Map generation using a smartphone's built in localisation and mapping algorithms

Emil Harvig, mte14eha, emilharvig@hotmail.com

March 28, 2023

Abstract

Localisation and mapping algorithms for smartphones have in recent years seen a renewed interest due to their technological advancements. Using these systems to generate an indoor blueprint where none is available or incomplete is useful in tracking applications. This report examines the possibilities of adapting ARcore as a tool for map generation of unknown areas. Several tests of indoor environments were performed in office and university environments, recording pose track and environment data. This study finds it feasible to perform a combined pose tracking and indoor mapping, capturing the overall shape of a surveyed area with some distortions and drift in the resulting map.

Keywords: Point Cloud, SLAM, vSLAM, Pose, Occupancy Grid Map, OGM, ARcore

Acknowledgements

I want to thank everyone involved in the project, Martin and Björn at Combain Mobile, Kalle and Magnus at Centre for Mathematical Sciences at LTH, and for all the feedback on my work and ideas during this period. Additionally I would like to extend a big thank you to all of you who showed up at my presentation of this report.

Contents

1 Introduction

Mapping and localisation algorithms for autonomous robots have in recent decades improved dramatically both in terms of speed, accuracy and robustness. These systems are now also available on both Apple and Android smartphones. This report aims to use the simultaneous localisation and mapping, SLAM, present in most mid- and high-end smartphones [1], [2].

Using a smartphone equipped with SLAM this report aims to develop a method of creating an indoor 2D map. The SDK chosen for this task is Googles Visual SLAM system, ARcore for Android.

The first three quarters of the work in this report was focused on creating an occupancy grid map using sparse point clouds. In the last stage it was discovered that ARcore could supply point clouds three orders of magnitude denser. Additionally, these denser point clouds were of similar properties and qualities as the sparser ones. This introduced the need for filtering and fundamentally changed the sparse data issues experienced on its head. Most of the later results presented in the report are scatter plots instead of using the occupancy grid map method developed.

1.1 Problem formulation

The aim of this report is to test and find methods of using ARcore's vSLAM to generate building blueprints. This is done to test the usability of mapping an indoor space and attempting to recreate accurate building blueprint equivalents. By evaluating accuracy, resolution and duration of measurements, an indication of feasibility and accuracy will be presented. The idea for this project originates from a desire to use crowd-sourced data to create indoor maps automatically, as a byproduct of using SLAM for pose tracking. The pose tracking is useful when recording signal strengths for indoor positioning and may automate the initial localisation during measurements.

The areas where this report places it is focus on are map generation using Occupancy Grid Mapping and how to use ARcore as a measurement instrument.

1.2 Report Structure

The report is structured as follows; In this section a brief overview of the most important topics are explained. Thereafter previous and related studies are discussed. After that a description of the process and some mathematical explanations. The reader may refer to section 1.7 or Appendix B.2 if some concepts are not sufficiently explained beforehand. After that follows description and presentation of testing results, results in general and discussions and a short conclusion, emphasising the most interesting findings.

1.3 What is ARcore and what was it made for?

ARcore is a tool to create augmented reality, AR, experiences for users and it is not intended as a precise mapping tool. It behaves as a black box and to the authors knowledge ARcore does not allow to input specific data sets [3]. It is not possible to access the SLAM process pipeline directly. The ARcore back-end reports landmarks into a debug log. These landmarks would be useful to acquire in the purpose of understanding the process deeper.

By limiting the access to an interface it also prohibits the possibility of attempting to modify behaviours to be more suited for mapping and tracking purposes over larger areas. There are other SLAM systems available which may be better suited as measurement tools. ARcore was chosen due to it is availability in Android smartphones and to avoid implementing a SLAM system from the ground up.

One may picture the author walking around with an advanced camera system that behaves in a erratic way. For example, the camera pose tracking is arguably precise and accurate for a cheap system. Although for this to be true there are some significant restrictions or procedural rules to follow to attempt to get a satisfying measurement result. This data recording methodology can be found described in the result section.

1.4 Image interpretation and scale

A 2D (mono)camera image innately lacks any sense of scale. We humans for example infer a substantial and critical amount of information to objects seen in an image. For example, we can usually judge if an elephant in an image is a toy or not by observing the setting of the image. However, without scale information it is not possible to tell apart a model scene from a real scene if it is done correctly. As in Figure 1, it seems reasonable to assume that the car is a small model car and that elephant is roughly the size of a hand. But that's just the author guessing. Furthermore, how big is the house in the background? Is it a real house or is the entire image just a scale model, constructed to illustrate the perception of depth and scale? We infer a significant amount of information to objects when we look at an image, because it inherently lacks any information on the scale, disregarding any information one can assume from objects in the image.

However, there are ways of estimating scale using only one camera, which will be explained further down. It requires a series of images recorded from slightly different angles. ARcore takes advantage of structure from motion methods to calculate and estimate depth [4].

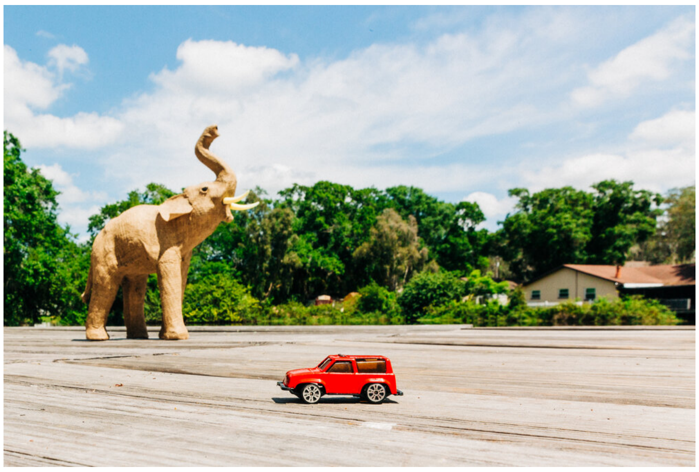


Figure 1: Images illustrating that estimating the scale of objects from a single image is relatively easy for humans, but may be difficult for a computer.

1.5 SLAM - Simultaneous localization and mapping

A commonly used technique to solve the mapping and localization issue is to use Simultaneous Localization and Mapping, SLAM. Finding oneself in an unknown world, a clever approach to achieve robust localization and mapping is to perform them at the same time. At the time of writing, the SLAM technologies are considered to be mature [5], [6]. SLAM systems are designed to function independent of type of sensor data and may combine different sensors [7]. It is also by its design modular. In car automation, higher quality sensors are often used, such as LiDARs and radars in combination with cameras. An advantage with a LiDAR is the accurate depth readings they produce. Compared to camera image the data is sparse. Data from a LiDAR is markedly different from a traditional image. The "image" received is a 3D point cloud representation of the scenery with high accuracy. Ideally, when using SLAM, a LiDAR-camera combination has the best of both worlds. The camera images can be complemented with accurate distances for certain features.

Mobile SLAM systems such as ARcore and ARKit are down-scaled versions of their less mobile counterparts. This is due to the limitations of computational power available in a smartphone. This results in worse results regarding tracking and positioning compered to their full scale counterparts [5].

1.5.1 viSLAM - Visual Inertial SLAM

ARcore combines pure visual SLAM with inertial readings from the Android Sensor system [3], [8], [9]. In the documentation, it is claimed that if only minuscule movements of the camera (smartphone) is detected, the sensor system is used for localisation. When larger movements are detected the system also uses the visual system.

1.6 Structure from motion, SfM

"SfM is the process of reconstructing 3D structure from its projections into a series of images taken from different viewpoints." [10].

This is partially how ARcore calculates depth [4]. By finding landmarks in overlapping scenes, it is possible to calculate how much these landmarks move between frames. By also using techniques such as loop closure [11] and bundle adjustment [12], a depth estimation is achieved.

In ARcores case, the depth estimation is also passed through a machine learning network which complements the depth image [4]. It is not explained if this algorithm is also used in the SLAM system itself or only for producing depth images.

1.6.1 Google ARcore and depth from motion

Google's ARcore framework calculates depth using depth from motion techniques [4]. By using ordered frames previous and current frames are used to calculate depth. The API supplies a depth image or specific points in point cloud, PC, format. The given PC-points have an estimated accuracy and are sparse. Every frame yields only a few points, but given that the frame rate of the camera is between 30 and 60 Hertz, a dozen of points may be acquired per second. This highly depends on the landscape the camera is observing. For example, a brick wall produces more points than a white smooth wall [13].

Google themselves claim "The most accurate results come when the device is half a meter to about five meters away from the real-world scene" [14], [15].

1.6.2 Depth Images

A depth image is an image with depth information encoded in some or all pixels. It is in principle a point cloud in it is representation, however the data is of lower quality because it has been estimated by calculations, not direct measurements. A LiDAR measures the time of flight, ToF between each reflection of its rays. ARcore likely calculates depth images by triangulate the distance to some regions of the image and uses a AI-network to interpolate distances where there are missing regions. Android also supplies depth image functionality and ARcore supplies a depth image of , for the

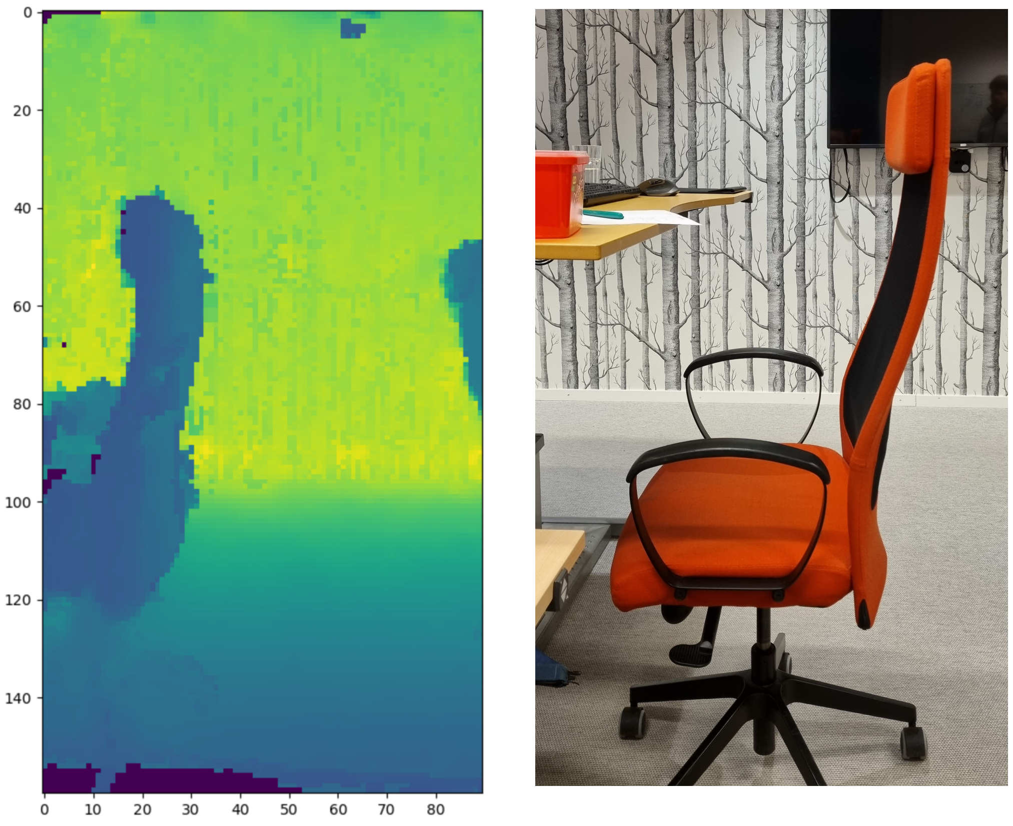


Figure 2: A depth image of a office chair (left,blue), with a wall in the background (green). The office chair depicted as a reference in the right image.

smartphone used, a Samsung Galaxy S21, the images were of size 90 × 160 pixels, a total of 14400 pixels per frame [16]. Unless the camera is moving slowly, by walking at a "snails pace", these depth images are usually mostly empty. But given enough time to perceive the environment more and more of these pixels contain information.

1.7 Key Concepts of ARcore

A point cloud is usually produced by a LiDAR or radar measurement. In this context however, no LiDAR is used. But some of the data, the depth images and point cloud points are used in the same way as one might use LiDAR readings.

1.7.1 Image, Depth image, Anchors, Poses and Point Clouds

Below are some keywords and concepts crucial to understanding the report, they are briefly explained here but a more extensive explanation can be found in appendix B.2. All range data supplied by

ARcore is in meters and always has 3D coordinates. Depending on what kind of object it is, it may also have a heading, extent or confidence level.

A **Pose** is the estimated camera position in a 3D world, with 6 DoF. x, y, z and orientation. A **Camera pose** is the raw untracked position supplied each frame by the API. It is the coordinates of the current position, to the best of ARcores understanding at the sample time.

A **Point Cloud point** , PC-point, is a reported point of a surface. This is acquired on a frame by frame basis and thus is directly connected to the camera pose of that frame.

A **Trackable** is a wrapped object of elements such as an **Anchor**, **Point** or **Pose** that has it is location continuously updated by the system. These cost significant computational resources and are not used unless needed.

An **Anchor** is any point or pose that has been sent to the system to be tracked.

A **Frame** refers to an image object and any additional information received from the camera. Each frame object can be seen as a sample point, and cannot be accessed again unless information is stored from that frame before releasing it.

Confidence is a value ranging from one to zero which estimates the quality and confidence of the reported object. A value of zero encodes an invalid pixel. All depth image points has a confidence value, as well as point cloud points. Google does not specify how it is calculated.

1.8 A relative map

ARcore supplies a frame of reference to an arbitrary center point. The point clouds and pose tracks only relate to themselves. A nontrivial issue is where and how to relate a map to the real world. In this report, if the OGMs are overlayed with real blueprints they are mapped manually. ARcore supplies a feature called VPS, an integrated GPS reading in the API [17]. It is possible to get an GPS estimation connected to each pose but this was not explored significantly.

The map registration may also be done with two GPS-points or a supplementary heading to one GPS-point.

2 Related studies

The OGM was initially developed using sonar radars and as a way of representing the sonar measurements in a 2D presentation [18], [19]. Over time many different versions of occupancy grid maps have been developed. The approach of using log-odds(ratios) was developed by Konoliges [20], and is partly what this reports current OGM implementation is based on. He contributed to improving the previous work of the early occupancy grid maps. Although his work was mostly intended for radar measurements, some of the key concepts are used in this report.

Collins, Collins and Conor provides and discusses the ways an occupancy grid map can be empirically evaluated [6]. Their ambition is to create ways to evaluate these maps without the subjective visual analysis. By standardising the results of what they classify as five different methodologies to create an OGM. They found the method devised by Konolige of producing an OGM to be a viable method, although not the best of the five.

A different approach was taken by implementing and extending a SLAM system with building structure line detection where they extended the SLAM system to take into account certain geometric regularities of man made buildings [21]. They achieved interesting and promising results in their testing. Although being an possible approach, it was not attempted to make any edits to the functionality of the SLAM-system. ARcore was used as a tool for gathering data.

Another recent study evaluated the accuracy of Microsoft's, Apple's and Google's vSLAM systems [3]. They set up a small scale test and a large scale test. The small scale test let the systems observe an office cubicle from a stationary perspective. The large scale test was attempted in a featureless indoor location with thirty meters between landmarks such as office equipment. In small scale testing all three systems were able to be initialised and thus tested. The large scale test was unsuccessful for all systems except the Microsoft system. The authors referred to their findings when evaluating the performance of the mobile SLAM-systems as "showstoppers". They found an accumulated error of 17m per 120 m. These settings were considerably more challenging than the environments tested in this report. This evaluation was done in 2020 and it seems ARcore performs considerably better in later versions.

In a blog article and a scientific article with striking similarities, ARcore was used to find paths in a building. Access to the building blueprints were available and both studies used image based landmarks to acquire the starting position. Both of the projects showed promising findings to use ARcore as a mapping tool [22], [23]

The increased amount of data retrieved from the depth images enables other techniques to generate a map than an OGM. In their work, Afsén and Boye Frick explore and develop methods to improve the visualisation of point clouds [24]. To fill in missing data points in the map for the ground surface, they tested Poisson Surface Reconstruction. Due to the increased amount of data, this may be an useful algorithm to use when filling in missing data points in a surface due to the increased amount of data.

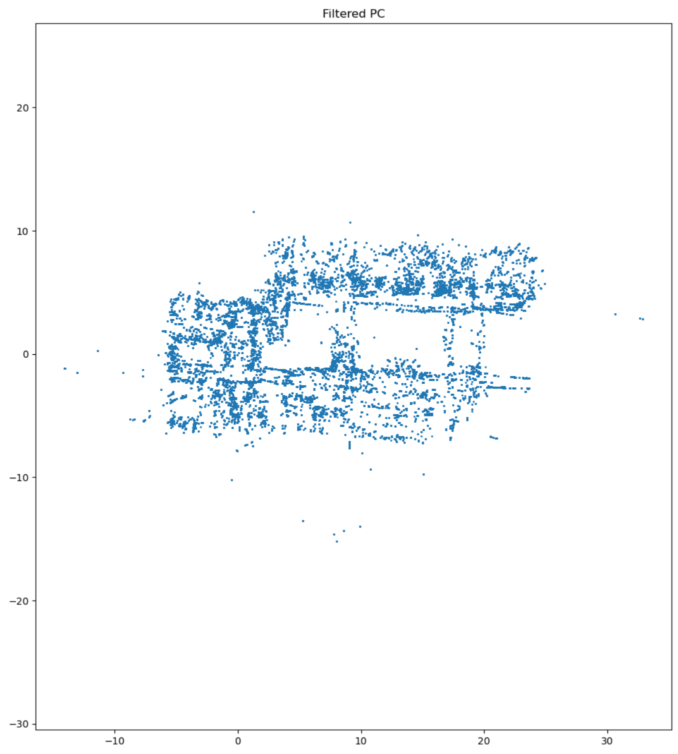


Figure 3: A point cloud representing an office building from a top down perspective.

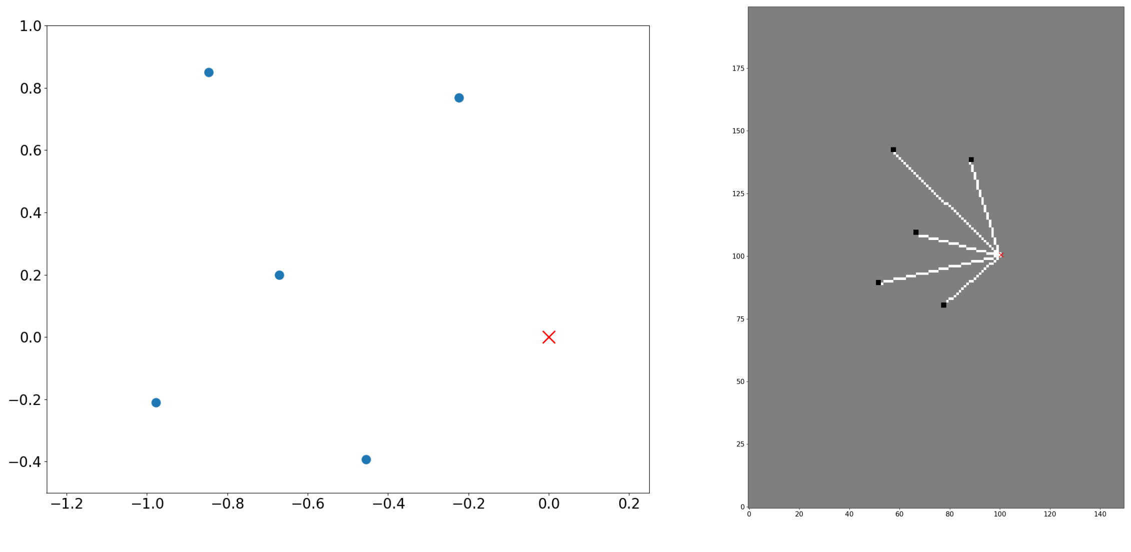


Figure 4: Five simulated points in blue and a camera position marked as a red "X". The left figure shows the raw data. The right figure shows the resulting binary OGM.

3 Method

3.1 A data point

A typical set of data points contains one camera pose point c and at least one PC-point p. A depth image may have a number of PC-points $(p_1, p_2, ..., p_n)$ where

$$c = (x_c, y_c, z_c) \;\in\; R^3, \tag{1}$$

$$p_i = (x_p, y_p, z_p) \;\in\; R^3. \tag{2}$$

Each depth image may have up to 14000 PC-points. Each data point is considered to be independent of each other. ARcore saves these data points and reports units in meters. Most visualisations shown are top-down 2D scatter plots or OGMs, also top-down.

3.2 Grid map

A grid map is used to arrange data for OGM creation and downsampling. Each grid cell g can have a value between 0 and 1. An arbitrary value, 0.5, is used to encode unknown areas. The value corresponds to a unknown cell, since it's equally likely to be occupied or empty. In the case of a binary OGM, explained below in 3.4.1, it serves to visualise by adding contrast to the white lines and black dots.

$$g(x, y) = 1 \;, Occupied \tag{3}$$

$$g(x, y) = 0 \;, Empty \tag{4}$$

$$g(x, y) = 0.5 \;, Unknown \tag{5}$$

Scaling and placing the data on a predetermined grid scale is done according to

$$maxwidth_x = \frac{max(data_x) - min(data_x)}{s}, \tag{6}$$

where s is the ratio of meters per pixel.

For example a measurement spanning 10 × 10 meters and an s-ratio of 20 would have a resolution of 5 cm and correspond to a grid containing 200 × 200 cells or pixels.

3.3 Bresenham's Line Algorithm

Bresenham's line algorithm projects a discrete line between the pose and recorded point on a grid map. The algorithm determines which cells in a grid to most correctly represent a continuous line by calculating an accumulative error [25]. It returns a list of grid cells representing the input line.

3.3.1 Usage

In the simplest implementation, this set of grid cells is now placed on the grid map. Every cell from the pose except the recorded point is set to 0 (empty). The recorded points corresponding cell is set to 1, and possibly some of the connected cells depending on implementation. Figure 4 and 8 illustrate the process.

3.4 Different Occupancy Grid Maps

3.4.1 Binary Occupancy Grid Map

A binary occupancy grid map is the simplest implementation of the method. It does not take into account intersecting lines. Besides an unexplored value, 0.5, each grid cell can only have to values, occupied or not occupied, see Figure 4.

3.4.2 Probabilistic Occupancy Grid Map

In order to take into account intersecting lines projected on the OGM and preserve detail, each cell on the grid is seen as a probability ranging from 0 to 1. 0.5 still represents unknown or in some cases indecisive grid cell. Figure 9 and 10 illustrates the importance of taking into account overlapping projected lines.

$$P(g(x,y)) = 1\ ,\ Occupied \tag{7}$$

$$P(g(x,y)) = 0\ ,\ Empty \tag{8}$$

$$P(g(x,y)) = 0.5\ ,\ Unknown \tag{9}$$

3.4.3 Odds and log odds representation

The log odds for a cell being unoccupied or not is initially set to 0. For the purpose of this example, the probability of a cell being occupied is arbitrarily set to 0.65. The probability for that cell to be free is $P(A_{free}) = 0.35$ and $P(B_{occupied}) = 0.65$ respectively,

$$odds(A) = \frac{P(A)}{P(B)}. \tag{10}$$

Reversed the probability of a cell being occupied is

$$odds(B) = \frac{P(B)}{P(A)}. \tag{11}$$

Using $A_{free} = 0.35$ and $B_{occupied} = 0.65$ yields the odds that a cell is occupied as

$$O_{occ} = \frac{0.65}{0.35} = 1.86. \tag{12}$$

The logarithmic odds of a cell being occupied is then

$$l_{occ} = \log(O_{occ}) = 0.62 \tag{13}$$

and

$$l_{free} = \log(O_{free}) = -0.62 \tag{14}$$

respectively.
The log odds for each cell can be calculated by the equation

$$\sum_{n=1} c_{ij} = l_{init} + l_{free}, \tag{15}$$

with k being a mix of sensor model and different probabilities described later. For a empty cell measurement see below. If the measurement would give an occupied cell reading $l_{occupied}$ is added instead

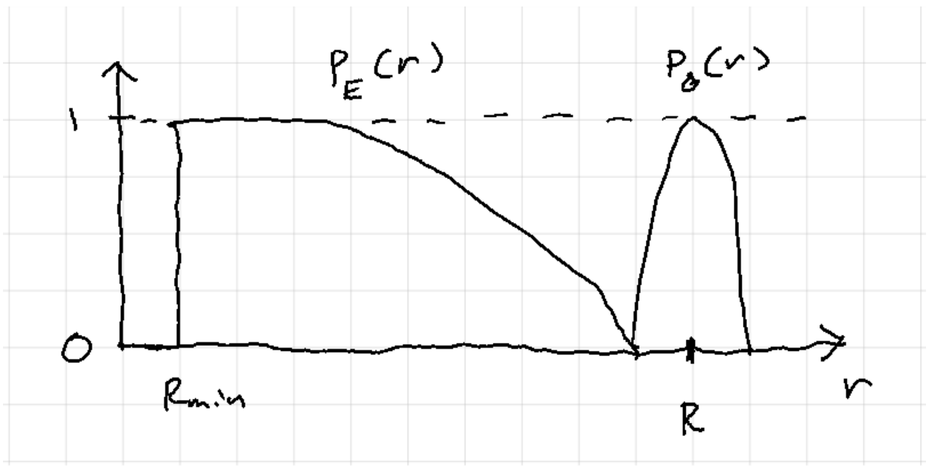


Figure 5: Sensor model developed by Elfes [18]. $P_E(r)$ is a function of the probability for the distance r from the sensor to be occupied. $P_O(r)$ is a function for the probability of occupied. R_{min} is the closest detectable distance and R is the reported distance of the hit.

3.4.4 Generating the Occupancy Grid Map

To convert the log-odds back to probabilities the following formula is used

$$P(o_{ij}) = \frac{e^{(o_{ij})}}{1 + e^{(}(o_{ij})}, \tag{16}$$

where o_{ij} denotes the log odds of each cell, i and j are the indexes for row and column.

3.5 Sensor model

To represent the behaviour of a sensor, a sensor model is often used [18]. In radar applications a sensor model may look like in Figure 5.

The simplest representation is a constant value. In this report two different sensor models were tested and implemented, "v1" and "v2", explained below. Due to the undefined nature of the reported confidence value from ARcore, there was at least two possible ways to interpret this confidence value. "v1" interprets it as a factor that affects the entire projected ray. "v2" interprets it as a depth uncertainty.

3.5.1 Sensor model v1

The sensor model called v1 interpreted the reported confidence values as a probability of the point existing or not in the physical world. Thus, it affected the OGM in a similar way to intensity of a image. This is illustrated in Figure 6.

$$\sum_{n=1} c_{ij} = l_{free} * k_{confidence} \tag{17}$$

$$k_c = [1,\ 0.75,\ 0.3,\ 0.2,\ 0.1] \tag{18}$$

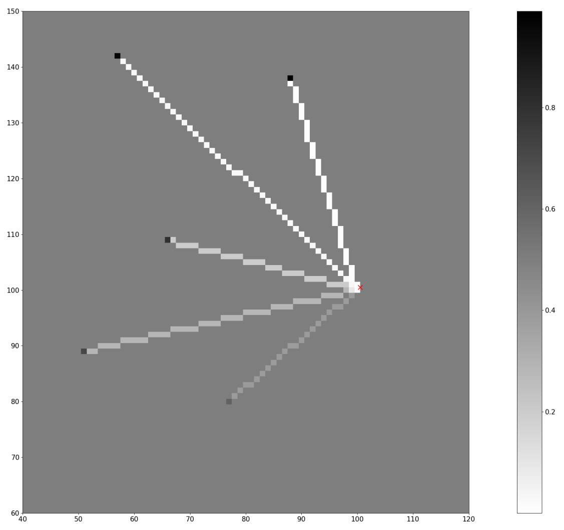


Figure 6: Illustrating the impact of using the confidence as a constant. Constant values used where k_c had the values seen in 18. The same data points as in Figure 4 are used. The lower the confidence value, the less the ray impacted the OGM.

3.5.2 Sensor model v2

Sensor model v2 interprets all reported points as true points, existing in the physical world. The confidence value encodes the uncertainty in the exact distance to the point. A confidence value of 1 would have a small error in the centimetre range, whereas a value closer to 0 would have a uncertainty of about a meter. v2 is visualised in Figure 7.

3.6 Z-score filtering

In order to reduce outliers, need for cropping resulting OGMs and reduce computing time in post processing, a simple Z-score filtering was done, see Figure 32 and 33 in Appendix A. This method

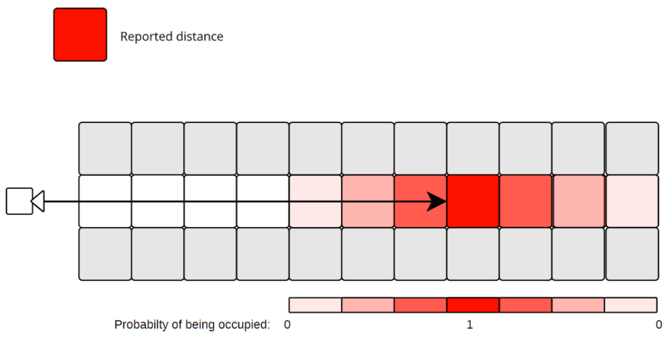


Figure 7: Improved Sensor Model, v2.

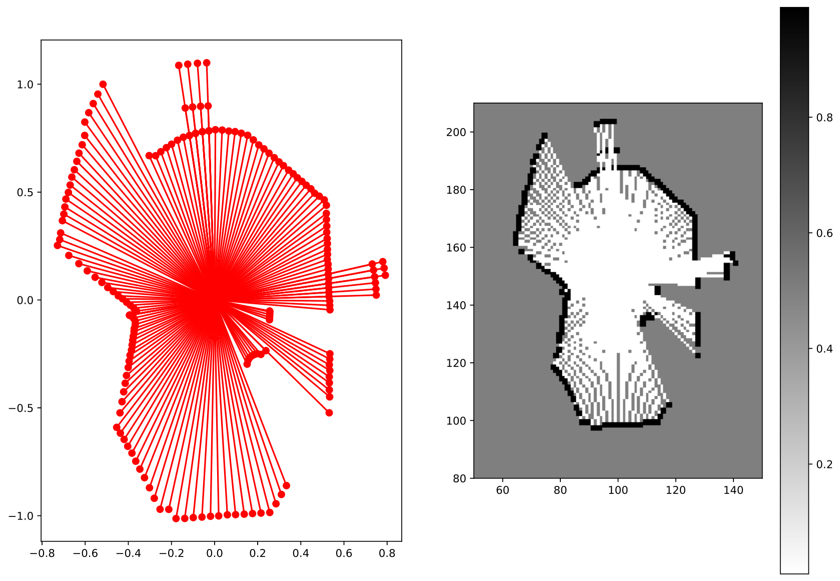


Figure 8: A simple representation of each ray (left), and the corresponding OGM (right).

was not tested on the denser pointclouds recorded using depth images.

Per each axis in all three dimensions, the mean values for x,y and z axis was calculated. These points were then scored using the formula below. Where x_n is any point along a specific axis.

$$Zscore = \frac{x_n - x_{mean}}{\sigma} \tag{19}$$

After scoring all points along each axis, different thresholds and combinations of scores and thresholds were tried until a sufficient result was achieved.

The thresholding was done according to the formula below, where k_{th} was set to 1.65:

$$z_x + z_y < 2 * k_{th} \tag{20}$$

$$z_z < 3 * k_{th} \tag{21}$$

This removed most outliers along each axis and was mainly done to limit the need of manually setting plot visualisation limits, besides reduce computing time.

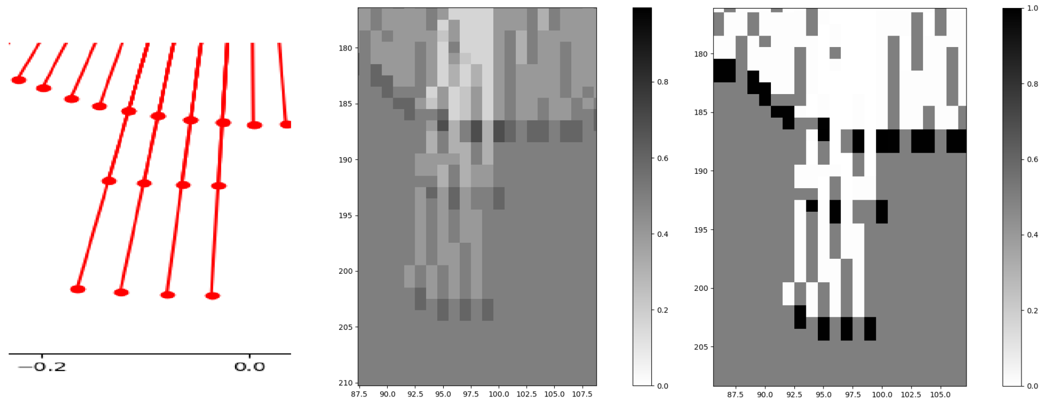


Figure 9: The raw data to the left, in the middle a probability of 0.6 was used. Right image visualises the loss of information using a large difference between occupied and free cells.

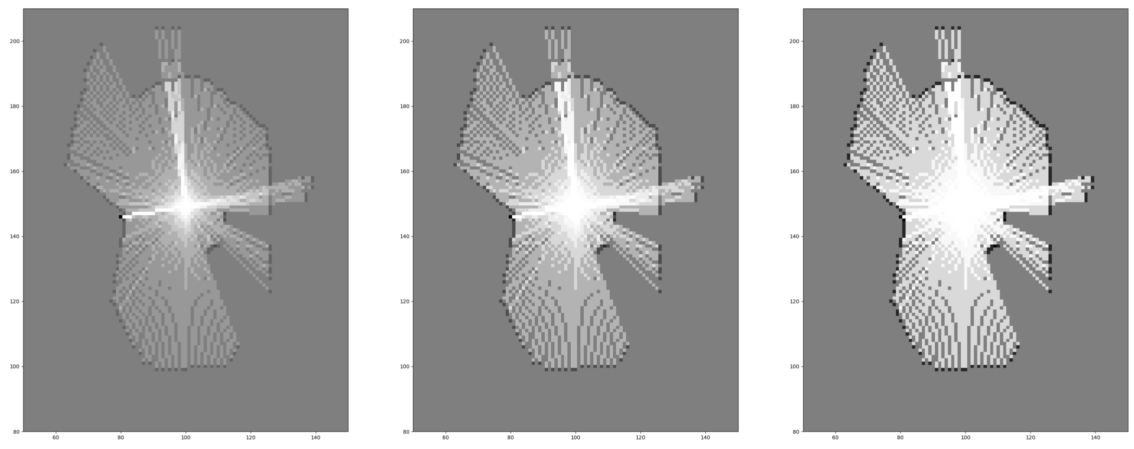


Figure 10: Probability increasing from 0.6 to 0.7 to 0.85 from left to right. As can be seen in the figure, the larger the odds between occupied and free cells are, the sharper the image becomes.

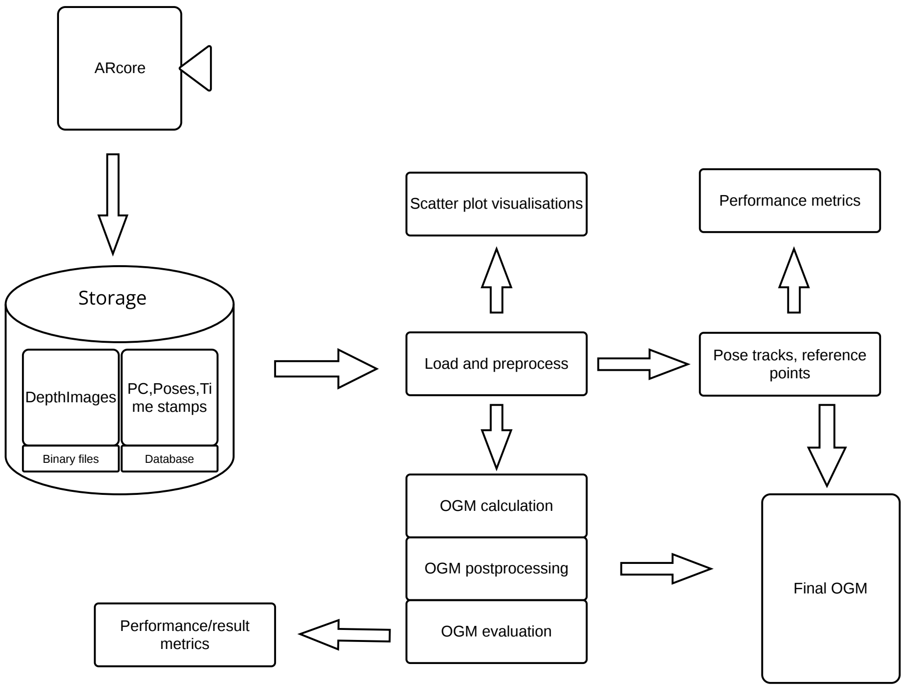


Figure 11: A simple flowchart of the application and a representation of the post processing steps

4 The vSLAM application - HelloAREmil

For the Android application, ARcore SDK v1.31 for Java was used [26]. The code for the report can be divided into two parts; one application part running on Java and a post processing application part running on Python.

4.1 The Java application

The code was based on a sample code found in the SDK called HelloAR. In it is current state the application is saving data both continuously during measurement and at the end of recordings. There are a few ARcore specific objects which have to be saved at the end of recording, since they may be updated every frame. In Figure 11 the different sections of the app and post processing is shown.

4.2 Data structure

PC-points, anchors and poses are saved to a local database on the smartphone. Only the Anchors, of type Trackable, need to be saved at the end of measurement. A thirty minute measurement yields around 50,000 PC-points. A 10 minute recording using depth images yields roughly 20 million PC-points.

Each depth image was saved as they were recorded by the session, frame by frame. The images were converted to a point cloud and stored as a binary file. Each depth image was about 200 kilobytes, recorded at ten frames a second. Thus each measurement took up a considerate amount of local storage, up to a gigabyte of space. Each depth image had a maximum amount of 14,400 pixels corresponding to dimensions 160 $\times$ 90 pixels. The reason for storing data this way was due to simplicity. The methods needed were already available in the SDK.

4.3 Post processing

The post processing code was written in Python and consisted of several scripts. In Figure 11, the most important steps are illustrated as boxes. In the load and ”preprocess” part, the data was loaded and converted to unified format for use in later calculations. The data was rotated if needed in this step. As described before, Bresenham’s line algorithm was used to draw each ray in the OGMs. The depth image data was downsampled to images where one pixel corresponded to one cm and potentially an offset.

5 Data recording

All data collection was made using a Samsung Galaxy S21 5G smartphone with Android OS version 12. The device has several cameras but it uses one camera when recording SLAM 12, the middle one. This was asserted by occluding the middle camera and attempting to record. It may use two or more cameras, although it would be costly on battery life, performance and heating. As a user on Stackoverflow claims, *”... will be hot as a boiler”* [9]. Compared to an IOS device, such as an iPhone 14 Pro it lacks a LiDAR sensor or LiDAR equivalent. Thus the only way to acquire depth data is from structure for motion. It would be interesting comparing the results from their platform, however that work is left for someone else.

When running SLAM, the device thermally throttles resulting in less computational power, after about 10 minutes of use. For stability reasons most tests were done with some form of passive

Figure 12: Samsung S21 5G, released 2021. It is equipped with three cameras and one is used for vSLAM. ARcore uses the middle camera, a 12MP Wide-angle Camera [27].

cooling used to absorb heat from the phone. This seemed to reduce frequency of crashes and reduce throttling. How it was done can seen in Figure 36 in Appendix B.1.

5.1 Testing setup

The final set of measurements, with depth image capturing implemented, were done during the evening. On a few occasions where there was people walking by and disturbing the measurements. During these few occasions when people were in the way, the recording was put on hold by not moving forward and wait for the people to pass.

To a bystander it may seem as a video recording is occurring during a measurement. To avoid disturbing bystanders and potential integrity issues some measurements were discarded due to this. The recorded depth images are also low resolution, making a distant bystander present in the recording almost impossible to find in the data.

5.2 Combain Office Tests

To evaluate the quality of the camera poses registered by ARcore, several tests were planned in the office space. These tests were done in a real office environment. Additionally the tests were recorded just before a Christmas celebration dinner resulting in most of the office space occupied by tables, chairs and decorations. This made the measurement challenging for the system. A sample of viewpoints from the measurements can be seen in Figure 13 and the map in Figure 14.

5.2.1 Pose evaluation

A circle path around the office was planned, with 9 marked and measured locations spread in the office. The ground truth of these marked spots was measured relating to the blueprint, with the intent to be used for registration and validation. These laps can be seen in Figure 20. These laps took about 3 minutes each, resulting in measurements between 3 and 12 minutes long.

Each test was initialised viewing the same scene, and the same lap was done several times.

5.3 Campus testing

These tests were performed during the evening. As can be seen in Figure 15 and 17 both settings have a similar layout with a large entrance hallway and a series of corridors. E-huset however has mostly white walls versus A-huset, with it is mostly brick and coarse surfaces. The reference points or ground truths, were marked using tape, mentioned below. Reference maps can be seen of each building in Figure 16 and 18.

5.3.1 E-huset

The measurements were performed three times from the same starting location, initialising at the same location every time, viewing the same scene. In each of the three measurements two laps were made. Eight reference points were placed. The distance traversed each lap was about 150 meters. In Figure 17 a few viewpoints are shown. In Table 1, average error between ground truth and reported points and vertical drift is shown.

5.3.2 A-huset

The A-huset measurements were done in a similar way as the measurement above but using four reference points. A distance of about 160 meters was traversed per lap. A few of the viewpoints can be seen in Figure 15. Loss of tracking or obvious deviations from the pose track occurred during both the second and third measurement. In Figure 22, the height drift is visualised. The vertical drift occurs in every measurement also shown in Table 1.

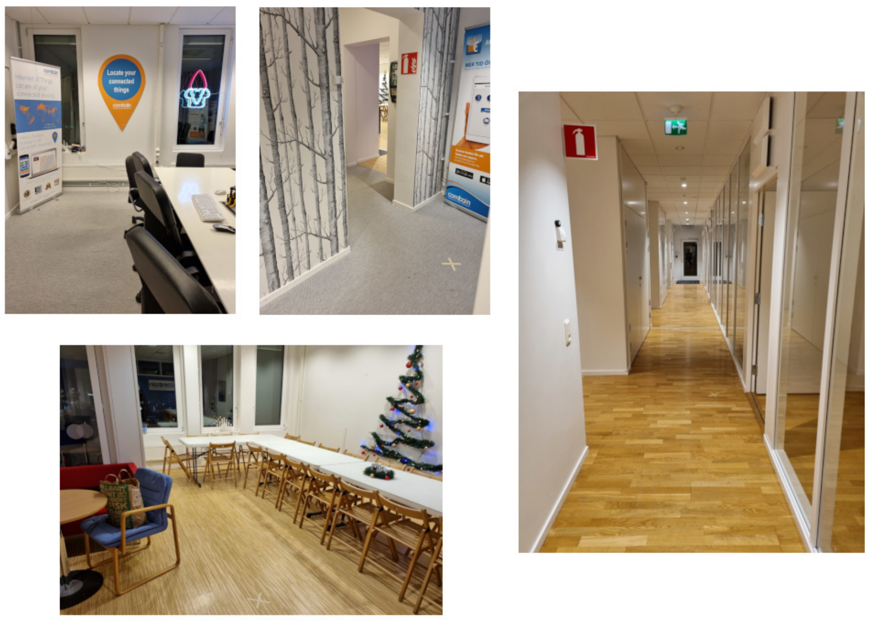


Figure 13: A selection of views from the measurement at Combain Office. The upper left view was the starting view each measurement, marked as 1) in Figure 14.

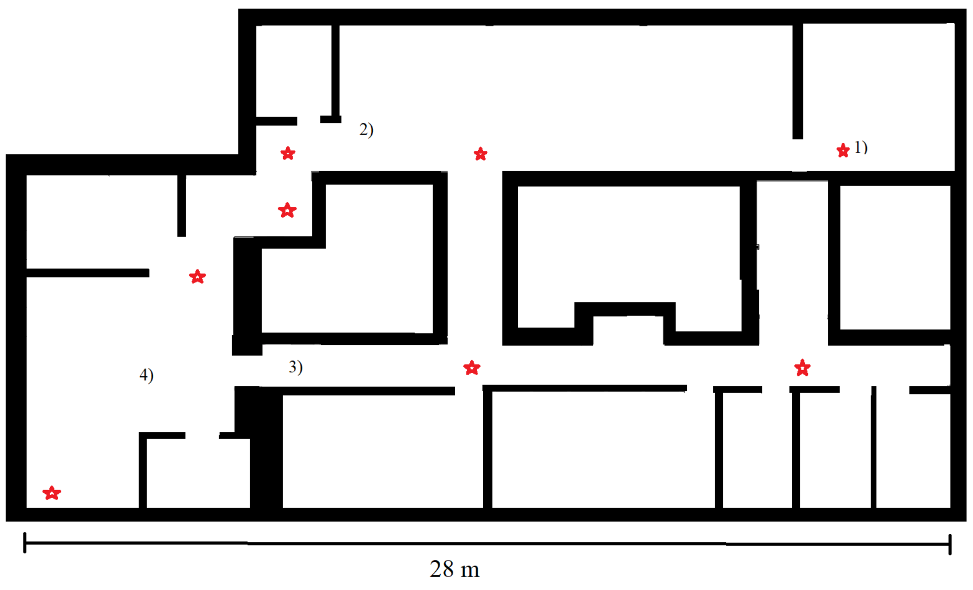


Figure 14: The office building blueprint. Each red star marks a reference points marked with tape during the measurement. The numbers correspond to the pictures in Figure 13, 1) to 4), left to right. Image 4) was taken facing the bottom left corner of the blueprint.

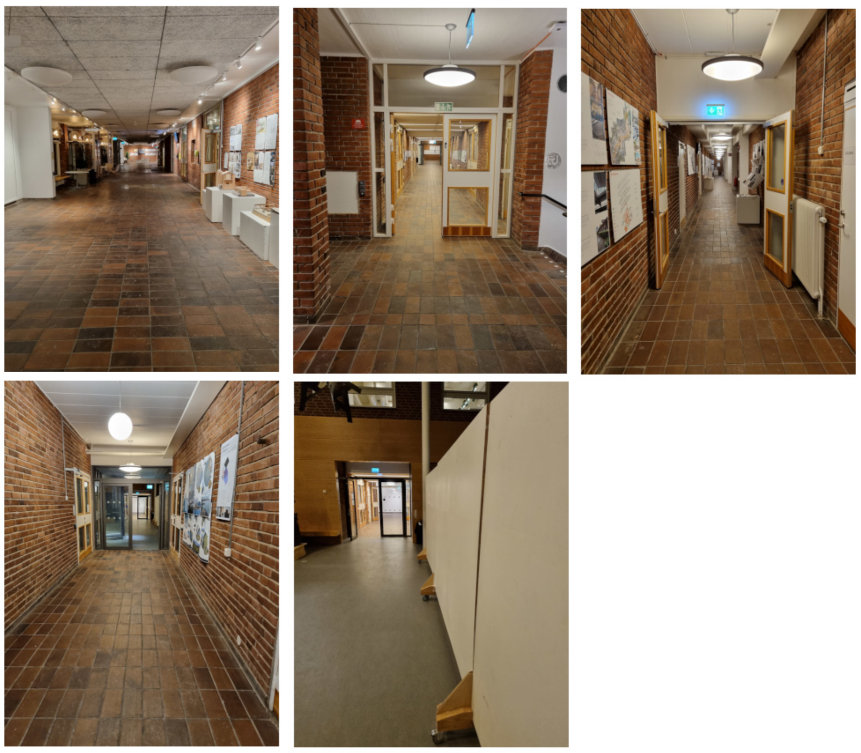


Figure 15: A selection of views from the measurement at A-huset, LTH, in chronological order. Compared to E-huset there are few white walls. In image five the temporary walls occluding the large room marked 5) in Figure 30 can be seen.

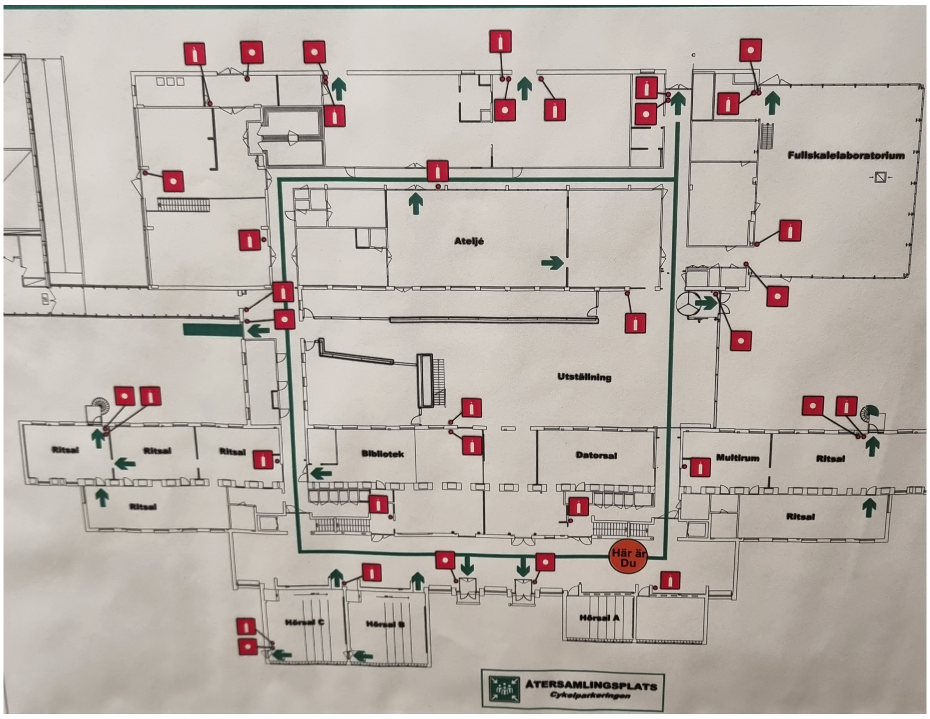


Figure 16: The evacuation plan map used as a reference. The already marked green line marking the pose track path. The red circle with the text "Här är Du" is slightly to the left of the starting position.

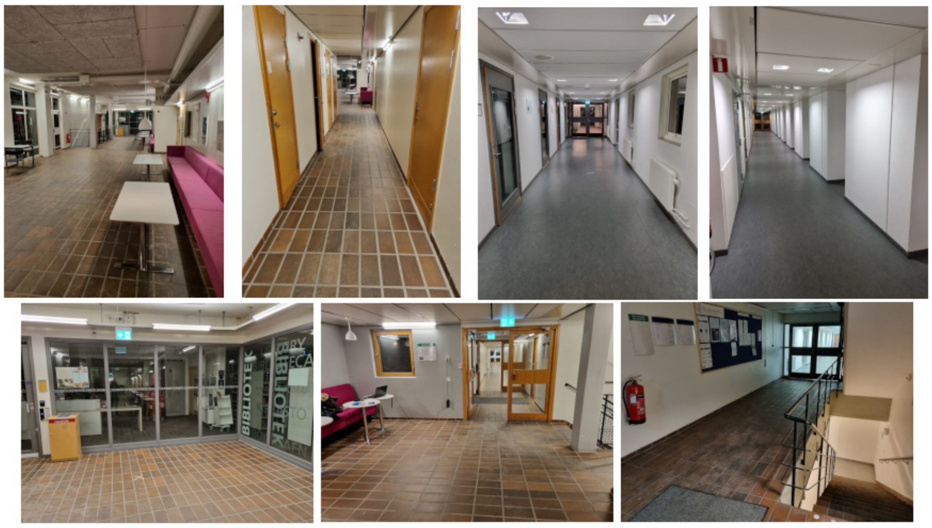


Figure 17: A selection of views from the measurement at E-huset, LTH, in random order.

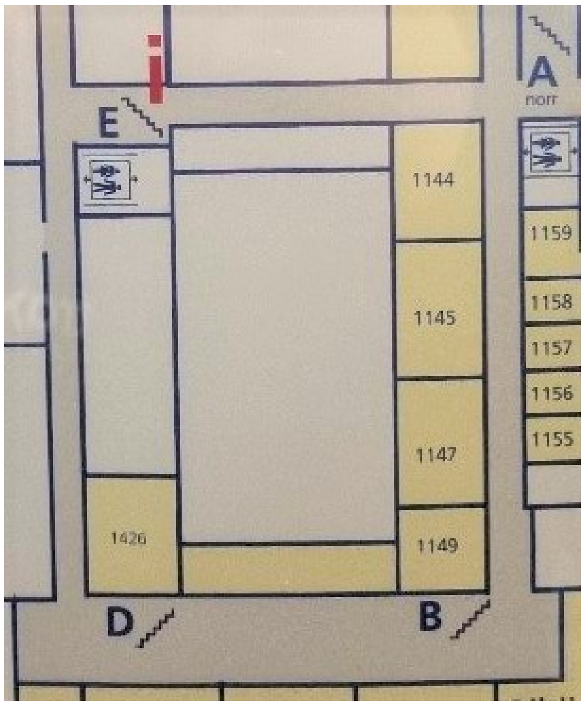


Figure 18: The evacuation plan map used as a reference. The staircase market B, bottom right, is next to the starting scene area. The route went from B to D and E and then back again to B. The staircase E can be seen in the bottom left image of Figure 17.

6 Results and Observations

The results presented below are divided into three sections, a section explaining the data recording and testing settings. After that, a summarised recording check list is presented. The last section presents and explains the map creation and analytic plots of the data. Some comments and discussion is included in this section if it applies to a specific measurement and not the overall results. About accuracy and precision; The accuracy of the results is used to describe how close a measurement is to the real world. Precision is used to describe the relative behaviour between measurements. In this sense, the accuracy of the measurement is considerably worse than the precision of the measurements. The system seems to produces inaccurate results in a reproducible (precise) way.

6.1 Data collection and measurements

6.1.1 Combain Office

Many more measurements had been done at Combain Office previously, than shown in this report. During the last set of measurements, some peculiar behaviours were noted. The latest set of tests were done just before Christmas celebrations. This made most areas of the office space occupied by some sort of furniture or decoration. In the depth image scatter plots it is possible to tell where a white table was place in a conference room, due to lack of data over that spot. The table was a glossy white, which the vSLAM system struggles to collect data from.

The distortion and warping seen in the pose track of Figure 20had not been observed in previous measurements. It is interesting to note that although the measurements are not accurate, they are to some extent precise. The area which seems to cause confusion in the system is seen in the image to the right, in Figure 14. In this set of measurement, the augmented image functionality, Appendix B.2.5, was not used. It is possible these images provides a way to force the system to correct for drift when returning the previous locations.

6.1.2 A- versus E-huset

What we can see in Table 1 is a encouraging and reinforces the intuition that for the most part, the camera track is precise. Given the difficulties of walking the exact same path repeatedly, the pose tracks seem to be consistent. This does not mean they are accurate. In every measurement there was a vertical drift, which seemed to affect the recognition of previously visited areas.

A-huset measurement 2, Table 1 in bold, differ from the two other measurements, as can also be seen in Figure 23. For consistency, the measurement should have been redone. The measurement was done at night and when entering a corridor, the lights timed out and shut off. This caused a sudden loss of traction which impacted the measurement. Ultimately the measurement was not redone due to it showing a possible behaviour of loss of tracking or impact of unwanted disturbances. If something goes wrong during a measurement, say a user does not understand how to properly do a measurement, the results will most likely have errors without robust post processing.

Looking closer here on the camera track and the anchor track, there are some odd behaviours that occur when tracking conditions worsen or calculations get overloaded. This may cause forced termination of some calculations in ARcore, resulting in more worse tracking. This can be seen as phantom areas, such as some pose tracks which appear shift position abruptly, Figures 21 and 22.

In test 2 at A-huset, Figure 35 in Appendix, the reported camera track was mostly continuous, without the sudden shifts in position resulting from loss of tracking or poor tracking. This is usually the case for loss of traction. However, the anchor positions differed more than a typical measurement. This is usually an indication of worse tracking resulting in an overall worse measurement. One way

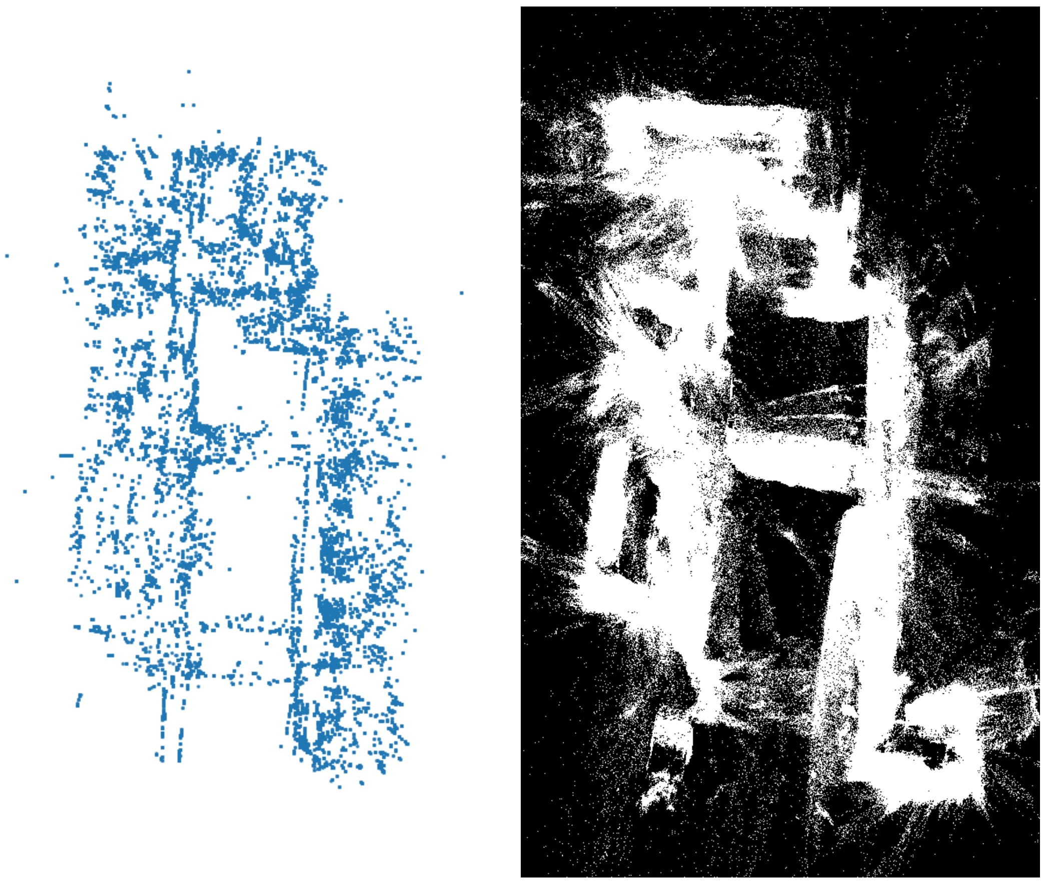


Figure 19: PC-points versus Depth Images, not from the same measurement. The right figure was recorded during 10 minutes and contains about 2000 times more PC-points. In the depth image recording, right plot, there was substantial distortion of the poses. This was not present in the left figure. Axes are not included but the scale in meters is known.

to possibly counter this is calculate a distance between corresponding anchor versus camera track points. If they misaligned, the measurement might be inconclusive, although not every case.

6.1.3 Performance issues during testing

It was also possible to visibly tell when the CPU was overloaded as the camera feed was stuttering and the app was written to report loss of tracking.

6.2 Recording techniques

An important aspect of the measurement is the recording technique itself. Below is a list outlining rules to follow when performing a measurement.

- Keep the camera pointed in your direction of movement
- Move slowly and keep camera steady

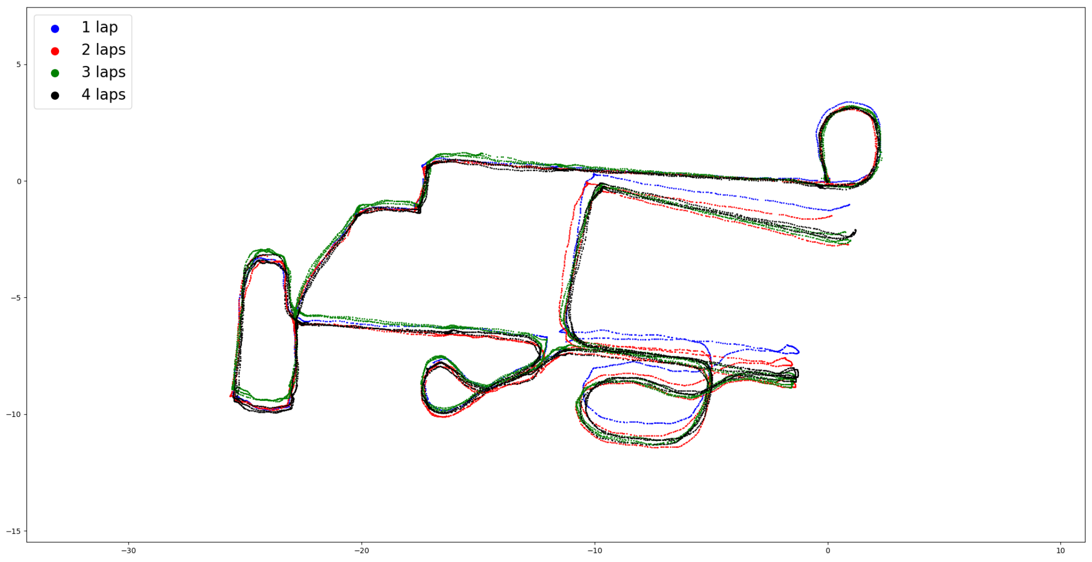


Figure 20: All 4 sets of corrected camera pose tracks from the Combain Office Measurements. Ignoring and correcting the warping of the pose track seen in the lower part of the pose track, possibly by loop closure, the poses are accurate and precise.

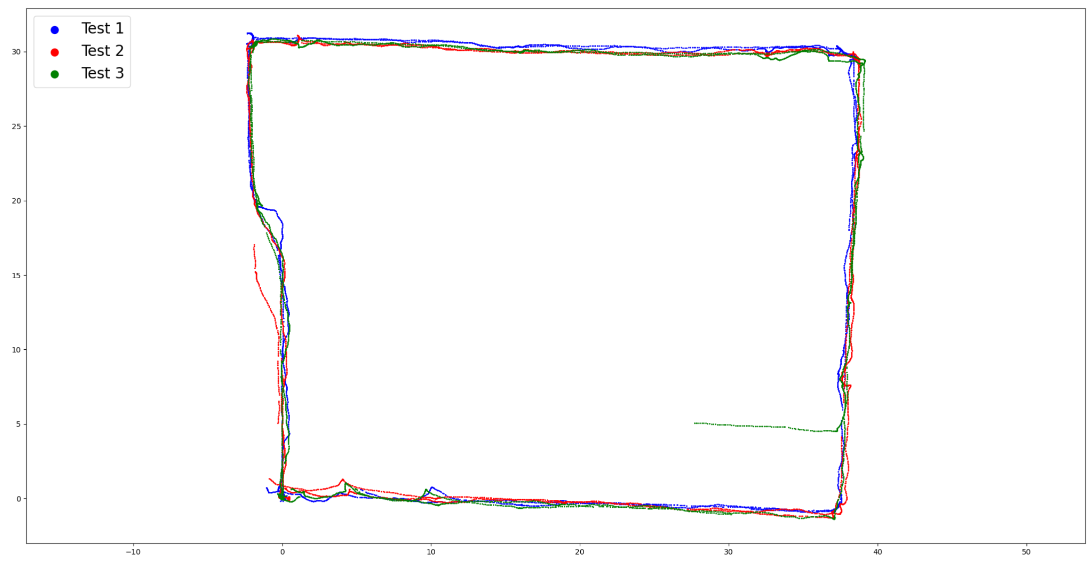


Figure 21: All 3 sets of angle offset corrected camera pose tracks from the E-huset Measurements. Both test 2 and 3, red and green, had major deviations in the track that snapped back to correct poses.

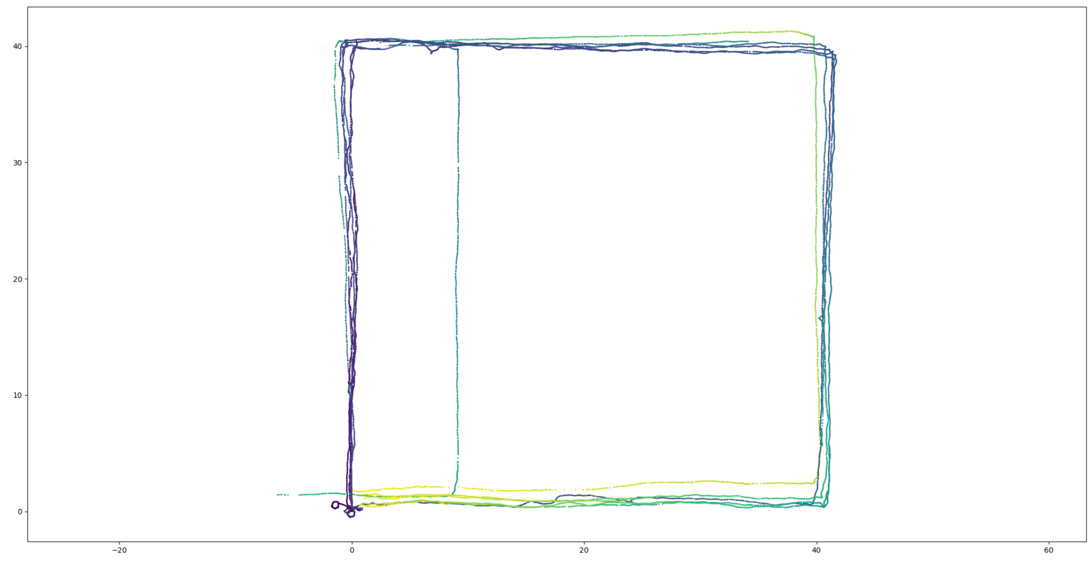


Figure 22: All 3 sets of angle offset corrected camera pose tracks from the A-huset Measurements. Both test 2 and 3, red and green, had major deviations in the track that snapped back to ”correct” poses.

Location	E-huset	E-huset	E-huset	A-huset	**A-huset**	A-huset	Vertical drift*
Test	1	2	3	1	**2**	3	
Mean distance (m)	0.09	0.09	0.1	0.15	**1.02**	0.06	1.46
Median distance (m)	0.06	0.05	0.06	0.08	**0.78**	0.05	1.5

Table 1: Evaluation of camera track precision and accuracy. Vertical drift was estimated visually by between comparing pose height at the start of each lap versus the end of a lap. Number 2 measurement in A-huset, marked bold is significant due to it was the only measurement were the mean distance was significantly larger than the others and the resulting map had a phantom corridor.

Location	E	E	E	A	A	A	CO	CO	CO	CO
Test	1	2	3	1	2	3	1	2	3	4
Duration (min)	7.36	7.3	6.73	7.97	8,47	6.90	3.85	7.43	10.35	10.2
PC-points (M)	0.0126	0.0127	0.0117	0.012	0.0125	0.0114	0.0059	0.0116	0.0171	0.0161
DP-points (M)	21.3	18.30	17.39	16.9	19.8	14.4	14.1	28.4	35.0	31.8

Table 2: E is E-huset at LTH campus, A is A-huset and CO is Combain Office at Ideon. Note the difference in units between PC-points and DP-points. Additionally measurement 4, with 4 laps was recorded in the same time as measurement 3. Duration (min) is in minutes and amount of points (M) is millions of points.

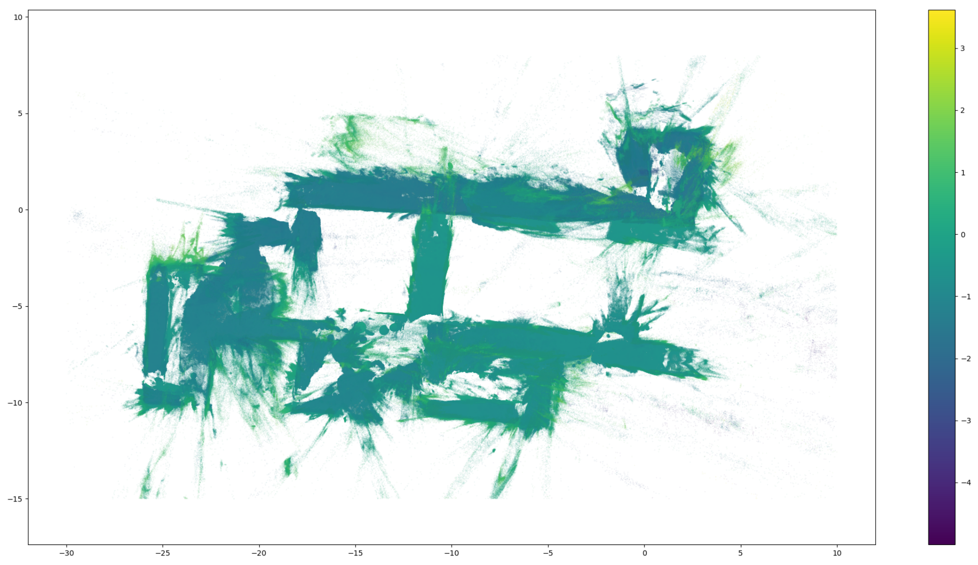


Figure 23: Scatter plot image of Combain Office walking one lap, colourcoded and a colourbar encoding for height. Most data points are on floor level, encoded with a deep green colour corresponding to a negative one meter height.

- Keep a clear view ahead, especially when navigating in tight spaces
- Avoid rotating around the spot more than 180 degrees
- Never rotate or turn in tight spaces looking into walls or other close and featureless objects

It is likely wise to never attempt to scan a room for supplementary depth data by rotating a whole revolution on the spot. If the intent is to map an area, a room or building the author finds the best way to do it would be as listed below:

- Plan out a route around the area and mark some spots the surveyor will return to
- Walk this route following the rules from the previous list 6.2
- Once a reference lap has been done, the scanning of the room can be done
- Walk the same route as before
- Deviate from path to slowly pan over an area of interest, being sure to scan the entire area, without rotating breaking any rules set above.
- Place out a few images scattered along the planned path to use as reference points and scan these every time they are passed.

Following this procedural it is possible to achieve a precise mapping of an area. It will likely need pose correcting, possibly by loop closure taking advantage of the reference points. There will likely be errors, drift and distortion in the camera pose track and ways to mitigate this will need to be developed.

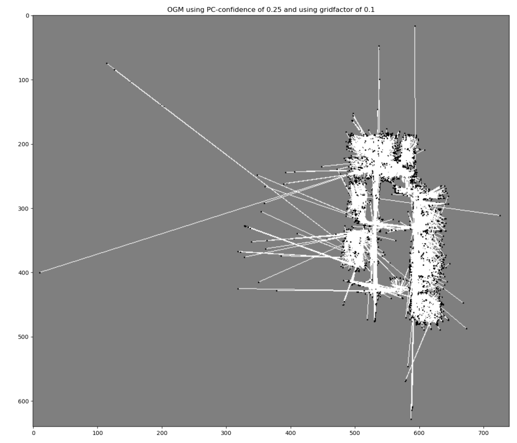


Figure 24: OGM using sensmodv1 without filtering the outliers

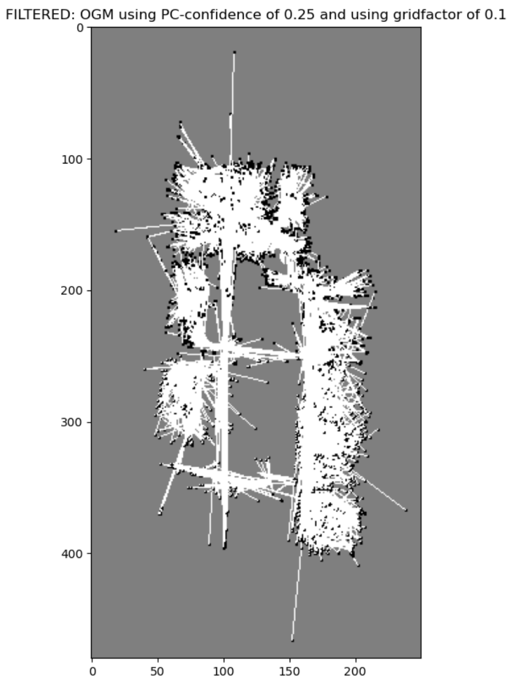

FILTERED: OGM using PC-confidence of 0.25 and using gridfactor of 0.1

Figure 25: OGM using sensmodv1 filtering the outliers. This OGM was created using the 30 minute data set containing 40,000 points and was the subjectively best OGM created of the Combain Office.

6.3 Map creation

6.3.1 Best Occupancy Grid Map using only point cloud data

As can be seen here in Figure 24,25 even though the data is sparse, the map generated is still fairly accurate. Some areas are innately hard to map, such as the main office room with many windows, screens and desks. Those areas are captured poorly on the OGMs, due to missing data.

6.3.2 Scatter plots using Depth Images

The resulting plots for E-huset can be seen in Figures 26, 27 and 28. The best and worst plots are shown. The last plot can be seen in Appendix 34. The same plots for A-huset can be seen in Figures 29, 30 and 31. The last plot is found in Appendix 35.

6.3.3 Best Occupancy Grid Map using Depth Images

A depth image measurement has potentially about 2000 times more data points than previous measurements. Thus it significantly increases the needed computation time. But it allows for greater flexibility. The map also arguably contains more noise that before, Figure 23

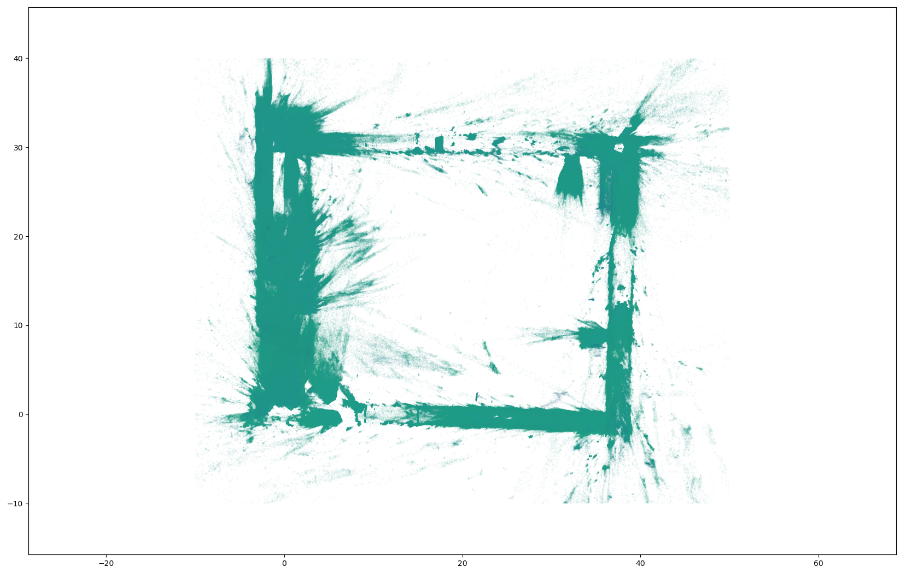


Figure 26: The first E-huset measurement. The resulting plot might look noisy and with a lot of erroneous points. But these outliers are comparatively fewer than the correctly placed PC-points.

6.3.4 OGM Sensmodv1

For the initial sensor model representation different types of weights to represent the line was used. These had minor impact on the final OGM. In part this was due to the line drawing algorithm. As previously mentioned, ARcore returns an confidence lever between 0 and 1. This was initially interpreted as the likelihood of that whole line to exist. This means drawing a line from the camera to a point using a low confidence point yielded in a "pale" line.

6.3.5 OGM Sensmodv2

After testing of the Sensmodv1 implementation of the sensor model it was realised a more unified and sophisticated sensor model was needed. It was described above in the Method section of the report. Due to a shift to focus to test depth images, Sensmodv2 was not extensively tested.

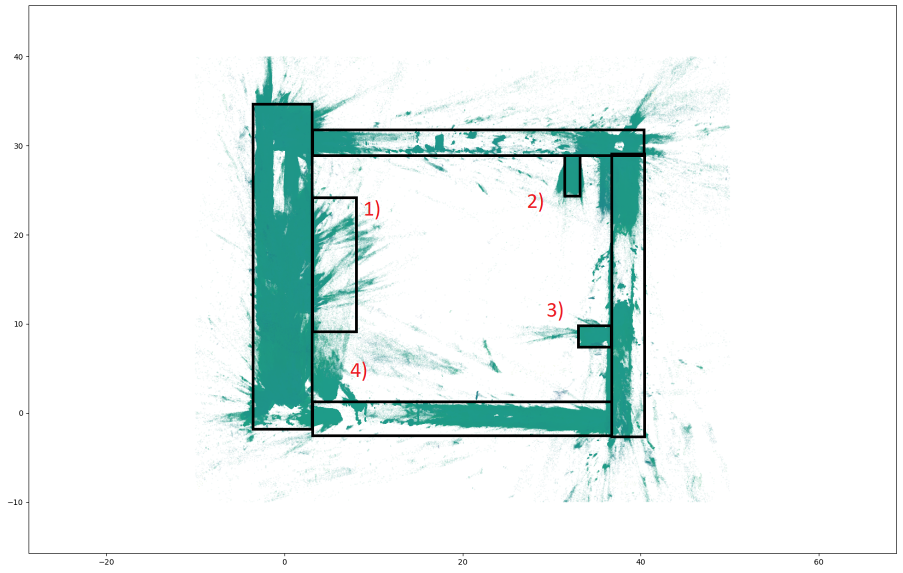


Figure 27: The first E-huset measurement, with areas of interest marked. Area 1) is a room with walls of glass. These readings are objects seen through the glass walls. They can still be seen as noise however.

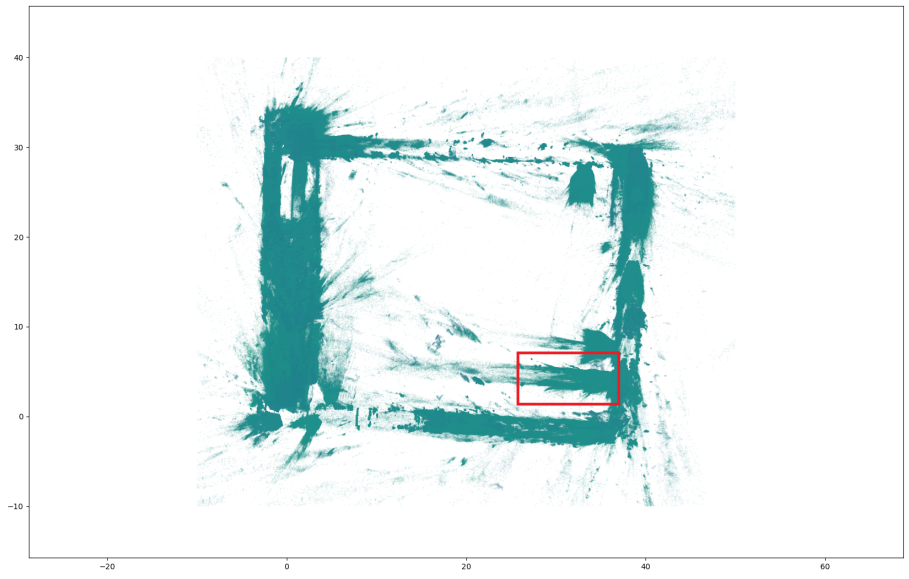


Figure 28: The result of a pose error, marked with red. This phantom area above the lowest corridor is incorrect by several meters.This is an example of a possible result of combined height drift and computational overload resulting in an error in pose estimation.

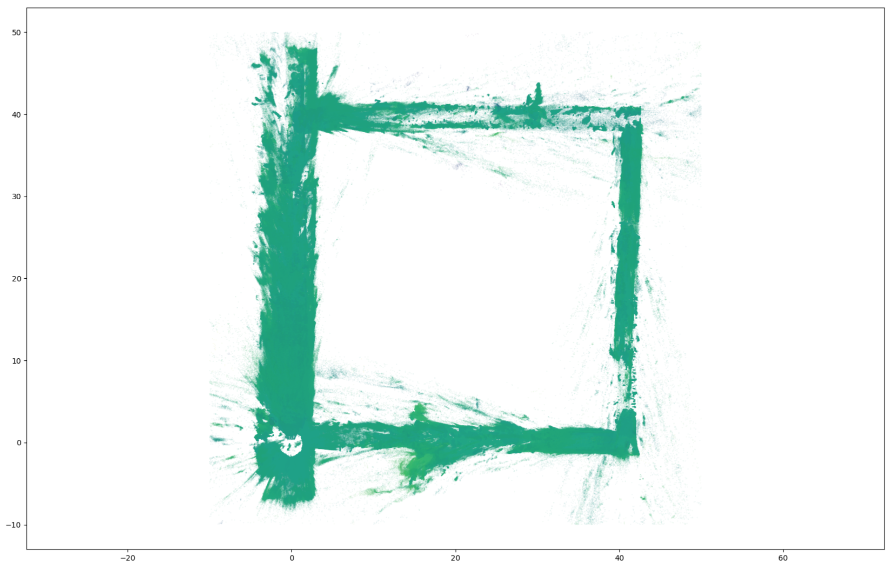


Figure 29: The second measurement i A-huset at LTH campus. In Figure 30 the rooms are placed as a reference. This building was mostly constructed of bricks, both floor and walls, and resulted in less empty areas than the measurement in E-huset. This was expected and was why these two building were selected.

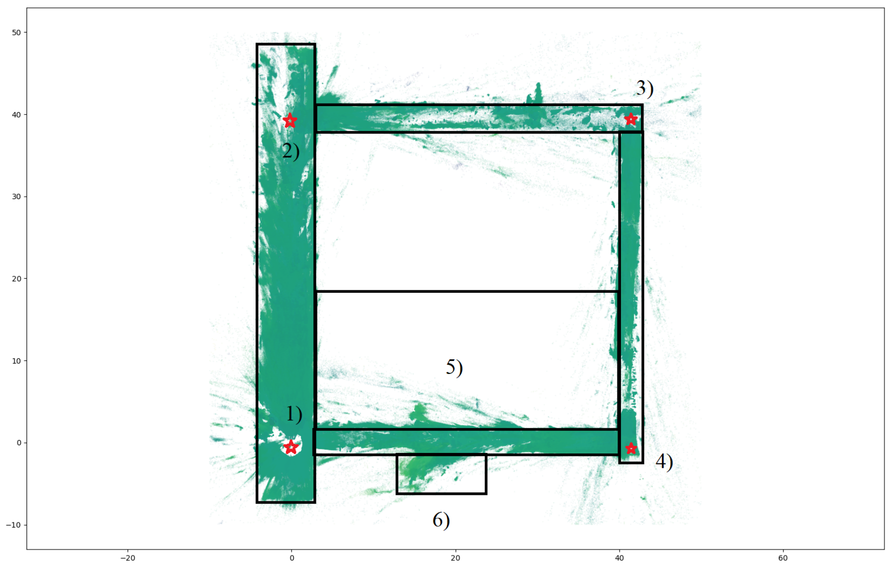


Figure 30: Each reference point marked with red and a number, corresponding to the images in Figure 15. The area marked 5 was at the time of measurement partially hidden from view by temporary walls. Area 6 was a glass entrance. Beside the slight warp between the number 3 and 4 reference points, the recorded map fills most of the map rooms.

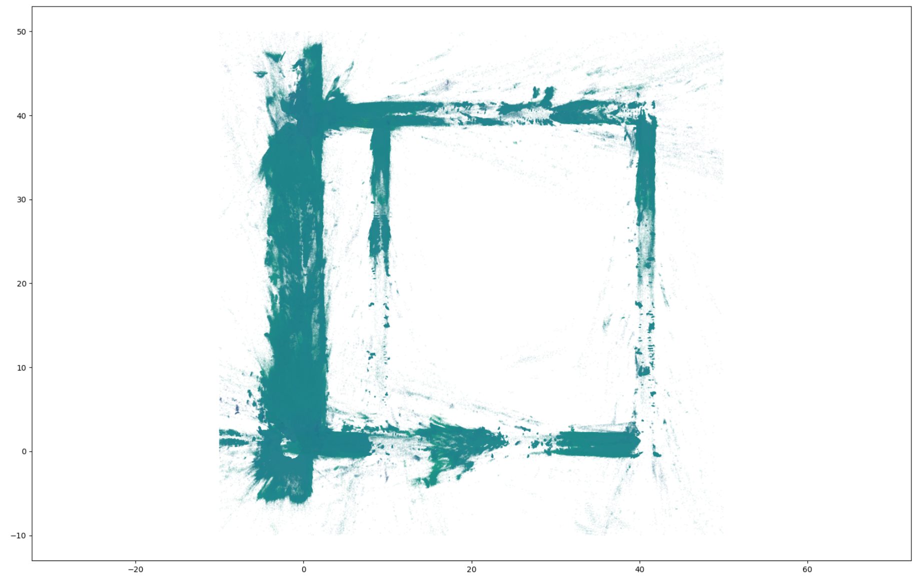


Figure 31: Third measurement at A-huset. The same issue of phantom corridors can be seen here in this figure. The loss of tracking at one point led to odd behaviour.

7 Discussion

The results and methods provided and tested in this report may not themselves supply a traditional 2d blueprint with straight walls and square rooms. But it does supply surprisingly accurate visualisation of an indoor space. An interesting next step would be to send in this representation into a neural network or tradition image processing that would convert this map into a likely layout of rectangular rooms. In other words, there is a good potential to use this work and methods to estimate your own maps and build upon this work.

As smartphones and cameras improves it is unlikely the computational limitations will not improve due to access to more processing power. Hopefully each new generation of smartphone will produce more data and house a more accurate vSLAM system.

The set of measurement most figures are created from was devised for a few reasons; To confirm the intuition that the camera track recorded was precise given a specific recording method. Beside the occurrences where the pose track is creating phantom tracks due to poor tracking, this seems to be true. The idea behind this report comes from the need to evaluate the performance of ARcore and of the pose track it produces. If this track proves to be inaccurate, the very foundation of this project looses some of it is appeal. Another reason for focusing on poses is due to the map accuracy and precision directly correlates to the accuracy and precision of the pose track. Since it is not feasible to create an unlimited amount of anchors, due to their processing cost and instability issues, the untracked camera track is what is being used to produce an OGM. Thus evaluating the pose track is key to evaluate performance and find error sources. In Figures 28 and 30 the resulting errors of pose estimations are marked. By achieving an accurate pose estimation first, it is arguably easier to deal with the inaccuracies of the PC-points themselves.

7.1 Performance

ARcore runs on the mobile system using several threads. When stationary and having ARcore running, it uses up to 30% of computational resources. When adding features and data storing to the application, the performance of the SLAM system may be affected. When developing functions and testing unoptimised code, that code would seemingly disrupt and steal resources from the backend. Due to this it is important to minimise the amount for computational load added to the ARcore system as it may negatively affect the entire SLAM session and recording accuracy. It was also possible to visibly tell when the CPU was overloaded as the camera feed was stuttering and the app reported loss of tracking. And these instances seems to correlate to when the resulting pose track produces erroneous track segments, such as can be seen in Figure 28.

7.2 Vertical drift

The vertical drift of the pose track was a unforeseen discovery of the tests at Campus. It seemed to indicate drift and distortion as seen in Figure 22. During the measurements at A-huset the vertical drift increased during each lap, and seemed to cause issues for the system to recognise it was back at the start of the measurement again. At some point, the system snapped the pose track back to the starting height level and recognised it was back at the start. Attempting to use the vertical drift as an indicator of overall drift may be useful. Although it will be challenging to deal with true vertical changes if always assuming a building is flat. Performing measurements in closed loops gives a reference between end of a lap and start of a lap in regards to the height. If they are not reporting the same height there may be drift related errors in the measurement.

7.3 Further work

The OGMs generated are possibly suited as the input in an image processing model or neural network. From the map it would be possible to train a system to recognise building shapes and using this map to place walls and rooms where it seems likely based on training and information in the map itself.

If this project was continued for another four months, some next step would be to look into depth images and loop closure of pose estimations. Another aspect the depth image created maps brought to light is if OGMs are the best way forward. As mentioned in related studies there are other different methods available to fill gaps in surfaces, such as Poisson Surface Reconstruction.

As shown in this report, generally the camera pose track is good and precise, but not sufficiently accurate without post processing methods to correct for errors. During longer measurements of larger areas and building additional drift and distortion is likely. A robust approach to combat this is likely needed.

7.4 Alternative methods to create a blueprint of an area of interest

Another way of approaching the issue of map creation would be to ask the user to simply find the emergency map, present in public building, Figures 16 and 18. Considering the detail found in those maps, it may be feasible to use image analysis methods of these maps and produce a simple blueprint that way. Additionally this map might be used as a baseline, to build upon for the ARcore generated map.

7.5 Using data from depth images

The results from using the depth images was deemed to most promising way forward. The depth image part of this project was started a few weeks before the end of the project. Due to the difference in amount of data between methods one and two, most plots shown created using depth images were scatter plots. This was due to challenges in how to use this different data with the same OGM method. The second method produced mostly floor points, whereas method one rarely recorded floor points.

There are new issues and advantages introduced by method two. In a way, it flips the issue around. Not only that there are more data, but previously most measurements of the floor was avoided. Now all points of the floor are advantageous, they can be used to mark out the floor. Most areas of a recording using method two are covered. As previously discussed, this questions the need for using OGM to create a fully filled map.

Because the depth images introduced more than three orders of magnitude more data, there was a lack of time to examine new possibilities of these results. There is now a greater need to downsample the data in order to within reasonable limits, calculate an OGM. This creates a need for further development when discretising the image for example.

7.6 The unexpectedness of the testing results

The Campus testing results resulted in surprising results and observations. For example, no previous tests at Combain Office had resulted in the distortion seen in Figure 19. For some or several reasons, the results showed the same distortion over several measurements. It is interesting to see that the distortion is precise, meaning the same error occurs every lap. ARcore is sometimes wrong, but it seems to still be precise, even when it is wrong.

Another unexpected result was the quality and amount of depth image data. During most of the project, the focus was on sparse PC-points supplied by a point cloud function, method one.

7.7 Challenges for the SLAM system

A vSLAM system has difficulties detecting points on texture sparse features such as walls and windows. A white wall tends to generate fewer features than a brick wall. Some areas will inevitably return less data than others. Floors tends to have more texture and this is taken advantage more from method two.

Windows cause issues with reflections and the transparency of glass is challenging for a visual system. The measurement may become inconclusive, there may be recorded points far away in the distance or they may record something that is actually there on the other side of the window. ARcore has a built in feature which estimates the quality of PC-points, but this seems not to be a reliable way to filter out phantom points from windows.

8 Conclusions

To summarise the work presented in this report the author would like to emphasise a few findings.

The vertical drift that occurs during longer measurements seems to be a cause or symptom of the warping seen in the presented results.

Collecting and processing depth images is the best way forward due to of the reasons discussed above, flexibility, amount of data, relative accuracy. The author deems it ineffective and impractical to only rely on the point clouds from the API directly, due to their randomness and sparseness.

There is testing and more work needed to make this a working product, especially ways to efficiently and robustly minimising the amount of data collected and processed without impacting results. The optimisation of data storage and recording is especially important as it seems the performance of ARcore may degrade if the computational load is too high.

Accurate pose estimation is key and the author suggests doing a reference lap first, a lap with as little other processing but tracking from SLAM, and then the second lap which can scan and record environment data. By setting up reference points, possibly using the image detection, it may force the system to update position. Additionally these reference points will be useful to compensate for drift and warping of the pose track and corresponding map.

References

[1] Google. About arcore. `https://arvr.google.com/arcore/`.

[2] Apple. More to explore with arkit 6. `https://developer.apple.com/augmented-reality/arkit/`.

[3] Tobias Feigl, Andreas Porada, Steve Steiner, Christoffer Löffler, Christopher Mutschler, and Michael Philippsen. Localization limitations of arcore, arkit, and hololens in dynamic large-scale industry environments. pages 307–318, 01 2020.

[4] Julien Valentin, Adarsh Kowdle, Jonathan T. Barron, Neal Wadhwa, Max Dzitsiuk, Michael John Schoenberg, Vivek Verma, Ambrus Csaszar, Eric Lee Turner, Ivan Dryanovski, Joao Afonso, Jose Pascoal, Konstantine Nicholas John Tsotsos, Mira Angela Leung, Mirko Schmidt, Onur Gonen Guleryuz, Sameh Khamis, Vladimir Tankovich, Sean Fanello, Shahram Izadi, and Christoph Rhemann. Depth from motion for smartphone ar. *ACM Transactions on Graphics*, 2018.

[5] Yuwei Chen, Jian Tang, Changhui Jiang, Lingli Zhu, Matti Lehtomäki, Harri Kaartinen, Risto Kaijaluoto, Yiwu Wang, Juha Hyyppä, Hannu Hyyppä, Hui Zhou, Ling Pei, and Ruizhi Chen. The accuracy comparison of three simultaneous localization and mapping (slam)-based indoor mapping technologies. *Sensors*, 18:3228, 09 2018.

[6] Thomas Collins, J.J. Collins, and Donor Ryan. Occupancy grid mapping: An empirical evaluation. In *2007 Mediterranean Conference on Control Automation*, pages 1–6, 2007.

[7] Flood, Gabrielle and Gillsjö, David and Heyden, Anders and Åström, Kalle. Efficient Merging of Maps and Detection of Changes. In Felsberg, Michael and Forssén, Per-Erik and Unger, Jonas and Sintorn, Ida-Maria, editor, *Image Analysis - 21st Scandinavian Conference, SCIA 2019, Proceedings*, volume 11482 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 348–360. Springer, 2019.

[8] Google Inc. Google, fundamentals | arcore | google developers, 2022. `https://developers.google.com/ar/develop/fundamentals#motion_tracking`.

[9] Stackoverflow. What sensors does arcore use? `https://stackoverflow.com/questions/54356589/what-sensors-does-arcore-use`.

[10] Johannes Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. 06 2016.

[11] Karl Granström, Thomas B Schön, Juan I Nieto, and Fabio T Ramos. Learning to close loops from range data. *The International Journal of Robotics Research*, 30(14):1728–1754, 2011.

[12] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment — a modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 298–372, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[13] Frincy Clement, Kirtan Shah, and Dhara Pancholi. A review of methods for textureless object recognition, 2019.

[14] Google Inc. Depth adds realism | arcore | google developers., 2022. `https://developers.google.com/ar/develop/depth#understand-depth-values`.

[15] Google Inc. Google, frame | arcore | google developers., 2022. `https://developers.google.com/ar/reference/java/com/google/ar/core/Frame`.

[16] Google Inc. Google, use raw depth in your android app | arcore | google developers, 2022. `https://developers.google.com/ar/develop/java/depth/raw-depth`.

[17] Google. Build global-scale, immersive, location-based ar experiences with the arcore geospatial api. `https://developers.google.com/ar/develop/geospatial`.

[18] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, 1987.

[19] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[20] Kurt Konolige. Improved occupancy grids for map building. *Autonomous Robots 4*, pages 351–367, 1997.

[21] Huizhong Zhou, Danping Zou, Ling Pei, Rendong Ying, Peilin Liu, and Wenxian Yu. Struct-slam: Visual slam with building structure lines. *IEEE Transactions on Vehicular Technology*, 64(4):1364–1375, 2015.

[22] Ekaterina Sukhareva, Tatiana Tomchinskaya, and Ilya Serov. Slam-based indoor navigation in university buildings. pages 611–617, 01 2021.

[23] Hannah Patronoudis. Creating an arcore powered indoor navigation application in unity. `https://www.raccoons.be/resources/insights/creating-an-arcore-powered-indoor-navigation-application-in-unity`.

[24] P. Afsén and K. Boye Frick. Aiding in the visualization of tagged items in a point cloud. 2022.

[25] Colin Flanagan. The bresenham line-drawing algorithm. `https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html`.

[26] Github. Arcore sdk for android v1.31.0. `https://github.com/google-ar/arcore-android-sdk/releases/tag/v1.31.0`.

[27] Samsung. Specifications. `https://www.samsung.com/global/galaxy/galaxy-s21-5g/specs/`.

A Additional Figures

In this section extra plots are shown.

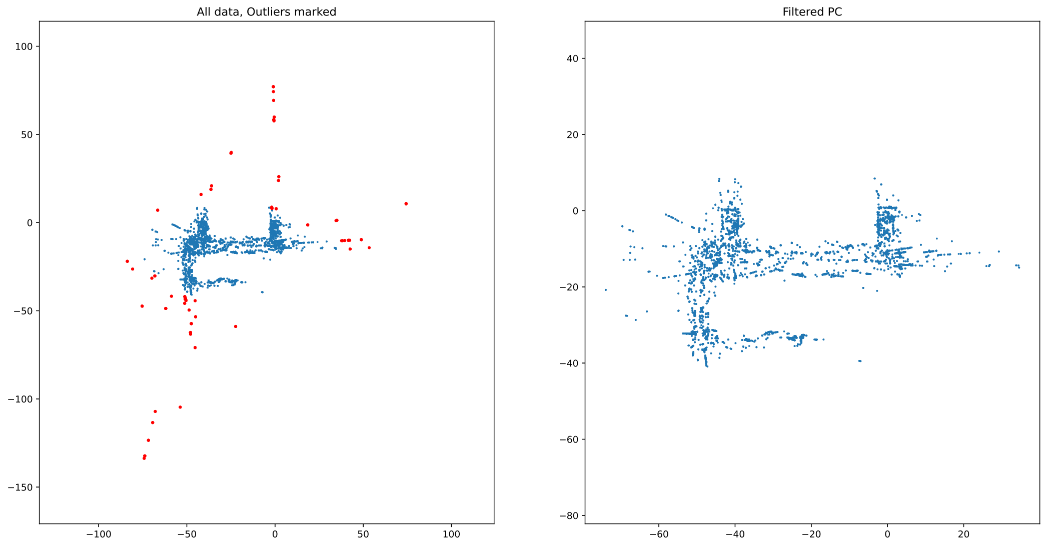


Figure 32: Recording of "Matteannexet" at Campus LTH. To the left, the outlier filtering is visualised by marking outliers as red. These were removed in the right image.

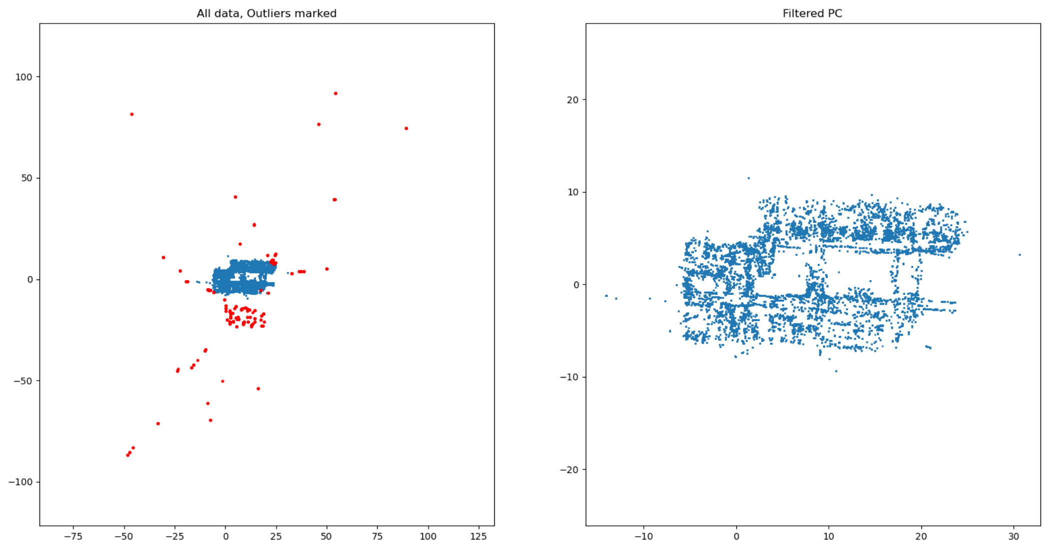


Figure 33: Recording of Combain Office at Campus LTH. To the left, the outlier filtering is visualised by marking outliers as red. These were removed in the right image.

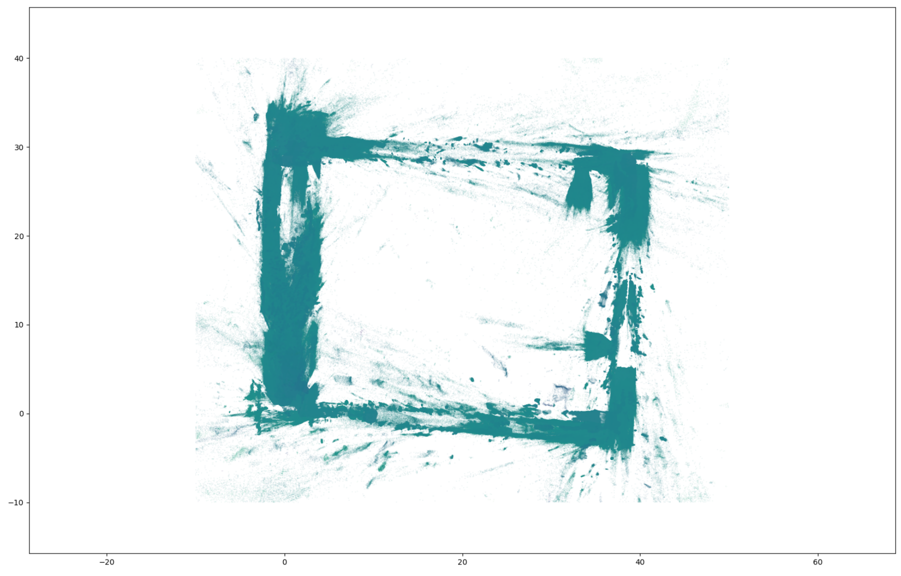


Figure 34: The second measurement at E-huset.

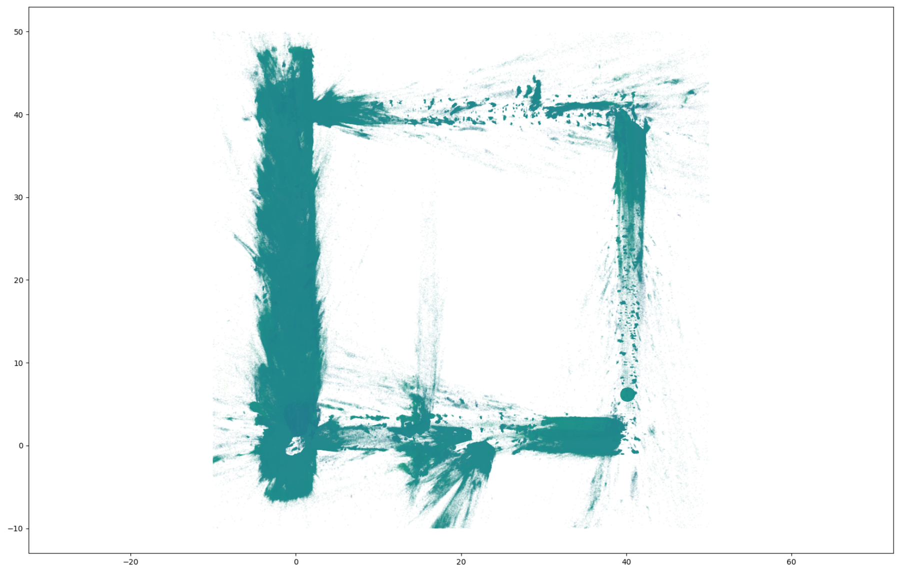


Figure 35: The second measurement at A-huset.

B Cooling and additional key concepts

B.1 Cooling the hardware

In order to avoid software crashes, an innovative water cooling system was used. It consisted of a simple 3d printed holder. The cooling system was placed in direct contact with the phone. The cooling system of choice was a fibrous piece of thin papercloth. It was drenched in cool water and was found to aid the stability of a measurement significantly. The watercooling holder can be seen here in Figure 36.

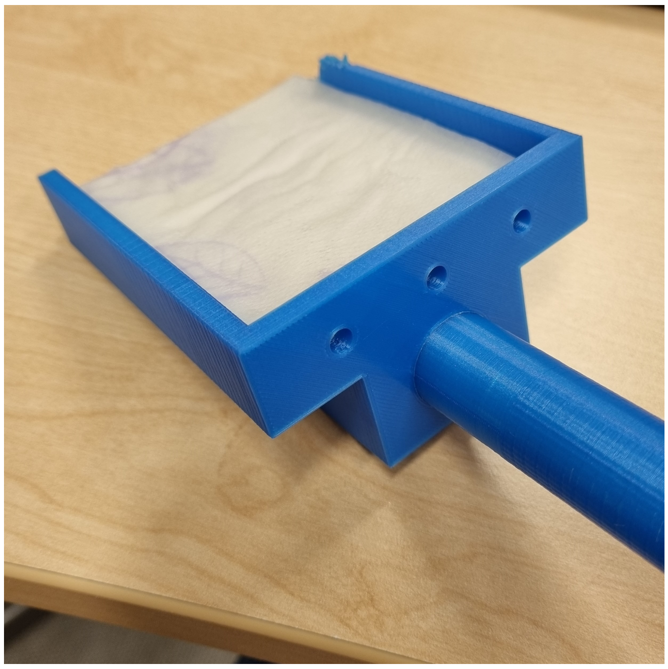


Figure 36: (Passively) water cooled phone holder, which significantly helped cool the phone during measurements

B.2 Key concepts

Below are sections describing important functions and behaviour used in testing, along with relevant links. Intended as a cheat sheet both for the writer and the reader. Most information is from Google's developer guide with some information complemented or excluded.

B.2.1 Pose

A pose in AR-context is a description of a objects position and orientation in the world. Mostly used in this project is the camera pose.

> "A pose represents an immutable rigid transformation from one coordinate space to another. Poses always describe the transformation from object's local coordinate space to the world coordinate space.

- Can be thought of as equivalent to OpenGL model matrices.
- The transformation is defined using a quaternion rotation about the origin followed by a translation.
- Coordinate system is right-handed, like OpenGL conventions.
- Translation units are meters."

`https://developers.google.com/ar/reference/java/com/google/ar/core/Pose`

B.2.2 Camera Pose and Android Sensor Coordinate System

The camera pose estimation uses android sensor system, which itself uses different sensors. There's acceleration-, gravity-, linear acceleration-, geomagnetic field- sensors and a gyroscope. When using Geospatial Anchors the GPS-sensor is also used.
`https://developer.android.com/guide/topics/sensors/sensors_overview#sensors-coords`

B.2.3 Trackables

A Trackable is an object that maps position data such as poses and points between the real world and the internal SLAM map. Any measurement that needs to be continuously tracked needs to be a Trackable. As explained below a Trackable does not have a fixed position in the internal SLAM map. Their positions may be updated on a per frame basis. Thus it is not reliable to compare the same objects coordinates between two frames. As the world understanding improves, these Trackables are constantly being iterated and update. There's a limit to the amount of Trackables one can reliably store without stability issues.

B.2.4 Geospatial Anchors and VPS

The Geospatial API enables the use of VPS, Google's "Visual Positioning Service". By Google developers it was pitched as improving the accuracy in places such as big cities where the GPS signal is poor. This is achieved by finding landmarks from Google's 3D mapping of cities. It also gives indoor readings, with a reported accuracy between 2-10 meters depending on the building. Hence the Geospatial mode is useful when trying to automatically align a real world map such as OSM with the relative SLAM map. It uses more than just the VPS when indoors, and somehow merges local SLAM map to VPS readings, but I haven't found any description of how it achieves this. The accuracy in latitude, longitude and height is defined by a probabilistic reading where it is a 68% probability of finding the points within given range in meters.
`https://developers.google.com/ar/develop/java/geospatial/developer-guide#java`

B.2.5 Augmented Images, ARcore built in Image Detection

There's built in image detection for detecting known images. To use it one creates an image database with supplied tool. The tool also grades the quality of the images in regards to detecting of the image.
`https://developers.google.com/ar/develop/java/augmented-images/guide`

C Populärvetenskaplig sammanfatting: Kartläggning av inomhusmiljö med en smartphone

Här står du, vilsen i en helt nybyggd galleria du aldrig besökt tidigare. Du vill finna den nya butiken Truls Tablåer. Men du hittar ingen karta, det är tomt i gallerian och den är dessutom stor. Men med den nya appen Galleriaguiden på din mobil kommer du med enkelhet hitta till rätt butik, bara följ markeringarna i skärmen på och låt appen guida dig rätt!

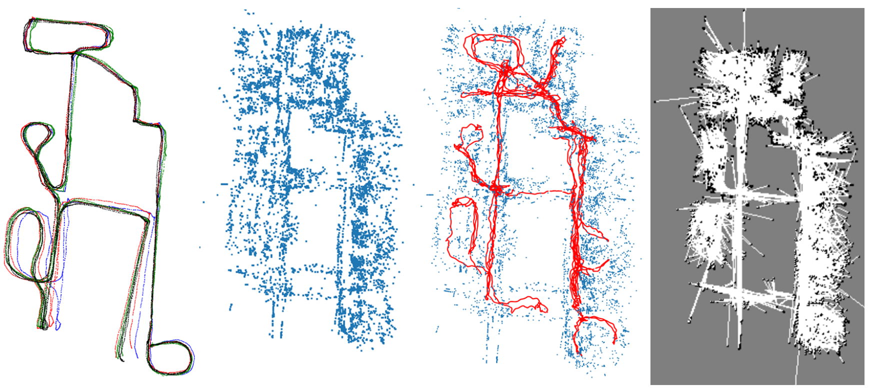
Inomhuspositionering är ett viktigt område för en användare som exempelvis försöker hitta till en viss butik i ett köpcentrum. I detta arbete används Googles inbyggda teknik för ”Augmented Reality”, AR, för positionering och ritning av karta. En GPS, såsom den som du använder redan i din smartphone, får ingen GPS-mottagning inomhus. Detta innebär att andra metoder behöver användas för att uppnå god inomhuspositionering. Då lämpar sig den teknik jag har utvärderat i mitt arbete väl.

En kamerateknik kallad SLAM har använts*. Tekniken utnyttjar sig av mobilkamera och andra sensorer i mobiltelefonen för att uppskatta hur en användare rör sig. Detta kombineras med aversuståndsuppskattningar till väggar och andra objekt, såsom dörrar eller andra fasta punkter i ett rum.

I figuren nedan illustreras ett förenklat flöde av hur man skapar en karta. Genom att utnyttja att man vet relativt säkert hur ”filmaren” har gått under, dversus hur mobilen har rört sig, och aversuståndet till olika punkter (punktmoln) kan man skapa kartan längs till höger i bild. Denna karta har en känd skala, vilket innebär att storlek på rum och aversuštånd till väggar etc kan bestämmas med en decimeter till meterprecision. En undersökning av ett litet kontorslandskap tar omkring 10 minuter!

Rapporten finner att de är möjligt att skapa inomhuskartor av områden. Dock återstår arbete för att vidareutveckla tekniken till en produkt för en typisk användare. I och med den eviga tekniska utvecklingen kommer troligtvis begränsande faktorer såsom beräkningskapacitet i smartphones att minimeras och kartorna kan då förbättras markant.

*SLAM står för Simultaneous Localization And Mapping och tekniken utför lokaliseringen och spårning samtidigt.



Från vänster till höger: Spåret från en undersökning läggs ihop med varandra, bild 1 och 2. Dessa resulterar i bild 3. Bild fyra visar en karta som skapas genom att linjer dras från varje kameraspårspunkt till korresponderande punkter av fasta objekt.