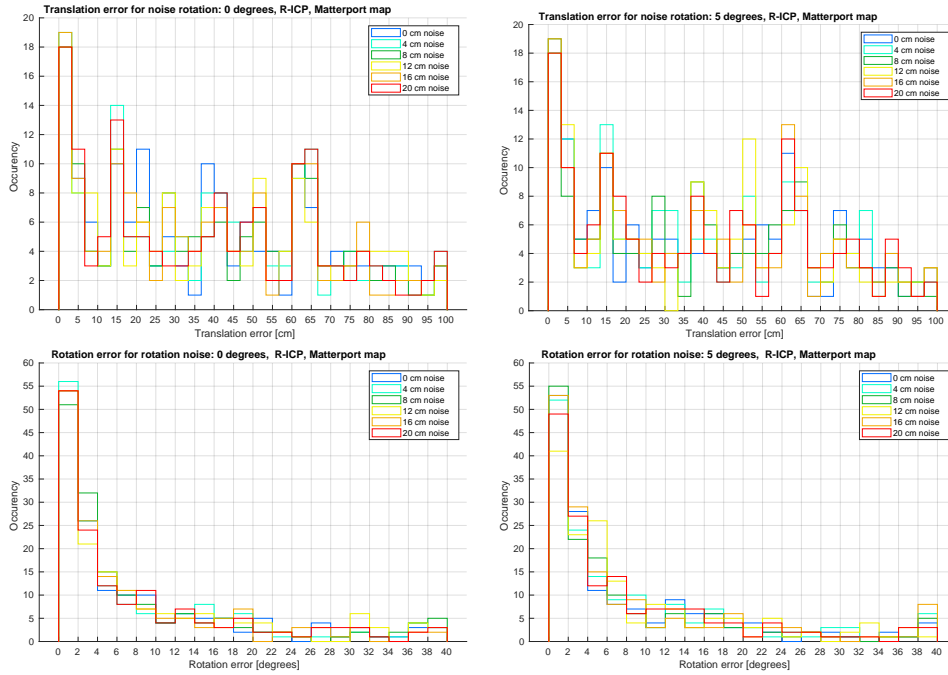
**Figure 6.3:** Matterport (upper) and depthmap (lower) maps cutout example. Purple point cloud is the map, green is the cutout of the map and blue is ground truth depthmap from HoloLens assuming the camera pose.

### 6.2.3 Alignment methods evaluation

In this section we evaluate the accuracy of alignment of 4 methods: R-ICP, FCGF + TEASER++, FCGF + RANSAC and Overlap PREDATOR.

### 6.2.4 R-ICP

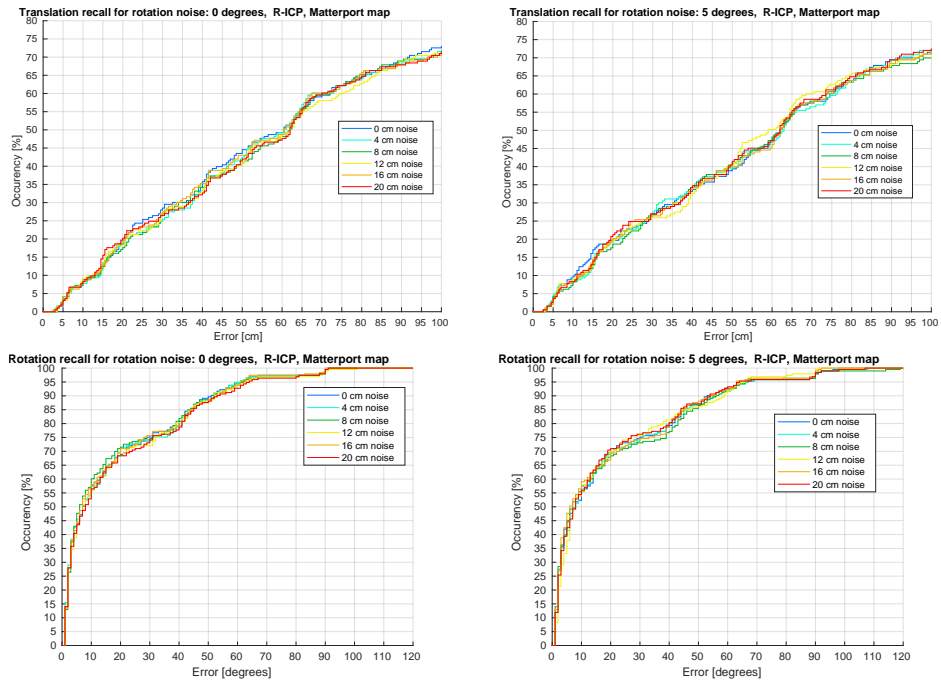
The first tested method is R-ICP [6]. It is publicly available under the MIT license as a C++ library. Output of the R-ICP is a transformed point cloud and a 4x4 transformation matrix. First, the accuracy of R-ICP alignment was evaluated on HoloLens depthmaps and cutouts of Matterport scan. The HoloLens depthmaps are shifted randomly in translation and rotation (as has been described in section 6.2.2). As can be seen in figures 6.4 and 6.5, shown errors follow closely similar (almost identical) pattern. As other values of noise produce same results, only two examples have been included. Full results can be seen in the appendix B.



**Figure 6.4:** The alignment errors for R-ICP over all HoloLens point clouds constructed with approximate camera poses on the Matterport map.

The same evaluation was also performed on cutouts of a map composed of HoloLens depthmaps. This leads to alignment of two point clouds with no changes in the environment and from the same source. It can be observed in results 6.6, 6.7, that measured errors again follow very similar trends (full evaluation in appendix B).

## 6. Localization improvement

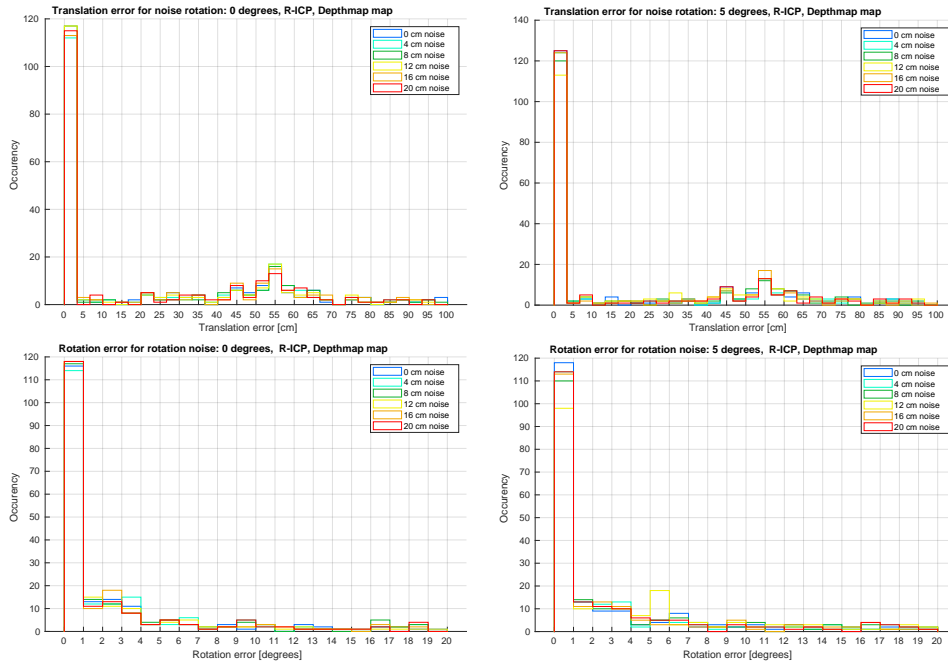


**Figure 6.5:** Alignment recall for R-ICP over all HoloLens point clouds constructed with approximate camera poses on the Matterport map.

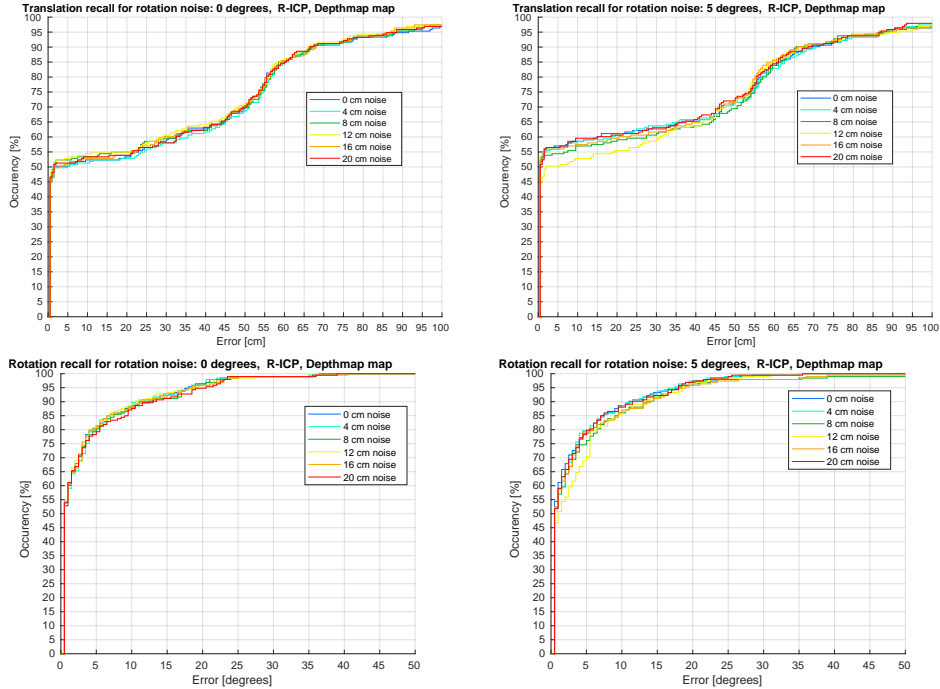
Note that in previous evaluations R-ICP fails to align point clouds even with no translation and rotation noise. As can be noticed in 6.8, The first case, when R-ICP does not converge to the correct result is when there is not enough distinctive features (floors, for example). The second case is when the aligned point cloud misses some distinctive parts of the map. The alignment is also influenced by the fact, that a larger field-of-view is considered for the map cutout to include the same points that can be seen with any approximate camera position.

In conclusion, we can see, that R-ICP performs better on a map composed from HoloLens depthmaps.

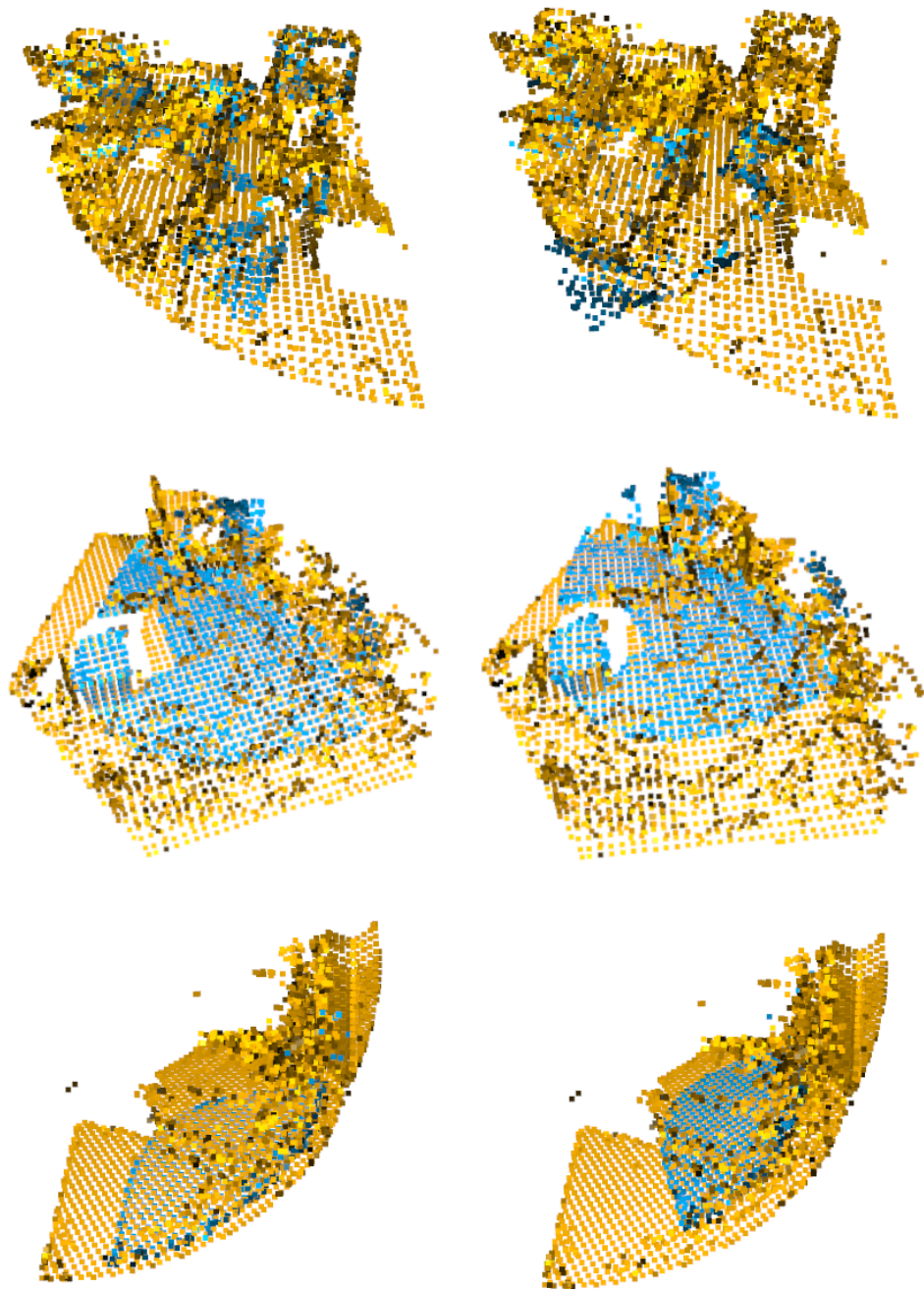




**Figure 6.6:** Alignment errors for R-ICP alignment over all translation noised point clouds on a map composed from HoloLens depthmaps.



**Figure 6.7:** Alignment recall for R-ICP alignment over all translation noised point clouds on a map composed from HoloLens depthmaps.

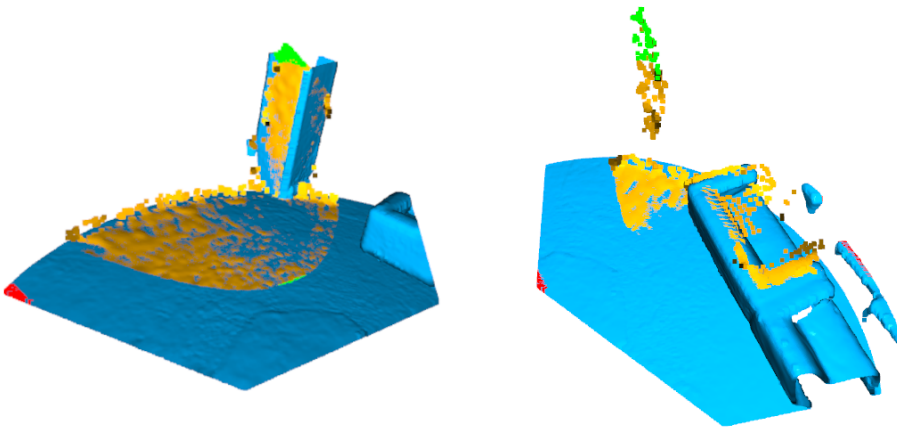


**Figure 6.8:** Examples of high error alignment by R-ICP, the initial state of point clouds is on the left and on the right is the resulting alignment. In the first row, the alignment fails because there are some objects on the map cutout that are missing in the depthmap. The second and third row shows, that point cloud with little to no objects in it is very likely to be aligned incorrectly.

### 6.2.5 FCGF + TEASER++

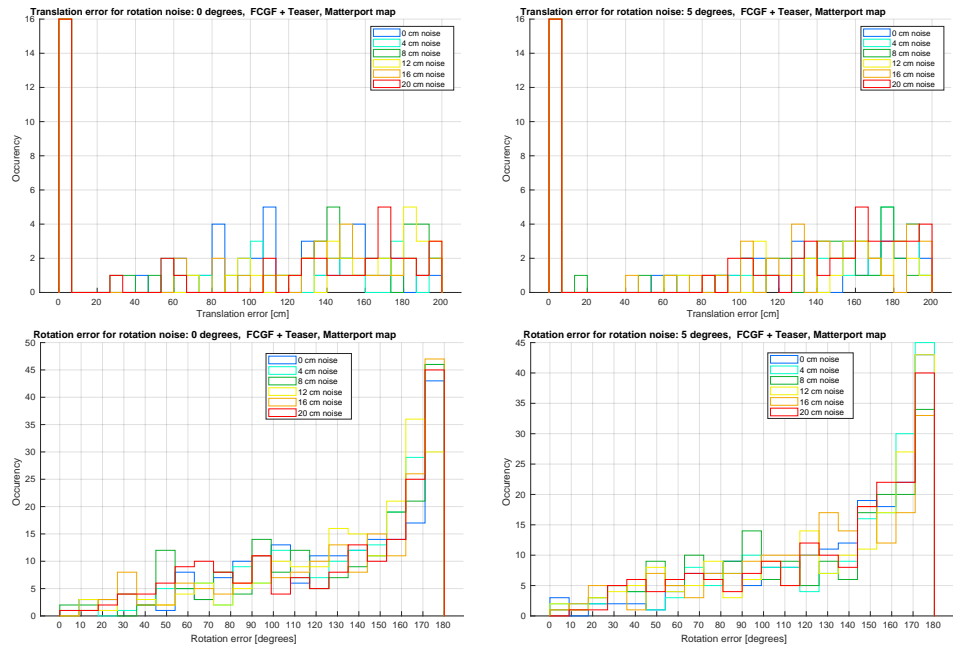
The second tested method is FCGF [7] + TEASER++ [9]. The code of TEASER++ is publicly available as C++ and Python libraries under the MIT license. The Python library was used for this evaluation. The FCGF + TEASER++ method first runs FCGF to obtain feature descriptors for both point clouds. We use a pre-trained model (ResUNetBN2C) provided by the authors (that was trained on indoor datasets). FCGF output are the feature descriptors for voxels of the point cloud so it requires a fixed voxel size. In this experiment, the voxel size is set to 2.5 centimeters. These descriptors are further compared to find tentative matches between a pair of point clouds. Only the closest 100 tentative matches were considered as input for TEASER++ alignment method. We selected the points with closest descriptors as tentative matches. When TEASER++ got point clouds created from all tentative matches, it would run out of memory and get killed by the operating system. According to the authors, the problem is "due to the algorithm of generating the compatibility graph, with the memory consumption of  $O(N^2)$ ." [44]. Since there are approximately between 20 to 30 thousand correspondences the computer runs out of memory. We faced this challenge by choosing only mutually closest descriptors from tentative matches as the input of TEASER++. Further, we formatted the point cloud correspondences into the required format, i.e., TEASER++ assumes correspondences between two point clouds to be at the same row in the aligned .ply files.

We can see in the alignment results (fig. 6.10, 6.11, 6.12, 6.13) errors are high and only a small fraction of point clouds get aligned correctly. This is caused by using only mutually nearest features. For both point clouds there are approximately only 100 mutually nearest features (see fig. 6.9). If the detected features are incorrectly paired, TEASER++ will not align the point clouds correctly.

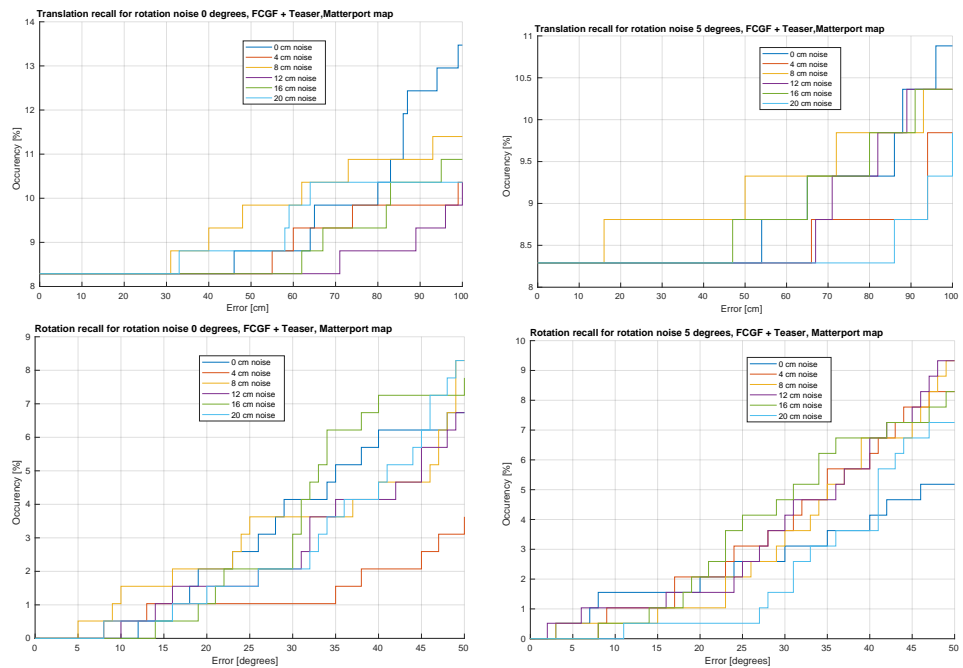


**Figure 6.9:** Examples of only mutually nearest feature descriptors from FCGF, features considered for alignment are red and green.

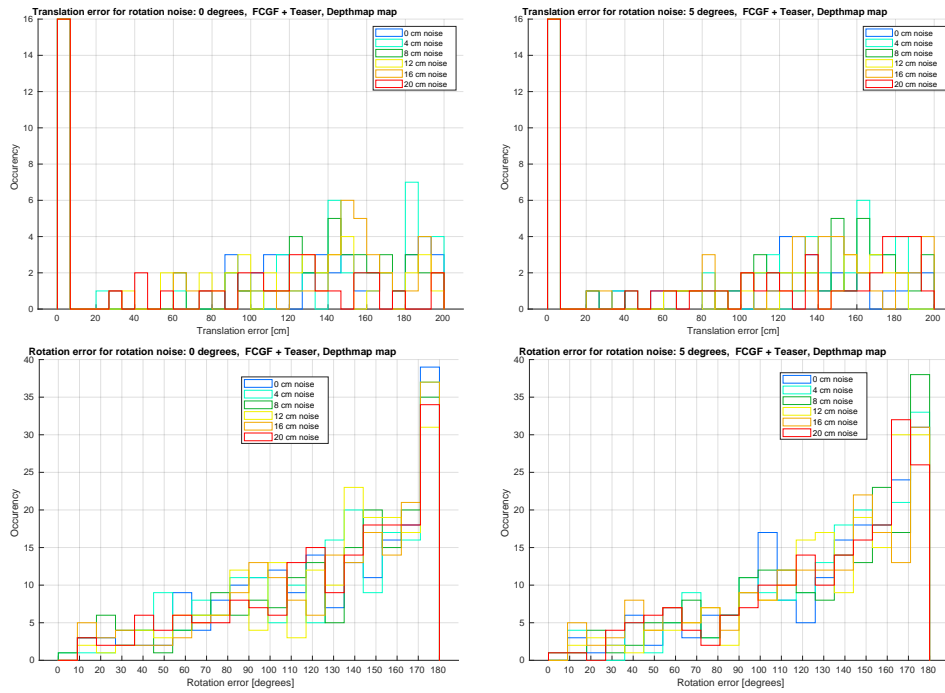
## 6. Localization improvement



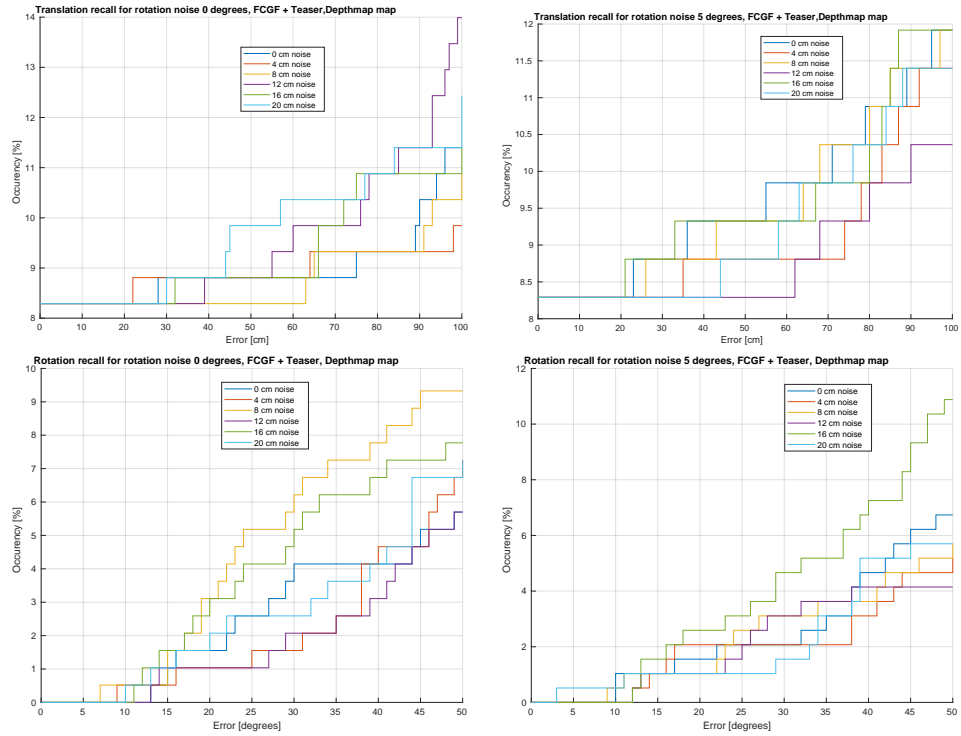
**Figure 6.10:** Alignment errors for FCGF + Teaser++ over all translation noised point clouds on the Matterport map.



**Figure 6.11:** Alignment recall for FCGF + Teaser++ over all translation noised point clouds on the Matterport map.



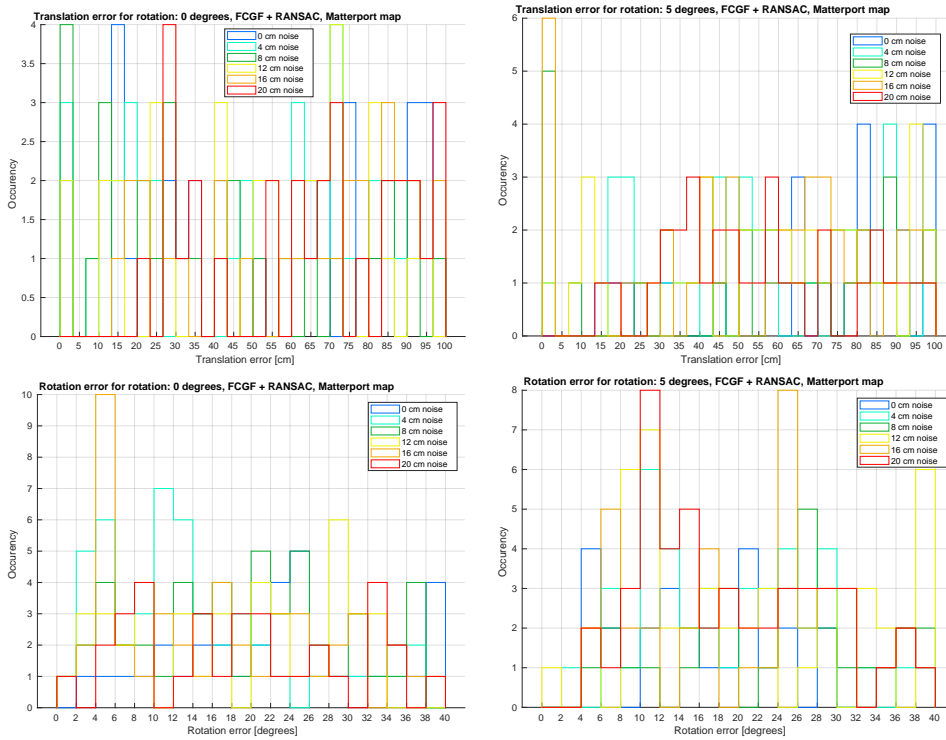
**Figure 6.12:** Alignment errors for FCGF + Teaser++ over all translation noised point clouds on the depthmap map.



**Figure 6.13:** Alignment recall for FCGF + Teaser++ over all translation noised point clouds on the depthmap map.

## 6.2.6 FCGF + RANSAC

The third tested method is FCGF[7] + RANSAC. The FCGF describe the point cloud as in the previous section. RANSAC is performed on obtained tentative matches and estimate the similarity transformation from minimal sample of correspondences. Unlike Teaser++ we can consider all candidate features, not only mutually nearest ones. RANSAC implementation used in this thesis is the MATLAB implementation of MLESAC [45] which chooses a solution based on the likelihood rather than the number of inliers. Results of this alignment method can be seen in fig. 6.14, 6.15, 6.16, 6.17.



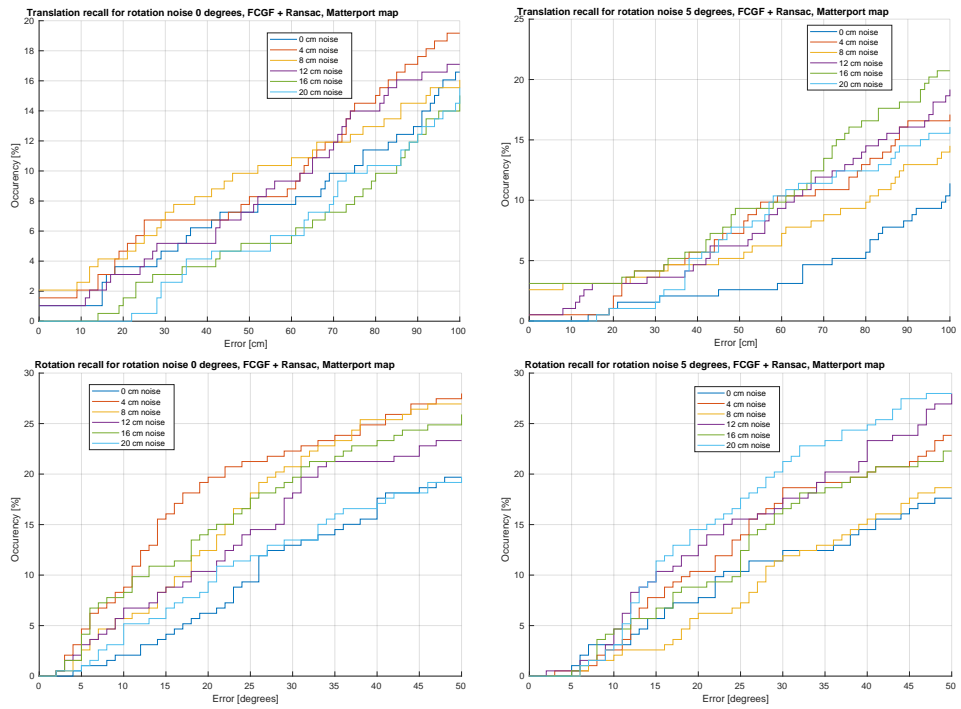
**Figure 6.14:** Alignment errors for FCGF + RANSAC over all translation noised point clouds on the Matterport map.

### 6.2.7 Overlap PREDATOR

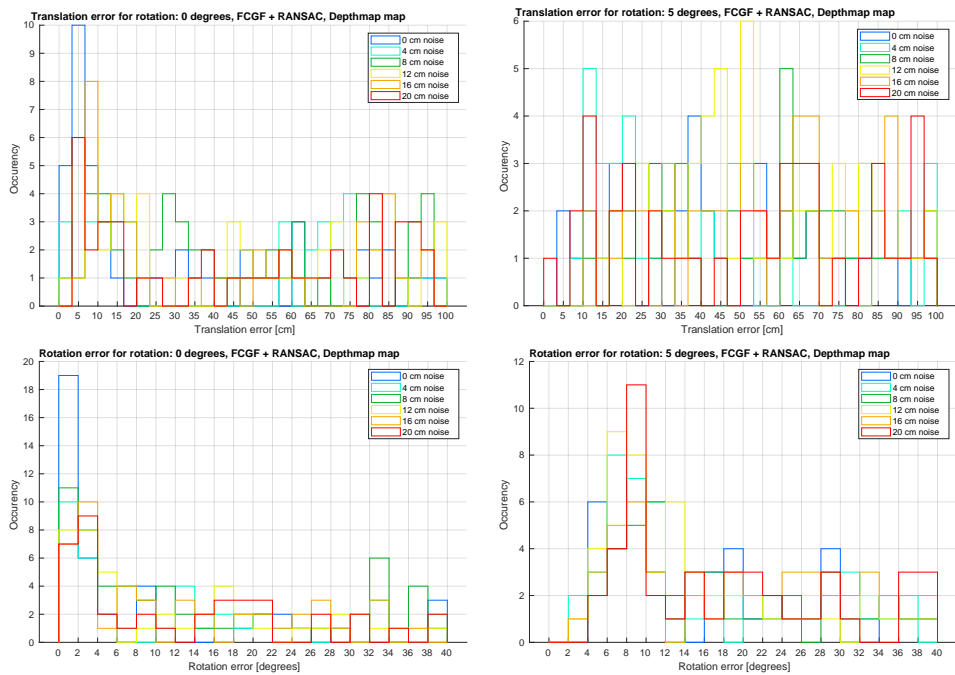
The last tested method is Overlap PREDATOR. Overlap PREDATOR is a model for pairwise point cloud registration specialized in point clouds with low overlap ( $\leq 30\%$  overlap). PREDATOR focuses the sampling of tentative matches to overlapping areas. The code is publicly available under the MIT license [46]. The library is written in Python using Pytorch and CUDA. The pre-trained model for indoor scenes is being used in our evaluation. The output from the PREDATOR alignment is a 4x4 transformation matrix. It has been observed that this method does not behave consistently and for the same input we get varying results (fig. 6.18). We found out that it is caused by PREDATOR running RANSAC as the last step to estimate similarity transformation from selected tentative matches from overlapping areas. The version of RANSAC used is the open3d version [47].

We tested the alignment by PREDATOR on all noised data and both maps, i.e., Matterport and HoloLens. We can see (in fig. 6.19, 6.20, 6.21, 6.22), that PREDATOR does not produce reliable results.

## 6. Localization improvement

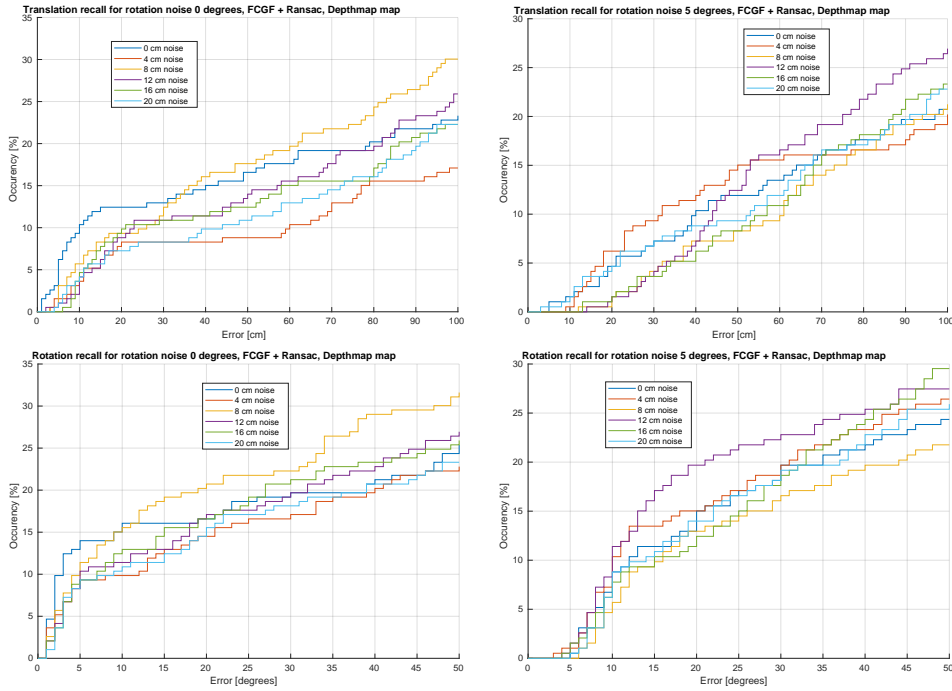


**Figure 6.15:** Alignment recall for FCGF + RANSAC over all translation noised point clouds on the Matterport map.



**Figure 6.16:** Alignment errors for FCGF + RANSAC over all translation noised point clouds on the depthmap map.





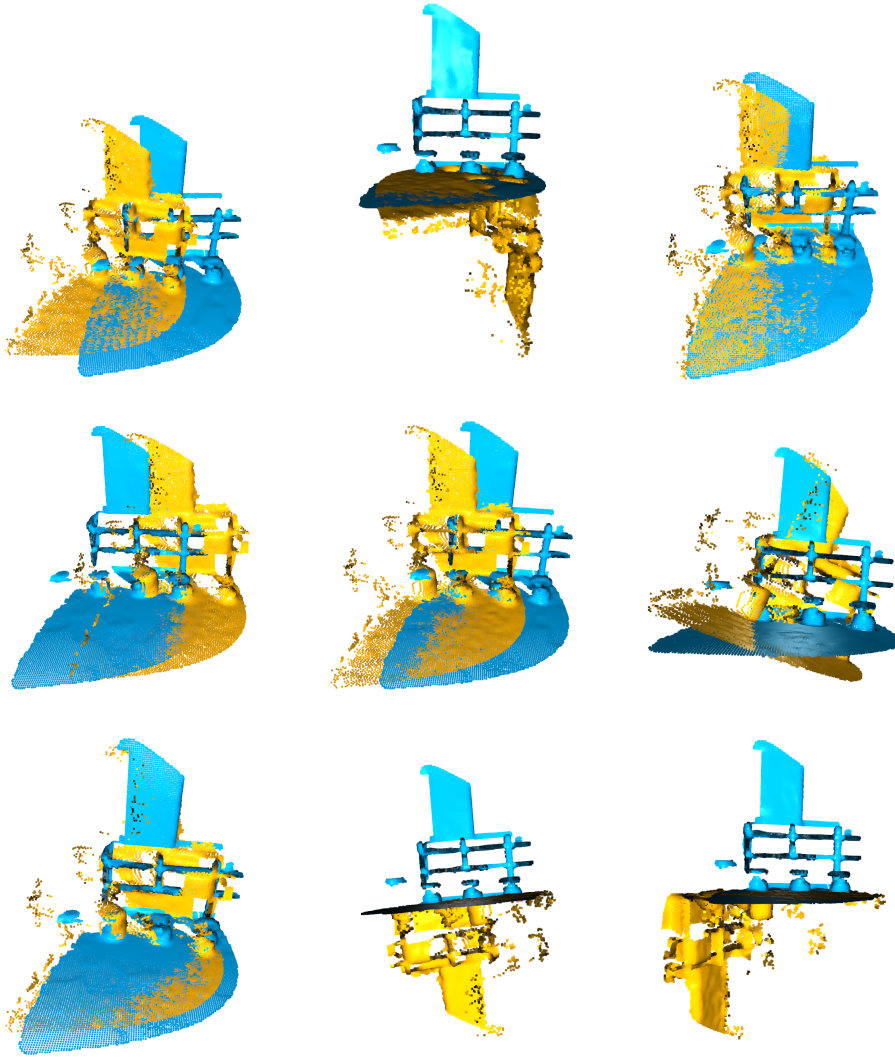
**Figure 6.17:** Alignment recall for FCGF + RANSAC over all translation noised point clouds on the depthmap map.

### 6.2.8 Localization fine-tuning

As the last step, we choose the best performing method and run it on the real camera poses obtained by HLoc on our validation dataset. Based on the previous evaluation, the R-ICP method is chosen as the best fine-tuning method. Its runtime is the fastest, and outputs are the most reliable out of all evaluated methods.

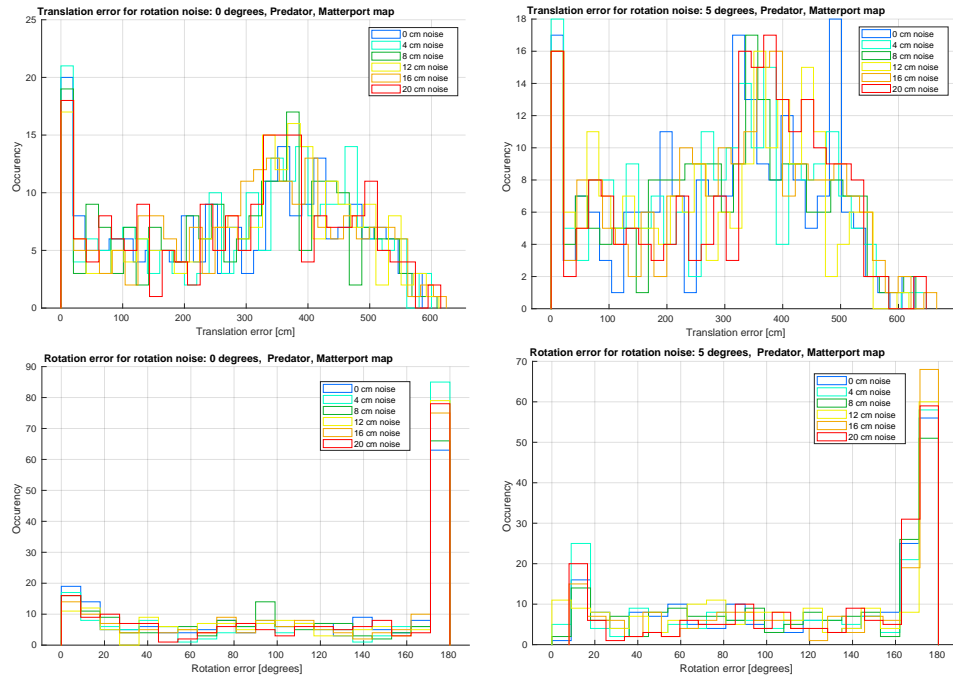
At first, HLoc localization runs on 200 images from the photo-video camera. Then, for each of the image camera poses, we consider the depthmap temporally closest, i.e., the depthmap with the closest timestamp to the query image timestamp. The depthmap is then transformed by found camera extrinsics by HLoc to the Vicon coordinate system. The approximate camera pose is further fine-tuned by R-ICP (both Matterport map and depthmap map are tested as the map of the environment). The transformation from the alignment is then applied to obtained poses. It can be seen (fig. 6.23), that on our dataset the fine-tuning does not improve the localization accuracy.

This may be due to two things. First, the photo-video camera and the depth sensor are not synchronized. This leads to additional errors. Second, R-ICP aligns approximately 60% point clouds within 5 cm and 5 degrees. This may be caused by depth data from HoloLens not being accurate enough. We can also see the influence of using the map composed from different sources, i.e., the Matterport instead of HoloLens depthmaps. Aligning of the depthmaps to Matterport scan works much worse than aligning to the map realized by depthmaps.

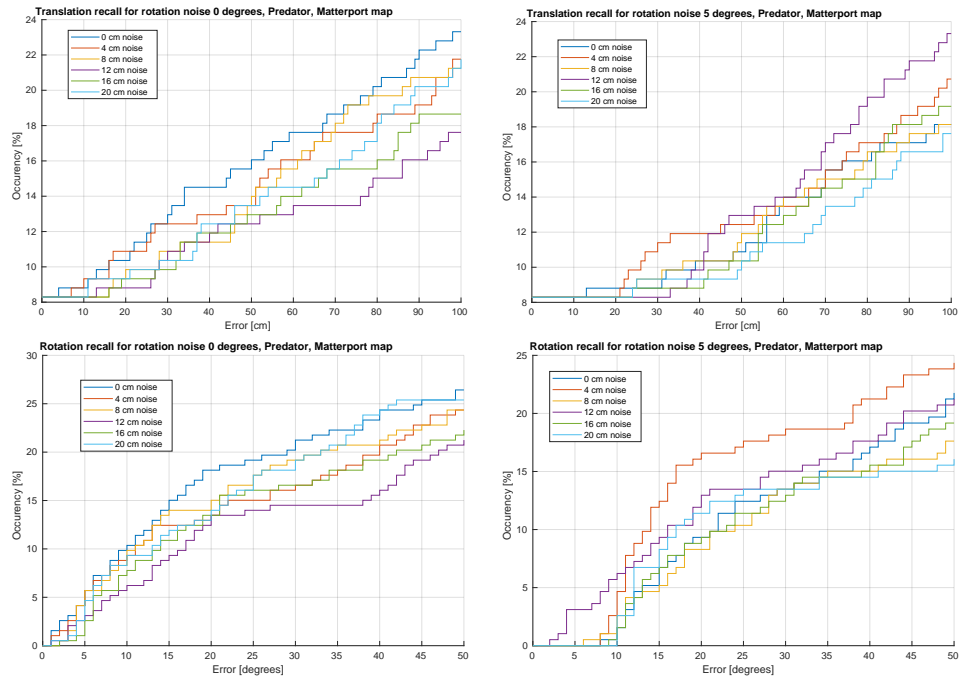


**Figure 6.18:** Results of HoloLens and Matterport point clouds alignment by PREDATOR several times with same pair of point clouds and same parameters, each time the alignment gave a different result.

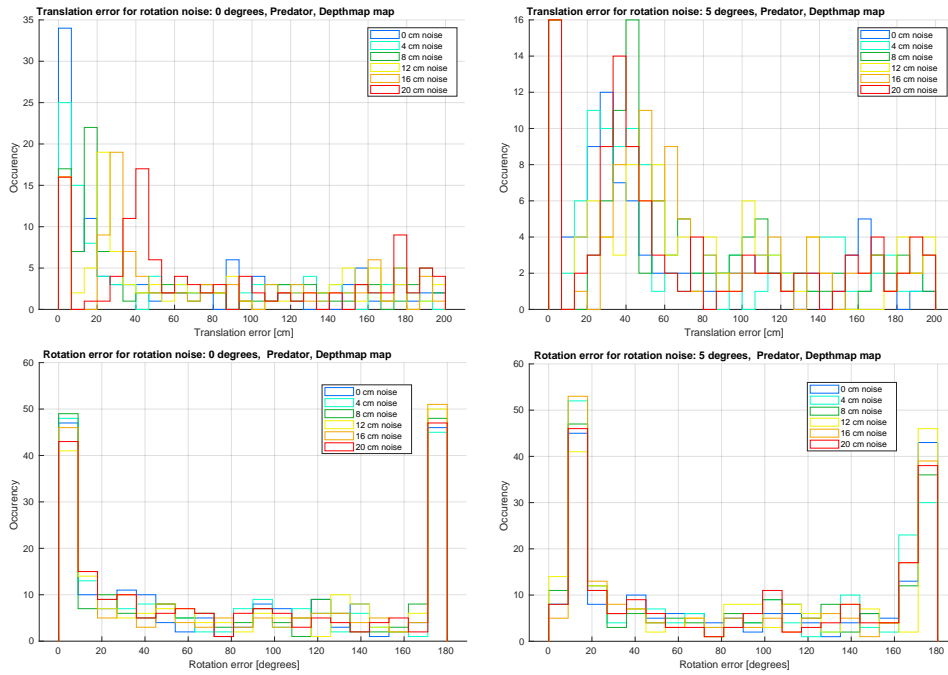
## 6. Localization improvement



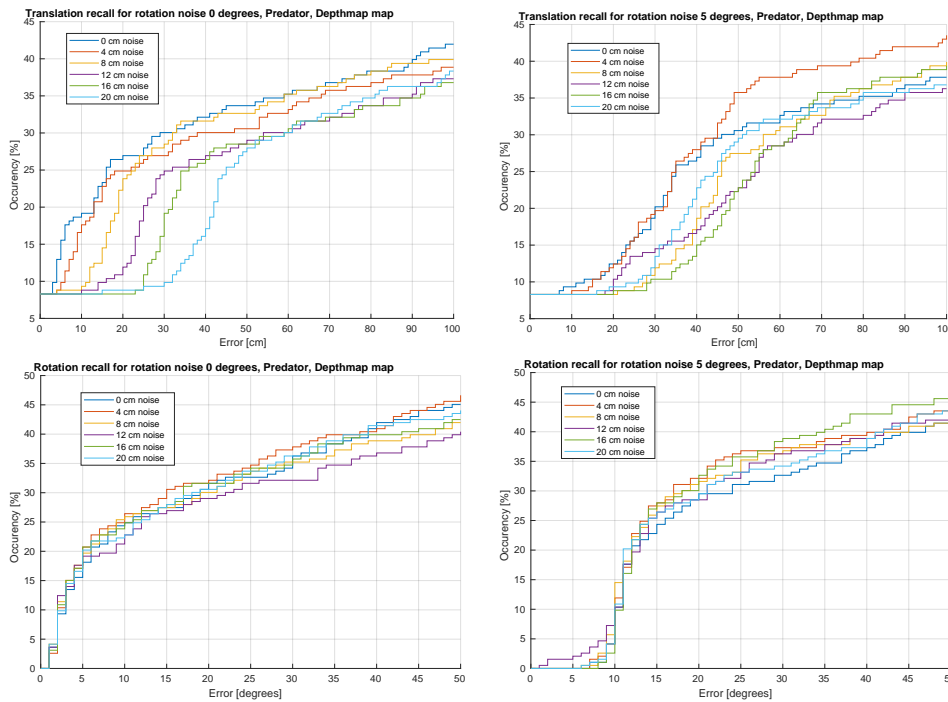
**Figure 6.19:** Alignment errors for PREDATOR over all translation noised point clouds on the Matterport map.



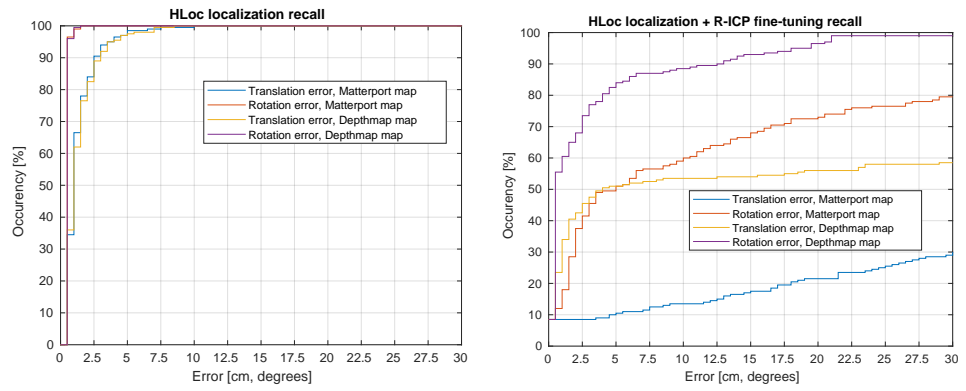
**Figure 6.20:** Alignment recall for PREDATOR over all translation noised point clouds on the Matterport map.



**Figure 6.21:** Alignment errors for PREDATOR alignment over all translation noised point clouds on a map composed from HoloLens depthmaps.



**Figure 6.22:** Alignment recall for PREDATOR alignment over all translation noised point clouds on a map composed from HoloLens depthmaps.



**Figure 6.23:** Localization recall before (left) and after (right) fine-tuning by point cloud alignment.

# Chapter 7

## Conclusion

Most modern smartphones can be treated as AR devices if an accurate localization is available. Current techniques of the localization from images perform well in an environment rich in textures and distinctive image features. However, texture-less areas with repetitive structures are still challenging, which was observed in the scope of the related EU H2020 grant ARTwin. The localization of images from the construction site or a factory setting is tricky.

The purpose of this thesis was to investigate if the data from the depth camera improves the accuracy of found camera poses from RGB images. All the relevant literature is summarized for this task, and related datasets are reviewed. Because of the lack of a ground truth dataset consisting of RGB-D images with known camera poses and accurate dense point cloud realizing the map, we collected, described, and published one. Based on this dataset, several recent methods for dense point cloud alignment (i.e., the alignment of the AR device depthmaps with the environment map) using the rough pose estimate from image-based localization were tested. In detail, we tested the HLoc localization together with the dense point cloud alignment methods R-ICP, FCGF + TEASER++, FCGF + RANSAC, PREDATOR. Moreover, we wrapped all the tested methods into containers and implemented the graphical user interface to simplify the repeated use and parameter tuning of all the proposed techniques and experiments.

The conclusion is that we were not able to improve the localization accuracy for several reasons. The first one is related to the ground truth data composition. There are several challenges we faced, i.e., the camera images captured by the HoloLens device are not temporally synchronized, Vicon tracking system gets lost if multiple markers are tracked at the same time, and Matterport map was recorded at different time, which incorporated dynamic objects in the map. We evaluated a list of algorithms, robust estimators, and optimizations of different objective functions to recover the ground truth HoloLens camera poses. However, there is still space for improvement. Because of the non-trivial challenges and limited accuracy of the Matterport map, the ground truth camera poses are loaded with an error in small units of centimeters. The second reason is that we did not have a technology for accurate localization (i.e., the composition of the ground truth camera poses and related RGB-D images) in the places of interest, e.g., the corridors of a

factory or the construction site. We captured the HoloLens recordings of these environments, but any accurate tracking service (like Vicon) is unavailable there. Thus, the problem of collecting ground truth data becomes even more challenging. On the opposite side, evaluating our methods allows understand this yet not investigated task even more deeply. For example, the experiments show that FCGF point cloud descriptor is highly sensitive to the source of data, e.g., the point cloud from a depthmap has different descriptors than the point cloud of the same area from the Matterport scanner. The same holds (not as strictly as in the previous example) even if we assume the alignment of the depthmap to the map composed of depthmaps, i.e. when the map is composed of multiple depthmaps while localization assumes only one and the point density differs.

The interesting output of this thesis is also a list of research ideas that appeared during the evaluation. For example, the camera pose fine-tuning should be improved if the FCGF descriptor is retrained to our use-case depthmaps and point clouds. The recent method PREDATOR finds overlapping regions by GNN (if the FCGF descriptors are consistent). However, an old implementation of RANSAC for robust Similarity transformation estimation on top of these correspondences is employed. Moreover, we observed that the performance of TEASER++ would be improved if the point cloud size, i.e., the number of correspondences between two point clouds, is small. This may be achieved by initialization of the TEASER++ by correspondences provided by PREDATOR. The same question remains open for the R-ICP, where better initialization, i.e., the selection of overlapping points, plays a crucial role.

To summarize, we investigated the topic of camera pose fine-tuning by captured depthmaps. Composed a new challenging dataset and evaluated a list of recent dense point cloud alignment methods on it. The GUI for easy follow-up of the development was published. The interesting observations are described and highlighted. In the end, we proposed several ideas for future research and improvement.





## Bibliography

1. NATIONAL COORDINATION OFFICE FOR SPACE-BASED POSITIONING, Navigation; TIMING. *The Global Positioning System* [online]. 2021 [visited on 2022-02-12]. Available from: <https://www.gps.gov/>.
2. AGENCY, European Space. *Galileo* [online]. 2022 [visited on 2022-02-12]. Available from: <https://www.esa.int/Applications/Navigation/Galileo>.
3. MARCEL, Jason. *Bluetooth Technology Is Getting Precise With Positioning Systems* [online]. 2019 [visited on 2022-03-10]. Available from: <https://www.bluetooth.com/blog/bluetooth-positioning-systems/>.
4. POURHOMAYOUN, Mohammad; FOWLER, Mark. Improving WLAN-based indoor mobile positioning using sparsity. In: *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*. 2012, pp. 1393–1396. Available from DOI: 10.1109/ACSSC.2012.6489254.
5. SARLIN, Paul-Edouard; CADENA, Cesar; SIEGWART, Roland; DYM-CZYK, Marcin. *From Coarse to Fine: Robust Hierarchical Localization at Large Scale*. 2019. Available from arXiv: 1812.03506 [cs.CV].
6. ZHANG, Juyong; YAO, Yuxin; DENG, Bailin. Fast and Robust Iterative Closest Point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021, pp. 1–1. ISSN 1939-3539. Available from DOI: 10.1109/tpami.2021.3054619.
7. FENG, Qiaojun; ATANASOV, Nikolay. Fully Convolutional Geometric Features for Category-level Object Alignment. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020. Available from DOI: 10.1109/iros45743.2020.9341550.
8. CANTZLER, H. Random Sample Consensus (RANSAC). 1981.
9. YANG, Heng; SHI, Jingnan; CARLONE, Luca. *TEASER: Fast and Certifiable Point Cloud Registration*. 2020. Available from arXiv: 2001.07715 [cs.R0].



20. HAMMARSTRAND, Lars; KAHL, Fredrik; MADDERN, Will; PAJDLA, Tomas; POLLEFEYS, Marc; SATTLER, Torsten; SIVIC, Josef; STENBORG, Erik; TOFT, Carl; TORII, Akihiko. *Workshop on Long-Term Visual Localization under Changing Conditions* [online]. 2020 [visited on 2022-05-11]. Available from: <https://www.visuallocalization.net/workshop/eccv/2020/>.
21. NATIONAL COORDINATION OFFICE FOR SPACE-BASED POSITIONING, Navigation; TIMING. *GPS Accuracy* [online]. 2022 [visited on 2022-03-10]. Available from: <https://www.gps.gov/systems/gps/performance/accuracy/>.
22. AGENCY, European Space. *Galileo* [online]. 2021 [visited on 2022-03-10]. Available from: [https://gssc.esa.int/navipedia/index.php/Galileo\\_Performances](https://gssc.esa.int/navipedia/index.php/Galileo_Performances).
23. GUSTAFSSON, F.; GUNNARSSON, F. Mobile positioning using wireless networks: possibilities and fundamental limitations based on available wireless network measurements. *IEEE Signal Processing Magazine*. 2005, vol. 22, no. 4, pp. 41–53. Available from DOI: 10.1109/MSP.2005.1458284.
24. CHEN, Chen; CHEN, Yan; LAI, Hung-Quoc; HAN, Yi; LIU, K.J. Ray. High accuracy indoor localization: A WiFi-based approach. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 6245–6249. Available from DOI: 10.1109/ICASSP.2016.7472878.
25. FACHRI, M; KHUMAIDI, A. Positioning Accuracy of Commercial Bluetooth Low Energy Beacon. *IOP Conference Series: Materials Science and Engineering*. 2019, vol. 662, no. 5, p. 052018. Available from DOI: 10.1088/1757-899x/662/5/052018.
26. SICILIANO, B.; KHATIB, O. *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008. Springer Handbook of Robotics. ISBN 9783540239574. Available also from: <https://books.google.cz/books?id=Xpgi5gSuBxsC>.
27. YUAN, Qilong; CHEN, I-Ming. Localization and Velocity Tracking of Human via 3 IMU Sensors. *Sensors and Actuators A: Physical*. 2014, vol. 212. Available from DOI: 10.1016/j.sna.2014.03.004.
28. LÓPEZ-CERÓN, Alberto; CAÑAS, José María. Accuracy analysis of marker-based 3D visual localization. *Actas de las XXXVII Jornadas de Automática 7, 8 y 9 de septiembre de 2016, Madrid*. 2022.
29. MERRIAUX, Pierre; DUPUIS, Yohan; BOUTTEAU, Rémi; VASSEUR, Pascal; SAVATIER, Xavier. A Study of Vicon System Positioning Performance. *Sensors*. 2017, vol. 17, p. 1591. Available from DOI: 10.3390/s17071591.
30. CORPORATION, Microsoft. *HoloLens Research Mode* [online]. 2017 [visited on 2022-02-22]. Available from: <https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/research-mode>.

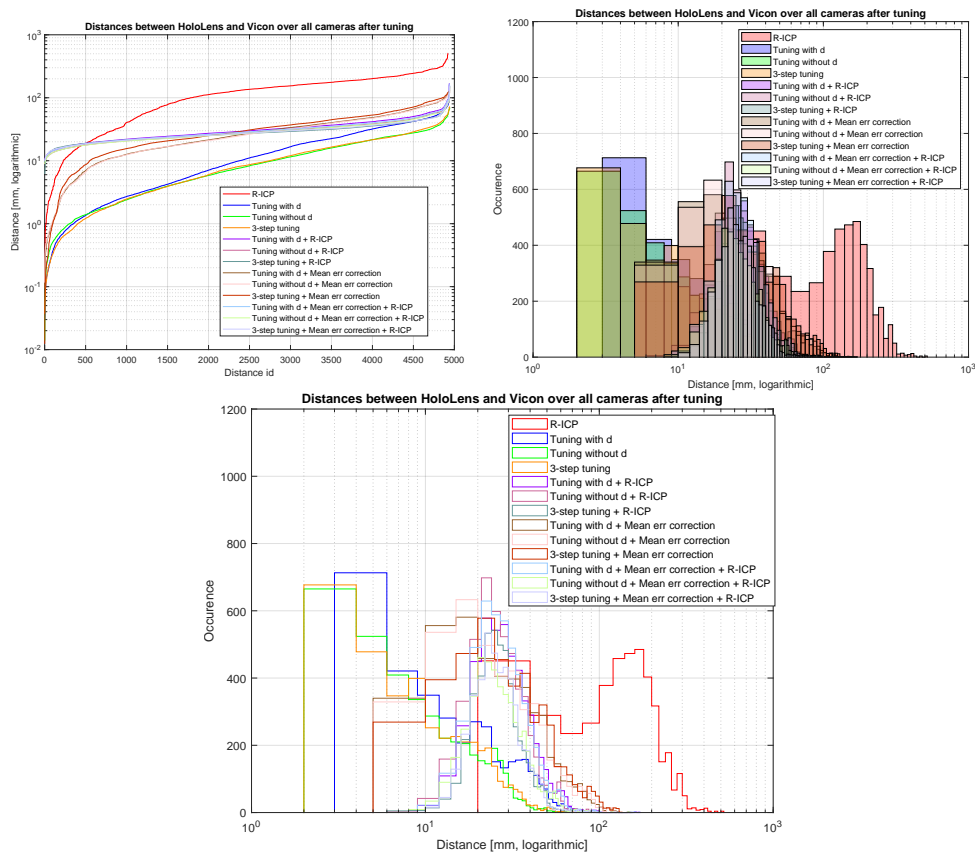
31. HÜBNER, Patrick; CLINTWORTH, Kate; LIU, Qingyi; WEINMANN, Martin; WURSTHORN, Sven. Evaluation of HoloLens Tracking and Depth Sensing for Indoor Mapping Applications. *Sensors (Basel, Switzerland)*. 2020, vol. 20.
32. CORPORATION, Microsoft. *HoloLens environment considerations* [online]. 2017 [visited on 2022-02-22]. Available from: <https://docs.microsoft.com/en-us/hololens/hololens-environment-considerations>.
33. HÜBNER, Patrick; WEINMANN, Martin; WURSTHORN, Sven. MARKER-BASED LOCALIZATION OF THE MICROSOFT HOLOLENS IN BUILDING MODELS. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. 2018, vol. XLII-1, pp. 195–202. Available from DOI: 10.5194/isprs-archives-XLII-1-195-2018.
34. HERBERS, Patrick; KÖNIG, Markus. Indoor Localization for Augmented Reality Devices Using BIM, Point Clouds, and Template Matching. *Applied Sciences*. 2019, vol. 9, p. 4260. Available from DOI: 10.3390/app9204260.
35. ROSS, Amy. Procrustes Analysis. 2005. Available from DOI: 10.1.1.119.2686.
36. BARATH, Daniel; MATAS, Jiri; NOSKOVA, Jana. MAGSAC: marginalizing sample consensus. In: *Conference on Computer Vision and Pattern Recognition*. 2019.
37. MYATT, D.; TORR, Philip; BISHOP, John; CRADDOCK, R.; NASUTO, Slawomir. NAPSAC: HIGH NOISE, HIGH DIMENSIONAL MODEL PARAMETERISATION - IT’S IN THE BAG. 2002.
38. BARATH, Daniel; IVASHECHKIN, Maksym; MATAS, Jiri. Progressive NAPSAC: sampling from gradually growing neighborhoods. *CoRR*. 2019, vol. abs/1906.02295. Available from arXiv: 1906.02295.
39. CHEN, Yang; MEDIONI, Gérard. Object Modeling by Registration of Multiple Range Images. *Image Vision Comput*. 1992, vol. 10, pp. 145–155. Available from DOI: 10.1109/ROBOT.1991.132043.
40. CORPORATION, Microsoft. *HoloLensForCV* [online]. 2017 [visited on 2022-02-22]. Available from: <https://github.com/microsoft/HoloLensForCV>.
41. ALICEVISION. *Meshroom: A 3D reconstruction software*. 2018. Available also from: <https://github.com/alicevision/meshroom>.
42. ALICEVISION. *Meshroom: A 3D reconstruction software*. 2018. Available also from: <https://meshroom-manual.readthedocs.io/en/latest/feature-documentation/core/nodes.html>.
43. SCHÖNBERGER, Johannes Lutz; FRAHM, Jan-Michael. *Camera Models*. 2016. Available also from: <https://colmap.github.io/cameras.html>.

44. SHI, Jingnan. The cloudPoint size more than 5000, RobustRegistration-Solver will be collapse. In: 2021. Available also from: <https://github.com/MIT-SPARK/TEASER-plusplus/issues/118>.
45. TORR, P.H.S.; ZISSERMAN, A. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*. 2000, vol. 78, no. 1, pp. 138–156. ISSN 1077-3142. Available from DOI: <https://doi.org/10.1006/cviu.1999.0832>.
46. HUANG, Shengyu; GOJCIC, Zan; USVYATSOV, Mikhail; WIESER, Andreas; SCHINDLER, Konrad. Predator: Registration of 3D Point Clouds With Low Overlap. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 4267–4276. Available also from: <https://github.com/overlappredator/OverlapPredator5>.
47. HUANG, Shengyu. RANSAC used? In: 2022. Available also from: <https://github.com/prs-eth/OverlapPredator/issues/39>.

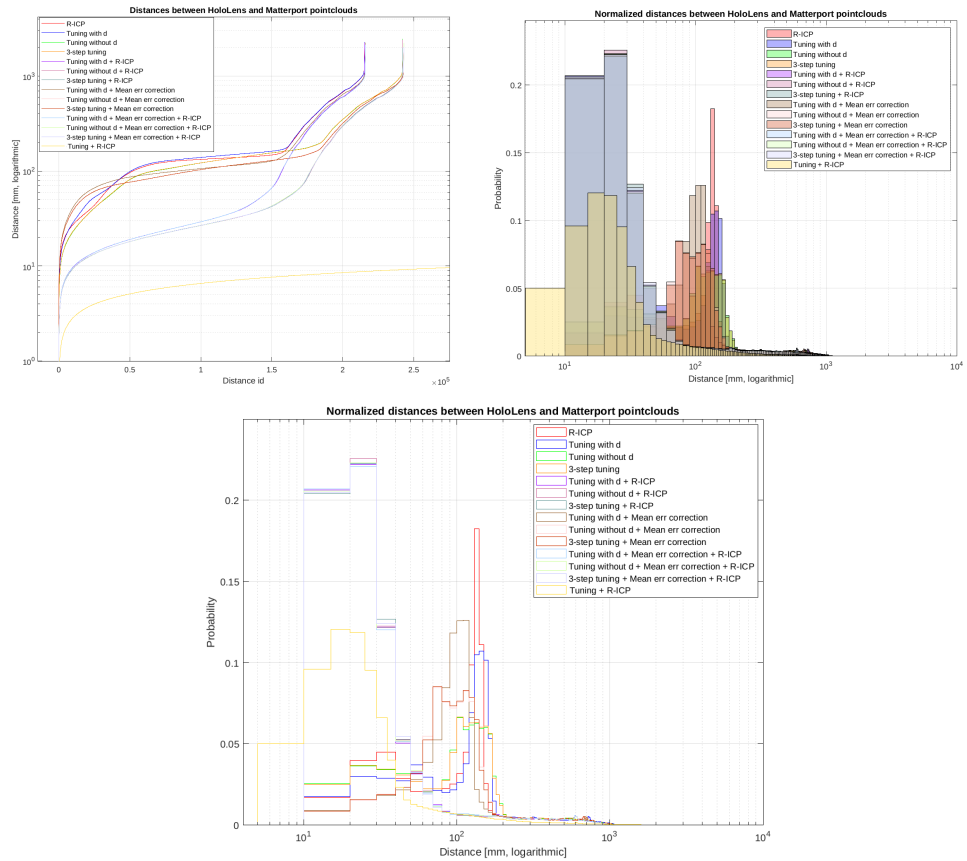


# Appendix A

## Dataset acquisition additional materials



**Figure A.1:** Distances between all aligned HoloLens camera centres and Vicon marker centres.



**Figure A.2:** Distances between aligned HoloLens point clouds and Matterport point cloud. (Note that the number of points in point clouds differ as point clouds are being cut accordingly to alignment, thus leading to normalizing the histograms (the value of the bin equals to occurrence/sum(occurrences)))






## Appendix B

### Full evaluation results

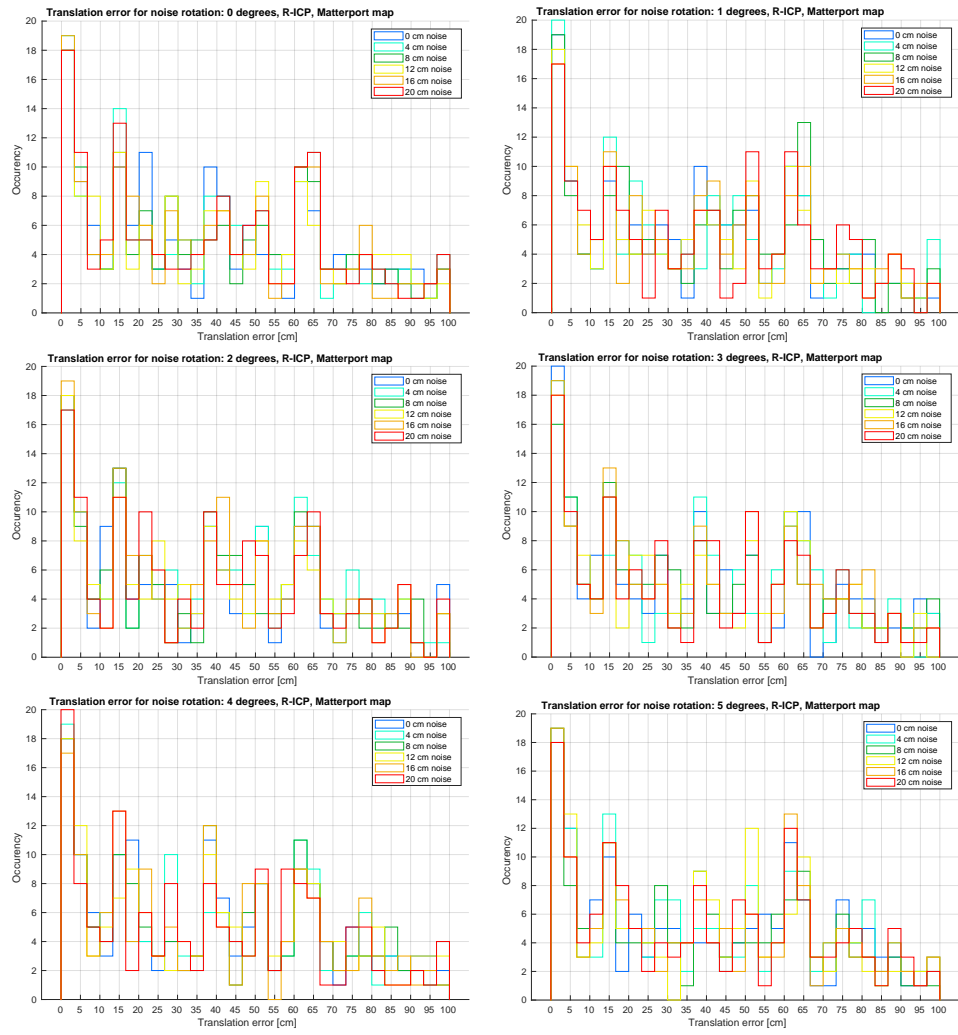
This section contains full evaluation graphs for all methods evaluated in this thesis.



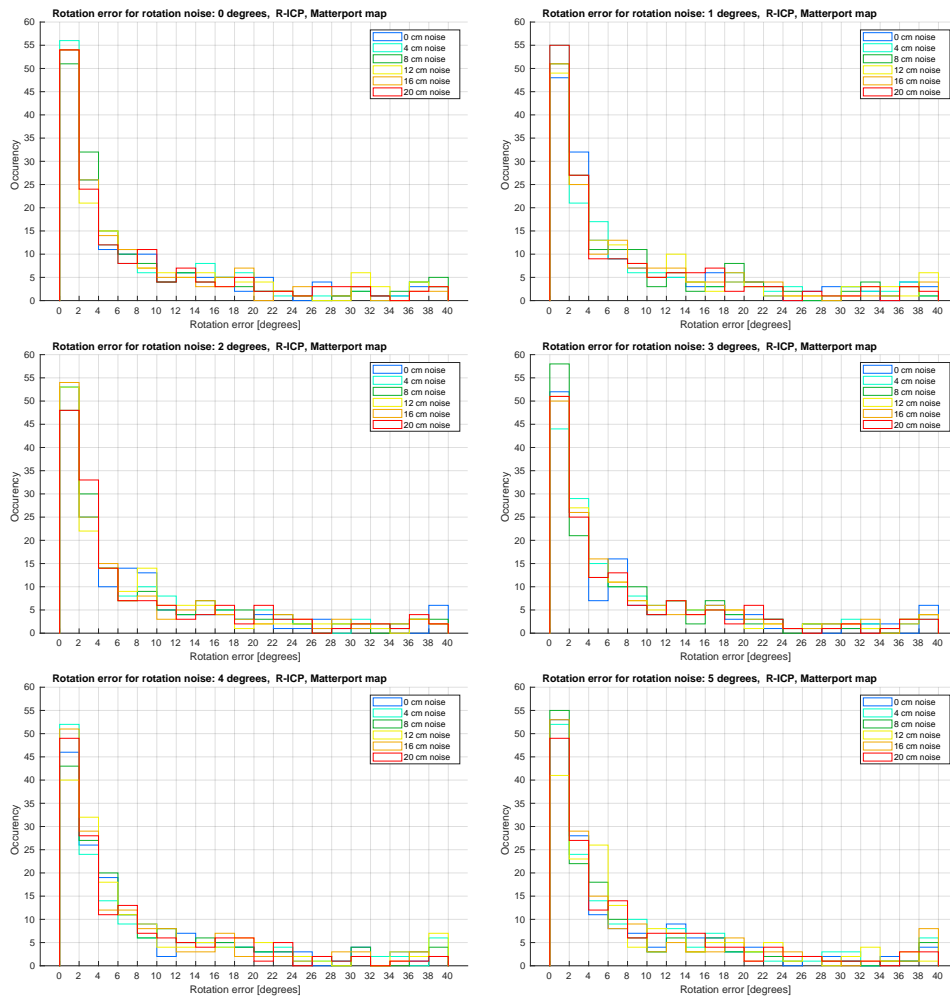
#### B.1 R-ICP

This section focuses on full evaluation of R-ICP alignment.

B. Full evaluation results



**Figure B.1:** Translation error for R-ICP alignment over all translation and rotation noised point clouds on the Matterport map.



**Figure B.2:** Rotation error for R-ICP alignment over all translation and rotation noised point clouds on the Matterport map.

B. Full evaluation results

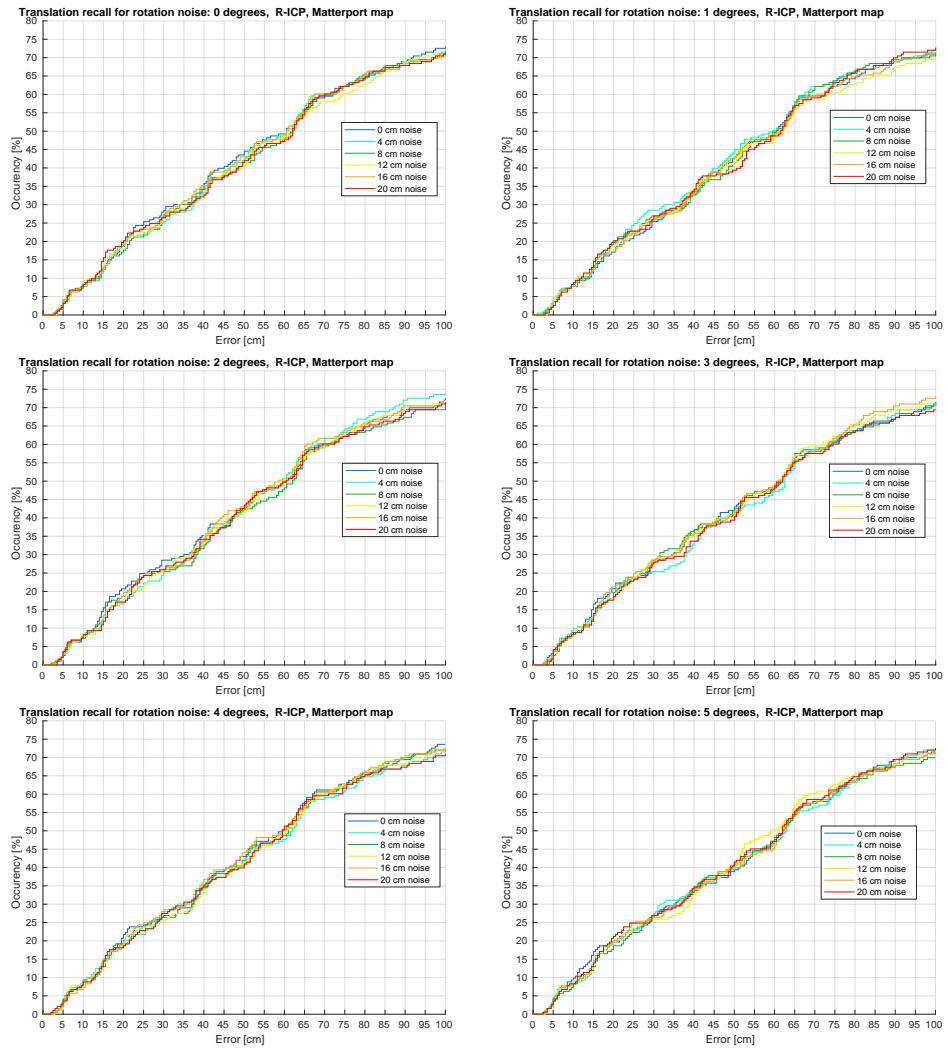
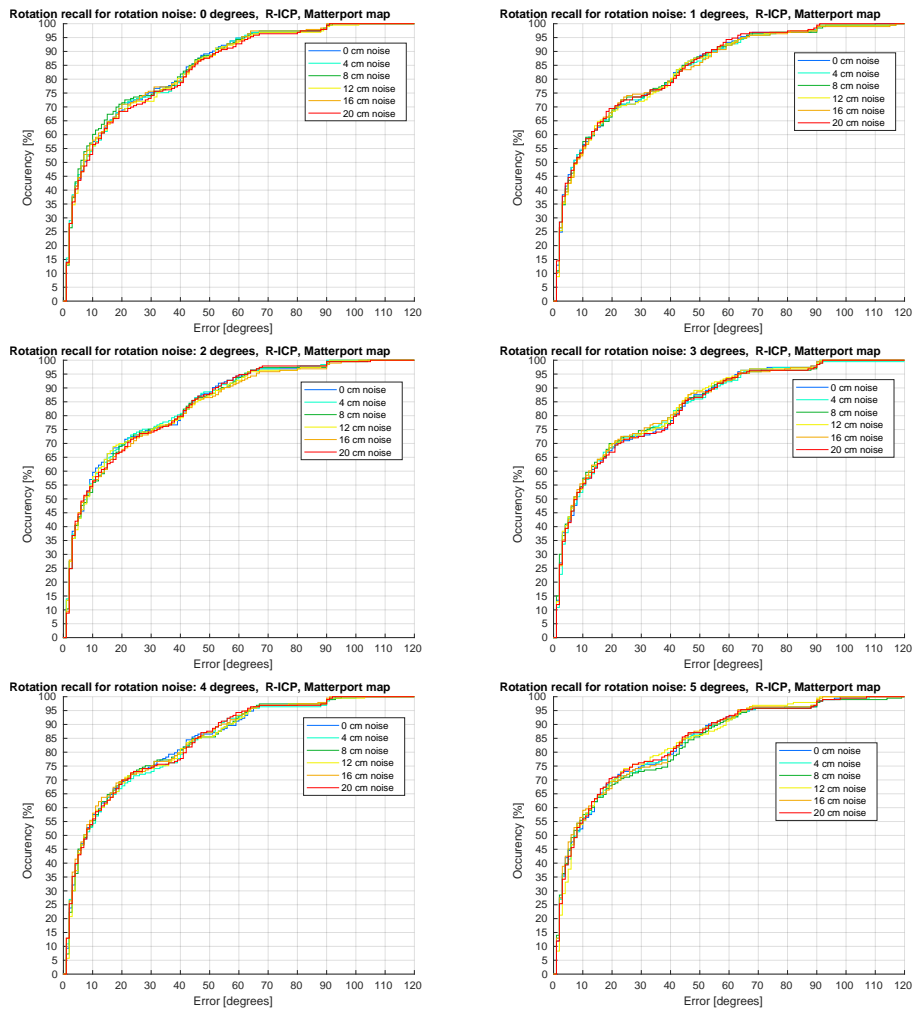
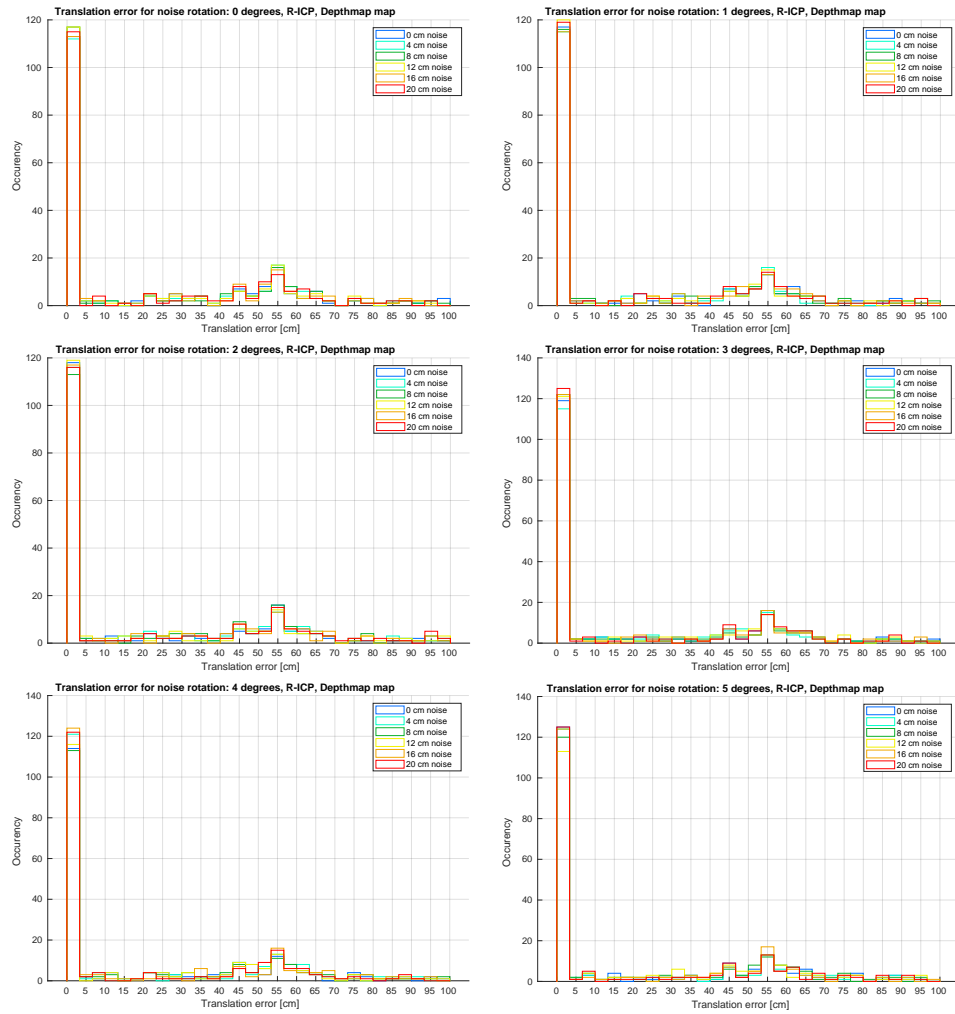


Figure B.3: Translation recall for R-ICP alignment over all translation and rotation noised point clouds on the Matterport map.

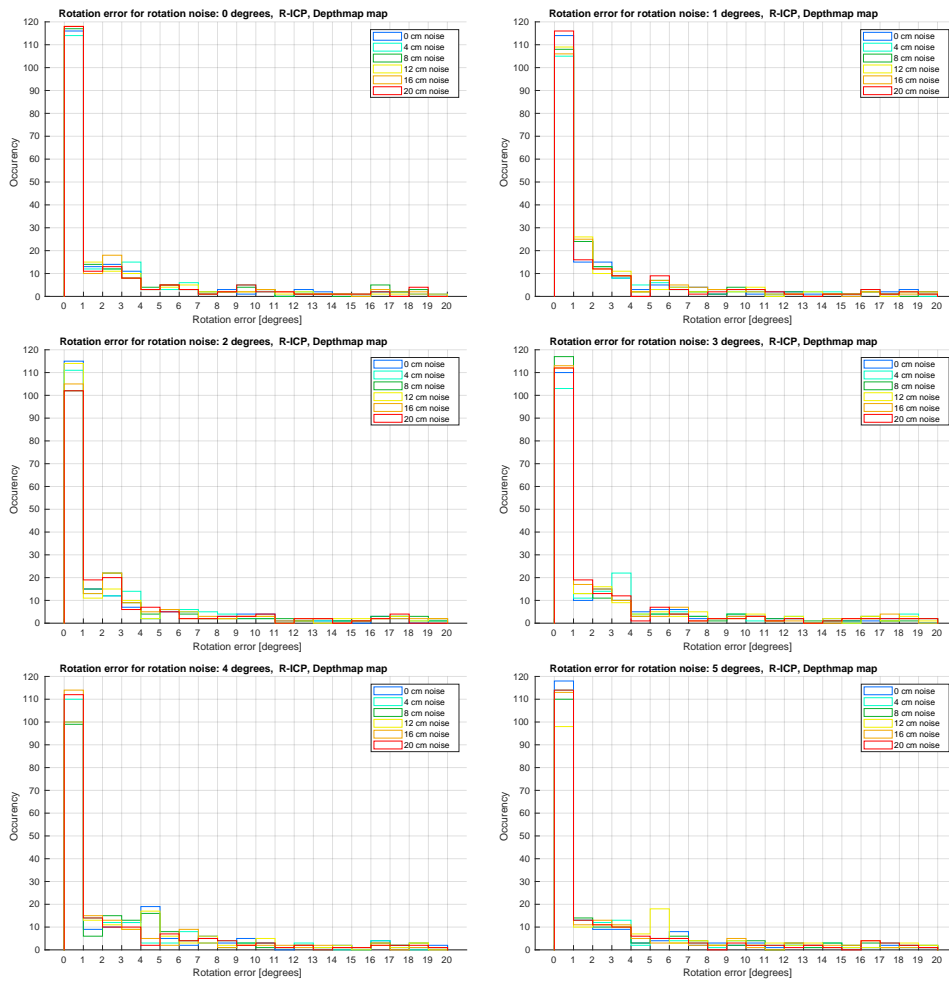


**Figure B.4:** Rotation recall for R-ICP alignment over all translation and rotation noised point clouds on the Matterport map.

B. Full evaluation results

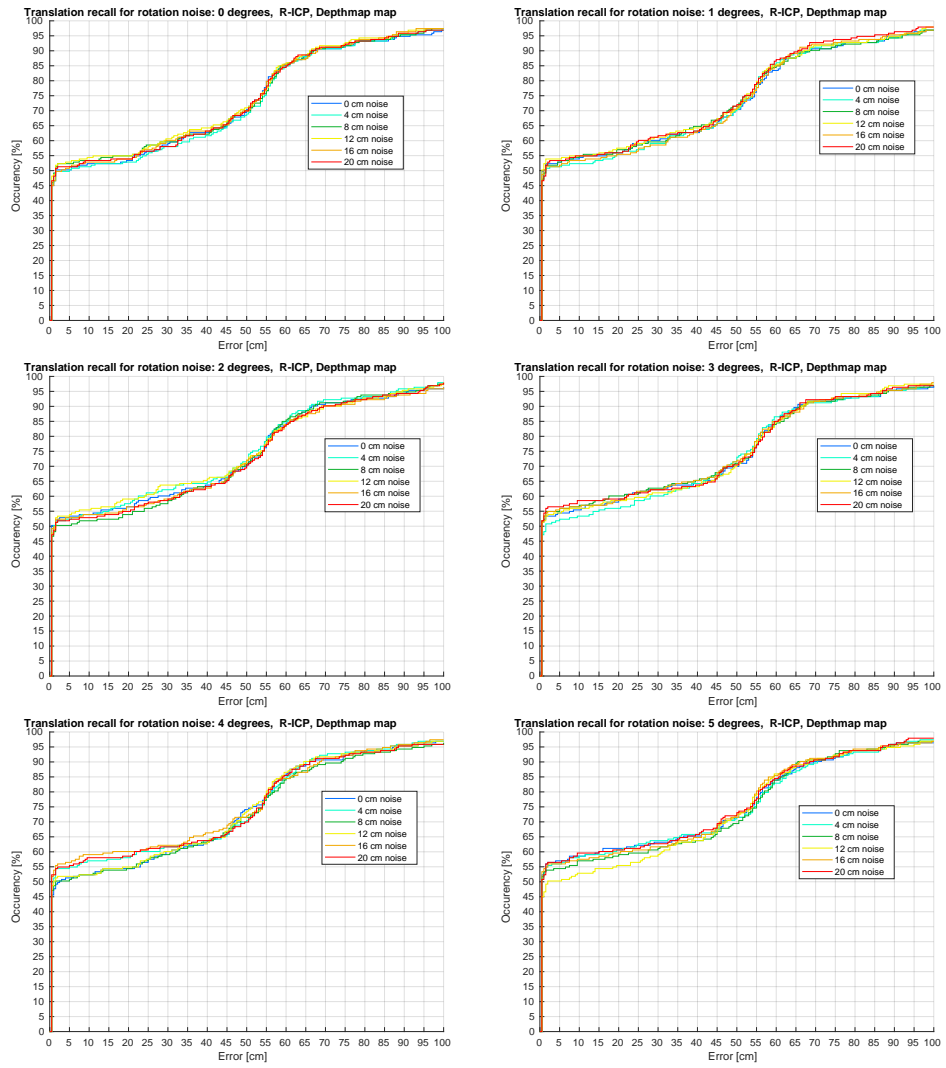


**Figure B.5:** Translation error for R-ICP alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.



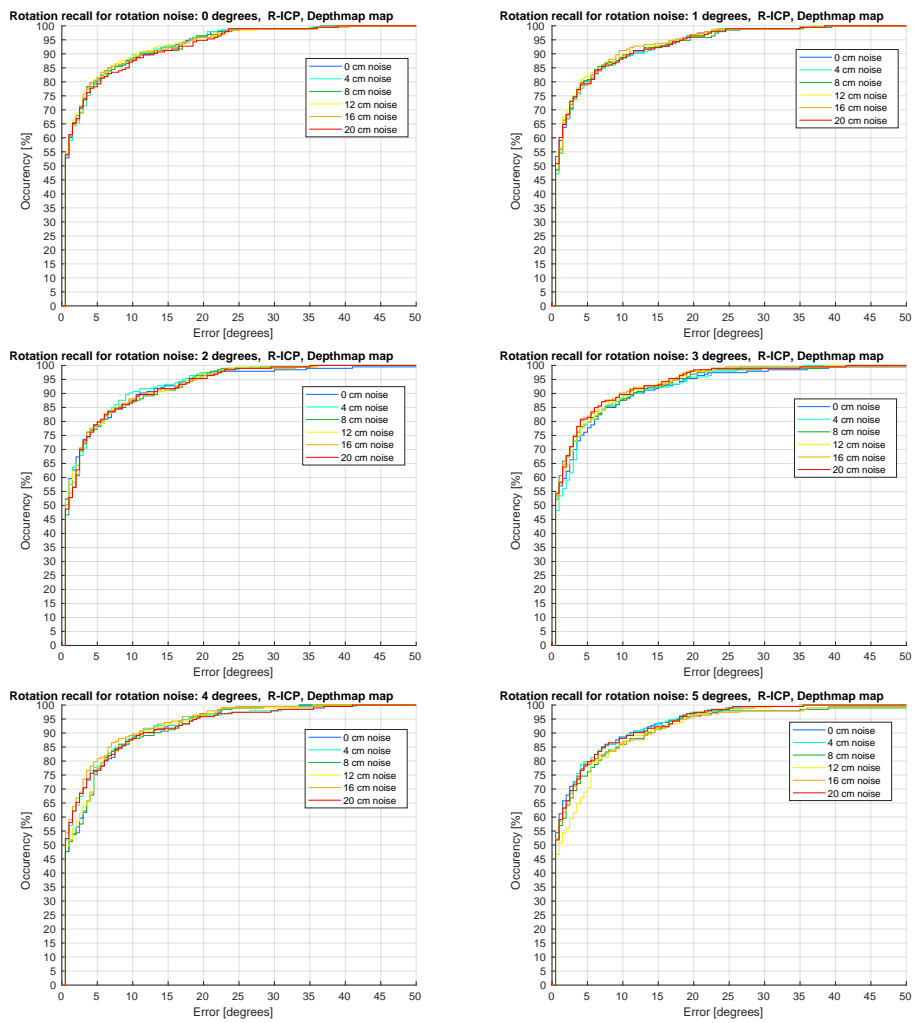
**Figure B.6:** Rotation error for R-ICP alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.

B. Full evaluation results



**Figure B.7:** Translation recall for R-ICP alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.

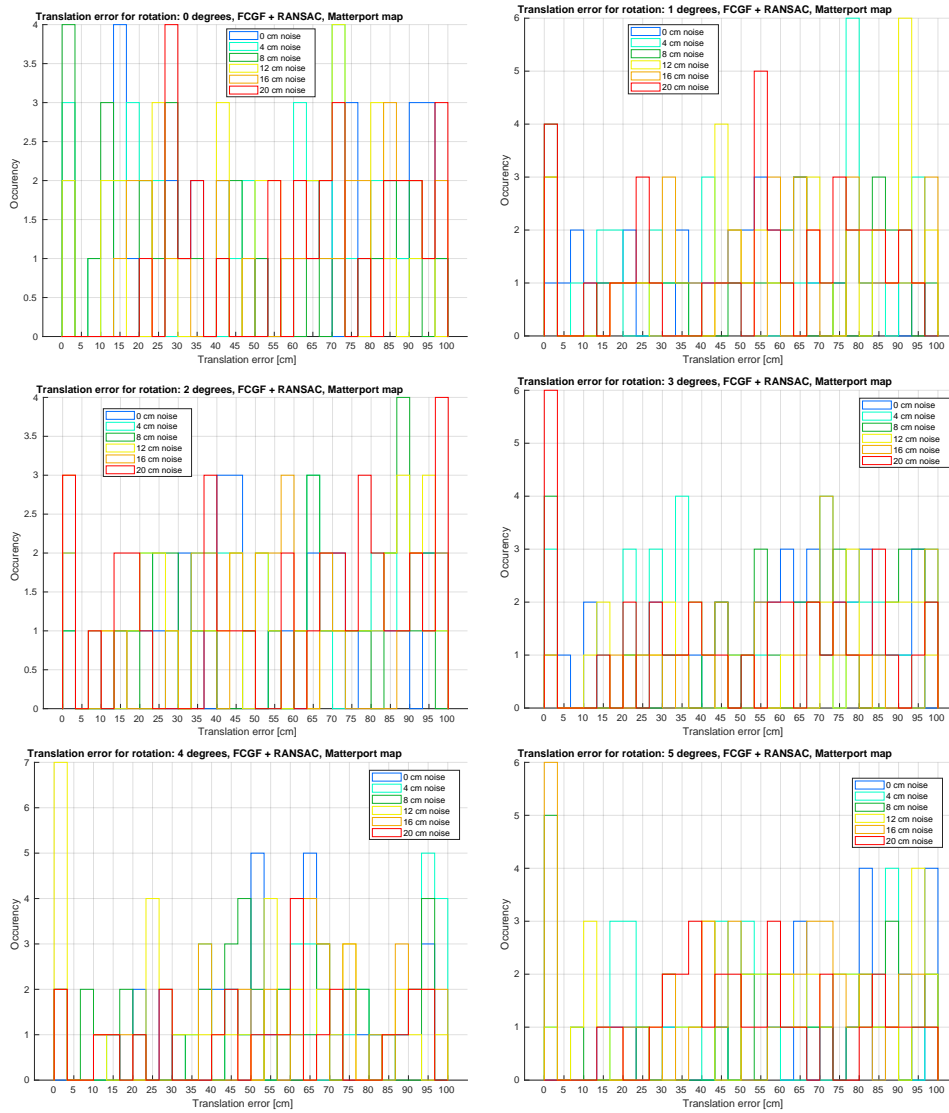




**Figure B.8:** Rotation recall for R-ICP alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.

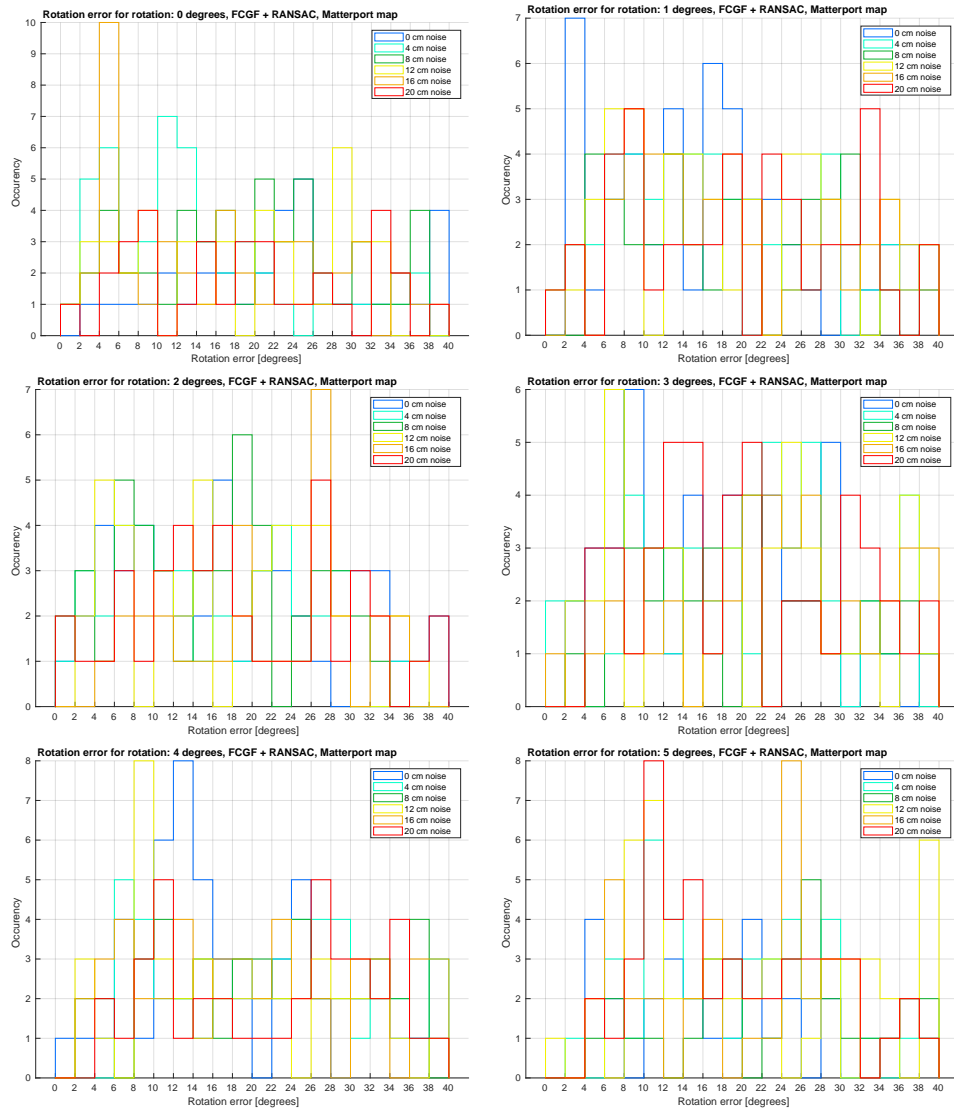
## ■ **B.2 FCGF + RANSAC**

This section contains full evaluation of FCGF + RANSAC alignment.

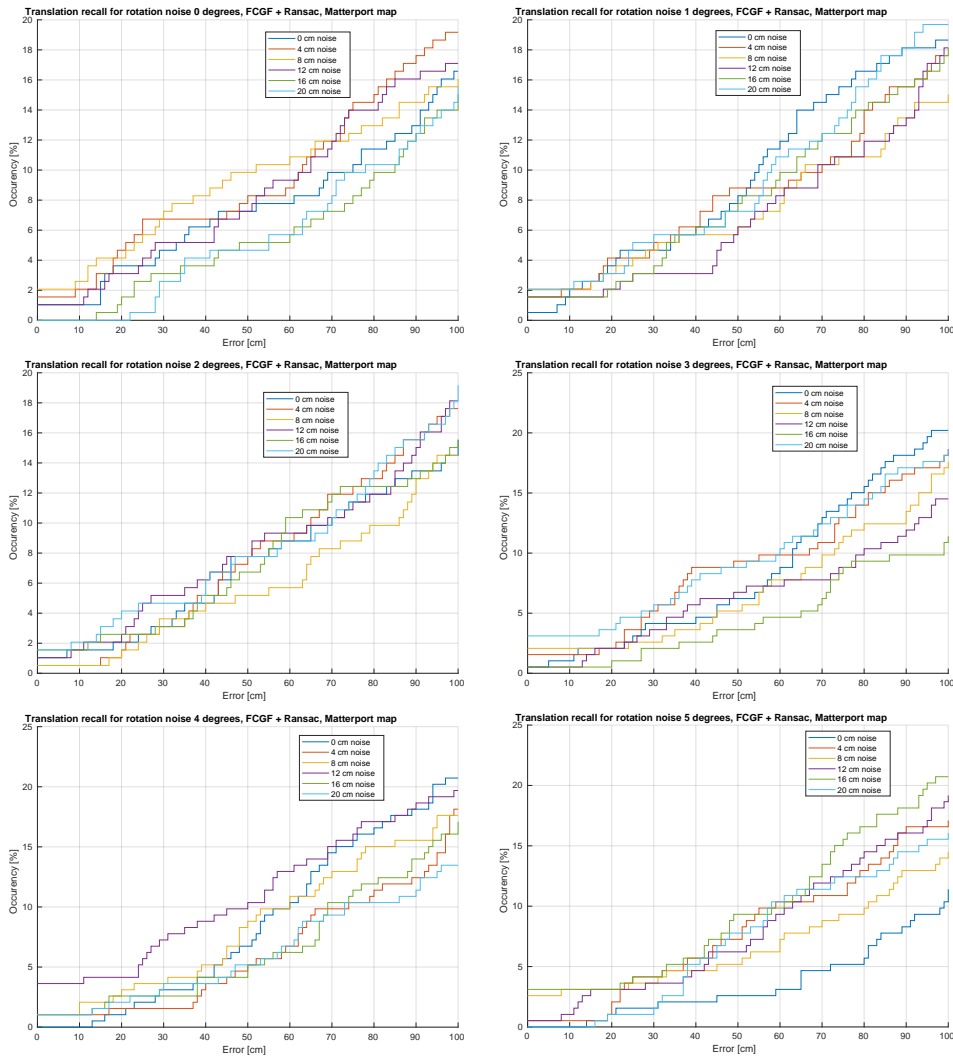


**Figure B.9:** Translation error for FCGF + RANSAC alignment over all translation and rotation noised point clouds on the Matterport map.

B. Full evaluation results

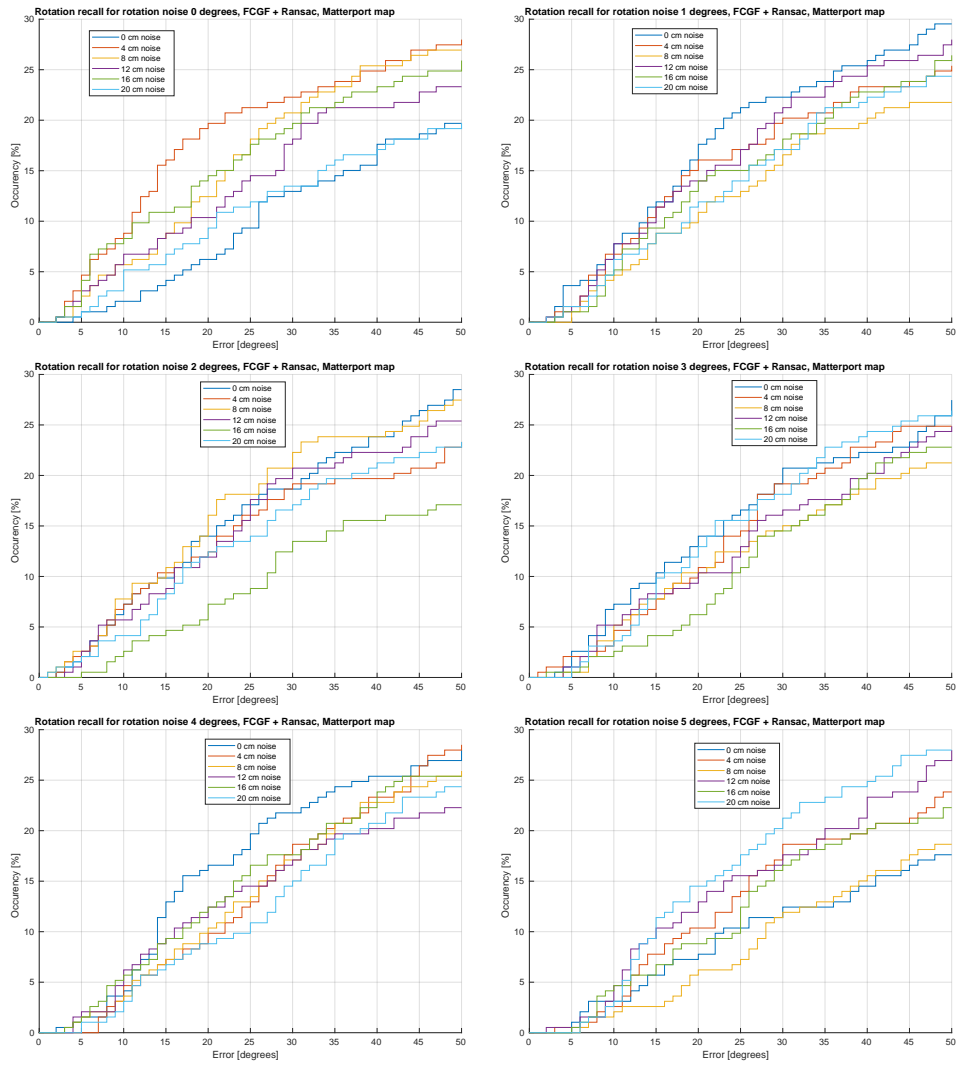


**Figure B.10:** Rotation error for FCGF + RANSAC alignment over all translation and rotation noised point clouds on the Matterport map.

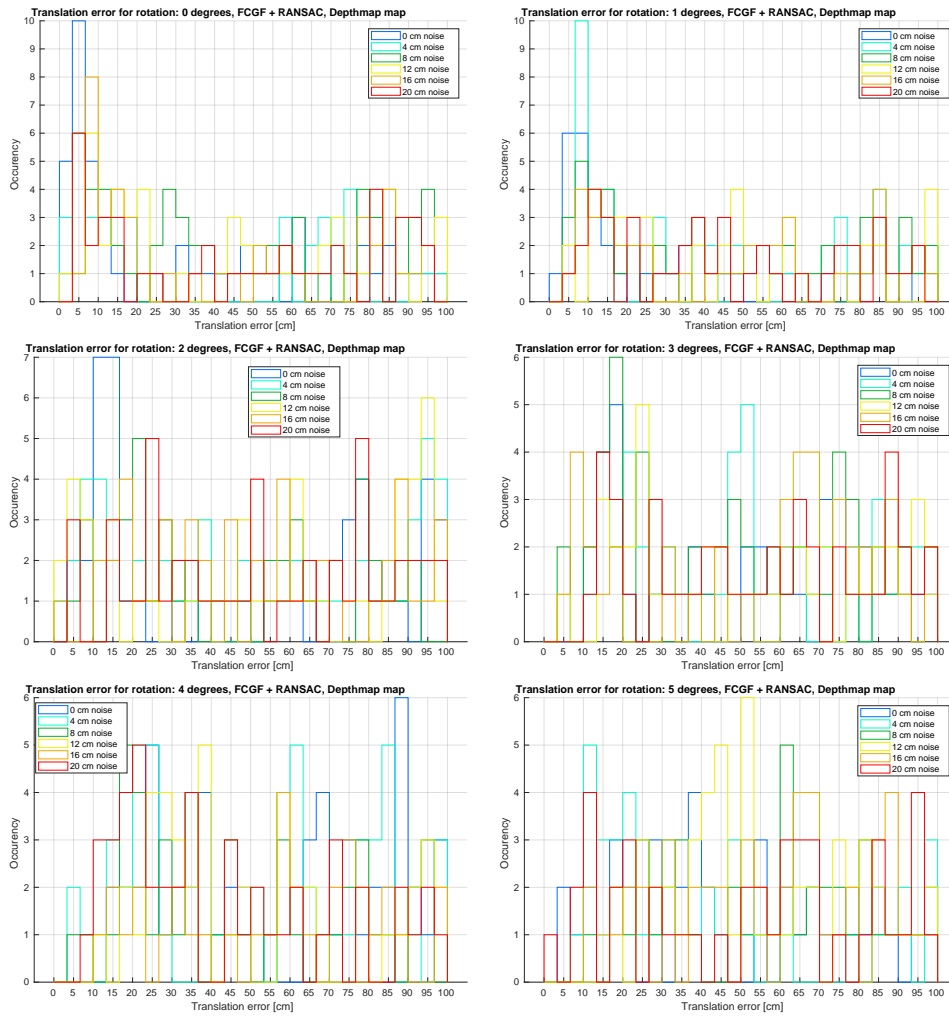


**Figure B.11:** Translation recall for FCGF + RANSAC alignment over all translation and rotation noised point clouds on the Matterport map.

## B. Full evaluation results

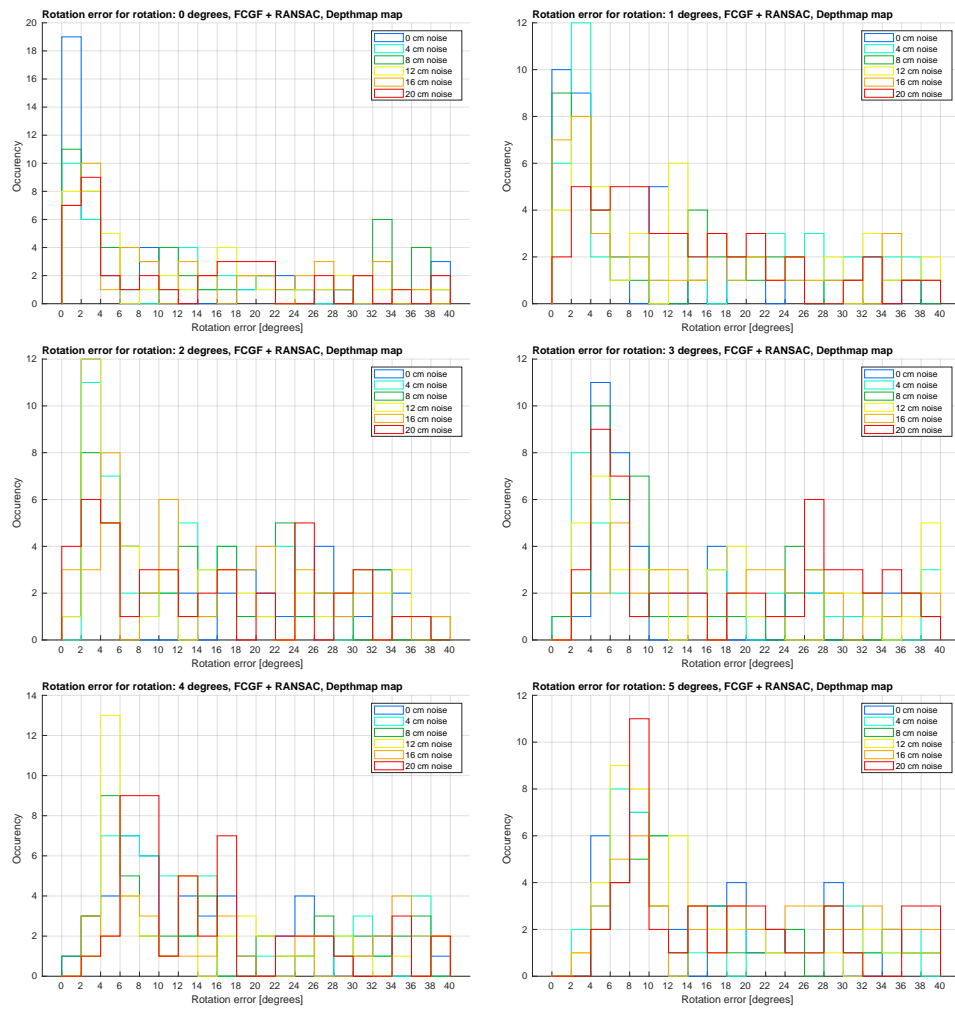


**Figure B.12:** Rotation recall for FCGF + RANSAC alignment over all translation and rotation noised point clouds on the Matterport map.



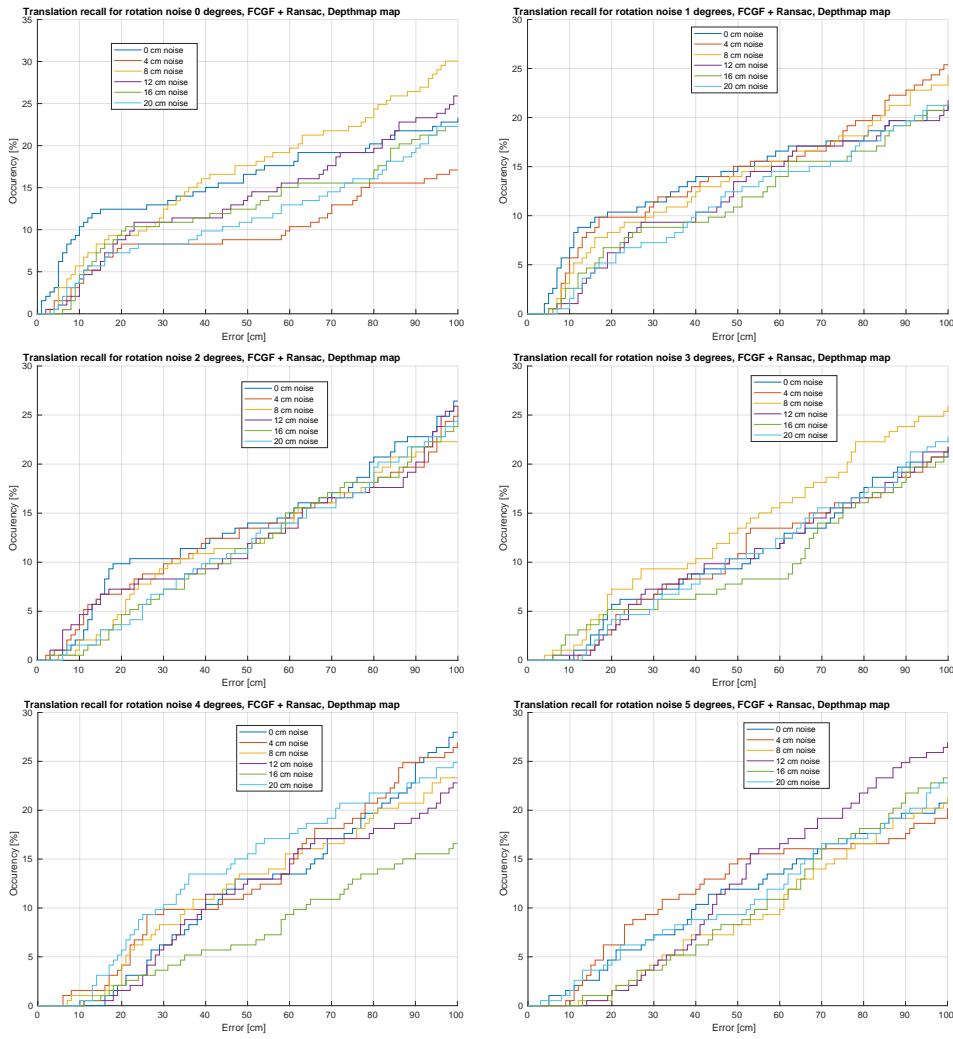
**Figure B.13:** Translation error for FCGF + RANSAC alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.

B. Full evaluation results



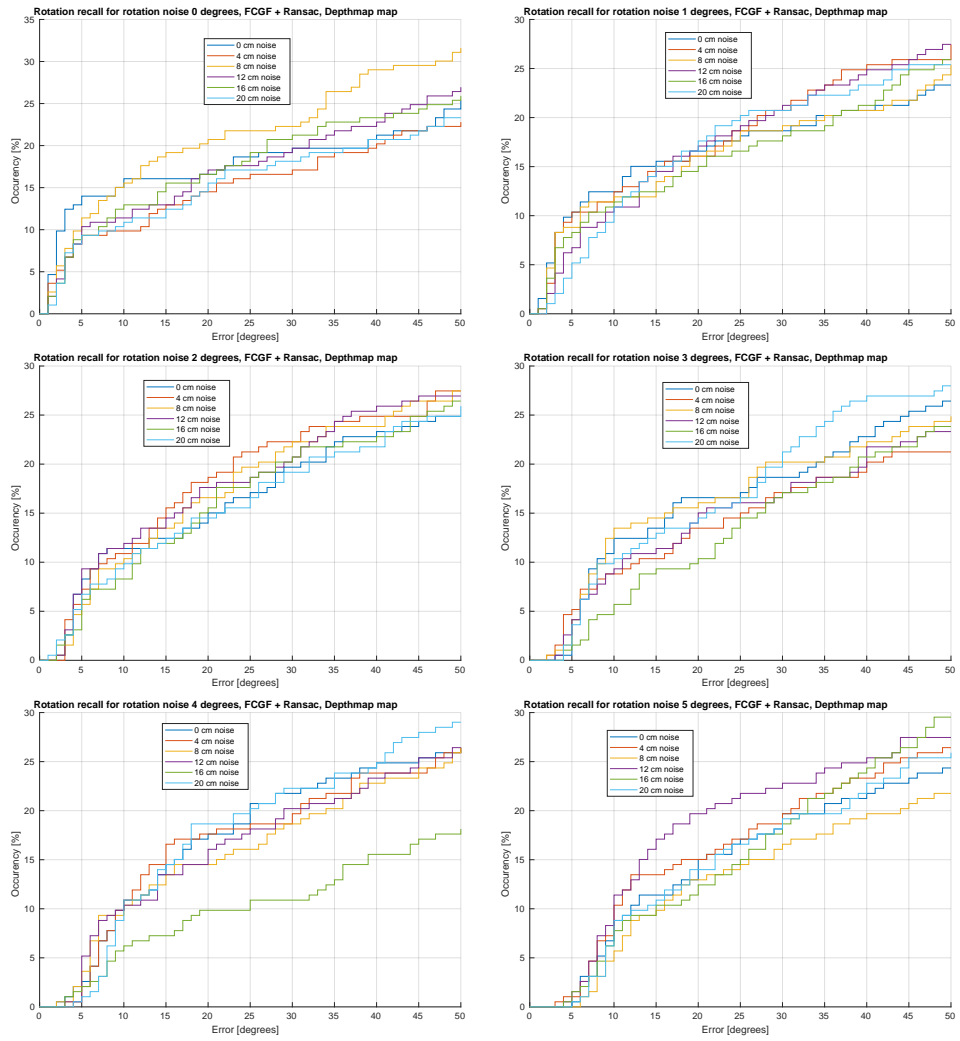
**Figure B.14:** Rotation error for FCGF + RANSAC alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.





**Figure B.15:** Translation recall for FCGF + RANSAC alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.

B. Full evaluation results

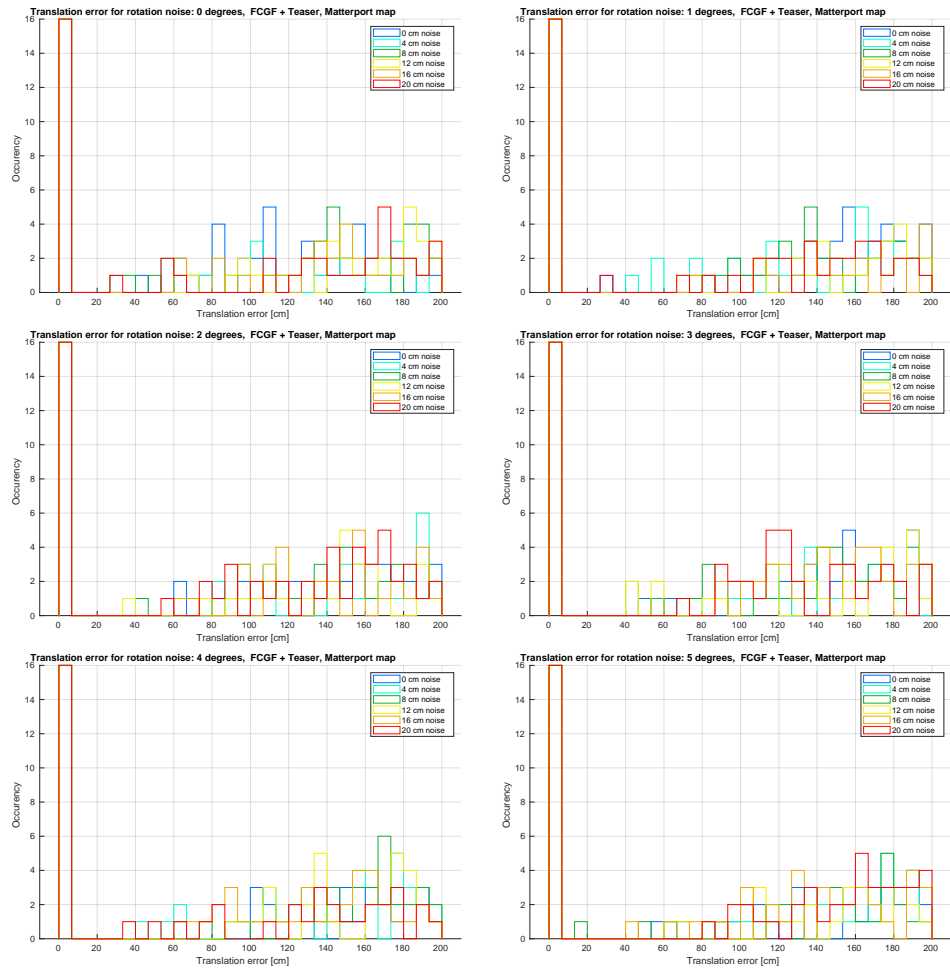


**Figure B.16:** Rotation recall for FCGF + RANSAC alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.

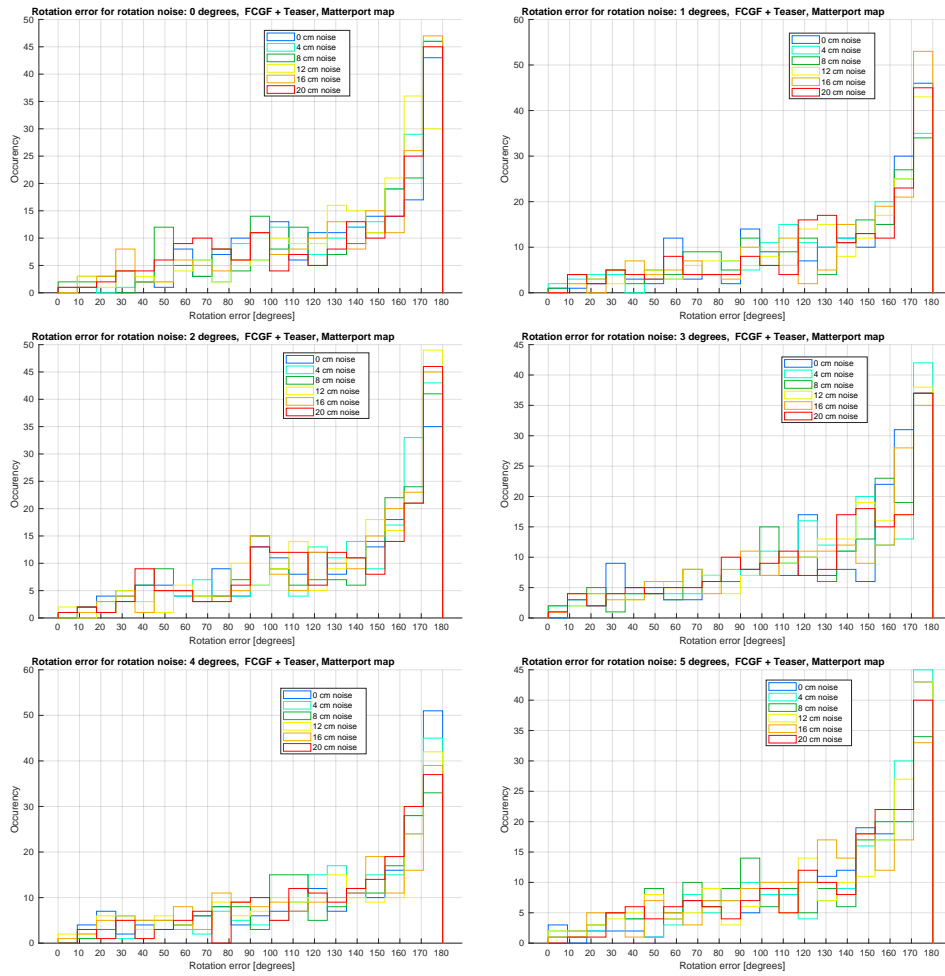
## ■ B.3 FCGF + TEASER++

This section contains full evaluation of FCGF + TEASER++ alignment using the mutually nearest feature descriptors only.

B. Full evaluation results

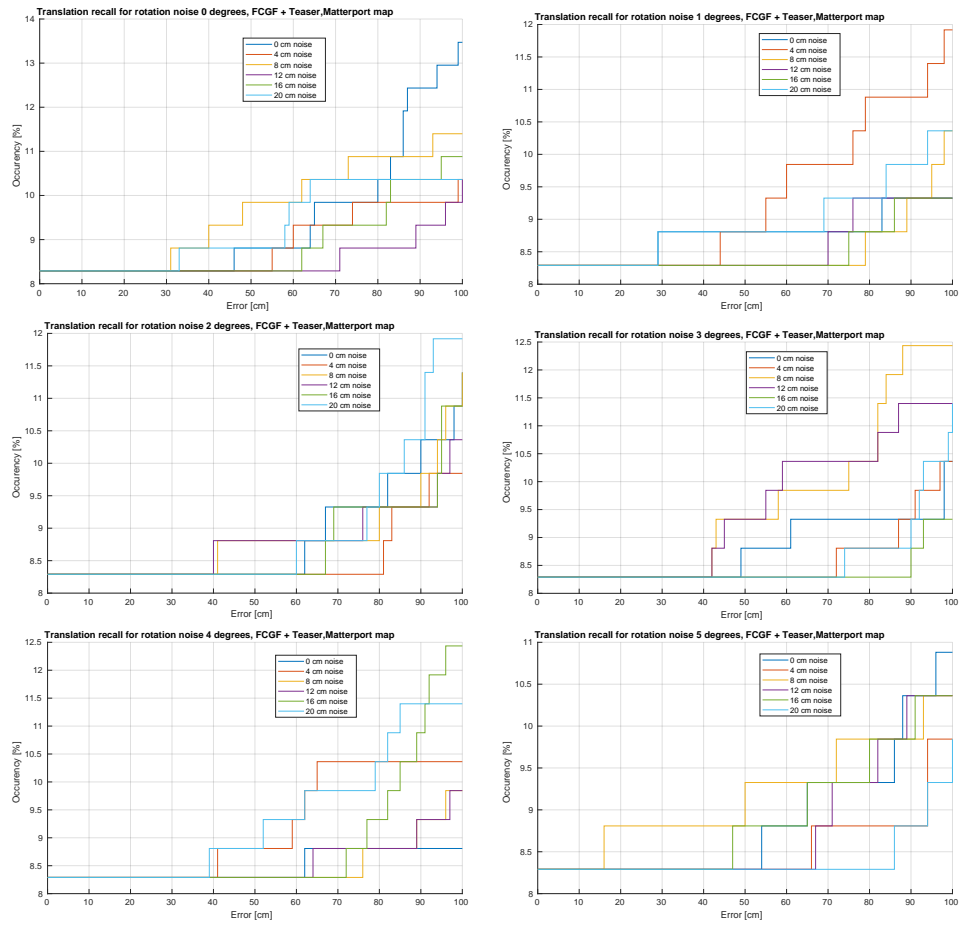


**Figure B.17:** Translation error for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on the Matterport map.

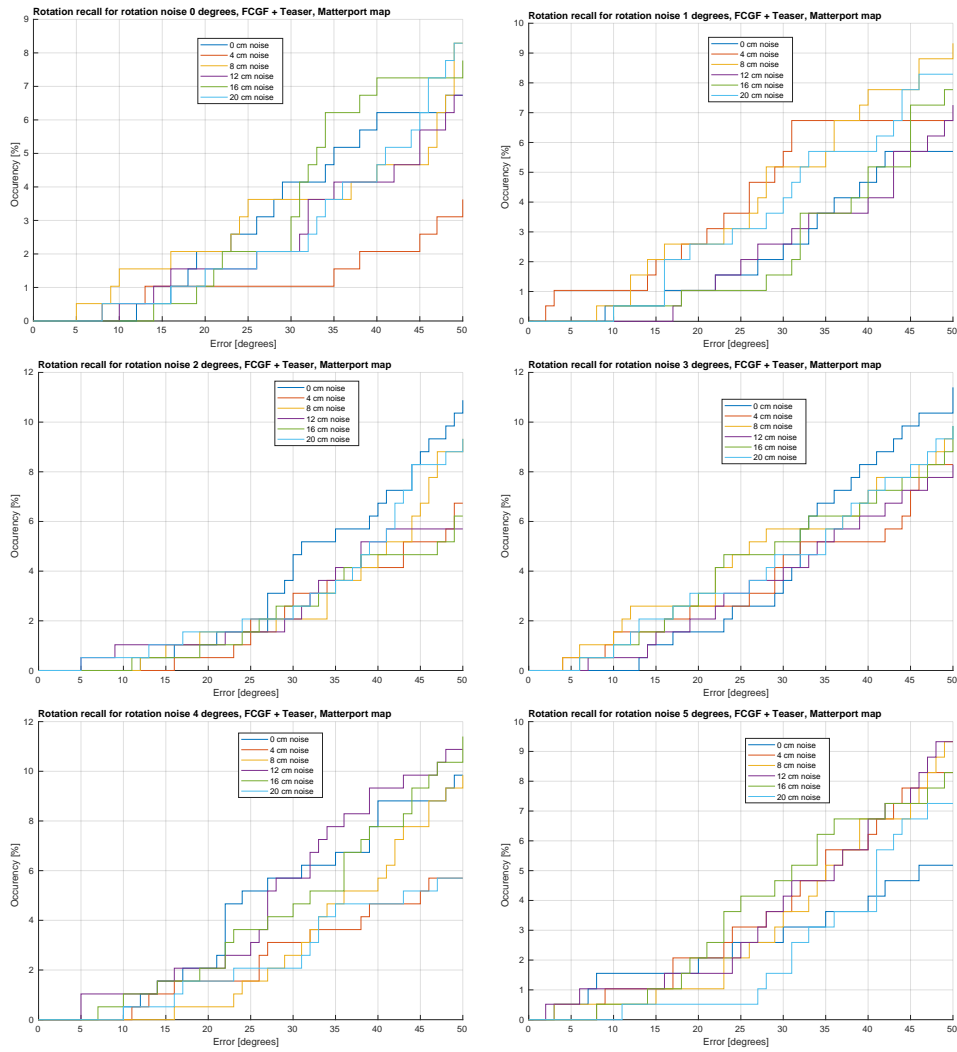


**Figure B.18:** Rotation error for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on the Matterport map.

B. Full evaluation results

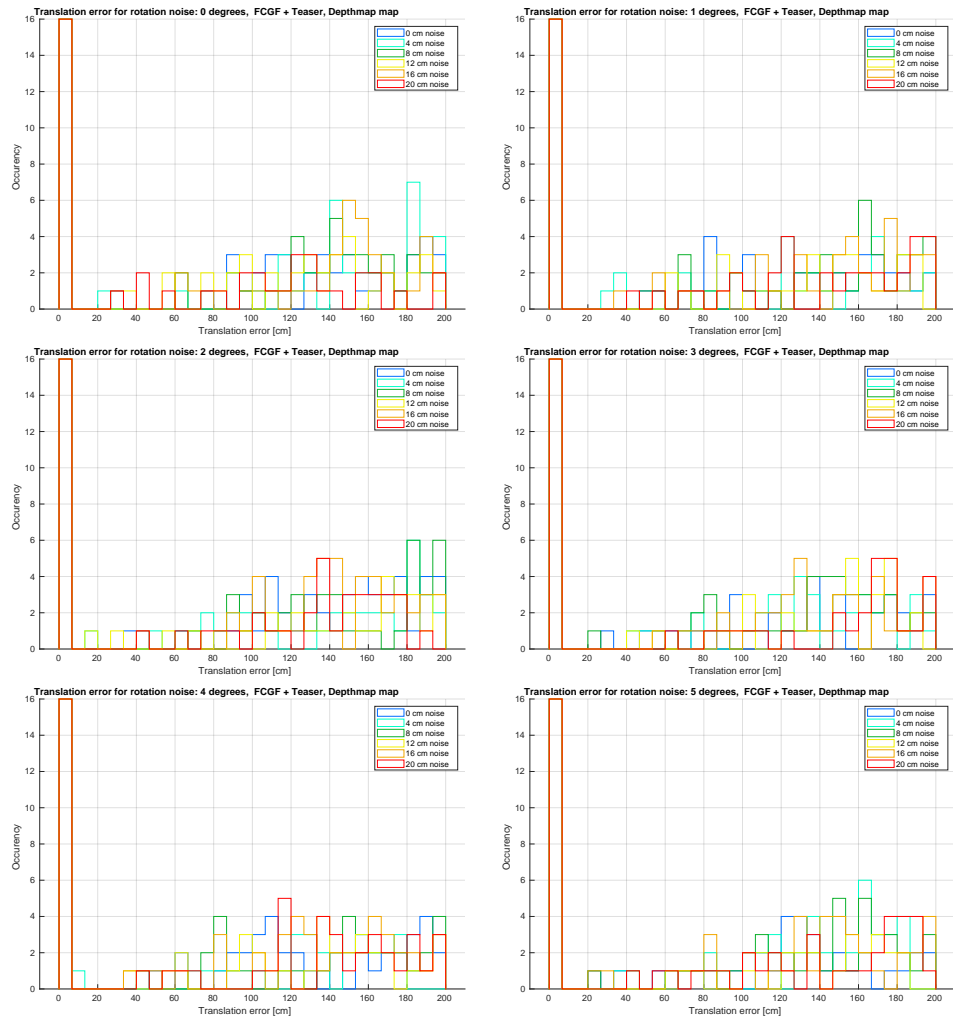


**Figure B.19:** Translation recall for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on the Matterport map.



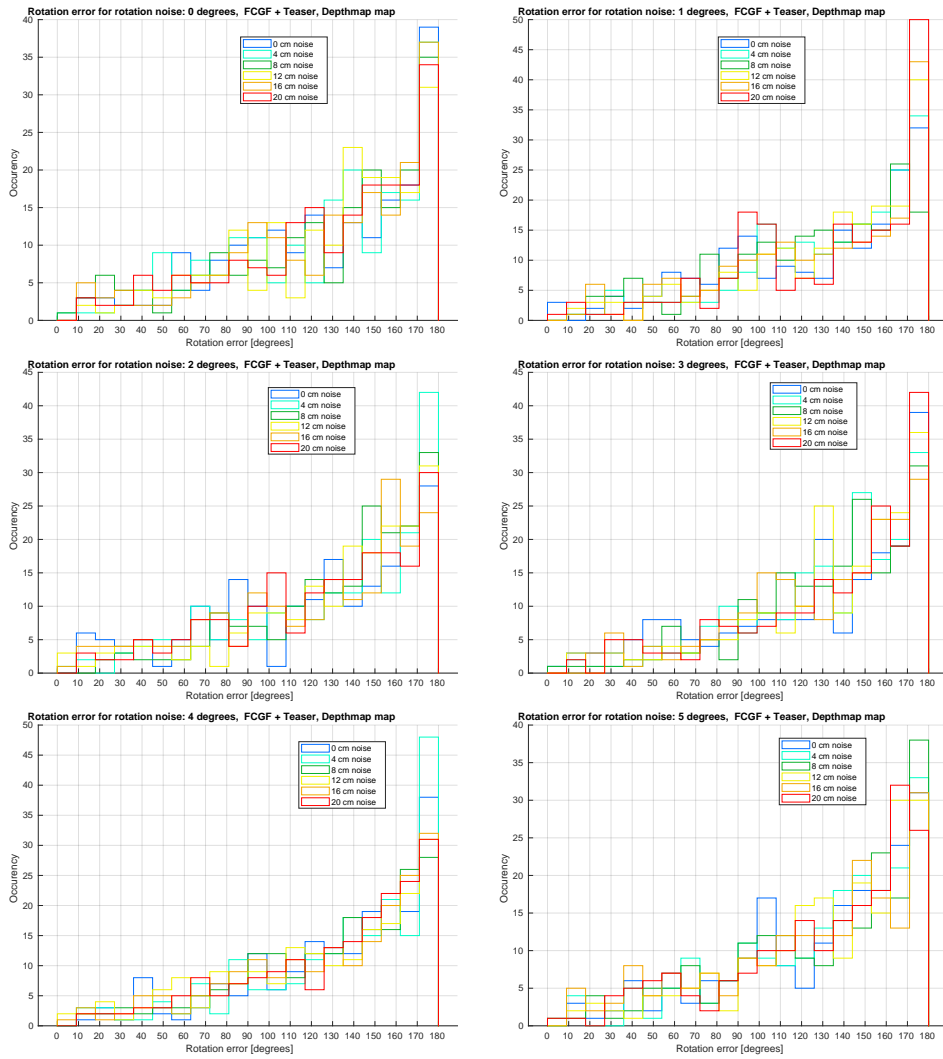
**Figure B.20:** Rotation recall for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on the Matterport map.

B. Full evaluation results



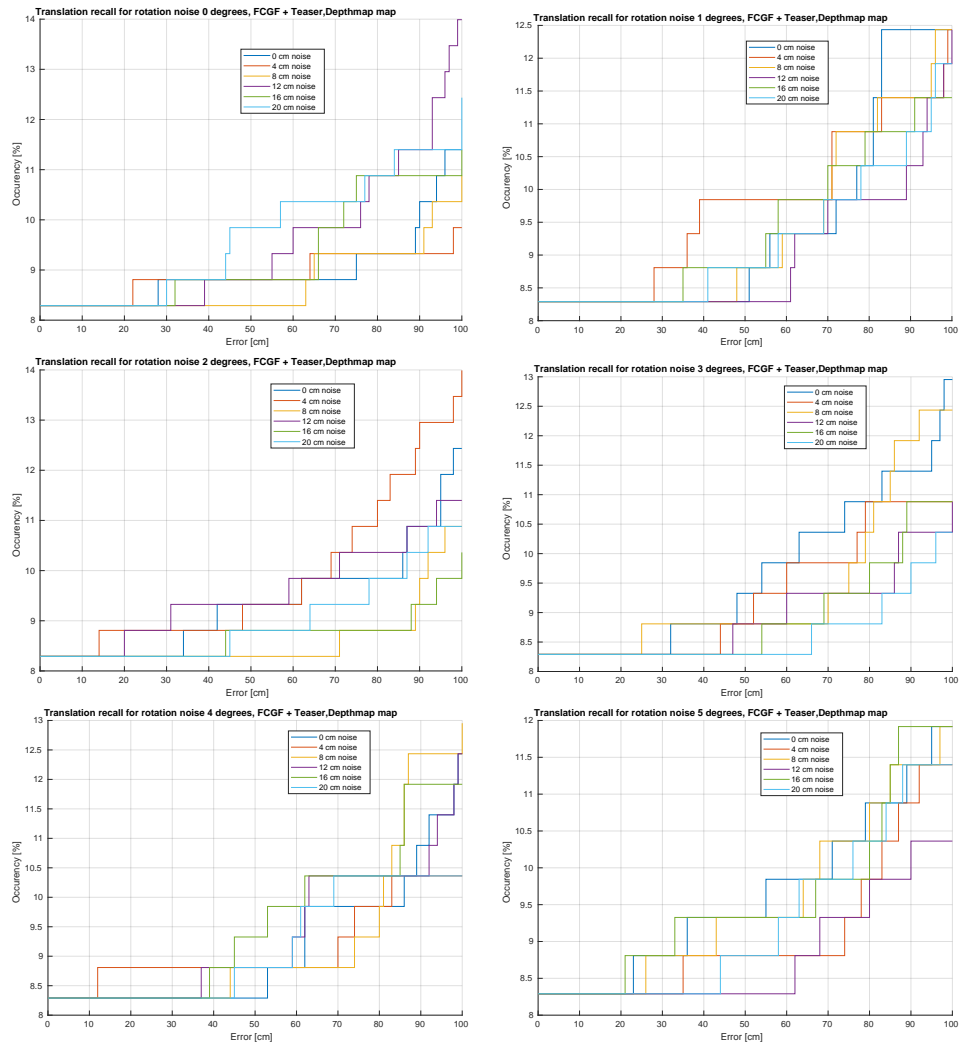
**Figure B.21:** Translation error for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.



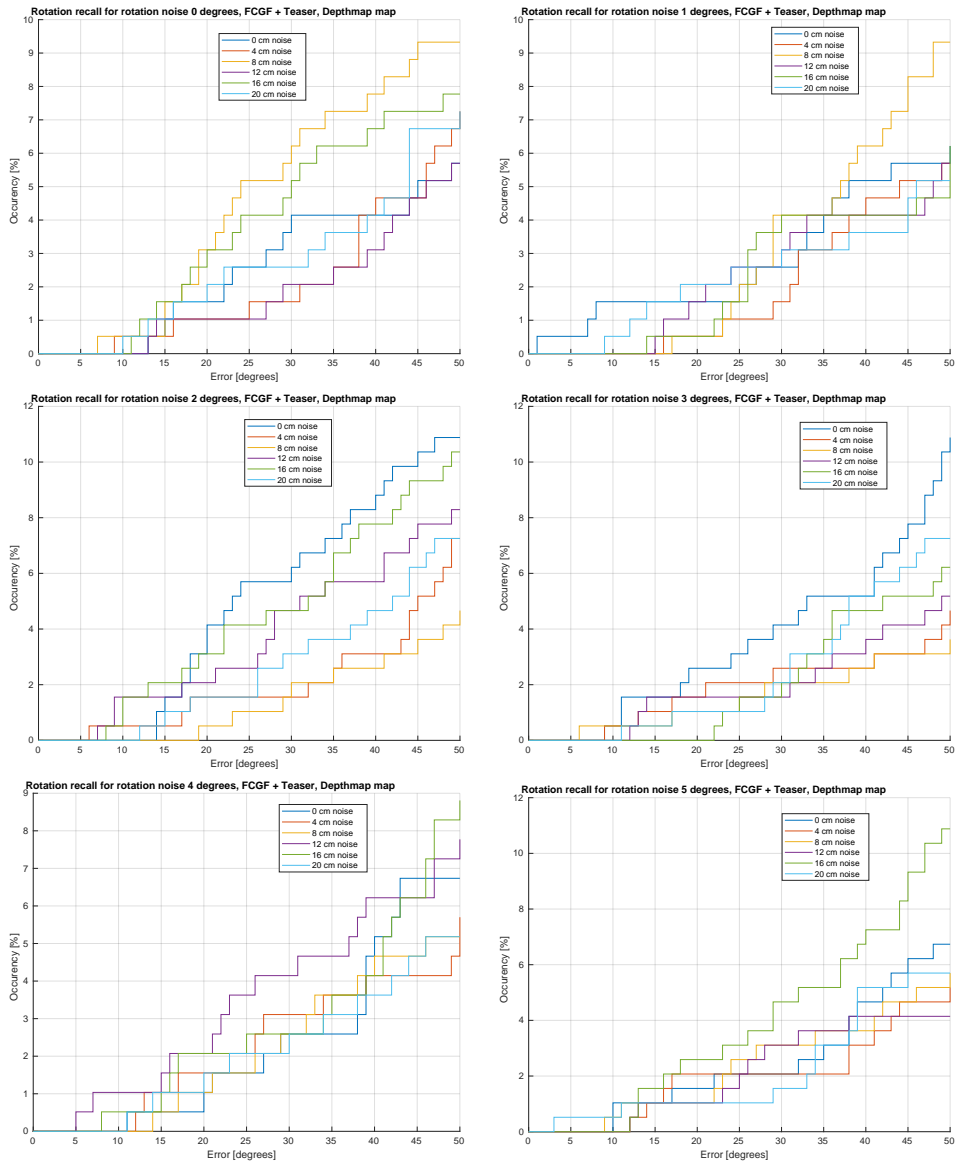


**Figure B.22:** Rotation error for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.

B. Full evaluation results



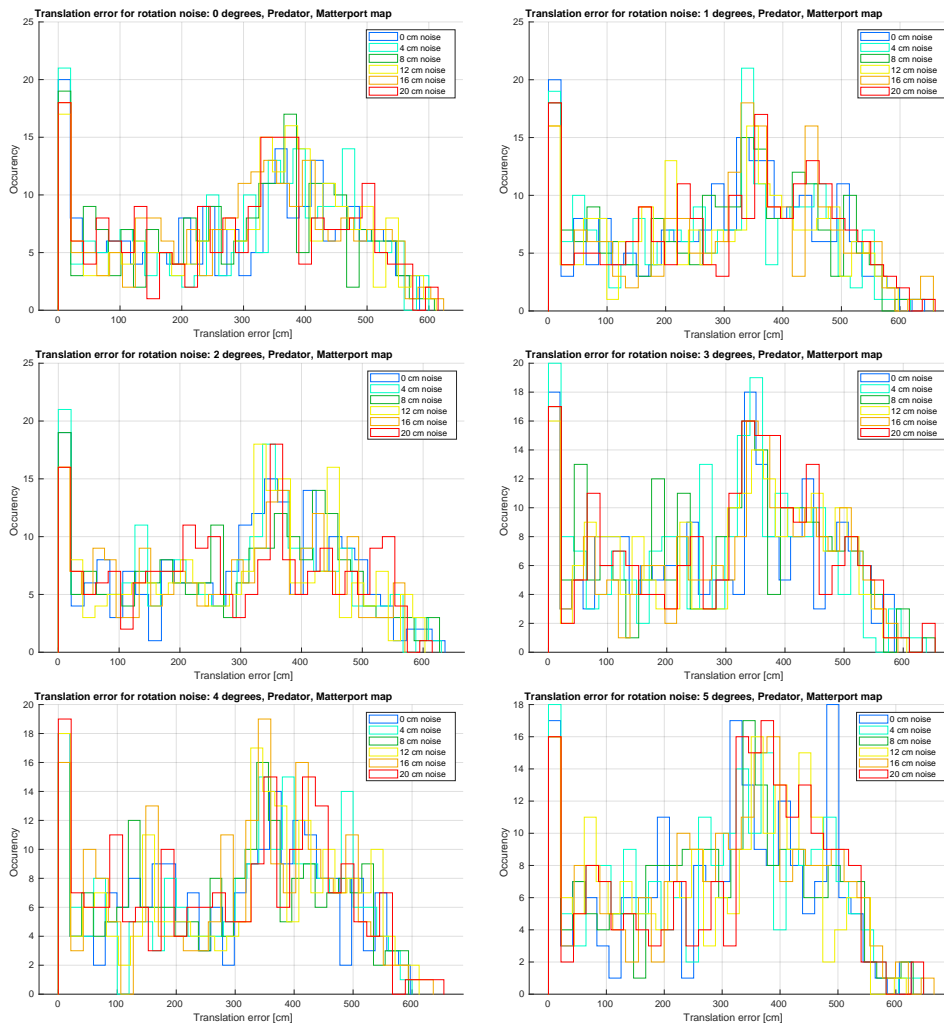
**Figure B.23:** Translation recall for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.



**Figure B.24:** Rotation recall for FCGF + TEASER++ alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.

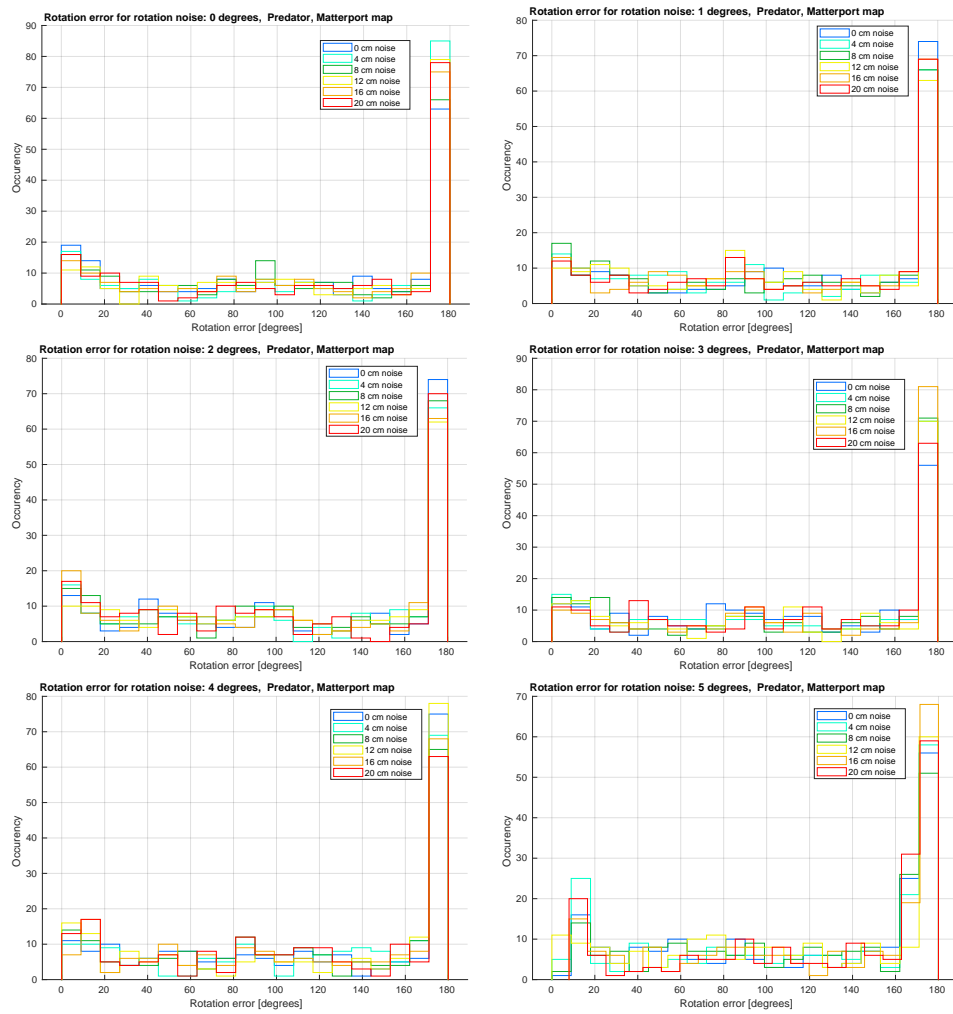
## **■ B.4 Overlap PREDATOR**

This section focuses on full evaluation of Overlap PREDATOR alignment.

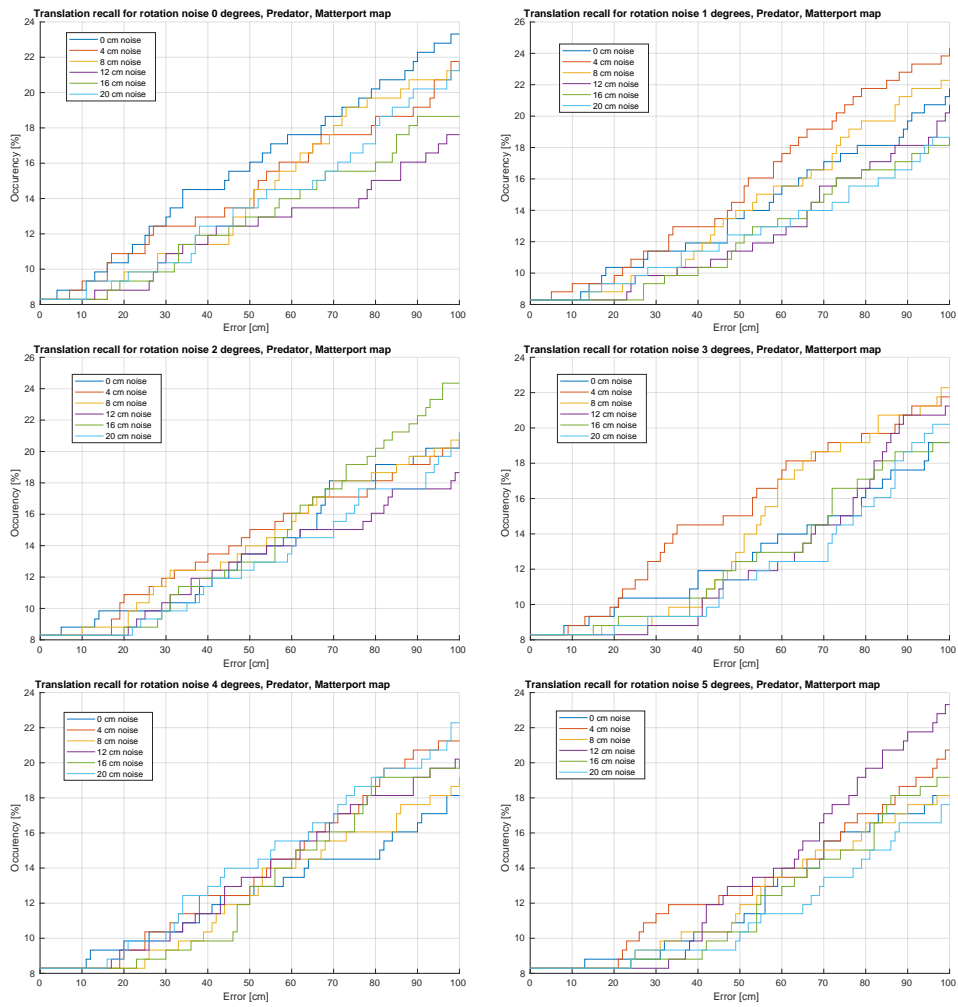


**Figure B.25:** Translation error for PREDATOR alignment over all translation and rotation noised point clouds on the Matterport map.

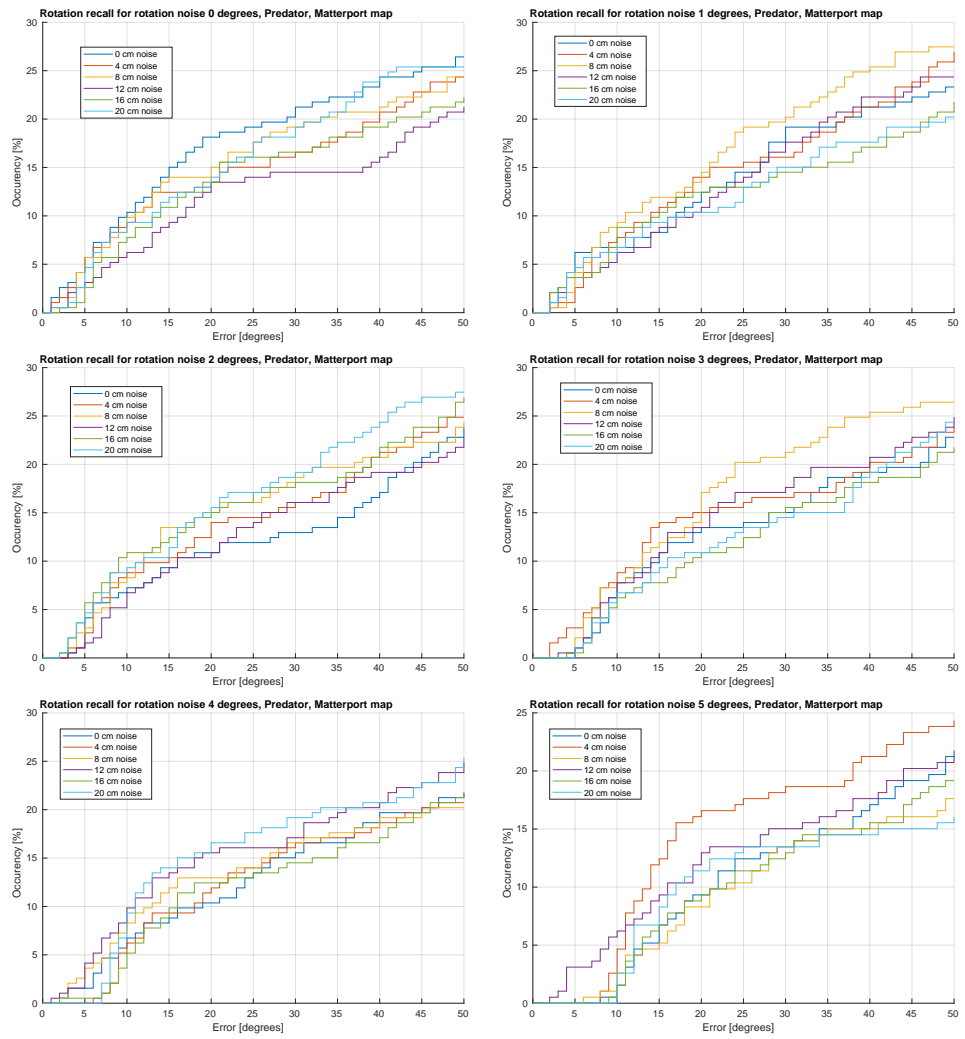
B. Full evaluation results



**Figure B.26:** Rotation error for PREDATOR alignment over all translation and rotation noised point clouds on the Matterport map.

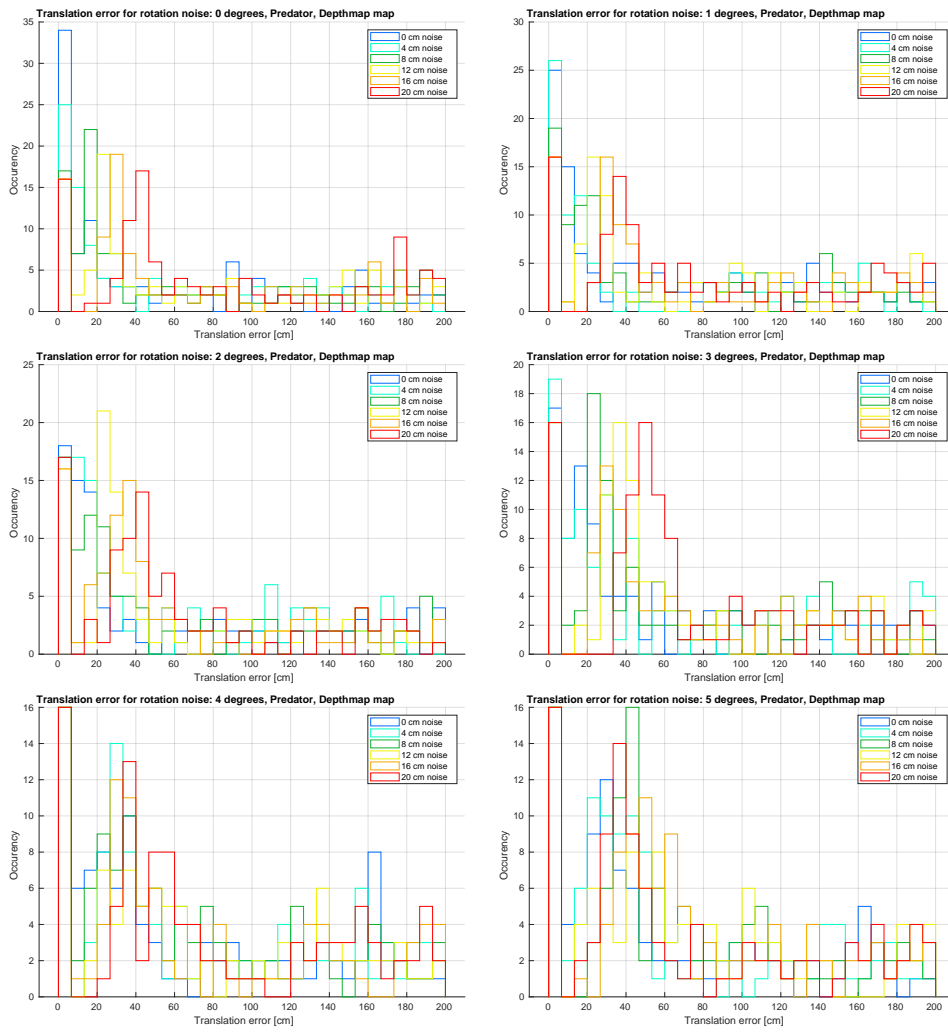


**Figure B.27:** Translation recall for PREDATOR alignment over all translation and rotation noised point clouds on the Matterport map.



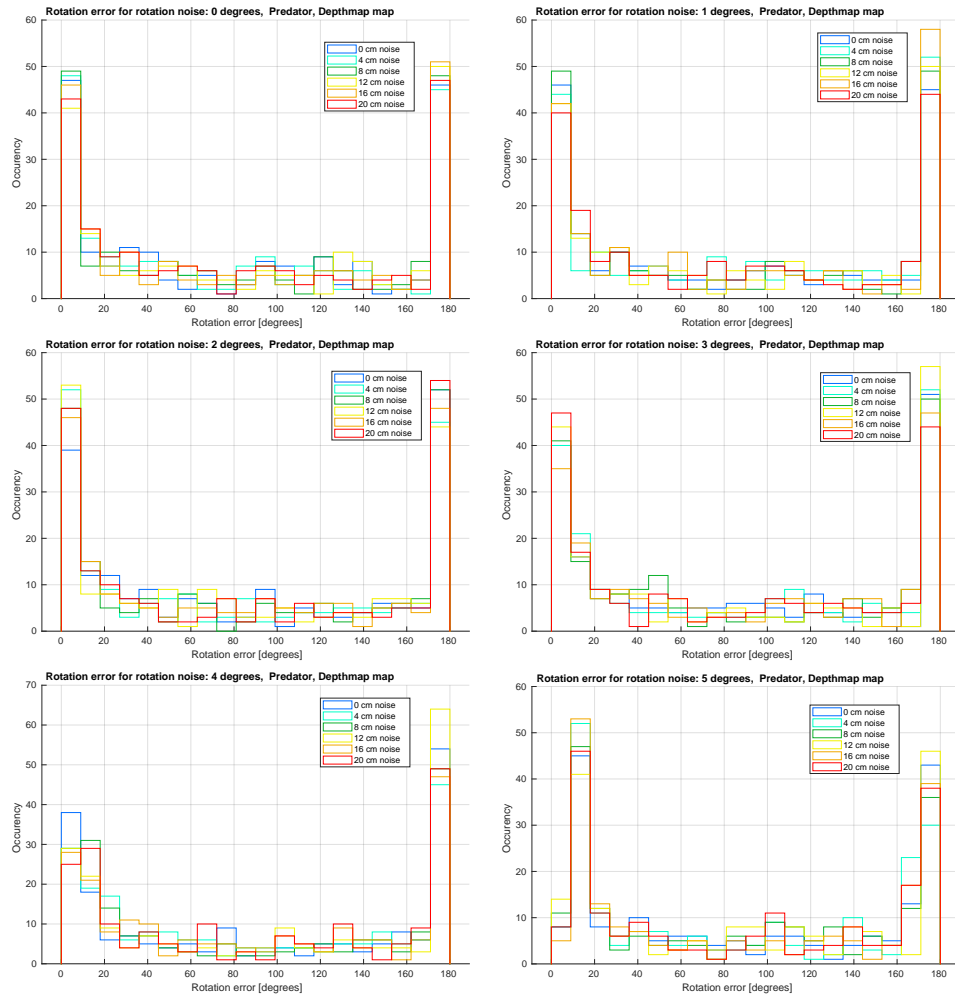
**Figure B.28:** Rotation recall for PREDATOR alignment over all translation and rotation noised point clouds on the Matterport map.



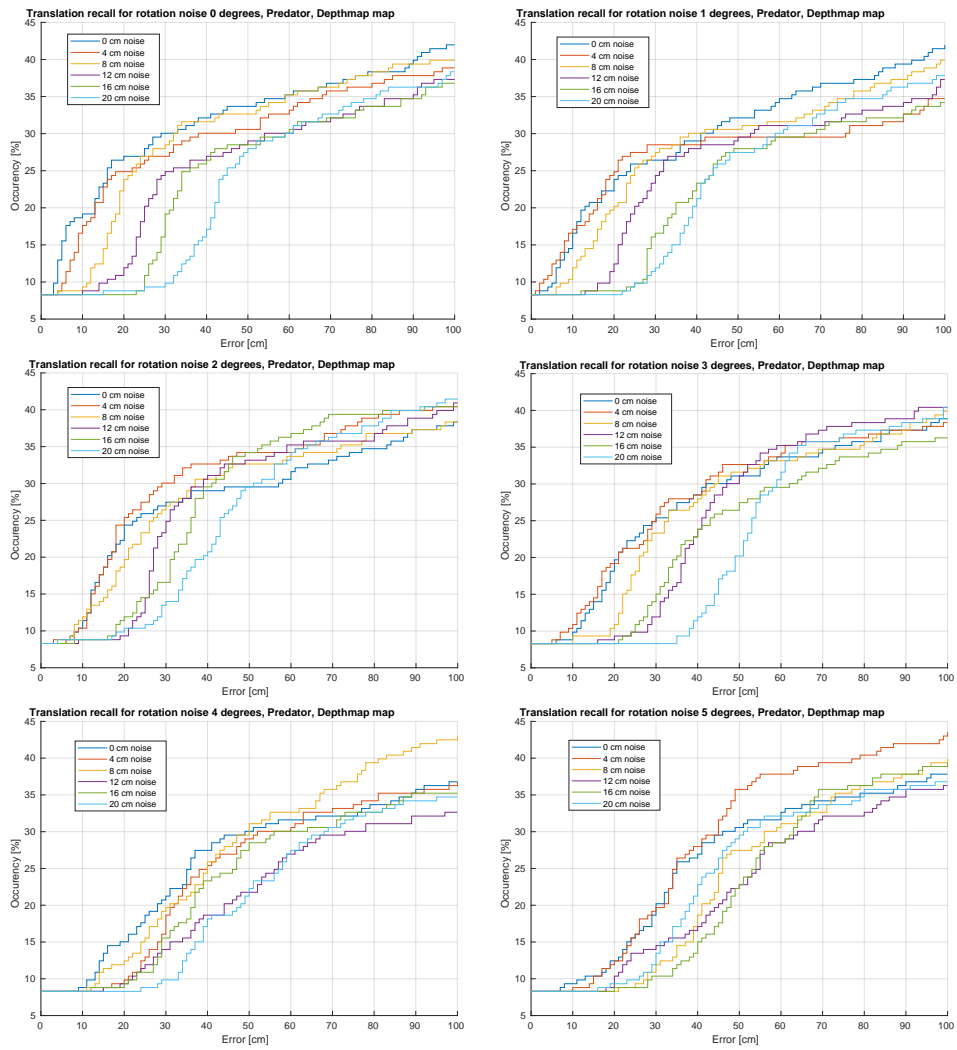


**Figure B.29:** Translation error for PREDATOR alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.

B. Full evaluation results



**Figure B.30:** Rotation error for PREDATOR alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.



**Figure B.31:** Translation recall for PREDATOR alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.

B. Full evaluation results

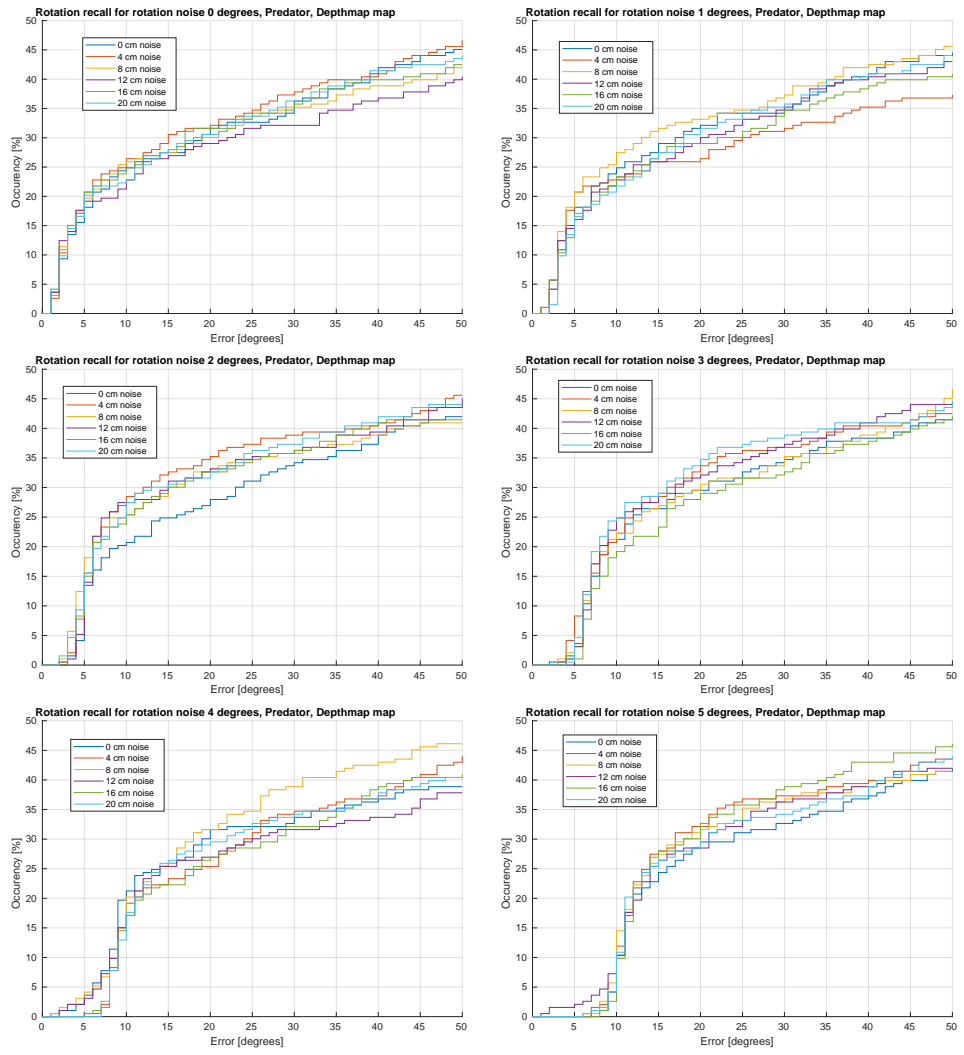


Figure B.32: Rotation recall for PREDATOR alignment over all translation and rotation noised point clouds on a map composed from HoloLens depthmaps.