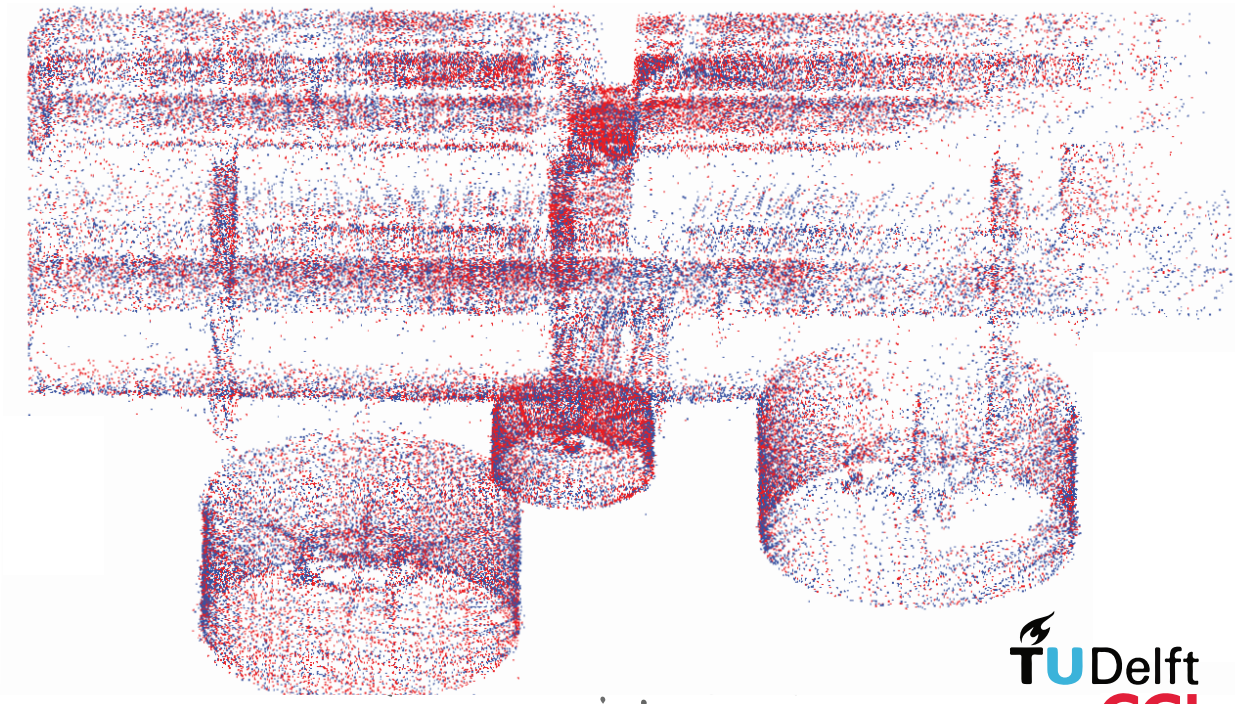


TU Delft
CGI

Synthesis project - 2019

Indoor localisation based on point clouds of the ceiling

Christina Fratzeskou
4934040

Chirag Garg
4817818

Karin Staring
4952510

Mutian Deng
4843312

Celine Jansen
4438205

Indoor localisation based on point clouds of the ceiling

Final Report

Synthesis Project

by

Christina Fratzeskou
4934040
Chirag Garg
4817818
Karin Staring
4952510
Mutian Deng
4843312
Celine Jansen
4438205

Supervisors

ir. E. Verbree
dr. ir. M. Meijers

Summary

Indoor localisation is a highly relevant topic. It can be used for many applications, such as indoor navigation. Current indoor localisation approaches all have certain downsides. In this report, the results of a completely new indoor localisation approach are described. The aim of this approach is to perform indoor localisation on room level based on a fingerprint solution using point clouds of the ceilings. The ceiling is used, because the ceiling does generally not change much and therefore it is easier to keep an up-to-date database. This research considers both the use of Dense Image Matching (DIM) input from pictures or videos made with a mobile phone and Light Detection And Ranging (LiDAR) input.

A database with point clouds of the global indoor environment is used to extract a signature in order to perform the fingerprinting approach. The signature will be matched with a signature extracted from a point cloud submitted by the user (LiDAR or DIM).

The full work flow of our indoor localisation implementation consists of the following steps:

1. Pre-processing: in such a way that the fingerprinting procedure can be performed more efficiently.

    - Handling tilt: This step entails aligning the ceiling of the input point cloud with the x,y-plane, just like the database, to compare z-values, which are height values of the ceiling. The most promising way to do this is by providing the local coordinates and orientation of the images/frames from videos before the reconstruction of the point cloud.
    - Voxel down sampling: In this step a voxel grid is used to down sample the user DIM input. In the input that is used, manual cleaning is applied first. This entails removing tilt and wall points. The resulting input was down sampled using the centroid of the points within the voxel boundaries. This generated a file with regularly distributed points.
    - Noise removal: Because the noise points affect the quality of the matching results, it is important to perform a noise removal step. In order to obtain more clean point clouds, radius-based and statistical-based noise removal are applied. During this process as much noise points are removed as possible, while retaining relevant object points.

2. Fingerprinting: There are different ways in which we can extract a unique signature from an input point cloud and a database point cloud:

- Histogram matching: For the histogram matching, the different normalised height values of the point cloud are represented in the different bin intervals of a histogram. The different occurrence frequencies of these height values are also normalised. The database and the input histogram can be compared through different distance measures. An advantage of this method is that it is a quick, simple (one-dimensional) approach. A disadvantage is that the solution is not as accurate because relevant spatial relationships between points are disregarded. It requires compatible input and database point clouds, meaning that preprocessing is required.
- Feature matching: For this method, the geometric property of points is used to find similarity between input and database point clouds. Here, features are theoretical representation of local surfaces, mathematically defined by a point and normals of its n neighbours. Corresponding features are identified by using certain criteria such as normal orientation, edge length of two points, distance between features in n dimension. Transformation is done in a pairwise manner based on these key correspondences and alignment is done in a iterative manner to perform final matching. One advantage of this method is that it takes into consideration the fitness of local surfaces of the ceiling with respect to geometry and topology to detect correct location. A disadvantage is that it requires refinement of parameters based on database or environment wherein localisation is to be performed. Also, the difference in realistic scale and coordinate systems of input and database point clouds influences the results, which can be resolved up to certain extent by rigid alignment.

After the fingerprinting, the quality of the results is assessed in a confusion matrix. The quality of the results of the LiDAR input is higher than the DIM matching. In general, the feature matching approach has the highest quality.

For future work, dead reckoning could be used to support indoor localisation approaches that use the ceiling. Besides this, incorporating dead reckoning with the scanner trajectory and adjacency information between different rooms will be a step forward to an indoor navigation solution.
Furthermore, an automated application of our approach would benefit from dealing with classification of the point clouds in order to filter ceiling points. It is recommended to look into different ways to combine the results of different matching methods in such a way that they can support each other.

Preface

In this report, our results for the Synthesis project 2019 are shown. The Synthesis project concludes the first year of the master's programme of Geomatics for the built environment at the Delft university of technology. The aim of the project is to "synthesise" the knowledge we gained during our first study year. We feel we fully managed to accomplish this. As an interdisciplinary team with lots of different backgrounds, we learned a lot from each other. Our project was not set in stone. The initial project that was appointed to us involved the Dutch Kadaster. After a major change of plans, we got an entirely different project at CGI to perform indoor localisation by using point clouds. This new project was not only challenging due to its contents, but also because we had the freedom to fully define everything we wanted to incorporate into our project. However, this freedom was not only a challenge, we managed to turn it into an advantage. We managed to learn many new things and we can now proudly present a unique end result.

We would like to thank our supervisors Edward Verbree and Martijn Meijers for their advise and for being open to the different directions we wanted to discover. We would also like to thank Robert Voûte and Bart Staats from CGI for this, and for providing the laser scanning data and arranging for us to scan our own point cloud. Also, a special thanks to Bart Staats for pushing us in the last week to think about providing information before the point cloud reconstruction instead of correcting it afterwards. Lastly we would like to thank Bert Meuleman from Moseg Technologies, for assisting us with the scanning of our own point clouds.

Christina Fratzeskou
Chirag Garg
Karin Staring
Mutian Deng
Celine Jansen
Delft, June 2019

Contents

Acronyms

DIM Dense Image Matching iii
LiDAR Light Detection And Ranging iii
GNSS Global Navigation Satellite System 1
SLAM Simultaneous Localisation and Mapping 2
ICP Iterative Closest Point 6
FPFH Fast Point Feature Histograms 6
MAT Medial Axis Transform 8
IMU Inertial Measurement Unit 13
API Application Programming Interface 14
TN True Negative 23
TP True Positive 23
FN False Negative 23
FP False Positive 23
IQR InterQuartile Range 34
CPD Coherent Point Drift 39
RANSAC Random sample consensus 6
RMSE Root Mean Square Error 76
SONAR Sound Navigation Ranging 95

1 Introduction

Indoor localisation is a highly relevant topic, since it can be used not only for finding the location of a user indoors, but also for indoor navigation. This can be useful for a wide scale of applications, for example for improving the efficiency of emergency management (e.g. rescue operations), providing guidance to people who are visually impaired, maintenance, the reconstruction of indoor spaces and to help civilians find their way in a public building (e.g. hospital, museum, airport, university building).

Using point clouds for indoor localisation and navigation is a topic that has received some attention over the past few years. With a mobile or static laser scanner, point clouds can be obtained. After "voxelising" these point clouds, the corresponding trajectory of a mobile laser scanner, can provide information for indoor navigation. Using this information, it is possible to identify walk-able spaces. There are three different kinds of walk-able spaces, namely staircases, slopes and horizontal surfaces [Staats, 2017]. These types of walking surfaces are important to make indoor navigation possible and to provide the basis of an indoor navigation graph. Another research aims to automatically obtain a navigation graph by detecting doors from a point cloud and its trajectory, because doors are important elements in the indoor environment. They connect spaces and play an important role in path finding indoors [Flikweert, 2019]. Navigation requires finding a path from the location of the user to their desired destination by using the navigation graph. This could be done by identifying and visualising the empty space directly from the interior of a point cloud after being structured and to calculate the shortest path through the empty space [Broersen et al., 2015].
Nonetheless, indoor navigation is impossible without having the location of the user and providing feedback and instructions until the desired destination is reached. There is research done that can be helpful for providing this information along the way. The research focuses on a graph-based indoor localisation method that uses mesh models resulting from Inertial Simultaneous Localisation and Mapping (VI-SLAM) on a hand held device. However, this method is not sufficient for an autonomous localisation process, because it requires an initial indoor location [Bot et al., 2019]. This initial location estimate can only be acquired by asking user input or maybe through support of a Global Navigation Satellite System (GNSS) position from when the user has not yet entered the building. The research presented in this report aims to provide an initial location to the user in an indoor space.

During the Positioning and Location Awareness course [Verbree, 2018], several indoor localisation and positioning solutions were discussed. An overview of these

methods and there disadvantages are described below.

- Wi-Fi fingerprinting is vulnerable to signal interference and therefore not always precise. It also requires an accurate, up-to-date radio-maps and a dense network of Wi-Fi access points [Conesa et al., 2018], [Xia et al., 2017].
- Bluetooth low energy beacons require the installation of hardware and a dense network of beacons. Besides this, signal fading is also a risk [Faragher and Harle].
- Simultaneous Localisation and Mapping (SLAM) could be used to estimate the position of a user and map the environment at the same time. This requires a base map and an accurate estimation of the initial position as described above. Localisation drift is a challenge as well [Bresson et al., 2017], [Mautz, 2012].

In general, a lot of methods are based on fingerprinting approaches which are vulnerable to signal interference and require accurate and up-to-date radio-maps [Huang et al., 2015]. In addition, too coarse location or positioning with a precision of several metres might not be suitable enough for actual indoor applications such as navigation. In this research, we aim to eliminate some of the disadvantages of the prevailing methods. Therefore, this research is aimed at finding a suitable indoor localisation solution that satisfies the following requirements:

1. The ability to provide an initial location to the user on room level.
2. A reference database that is as stable as possible to match the user input.
3. No, or just inexpensive, (hardware) infrastructure required.
4. The ability to provide an almost immediate response.
5. As limited signal interference as possible.

1.1. Research questions

In this research, we looked for a new approach in which we use point clouds to estimate the initial localisation of the user indoors. Instead of acquiring point clouds from the full indoor environment, we shifted our attention to ceilings. The selection of ceiling information as our main topic is considered to be an innovative approach, since there is nothing similar mentioned in the existing literature. Using the ceiling has several advantages. Ceilings do generally not change much. Changes in furniture or occupation are of little to no influence for the ceiling, meaning that maintaining an up-to-date reference database would take less effort for ceilings.

The use of user input in the form of pictures is interesting to consider, because this would widen the possible user base of an application. A mobile phone has an integrated camera sensor and is a commonly accessible device. User pictures could be supplied to DIM software to generate point clouds that could be used to match

with the reference database. This approach would therefore not necessarily require access to an infrastructure. In the case that user pictures do not work, it will also be possible to consider other types of input.

We would like to perform localisation on room level, because this might be useful for indoor navigation applications in a graph based structure where there rooms are the separate nodes.

Based on this information, we formulated the main research question and a few sub questions that will help answer our main research question:

Can point clouds from the ceiling be used for indoor localisation purposes on room level?

In the build environment, room names or numbers can be considered as a feasible reference for localisation purposes. However, a room does not always have a clear, unambiguous definition. Therefore, the definition of a room with special attention to the ceilings within the scope of our project has to be defined.

1. **What is a room?**
    - **How to distinguish hallways?**
    - **How to distinguish the ceiling of a room?**
2. **What ceiling characteristics does the user need to capture?**
    - **What part of the ceiling should be captured?**
    - **What user instructions are necessary to make sure that the right parts are captured?**
3. **What is a possible method to perform indoor localisation using point clouds of the ceiling?**
4. **What kind of pre-processing is required for these possible methods?**

1.2. Reading guide

This report is structured in eleven chapters, of which this is chapter 1. A research overview has been given and a problem statement is conducted from this. Based on the problem statement, we defined our research questions. In chapter 2, a theoretical background of related work and concepts will be given. Chapter 3 defines the scope and goal of our project. It provides an overview of the things we would like to achieve during this project. In chapter 4, the scientific workflow of our indoor localisation solution is described. Chapter 5 will discuss the project-wise decisions made during the process. The methodology in chapter 6 will outline the followed steps along with the technical details. Following the methodology, the results of our experiments will be included in chapter 7. The conducted results will

be interpreted and assessed in chapter 8. After this, we will be able to answer our research questions in chapter 9. In chapter 10, we will give a conclusion with recommendations for further research. In the end, we will reflect on this project in chapter 11.

2
Theoretical background

In this chapter entails the theoretical background of the methods used in this report.

2.1. Sensors mobile phone

A mobile device has different sensor types listed in figure 2.1. The gyroscope and accelerometer are used systems to capture motion [Brezmes and Cotrina, 2009].

2.2. Down sampling methods using a voxel grid

Voxel down sampling is most often used for the reduction of the number of points with the aim of generating a representative output that will improve the processing of the point cloud as a whole.
It is based on a grid of volumetric pixels called voxels, which is superimposed over the point cloud [Pirouz, 2016].
For the purpose of controlling the point distribution of the point cloud two methods were considered. Both methods reduce points within a voxel to a single point. The point can be either the geometrical centre of the voxel or the centroid of the point distribution within a voxel. In this project the latter was used since it provides a more accurate approximation of the point cloud's distribution. The drawback of the method is that it more computationally intensive in comparison to the geometrical centre implementation [Rusu and Cousins, 2011].

2.3. Noise Removal

Point clouds are often contaminated by noise which would affect the further process and analysis of point clouds. The aim of noise removal is to clean the point clouds thus improving the accuracy of finger printing matching. The sphere can

Sensor Type	What is measured	Unit of measurement
3 axis Accelerometer	Acceleration on X,Y and Z	G force or meters/second^2
3 axis Gyroscope	Rotation around X,Y and Z	radians/second
3 axis Magnetometer	Magnetic field around X,Y and Z	microTesla
GNSS reciever	Latitude and Longitude ,in WGS84	degrees
Microphone	Sound	dB
Camera	Colour intensity	RGB radiance
Proximity sensor	Distance between device screen and object	centimeters
Light sensor	Light intensity	Lux
Temperature sensor	Temperature	degrees Celcius
Barometer	Air pressure	hectoPascal or millibar

Figure 2.1: Different sensor types of a mobile device [Lightvoet, 2017].

be used to detect and remove the outliers, the method is to calculate the number of points in a defined-radius sphere and removes points with few neighbours. In addition, distance can be calculated to remove noise, points that are further away from neighbours are detected as noise points.

2.4. Histogram approach

In the literature, three-dimensional histograms are used to describe point clouds. These approaches are more similar to the feature matching approach described in this report, using Fast Point Feature Histograms (FPFH). Some require computing the normals for each point in the point cloud [Jones and Aoun, 2009]. Others generate histograms based on the curvature and diffusion distances [Mahmoudi and Sapiro, 2009]. In robotics, histograms containing for example sensor data from lasers are also used for localisation purposes [Kwolek, 2004]. However, an approach such as implemented in this research seems to be completely new. By capturing a point cloud and storing the heights values in a one-dimensional histogram, a quick and efficient signature is created. Using this signature in a fingerprinting solution can be a solution for indoor localisation purposes. However, the simplicity of a one-dimensional histogram also has its downsides. Large rooms with many similar ceilings might not lead to the level of distinction between different histograms that is needed within a successful fingerprinting approach.

2.5. Feature Matching approach

Point cloud registration, a process to align two point clouds by determining the relative transformation between them, has been at the core of many localisation applications [Eckart et al., 2018]. Some of the traditional algorithms such as Iterative Closest Point Iterative Closest Point (ICP) become inaccurate and time consuming when there is noise and variation of point distribution in point cloud data. New methods such as greedy initial alignment method, which improves accuracy by providing robustness and improved selection of point features but gives high processing time [Rusu et al., 2009]. In the proposed feature matching approach, Random sample consensus (RANSAC) is used to perform pairwise registration. In this method, the correspondences between two point clouds are detected by comparing the features based on their geometric relations and transformation is done in a iterative manner to perform final matching [Zhou et al., 2018]. This feature matching based fingerprinting approach uses the geometric property of points to find similarity between input and database point clouds. To create a signature, mathematical features are formulated by first estimating the normals of the points based on their neighbouring points and computing a 33-dimensional FPFH feature for each point by using hybrid KD-tree on point clouds. Hence, each direct neighbour of a feature is connected to nearby features. This provides the advantage that there is spatial coherence between two point clouds and they are aligned in such a way that the local surfaces fit together with respect to geometry and topology. However, without knowledge of initial pose, scale and absence of colour values in points can make it difficult to identify correct correspondences and perform transformation thus leading to false results.

3
Project expectations

In this chapter, we provide an overview of the things we like to achieve during this project. For this we also used the MoSCoW method [Mulder, 2017], as was listed in our inception report.

Within our project, we aim to investigate the possibilities of indoor localisation by using point clouds of the ceiling. For this, we consider a fingerprinting approach, in which we compare the signature of a user input point cloud to the signature of the database point cloud.
Because we would like our implementation to be a building block for a wide range of further applications, our solution should be usable for the largest amount of people possible. Not all everyday users will have the possibility to use a LiDAR scanner, but nowadays almost everybody has a smart phone. For this reason we look into the possibility to use input images or videos from smart phones for the user input DIM point cloud generation.
However, to make matching in a fingerprinting approach possible, the input and the database should be comparable. We realise that this might be difficult to achieve with a point cloud generated through DIM. For this reason, we also consider user input from a LiDAR scanner, which matches with a different type of use-case.

During this project, we consider the separate steps to get our specific indoor localisation approach to work. The final result entails the combined results of these separate steps. There be no fully automated application for our approach. When this project is finished, we are able to indicate whether or not, and under what conditions, it will be feasible to perform indoor localisation using point clouds from pictures of the ceiling.

3.1. MoSCoW

Must:

- See if pictures from the ceiling provided by users can be used for localisation (are they differentiating enough). For this we will investigate in which way (e.g. which subsection of the ceiling, pictures or videos etc.) the users should provide their input to acquire a feasible result. We will also need to look at the quality of the localisation.
- Extracting the (distinct) signature of a point cloud.

Should:

- Compare the performance of different software packages for DIM, such as COLMAP, Pix4D, Archicad and VisualSFM, to see which software package provides the most suitable results for our approach.
- Accurate localisation based on room level by using additional information like dead reckoning combined with topology information or the scanner trajectory.
- Implement a minimal user interface (prototype).

Could:

- Perform automatic classification of ceiling points from the point cloud. For this we could for example use the Medial Axis Transform (MAT), or look at intersections between a line perpendicular to scanner trajectory and fitted planes planes to discover a possible ceiling height at which we can extract points.
- Incorporating door, window and furniture classification from the point cloud to improve the efficiency and accuracy of localisation.
- Incorporating machine learning; Include user measurements in a database to make future localisation more accurate and more efficient (crowd sourcing).
- Create a web platform and application for taking input and sending data to a server, process it and show the location of the user on a map.
- Look into different necessities for different user types (e.g. firefighters need different things than students).

Would not:

- Incorporate Wi-Fi fingerprinting.
- Look at navigational graph between rooms for navigational purposes.
- Look at different buildings than the architecture building.

4
Conceptual model

The scientific work flow of the indoor localisation solution described in this report, is described in this chapter.

All major steps needed to come to an indoor localisation solution as described in this report, are shown in figure 4.1. In this figure, references are listed to the subsection in which the methodology corresponding to the mentioned step is explained.

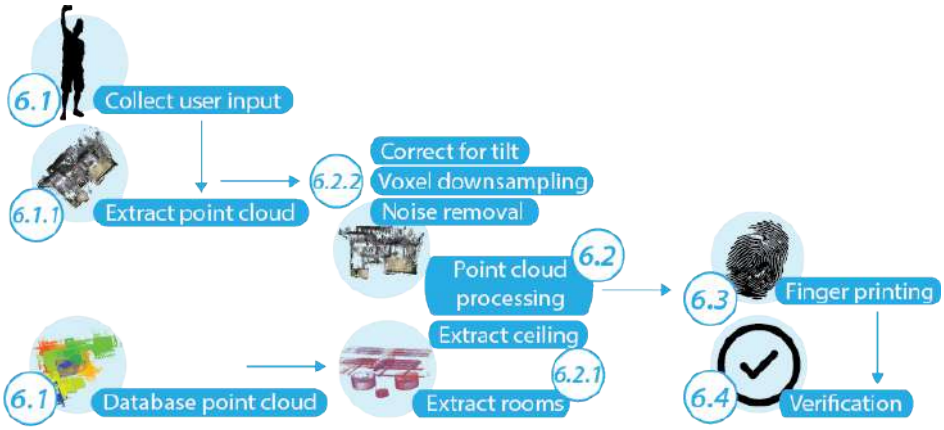


Figure 4.1: The main steps of a methodology to perform indoor localisation based on a database point cloud, and a user input point cloud generated through DIM.

4.1. Collecting user input

User input point clouds can be captured in several ways. In this report, LiDAR and DIM are considered. Within the LiDAR approach, the user directly captures a LiDAR point cloud. For the DIM approach, the user will capture an input video. DIM is applied on frames extracted from this video, resulting in a point cloud.

4.2. Pre-processing

To prepare both the database and the user input point cloud for the fingerprinting solution, some pre-processing is required. For the LiDAR database point cloud, this pre-processing involves room and ceiling point extraction. This process is described in subsection 6.2.1. The point clouds of the ceilings of the separate rooms are each stored as a separate database entry. Depending on the type of user input, more pre-processing is required. The pre-processing of a LiDAR input point cloud involves

the same steps as the database point cloud. For a point cloud that is created through DIM, the following additional steps, of which the methodology is described in subsection 6.2.2, need to be taken into consideration.

4.2.1. Handling tilt

The point clouds captured with DIM are captured in a local coordinate system which can be subject to tilt. When an input point cloud is tilted, the input ceiling is not parallel to the corresponding ceiling in the database. Among others, the tilt depends on how the users hold their camera when capturing the video which is used as input for the DIM process. This tilt can be handled beforehand by incorporating this angle in the DIM process. Another option would be to remove the tilt afterwards. The methodology for tilt handling, is described in subsection 6.2.2. This step uses the output which contains the extracted ceiling points.

4.2.2. Voxel down sampling

The LiDAR database point cloud and the DIM user input point cloud are not necessarily compatible. The make the point clouds more comparable in point density, and to speed up the processing, voxel down sampling is applied. The methodology for this process is described in subsection 6.2.2. This step is conducted uses the output file from the tilt handling.

4.2.3. Noise removal

Relatively speaking, there will be less dynamic noise present when focusing on the ceiling. However, the DIM process might introduce some noise. A methodology for the noise removal process is discussed in subsection **??**. This step uses the output file from the voxel down sampling.

4.3. Fingerprinting

The work flow of a general fingerprinting approach is displayed in figure 4.2. In fingerprinting, a distinguishable signature extracted from a database of locations, is compared to a signature extracted from user input. The best matching signature will indicate the location of the user. The methodology for the fingerprinting step is described in section 6.3. The fingerprinting approach uses the output file containing the pre-processed result.
Generating a signature of a point cloud can be done in several ways. The matching step in the fingerprinting approach depends on the type of signature that is extracted. Within this report, two types of signatures are taken into consideration;

4.3.1. Histogram matching

Within histogram matching, the normalised height values of the point clouds are divided over the bins of a histogram. By doing this, a one-dimensional signature is created. By comparing the minimal distance between bins of the database histograms and the input histograms, the most likely location is selected. A methodology for this type of fingerprinting is described in subsection 6.3.1.

4.3.2. Feature matching

Within feature matching, the point cloud itself behaves as the signature. Based on the smallest distance between the points in a database point cloud and the input

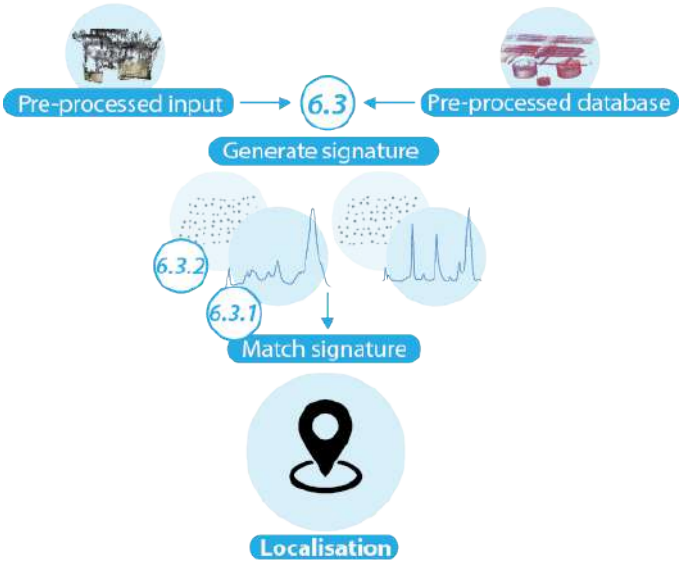


Figure 4.2: The general steps of a fingerprinting approach.

point cloud, the most likely location is selected. A methodology for this type of fingerprinting is described in subsection 6.3.2.

4.3.3. Combined matching

To improve the quality of the results, it is possible to combine the results of different fingerprinting approaches. For this, the methodology is described in subsection 6.3.3.

4.4. Verification

As a final step, the quality of the localisation solution are assessed. Quality measures can be used to provide the user feedback on how sure the localisation is. Within the research described in this report, a verification step is implemented to indicate the quality of the localisation approach. The methodology of this verification process is described in section 6.4.

5
Project specifications

In this chapter, we will discuss the project-wise decisions made during the process described in this report. In the first place, the ways in which we collected data are described. After this, the different experiments we performed to test our localisation approach will be discussed. We will then describe the choices we made regarding pre-processing, quality assessment and combining the separate steps of our implementation. Finally, we will list the choices about resources, such as different software packages and libraries, we made during the project.

5.1. Data collection

We collected data in two different ways, through LiDAR scanning and by capturing videos. A DIM approach is applied on the captured videos to acquire a point cloud.

5.1.1. Collecting LiDAR point clouds

We captured our own LiDAR point clouds with the `GeoSLAM ZAP REVO RT`[1]. The device is displayed in figure 5.1. The `GeoSLAM ZAP REVO RT` is a hand-held device that uses a SLAM algorithm to connect newly captured points to the points that are already captured.

The results can be visualised in real time. The main advantage of a hand-held scanner, is that it is easier to deal with occlusion. The user can physically walk around the subject of interest to do a few crossings. This ensures that the subject of interest will be captured in the best way possible. The handling of the device requires some practice. It is quite heavy and there are a few best practices that should be honoured to gain the best results possible. When for example walking through a door, you have to keep a gradual transition between features for the SLAM algorithm to be able to match the new points to the points that are already captured. For all changes in direction, it is important to keep focusing at a subject with several features of interest so that the changes between different features are gradual and the SLAM algorithm will still be able to match the points.

A challenge with using this LiDAR laser scanner for user input, is in the first place the cost. Day to day users will not have the possibility to own such a costly device. Furthermore, the device takes a while to prepare for scanning, and the optimisation also takes some time. This can be a disadvantage for our solution, in which we would prefer the localisation to be quick. The point cloud data itself is directly forwarded to a data-logger web-server, which is advantageous because this could fit right within our work flow. The point cloud itself is captured quickly (walking speed makes almost no difference in quality), but it is difficult to solely capture a

[1] https://geoslam.com/zeb-revo-rt/

Figure 5.1: The GeoSLAM ZAP REVO RT (Moseg Technologies).

specific feature. The surrounding features will also be captured. For this reason, some form of automatic classification is required to automatically filter the ceiling points our application requires.

The scanner captures the points in a local coordinate system. This system cannot be specified beforehand and is different between different measurements. The scale and the orientation towards the x,y-plane remains the same. Theoretically, it would be possible to transform the point clouds to the same coordinate reference system by matching certain reference points.
The Inertial Measurement Unit (IMU) inside the scanner can be subject to drift when you perform long continuous measurements. To be able to correct this drift as much as possible, we finish each scan at the same location as we started. For every detected error, there will be a weighed correction applied over the entire trajectory. This correction ensures that the actual error decreases, but the overall measurements will be less precise. Within the first optimisation step, the so called 9 % point clouds will have a precision of two to three centimetres. This is more than sufficient for our approach. After optimisation, in which enriching the point cloud with RGB-values is an option, the 100 % point clouds will have a precision of eleven millimetres. During our project we were not able to use the 100 % point clouds, because we measured at the 13th of June and our deadline is at the 24th, meaning that the processing took too long to be able to analyse the results and process them in our report. For future implementations, using the 100 % point clouds in our database could further improve the results.

5.1.2. Collecting DIM point clouds

To test whether or not it would be feasible to capture point clouds of the ceiling based on a DIM approach, we conducted several small tests in our inception report. Pix4D seemed to be the software package that provided the most suitable results for our specific implementation. For this reason, we decided to stick to Pix4D for further testing purposes. Note that the use of Pix4D in an automated application that performs our full indoor localisation work flow, might be feasible through for example an Application Programming Interface (API). It would also be possible to use the Open3D library [2]. This is an open-source Python library for 3D data. Another possibility would be to use the non open-source module `photoscan` from Agisoft, which could also be part of a Python work flow. During our project we will not look into implementing such solutions.

For further experiments with Pix4D, we tried to capture the ceiling with many different strategies, for example at different angles and while rotating. We also experimented with different amounts of overlap between images. For capturing the videos, three mobile phones were used of which the specifications of the devices are described in table 5.1.

Mobile device	Quality indicator	Frames per second
Sony Xperia Z3 compact	1920 x 1080 pixels	30 fps
Samsung J5 Prime	1980 x 1080 pixels (Full HD)	30 fps
Iphone 7	1980 x 1080 pixels (Full HD)	30 fps

Table 5.1: Quality specifications of the mobile devices used

After experimenting with different types of software and with different kind of user input, we decided to capture user input videos for all ceilings in the database.

Figure 5.2: Point cloud extracted based on 159 video frames while filming for 28 seconds. Room: Hallway Geolab in the Architecture building. Software used: Pix4D with the standard setting **3D Models**.

Figure 5.3: Point cloud extracted based on 159 video frames while filming for 28 seconds. Room: Hallway Geolab in the Architecture building. Software used: Pix4D with the standard setting **3D Maps**.

[2] http://www.open3d.org/

The arbitrary way of capturing user input, as described in subsection 6.1.1, has not been optimised yet.
Once the data was collected for different ceilings, we processed the videos in Pix4D with the standard settings for 3D models. We decided on this standard setting, because this setting takes input of any set of images with high overlap. The standard 3D maps requires aerial images using a grid flight plan with high overlap, mostly oriented towards the ground. The results generated with 3D models are not necessarily better than the results generated with 3D maps. This can be seen in figure 5.2 and figure 5.3. The differences between those two settings are shown in table 5.2.

Main setting	**Matching image pairs**	**Matching window size**	**Limit camera depth automatically**
3D models	free flight / terrestrial	9 x 9 pixels	yes
3D maps	aerial grid / corridor	7 x 7 pixels	no

Table 5.2: Description of the differences between the standard setting 3D models and the standard setting 3D maps

An example of our experimenting process of the use of different amount of overlapping pictures, is shown in figures 5.4 and 5.5. For this extreme example, the processing times did differ quite a lot. The processing time for figure 5.4 was close to three hours, whilst the processing of figure 5.5 took about five minutes. The three hour wait did provide better results, but the results from five minutes still seem sufficient for our application. There is a trade-off between using too many frames resulting in slow processing, which is unrealistic for a real-world application, and using not enough frames to capture the details our application requires. For future implementations more research is needed to improve this.
However, the amount of time the point cloud extraction takes in Pix4D, does not solely depend on the amount of pictures used in the process. In general, rooms with many repeating elements and rooms with smoother textures lead to slower processing. In these situations, the algorithm has more difficulty in finding distinct, corresponding points between images. The quality of the DIM point cloud also depends on the amount of distinctive features present on the ceiling. This will be dissimilar for different kinds of ceilings.
Note that the points clouds in figures 5.4 and 5.5 contain wall points. Especially in hallways, capturing the walls is hard to prevent. Removing undesired wall points is a possibility. It would also be possible to keep some wall points in both our database and our user input for the cases where the user is more likely to capture wall points. In section 9.1 and later in this chapter in section 5.3, we will discuss handling wall points more elaborately.
Even though the point cloud in figure 5.4 is less noisy than in figure 5.5, it still contains noise. This is the case for all the point clouds we extracted with DIM. In

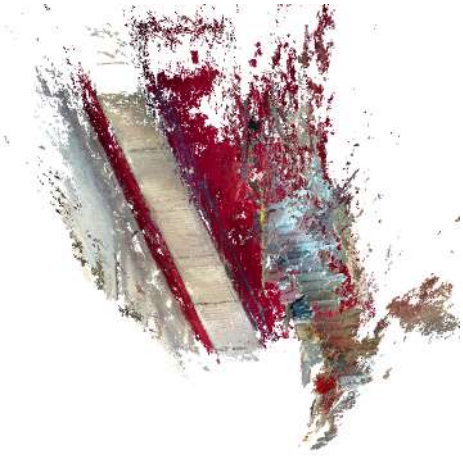


Figure 5.4: Point cloud extracted based on 510 video frames while filming for 35 seconds. Room: West hallway to the library in the Architecture building. Software used: Pix4D.

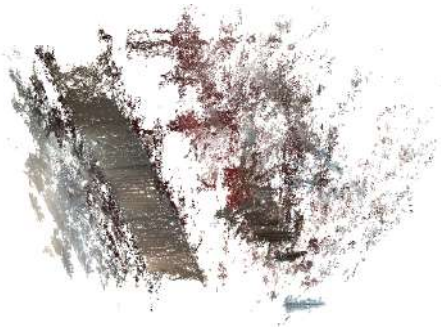


Figure 5.5: Point cloud extracted based on 69 video frames while filming for 34 seconds. Room: West hallway to the library in the Architecture building. Software used: Pix4D.

subsection **??** we will discuss our methodology to deal with this noise.

5.2. Experiments

To test our localisation approach, we conducted several experiments. The first experiment involved using an initial LiDAR point cloud database which we extracted from data we received from CGI. After this, we captured our own LiDAR database point cloud with the help of Moseg Technologies. Note that the point clouds of the ceiling are captured in the Architecture building of the TU Delft. This specific building has ceilings with many distinct features, which is not the case for every building. The performance of our localisation method will be different for different types of ceilings.

5.2.1. Initial LiDAR database

To be able to perform some tests before capturing our own point cloud, we extracted some rooms from an existing point cloud database from CGI. These rooms are displayed in figure 5.6. The corresponding point clouds are shown in appendix A.2. Note that the original point cloud contained all connected rooms. In our approach, we disregard all adjacency information between rooms and the captured scanner trajectory. Incorporating this additional information could improve future implementations of our localisation solution.

Experimenting with DIM user input

We tested this initial LiDAR database with 27 DIM user input point clouds extracted from videos. The initial idea was to capture two for each room, however, room P and the orange hall were occupied when we went to measure as a group. To have at least one measurement, one team member went back to capture them later. For

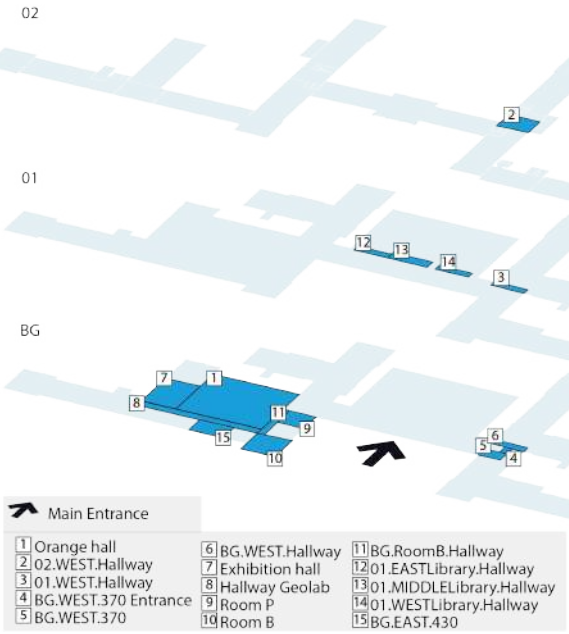


Figure 5.6: The rooms we originally extracted from the CGI database point cloud.

the east section of the hallway to the library, Pix4D could not find enough feature points for one of the videos. This can be due to the fact that we filmed this case in a narrow hallway in the width instead of the length. This probably involved too much turning of the user, combined with the repeating features without enough depth difference between the pictures.

Experimenting with subsets as LiDAR user input

The user point clouds generated with DIM might not be compatible in point density and quality with the database point cloud acquired through LiDAR scanning. This means they might not be suitable for comparison. However, it is not necessarily the case that our application needs pictures to extract a user input point cloud. It could for example also be possible that a certain type of user has access to a LiDAR scanner. To be able to see whether or not our approach is suitable for different types of input points clouds, we will also generate user input with a LiDAR scanner. Before we had access to a real LiDAR scanner, we wanted to perform some testing. Our test user input LiDAR point cloud should be as realistic as possible. In order to acquire such realistic "test" LiDAR user point clouds, we took subsections from a selection of our original database point clouds, re-oriented and re-scaled them and applied thinning. The thinning is applied because the point cloud is likely collected in a quick way, meaning that it will be less dense than the database point cloud. A realistic input is likely to be orientated and scaled differently than the database, since the user uses a different local reference system than the database.

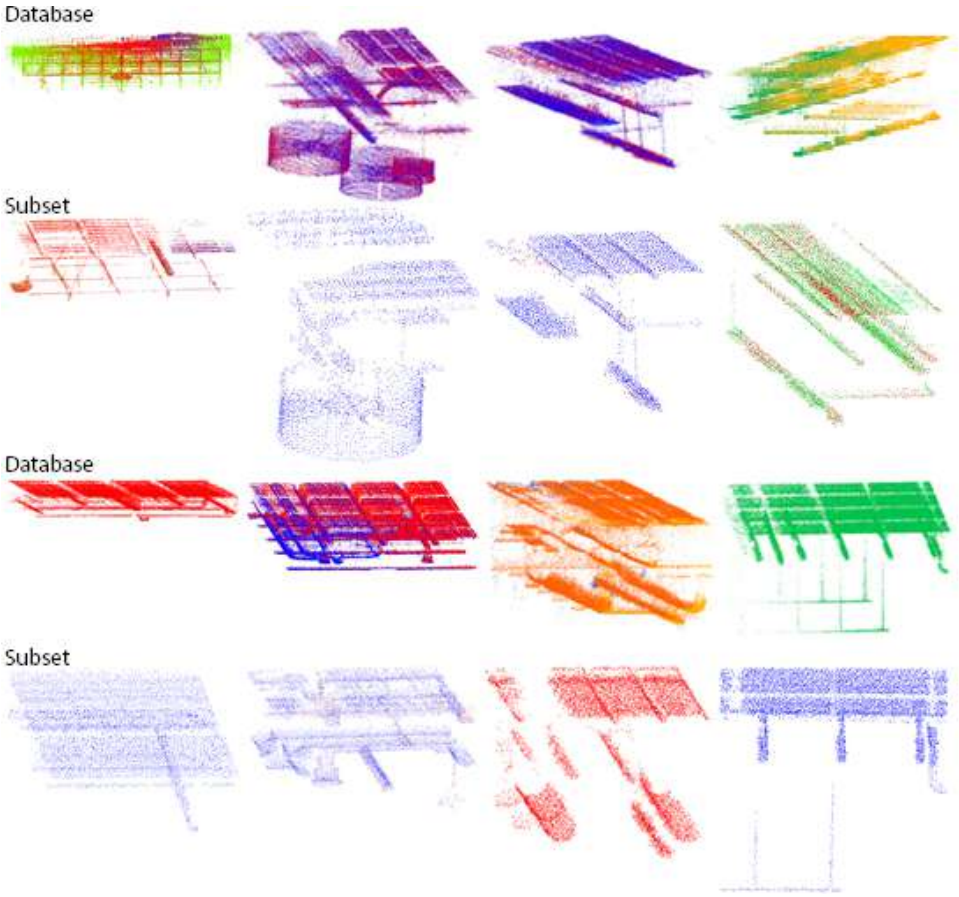


Figure 5.7: Subsets extracted for testing the histogram approach. We selected the orange hall, BG.WEST.370, the BG.WEST.Hallway, the hallway from the Geolab, Room P, Room B, the hallway from room B, and BG.EAST.430.

The original point clouds and their resulting subsets are shown in figure 5.7. On these subsets we implemented a heavy noise filter and spatial thinning in Cloud Compare. It is important to note that despite the fact we used only eight subsets from figure 5.7 as new user input, we performed out tests against the original database contained all fifteen rooms as displayed in figure 5.6.

Note that, as has already been described in section 5.1, the idea of capturing a subset of a room with a LiDAR scanner turned out to be less realistic than using the full ceiling.

5.2.2. New LiDAR database and user input

The separate rooms we extracted from the database we captured ourselves together with Moseg, are shown in figure 5.8. The 10 matching database point clouds

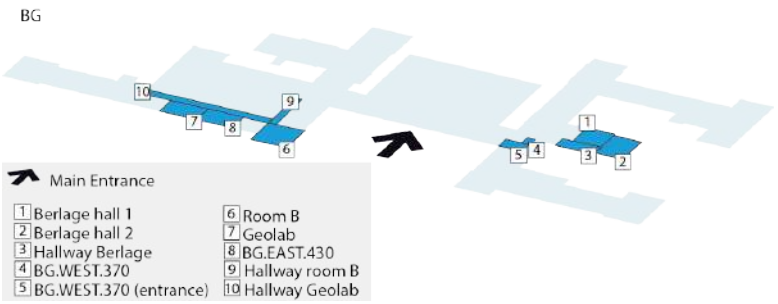


Figure 5.8: The rooms we extracted from the point cloud we captured ourselves with the help of Moseg Technologies.

are shown in appendix A.4. We not only captured a point cloud database, but we also experimented with capturing different LiDAR point clouds as a different type of user input. A more elaborate explanation on this is provided below.

LiDAR Scanner database and DIM input

For this experiment, we captured videos of all the rooms that we selected from the LiDAR scanner database. On these videos, we applied DIM by extracting frames in Pix4D. Because we let Pix4D determine the camera parameters based on the input we provided, the point clouds were not parallel to the x,y-plane. For this reason, manual tilting was required. The point clouds from the Berlage halls and the Geolab are captured at a later moment in time, so in terms of cleaning, more is left to the automatic processing here. The other point clouds also have been manually clipped to a bounding box. All DIM input point clouds an their matching LiDAR database components are shown in appendix A.4.

LiDAR Scanner database and LiDAR input

When capturing user input with the LiDAR scanner, the scanning itself is quick. Because of this, it is not necessary, and even unlikely, for the user to only capture a subset of a room. For this reason, for every user input a full room is tested against the database during this experiment.

Because we started several of our measurements in room B, we have many user input point clouds that did not focus on room B, but that did provide some results for this room. The same is the case for the hallway to the Geolab and the hallway to room B. For these rooms, we have some point clouds with less uniform density. Added to this, we took measurements in a quick way from every room. In general, these quick measurements did not show lots of differences in point density and quality. Rooms that were also present in our original laser database we extracted from the point cloud from CGI, are also used as additional LiDAR input. In the end, this resulted in a total of 32 rooms for LiDAR user input. The LiDAR scanner input and the matching database point clouds are shown in appendix A.4.

5.3. Pre-processing steps

It is required to automate the pre-processing of the user point cloud for a fully functioning final application. However, note that we did not manage to completely automate all the necessary steps.

One of the major manual pre-processing steps involves the removal of wall points. It would theoretically be possible to automatically cut away wall points based on the bounding box of the point cloud. However, it will be challenging to decide how big the buffer for point removal should be. More refined automation of wall point removal would require some form of classification. Currently, we lack the semantics to perform such a classification.
However, for testing purposes, we still decided on removing the wall points, because the amount of walls captured in the user input varied strongly based on the room characteristics. Smaller rooms such as hallways contained lots of wall parts, but larger rooms sometimes contained no wall parts at all. Wall parts become a significant part of the histogram, it would not be good for the matching to leave them in in all cases. Furthermore, more points in the point cloud would elongate the processing time, which is undesirable. For future applications, it would be nice to look into which parts of the wall would be feasible to keep, because wall point can provide a nice boundary description for the ceiling, and whether or not complete removal of all wall points is possible and desirable.

Not all rooms we tested contain clearly separated wall points. The Berlage hall 2 for example contains very low hanging curtains. The curtain rails are part of the unique ceiling signature of that room, so we did not want to disregard them. However, incorporating all the points of the curtains would lead to unrealistic results, because a potential user will never fully capture them when focusing on the ceiling. As an in between solution, we decided to only keep the upper part of the curtains. This is displayed in figure 5.9. When looking at a possible corresponding user input, displayed in figure 5.10, you can see that this matches the way the user would possibly capture the curtains. It would be challenging to automate keeping only part of the curtains in the user input. However, it would be possible to estimate how much of the wall and how much of features such as curtains, are likely to be captured by the user, and match our database to this.

5.3.1. Required pre-processing DIM input

Manual pre-processing

As mentioned before, the point clouds acquired through DIM have different orientation and scale than the ones from our database. Among others, this depends on how the users hold their camera.
In our approach, we did not yet deal with tilt automatically. For this reason, we manually removed tilt by orienting the point clouds in Cloud Compare. It would also be possible to manually align the point cloud from user input to the one from the database through rotation, scaling and translation. This can be done by selecting four corresponding point pairs between the two point clouds.

Figure 5.9: Example of keeping only the upper part of the curtains of Berlage hall 2.

Figure 5.10: Example of how much of the curtains of Berlage hall 2 could be incorporated in the user input.

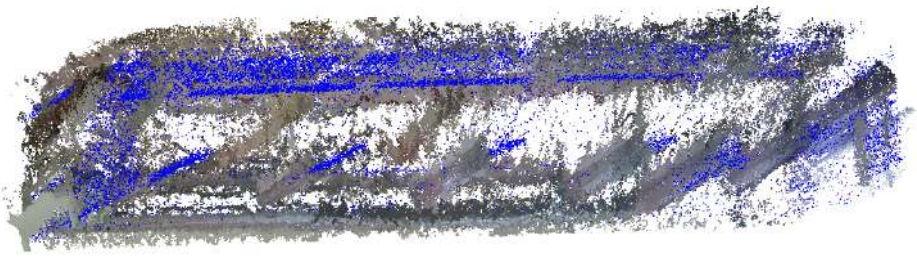


Figure 5.11: Noise in the DIM point cloud becomes visible after manually aligning them to the database point cloud.

By manually aligning point clouds, some artefacts in the DIM point cloud becomes evident. An example of this is displayed in figure 5.11, here we see that the same lamp is generated multiple times. Such repeating elements in the ceiling can cause problems for the DIM algorithm.

For the signature matching as described in subsection 6.3.2, a user point cloud will be automatically aligned to all the database point clouds, and the best fit will be kept.

Large chunks of noise are manually removed. An example of this is shown in figure 5.12. Automatically remove large chunks of noise can be challenging. Noise removal algorithms generally assume that noisy points are outliers [Ledoux et al., 2018]. When setting the threshold for noise removal to be large enough to also remove the large chunks of noise, important details of the point clouds are removed as well.

If possible, parts of the user point cloud that turned out to be warped, most likely due to lens distortions, are also manually removed.

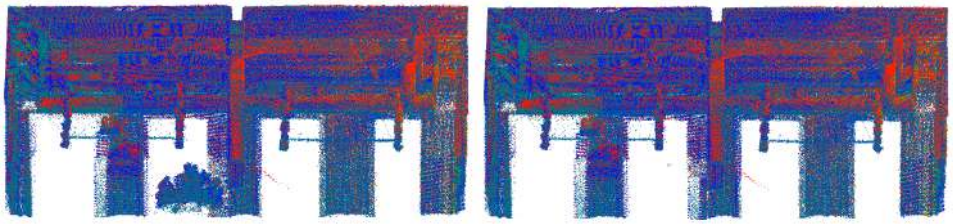


Figure 5.12: Example of manual removal of large chunks of noise.

Figure 5.13: Example of the results of manual wall points removal. From left to right: a full point cloud measurement, clipping rooms as separate subsets and then extracting the ceiling without wall points.

Automated pre-processing

The automated pre-processing that is applied on the DIM point clouds involves noise removal and voxel down sampling to speed-up the fingerprinting process and to acquire an input point cloud of more compatible point distribution when compared to the database. An in-depth explanation of the methodology related to the automatic pre-processing of the DIM input point clouds is provided in subsection 6.2.2.

5.3.2. Required pre-processing LiDAR input

For the LiDAR point clouds, only manual pre-processing is preformed. For the LiDAR input there is no tilt present. Handling tilt is therefore not necessary. The point clouds are already clean and regular, meaning that the remaining pre-processing steps, voxel down sampling and noise removal, are not applied. For the database point clouds, manual pre-processing does not introduce any problems. However, for the user input, the pre-processing steps still require automation.

In general, the pre-processing of LiDAR process involves clipping subsets containing separate rooms. From these subsets, we will extract the ceiling points. This is shown in figure 5.13. We realise that by cutting up our original point cloud, we are disregarding useful topology information between the different rooms. This data can be useful for indoor navigation and also as a support for our indoor localisation approach when incorporating methods such as dead reckoning. For future applications it would be good to take this additional information into consideration.

Something that has to be taken into consideration when capturing a laser database user input, is that separate rooms that are not bounded by walls will be captured connected to each other. For now we split them manually. In an actual application, this splitting should be automated or not performed at all, because using the adjacency information between separate rooms combined with dead reckoning, could

lead to more efficient, more accurate localisation.

The LiDAR scanning results in more surrounding points. During our scanning process, it seemed impossible to just capture the ceiling of a room without capturing anything of the walls and floor. For both the LiDAR user input as the database, we had to manually extract the ceiling. This can for example be be automated based on classification of the point cloud.

5.4. Quality assessment through confusion matrix

The information in this section is based on the contents of the lectures of the course Geo Datasets and Quality [Lemmens, 2019]. One of the things covered in this course was the confusion matrix. This is a concept which can be used to assess the performance of a classification method. Within our research, we will use such a confusion matrix to assess the performance of our indoor location solution. An example of a confusion matrix is given in figure 5.14. In this particular example, 350 samples for three rooms, called A, B and C, are tested. For the actual quality assessment, the full database from the rooms as displayed in figure 5.8 is tested. Something to note from this use of a confusion matrix is that the amount of samples is limited (32 samples for 10 rooms in case of the LiDAR user input and 20 samples for 10 rooms for the DIM input. This is less ideal, for a more solid quality indication, more samples need to be tested.

As for our example from figure 5.14, room A was correctly classified 100 times (True Positive (TP)). 50 Times room B was classified as room A (False Positive (FP)). This means that for room B, 50 values are False Negative (FN). True Negative (TN) Means that values that are not room A in reality, that are not classified as room A.

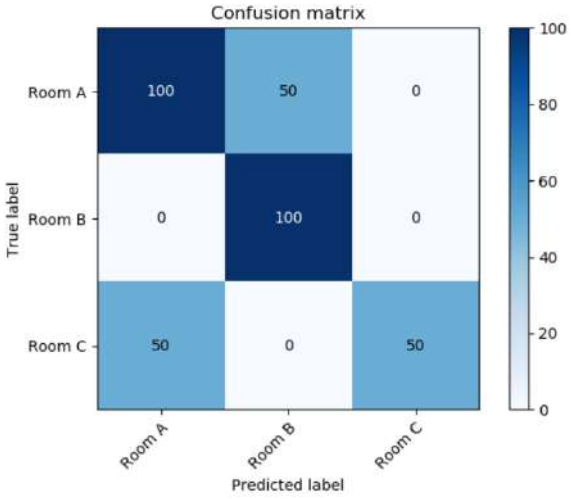


Figure 5.14: Example of a confusion matrix based on code from Scikit-learn [Pedregosa et al., 2011].

5.5. Combining separate steps

In a final application, all separate components of our methodology need to be combined. For now, we did not yet manage to combine all our separate steps in a complete application. This means that to get combined results, we manually used the output files from one step, as input for the next and so on.
Furthermore, we only did some limited testing on how to combine different fingerprinting outputs due to time constraints. For future implementations this should be investigated further.

5.6. Software resources

During our project, we use several software packages.

- Pix4D is used for creating a point cloud from user input videos.[3]
- Cloud Compare is used for manual point cloud editing and visualisation.[4]
- The Python programming language is used to write the code of this project. Within Python, we used the following major libraries:
  - NumPy[5]: To create histograms and to perform efficient calculations using arrays.
  - SciPy[6]: For the distance measures to compare two histograms.
  - OpenCV[7]: To create and compare histograms.
  - Open3D[8]: To perform noise removal and for the feature matching.
  - PyCPD[9]: To perform rigid alignment.
  - Sklearn[10]: To create confusion matrices and quality indicators for the results.

[3] https://www.pix4d.com/
[4] https://www.danielgm.net/cc/
[5] https://www.numpy.org/
[6] https://www.scipy.org/
[7] https://opencv.org/
[8] http://www.open3d.org/
[9] https://github.com/siavashk/pycpd
[10] https://scikit-learn.org

6
Methodology

In this chapter, each section will be used to explain a specific part of our methodology, as displayed in figure 4.1.

6.1. Data collection

To be able to perform localisation using a fingerprinting approach, data is needed. For the localisation approach described in this research, a LiDAR point cloud is collected. The separate rooms of this point cloud are stored as a database. Both LiDAR and DIM point clouds will be used as user input to match to this database in a fingerprinting approach. Decisions made about the data collection process are specified in section 5.1.

6.1.1. From input videos to DIM point cloud

DIM User input point clouds are extracted by implementing a DIM approach on overlapping frames extracted from an input video. These input videos of the ceiling have a duration from fifteen seconds to less than two minutes without changing the focal length. They are taken with a small angle. This was done by walking in a straight line forward, turning around and walking the same way back. In this way, the ceiling is captured from different angles. This could make it easier to reconstruct the ceiling from the video. With a walking speed of approximately two seconds per metre, enough overlapping frames could be extracted from the video.

Once the data was collected for different ceilings, the videos are processed in Pix4D with the standard settings for 3D models. Decisions made on the DIM process are specified in section 5.1.2.

6.2. Point cloud pre-processing

The localisation approach described in this report requires ceiling point extraction of the separate rooms. Further more, the user input point cloud generated from DIM is of different quality and has different point density than the ones acquired through LiDAR scanning. In this section, possibilities for pre-processing are discussed to make the point clouds more compatible. This way, a more fair and more efficient comparison can be made during the fingerprinting process.

In the sections that follow, manual and automated implemented processes will be discussed in more detail.

6.2.1. Manual pre-processing

The LiDAR point clouds, both database and user input, are manually divided into separate rooms. For both the LiDAR and the DIM point clouds, the ceilings points

are manually extracted by slicing away the wall, and floor points. For the DIM point clouds, additional manual pre-processing as described in section 5.3, is applied. Within these steps, tilt was removed in order to align the ceiling of the input point cloud to the ceiling of the database.

6.2.2. Automated pre-processing

This section is structured in three parts. Firstly, attempts to handle tilt in the DIM input will be discussed, then down sampling and noise removal will be explained.

Handling tilt

The z-values of the user point cloud can be biased, because the ceiling may be tilted and will therefore not be parallel to the x,y-plane. To compare unbiased z-values of the user point cloud to the z-values of the database point cloud, both ceilings must align with the x,y-plane. The suggested method described below is not incorporated in the final workflow, instead the tilt is handled such as described in section 5.3.

To compare the z-values without a bias, the 3D camera positions and orientations should be provided to the DIM software in the beginning, instead of correcting the tilt afterwards.
Correcting for the tilt afterwards causes some difficulties. One example is that the ceiling point clouds do not always have a homogeneous distribution of points along to ceiling. The point clouds can contain wall points and objects beneath the ceiling like lamps, that ensure that the points in the point cloud are not homogeneous distributed. This is problematic when fitting a plane through a 3D point cloud, which uses the root mean square, or aligning the ceiling point cloud to a generated x,y-plane. Both methods are unsuitable for removing the z-bias, because they minimise the distances between the points and the generated plane. This makes these methods dependent on a homogeneous point distribution.
Another example of correcting the tilt afterwards, are methods that generate or select a plane that would fit the ceiling in the point cloud. These methods require an indication on whether or not the selected or created plane is the ceiling. When the ceiling is captured, the cameras are oriented towards the ceiling. Therefore, it may be assumed that the z-orientation of the camera coordinate systems would point towards the ceiling. This is however not always the case as can be seen in picture 6.1 and 6.2. The reason behind the reconstruction is not fully explored, but it can be concluded that the z-orientation is unsuitable to give an indication that the right plane is created or selected.

To unravel if providing the 3D camera positions and orientations to the software, to prevent tilt from happening, could work, a test setup was constructed. The results are shown in the result section 7.1.1.

- Images: 10 images were taken every 10 centimetre along a table by laying a phone upside down on the edge of the table. The image plane is thus parallel to the ceiling.

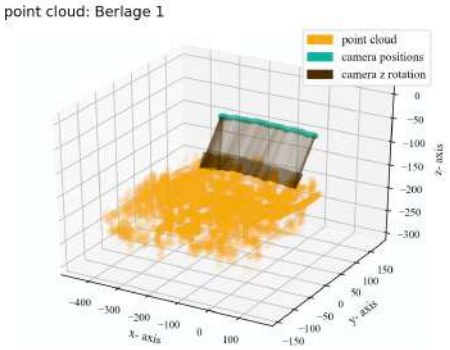


Figure 6.1: Point cloud extracted based on 138 video frames while filming for 26 seconds. Room: Berlage 1 in the Architecture building. Software used: Pix4D with the standard setting 3D Models.

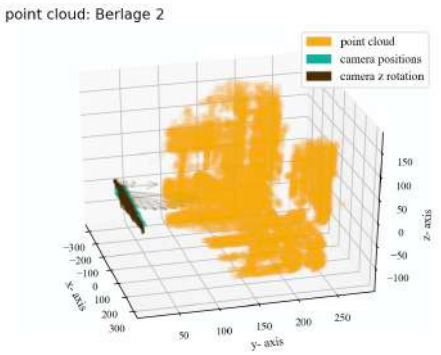


Figure 6.2: Point cloud extracted based on 228 video frames while filming for 32 seconds. Room: Berlage 2 in the Architecture building. Software used: Pix4D with the standard setting 3D Models.

- local coordinates and orientation: The camera positions (x, y, z coordinates) beginning at (0, 0, 0) and the orientations (ω: 160 degrees, ϕ: 0 degrees, κ: 90 degrees) will be provided before the reconstruction to Pix4D.

Voxel down sampling

Down sampling is performed using a voxel grid, where every voxel is represented by the centroid of the points that are contained within its boundaries. For this, two things are needed; A structure that will efficiently store the voxel information and a voxel size.

Voxel information relates to the index of the voxel, which is derived using the distance of each point included in the point cloud from the point cloud's minimum values in the x, y and z direction. Then this distance is divided by the chosen voxel size providing the index of the voxel [Pirouz, 2016]. This process is done for every point in the point cloud and then stored in a Python dictionary. The downside of this method is that empty space is not stored. The implementation pipeline is shown in algorithm 6.1 and algorithm 6.2. However, for this implementation empty space is not needed. The use of empty space is discussed in section 7, where the amount of empty space within the bounding box of the ceiling point cloud is used for noise removal.

The voxel size parameter was selected based on experimentation.To meet the goal of average down sampling i.e. generation of a grid that approximates all ceiling features as good as possible the voxel size permitted threshold can vary depending on the database. In our case a threshold was set between 5-10 units(since scale is unknown it is not possible to specify the metric system). The factors that directed our final choice were scale, point density, time performance, level of detail and the calculated error in the laser scanner point clouds. These factors are explained further below:

Algorithm 6.1: Creation of voxel grid

Input: A pre-processed user input point cloud generated from a mobile phone camera in xyz form. The input is inserted in the code as an array of shape (3,N) where N:number of points in the point cloud.

Output: A dictionary representing the voxel grid in the form: voxel_grid = {"voxel_index":centroid coordinates}.

```
bounding_box = min_x,min_y,min_z voxel_size
voxel_grid = {}
for point in point cloud do
    voxel_x = (point_x - min_x) / voxel_size
    voxel_y = (point_y - min_y) / voxel_size
    voxel_z = (point_z - min_z) / voxel_size
    voxel_idx = voxel_x, voxel_y, voxel_z
    if voxel_idx in voxel_grid
    centimetrevoxel_grid[voxel_idx].append(point)
    else
    centimetrevoxel_grid[voxel_idx] = [point]
```

Algorithm 6.2: Average down sampling

Input: A dictionary representing the voxel grid in the form: voxel_grid = {"voxel_idx":[points]}.

Output: A dictionary containing the centroid of the points' distribution within a voxel for all voxels in the form: voxel_grid = {"voxel_idx":[centroid]}.

```
voxel_avg = {}
for voxel_idx in voxel_grid do
    centroid_x = avg_x(voxel_grid[voxel_idx])
    centroid_y = avg_y(voxel_grid[voxel_idx])
    centroid_z = avg_z(voxel_grid[voxel_idx])
    centroid = centroid_x, centroid_y, centroid_z
    voxel_avg[voxel_idx] = centroid
```

- Scale: The difference in scale between the LiDAR database and the DIM database, explain why it is not possible to choose the same voxel size for both databases, even after approximating the point density.

- Point density: For the point clouds generated from DIM depending on the different features that formulate a ceiling and the user recording of the ceiling information the density of the points within a point cloud varied. Considering that all recordings were captured within a given time frame, as described in section 6.1 and the most stable (no significant changes in camera parameters) video frames were chosen for DIM, the different densities are more dependant on the type and variability of the ceilings' formations as well as the incorporated noise. The occurrence of these formations, the type of features that comprise them and the amount of noise are what causes the different point densities not only between the different point clouds but also within the same point cloud. An example of the difference in point density within a point cloud is shown in figures 6.4, 6.3 .

Figure 6.3: Ceiling features from Berlage hall 2.

Figure 6.4: Zoomed in lamp element from Berlage hall 2.

Within this voxel down sampling method, the fluctuations in point density were approximated using the centroid of the points distribution within a voxel and an optimal voxel size. For this reason, we experimented with different voxel sizes with the aim of finding an optimal value that would improve the time performance of the matching methods and represent as accurately as possible the initial point cloud after the implemented point reduction. Examples of different voxel sizes for "BG.West.370" are shown in figures 6.5, 6.6, 6.7.

- Time performance: From experimentation processing time for the voxel averaging method varied between 1 - 8 seconds and it depended on the number of points of the input point cloud, the voxel size and the density of the points. Specifically, the smaller the voxel size the more time was needed for the down sampling to complete. However that was not the case for the number of points. In general, the more points of a point cloud the longer it was needed but for some cases with difference in point number time performance

Figure 6.5: Generated point cloud for voxel size: 5

Figure 6.6: Generated point cloud for voxel size: 7

Figure 6.7: Generated point cloud for voxel size: 10

was disanalogous, which we attributed to the different point densities implying larger number of complex features or more noise included (See chart in figure 6.8). In figures 6.9 and 6.10, it is shown how the difference in point density due to noise (figure 6.9) can affect processing time.

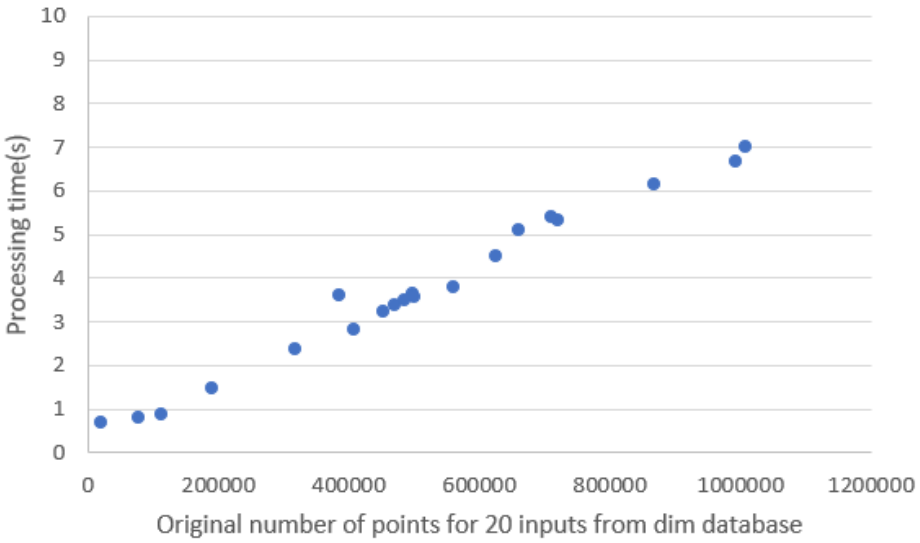


Figure 6.8: Chart showing the relation between processing time for average down sampling and number of points for input point clouds (DIM database used).

- Level of detail: The chosen voxel size needed to have a reference to the size of details contained in the ceiling. For the point clouds from DIM, the metric system was unknown so the voxel size was derived solely from experimentation. However, for the laser scanner point cloud it was known that distances were measured in metres and thus the selected voxel size would have to produce a grid that represents accurately the chosen minimum level of detail. For this, the lamps were used as reference and as minimum 0.1 i.e. 10 centimetre was chosen.
- Calculated error from laser scanner: The used laser scanner point clouds in-

Figure 6.9: Point cloud with more points,lower density and more noise.

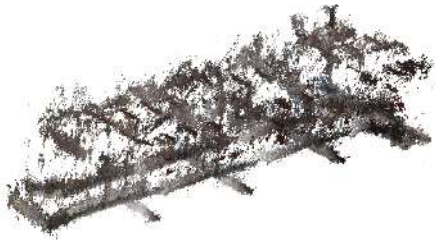


Figure 6.10: Point cloud with less points,higher density and less noise.

clude 9 % of the total generated points and the incorporated error is measured to be between 2 – 3 centimetre. Based on this, selecting a minimum 10 centimetre voxel size is considered to be outside the error threshold limitation.

The chosen voxel sizes for the selected matching methods are:

1. LiDAR database and user input: 0.1, since points from the LiDAR database are well distributed for all ceiling features.
2. User input from DIM: 10, since the time performance is higher and there is features are adequately represented.

After constructing the grid, down sampling was performed by computing the centroid of all points within the boundaries of the voxel. The results of the averaging down sampling experiments for voxel size 10 are shown in appendix A.1.

Noise removal

It is necessary to remove the noisy points to improve the quality of the results. For this project, the method used to remove the noise is a combination of a radius-based method and a statistical-based method.

1. The radius-based outliers removal aims to remove the points that have few neighbours which means less than a threshold in a given sphere. To apply this, two parameters have to be defined, one is the minimum amount of points that the sphere should contain and the another one is the radius of the sphere that will be used for counting the neighbours.
   (a) number of points: 10
   (b) radius of sphere: 5
2. The statistical-based method removes points that are further away from their neighbours compared to the average for the point cloud. This method also

has two parameters, one is the number of points that are taken into account to calculate the average distance for a given points. The other one is the threshold level based on the standard deviation of average distance across the point cloud. The lower this parameter the more aggressive the filter will be.

(a) number of points: 20

(b) ratio threshold: 2.0

The noise removal process attempt to remove as much outliers as possible and at the same time less important points like the points on the ceiling objects would be omitted. Based on this objective of noise removal, the parameters for both methods are selected from trials and error.

The aforementioned noise removal processing was implemented using the Open3D library [Zhou et al., 2018].

Additionally, two more filters were implemented experimentally. One of the filters included comparison with the amount of empty space, and thus required a voxel grid that incorporates the parts of the ceiling that do not contain any points. For this a Numpy 3D array was constructed. Note that this method of constructing a grid can be computationally intensive and thus risky to include in the final workflow. The constructed filters are described below:

1. Noise removal based on the amount of layer occupancy, indicated by the percentage of non empty voxels in the total number of voxels included in a layer. For this, the percentage of non empty voxels per voxel grid layer (a layer includes all voxels with same height) was computed and compared to a threshold. The threshold was defined as 0.3 of the total grid occupancy.

2. Noise removal based on the number of points per voxel. This filtering removes all voxels that contain a number of points that rest below a given threshold. This threshold is defined as the average number of points that a voxel contains within a layer reduced by one standard deviation.The standard deviation refers to the standard deviation of the number of points and is computed for the distribution of the number of points for an incrementally.

In both cases, the threshold was a determined by experimentation. This filtering down samples drastically the point clouds by keeping all important ceiling features. However it does not take into account smaller details. This is shown in Figures 6.12, 6.14. More analytical results of the additional filtering are shown in appendix A.1.

6.3. Fingerprinting

To perform localisation, a database point cloud needs to be matched to a user input point cloud. For this, a so-called fingerprinting approach is used. A general expla-

Figure 6.11: Original point cloud of room BG.EAST.430 from user input database

Figure 6.12: Down sampled point cloud of room BG.EAST.430, part of user input pointcloud.

Figure 6.13: Original point cloud of Berlage hall 2 from user input database.

Figure 6.14: Down sampled point cloud of Berlage hall 2, part of user input pointcloud.

nation of fingerprinting is provided in section 4.3. Within this research, histogram and feature matching are the two fingerprinting options considered.

6.3.1. Histogram matching

One possible way to perform signature matching, is by extracting histograms from the different rooms in the database and the user point cloud. To create histograms from the normalised height values (the z-coordinate), the height values of a single point cloud should be compatible. In other words, a planar ceiling should be parallel to the x,y-plane as described in subsection 6.2.2. Because of the use of normalised height values, the histogram approach is scale invariant. Except from the z-value, all other spatial characteristics of the points are omitted, resulting in a simple, one-dimensional comparison between the different signatures. To make this comparison more efficient, some pre-processing of both the input and the database point cloud is required.

To test the histogram matching approach, manual pre-processing as described in subsection 5.3 is applied. In theory, regularising the density of the point distribution within the point cloud is not strictly necessary, because it is possible to normalise to occurrence frequency listed of the histogram. However, to speed up to processing and to be able to make the user input histogram and the laser database histogram

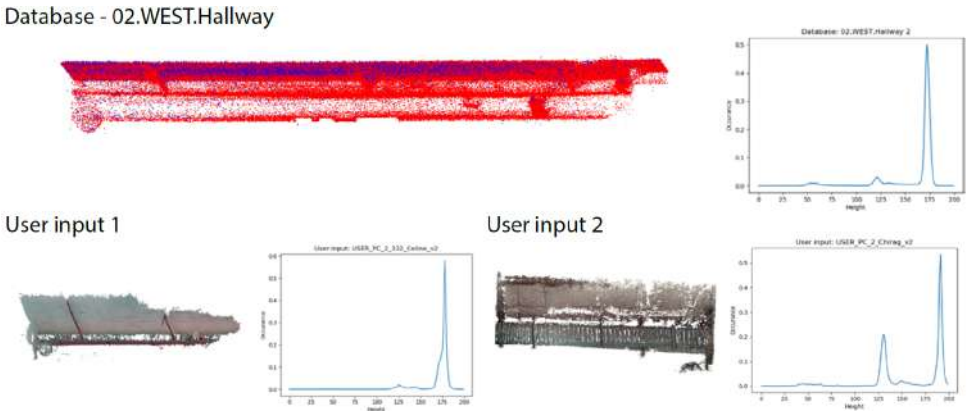


Figure 6.15: Histogram comparison between database and user input seems to be successful.

more compatible, the voxel down sampling and noise removal as described in section 6.2 are applied in the case of DIM user input files.

The main input parameter for histogram creation, is the amount of bins. If this amount is too small, the different histograms might not be distinct enough, and loss of detail occurs. If the amount of bins is too high, the processing will take too long and noise, or less important values, will get too much weighing. The Freedman–Diaconis rule as displayed in equation 6.3.1 and 6.3.2 [Freedman and Diaconis] is used to determine a suitable amount of bins. A big advantage of this method is that it is very robust and thus insensitive to outliers. It uses the InterQuartile Range (IQR), which is a measure of statistical dispersion, listing the difference between the bottom 25 % and the top 75 % of the data set.

$$h = 2 * IQR * n^{-\frac{1}{3}} \tag{6.3.1}$$

$$n \quad bins = \frac{max(z) - min(z)}{h(roundedup)} \tag{6.3.2}$$

The optimal number of bins depends on the number of data points, which is not fair when comparing histograms created by different point clouds. A more uniform point density between input and database point clouds could improve this. For now, the number of bins is set based on the user input. Thanks to the voxel down sampling described in section 6.2, the point density will become more uniform.

When attempting histogram generation, it was estimated whether or not the different histograms in the database might be distinct enough. For this estimation, the database histograms are visually compared against the user input histograms. For some cases, the initial visual inspection of the histograms indicated that a match between the database histogram and the user histogram would be likely. An example of such a case is displayed in figure 6.15.

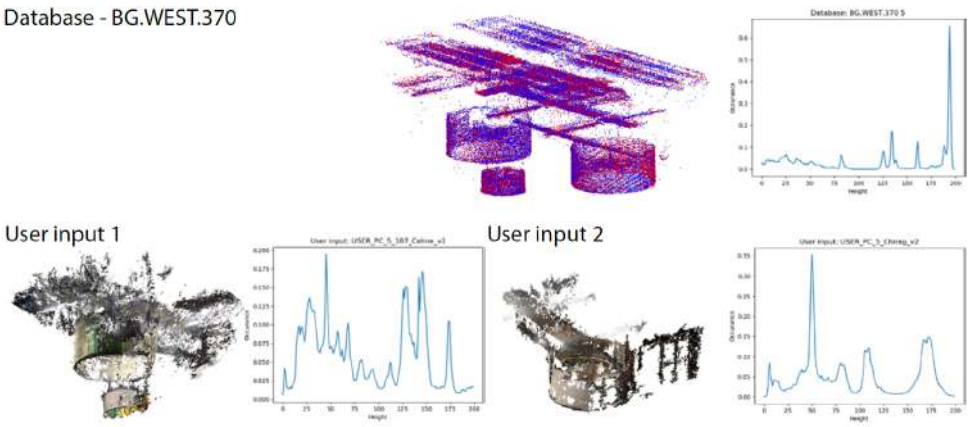


Figure 6.16: Histogram comparison between database and user input seems to be less successful.

There are also a few cases that visually do not seem to match as good. An example of this is displayed in figure 6.16. The full result of the initial comparison is provided in appendix A.2.
The general pipeline of the histogram comparison can be described with the pseudo code listed in algorithm 6.3.
A weighing is applied to be able to take the difference in occurrence value at a certain height into account. Additionally, some experiments with taking not all bins into account to filter for outliers are performed. Here it was considered to keep the middle n bins and omit the outer ones. This was not successful because outer bins are not necessarily outliers. One attempt would be to filter outliers based on weighing and omit the bins with the lowest weights. However, this is less necessary because the weighing factor as listed in algorithm 6.3 is already implemented.

As can be concluded from the results shown in section 7.2, the so-called NumPy histogram approach is the most favourable option for the specific situations in which the localisation solution described in this report is tested. In the next paragraph, a more in depth explanation of this histogram comparison methodology will be provided. Next to this NumPy approach, an OpenCV and a SciPy histogram matching approach are also tested.

NumPy approach

The NumPy histogram comparison method involves creating a histogram using the NumPy library in Python. Here, the so-called `Density` parameter can be used for normalisation purposes. If this parameter is set to default ("False"), the result contains the number of samples in each bin. In other words, the result depends on the density of the input and database point cloud. If these densities are different, fair comparison might not be possible. For this reason, the `Density` parameter is set to "True". The result is the value of the probability density function at the bin, normalised such, that the integral, so the area, over the range is one [Oliphant,

Algorithm 6.3: Histogram localisation approach

Input: A pre-processed user input point cloud and a database containing signatures for different (sub)rooms in the form of histograms. The database is a dictionary in the form: db = {"room_name": room_histogram}.

Output: The most likely location of which the difference in signature between the input and the database is the smallest.

```
hist_user = create_hist(user_input)
hist_user_freq = create_hist_freq(user_input)
signature = {}
for room in database: do
    count_distance = 0
    for bin in database[room] do
        weighing = |hist_user_freq[room][bin] - hist_db_freq[room][bin]|
        count_distance += weighing * distance(hist_user[room][bin],
        hist_db[room][bin])
    signature.update(room: distance)
location = min(signature)
```

2006]. An example is shown in figure 6.17.

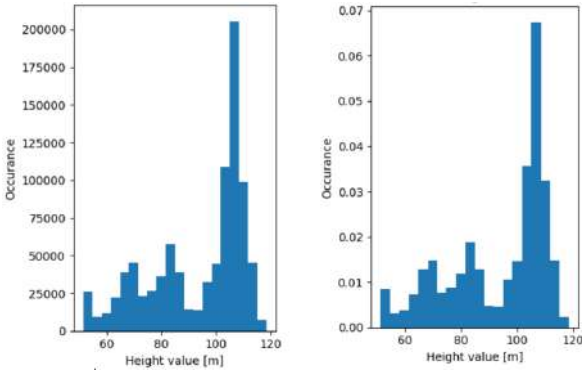


Figure 6.17: Original histogram of the user point cloud of the Orange hall (left) and the normalised alternative (right).

An alternative approach could be the z-score normalisation, which indicates how far away a certain score is from the mean of a normal distribution. The shape of the z-score distribution will be the same as the original distribution shape. For a normal distribution, one can compute the probability of obtaining a certain z-score. The equation that could be used for z-score computation, is shown in equation 6.3.3 [Lemmens, 2018]. However, within the data sets used for this research, the distributions will not always be normal, so this approach might not be suitable here.

$$z - score = \frac{x - \mu}{\sigma} \tag{6.3.3}$$

For the way NumPy generates their histograms, just setting the `Density` parameter to "True" is not sufficient to make histograms comparable between different scales. The user input point cloud is generated from pictures taken by different people with different cameras, meaning the origin of the local coordinate system in which the pictures are taken will be different if no additional information is provided during the DIM process. For this reason, the height values should also be normalised. This is done by dividing all height values by the median of said height values.

A common approach to compare histograms, is to look at the distance between corresponding bins [Ma et al.]. There are more advanced ways possible, which can be recommended for future research.

As can be seen in subsections 7.2.1, 7.2.2, 7.2.3 and 7.2.4, the `Chi-squared` distance [Gagunashvili, 2009], displayed in equation 6.3.4, turns out to provide the most favourable results. Two other popular methods, the `Jaccard` distance [Zhang et al., 2017], displayed in equation 6.3.5, and the `Euclidean` distance, displayed in equation 6.3.6 are also tested.

$$d_{Chi-squared} = \frac{(hist_1 - hist_2)^2}{2 * (hist_1 + hist_2)} \tag{6.3.4}$$

$$d_{Jaccard} = 1 - \frac{|hist_1 \cap hist_2|}{|hist_1 \cup hist_2|} \tag{6.3.5}$$

$$d_{Euclidean} = \sqrt{(hist_1 - hist_2)^2} \tag{6.3.6}$$

OpenCV approach

The OpenCV library in Python has four built-in histogram comparison measures. In this approach, the general idea remained similar to the pseudo code listed in algorithm 6.3, only the weighing is omitted.
As can be seen in subsections 7.2.1, 7.2.2, 7.2.3 and 7.2.4, the `correlation` OpenCV histogram comparison metric, displayed in equation 6.3.7, provided the best solution for the experiments conducted in this report. OpenCV has several other histogram comparison metrics which are also tested; The `Chi-squared` distance, a different variant than the one displayed in equation 6.3.5 is displayed in equation 6.3.8, the `intersection`, displayed in equation 6.3.9 and `Bhattacharyya` distance displayed in equation 6.3.10 [Bradski, 2000].

$$Correlation = \frac{\sum_{i=1}^{n}(hist_{1,i} - \bar{hist_1})(hist_{2,i} - \bar{hist_2})}{\sqrt{\sum_{i=1}^{n}(hist_{1,i} - \bar{hist_1})^2 \sum_{i=1}^{n}(hist_{2,i} - \bar{hist_2})^2}} \tag{6.3.7}$$

$$d_{Chi-squared} = \sum_{i=1}^{n} \frac{(hist_{1,i} - hist_{2,i})^2}{hist_{1,i}} \tag{6.3.8}$$

$$Intersection = \sum_{i=1}^{n} min(hist_{1,i}, hist_{2,i}) \tag{6.3.9}$$

$$d_{Bhattacharyya} = \sqrt{1 - \frac{1}{\sqrt{\bar{hist}_1 * \bar{hist}_2 * n^2}} \sum_{i=1}^{n} \sqrt{hist_{1,i} * hist_{2,i}}} \tag{6.3.10}$$

SciPy approach

The SciPy approach uses a NumPy histogram and the SciPy library. SciPy has 23 different measures to calculate the distance between two histograms [Jones et al., 2001]. It was challenging to find literature that assessed the suitability of all these measures for specific use cases. To be able to see which ones performed best for the data sets listed in appendix A.2, all methods that did not require additional, adaptive input parameters are tested. In the end this resulted in 19 different distance measures for 27 user input cases. As can be seen in subsections 7.2.1, 7.2.2, 7.2.3 and 7.2.4, the `Chebyshev` distance measure, displayed in equation 6.3.11, `Canberra` distance measure, displayed in equation 6.3.12, `Braycurtis` distance measure, displayed in equation 6.3.13, `Cityblock` distance measure, computes the Manhattan distance between points, `Minkowski` distance measure, displayed in equation 6.3.14, and `Euclidean` distance measure, displayed in equation 6.3.6, gave the most promising results. Because the results are the same for most cases, the `Braycurtis` distance was randomly selected.

$$d_{chebyshev} = max|hist_{1,i} - hist_{2,i}| \tag{6.3.11}$$

$$d_{canberra} = \sum_{i=1}^{n} \frac{|hist_{1,i} - hist_{2,i}|}{|hist_{1,i}| + |hist_{2,i}|} \tag{6.3.12}$$

$$d_{braycurtis} = \frac{\sum_{i=1}^{n} |hist_{1,i} - hist_{2,i}|}{\sum_{i=1}^{n} |hist_{1,i} + hist_{2,i}|} \tag{6.3.13}$$

$$d_{minkowski} = \sum_{i=1}^{n} |hist_{1,i} - hist_{2,i}| \tag{6.3.14}$$

The other distance measures seem to be less suited for the type of data and comparison conducted in this research. Some did just provide wrong results without any clear pattern. The `Cosine` distance often produced room BG.EAST.430, BG.WEST.370 and the lower part of the exhibition hall as a result. The `Correlation`

seemed to provide random results. This is interesting, because the correlation measure, as listed in equation 6.3.7 from the OpenCV approach performed well. After looking into the documentation, the SciPy correlation measure turned out to be constructed in a slightly different way. The `Yule`, `Hamming` (which is the same as `Matching`), `Dice`, `Kulsinski`, `Jaccard`, `Seuclidean`, `Rogerstanimoto`, `Russellrao`, `Sokalmichener` and `Sokalsneath` distance measures always provide the same results. Here the first room of the database is matched to be the most likely room, the second database room the second most likely and so on. This indicates that the SciPy implementation of these distance measures is not suited for the specific data set used during the specific experiments conducted during this research.

6.3.2. Feature matching

The feature matching based fingerprinting approach uses the geometric property of points to find similarity between input and database point clouds. The two point clouds have to be aligned in such a way that the local surfaces fit together with respect to geometry and topology. Thus a feature is a theoretical representation of local surfaces, mathematically defined by a point and normals of its n neighbours. The correspondences between two point clouds are detected by comparing the features or local surfaces based on certain criteria such as normal orientation, edge length of two points, distance between corresponding pairs. Transformation is done in a pairwise manner based on these key correspondences and alignment is done in a iterative manner to perform final matching.

Open3D is an open source library that has been used extensively for performing all major operations in this approach. The combination of C++ for back-end and Python for front end, is very helpful in fast processing with smooth work flows.[Zhou et al., 2018] This gives ease of coding and fast execution of data structures for algorithms.

Rigid alignment of input point clouds

Firstly, the two point clouds which are in different coordinate systems are roughly aligned to bring them into same space by iterative transformation based on minimising the distance between two point clouds. For this, PyCPD library was used which uses a pure NumPy implementation of the Coherent Point Drift (CPD) algorithm [Myronenko and Song, 2009]. The rigid transformation method is used to find the transformation that best aligns the two point clouds. Since, correspondence is not explicitly known, they are inferred through point proximity. In other words, points that are spatially close to each other correspond to one another. The solution to the rigid registration is known as the orthogonal Procrustes problem depicted in equation 6.3.15, where R and t are rotation and translation matrix, X and Y are the two point clouds.[Khallaghi, 2017]
The process takes the source and target point clouds as input and provide a rough transformation matrix based on number of iterations used to minimise the error.

$$argmin_{R,t}||X - RY - t||^2, \quad s.t \quad R^T R = I \tag{6.3.15}$$

Creating features using geometry property of points and their normal orientation

To create a signature from a point cloud, mathematical features are formulated using by first estimating normals of points based on neighbours and reorienting normals if possible. Then compute a FPFH feature for each point by using hybrid KD-tree on point clouds. The FPFH feature is a 33-dimensional vector that describes the local geometric property of a point. A nearest neighbour query in the 33-dimensional space can return points with similar local geometric structures. This has been depicted in figure 6.18

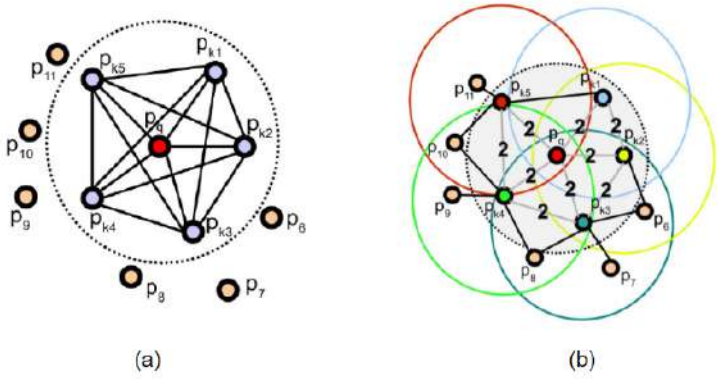


Figure 6.18: (a) "The influence region diagram for a FPFH inter-connected to its direct k-neighbours(blue) (b) Each query point (red) is connected only to its direct k-neighbours (enclosed by the grey circle). Each direct neighbour is connected to its own neighbours and the resulted histograms are weighted together with the histogram of the query point to form the FPFH. For example, the connections marked with two will contribute to the FPFH twice" [Rusu et al., 2009]

Finding key corresponding feature pairs between input and database

RANSAC is used to refine the global registration. In each RANSAC iteration, n random points are picked from the source point cloud. Their corresponding points in the target point cloud are detected by querying the nearest neighbour in the 33-dimensional FPFH feature space. A pruning step takes fast pruning algorithms to quickly reject false matches early. Only matches that pass the pruning step are used to compute a transformation, which is validated on the entire point cloud. Open3D provides three types of thresholds that helps in optimising output[Zhou et al., 2018]:

- `distance_threshold`: to check if the closeness is less than specified threshold.
- `edge_threshold`: to check if the lengths of any two arbitrary edges individually drawn from two point clouds are similar.

- `normal_threshold` to check if vertex normals of any correspondences are aligned within radian value for the threshold.

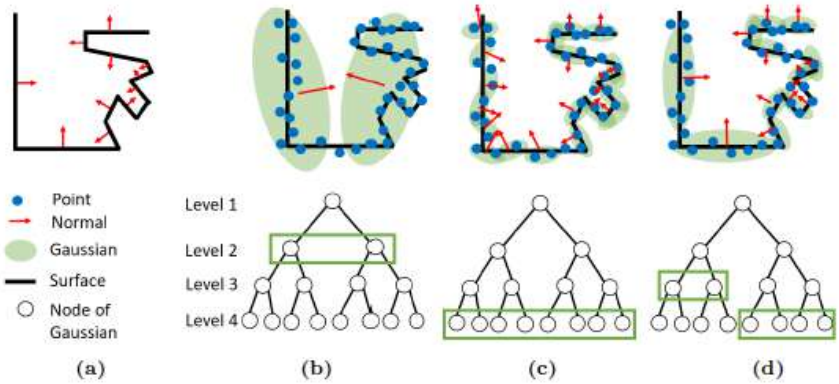


Figure 6.19: "Multi-Scale Representation using a Hierarchy of Gaussian Mixtures: Top-row shows identical geometries (black lines) and associated points (blue circles), which are represented by different levels of Gaussian models (green contour for 1 σ)(a) (Top) Ideal Normals (red arrows) on the surfaces, (b) Too coarse (only two Gaussians in Level 2): poor segmentation leads to incorrect normals, which will degrade the accuracy when registering points to model, (c) Too fine (using finest level of Gaussian models): over-segmentation leads to erroneous normals as sample noise overtakes real facet geometry (d) Adaptive multi-scale (Mixture of level 3 and level 4 models): point-to-model association can be much more robust when fidelity adaptively changes according to data distribution so that facets can be well-modelled given differing spatial frequencies and sampling densities." [Eckart et al., 2018]

Align point clouds by pairwise registration based on key correspondences

Once the key correspondences are known, the point clouds are pairwise registered. In each iteration takes n random samples from correspondonce pairs and a point-to-point transformation is computed that aligns those samples to their counterparts in a least-squares sense[Zhou et al., 2018]. Transformations are passed through multiple validation steps mentioned in section 6.3.2 to maximise the overlap of registered pairs. A core function in Open3D library for this purpose is `ransac registration based on feature matching`, which has an important hyperparameter; `RANSAC Convergence Criteria`. It provides the maximum permitted number of RANSAC iterations as well as the validation steps for selecting correspondences. The larger these values are, the more better the result is, however the time will increase for processing.[Zhou et al., 2018]

Improving alignment and refining parameters for optimum results

There are several optimising techniques available in Open3D library to improve alignment of point clouds such as Point-to-plane, Iterative Closest Point (ICP). Another method is pose optimisation using fine-grained registration and then, multi-way registration. This method takes advantage of colour values for pairwise local refinement and then, performs rigid transformation. [Zhou et al., 2018] Both these

refinement methods have been implemented but their results will not be considered in final analysis keeping in mind the project scope.
The parameters used in the whole work flow influence the level of optimisation during matching process. Formation of features of point clouds should performed such that the spatial coherence of points is respected and there is correct representation of point cloud and orientation of normal vectors with respect to real world. This has been further depicted in figure 6.19. The values of RANSAC registration used are initially inspired from [Choi et al., 2015] and then further testing was done to reach the optimum values. The main processing parameters have been depicted in figure 6.20 and algorithm for feature matching is described in algorithm 6.4.

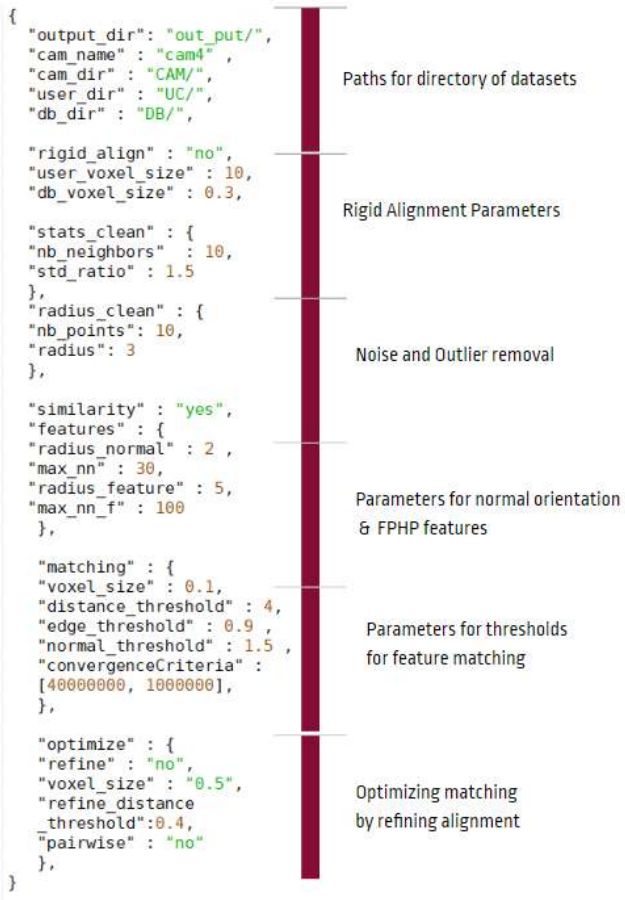


Figure 6.20: Input Parameters fixed for algorithm of feature matching approach. Modular approach enables working with different data sets and experiment with different methods

Algorithm 6.4: Feature Matching localisation

Input: A raw/pre-processed user input point cloud along with algorithm parameters and directory of all (sub)rooms. The database is a dictionary in the form: db = {"room_name": room_pointcloud}.

Output: The most likely location of which the correspondences between the input and the database are the largest based on fitness value and rmse error.

```
signature = {}
for db_room in database: do
    // if inputs have different coordinate systems, scale,
       noise and outliers
    statistical_outlier_removal(pcd,no_neighbors,std_ratio)
    radius_outlier_removal(pcd,no_neighbors, radius)
    do_rigid_alignment(input,room)
    for user_pcd and db_room_pcd : do
        pcd_down = voxel_down_sample(pcd, voxel_size)
        // Estimate normals of both point clouds using
           flann hybrid kdtree
        estimate_normals(pcd_down,
            KDTreeSearchParamHybrid(radius_normal,max_nn))
            // Reorient normals with respect to camera
            position if DIM PC
        orient_normals_towards_camera_location(pcd_down,cam_pos)
            // Compute FPFH feature using radius and max
            nearest neighbors
        pcd_fpfh = compute_fpfh_feature(pcd_down,
         KDTreeSearchParamHybrid(radius_feature, max_nn_f))
    // Find key corresponding FPFH features between 2
       point clouds based on thresholds and perform
       transformation using RANSACConvergenceCriteria
    fitness, rmse = registration_ransac_based_on_feature_matching(
    user_down, db_room_down, user_fpfh, db_room_fpfh,
    TransformationEstimationPointToPoint(True), no_corresponding_pairs
    [CorrespondenceCheckerBasedOnEdgeLength,Distance and Normal],
    RANSACConvergenceCriteria)
    signature.update(room: fitness,rmse)
Sort the signatures based on highest fitness value and least rmse
location = top1(signatures)
```

6.3.3. Combined fingerprinting methods

On their own, possible methods to extract a signature from a point cloud might not be accurate enough for indoor localisation. However, there is a possibility to combine the outcome of multiple methods. By doing this, the separate methods could support each other if necessary.
The general pipeline of of the method to combine two fingerprinting outputs can be described with the pseudo code listed in algorithm 6.5.

Algorithm 6.5: Combine output of two different fingerprinting methods.

Input: Two CSV files. Each should contain a column with the name of the actual file that was tested (for our own testing purposes) and the top x order in which the fingerprinting approach classified the result. Here x is the length of the database.
Output: CSV file containing the top n most likely locations based on the combined fingerprints.

```
1  location1 = read_csv("Fingerprint1.csv", column=1)
2  location2 = read_csv("Fingerprint2.csv", column=1)
3  file1 = read_csv("Fingerprint1.csv" , column=2) // (==file2)
4  length_database = 10
5  top_n = 5
   // The list containing the different weights should have
      the same length as the database.
6  weighing = [2^10, 2^9, 2^8, 2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1]
   // For this pseudocode, it is assumed that only one
      input file is tested. (In our code, it works for a
      list of multiple input files, because this is useful
      for testing purposes.):
7  combined = {}
8  w = 0
9  while w < length_database: do
10     for i in range(length(weighing)) do
11        if location1[i] == location2[w]:
12        combined.update(location1[i]: weighing[w] + weighing[i])
13     w += 1
14 location = sorted(combined, key=combined.get, reverse=True)[:top_n]
```

There are several ways to combine the weights of the different locations at which a matching fingerprint occurs. The example in algorithm 6.5 adds up the weights. As can be seen in the results described in subsection 7.4.1, this method performed best during the testing.
It would also be possible to add the different weights, or to use a least squares solution such as for example displayed in equation 6.3.16. For this example, the location with the lowest "LS" value is the most likely. Different scaling between

the weights would also be a possibility. For future implementations this should be investigated further.

$$LS = (index1 - index2)^2 \tag{6.3.16}$$

Another concept that could be implemented to combine two fingerprinting results, involves taking the top n output of the feature matching and provided only these possibilities as a database for the histogram approach. This way the histogram approach has less options to pick from and is more likely to be correct. However, this method has one major disadvantage; The correct location should be guaranteed to be within the top n results of the feature matching. If this is not the case, combining the two methods in this way cannot lead to correct localisation. The same goes for an implementation in which the top n results of the histogram approach are provided as input to the feature matching approach.

The approach described in algorithm 6.5 will also be used to combine the results of the three different histogram approaches. Here, first the OpenCV and the SciPy approach are combined. After this, the resulting classification is combined with the NumPy approach.

6.4. Quality assessment through confusion matrix

There are several measures possible to determine the quality of a localisation method. One of the more common ways would be to calculate the accuracy of the method by dividing the number of correctly classified rooms by the total number of rooms that are tested. In this section, the values from a confusion matrix are used as a measure to indicate the quality of the fingerprinting approaches described in this report. More background information about the confusion matrix is provided in section 5.4.

Several quality indicators can be extracted from the confusion matrix. The accuracy is shown in equation 7.2.1 and the κ-coefficient, which is a measure eliminating the random chance component for correct classification of the accuracy, is shown in equation 7.2.2. Precision (equation 7.2.3), recall (equation 7.2.4) and the F1-Score (equation 7.2.5) are also given.

$$Accuracy = \frac{TP}{TP + TN + FP + FN} \tag{7.2.1}$$

$$\kappa - coefficient = \frac{N * \sum_{i=1}^{k} n_{ii} - \sum_{i=1}^{k} n_{i+} n_{+i}}{N^2 - \sum_{i=1}^{k} n_{i+} n_{+i}} \tag{7.2.2}$$

In equation 7.2.2, N describes the total amount of classified values, n_{ii} describes the on-diagonal values of confusion matrix (TP). n_{i+} Describes the summed values per row and n_{+i} describes the summed values per column.

$$Precision = \frac{TP}{TP + FP} \tag{7.2.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{7.2.4}$$

Notice that one could "cheat" by using the recall method. If all values are marked positive, then there will be no false negatives and the value for recall will be high despite possible incorrect classifications.

$$F1 - Score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{7.2.5}$$

7
Results

In this chapter, the results of the separate steps of the work flow are displayed. In the end the results from combining these steps are displayed.

7.1. Results point cloud pre-processing

In the sections below, the results of the following pre-processing steps will be shown: handling tilt, voxel down sampling and removing noise.

7.1.1. Results handling tilt

The results of handling tilt will be described below. It can be seen that, by providing the local coordinates and orientation on forehand, the ceiling will be more in line with the x, y-plane. This can be seen in figures 7.1 and 7.2. The left image, where the local coordinates and orientation were provided on forehand to the DIM software, is more in line with the x, y-plane than the right image.

Another difference is that the scale will become more representative to the "real" world. The height of the Geolab is approximately four metre. This becomes visible in the left image.

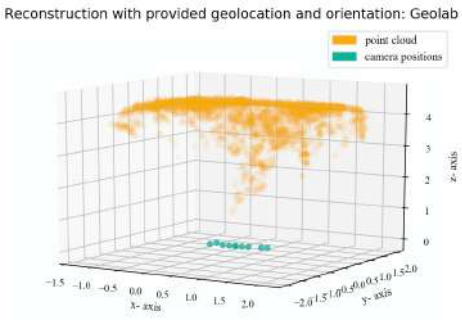


Figure 7.1: Point cloud extracted based on 10 images. Room: Geolab in the Architecture building. Software used: Pix4D with the standard setting 3D Models **with provided local coordinates and orientation**

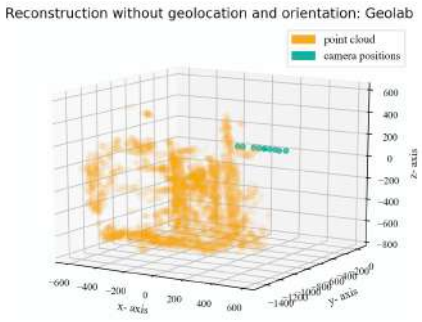


Figure 7.2: Point cloud extracted based on 10 images. Room: Geolab in the Architecture building. Software used: Pix4D with the standard setting 3D Maps **without local coordinates and orientation**.

However, Pix4D as DIM software for performing the reconstruction may alter the provided local coordinates and orientations as can be seen in picture 7.3. Omega may be changed by the software to approximately zero instead of 180 degrees and other parameters may be changed slightly. This change results in a reversed point cloud of the ceiling.

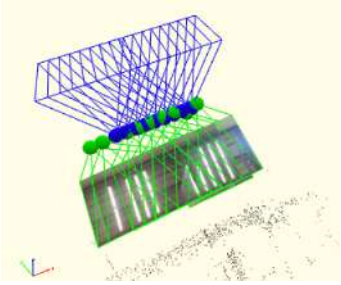


Figure 7.3: Screen shot from the reconstruction of 10 images with Pix4D as DIM software. The blue image planes are the provided local coordinates and orientations. The green image planes are the ones Pix4D used for the point cloud reconstruction.

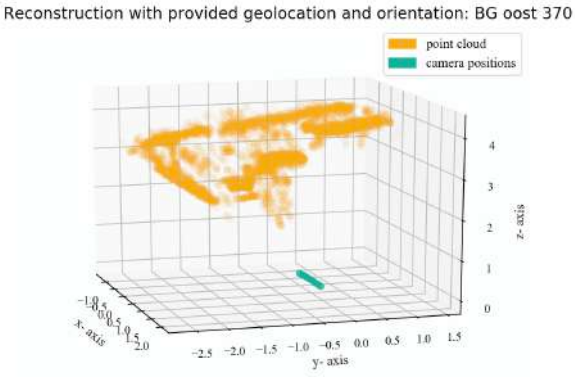


Figure 7.4: Point cloud extracted based on 10 images. Room: BG EAST 370 in the Architecture building. Software used: Pix4D with the standard setting 3D Maps **with provided local coordinates and orientation**.

If the point cloud is not reversed, there may still be a tilt present along the y-axis as can be seen in 7.4. This tilt may be due to a changed omega orientation by Pix4D as DIM software. Omega is changed with approximately 10 degrees.

7.1.2. Results Voxel down sampling and noise removal

In this section, the two noise removal methodologies were discussed and compared in terms of time performance (see figure 7.5 and figure 7.6) and impact on the input point clouds (see figure 7.7,figure 7.8, figure 7.9, figure 7.10). The point clouds used as input are part of the user database generated from DIM.

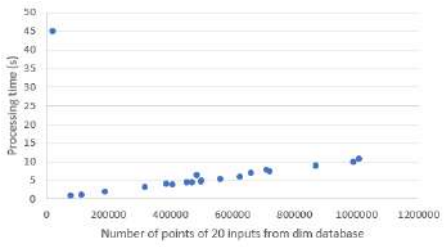


Figure 7.5: Chart showing the relation between the processing time of the additionally suggested noise filtering and number of points of original input point clouds generated from DIM.

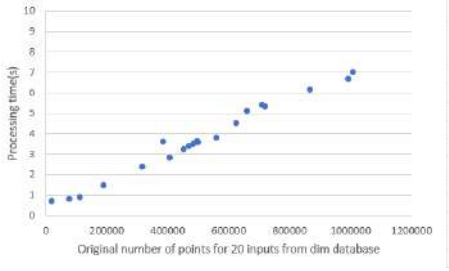


Figure 7.6: Chart showing the relation between the processing time of the Open3D noise filtering and number of points of original input point clouds generated from DIM.

In terms of processing time the additionally suggested noise removal is more efficient as shown in charts figure 7.5 and figure 7.6. On the other hand, as shown in Figures figure 7.7,figure 7.8, figure 7.9, figure 7.10 Open3d noise filtering removes less points but also preserves more ceiling features. As shown in the case depicted in Figure figure 7.8 many random concentrations have been kept in the output point

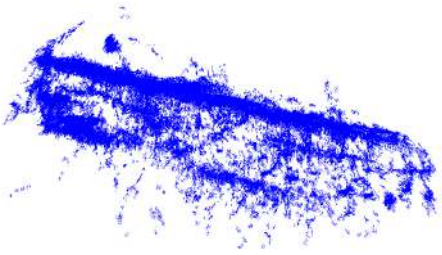


Figure 7.7: Screen shot of the Open3D down sampled point cloud displaying the hallway near the entrance of Berlage hall from DIM.

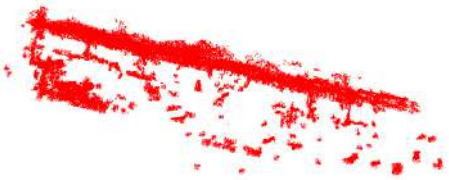


Figure 7.8: Screen shot of the point cloud displaying the hallway near the entrance of Berlage hall from DIM.The point cloud is down sampled using the additionally suggested noise removal.

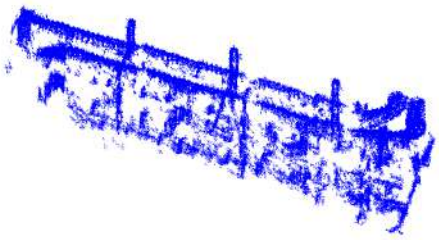


Figure 7.9: Screen shot of the Open3D down sampled point cloud displaying the ceiling of Room B from DIM.

Figure 7.10: Screen shot of the point cloud displaying the ceiling of Room B from DIM.The point cloud is down sampled using the additionally suggested noise removal.

cloud that do not resemble any ceiling features. For this reason, for the suggested input the use of the Open3d method was suggested. The use of this method would ensure handling of noise, however in order to also tackle the problem of the varying point densities a combination was proposed with the voxel averaging method.A visualisation of the outputs is included in appendix A.1.
Additionally, to see the effects of both noise removal and voxel down sampling the NumPy histogram approach with the raw, but tilted input from Pix4D was used. Specifically, the used inputs are:

- User DIM down sampled using Open3D noise removal methods.
- User DIM down sampled using additional suggested noise removal methods (labelled as voxel noise removal).
- User DIM down sampled using voxel averaging.
- User DIM down sampled using voxel averaging and Open3D noise removal.

The voxel size used in all cases was 10.
The Numpy histogram approach was considered to be the best choice since it produced the most promising results. For the user inputs from point cloud 4, 5, 6, 8, 9

and 10 there already was some manual clipping performed according to a bounding box. The file with only the noise removal and only the voxel based noise removal are also tested. Finally, the completely processed files are tested. The results are shown shown in figure 7.11.
In this figure, a map of all tested rooms is shown along with the symbols representing each of the three types of filtering. The number contained in the symbols show the position in which the actual position was matched correctly to the database, i.e. the location of the user was found based on the histogram matching. From the analysis of the results we used the total score for every method as indicator. The total score per method is the sum of the positions that the correct match is found. For the cases where no match is found, six is chosen as entry for the table. The higher the score, the worse the quality of the results produced by the histogram approach. This is shown in figure 7.13.
As can be observed from the table shown in Figure figure 7.13 the methods with the highest scores, are the combined method (average down sampling and noise removal) and average down sampling. Furthermore, as shown in Figure 7.12 from the comparison between the two noise removal methodologies the one where Open3D was applied (labelled "Noise removal") produced lower quality results.
Overall, the additionally suggested noise removal seems to produce better results than the Open3D noise removal and the combined outputs.

7.2. Results histogram approach

The results of the histogram matching are described for the three different implementations that are attempted; the Numpy approach, OpenCV approach and the SciPy approach.

7.2.1. Initial LiDAR database and DIM input

In this subsection, the results of the experiments as subsection 5.2.1 are listed. An overview of the input and the database for this experiment, is provided in appendix A.2.

NumPy approach

Within the most advantageous variation, the amount of correct classifications where the correct database room is within the top three of the different distance measures is displayed in table 7.1. The entrance to BG.WEST.370 and the hallway to room B are classified correctly for user input two. For user input one, the west hallway to the library and the entrance to BG.WEST.370 are classified correctly.

Method	1st	2nd	3rd	Total
Euclidean	4	1	3	8
Chi-squared	4	2	2	8
Jaccard	2	1	1	4

Table 7.1: Correctly classified rooms from most promising NumPy histogram comparison measures, top 3 out of 27.

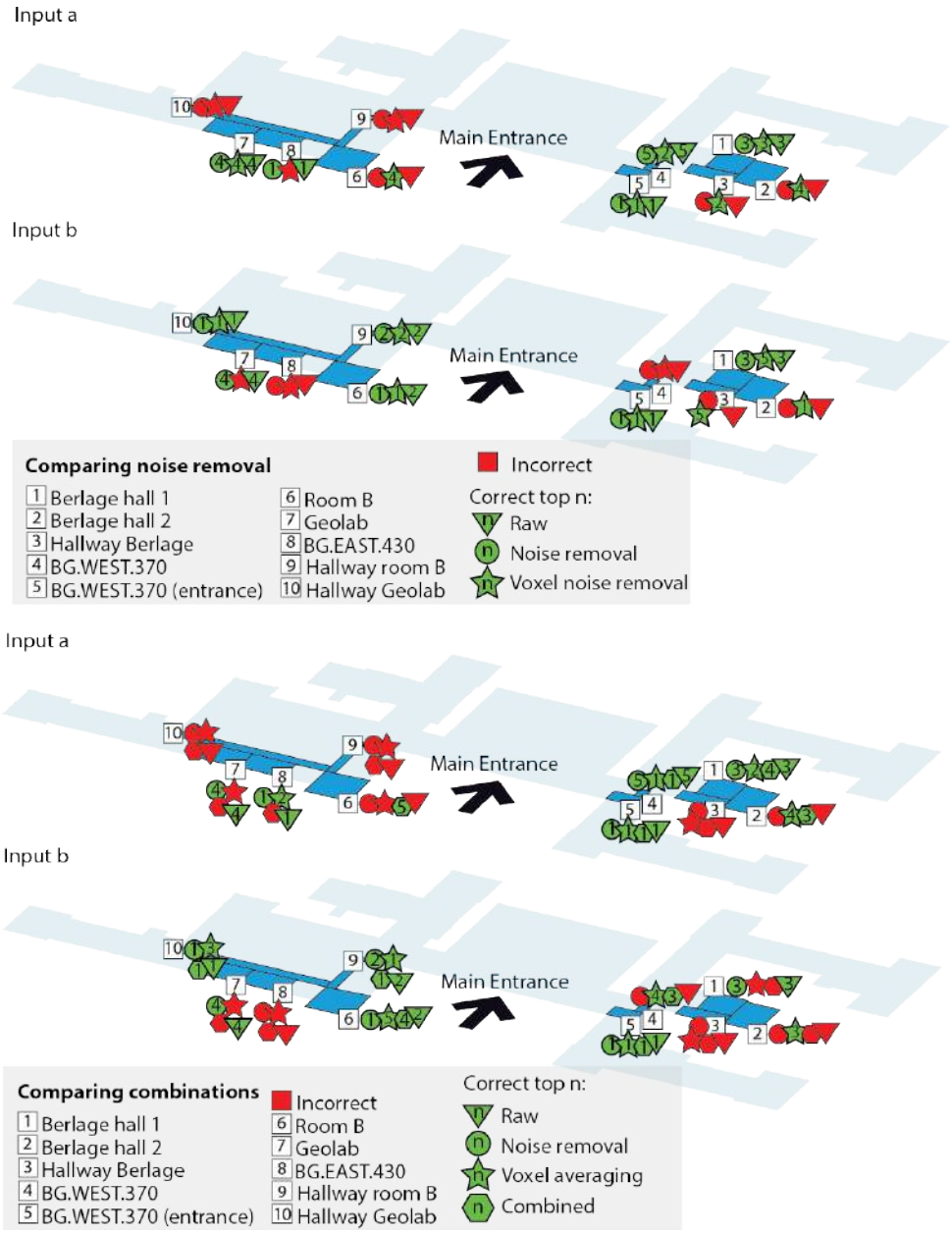


Figure 7.11: Testing different pre-processing steps on the NumPy histogram approach.

Room name	Position of matched location						
	Raw		Noise removal		Voxel noise removal		Total Score
	Input a	Input b	Input a	Input b	Input a	Input b	
Berlage Hall 1	3	6	3	6	3	5	26
Berlage Hall 2	6	6	6	6	4	1	29
Hallway Berlage	6	6	6	6	2	5	31
BG.WEST.370	5	6	5	6	2	6	30
BG.WEST.370(entrance)	1	1	1	1	1	1	6
Room B	6	1	6	2	4	1	20
Geolab	4	4	4	4	4	6	26
BG.EAST.430	1	6	1	6	6	6	26
Hallway room B	6	2	6	2	6	2	24
Hallway Geolab	6	1	6	1	6	1	21
Total score	83		84		72		

Figure 7.12: Table depicting position of matched location per noise removal method

Room name	Position of matched location								
	Raw		Noise removal		Voxel averaging		Combined		Total Score
	Input a	Input b	Input a	Input b	Input a	Input b	Input a	Input b	
Berlage Hall 1	3	3	3	3	2	6	4	6	30
Berlage Hall 2	6	6	6	6	4	3	3	6	40
Hallway Berlage	6	6	6	6	6	6	6	6	48
BG.WEST.370	3	6	5	6	2	4	1	3	30
BG.WEST.370(entrance)	1	1	1	1	1	1	1	1	8
Room B	6	2	6	1	6	5	5	4	35
Geolab	4	4	4	4	6	6	6	6	40
BG.EAST.430	1	6	1	6	2	6	6	6	34
Hallway room B	6	2	6	2	6	1	6	1	30
Hallway Geolab	6	1	6	1	6	3	6	1	30
Total score	79		80		82		84		

Figure 7.13: Table depicting position of matched location per method

OpenCV approach

For this histogram comparison approach, the amount of correct classifications where the correct database room is within the top three of the different distance measures is displayed in table 7.2.

Method	1st	2nd	3rd	Total
Correlation	2	2	1	5
Chi-squared	2	0	1	3
Intersection	2	2	1	5
Bhattacharyya	0	3	3	6

Table 7.2: Correctly classified rooms from most promising OpenCV histogram comparison measures, top 3 out of 27.

SciPy approach

The amount of correct classifications where the correct database room is within the top three of the different distance measures is displayed in table 7.3. BG.WEST.370 Entrance provided the most correct classifications for both user inputs. The hallway to Room B was also correctly classified a lot.

Method	1st	2nd	3rd	Total
Chebyshev	2	3	4	9
Canberra	4	3	1	8
Braycurtis	4	2	2	8
Cityblock	5	2	2	9
Minkowski	5	2	1	8
Euclidean	5	2	0	7

Table 7.3: Correctly classified rooms from most promising SciPy histogram comparison measures, top 3 out of 27.

7.2.2. Experimenting with subsets as LiDAR user input on initial LiDAR database

In this subsection, the results are shown for the experiment as described in subsection 5.2.1.

NumPy approach

The time the processing of this method takes is shown in table 7.4.
The results of the NumPy histogram comparison using a LiDAR subset as input are shown on the map displayed in figure 7.14. For the orange hall, the hallway to room B was deemed more likely. For BG.WEST.370 the `Chi-squared` distance ranked BG.WEST.Hallway and the upper part of the exhibition hall to be more likely. Room P was ranked correctly on the third try of applying the `Euclidean` distance measure. The middle and east part of the hallway to the library seemed more likely.

Amount of points	Time [sec]	Time [sec]	Time [sec]
19564	0.13	0.12	0.12
3698	0.14	0.13	0.13
4703	0.14	0.12	0.13
10234	0.12	0.14	0.14
9105	0.16	0.13	0.15
8009	0.16	0.14	0.15
8642	0.22	0.19	0.15
6181	0.15	0.15	0.16

Table 7.4: Listing the time the histogram approach takes for different input point clouds using the NumPy comparison. The timing is a sum for all three methods tested. The timing has been performed three times to prevent outliers.

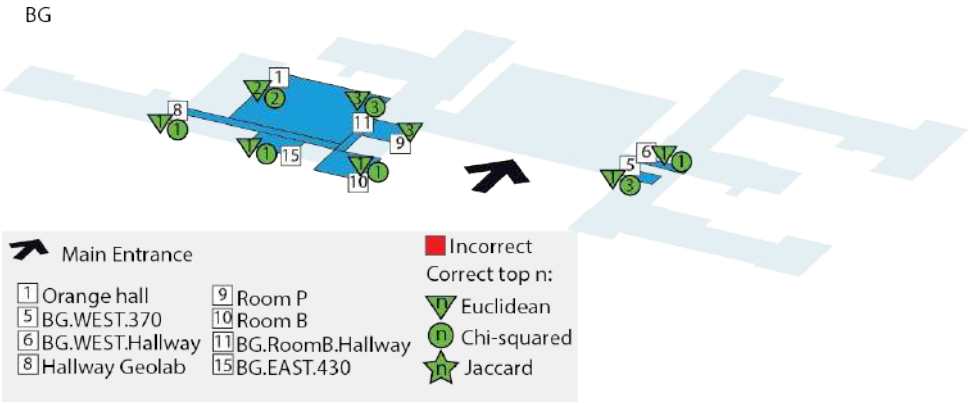


Figure 7.14: Results of the NumPy histogram comparison. For four out of the eight cases, both the `Chi-squared` and `Euclidean` distance provided correct classification results within the first try.

For the hallway to room B, room B and the hallway to the Geolab are first wrongly assigned.

OpenCV approach

The time the processing of this method takes is shown in table 7.5.
The results of the OpenCV histogram comparison are shown on the map displayed in figure 7.15. The hallway of the Geolab was correctly classified in the third try of the `Intersection`. The top `Correlation` and `Intersection` result are both other hallways (BG.WEST.Hallway and the middle part of the hallway to the library). Room P has been correctly classified in the third try using `Correlation`. The BG.WEST.370 entrance, has been identified as the top one result for both the `Correlation` and the `Intersection`. Room B has been correctly classified by the `Correlation` within the first try, for the `Intersection` the second try was necessary and the upper part of the exhibition hall was perceived as more similar. For room B, the room was correctly classified within the second try of both the

Amount of points	Time [sec]	Time [sec]	Time [sec]
19564	0.18	0.18	0.17
3698	0.18	0.18	0.17
4703	0.18	0.17	0.18
10234	0.17	0.17	0.17
9105	0.18	0.18	0.18
8009	0.17	0.18	0.18
8642	0.17	0.17	0.19
6181	0.18	0.18	0.17

Table 7.5: Listing the time the histogram approach takes for different input point clouds using the OpenCV comparison. The timing is a sum for all four methods tested. The timing has been performed three times to prevent outliers.

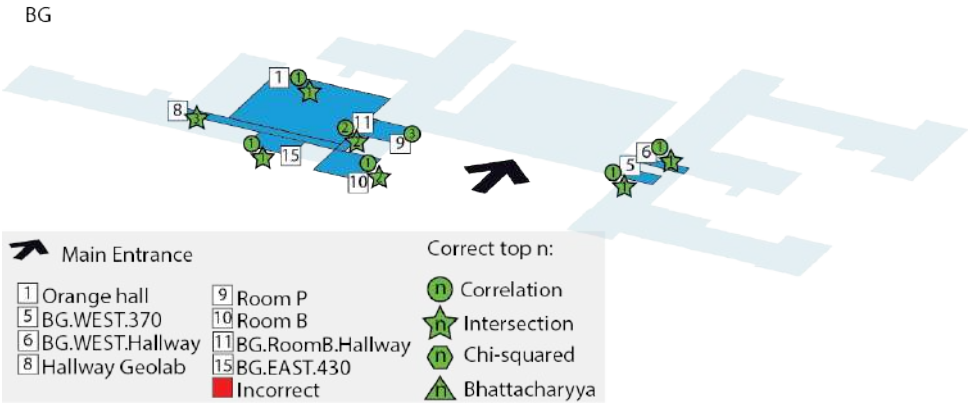


Figure 7.15: Results of the OpenCV histogram comparison. For four out of eight cases, the correct room has been assigned by both the `Intersection` and the `Correlation` in the first try.

`Correlation` and the `Intersection`. The middle hallway of the library and BG.WEST.370 are perceived as a closer match. For all other cases, four out of the eight cases that are tried, the correct room has been assigned by both the `Intersection` and the `Correlation` in the first try.

SciPy approach

The time the processing of this method takes is shown in table 7.6.
The results of the SciPy histogram comparison are shown on the map displayed in figure 7.16. For BG.WEST.370, the `Canberra` measure deemed the upper part of the exhibition hall to be more likely. For the rest, the correct room was the first result for BG.WEST.370. Instead of the Orange hall as first choice, Room P and 01.WEST.Hallway library are selected. Instead of the hallway to the Geolab, room B and BG.WEST.Hallway or room P are selected. Instead of room P, the middle, east or west part of the hallway to the library was selected. Instead of the hallway to room B, room B, Room P or BG.WEST.Hallway 6 was selected.

Amount of points	Time [sec]	Time [sec]	Time [sec]
19564	0.76	0.77	0.95
3698	0.71	0.81	0.78
4703	0.79	0.73	0.74
10234	0.78	0.72	0.72
9105	0.79	0.77	0.80
8009	0.76	0.70	0.72
8642	0.70	0.70	0.73
6181	0.72	0.88	0.72

Table 7.6: Listing the time the histogram approach takes for different input point clouds using the SciPy comparison. The timing is a sum for all six methods tested. The timing has been performed three times to prevent outliers.

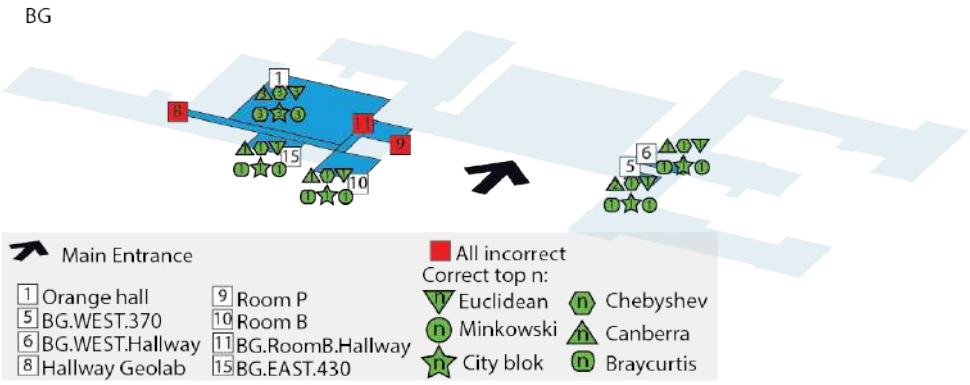


Figure 7.16: Results of the SciPy histogram comparison. Four out of eight rooms are correctly classified in the first try.

7.2.3. New LiDAR database and LiDAR user input

In this subsection, the results are listed for the experiment as described in subsection 5.2.2. For this experiment, no automatic voxel down sampling was added.

The results for the LiDAR user input and the new LiDAR database are displayed in appendix A.4. For the NumPy method, the `Chi-squared` distance provided the most correct results. For the OpenCV method the `intersection` provided the most correct results and for SciPy several measures provided exactly the same results, so it was randomly decided to pick the `Braycurtis` distance. The top 5 correct classifications are displayed in table 7.7. Using the approach as described in subsection 6.3.3 the combined result of all histogram methods are also generated.

Method	**1st**	**2nd**	**3rd**	**4th**	**5th**	**Total**
NumPy	17	4	7	1	1	30
OpenCV	12	6	2	4	1	25
SciPy	15	4	8	1	0	28
Combined	16	5	4	2	4	31

Table 7.7: Listing the top 5 results of the histogram approach for the LiDAR input. Out of 32 tests.

For the top 1 classified values, the confusion matrix of the NumPy approach is displayed in figure A.4. For the OpenCV approach this is displayed in figure A.5, for the SciPy approach in figure A.6 and for the combined approaches in figure A.7, all in appendix A.5.1.
Some of the quality indicators resulting from these confusion matrices can be compared through table 7.8.

Indicator	**NumPy**	**OpenCV**	**SciPy**	**Combined**
Accuracy	0.53	0.50	0.47	0.5
κ - coefficient	0.47	0.43	0.41	0.45
Precision	0.60	0.42	0.52	0.61
Recall	0.57	0.47	0.54	0.58
F1-Score	0.55	0.40	0.41	0.56

Table 7.8: Table showing the different quality indicators (rounded of to two decimal numbers) resulting from the confusion matrices as displayed in figures A.4, A.5, A.6 and A.7. The closer the quality indicators are to one, the better.

7.2.4. New LiDAR database and DIM user input

In this subsection, the results are listed for the experiment as described in subsection 5.2.2. Automatic voxel down sampling combined with and noise removal was conducted based on average values. For the NumPy method, the `Chi-squared` distance gave the most high results, for the OpenCV method the `intersection` and for SciPy again the `Braycurtis` distance was selected. The results of the top 5 amount of correct classification are displayed in table 7.9.

Method	1st	2nd	3rd	4th	5th	Total
NumPy	5	0	2	3	0	10
OpenCV	5	2	3	0	1	11
SciPy	2	4	4	0	1	11
Combined	3	3	0	0	3	9

Table 7.9: Listing the top 5 results of the histogram approach for the DIM input. Out of 20 tests.

For the top 1 classified values, the confusion matrix of the NumPy approach is displayed in figure A.8. For the OpenCV approach this is displayed in figure A.9 and for the SciPy approach in figure A.10 and for the combined approaches in figure A.11, all in appendix A.5.1.
Some of the quality indicators resulting from these confusion matrices can be compared through table 7.10.

Indicator	NumPy	OpenCV	SciPy	Combined
Accuracy	0.25	0.15	0.1	0.15
κ - coefficient	0.17	0.06	0	0.06
Precision	0.22	0.05	0.1	0.125
Recall	0.25	0.15	0.1	0.15
F1-Score	0.21	0.07	0.1	0.13

Table 7.10: Table showing the different quality indicators (rounded of to two decimal numbers) resulting from the confusion matrices as displayed in figures A.8, A.9, A.10 and A.11. The closer the quality indicators are to one, the better.

A more detailed overview of the results of the NumPy histogram approach with DIM user input is shown in appendix A.4 and in figure 7.11.

7.3. Results feature matching

In this subsection, the results are listed for the experiment as described in subsection 5.2.2. The general results of the feature matching approach are shown in appendix A.4.

LiDAR User input

The top 5 correct classifications for different voxel sizes are displayed in table 7.11.

Voxel size	1st	2nd	3rd	4th	5th	Total
v0.2	30	1	0	0	0	31
v0.4	16	8	4	0	2	30
v1.5	6	6	3	3	9	27
v10	3	3	2	3	3	14

Table 7.11: Listing the top 5 results for different voxel sizes of the feature matching approach for the LiDAR input. Out of 32 tests.

The results of the feature matching approach for different voxel sizes are displayed in figure 7.17.

Figure 7.17: Results of the feature matching approach on various LiDAR user inputs. If the top location from signature comparison is the actual input location, then it indicates a 100 % match. If the location does not match correctly, the value indicates the closeness of input to the actual location.

The timing of the feature matching approach for different voxel sizes are displayed in figure 7.18.
For the top 1 classified values, the confusion matrix of the feature matching approach is displayed in appendix A.5.2. For a voxel size of 0.2 in figure A.12, for a voxel size of 0.4 in figure A.13, for a voxel size of 1.5 in figure A.14 and for a voxel size of 10 in figure A.15.
Some of the quality indicators resulting from this confusion matrix can be seen in table 7.12.

Indicator	**v0.2**	**v0.4**	**v1.5**	**v10**
Accuracy	0.94	0.50	0.19	0.09
κ - coefficient	0.93	0.44	0.12	0.00
Precision	0.84	0.43	0.21	0.01
Recall	0.90	0.47	0.21	0.10
F1-Score	0.87	0.40	0.15	0.02

Table 7.12: Table showing the different quality indicators (rounded of to two decimal numbers) resulting from the confusion matrices as displayed in figures A.12, A.13, A.14 and A.15. The closer the quality indicators are to one, the better.

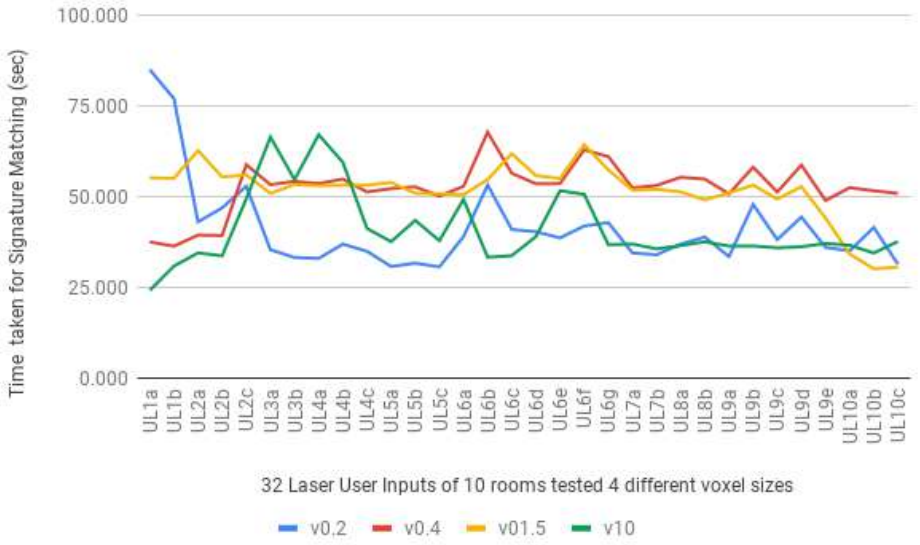


Figure 7.18: Time taken by 32 LiDAR inputs for localisation from 10 rooms.

DIM User input

The top 5 correct classifications are displayed in table 7.13.

Voxel size	1st	2nd	3rd	4th	5th	Total
v0.3	4	2	2	4	0	13
v0.4	2	3	1	4	2	10
v1.5	1	2	3	2	9	10
v10	1	0	2	3	3	9

Table 7.13: Listing the top 5 results of the feature matching approach for the DIM input. Out of 20 tests.

The results of the feature matching approach for different voxel sizes are displayed in figure 7.19.

The timing of the feature matching approach for different voxel sizes are displayed in figure 7.20.

For the top 1 classified values, the confusion matrix of the feature matching approach is displayed in appendix A.5.2. For a voxel size of 0.2 in figure A.16, for a voxel size of 0.4 in figure A.17, for a voxel size of 1.5 in figure A.18 and for a voxel size of 10 in figure A.19.

Some of the quality indicators resulting from this confusion matrix can be seen in table 7.14.

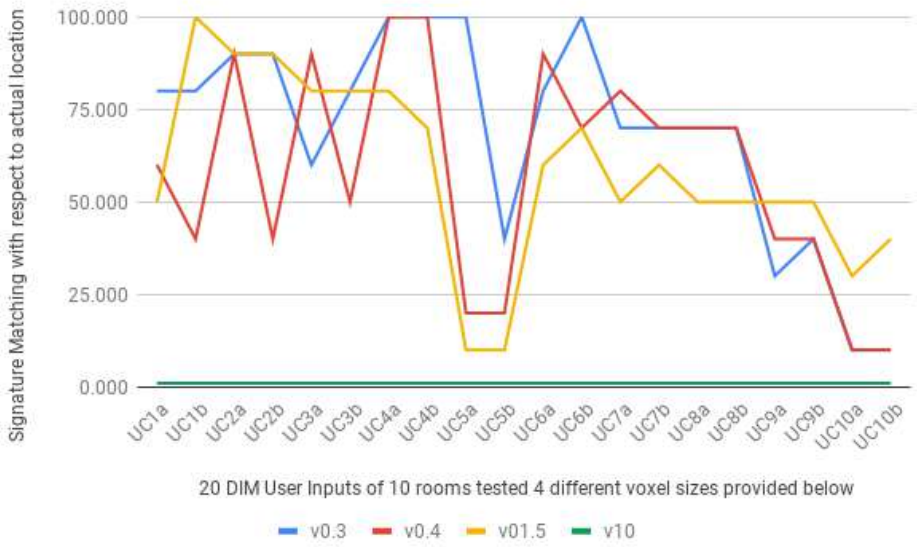


Figure 7.19: Results of the feature matching approach on various DIM user inputs. If the top location from signature comparison is the actual input location, then it indicates a 100 % match. If the location does not match correctly, the value indicates the closeness of input to the actual location.

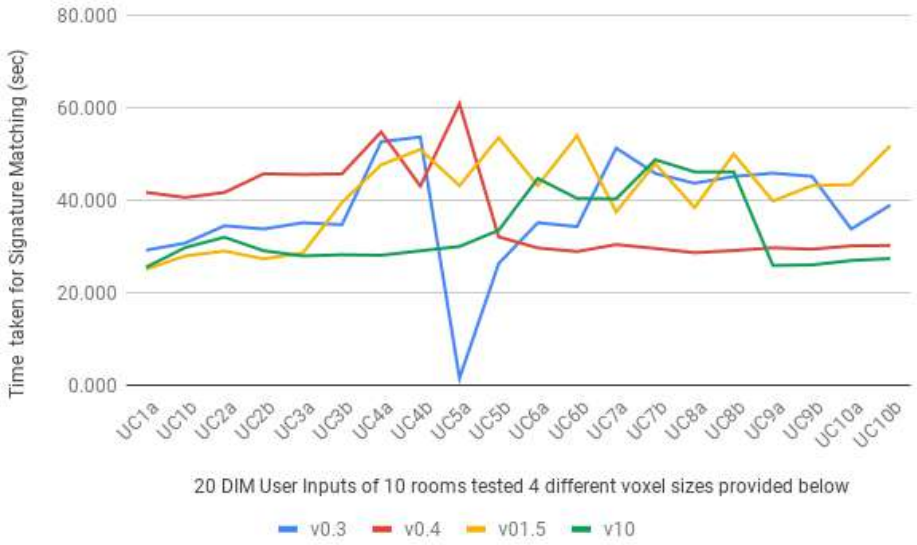


Figure 7.20: Time taken by 20 DIM user inputs for localisation from 10 rooms

Indicator	v0.2	v0.4	v1.5	v10
Accuracy	0.20	0.10	0.05	0.05
κ - coefficient	0.11	0.00	-0.05	-0.05
Precision	0.12	0.03	0.01	0.01
Recall	0.2	0.10	0.05	0.05
F1-Score	0.13	0.04	0.01	0.01

Table 7.14: Table showing the different quality indicators (rounded of to two decimal numbers) resulting from the confusion matrices as displayed in figures A.16, A.17, A.18 and A.19. The closer the quality indicators are to one, the better.

7.4. Results combined

In this subsection, the results are listed for the experiment as described in subsection 5.2.2. An overview of the combined results from the best performing histogram approach and the feature matching approach is shown in appendix A.4. The `Chi-squared` distance measure of the NumPy approach turns out to be the most thrust worthy histogram matching measure in both the case of LiDAR input and DIM input. The feature matching approach turns out to be the most thrust worthy with voxel size 0.2 for LiDAR input and 0.3 for DIM input. For this reason, the NumPy approach will be used for combined testing with feature matching approach. A more detailed overview of the results of the combined approach is shown in appendix A.4.

LiDAR User input

For the top 1 classified values, the confusion matrix of the feature matching approach is displayed in figure A.20 in appendix A.5.3.
Some of the quality indicators resulting from this confusion matrix are shown in table 7.15.

Indicator	Combined approach
Accuracy	0.84
κ - coefficient	0.82
Precision	0.85
Recall	0.81
F1-Score	0.81

Table 7.15: Table showing the different quality indicators (rounded of to two decimal numbers) resulting from the confusion matrix as displayed in figure A.20. The closer the quality indicators are to one, the better.

DIM User input

For the top 1 classified values, the confusion matrix of the feature matching approach is displayed in figure A.21 in appendix A.5.3.
Some of the quality indicators resulting from this confusion matrix are shown in table 7.16.

Indicator	Combined approach
Accuracy	0.2
κ - coefficient	0.11
Precision	0.1
Recall	0.2
F1-Score	0.124

Table 7.16: Table showing the different quality indicators (rounded of to two decimal numbers) resulting from the confusion matrix as displayed in figure A.21. The closer the quality indicators are to one, the better.

7.4.1. Weights combined fingerprinting methods

In subsection 6.3.3 a methodology to combine the results from several fingerprinting methods is described. For this, different types of weighing can be applied. The weights could go from 10 to 0, and can be either multiplied or added. The weights can also go for example from 2^{10} to 2^0 or the indices can be compared in a least squares solution.

Different solutions are tested on two histogram methods, the SciPy and the OpenCV approach for the laser database input. The accuracy and the κ-coefficient are compared for all the measures to see which one is the most likely to provide good results. These results are displayed in table 7.17.

Method	Accuracy	κ-coefficient
Original file 1	0.5	0.47
Original file 2	0.43	0.41
[10-0], multiplied	0.25	0.15
[10-0], added	0.25	0.15
[2^{10}-2^0], multiplied	0.25	0.15
[2^{10}-2^0], added	0.40	0.34
Least squares	0.06	-0.06

Table 7.17: Comparing the quality of the different weight measures when comparing fingerprints. This is based on the top 1 correct classified values. The closer the quality indicators are to one, the better.

7.5. Final results

In this section, the final quality indicators from the different fingerprinting methods attempted in this report are provided for comparison's sake. The results listed this subsection, are related to the experiment as described in subsection 5.2.2. An overview of the combined results from the best performing histogram approach, the feature matching approach and the combined approaches is also shown in appendix A.4.

LiDAR User input

The accuracy indicators of the final results of the NumPy histogram approach, the feature matching approach and both approaches combined for a LiDAR user input are displayed in figure 7.21. The top 5 locations of the different inputs are displayed in figure 7.22.

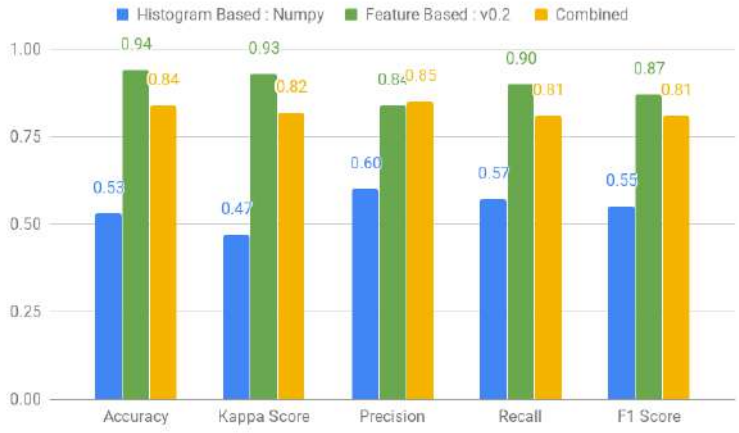


Figure 7.21: Quality of localisation results of 32 LiDAR user inputs tested by three different approaches

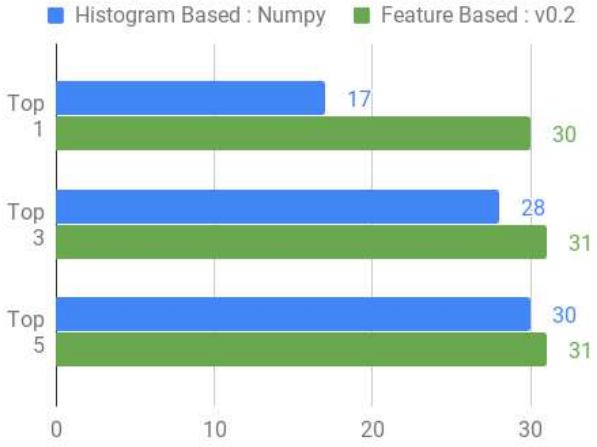


Figure 7.22: Top 5 localisation results of 32 LiDAR user inputs tested by three different approaches; The NumPy histogram fingerprinting, the feature matching approach with voxel size 0.2 and a combination of both of these approaches.

DIM User input

The accuracy indicators of the final results of the NumPy histogram approach, the feature matching approach and both approaches combined for a DIM user input are displayed in figure 7.23. The top 5 locations of the different inputs are displayed in figure 7.24.

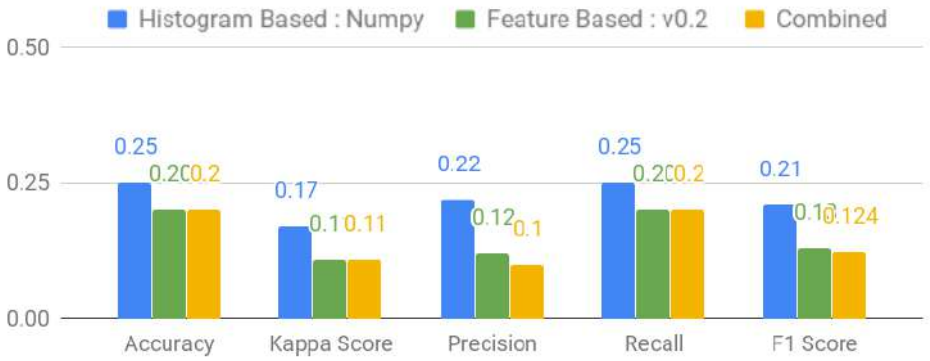


Figure 7.23: Quality of localisation results of 20 DIM user inputs tested by three different approaches.

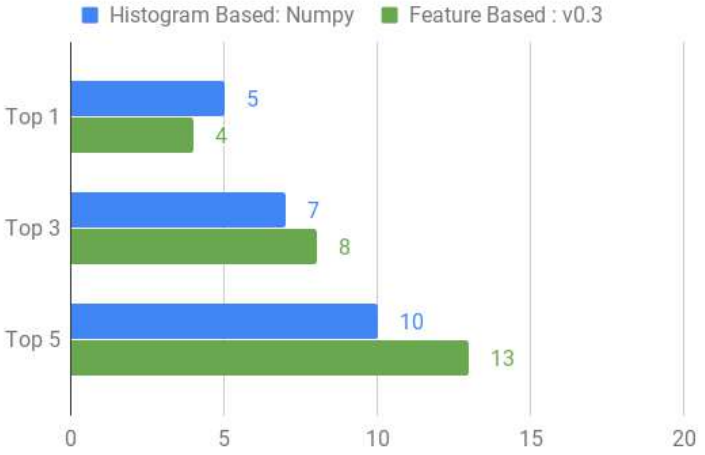


Figure 7.24: Top 5 localisation results of 20 DIM user inputs tested by three different approaches. The NumPy histogram fingerprinting, the feature matching approach with voxel size 0.3 and a combination of both of these approaches.

8
Discussion

In this chapter we will discuss our results and the pros, cons and uncertainties of the different steps in our research.

8.1. Pre-processing

In the section below, the following pre-processing results will be discussed: handling tilt, voxel down sampling and noise removal. This will be done by giving answers to the following questions. What are the pros and cons of these pre-processing methods? What are the uncertainties of these pre-processing methods? Can these pre-processing methods be used in practise?

8.1.1. Handling tilt

This subsection describes the challenges of the current theoretical setup. There may still be tilt present after providing the local coordinates and orientation. Pix4D as DIM software changes the provided orientation for the reconstruction. However, we are not sure if this is the only cause of the tilt along the y-axis, because this is not further investigated yet. The tilt may also be caused by other artefacts. To eliminate other artefacts, the following test can be conducted. A rotation matrix based on the differences between the provided ω, ϕ and κ and the used ω, ϕ and κ could be applied to remove the tilt.

Besides this, there is not yet a link made between the theoretical setup and sensors available in mobile devices. This may be done by using sensors in a mobile phone for instance, namely the three axis gyroscope and the three axis accelerometer. However, the accuracy of these sensors and how they would be used is not yet explored in this research.

If those uncertainties can be removed, our general approach will become more realistic and will almost certainly work better. The z-bias must be removed to turn the histogram approach into a real world application, but the following things will become an additional advantage too.

- The generated point cloud may become less distorted and therefore more accurate, which will improve the matching between the database and the user input. This means that there will be less noise present.
- The scale will become closer to the scale of the database and will therefore improve the feature matching approach. The point cloud of the user and the point cloud of the database will become closer in space.

- The scale may improve the voxel down sampling approach, because all point clouds can be down sampled according to the same scale.
- Our methods are based on an unknown scale. Incorporating scale will make other methods for ceiling point cloud possible.

8.1.2. Voxel down sampling

Even though the voxel down sampling method provides a regular grid, it is highly dependant on the selected voxel size and thus requires experimentation in order to generate a representative output. For this reason, an approach that takes into account the local distribution of points is suggested. By this we mean, that neighbourhoods with different point densities representing more complex features would be taken into account and down sampled using a different voxel size. In order for this to be done automatically an octree based down sampling method could be a better match for our approach and improve the pre-processing methodology. However, because time performance is an important factor; a comparative investigation of the two methods is suggested not only in terms of the output grid's accuracy but also in terms of processing performance.

8.1.3. Noise removal

The applied noise removal is the combination of the two methods both of which have scientific basis. It also takes into consideration two possible appearances of outlier points. Points with only few neighbours are seen as the outliers and the points that are far from their neighbours are likely to be the noisy points.

However, it has drawbacks that the selection of parameters may not be the best, because the parameters for both methods are defined by trials and errors. Although during the experimentation the results that more noisy points are removed while less object points are omitted are regarded as the better outputs. This experimentation process may be subjective.

The uncertainty of the noise removal method is that the density of input point clouds influences the parameter selection especially the radius-based method. Two inputs are tested. One are the original point clouds from DIM and another one is those after down sampling. During the experimentation, the choices of radius parameter are different since the down sampling reduce the density of raw point clouds, the radius for searching outliers are selected to be smaller. The effect of point clouds density on the parameter choices may weaken the usability of this noise removal into automation for all kinds of input point clouds.

8.2. Histogram approach

The results from the different experiments are discussed in the following subsections.

8.2.1. Initial LiDAR database

The first test we conducted used user input and database as described in subsection 5.2.1. The results from this experiment are listed in subsection 7.2.1.

NumPy approach

The `Jaccard` distance turned out to be the least suitable for our specific scenario.

In general the histograms that are visually similar, also provide the best matching. For example, for user input 1, the west part of the hallway to the library and the entrance to BG.WEST.370 are classified correctly. The histograms for the entrance to BG.WEST.370 are displayed in figure 8.1. On the first glance, they might not seem similar, but when compared to histograms that did not lead to a correct match, such as for example room BG.WEST.370 as displayed in figure 8.2, they are indeed more alike.

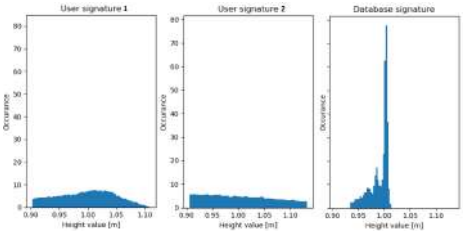


Figure 8.1: NumPy histograms Entrance BG.WEST.370 which match good.

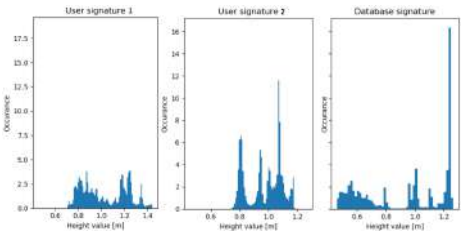


Figure 8.2: NumPy histograms BG.WEST.370 which did not match.

OpenCV approach

The classification results behaved as expected, visually less comparable histograms such as the example of BG.WEST.370 (figure 6.16), did not result in the correct classification. The `intersection` and `correlation` lead to more top one results than the other metrics. However, these results are still not accurate enough to be usable in an actual application.

SciPy approach

The hallway to room B was correctly classified as when using the point cloud that was created from less frames (185 from a video of 26 seconds compared to 256 from a video of 31 seconds). When looking at the visualised histograms as displayed in appendix A.2, these indeed seem to be alike. Room BG.EAST.430 was correctly classified for user input one, here 264 frames from a video of 45 seconds are used compared to 147 frames from a video of 24 seconds.

On the first glance, the results displayed in table 7.3 do not seem thrust worthy enough for an actual indoor localisation application. The user input point cloud generated from input pictures seems to be incompatible with the database point clouds. However, for types of input more similar to the database, it could work. Combined with another matching methods, the histogram approach could be a valuable asset.

8.2.2. Experimenting with subsets as LiDAR user input on initial LiDAR database

The results from the experiments conducted using subsets of the original LiDAR database as new user input, are displayed in subsection 7.2.2.
Within our test setup, all three methods performed almost equally well in terms of correct first try assignments. However, for the SciPy method, there are three cases where the correct classification was not even in the top three. For both the Numpy and the OpenCV approach, this never happened.

NumPy approach

The time the processing of this method takes is shown in table 7.4. The processing is near real-time. It can be done on the fly in an indoor navigation approach whilst a potential user is moving from one location to another.

The results of the NumPy histogram comparison are shown on the map displayed in figure 7.14. The `Jaccard` distance did not provide any correct matches, this can indicate that it is not a suitable measure for our specific data set.
In all cases of wrong assignment, the histograms are indeed not the most distinctive and for some the ceiling of different locations is visually alike (as is the case when for example mixing up two hallways).

OpenCV approach

The time the processing of this method takes is shown in table 7.5. Note that these times are slightly longer than the ones from the NumPy approach, this was to be expected because we incorporated an additional measure for comparison (four instead of three).

The results of the OpenCV histogram comparison are shown on the map displayed in figure 7.15. No rooms are classified correctly using the `Bhattacharyya` or `Chi-squared` distance. From this, we can conclude that these distance measures are not suited for comparison of our type of data set.

SciPy approach

The time the processing of this method takes is shown in table 7.6. This method required a longer processing time, this is partly due to the fact six distance measures are tested, but in general the method seems to be the slowest.

The results of the SciPy histogram comparison are shown on the map displayed in figure 7.16. For BG.WEST.370, the `Canberra` measure deemed the upper part of the exhibition hall to be more likely, this is surprising, both have differently looking ceilings. The rest of the results can all be explained.

8.2.3. New LiDAR database and user input

For the actual LiDAR user input as displayed in appendix A.4, the histogram approach is tested for all three methods (NumPy, OpenCV and SciPy).

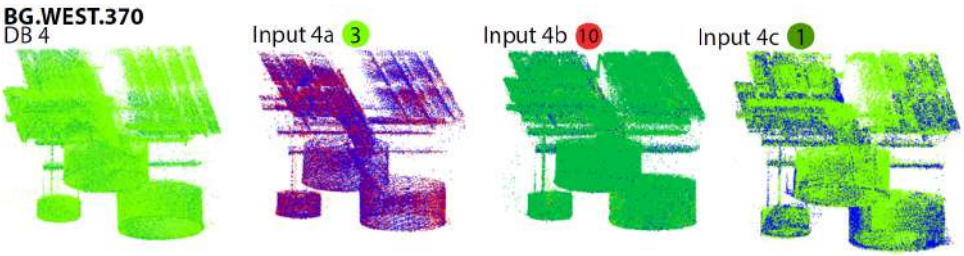


Figure 8.3: BG.WEST.370 histogram approach classified wrong.

Some of the quality indicators resulting from these confusion matrices can be compared through table 7.8. The closer the quality indicators are to one, the better. From the indicators listed in table 7.8, we can concluded that on its own, the histogram approach does not have a high enough quality for thrust worthy localisation on the first try. The NumPy approach seems to be lead to more thrust worthy results than the other two measures.

For the NumPy approach, Room BG.WEST.370, as is displayed in figure 8.3, turned out to be classified wrong in several cases. For input 4a, Berlage hall 1 and the hallway to the Geolab seemed more likely. For input 4b, the hallway to the Geolab, the hallway to room B and the hallway to the Berlage halls are the top three results. The matching histograms are displayed in figure 8.4. There seems to be a shift in the height values of the histogram which is especially evident for input 4b, and to some degree also for input 4a. Expect from these shifts, the histograms are very similar. An explanation for the shift has not been found. We tried different methods for normalisation and not normalising at all, which should work for the LiDAR input which is of the same scale. The input files with shift are all from the original database from CGI, the fact that they are captured with a different scanner could be a cause. However, it is still unexpected that normalisation does not correct any possible differences and that it is not happening for all cases in which we used these point clouds.
The results from the hallway to room B and the hallway to the Geolab are displayed in figure 8.5. The hallway to room B was classified correct in three from the five cases. The mismatch for input 9c and 9e might be explained by the fact that both input point clouds are not dense, and capture only part of the hallway. Similar point clouds such as the hallway to the Berlage hall, are deemed more likely. For the hallway to the Geolab input a, among others the Geolab (room 7), the hallway to the Berlage hall (room 3) and the hallway to room B (room 9) seemed more likely. For input c this was also the case. For input b, the hallway to room B seemed more likely. As can be expected, similar rooms such as hallways, have similar histograms that might not be distinct enough for localisation purposes, this can be seen in the histograms displayed in figure 8.6. The histogram from input 10a is again shifted. Input 10b is classified reasonably well and input 10c might no be a representative subset.

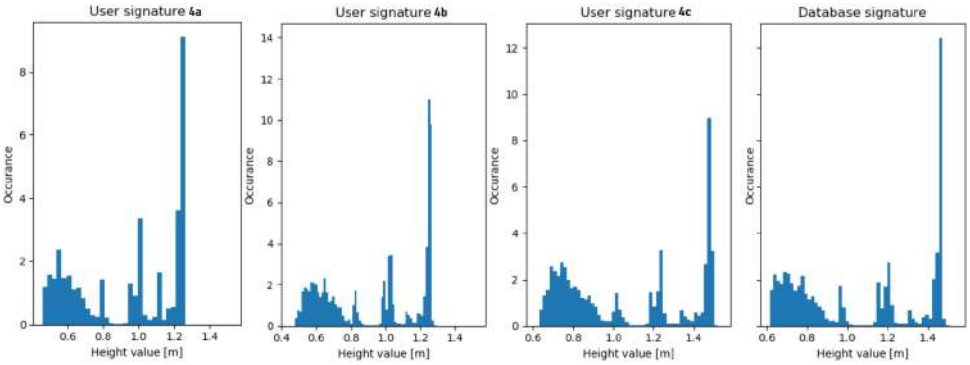


Figure 8.4: BG.WEST.370 histogram approach classified wrong, matching NumPy histograms.

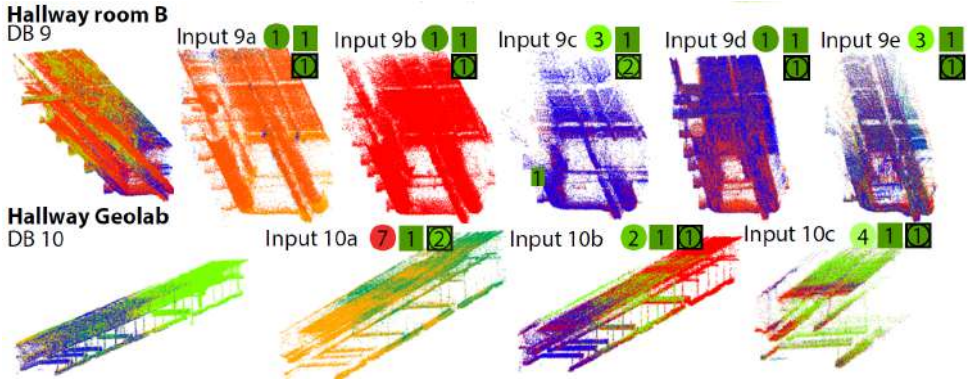


Figure 8.5: Hallway to room B and hallway to the Geolab histogram approach classification results.

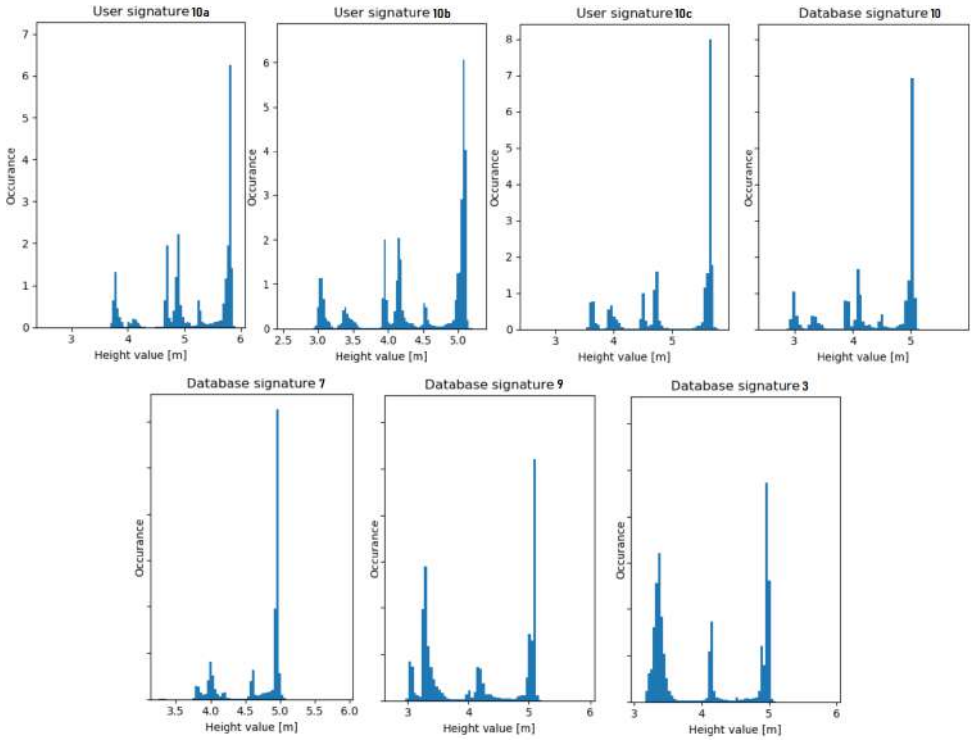


Figure 8.6: Histograms from input 10a, 10b, 10c and databases 3, 7, 9 and 10.

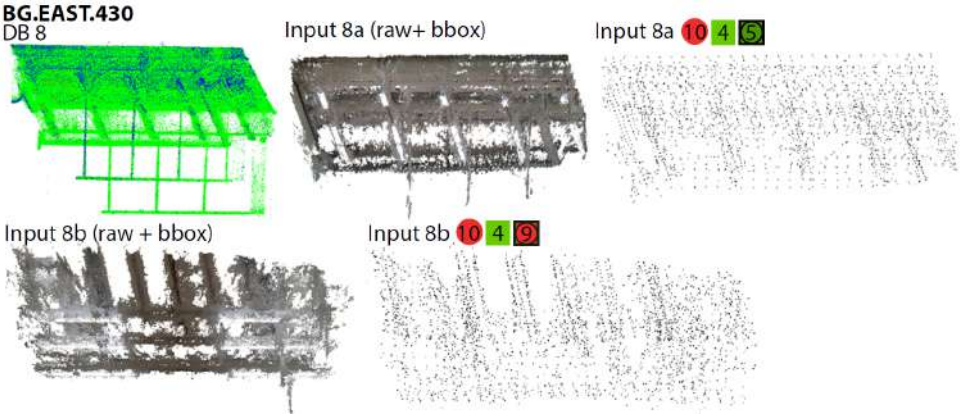


Figure 8.7: BG.EAST.430 NumPy histogram approach classified wrong.

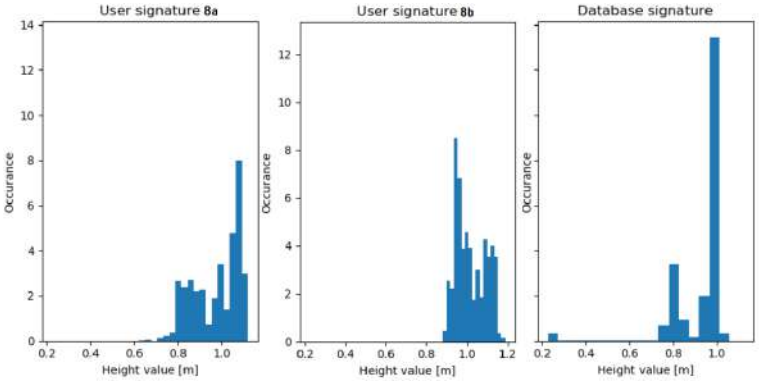


Figure 8.8: Matching histogram BG.EAST.430 NumPy histogram approach classified wrong.

8.2.4. DIM User input

Combined with the resulting accuracy measures listed in table 7.10, the NumPy approach turns out to be the most thrust worthy histogram matching measure in case of laser database input.

An example of a wrong classification is Room BG.EAST.430, as shown in figure 8.7. When looking into the matching histograms, as displayed in figure 8.8, and the way the point clouds correspond, this can make sense. A possible explanation for this specific case can be that too much details are lost during the pre-processing.

Another example of the results is displayed in figures 8.9 and the matching histograms in figure 8.10. Here we see that input a has a wider distribution of points along the normalised height. Input b, like the database, does not have this, and thus matches the database histogram in a better way.

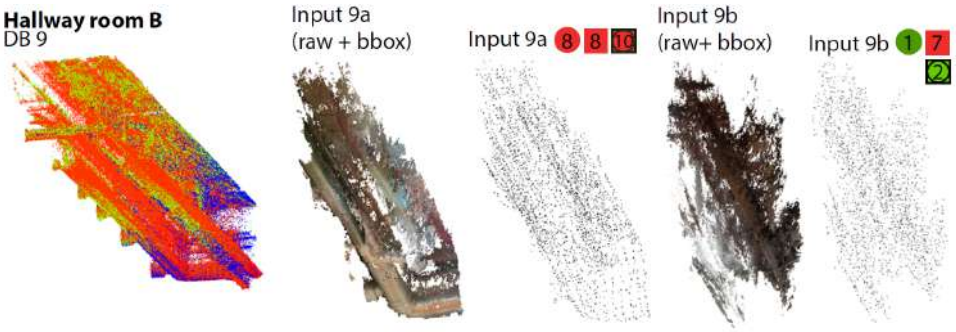


Figure 8.9: Room B NumPy histogram approach classified wrong for input a and right for input b.

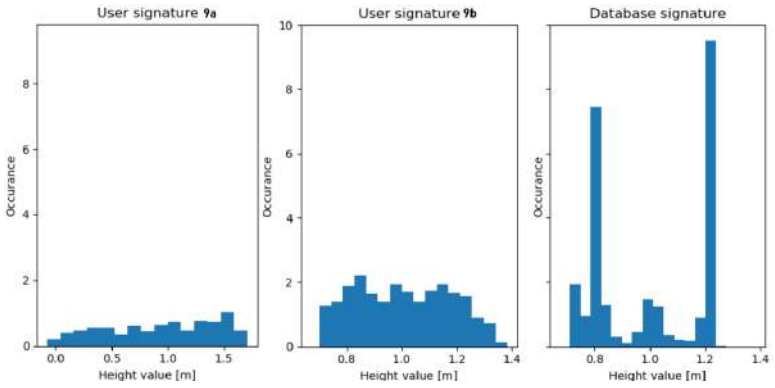


Figure 8.10: Matching histogram Room B NumPy histogram approach classified.

8.2.5. Histogram matching pros and cons

The one dimensional signature acquired in the form of a histogram has several advantages for the matching process. Among others, it is rotation and scale invariant. Furthermore, the processing is quick. This means that it could be implemented for on the fly localisation feedback in a navigation solution.
However, the histogram approach also has a few challenges. The method works best with pre-processed point clouds without much noise and the amount of wall points that is captured can be of big influence. Furthermore, the user point cloud ceiling should be parallel to the database ceiling, which means tilt handling is required.

In general, it might be that the database histograms for different rooms are not distinct, or not representative enough to be used as a unique signature in a fingerprinting process. This might be the case for different types of buildings than the Architecture building in which our tests are conducted. However, for some kind of incremental approach of a navigation solution, the trajectory information will be available and the histogram approach might become more accurate because locations that are not within reach of the user can be excluded.

DIM Specific pros and cons

For the DIM the results are strongly influenced by the subset the user captures. If a specific room contains many different elements in only a specific subsection of the ceiling, and the user measures a subset of that specific room which does not contain any of such elements, the different signatures might not match. This is something that could theoretically be solved by subdividing the room in subsets, and perform matching based on the subsets. However, the use of subsets introduces new challenges. We cannot know for sure that the user will capture one of the subsets we selected, the captured section might be a combination of different subsection. Furthermore, more smaller histograms can lead to less distinction between the separate signatures, which makes the matching process less effective. Another important factor is the fact that both the database point cloud and the user input point cloud should be compatible in order for the histograms to be equally smooth. This is somewhat enforced through the pre-processing as described in section 6.2.

LiDAR Specific pros and cons

For LiDAR user input and LiDAR database, both point clouds are already compatible. However, with a LiDAR scanner it is difficult to only capture a specific section. This means that not only the ceiling will be captured, which means ceiling extraction is required. Furthermore it is also possible that the point cloud is not captured on room level. This means that separate rooms need to be extracted to be able to match to the database.

8.3. Feature matching

The results from the different experiments are discussed in the following subsections. Combined with the resulting accuracy measures, the voxel size 0.2 turns

out to be the most thrust worthy feature matching measure in case of a LiDAR input while it is 0.3 for DIM input . For this reason, these will be used for combined testing with a histogram matching approach

8.3.1. LiDAR database and user input

For the actual LiDAR user input as displayed in appendix A.4, the feature matching approach is tested for four voxel sizes (0.2,0.4, 1.5 and 10).

Some of the quality indicators resulting from these confusion matrices can be compared through table 7.12. The closer the quality indicators are to one, the better. From the indicators listed in table 7.12, we can concluded that on its own, the feature matching approach has a good reliability for localisation on the first try.

From the overall performance results mentioned in figure 8.11 for 32 inputs, it can be seen that the algorithm produced 98% correct results with average fitness of 0.97 and Root Mean Square Error (RMSE) of 0.15. It is observed that as the voxel size increases, the strike rate of algorithm decreases. However, the average fitness is still good, the RMSE has increased indicating spatial incoherence. From figure 7.22 and 7.17,it can inferred that the voxel size 0.2 produced best results with correctly locating all inputs except for Geolab. The user inputs of Room B, Hallway to Geolab and Room B showed good results with 0.2 and 0.4 but fails when small user inputs are compared with large hallways for voxel size 1.5. Berlage rooms were confused in localisation by both voxel size 0.4 and 1.5. Overall, the algorithm provides good top5 results as the voxel size increases, although the ability to provide top1 results decreases.

Timing is also crucial in determining location. The figure 7.18 shows that the timing of localisation increases as the voxel size increases. The v0.2 has time range of 30-45 seconds for 10 rooms. If only 3-4 rooms are provided to the algorithm using the information, then the timing might reduce to 5-10 sec depending on voxel size. It can be conclude that the variability in point distribution and density also gives variation in timing and outcome of localisation.

Ability to distinguish location in neighbourhood rooms in database

The feature matching approach showed promising results (within voxel size 0.1 to 0.5) to provide localisation solution for LiDAR user inputs. For example, as depicted in figure 8.12, Berlage 1, Berlage 2 and Room B have many similar features but the algorithm provided correct location. In both cases, Room B comes at second position with fitness value 0.89 and 0.90 which indicates that the alignment is very similar. Also, the increase in root mean square error of Room B gives indication that the corresponding feature pairs are not spatially locking. It is able to differentiate rooms based on the local refinement of surfaces of ceiling points.

Ability to identify hallways with different scale of user inputs

It was interesting to observe that the algorithm was able to provide correct location for Hallways to Room B and Geolab. An example for localisation of inputs from

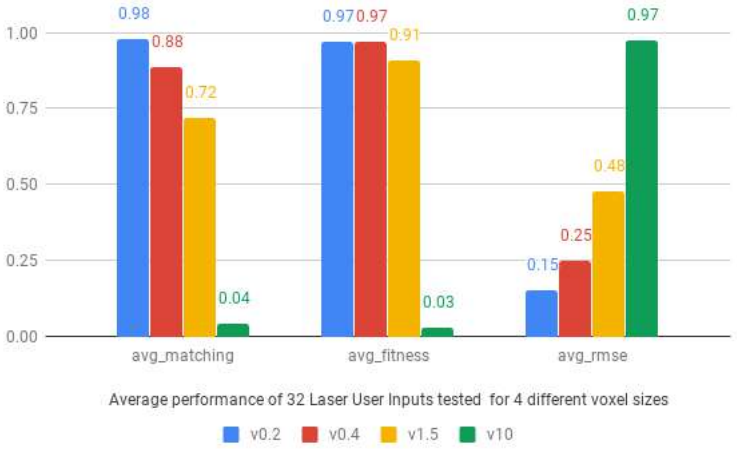
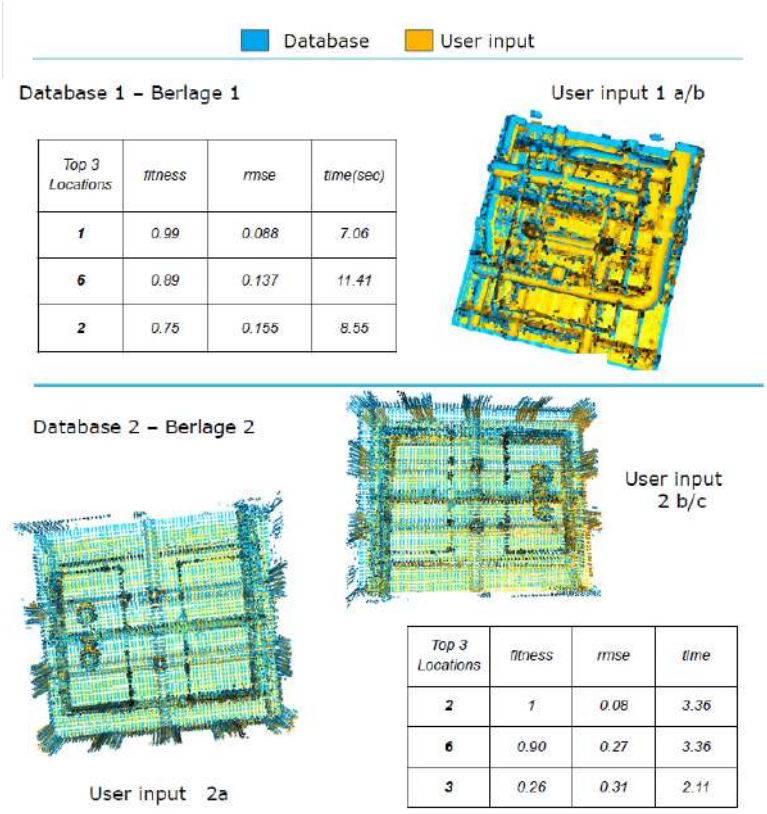


Figure 8.11: Performance overview of feature matching algorithm for LiDAR Inputs

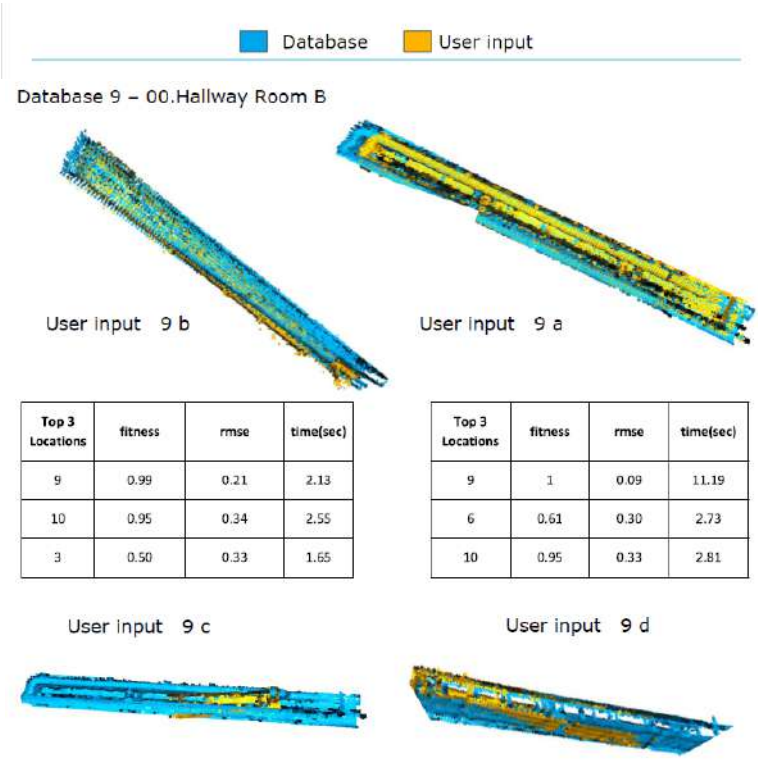


Top 3 Locations	fitness	rmse	time(sec)
1	0.99	0.088	7.06
6	0.89	0.137	11.41
2	0.75	0.155	8.55

Top 3 Locations	fitness	rmse	time
2	1	0.08	3.36
6	0.90	0.27	3.36
3	0.26	0.31	2.11

Figure 8.12: Berlage rooms feature matching correctly classified.

hallway to Room B has been depicted in figure 8.13. It was possible to provide correct location but there is a close competition from Hallway to Geolab in fitness values. The larger input a much smaller RMSE than the smaller input but a small section of ceiling as user input was able to align with the full hallway and reject other rooms.



Top 3 Locations	fitness	rmse	time(sec)
9	0.99	0.21	2.13
10	0.95	0.34	2.55
3	0.50	0.33	1.65

Top 3 Locations	fitness	rmse	time(sec)
9	1	0.09	11.19
6	0.61	0.30	2.73
10	0.95	0.33	2.81

Figure 8.13: Results from database 9.

8.3.2. DIM user input

For the actual DIM user input as displayed in figure 7.19, the feature matching approach is tested for four voxel sizes (0.3,0.4, 1.5 and 10).

Some of the quality indicators resulting from these confusion matrices can be compared through table 7.14. The closer the quality indicators are to one, the better. From the indicators listed in table 7.14, we can concluded that on its own, the feature matching approach does not have good reliability for localisation of DIM input on the first try.

From the overall performance results mentioned in figure 8.14 for 20 inputs, it can be seen that the algorithm produced correct results with average of 55%, with best

average fitness of 0.70 and RMSE of 0.55 for voxel size 0.3. It is observed that as the voxel size decreases, the strike rate of algorithm decreases, and the average fitness increases while the RMSE has increased. The figure 7.24 and 7.19 shows that there is similarity in trend of localisation among all voxel sizes. The hallway to Geolab and Room B and some user inputs of BG.WEST.370 entrance were the most mismatched locations while BG.WEST.370 (Coffee Corner), Room B and Berlage hall 2 showed best results. Overall, the algorithm provides top5 results for 10 out of 20 rooms, as the voxel size increases, although, the ability to provide top1 results decreased drastically.

The figure 7.20 shows that the timing of localisation remains in similar range of 30-50 seconds for 10 rooms in all voxel sizes. There is consistency of timing for most rooms. The v0.3 has best time of 30 seconds for 10 rooms for 10 inputs. If only three or four rooms are provided to the algorithm using the information, then the timing might reduce to 5-10 seconds depending on voxel size.

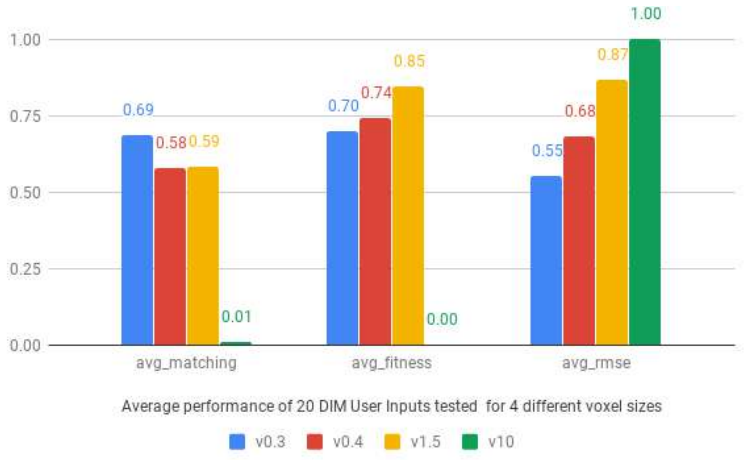


Figure 8.14: Performance overview of feature matching algorithm for DIM Inputs

Rigid Alignment for DIM user input

To apply matching algorithm for DIM input, it was necessary to do an initial alignment using rigid transformation described in 6.3.2. Because the DIM inputs are from different devices, they have different coordinate systems and scale. Hence, initial transformation is not known and thus difficult to perform matching for DIM input. It takes a lot of time to perform this method depending on number of points but it brings the two point clouds in same space for matching process as depicted in figure 8.15. The pre-processed input shows good rigid alignment with respect to its raw input 2. Raw input 1 has more features than 2 which helps in better alignment. From testing, it was observed that all pre-processed inputs align the point clouds in perfect manner while raw inputs shows lot of variation. Thus, an exploration is required for testing histogram approach after rigid alignment.

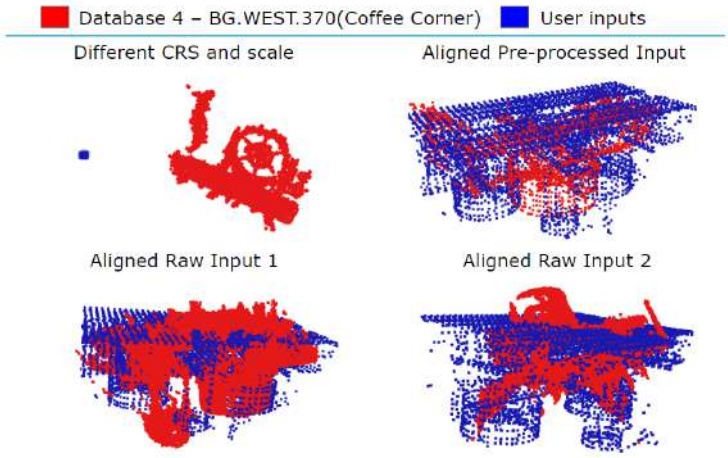


Figure 8.15: An example of rigid transformation of raw and pre-processed DIM Inputs

8.3.3. Feature matching pros and cons

The 33 dimensional signatures acquired in form of FPFH features provides an advantage of using the local refinement of surfaces inside ceiling point cloud with respect to geometry, and topology to detect correct location. This approach of aligning the point clouds on the basis of normal orientation, edge threshold and distance threshold rejects the insignificant features. This is helpful in reducing time of alignment and maintaining the complexity of point clouds even with high down sampling.

The feature matching approach sometimes generate random unreliable results for the inputs with which it has provided correct location mostly, as depicted in figure 8.16. This can happen because of several reasons. Some can be controlled, while others can not. The parameters set for a particular operation might generate false results while random testing or unexpected interruption. Since, the alignment of point cloud is based on picking random correspondence pairs transformation, the number of iterations for pairwise registration might not be able to successfully align the point clouds. The number of neighbours or search radius are not optimum enough to handle the complexity of point clouds and define local surfaces.

DIM Specific pros and cons

If the user inputs are in different coordinate systems and initial transformation in unknown as in case of DIM , it is not suitable for matching. The algorithm provides an option of rigid alignment for roughly aligning the two point clouds before actual matching. This makes it slightly invariant to scale, translation and rotation. However, there should be realistic scale balance between input and database, otherwise the results will become unreliable. For example, directly matching a small area input with a very large hall will be inconsistent, instead dividing the large hall in two or three parts will increase the reliability of results. The rigid alignment also takes

a lot of time depending on the size of point clouds. Furthermore, DIM is not able to provide accurate results for same parameters as depicted by 7.23 and 7.24. This is also because the levels of local surface described in figure 6.18 changes with inconsistency in point distribution and variation in number of points.

LiDAR Specific pros and cons

For the LiDAR input, the algorithm is also able to provide high accuracy and precision with small voxel sizes as depicted in 7.21. It is able to identify hallways with even small user inputs although this sensitivity decreases as the voxel size is increased. The algorithm is also able to distinguish similar rooms up to certain extent if a small voxel size is used for comparison. The timing of algorithm for one user input to find match in 10 rooms is about 30-50 seconds. This can be reduced by storing FPFH features in database instead of directly using the point cloud can reduce the time. Also, if a continuous feed of LiDAR input is provided, the algorithm can perform localisation in a faster way reducing the number of rooms to be compared using trajectory information.

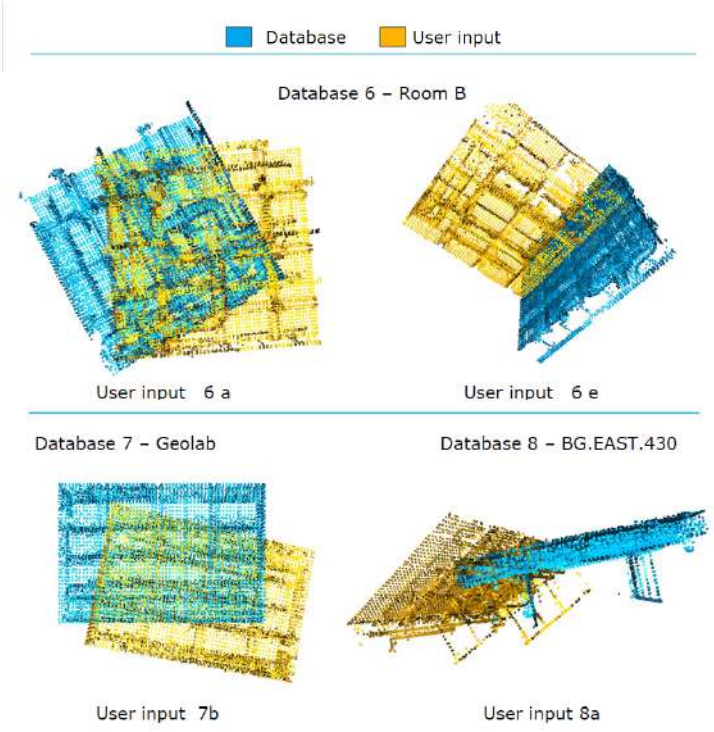


Figure 8.16: False sample results from database 6, 7 and 8.

8.4. Combined results

Combining different fingerprinting output through weighing, requires more experiments. The behaviour could be different for different files. Further more, the performance of the combined approach is not yet perfect. For future work it would be advisable to look into a more suitable method to combine the different measures. For now it seems that combining different methods with our approach, can reduce the absolute quality of the results. The might be explained by the fact that it in some way averages out the results, meaning that if one approach has a proper classification, it is relatively easily brought down.

8.5. Confusion matrix

Something to note from our use of a confusion matrix, is that the amount of samples is limited (32 samples for 10 rooms in case of the LiDAR user input and 20 samples for 10 rooms for the DIM input). This is less ideal, for a more solid quality indication, more samples need to be tested.

9
Research questions

In this chapter an overview of the answers to the research questions of this research is provided. To answer the main research question, first our sub-questions will be answered.

9.1. What is a room?

We try to perform localisation based on identifiable ceiling characteristics of each room. For this reason, the occurrence of a ceiling is an important characteristic every room should full fill. A room does not necessarily have to be enclosed between walls, but a crisp boundary between the ceilings of different rooms is desirable for our solution. This does not mean that the ceiling should be similar for a full room. Based on separated ceiling characteristics within a room, it could be possible to divide a room within subsets, and to perform localisation based on subset level.

How to distinguish hallways?

Because hallways are considered to be rooms, we will not make any distinctions between rooms and hallways. Because hallways can be long structures, it might not be very informative to solely notify to the user that they are in a hallway. For this reason, it could be advantageous to divide hallways in (sub)rooms. A drawback of this approach is that this will only be possible if there is enough distinction between the ceiling parts of such subsections.

How to distinguish the ceiling of a room?

The base structure of the ceiling often is a planar structure on the highest part of the room. Because our approach relies on user pictures, only the visible part of this structure will be considered to be part of the ceiling.

The walls, structures orthogonal to the ceiling, connected to both the ceiling and ground base structure, are not considered to be part of the ceiling. On the other hand, the upper wall points do indicate the ceiling boundary and are thus part of the ceiling characteristics. Keeping some wall points would provide a better indication of what the ceiling is. A dilemma with keeping a selection of wall points is deciding which wall points to keep. Depending on the size of the room, the user input will not have many wall points or maybe even no wall points at all. For this reason, keeping too many wall points in the database could lead to incorrect matching. Furthermore, using more points would lead to slower processing. It would also be possible to keep some wall points in both our database and our user input for the cases where the user is more likely to capture wall points. This would require

some indication as to in which situations wall points would be feasible to keep. Furthermore, automatically filtering out wall points will require some classification, currently we lack such semantics.

Generally speaking, all structures that are connected to the ceiling, but do not reach the ground, are considered to be part of the ceiling. This means that rather low hanging object are still considered part of the ceiling, because they strongly contribute to the uniqueness of the signature of each room. A downside of incorporating features that are too low, is that it might not be possible for the user to capture them together with the base ceiling with their camera pointing upwards.

9.2. What ceiling characteristics should be captured?

If a specific room has a very distinct section, you cannot provide pointers to users to capture this specific section, because their location is unknown. Asking the user to capture a ceiling part close to a certain wall or door would also require some form of additional orientation, which is currently not incorporated in our application.

Wall points are difficult to handle and can lead to noise and unfair comparison. To prevent wall points from being captured, the user can get an instruction to try to capture just the ceiling by filming as much in the middle of the room, where no walls are present, as possible.

What part of the ceiling should be captured?

For this approach to work, the part of the ceiling captured by the user should be a good representation for (sub)rooms stored in the database. When using a database which only contains the full ceiling of a room, it could happen that the user captures a smaller subset which does not necessarily match the corresponding database element. A solution would be to divide the room in distinct subsets. However, this would introduce new problems.

The subsets we create cannot be known to the user (e.g."Only capture similar parts of the ceiling." is no ambiguous instruction). Because the user might have a different idea of possible subsets than the person constructing the database. When subsets are used, the user might capture overlapping subsets.
Furthermore, more subsets would lead to a more elaborate database with more similar entries. Less distinctions between database elements means that it is more likely that a wrong room is assigned during the matching process.
For this reason, it would be best to keep the full ceiling as much as possible and only create subsets for very distinct boundaries. The user has to be aware that capturing overlapping subsets causes problems.

Another challenge is when the user is walking the stairs. The camera position extracted from the frames is no longer parallel to the ceiling. It should therefore be further investigated if it possible to generate a point cloud when walking the stairs and if it is possible to match this user input of the ceiling to the database.

Figure 9.1: A possible instruction image to show how users should capture the ceiling point cloud.

What user instructions are necessary to make sure that the right parts are captured?

Ease of use is very important if we want our application to be actually usable [Courage and Baxter, 2005]. If our user needs to specify too many things beforehand (e.g."How many windows does your room have?"), the application can not be considered as an on-the-fly efficient localisation application.

In this research, two possible methods of user input are considered.

1. One in which the user provides a short video with 30 or 60 frames per second of the ceiling from which a point cloud can be generated in a DIM approach. An advantage of using a mobile phone is that the user knows which room and which part of the ceiling is captured, because the user can immediately adapt to the visualisation of the images/video on a screen. For a mobile phone, a possible user instruction can be provided through the figure shown in figure 9.1. Here the user to captures the ceiling from different angles. Capturing different angles between feature points on the ceiling is desirable for the DIM process.

   Starting from the initial position (0, 0, 0), the local coordinates and orientations have to be connected to the frames of the images/video and provided

to the DIM software before the reconstruction. The reason for this, is that the point cloud has be parallel to the x,y-plane to make the z-values from the user input comparable with the database. It is however not explored if this information can be provided from sensors in mobile phones. In addition to this, it is therefore also unknown if this information can be provided under the condition of all possible angles. A stable solution in which the angle is fixed, can be achieved by providing users a stand for their mobile phone.

2. Another option is using an other device to capture point clouds. An example could be a LiDAR scanner. Besides the costs of a LiDAR scanning device, giving specific user input will be more difficult and will therefore require other pre-processing methods to enable indoor localisation. This is not considered to be very likely for "every day " use cases.

9.3. What is a possible method to perform indoor localisation using point clouds of the ceiling?

A possible method is one that, given the available resources and application goal, can help identify the user's initial location. Fingerprinting methods are used to identifying a unique signature consisting of key ceiling features based on geometric properties of points, and the pattern of height information of the points, for given "user" and "database" inputs. After experimenting on various possibilities and identifying the limitations and obstacles in our inception report, the feature matching and histogram matching approaches are considered the most feasible. The reason for this is that these methods are scale invariant and might provide a unique fingerprint on room level.

9.4. What kind of pre-processing is required for these possible methods?

The goal of pre-processing in a fingerprinting approach is to facilitate and improve the comparison between both a "user" and "database" point clouds. By this, we mean to process the point clouds in such a way that orientation, scale, reference system, noise and point density will not affect the outcome of the matching as much as would be the case for raw data. Depending on the used matching method, not all of the aforementioned factors are of equal importance.

For the histogram approach, the following aspects require pre-processing:

- Tilt influencing the z-values.
- Features such as wall points influence the fingerprint of the (sub)room and therefore the matching.
- The database gathered with a LiDAR scanner has a different point density and less noise than the DIM user input. To improve the quality of the user input, noise removal and voxel down sampling can be used.

- The amount of bins used for matching in the histogram approach is based on the amount of input points. If the density of the user input point cloud is equal to the density of the database, the amount of bins will represent the two point clouds better. Voxel down sampling can be used to achieve this. Voxel down sampling can be optimised when the scale of the user input is known.

For the feature matching approach, the following aspects require pre-processing:

- Variation in scale and coordinate systems of DIM inputs and database required rigid alignment for matching.
- To improve the quality of the user input, noise removal and voxel down sampling can be used.
- Input parameters for algorithm such as voxel size, thresholds for pruning and iteration number have to be refined for a particular database and input.
- To ensure that there is realistic scale difference between user input and database, initial transformation can be derived using device sensors.

9.5. Can point clouds from the ceiling be used for indoor localisation purposes on room level?

To answer the main research question, we decided to implement a fingerprinting approach. In this approach, we extract a unique signature from a user input point cloud, generated from pictures through DIM, and from database point clouds. By comparing both signatures, we can find the best match and in this way perform localisation. For point clouds generated from user input images, there are a few difficulties, namely the tilt and wall points are not yet automatically handled.

In this paragraph, we will describe the outcome of the fingerprinting approach and the accuracy. The accuracy of different fingerprinting methods is indicated with the κ-coefficient. The κ-coefficient with DIM input for the fingerprinting approach is 0.17. With the chance component, the accuracy is 25 %. This accuracy is not high enough to perform high thrust worthy localisation using input pictures. The matching did improve using LiDAR input. The κ-coefficient was 0.47. With the chance component, the accuracy is 53 %, which is also relatively low. This might be explained with the fact that in some cases, a shift in height values occurs as for example displayed in figure 8.4, but why this happens is not clear.

The results of the feature matching approach are better for the LiDAR input, but not for the DIM input. The κ-coefficient for the DIM user input is 0.11. With the chance component, the accuracy is 20 %. This is not that high. For the LiDAR user input, the κ-coefficient is 0.93. With the chance component, the accuracy is 94 %.

This means that using point clouds from the ceiling for indoor localisation purposes would be promising in a feature matching approach provided that the user could capture LiDAR input and the ceilings are distinct as in the Architecture building. For a DIM user input the results are less promising, this might have to do with the different characteristics of the input point cloud and the database point cloud and the fact that the captured subsections might not be representative for a full room.

10 Conclusion and recommendations

This is a concluding chapter explaining the scientific and technical implications for society of the research findings described in this report in considerable detail.
As discussed in chapter 8, our basic solution needs to be further adapted. Recommendations will be given for this.

10.1. Conclusion

The aim of our project is to answer the following main research question:

Can point clouds from the ceiling be used for indoor localisation purposes on room level?

To come to an answer for this question, a fingerprinting approach is. In such an approach, an identifiable signature is extracted from a user input point cloud which is generated from pictures through DIM and database point clouds. Indoor localisation is implemented by comparing both signatures and selecting the best match.

In the first step, the pre-processing of point clouds from DIM is performed. As shown in the results, tilt removal, down sampling and noise removal of point clouds are required for the further matching method. Handling tilt is conducted by providing camera parameters before the reconstruction of DIM. The results show that the point clouds after tilt removal is more in line with the x,y-plane. Other pre-processing steps involve down sampling in a voxel grid and noise removal. In order to see the effect of both down sampling and noise removal, inputs after different steps of down sampling and noise removal are tested in the NumPy histogram approach, the best results appear to be voxel noise down sampling.
An alternative for DIM could be to use the point clouds from LiDAR as user input. In this way, some challenges introduced by the DIM user input can be solved.

The histogram approach is one of possible signature matching methods which has three different implementations namely the NumPy approach, OpenCV approach and the SciPy approach. The histograms are generated from the normalised height and occurancy values. The DIM user input and laser user input are tested, the NumPy approach with the `Chi-squared` distance measure turned out to be the most thrust worthy histogram matching in both cases. This histogram matching is a simple one-dimensional approach, it is relatively robust and to some degree

insensitive to outliers. On its own, the histograms approach is not thrust worthy enough to lead to indoor localisation.

Another possible way of fingerprinting matching is feature matching, which uses the geometric property of points to get the similarity between input and database point clouds. An advantage of this feature matching is that it takes into consideration, the fitness of local surfaces of ceiling with respect to geometry and topology to detect correct location. It can also align the point clouds roughly before the actual matching if they are have different scale, rotation and translation as in the case of DIM. Feature matching is performed by detecting the correspondences by querying the nearest neighbour in the 33-dimensional FPFH feature space and fitting local surfaces by performing transformation based on key correspondence pairs. For a LiDAR input the feature matching provides trust worthy results for indoor localisation.

It is also possible to combine the outcome from different fingerprinting methods, and therefore the separate methods can support each other if necessary.

10.2. Recommendations

The recommendations deducted from the research described in this report can be divide into several categories :

Database point cloud

- Use of the more precise 100 % point cloud instead of 9 % as database might make a minor difference. However, especially for the feature matching the results with the LiDAR input are already of high quality.
- Investigate to what extent distinct ceilings are present in different buildings than the Architecture building.

Handling tilt

- To handle tilt in all possible use cases, further testing of the theoretical setup described in this report is required.
- Investigate how the theoretical setup can be linked to sensors available in mobile devices.
- Investigating how incorporating the obtained scale and orientation will influence our, and future methods.

Dense image matching

- Investigate which DIM software could generate a usable point cloud when providing the local coordinates and orientation, and under which conditions this is possible.
- Test the limits of the DIM approach between needed overlap of input frames and the processing speed point cloud.

- Investigate which area/subset of the ceiling must be captured for successful indoor localisation.
- Test automation of this process within a complete application.

Wall points

- Investigating how much wall points (or curtains as shown in figure 5.10) are likely to be captured in the user input and adapting the database to this.
- Deal with the classification of ceiling points and only keep the points classified as ceiling. This way, ceiling point extraction can be automated. A possible way to classify ceiling points makes use of the MAT.

Matching methods

- Look into more advanced ways of histogram comparison than simply using the distance between bins. An example could be to consider the use of the variable bin size distance described by Ma et al..
- Incorporate advanced techniques of point cloud structure simplification and registration methods for optimum comparison in features provided in [Zhou et al., 2018] and [Rusu et al., 2009].
- Investigate new methods of combining the outputs of different fingerprinting approaches. Different weights and different solution than the ones described in section 6.3.3 can be applied to improve the combined matching.

Incorporating supporting information

- Test how incorporating dead reckoning, trajectory information and topological information could improve indoor navigation. The trajectory information from for instance a mobile laser scanner could be used to built a navigational graph with the (sub)rooms as different nodes. Using this navigational graph would require an initial user location, but could be used after that for comparing the user input with specific sections of the database.
- Test if a certain range provided by GNSS could be used for comparing the user input with specific sections of the database.

11
Reflection

In this section we will reflect on the project described in this report. We will do this based on the expectations as listed in chapter 3. In this chapter, we specified what we wanted to achieve using the MoSCoW method (section 3.1), we will comment on these MoSCoW statements as part of the reflection. We will also assess the fitness for use of our approach for different types of users. Finally, we will assess the privacy issues involved with indoor localisation methods such as ours.

During this project, we wanted to consider the separate steps of a specific indoor localisation approach using point clouds of the ceiling. We managed to do this. By combining the separate steps of our work flow, we are now able to indicate under what conditions, it will be feasible to perform indoor localisation using point clouds. This means we fulfilled the aim we set for this project. We expected some challenges on the way, and these challenges indeed happened. Every step we took, we discovered new things we had to figure out. Because of this, we did not manage to achieve a fully automated application, this was in line with the expectations. We feel like we handled the challenges on our way well, we performed quite elaborate testing with many rooms and looked into the basics of a completely new indoor localisation approach. Future research on indoor navigation and localisation can benefit from our findings.

11.1. MoSCoW reflection

In our Inception report, we provided an overview of the things we would have liked to achieve during this project by using the MoSCoW method [Mulder, 2017]. In this section, we added some comments to the MoSCoW overview, to assess what we have achieved.

Must:

- See if pictures from the ceiling provided by users can be used for localisation (are they differentiating enough). For this we will investigate in which way (e.g. which subsection of the ceiling, pictures or videos etc.) the users should provide their input to acquire a feasible result. We will also need to look at the quality of the localisation. **The accuracy of different fingerprinting methods using DIM user input, is shown in figure 7.23. Within the fingerprinting methods we attempted, this accuracy is not high enough to perform high thrust worthy localisation using input pictures. There are also experiments conducted using LiDAR input, the results of these experiments are shown in figure 7.21. Depending on**

the application and type of use, the results of the feature matching fingerprinting approach can be of high enough quality for localisation.

- Extracting the (distinct) signature of a point cloud. **We found several ways in which the signature of a point cloud can be extracted. The signature could be a histogram of the height values of the points, or even the points in itself.**

Should:

- Compare the performance of different software packages for DIM, such as COLMAP, Pix4D, Archicad and VisualSFM, to see which software package provides the most suitable results for our approach. **As listed in our inception report, we performed several tests with DIM. Pix4D provided good results and was user friendly.**
- Accurate localisation based on room level by using additional information like dead reckoning combined with topology information or the scanner trajectory. **This is given as a recommendation for further research.**
- Implement a minimal user interface (prototype). **This could be done when dead reckoning is incorporated and gives plausible results.**

Could:

- Perform automatic classification of ceiling points from the point cloud. For this we could for example use the MAT, or look at intersections between a line perpendicular to scanner trajectory and fitted planes planes to discover a possible ceiling height at which we can extract points. **For automation of wall point removal, this is given as a recommendation for further research.**
- Incorporating door, window and furniture classification from the point cloud to improve the efficiency and accuracy of localisation. **This is not considered to be part of this research anymore.**
- Incorporating machine learning; Include user measurements in a database to make future localisation more accurate and more efficient (crowd sourcing). **This is not considered to be part of this research anymore.**
- Create a web platform and application for taking input and sending data to a server, process it and show the location of the user on a map. **This could be done in future research, provided that dead reckoning is incorporated and gives plausible results.**
- Look into different necessities for different user types (e.g. firefighters need different things than students). **See section 11.2.**

Would not:

- Incorporate Wi-Fi fingerprinting.
- Look at navigational graph between rooms for navigational purposes.
- Look at different buildings than the architecture building.

11.2. Suitability for use

Our solution for indoor localisation is only relevant if it caters for the different needs of possible use cases. For this reason, it is important to indicate whether or not our implementation can lead to a realistically usable solution.
In general, ease of use, processing time and accuracy are all important factors that determine if an application is usable. Most users prefer an application that is easy to use. Users cannot be expected to be active part of the localisation, but some users might be willing to share their own input data to enrich our database [Kaasinen, 2002]. However, whether or not an application is easy to use, has an acceptable processing time or accuracy, strongly depends on the use case. In general users for example like to use a fast application. However, for certain user groups, run time can be of bigger influence than for the "general" user. Depending on the amount of (sub)rooms in the database, the fingerprinting process can become slow and thus unsuitable for applications which rely on fast run times.

To determine what possible user groups could need, we developed a few user categories which could benefit from indoor localisation. Based on these user groups, we will list a few requirements to indicate whether or not our approach needs to be adapted to be suitable for that specific type of use. This is summarised in figure 11.1.

- Civilians: They want to find their way inside a public building under everyday conditions. Examples can be students looking for the room they have an exam or lecture, or people having an appointment at a specific location in a hospital. Time can be of influence, but in general slightly longer processing times would be acceptable (e.g. matter of minutes). Accuracy is important, because without a certain degree of accuracy, a user would not be tempted to use an application implementing our solution. User equipment can also be a problem, elderly people might not have a smart phone, and most users will certainly not have a LiDAR scanner device available. However, this type of user can be expected to have time and opportunity available to take out their mobile phone, provide data and wait for a response. Especially when provided clear instructions, the conditions will be suitable for our implementation.
- Staff: They need to find their way inside their work-space. New personnel of big organisations might not know their way around the office straight away. Until they do, indoor localisation can be a solution for them. Often, they have the same requirements as the civilian, however, they might have a little less time on their hands meaning that a quicker solution is preferred.

- Emergency managers: They need to find their way inside a building under potentially hazardous conditions. Examples can be firemen who needs to save someone from a burning building, policemen who need to chase a suspect or a paramedic needing to get to a patient. Time is a big factor, the faster the application works, the better. Coding in C++ and implementing indexing such as KD-trees and clustering of the point clouds can improve the speed of an application implementing our solution [van Oosterom and Quak, 2018]. Accuracy is crucial, mistakes cannot be afforded. Next to the safety of others, the main concern for emergency workers would be their own safety [Nilsson et al., 2014]. If the application cannot guarantee this safety, it cannot be used. The user will probably not have the possibility to physically hold a mobile device to provide input. This might be solved by attaching the device to the equipment of the user, such as the helmet of a firefighter. Furthermore, the equipment to capture input can be a problem. Under hazardous conditions, smart phone cameras might not work due to for example smoke development. This might be solved by using a Sound Navigation Ranging (SONAR) based input. Something that cannot be easily solved, is a situation in which the changes in environment caused by the hazardous situation, might not match our database point cloud anymore.

- Robots: In robotics indoor navigation, and thus localisation, is often required. An example could be the SPENCER project, this is "a European Union-funded research project that advances technologies for intelligent robots that operate in human environments." [Verbree, 2018], [Lilienthal and Arras, 2015] Here robots are used to help people find their way on Schiphol Airport. A possible implementation of our solution in this field, could be to use a underlying navigational graph containing different locations as nodes. The robot performs on the fly localisation to locate itself on the reference graph. The importance of processing time and accuracy depend on the type of use for the robot. Ease of use is less of an issue because the robot will consequently follow the programmed instructions. A LiDAR scanner mounted on the robot can be used to continuously capture the ceiling.

An application implement our approach requires a certain level of user capabilities and user effort. The user needs to be equipped with a modern mobile device with a camera and a network connection to link to the point cloud database and to connect to a server in which processing can be performed. The quality of the user input will determine the quality of the localisation result. If the user input is a long video, with many overlapping frames which capture the ceiling from different angles at high quality, the resulting point cloud will be of better quality. However, the high amount of frames could also lead to slow processing. This means there will be a trade-off between quality and speed, which is a drawback.

The implementation we came up with, can be considered to be a first building block for more specific applications that would need indoor localisation. Indoor navigation for example, would require an indoor localisation solution to build upon.

Figure 11.1: Assessing the importance of certain requirements of an indoor localisation method for different types of use. For staff and civilians, only DIM input will be feasible, which means the accuracy and processing time will be limited with our approach and might not suit their needs. For the emergency manages, LiDAR or SONAR input can lead to suitable results. Use cases with a robotics user do not require ease of use, but the desired accuracy and processing time depends on the use case of the robot.

For future research we would recommend to further look into the requirements of different potential user groups. This can be done through for example interviews, surveys and wants and needs analyses of the desired user groups. The results from such a study can lead to more in-depth user profiles, personas and scenarios, each with their own requirements [Courage and Baxter, 2005]. Based on these requirements, it would be possible to adapt the basic approach we came up with into a suitable solution for a specific user group.

11.3. Privacy

Indoor localisation incorporates the use of personal data according to the European data protection directive (95/46/EC) [European Commission, 1995]. For an indoor localisation application, privacy concerns could mainly be raised in relation to "tracking". If an application for example requests access to the user's location, and it is possible to identify the user, it is possible to match the location request with the user's identity. Put in other words, the indoor location of a natural person could lead to the identification of this person, and thus can be considered to be an intrusion on the privacy of our user [van Loenen, 2019].
However, users might not be aware of their own privacy [Kaasinen, 2002], for an application we need to take into account certain privacy measures. It is important to notify our user that their location is privacy sensitive information, ensuring informed consent. For an application using our methods, possible data that the users might need to share could be:

- The specifications of their personal mobile device such as camera specifications.

- Data derived from sensors and third party applications such as a compass application, applications counting steps etc.
- The input for they provide to the localisation application.

Making the application privacy preserving, would mean we are at least required to notify the users about all aspects of the application that could possibly endanger their privacy Nijhawan et al. [2013]. This means we need to notify the user regarding to:

- The types of data that are going to be used.
- The way these data are going to be processed.
- Which of these data are going to be shared with other applications and how.
- Provide the user with a privacy agreement that will incorporate the aforementioned information.

However, informing the user is the very least we can do. We also try to anonymise the information during our processes as much as possible, so that it is no longer possible to match the identity of our user to their location. This can for example be done by identifying users under a number instead of a name, and by changing this number randomly for every independent use.

References

Bot, F., Nourian, P., and Verbree, E. (2019). A graph-matching approach to indoor localization using a mobile device and a reference bim. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XLII-2/W13:761–767. https://doi.org/10.5194/isprs-archives-XLII-2-W13-761-2019.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 120:122–125.

Bresson, G., Alsayed, Z., Yu, L., and Glaser, S. (2017). Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2:3:194–220. ISSN-2379-8858.

Brezmes, T., G. J. and Cotrina, J. (2009). Activity recognition from accelerometer data on a mobile phone. https://link-springer-com.tudelft.idm.oclc.org/content/pdf/10.1007%2F978-3-642-02481-8.pdf.

Broersen, T., Fichtner, F., Heeres, E., De Liefde, I., Rodenberg, O., Meijers, B., Verbree, E., Van der Spek, S., and Ten Napel, D. (2015). Project pointless: Identifying, visualising and pathfinding through empty space in interior point clouds using an octree approach. *TU Delft: Student report*. http://resolver.tudelft.nl/uuid:c9e55f6a-c874-4aee-9ceb-6bbcf34d9dc7.

Choi, S., Zhou, Q., and Koltun, V. (2015). Robust reconstruction of indoor scenes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 10.1109/CVPR.2015.7299195.

Conesa, J., Antoni, P.-N., Joaquín, T.-S., and Montoliu, R. (2018). *Geographical and fingerprinting data for positioning and navigation systems: challenges, experiences, and technology roadmap*. Academic Press, an imprint of Elsevier. ISBN-9780128131909.

Courage, C. and Baxter, K. (2005). Understanding your users: A practical guide to user requirements methods, tools, and techniques. ISBN-13-978-1558609358.

Eckart, B., Kim, K., and Kautz, J. (2018). Fast and accurate point cloud registration using trees of gaussian mixtures. *arXiv:1807.02587*.

European Commission (1995). Directive 95/46/ec on the protection of individuals with regard to the processing of personal data and on the free movement of such data. Retrieved from: https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX%3A31995L0046 at 24-06-2019.

Faragher, R. and Harle, R. Location fingerprinting with bluetooth low energy beacons. *IEEE Journal on slected areas in communication*, 33-11:135. 10.1109/JSAC.2015.2430281.

Flikweert, P. (2019). Automatic extraction of an indoorgml navigation graph from an indoor point cloud. *TU Delft: master thesis*. http://resolver.tudelft.nl/uuid:b11f5b57-5362-4b45-bed6-d5bc154d86aa.

Freedman, D. and Diaconis, P. On the histogram as a density estimator: L2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 57:4:453–476. https://doi.org/10.1007/BF01025868.

Gagunashvili, N. (2009). Chi-square tests for comparing weighted histograms. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 614:2:287–296. https://doi.org/10.1016/j.nima.2009.12.037.

Huang, Y., Wang, H., Zhan, K., Junqiao, Z., Popo, G., and Feng, T. (2015). Image-based localization for indoor environment using mobile phone. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-4/W5:793–800. 10.5194/isprsarchives-XL-4-W5-211-2015.

Jones, B. and Aoun, M. (2009). Learning 3d point cloud histograms. CS229 Machine Learning Project.

Jones, E., Oliphan, T., and Peterson, P. (2001). Scipy: Open source scientific tools for python. http://www.scipy.org/.

Kaasinen, E. (2002). User needs for location-aware mobile services. *Personal and Ubiquitous Computing*, 7:1:70–79. https://doi.org/10.1007/s00779-002-0214-7.

Khallaghi, S. (2017). Pycpd: Tutorial on the coherent point drift algorithm. http://siavashk.github.io/2017/05/14/coherent-point-drift/, Retrieved on: 21-06-2019.

Kwolek, B. (2004). Finding location using a particle filter and histogram matching. ISBN-978-3-540-24844-6.

Ledoux, H., Peters, R., and Arroyo Ohori, K. (2018). Lecture notes on digital terrain modelling (geo1015).

Lemmens, M. (2018). Lecture notes on sensing technologies (geo1003).

Lemmens, M. (2019). Lecture notes on data quality (geo1008).

Lightvoet, B. (2017). Crowdsensing as a tool for up-to-date road asset distress detection. https://repository.tudelft.nl/islandora/object/uuid:61924097-f0f7-4d5a-979c-fe991017a3ce?collection=education.

Lilienthal, A. J. and Arras, K. (2015). About spencer. Retrieved from http://www.spencer.eu/project.html at 21-06-2019.

Ma, Y., Gu, X., and Wang, Y. Histogram similarity measure using variable bin size distance. *Computer Vision and Image Understanding*, 144:981 – 989. https://doi.org/10.1016/j.cviu.2010.03.006.

Mahmoudi, M. and Sapiro, G. (2009). Three-dimensional point cloud recognition via distributions of geometric distances. https://doi.org/10.1016/j.gmod.2008.10.002.

Mautz, R. (2012). *Indoor positioning technologies*. Schweizerische Geodätische Kommission. ISBN-978-3-908440-31-4.

Mulder, P. (2017). Moscow method: for setting requirements by order of priority. Retrieved from https://www.toolshero.com/project-management/moscow-method/ at 27-04-2019.

Myronenko, A. and Song, X. (2009). Point-set registration: Coherent point drift. *CoRR*, abs/0905.2635. http://arxiv.org/abs/0905.2635.

Nijhawan, L., Janodia, M., Muddukrishna, B., Bhat, K., Bairy, K., Udupa, N., and Musmade, P. (2013). Informed consent: Issues and challenges. 10.4103/2231-4040.116779.

Nilsson, J., Rantakokko, J., Händel, P., Skog, I., M., O., and Hari, K. (2014). Accurate indoor positioning of firefighters using dual foot-mounted inertial sensors and inter-agent ranging. *Position, Location and navigation Symposium (PLANS)*. 10.1109/PLANS.2014.6851424.

Oliphant, T. (2006). NumPy: A guide to NumPy. USA: Trelgol Publishing. http://www.numpy.org/.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830.

Pirouz, N. e. a. (2016). Voxelization algorithms for geospatial applications: Computational methods for voxelating spatial datasets of 3d city models containing 3d surface, curve and point data models. *MethodsX*, 3:69–86. (http://www.sciencedirect.com/science/article/pii/S2215016116000029).

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. *IEEE International Conference on Robotics and Automation*, pages 3212–3217. 10.1109/ROBOT.2009.5152473.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Staats, B. (2017). Identification of walkable space in a voxel model, derived from a point cloud and its corresponding trajectory. *TU Delft: master thesis*. http://resolver.tudelft.nl/uuid:6a827a88-ce09-43f1-adb9-da886448a1fc.

van Loenen, B. (2019). Lecture notes on geo-information organisation and legislation (geo1009).

van Oosterom, P. and Quak, W. (2018). Lecture notes on geo-database management systems (geo1006).

Verbree, E. (2018). Lecture notes on positioning and location awareness (geo1003).

Xia, S., Liu, Y., Yuan, G., Zhu, M., and Wang, Z. (2017). Indoor fingerprint positioning based on wi-fi: An overview. *ISPRS The International Journal Geo-Information*, 6:135. 10.3390/ijgi6050135.

Zhang, S., Wu, X., and You, Z. (2017). Jaccard distance based weighted sparse representation for coarse-to-fine plant species recognition. *PloS one*. https://doi.org/10.1371/journal.pone.0178317.

Zhou, Q., Park, J., and Koltun, V. (2018). Open3d: A modern library for 3d data processing. *arXiv:1801.09847*.

A
Appendix

A.1. Comparison between the different pre-processing methods

In this part of the appendix examples from the comparison between the different down sampling methods will be displayed as well as results of the combined methodology used to process the input files for the matching of the point clouds. The comparison for the methods was applied mainly on the user database that was collected using a mobile phone. The comparison is shown in the images[1] that follow.

[1] All displayed images in pages 99 - 103 have axis z facing upwards

Processing time for all methods on User DIM database				
Input name	total points	Loading time	Processing time(seconds)	
			Avg	Noise removal
Berlage Hall 1 (Input a)	485233	1.759	3.37	6.43
Berlage Hall 1 (Input b)	453669	1.63	3.18	4.49
Berlage Hall 2 (Input a)	869475	3.18	5.85	8.89
Berlage Hall 2 (Input b)	721583	2.56	4.81	7.51
Hallway Berlage (Input a)	711199	2.71	5.11	7.77
Hallway Berlage (Input b)	560679	1.96	3.93	5.28
BG.WEST.370 (Input a)	497626	1.78	3.53	4.74
BG.WEST.370 (Input b)	190077	0.7	1.4	2.04
BG.WEST.370 (entrance) (Input a)	79307	0.27	0.65	0.86
BG.WEST.370 (entrance) (Input b)	21112	0.09	0.72	44.89
Room B (Input a)	1008093	3.77	6.84	10.81
Room B (Input b)	407406	1.59	2.89	3.85
Geolab (Input a)	661160	2.33	5.02	7.07
Geolab (Input b)	386561	1.36	3.21	4.04
BG.EAST.430 (Input a)	992434	3.82	6.96	9.91
BG.EAST.430 (Input b)	319180	1.11	2.44	3.31
Hallway room B (Input a)	624799	2.28	4.18	5.9
Hallway room B (Input b)	470461	1.68	3.33	4.54
Hallway Geolab (Input a)	501694	1.75	3.5	4.96
Hallway Geolab (Input b)	114306	0.38	0.82	1.16

Figure A.1: Table with processing times for applied methods.

Point reduction with all methods on User DIM database				
Input name	Points			
	Avg		Noise removal	
	Output points	Points removed	Output points	Points removed
Berlage Hall 1 (Input a)	5503	479730	344964	140269
Berlage Hall 1 (Input b)	5330	448339	362829	90840
Berlage Hall 2 (Input a)	5641	863834	792578	76897
Berlage Hall 2 (Input b)	2557	719026	654653	66930
Hallway Berlage (Input a)	12106	699093	506367	204832
Hallway Berlage (Input b)	1973	558706	420949	139730
BG.WEST.370 (Input a)	2761	494865	369844	127782
BG.WEST.370 (Input b)	2828	187249	159963	30114
BG.WEST.370 (entrance) (Input a)	4072	75235	77557	1750
BG.WEST.370 (entrance) (Input b)	20689	423	21112	0
Room B (Input a)	1693	1006400	932919	75174
Room B (Input b)	1267	406139	330883	76523
Geolab (Input a)	6727	654433	526239	134921
Geolab (Input b)	6109	380452	275165	111396
BG.EAST.430 (Input a)	2606	989828	904918	87516
BG.EAST.430 (Input b)	3253	315927	269901	49279
Hallway room B (Input a)	3057	621742	530714	94085
Hallway room B (Input b)	2554	467907	392593	77868
Hallway Geolab (Input a)	2965	498729	452838	48856
Hallway Geolab (Input b)	1205	113101	97050	17256

Figure A.2: Table with the number of points reduced after applying down sampling with the additional noise filtering.

■ : Raw database

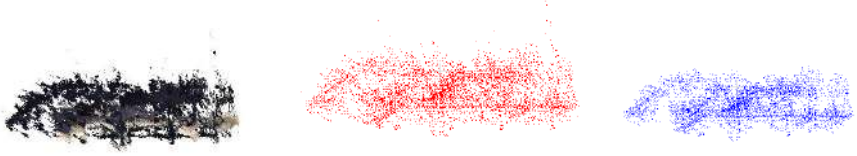
■ : Average down sampling

■ : Average down sampled point cloud after noise removal

Database – Berlage Hall 1

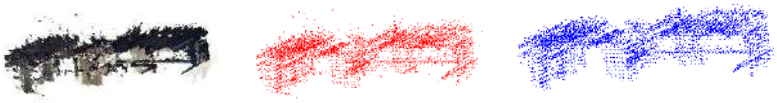
User Input a

User Input b

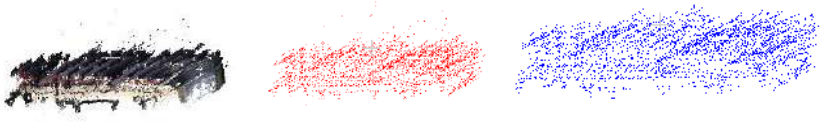


Database – Berlage Hall 2

User Input a

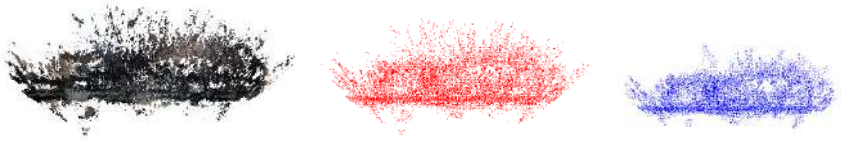


User Input b

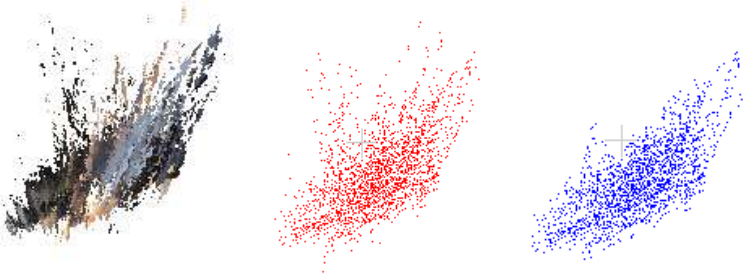


Database – Hallway Berlage

User Input a

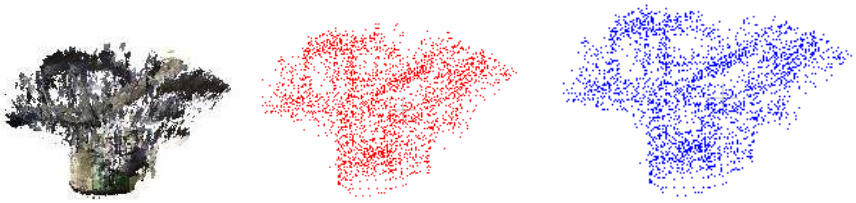


User Input b

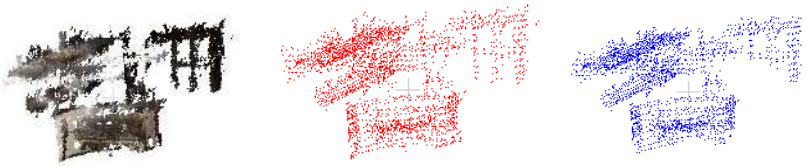


Database – BG.WEST.370

User Input a

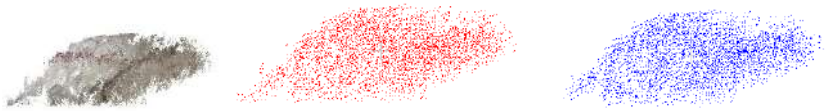


User Input b

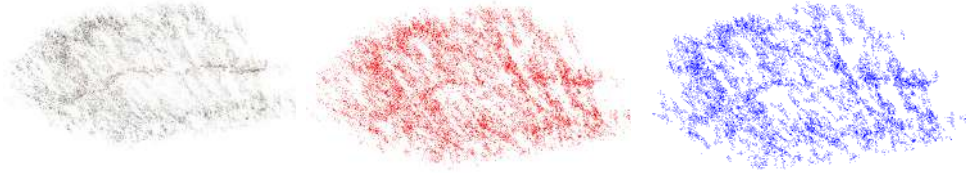


Database – BG.WEST.370(entrance)

User Input a



User Input b



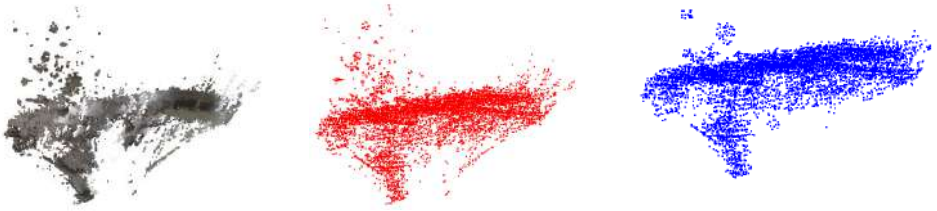
Database – Room B

User Input a

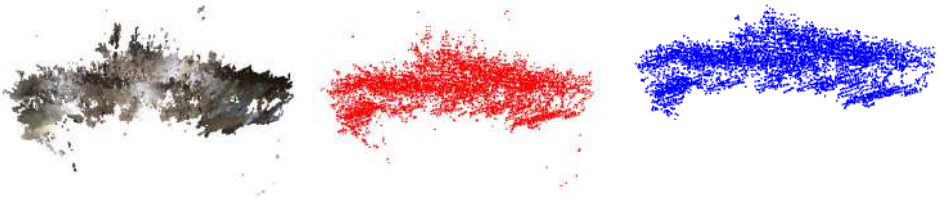
User Input b

Database - Geolab

User Input a

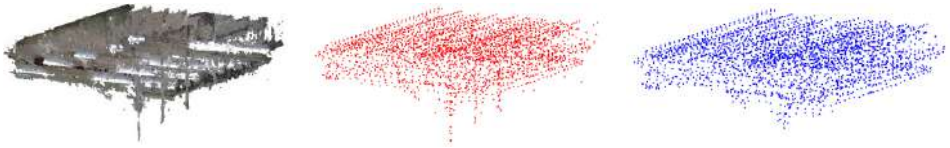


User Input b

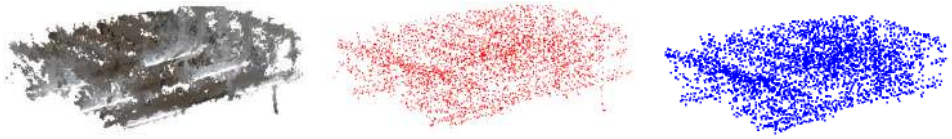


Database – BG.EAST.430

User Input a



User Input b



Database – Hallway room B

User Input a

User Input b

Database – Hallway Geolab

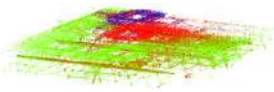
User Input a

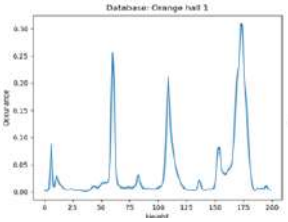
User Input b

A.2. Visual comparison between database and user histogram

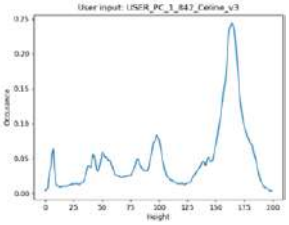
In this part of the appendix, a visual comparison between the database and user input histograms and their corresponding point clouds is provided.
The images might be a little too small to read on a print version. We still choose for this setup because we felt like it gave to most clear overview for comparison's sake. Regardless, it would be nice to maybe print this appendix on a larger scale or use a digital version to zoom in. The height values are plotted on the x-axis, and the occurrence values are plotted on the y-axis.

Database - Orange hall

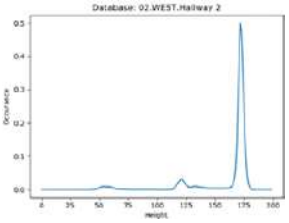




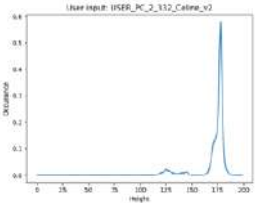
User input 1



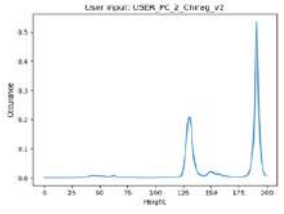
Database - 02.WEST.Hallway



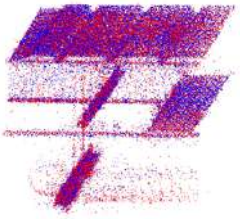
User input 1

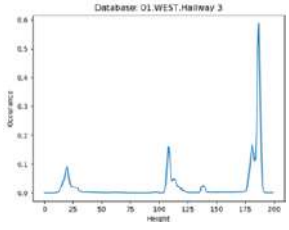


User input 2

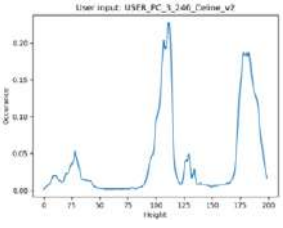


Database - 01.WEST.Hallway

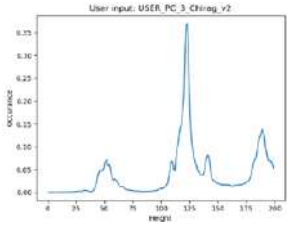




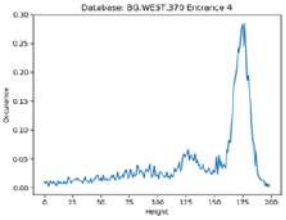
User input 1



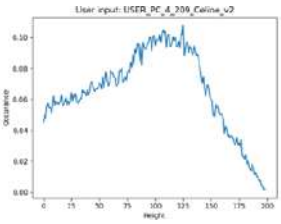
User input 2



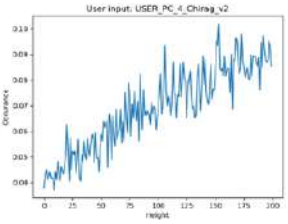
Database - BG.WEST.370 Entrance



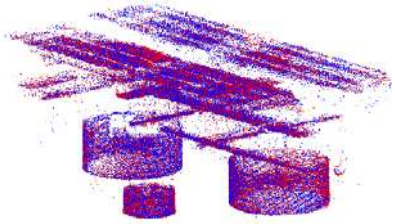
User input 1

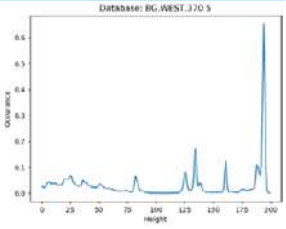


User input 2

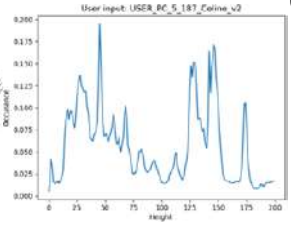


Database - BG.WEST.370

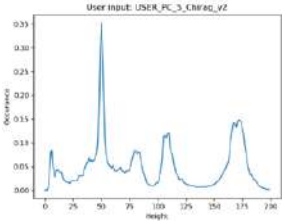




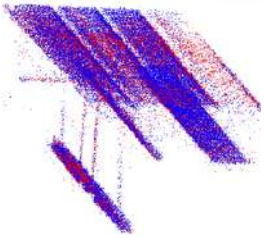
User input 1

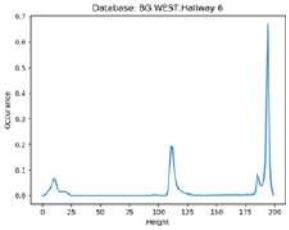


User input 2

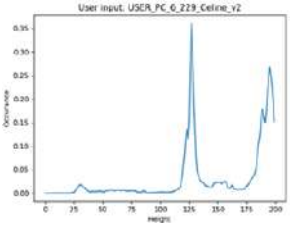


Database - BG.WEST.Hallway

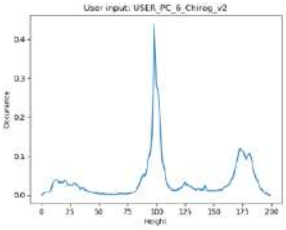




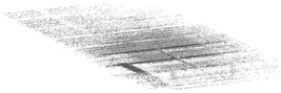
User input 1

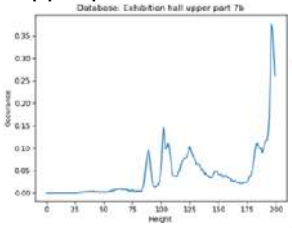


User input 2

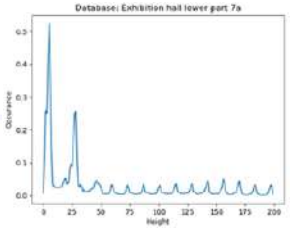


Database - Exhibition hall upper part

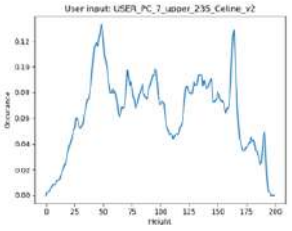




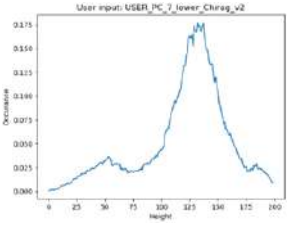
Database - Exhibition hall lower part



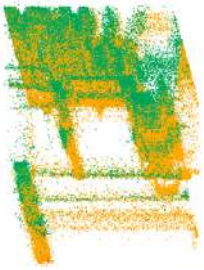
User input

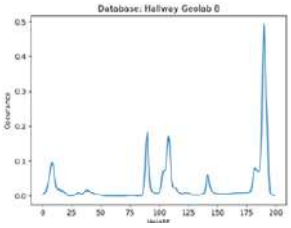


User input

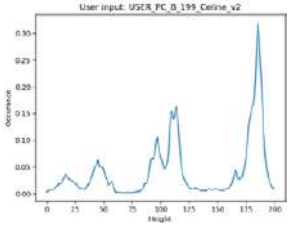


Database - Hallway Geolab

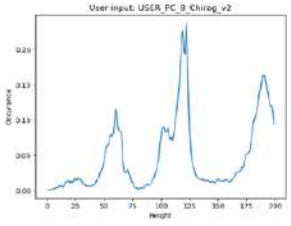




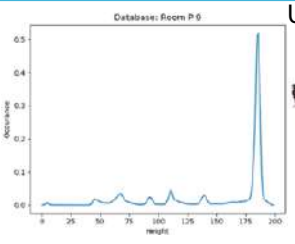
User input 1



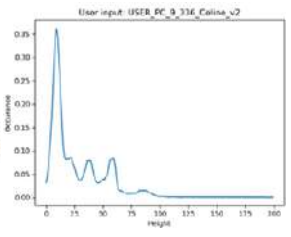
User input 2



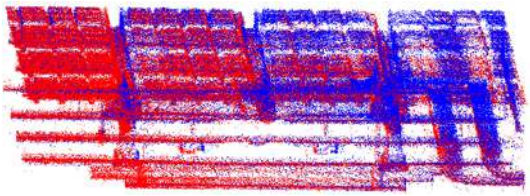
Database - Room P

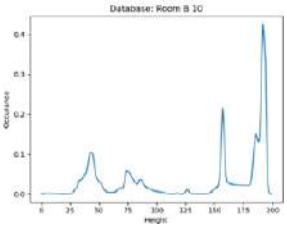


User input

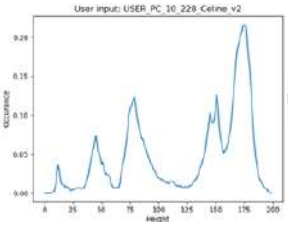


Database - Room B

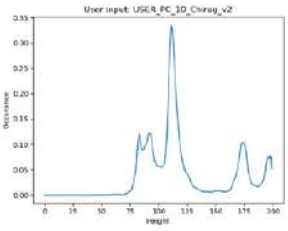




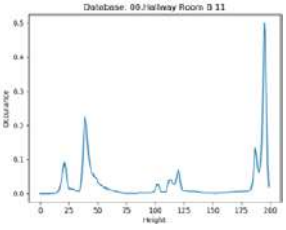
User input 1



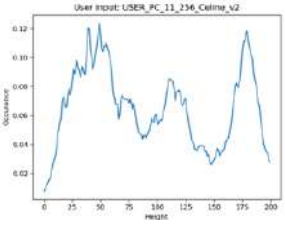
User input 2



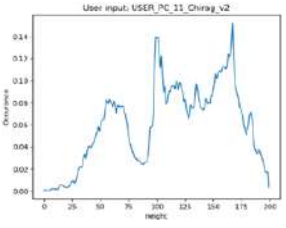
Database - 00.Hallway Room B



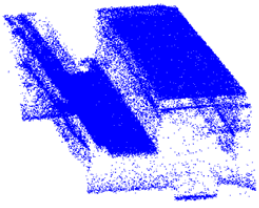
User input 1

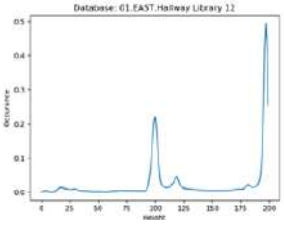


User input 2

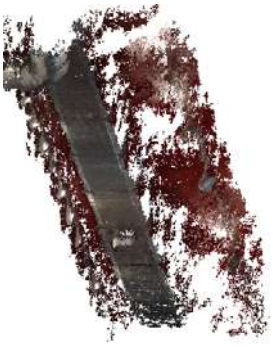


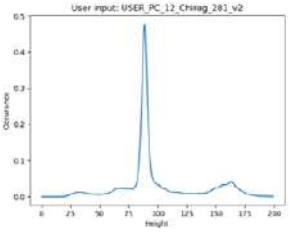
Database - 01.EAST.Hallway Library



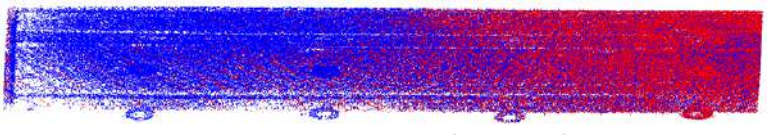


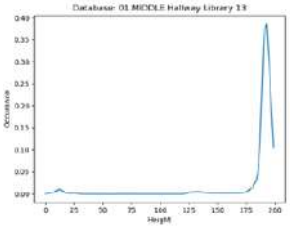
User input



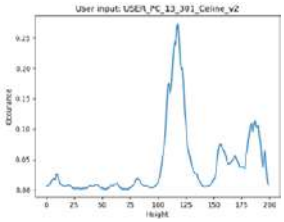


Database - 01.MIDDLE.Hallway Library

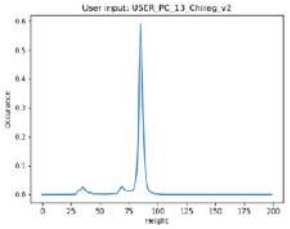




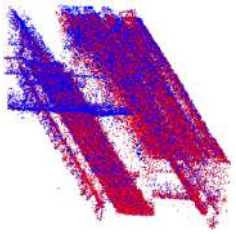
User input 1

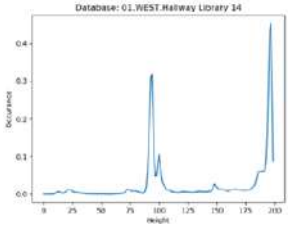


User input 2

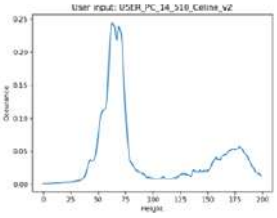


Database - 01.WEST.Hallway Library

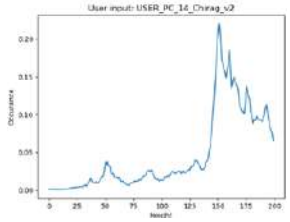




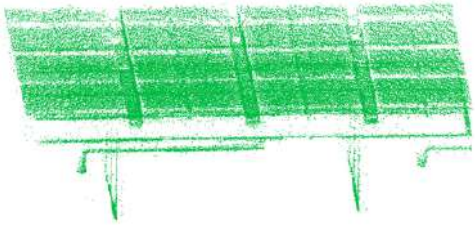
User input 1

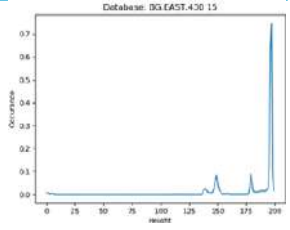


User input 2

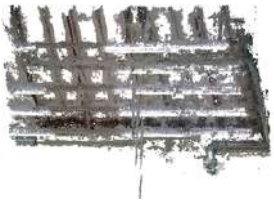


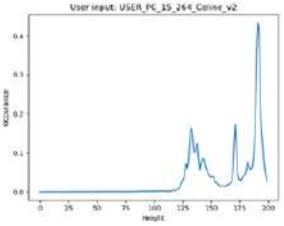
Database - BG.EAST.430



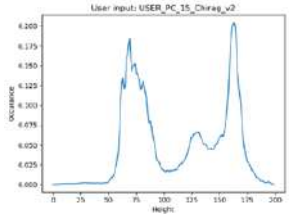


User input 1





User input 2



A.3. Visual comparison between database and user point cloud features

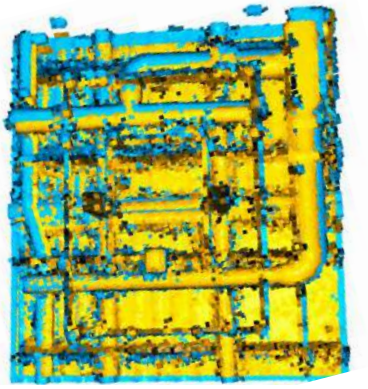
In this appendix, a visual comparison between the database and user input point clouds along with top 3 locations is provided for some input cases of the feature matching. The screen shots have been provided with the best view of alignment of ceilings. In all figures, blue colour represents the database, while yellow represents the user input cloud as depicted in figure A.3.

Figure A.3: Legend for localisation based on feature matching results

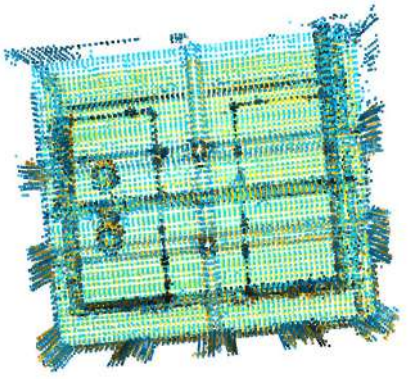
Database 1 – Berlage 1

Top 3 Locations	fitness	rmse	time(sec)
1	*0.99*	*0.088*	*7.06*
6	*0.89*	*0.137*	*11.41*
2	*0.75*	*0.155*	*8.55*

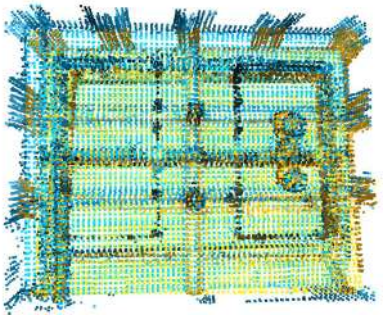
User input 1 a/b



Database 2 – Berlage 2



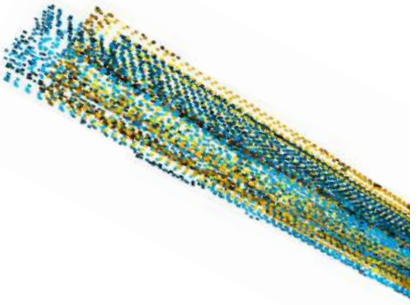
User input 2a



User input 2 b/c

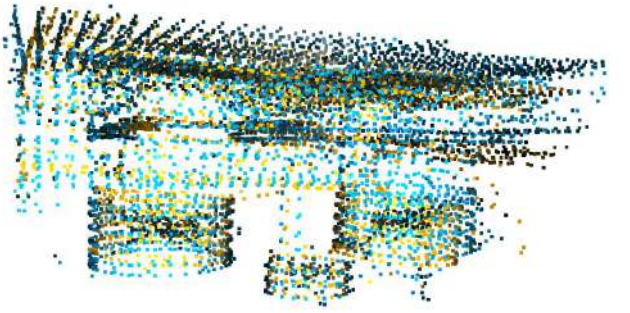
Top 3 Locations	fitness	rmse	time
2	*1*	*0.08*	*3.36*
6	*0.90*	*0.27*	*3.36*
3	*0.26*	*0.31*	*2.11*

Database 3 – Berlage Hall



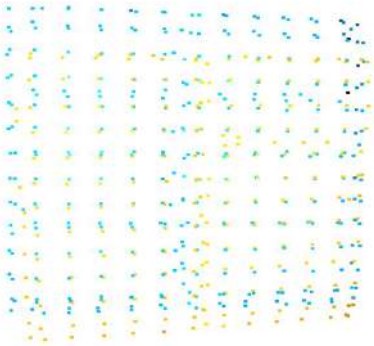
User input 3 a/b

Database 4 – BG.WEST.370 (Coffee Corner)

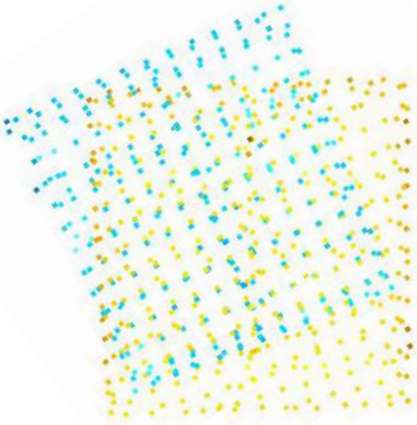


User input 4a/b/c

Database 5 – BG.WEST.370 Entrance

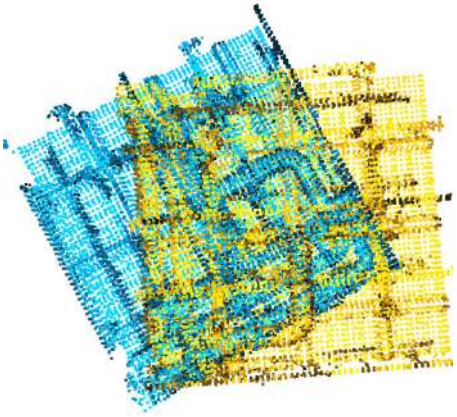


User input 5 b/c

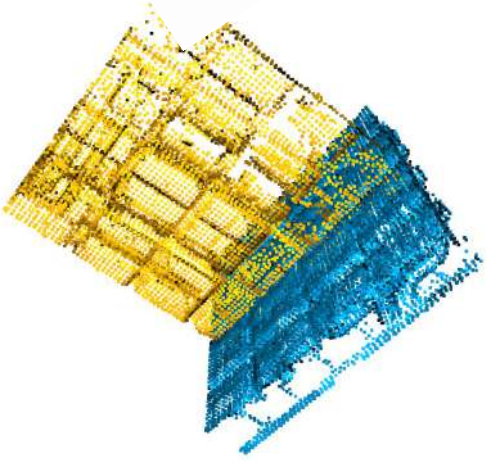


User input 5 a

Database 6 – Room B

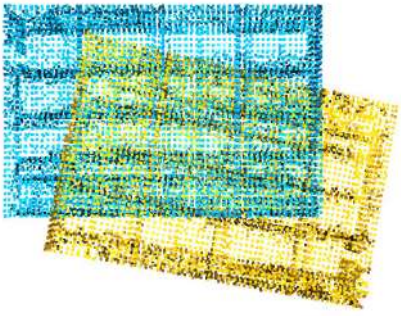


User input 6 a



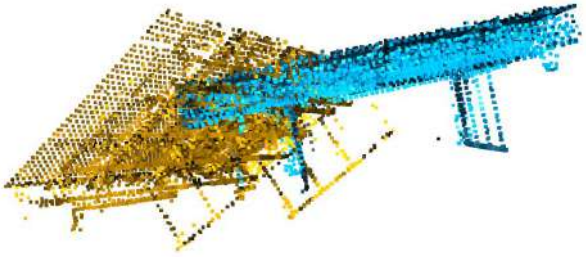
User input 6 e

Database 7 – Geolab



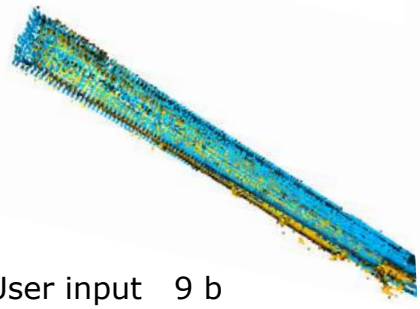
User input 7b

Database 8 – BG.EAST.430



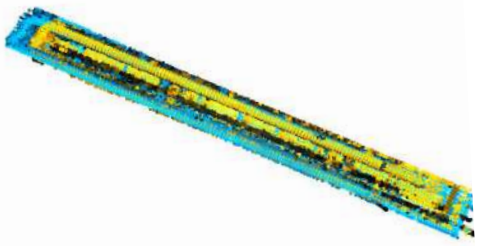
User input 8a

Database 9 – 00.Hallway Room B



User input 9 b

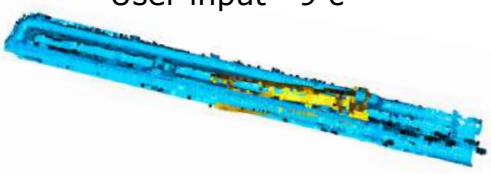
Top 3 Locations	fitness	rmse	time(sec)
9	0.99	0.21	2.13
10	0.95	0.34	2.55
3	0.50	0.33	1.65



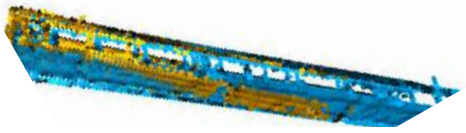
User input 9 a

Top 3 Locations	fitness	rmse	time(sec)
9	1	0.09	11.19
6	0.61	0.30	2.73
10	0.95	0.33	2.81

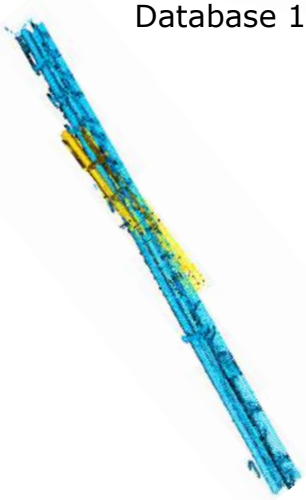
User input 9 c



User input 9 d



Database 10 – Hallway to Geolab



Top 3 Locations	fitness	rmse	time
10	0.99	0.35	1.97
6	0.81	0.27	2.22
2	0.75	0.43	1.91

User input 10 c

A.4. Matching results

In this appendix, the top five results are visualised for the different matching methods.

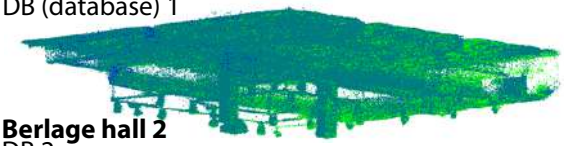
Laser input:

- ◯ **Histogram matching**
- □ **Feature matching**
- ◯ **Combined**

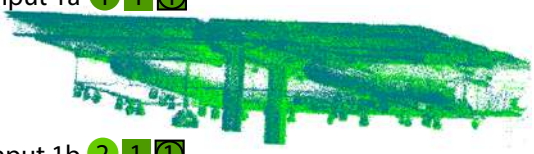
Not in top 5

5 In top 5

Berlage hall 1
DB (database) 1



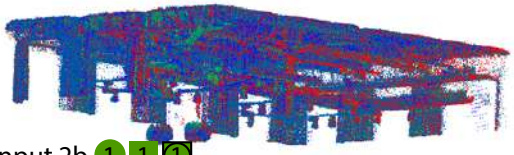
Input 1a 1 1 1

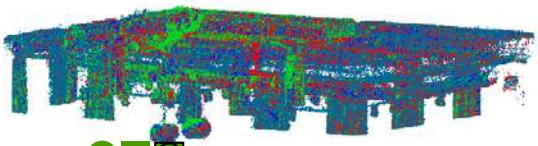


Input 1b 2 1 1

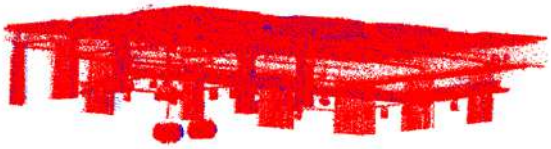
Berlage hall 2
DB 2

Input 2a 1 1 1





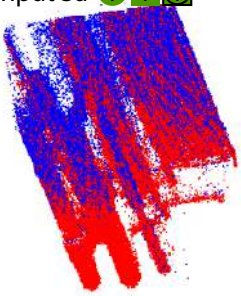
Input 2b 1 1 1



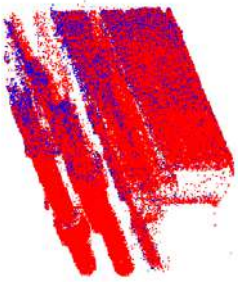
Input 2c 1 1 1

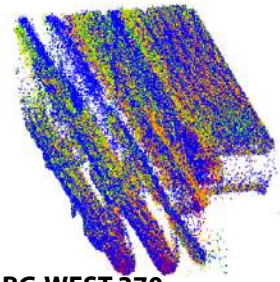
Hallway Berlage
DB 3

Input 3a 1 1 1



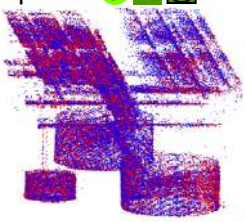
Input 3b 1 1 1





BG.WEST.370
DB 4

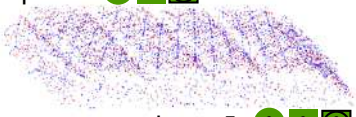
Input 4a 3 1 1



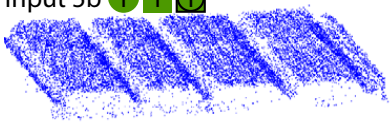
Input 4b 10 1 2

Input 4c 1 1 1

Input 5a 1 1 1

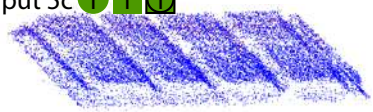


Input 5b 1 1 1

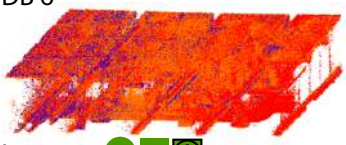


BG.WEST.370 (entrance)
DB 5

Input 5c 1 1 1



Room B
DB 6



Input 6a 2 1 1

Input 6b 5 1 1

Input 6c 1 1 1

Input 6d 2 1 1

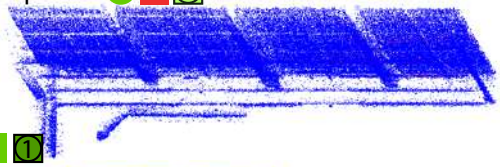
Input 6e 3 1 1

Input 6f 1 1 1

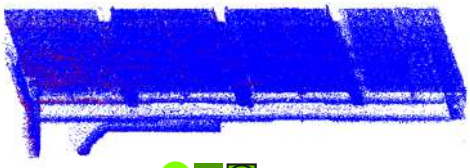
Input 6g 3 1 1

Geolab
DB 7

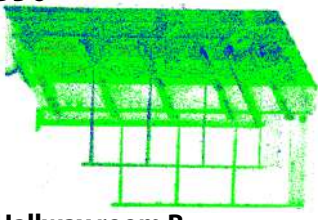
Input 7a 2 7 3



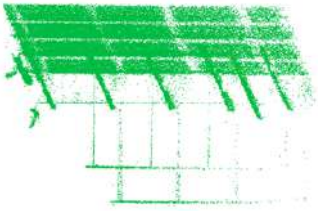
Input 7b 1 2 1



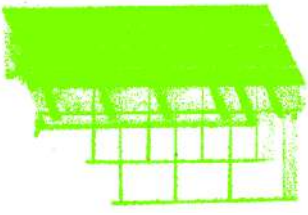
BG.EAST.430
DB 8



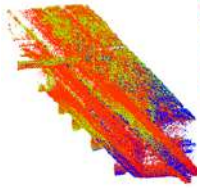
Input 8a 3 1 2



Input 8b 1 1 1

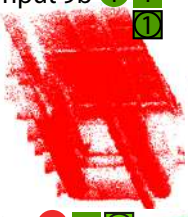


Hallway room B
DB 9

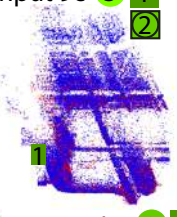


Input 9a 1 1 1

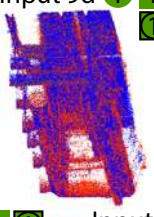
Input 9b 1 1 1



Input 9c 3 1 2

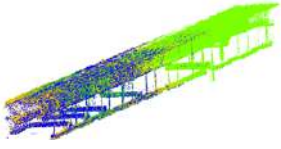


Input 9d 1 1 1

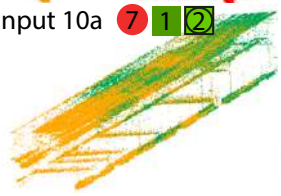


Input 9e 3 1 1

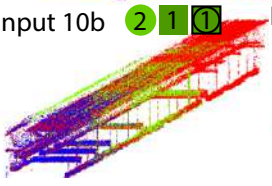
Hallway Geolab
DB 10



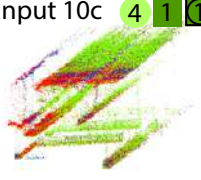
Input 10a 7 1 2



Input 10b 2 1 1



Input 10c 4 1 1



DIM input:

◯ **Histogram matching**

☐ **Feature matching**

Not in top 5

5 In top 5

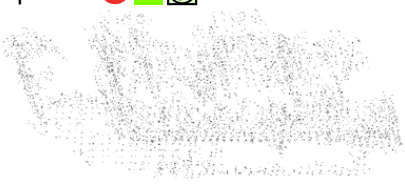
Input 1a (raw + bbox)

Input 1a 4 3 4

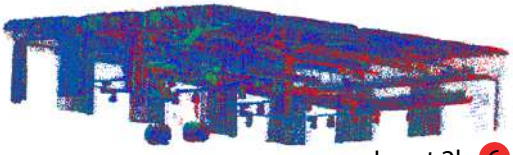
Berlage hall 1
DB (database) 1

Input 1b (raw + bbox)

Input 1b 8 3 5

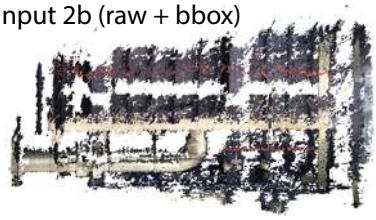


Berlage hall 2
DB 2



Input 2b 6 2 4

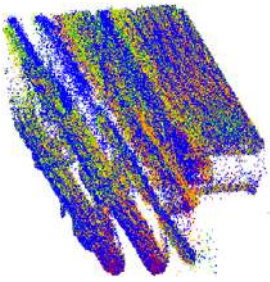
Input 2b (raw + bbox)



Input 2a (raw + bbox)

Input 2a 3 2 3

Hallway Berlage
DB 3

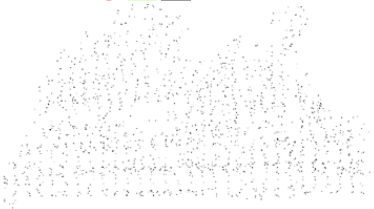


Input 3b (raw + bbox)

Input 3b 9 3 6

Input 3a (raw + bbox)

Input 3a 8 5 7



BG.WEST.370
DB 4

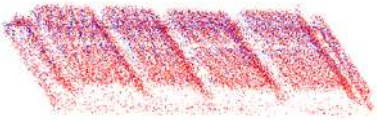
Input 4a (raw + bbox)

Input 4a 1 1 1

Input 4b (raw + bbox)

Input 4b 3 1 1

BG.WEST.370 (entrance)
DB 5



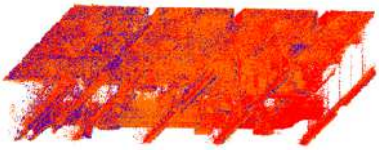
Input 5a (raw + bbox)

Input 5a 1 1 1

Input 5b (raw + bbox)

Input 5b 1 7 1

Room B
DB 6



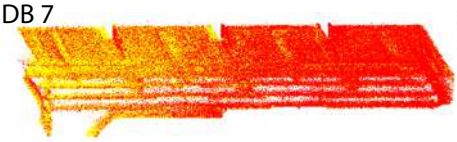
Input 6a (raw + bbox)

Input 6a 5 3 4

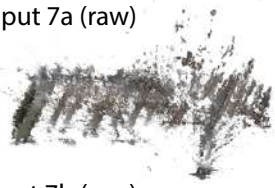
Input 6b (raw + bbox)

Input 6b 4 1 2

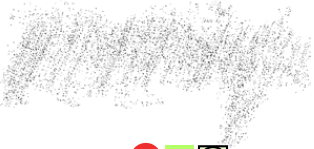
Geolab
DB 7



Input 7a (raw)



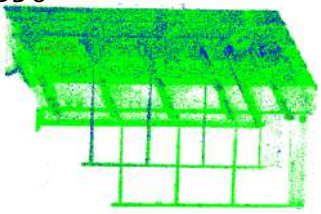
Input 7a 6 4 7



Input 7b (raw)

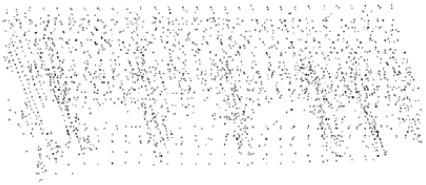
Input 7b 8 4 5

BG.EAST.430
DB 8



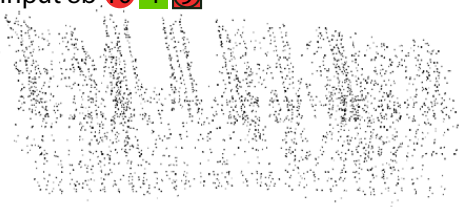
Input 8a (raw+ bbox)

Input 8a 10 4 5

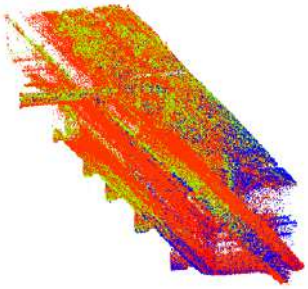


Input 8b (raw + bbox)

Input 8b 10 4 9



Hallway room B
DB 9



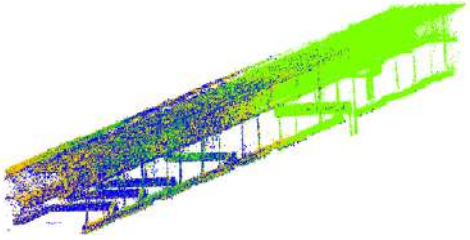
Input 9a
(raw + bbox)

Input 9a 8 8 10

Input 9b
(raw+ bbox)

Input 9b 1 7 2

Hallway Geolab
DB 10

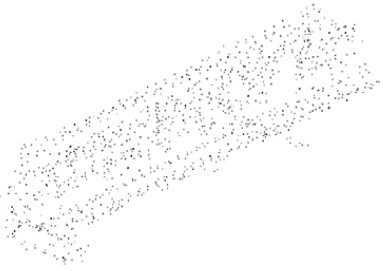


Input 10a (raw+ bbox)

Input 10a 6 10 9

Input 10b (raw + bbox)

Input 10b 1 10 2



A.5. Confusion matrices

In this appendix, the confusion matrices for the results are shown.

A.5.1. Histogram approach

LiDAR

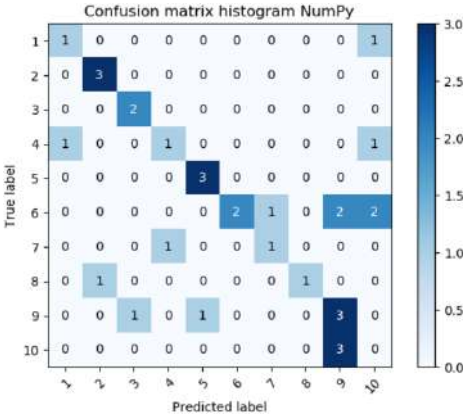


Figure A.4: Confusion matrix top 1 correctly classified locations for the NumPy approach using a laser scanner user input.

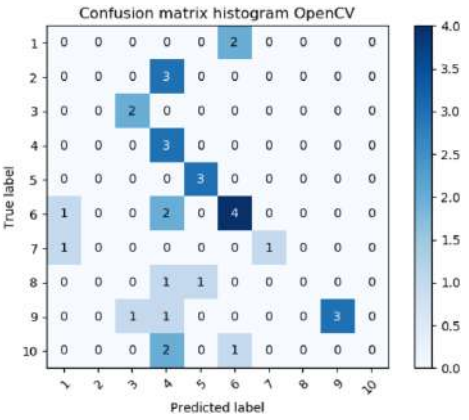


Figure A.5: Confusion matrix top 1 correctly classified locations for the OpenCV approach using a laser scanner user input.

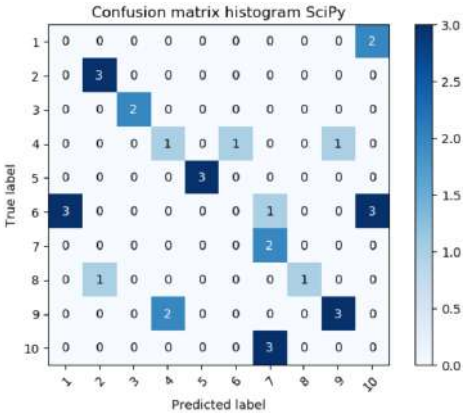


Figure A.6: Confusion matrix top 1 correctly classified locations for the SciPy approach using a laser scanner user input.

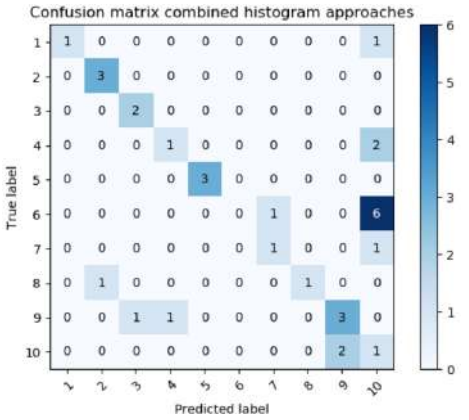


Figure A.7: Confusion matrix top 1 correctly classified locations for the combined approaches using a laser scanner user input.

DIM

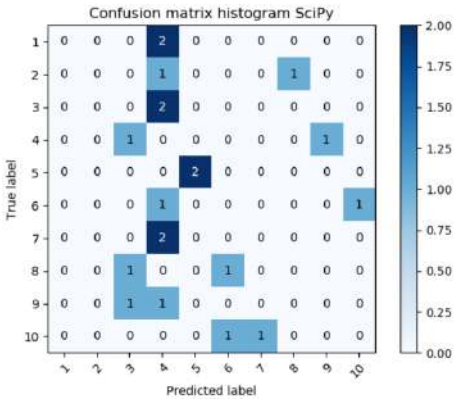


Figure A.8: Confusion matrix top 1 correctly classified locations for the NumPy approach using pre-processed DIM user input.

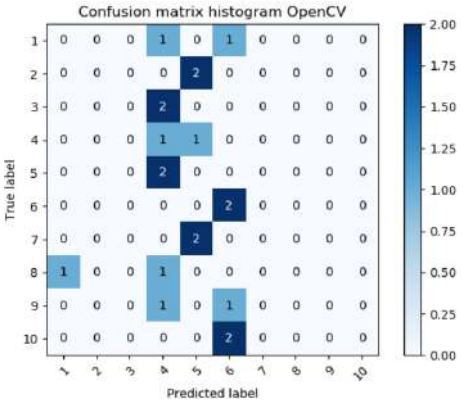


Figure A.9: Confusion matrix top 1 correctly classified locations for the OpenCV approach using pre-processed DIM user input.

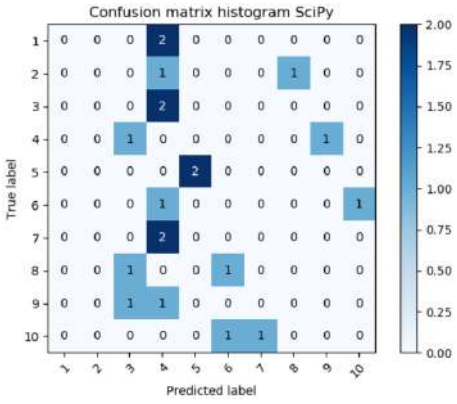


Figure A.10: Confusion matrix top 1 correctly classified locations for the SciPy approach using pre-processed DIM user input.

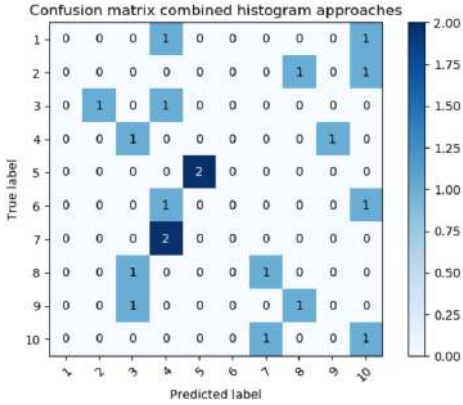


Figure A.11: Confusion matrix top 1 correctly classified locations for the combined approaches using pre-processed DIM user input.

A.5.2. Feature matching approach

LiDAR

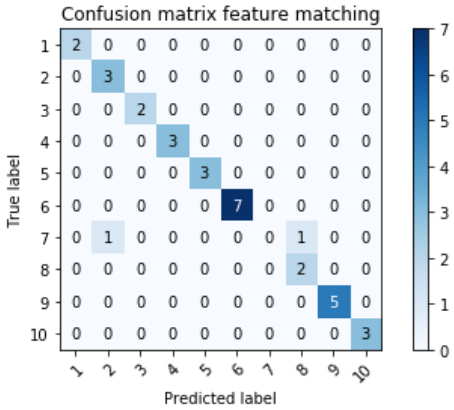


Figure A.12: Confusion matrix top 1 correctly classified locations for the RANSAC approach with voxel size 0.2 using a laser scanner user input.

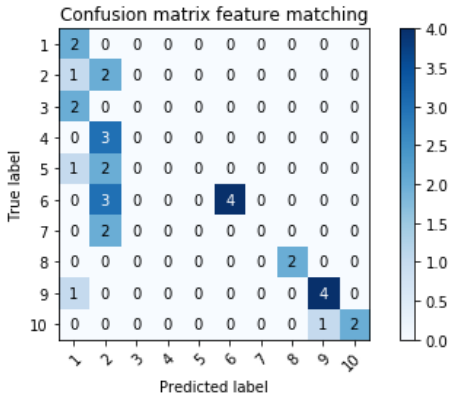


Figure A.13: Confusion matrix top 1 correctly classified locations for the RANSAC approach with voxel size 0.4 using a laser scanner user input.

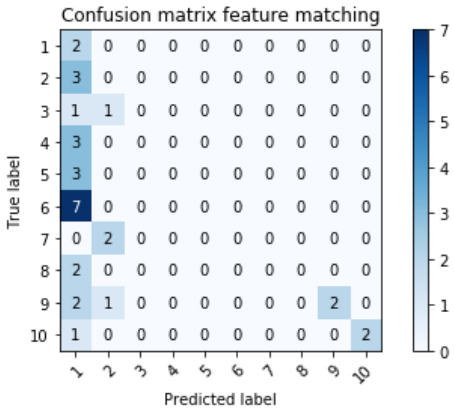


Figure A.14: Confusion matrix top 1 correctly classified locations for the RANSAC approach with voxel size 1.5 using a laser scanner user input.

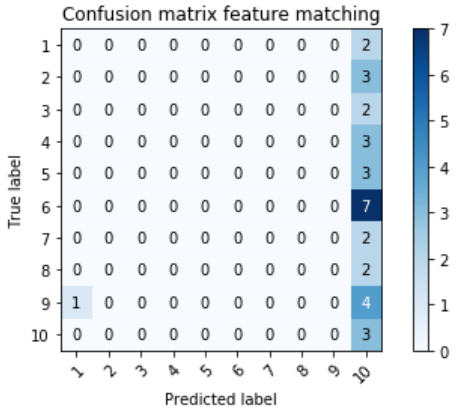


Figure A.15: Confusion matrix top 1 correctly classified locations for the RANSAC approach with voxel size 10 using approach using a laser scanner user input.

DIM

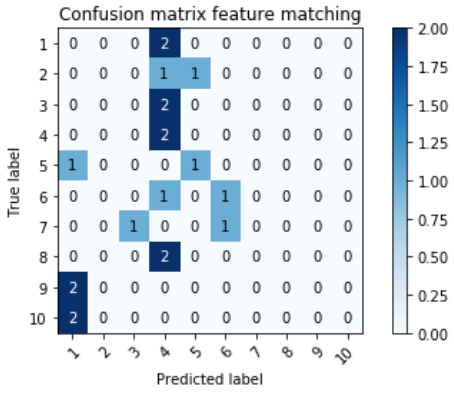


Figure A.16: Confusion matrix top 1 correctly classified locations for the RANSAC approach with voxel size 0.3 using pre-processed DIM user input.

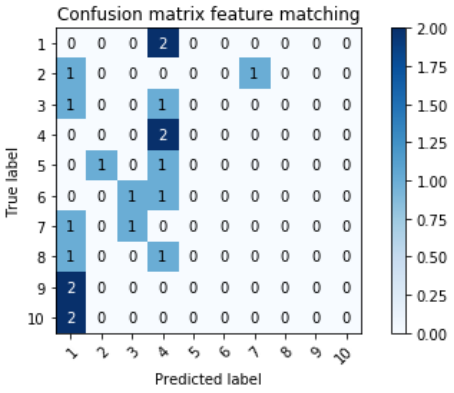


Figure A.17: Confusion matrix top 1 correctly classified locations for the RANSAC approach approach with voxel size 0.4 using pre-processed DIM user input.

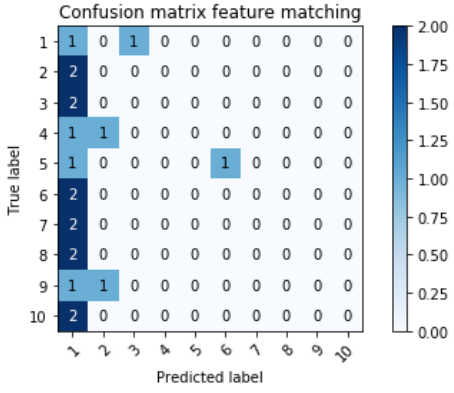


Figure A.18: Confusion matrix top 1 correctly classified locations for the RANSAC approach with voxel size 1.5 using pre-processed DIM user input.

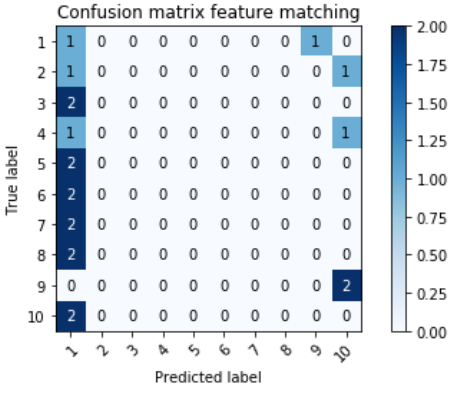


Figure A.19: Confusion matrix top 1 correctly classified locations for the RANSAC approach with voxel size 10 using approach using pre-processed DIM user input.

A.5.3. Combined approach

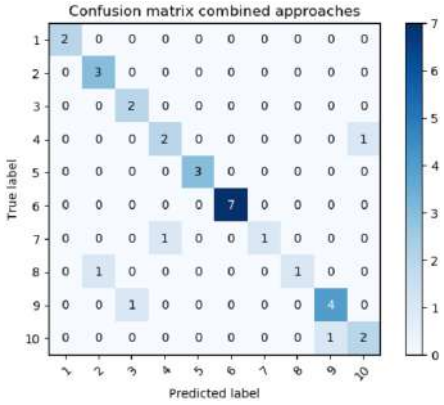


Figure A.20: Confusion matrix top 1 correctly classified locations for the combined feature matching and histogram approach using a laser scanner user input.

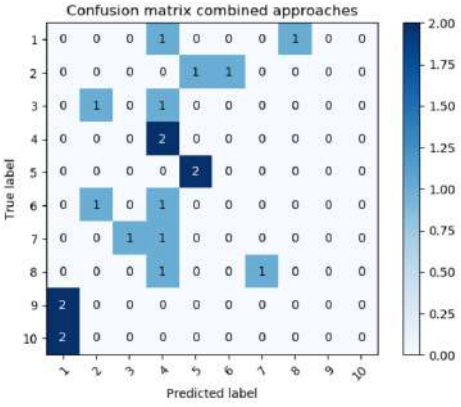


Figure A.21: Confusion matrix top 1 correctly classified locations for the combined feature matching and histogram approach using DIM scanner user input.