Pointcloud scan selection for indoor floor plan generation

Pointcloud scan selection for indoor floor plan generation

By Cristian Frincu, B.Eng

A Thesis Submitted to the School of Graduate Studies in the Partial Fulfillment of the Requirements for the Degree Masters of Science

McMaster University

Masters of Science (2019)

Hamilton, Ontario (Department of Computing and Software)

TITLE: Pointcloud scan selection for indoor floor plan generation

AUTHOR: Cristian Frincu (McMaster University)

SUPERVISOR: Dr. Rong Zheng

NUMBER OF PAGES: xi, 49

Abstract

Building Information Models (BIM) are becoming a standard in the construction industry for storing information about buildings and assets. Automatically creating BIMs has attracted a lot of attention, as it has great potential to improve efficient resource management. A detailed description of the building can decrease the cost of management, heating and cooling, and restoration. For pre-existing structures design documents are typically outdated or unavailable, making BIMs challenging to acquire.

The field of indoor floor plan creation has grown in recent years due to advancements in LIDAR technology. However, LIDARs create millions of points per scan, making it computationally expensive to process all of them. In order to properly create a floor it is imperative to acquire a sufficient number of scans to visualize the whole building, while simultaneously minimizing the number of scans for computational reasons. We propose a method for selecting a subset of the scans, as well as a method for clustering points into lines to be used for floor plan extraction. Our method works by clustering nearby points, creating a convex hull around them, and selecting scans based on the most area covered by the union of the hulls. The point clustering splits the pointcloud into potential lines by projecting each point along its surface normal, clustering points from the same line together. Those improvements allow for the efficient generation of floor plans for large buildings.

Acknowledgements

I would first like to thank my supervisor Dr. Rong Zheng for giving me opportunity to continue my studies at McMaster University. She has always been available and prepared answer anytime I ran into trouble with research. I am thankful for allowing me to choose the direction of my research and for encouraging independence.

From the time I started working in the lab as an undergraduate student to now, everyday has been a very rewarding experience. The IPS competition with Jun, Chenhe, Yuting and Zhe has been a great experience, a good introduction to indoor localization. I will also not forget all the time spent working on course projects with Mehdi , the coffee breaks with Jean Lucas and the interesting conversations with Yihao. Special thanks to Yongyong for all the help debugging our programs, he has been a very resourceful partner.

I am grateful for my parents for providing moral and emotional support all my life. I would not be able to get this far without their selfless love and support. I would also like to express my gratitude to my girlfriend, Loryn for all the understanding and encouragement she offered.

Contents

List of Figures

List of Tables

Acronyms

3D 3-Dimensional

AEC Architecture Engineering and Construction

AMCL Adaptive Monte Carlo Localization

BIM Building Information Model

CAD Computer Aided Design

HVAC Heating Ventilation Air Conditioning

ICP Iterative Closest Point

ITB Information Technology Building

LIDAR Light Detection And Ranging

MC Microcontroller

RANSAC Random Sample Consensus

RGB Red Green Blue

RGBD Red Green Blue Distance

ROS Robot Operating System

SLAM Simultaneous Localization and Mapping

USB Universal Serial Bus

Chapter 1

Introduction

Having a document describing the built stages a building has gone through, as well as the materials used and the building schedule, is a way of lowering the costs for further repairs. Building Information Model (BIM)s are becoming a standard in the construction industry for storing information about buildings and assets. Automatically creating BIMs has attracted a lot of attention, as it has great potential to improve efficient resource management. A detailed description of the building can decrease the cost of management, heating and cooling, and restoration. For pre-existing structures design documents are typically outdated or unavailable, making BIMs challenging to acquire. Modern distance measuring technology is making the task of constructing a floor plan more feasible but it comes with it's own challenges. One of the challenges introduced by the new technology is the great amount of data produced by those devices. In this thesis we examine methods of keeping a subset of data produced, with the goal of lowering the amount of storage and computation required.

1.1 Building information models

Semantically meaningful 3D models are becoming used more often in the Architecture Engineering and Construction (AEC) domain. The use of such models affect all stages of construction, starting from the design phase, to construction and following through to maintenance and repairs. These models, are generally known as BIMs. Depending on the available data, and the standard to which they comply, BIMs usually contain the 3D design drawings, costs, schedules, material characteristics, electrical, pluming and Heating Ventilation Air Conditioning (HVAC) designs as well an many other specifications which are relevant to the construction and maintenance of the building. BIMs gained acceptance in the construction and engineering community since they have the potential to provide great cost reductions and productivity improvements. Producing BIMs at design phase is a pretty straight forward process, and with the digitization of the design files, it is becoming common practice, in some cases it is a requirement for the building commissioning. Some structures such as historical buildings do not have any BIM, so efforts are made to manually create those designs, also typically called as-built BIMs.

The aim of an as-built BIM is to reflect the state of a building in its as-built conditions, typically many years after the initial construction completed. This is a much more challenging problem since a lot of the required information is not visible, documents are mission or outdated, and most of the required information is not easily observable (for example pipes or electrical wiring). Due to its complexity and rigor, creating an as-built BIMs is a very demanding manual endeavor. With the increase in technology, particularly Light Detection And Ranging (LIDAR) systems, a lot of effort is put into automating the process of creating as-built BIM.

In [31] the authors divide the process of creating an as-built BIM into two major steps: data collection, and data modeling. In the first step enough data to properly

define the building is collected, typically with a hand-held Red Green Blue Distance (RGBD) camera or LIDAR. Laser scanners produce a set of points which are distributed in the 3D space, so each point will have associated with it a location and in the case of RGBD, a color as well. Each scan produces millions of points, distributed around the room where the scan was made. For the rest of the thesis, those points will be referred to as a scan. Although the scans are arranged in the shape of the room, they do not provide any meaningful semantic information by themselves. Typically this information needs to be extracted from the scans, with the use of heuristics that can describe the area in terms of objects which are meaningful to the AEC industry, such as locations of walls, doors, windows, and furniture.

In the second step, the raw scans are modeled into a semantically rich BIM model. Current tools require human intervention, making them expensive and are often prone to error. Automatic methods for construction of a BIM from a scan is researched intensively with varying degrees of success regarding accuracy, efficiency and autonomy. There are several challenges in the extraction of the semantics from scans. Some of the challenges arise due to the different shapes rooms can have, artifacts that can be considered clutter and should be ignored, the complex and unorthodox geometries of some buildings, and the potential of unobserved areas. All those issues are compounded by the large file sizes collected by the cameras, which topically consist of millions of points. The semantics of interested when looking at a building are the definition of the object, such as the dimensions, relationships and properties [7].

As indicated in [31], there are no methods which provide a meaningful BIM and are also fully automatic. Most automatic approaches have limited ability, and are only able to produce a small portion of what would be required to create a BIM up to the accepted standard. Most research is at the stage where planar surfaces and their relationships can be determined. Some approaches can determine volumetric objects as well as their

relationships but this is limited, mostly because a comprehensive data collection of the building is not possible, leading to unobserved areas, or unknown properties of objects. The second dimension is the level of details available in the BIM, this is limited by the abilities of the scanning devices, and the computational complexity from the increase in the collected data set.

To address the issues of complexity due to the size of the data, this thesis focuses on a method of selecting a subset of scans with minimal impair to the final output.

1.2 Light Detection and Ranging (LIDAR)

LIDARs are an active device used to measure distance where light pulses are emitted and the reflected pulses are measured. The difference in return time and wavelength can be used to infer the distance traveled by the ray with high accuracy. By taking multiple measurements a three-dimensional representation of the scene can be made. Due to the high accuracy, LIDARs are often used to make high-resolution geographical maps, typically from airborne platforms. There is an increased demand in cheaper LIDAR technology, due to the recent advances in self driving cars and autonomous robots which heavily rely on on it to navigate. The availability of affordable LIDARS is opening up the possibility for small and medium companies to use this technology as part of their construction process as well. The resolution and accuracy of LIDAR systems range, typically depending on the price. The range of vision is typically anywhere from 1m all the way to 120m, and the accuracy is typically within centimeters, depending on the distance of the measurement. The number of measurement points in each scan also varies, they can range from a single measurement all the way to millions of points.

1.3 Challenges and Contributions

Although automatic floor plan creation is becoming more feasible due to the price of LIDARS dropping, it is far from being a fully automatic system. A fully automatic system still faces many challenges when tested outside of the lab. The first challenge when implementing an automatic mapping system is the data collection. In our experiments we are using a rover to collect data,, but every minute of data collected requires about 1GB of storage. For a large building, which might take hours to collect data everywhere, the approach requiring to store all data and analyzing it becomes unfeasible. We propose a method of scan selection which is able to select a subset of those scans, without a major hindrance to the final map produced.

The second challenge is the presence of clutter and furniture in a building. Most algorithms presented in literature typically assume straight walls, which will usually fail if there are any ornaments near the wall, or if a curvature is present. While it is unlikely to find an algorithm which will be able to handle all cases, and be robust to clutter, in Section 3.4 we present an addition to the popular RANSAC algorithm which addresses some of those issues.

The contribution of this thesis is a method of laser scan selection for the purpose of floor plan creation and a method for wall detection from noisy laser scans. The proposed algorithms have been tested on synthetics and real world data. The results are analyzed based on multiple metrics depending on the source of the data set.

By informatively selecting a subset of the scans, our experiments show a 10 times reduction in the number of scans required for the walls extraction algorithm, in comparison to using the whole data set.

The rest of the thesis is organized as follows, in Chapter 2, we investigate and summarize related work, in Chapter 3 we provide a high level overview of the suggested solution. In Chapter 4 we provide details about the implementation, including the software and hardware used. In Chapter 5 we introduce the experiments we conducted as well as the results observed, and finally in Chapter 6 we summarize the work and discuss future work.

Chapter 2

Related work

Building information models are becoming an accepted standard in the construction industry. Building modeling has been attracting attention since computer aided design became available. Research and industrial application have been limited due to the high cost of Light Detection And Ranging (LIDAR) and accurate cameras. With the advent of self driving cars and advances in robotics, cameras are becoming more readily available, which leads to many different solutions for floor plan extraction to be available, but none are close to being fully automatic. Automatic Building Information Model (BIM) creation is typically broken down into the following steps: Data collection, data reprocessing, and BIM modeling. Comprehensive surveys of the state of automatic BIM creation are the following [31, 34, 18].

Research can be classified by the complexity of objects being identified, the relationships between them, and the level of details extracted from the data. Further approaches are segmented by the technology used to collect the data. Typically laser scanning is required to collect enough data to make the solution practical, tape measures or even Red Green Blue (RGB) cameras tend to result in maps too coarse to create a usable result. Laser scanners collect millions of points with millimeter to centimeter accuracy, making them the ideal candidate for this problem. Computational challenges often arise

from the size of the data set, making it important to pick a scan location which properly represents the room shape.

In order to improve computation time, most algorithms use local heuristics, by breaking the scan up into smaller parts, and analyzing them individually. Typically the scans are segmented and then different parametric shapes are fit to the scan data.

2.1 Floor plan extraction

Detecting the floor plan is one of the first steps in creating a building model. Detecting the walls is typically the first step. There are many approaches but most of them take advantage of some heuristics which can separate walls from other objects. The authors in [10] take advantage of the orthogonality between the floor and walls. The authors in [38] find all the planes present in the pointcloud scan, and create a coarse description by looking at the planes in the principal directions. The detection of planar surfaces is not enough to make higher level inferences, such as the volumes and area of specific rooms, to make those inferences, typically the relationship between the planar surfaces is required. In [37] the relationship between planes was described using a graph, higher order configurations, such as rooms, could be recognized by performing sub graph matching. In certain cases, the floor plan is composed of straight walls as well as curved ones, [39] in those cases planar surfaces and quadratics are detected then classified. Other planar extraction algorithms include [20] where an energy minimization approach is used to select a small number of models to describe the whole data set, [26] takes advantage of surface normal and merging similar planes based on orientation and distance.

2.2 Relationship between objects

Finding the relationship between objects in a building is an important step in the creation of a BIM model, having relationship between different objects is critical when modeling pipes, valves or electrical circuits. Spacial relationships are also important for maintenance in order to specify the room where certain objects could be found, and helping with navigation to certain rooms. Spatial relationships can also provide contextual information and assist in object recognition, as shown in [28]. As described in [34] there are three relevant categories of spatial relationships, namely: aggregation, topological and directional relationships. Aggregation relationships identify if different objects are part of each other, such as two different walls being part of the same room, those relationships are typically modeled using a tree based representation [12]. Topological relationships are used to connect different parts of the building together, such as the joint between planes, or keeping track if a certain objects is inside or outside a room. Directional relationships indicate how objects are related to each other spatially, for example room 1 is to the right of room 2. An automatic approach for topological information deduction is presented in [27]. The approach presented works on a 3D CAD model, this approach would not be directly applicable to BIM inference from laser scans since a 3D model would be the end goal of a scan-to-BIM algorithm. In [25] the component geometry and relationships between them is used to identify features, and allow for queries to be performed.

2.3 Clutter segmentation

Often the building is cluttered with objects which are not considered to be part of the building structure itself. The clutter needs to be identified, and removed from the data set. The occlusions created by the clutter will leave areas behind them where no

measurements can be made of. Depending on the application, sometimes those gaps need to be filled. One method presented in [6, 40] is to identify another region that matches the occluded region and substitute data. In cases where many scans are taken, it can be assumed the missing data from one scan will be sampled by another scan [13]. The authors of [2] explicitly model occlusions and label it accordingly, in their experiments on average only 50% of the wall area is scanned.

2.4 Room segmentation

Typically the end goal of the process is to be able to derive higher level information from the data. Such information is typically related to the labeling of the different rooms, such as identifying the living room, kitchen, or bathroom. In order to make those semantically meaningful spaces, first the building needs to be segmented into rooms.

The method presented by [29] is making full use of the location of where the camera was placed when the scan was taken. The walls can have a specific room label, which corresponds to the scan which identified the wall. Any walls which do not separate two different room labels can be considered an artifact due to noise and be removed. This approach is not very computationally efficient since an energy minimization problem needs to be solved, but the computation can be paralleled easily. The approach proposed by [3] suggests identifying the rooms is a detection problem rather then a segmentation one. This approach uses methods popular in the signal processing field, a density histogram is created of the the empty space inside a room and with the walls. This density histogram signal is convoluted with a bank of peak-gap-peak pattern filters. This approach is computationally efficient and was demonstrated to work well in buildings with areas up to $6000m^2$.

Chapter 3

Solution Approach

3.1 System architecture

The system proposed by this thesis can be divided into several distinct steps. The first step is the acquisition of the scan pointcloud data, and its registration in the global coordinate system. Due to the size of the input data, an informative subset from all the pointclouds is selected. Using the subset of pointclouds, walls are extracted. In the case where a simulator is used to synthesize pointcloud data, walls can be extracted from the ground truth model which allows for an analytic estimation of the error in the algorithm. A more detailed description of the approach can be found in 3.1. The rest of this chapter will go into further detail about each stage.

3.2 Data collection, registration and pre-processing

Data collection is the method in which pointcloud data is collected from the building where a floor plan is to be constructed. The pointcloud is created by a Light Detection And Ranging (LIDAR), which works based on time of flight of emitted light.

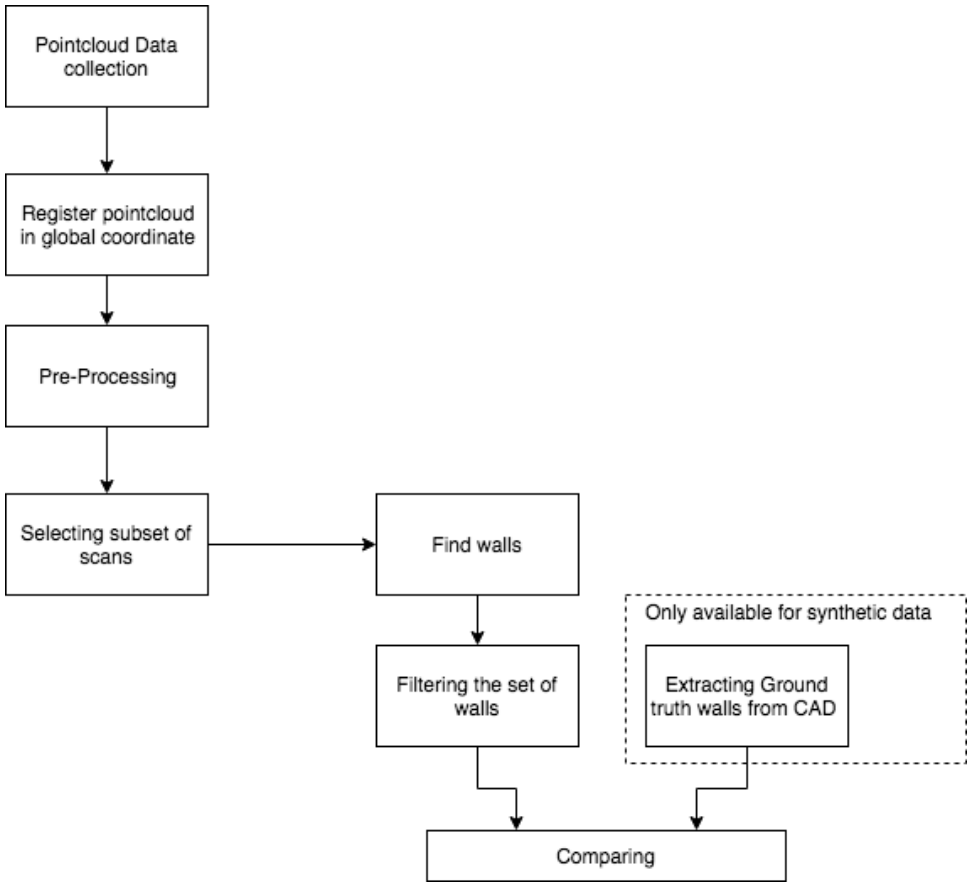


FIGURE 3.1: A high level overview of the system

3.2.1 Data Collection

Data is collected using a LIDAR mounted on a rover. Typically it is not necessary to have the LIDAR mounted on a robot, it can also be a human carrying the LIDAR around the desired building. The advantage of having it mounted on a rover is that all scans are taken from the same height above the ground. This makes it easier to fuse the multiple scans together as explained in Section 3.2.2.

3.2.2 Registration

The coordinates of the points are in reference to the LIDAR at the time the scan was taken. In most cases in order to cover a building multiple scans are needed to be merged together. The goal of the merging process (known as point cloud registration) is to find a transformation between the coordinate systems of all the scans, such that all the point coordinates are in reference to a single global origin point.

The goal is to find a set of translation and rotation values which will minimize the least square error measure between two sets of points. This process is usually formulated as an optimization problem. Typically known as Iterative Closest Point (ICP)[5].

A simple algorithm is a point-to-point match [1], this is where every single point between two scan is attempted to be matched, by applying the rigid transformation described by $T(\theta, x, y)$.

$$T(\theta, x, y) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & x \\ sin(\theta) & cos(\theta) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

Given a target pointcloud P and a source pointcloud Q, we wish to find a correspondence $K = (p, q)$ for every $p \in P$ and $q \in Q$, we wish to minimize:

$$C(T) = \sum_{p,q \in K} ||p - Tq||^2$$

The goal of the algorithm is to minimize $C(T)$ by altering T. By finding an optimal value of T, this transformation, would bring the source pointcloud Q in the global

coordinate frame of P.

In some circumstances, if more information about the distribution of the points in the pointcloud is known, such as knowing certain features are present, more efficient algorithms can be utilized. For example [8] planes can be extracted and used to find the transformation matrix.

3.3 Subset selection

A LIDAR will make million of measurements with precision up to 10 millimeters for every single scan taken of an indoor space. To perform any computationally intensive operations on this data is time consuming and inefficient if all the data is used.

Fortunately, a lot of the data produced by scanners is redundant, and can be discarded without loss of accuracy in the final analysis. Measuring the density of a pointcloud and trying to reduce it locally is typically a hard problem since there are no agreed upon measurements of pointcloud density [16]. Some techniques which act on the pointcloud include down sampling the pointcloud, by randomly discarding some of the points present, making the total data set smaller.

For most work [29, 24] regarding floor plan extraction, the pointcloud acquisition step and registration is assumed to be done by a professional and is not considered as part of their algorithm pipeline. Our goal is to create an automated approach where the human intervention is not required for the data collection, as this would allow for continuous monitoring of the building process as it is taking place. In our work we collect our data from a moving rover (see Section 4.2.1), and the scans are made at a rate of about 10Hz. This scanning technique leads to a large amount of data being collected, a lot of which is redundant.

We develop a method of scan selection which is meant to reduce the amount of overlap while still maintaining enough information to be able to extract the walls and floor plan.

3.3.1 Problem definition

The input to the problem is a set list of pairwise distinct scans $S = (s_0, ..., s_n)$. The desired output of the problem is $A \subseteq S$, of carnality k where $I(A)$ is maximized. The utility $I(A)$ is a measurement of the accuracy of the floor plan created when the scans in A are analyzed.

Currently there is no way to directly evaluate the function $I(\cdot)$. The utility of a scan would depend on many factors, including the algorithm which is used to extract the floor plan, the assumptions it makes, and any other heuristics and parameters it uses. In order to evaluate the utility function $I(\cdot)$ we use as a surrogate function the area covered by the scans. The intuition behind this approach is explained in subsection 3.3.2.

3.3.2 Solution details

Our solution to the subset selection problem is to use a greedy approach with a heuristic based on the area covered. We start with an empty set of scans $A = \emptyset$, and we greedily add scans until $|A| = k$. During each iteration the scan which increases the utility $I(A)$ the most is added to A.

The intuition behind the algorithm relies on the idea of area coverage. Since many scans cover the same area, this is a good metric for reducing the amount of overlap. Due to the distribution of the points in a scan along the walls, and the presence of outliers,

computing a single convex hull, and calculating the area of the hull will give a result which is overconfident in the area covered.

Our contribution includes a segmentation step for the scan, making the area metric a better estimation of the utility of the scan. The idea behind the subset selection is to be able to have the scans sorted by their informativeness and then select just a subset of those scans, and apply the floor plan extraction algorithm in decreasing order, starting with the most informative.

The overall algorithm design is based on calculating a convex hull for a scan, and then using those multiple hulls to cover as much of the area as possible. The convex hull area is extremely sensitive to outliers. For our application outliers are described by any points which cannot reliably be used for floor plan extraction. This can be the case where the density of the points in a certain area is too low to make any reliable measurements. Due to the way in which the LIDAR measurements propagate, outliers of this sort are inevitable. Outliers will cause the covered area to be overestimated, leading to potentially erroneous scan selections. We solve this problem by first clustering the points in each scan, computing the convex hull on each cluster, and then performing an

union of the hulls. The resultant hull is a much better approximation of the hull.

input : S, A list of scans

output: S^*, An ordered list of scans based on informativeness

$hulls = cluster\_and\_hull(S)$

$S^* = [argmax(hulls.area)]$; `// the hull with the maximum area is selected first`

$hulls^* = [hulls[argmax(hulls.area)]]$

$S = S/S^*$

$hulls = hulls/argmax(hulls.area)$

for $i \leftarrow 0$ **to** $|hulls|$ **do**
    $area = 0$
    for $j \leftarrow 0$ **to** $|hulls|$ **do**
        $potential\_area = hulls^* \cup hulls[j].area$
        if $potential\_area > area$ **then**
            $area = potential\_area$
            $index = j$
    end
    $S^* = S^* \cup S[index]$
    $hulls^* = hulls^* \cup hulls[index]$
end
return S^*

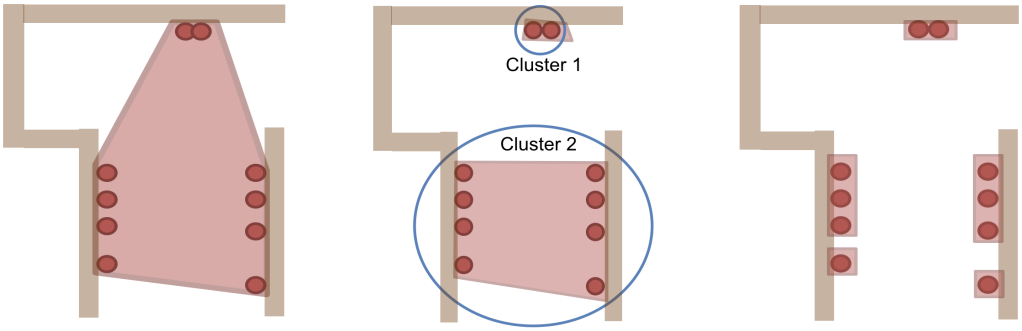
Algorithm 1: Hull selection

input : S, A list of scans

output: H, A list of lists of hull objects, each corresponding to a scan

foreach $s \in S$ **do**
    $C = KMeans(s)$
    foreach $c \in C$ **do**
        $hulls = hulls \cup convex\_hull(c)$
    end
    $H = H \cup hulls$
end
return H

Algorithm 2: cluster_and_hull()

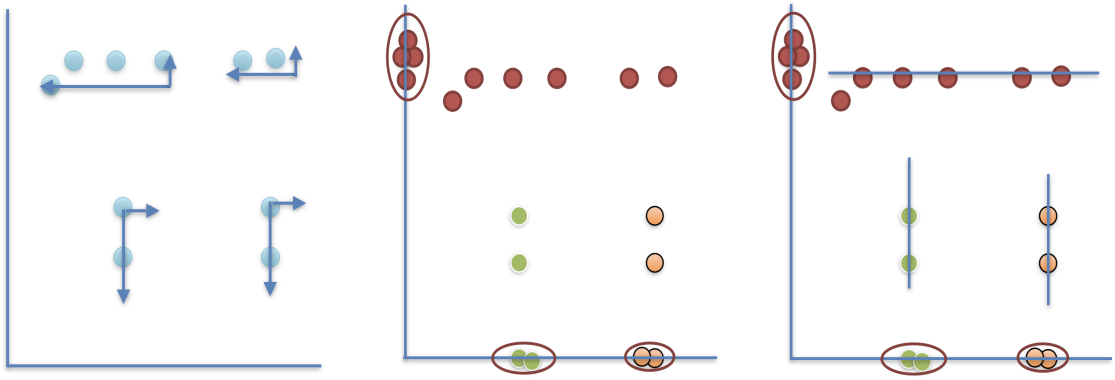


(A) Naive solution (B) Our solution (C) Ground truth

FIGURE 3.2: A graphical comparison between the naive solution, our solution and ground truth

Given the number of points in each scan, KMeans cannot be performed on all the points in a reasonable time. A small subset of those points is used for the clustering, and points around each cluster are selected, based on the minimum distance between all the cluster centers. This leads to reduction in the number of outliers, and a it is computationally efficient.

The advantage of using the proposed clustering hull algorithm is exemplified in Figures 3.2a and Figure 3.2b. In this example a convex hull is being placed around a set of points which are arranged in a similar distribution that would be characteristic of a LIDAR scan in an indoor space. By fitting a convex hull on all the points (Fig. 3.2a), this leads to an overestimation of area covered, since no measurements made in the center of the image. By increasing the number of clusters in which the scan is split, in this example to 2 clusters, a better estimation of the covered area is obtained (Fig. 3.2b). This intuition has been validated by experiments as explained in Section 5.



(A) Step 1 (B) Step 2 (C) Step 3

FIGURE 3.3: The main steps in our proposed wall finding algorithm

3.4 RANSAC clustering

The run time of RANSAC will be variable depending on the number of points in the cloud and how many are required to represent a line. This behaviour leads to sporadic run times and multiple walls will be found for the same plane in the pointcloud. For this reason we came up with a method of first segmenting the scans into multiple clusters each containing the points for roughly one wall. For a flat wall, the surface normal of all the points on a wall will be pointing in the same direction. We took advantage of this, and use the vector of the surface normal to project all the points along it. When the points are projected along the normal, the points from each wall will cluster roughly in the same location. Using Mean Shift [14], the number projected pointcloud is segmented into multiple clusters, each one corresponding to a single wall. On each cluster a single run on RANSAC algorithm is ran, finding a single model describing the wall.

A clustering step is shown in Figure **??**, where the points for two perpendicular walls are being projected along the dimension corresponding to the biggest surface normal, to another axis. The projected points can now be clustered (shown by two red ovals). The points in each cluster correspond to a wall. To get the equation coefficients describing the

value of the wall, a single run of RANSAC can be executed on the points corresponding to each cluster.

3.5 Scan registration

The points measured by the LIDAR mounted on the rover are in the coordinate frame of the LIDAR itself. In order for a floor plan to be made, the multiple scans need to be in a global coordinate frame among all the scans.

Open source libraries have been used to manipulate the point clouds, and apply the transformation to merge them. Open3D [41] has been used to perform the point cloud registration. There as several methods of performing the registration, if there are only two point clouds, and there is a lot of overlap between the clouds, a point-to-point ICP can be performed. This is fairly fast, as the transformation is only performed on a single cloud. If there are multiple pointclouds that are required to be merged, a multiway algorithm can be used [9]. The multiway registration will keep a single point cloud as a reference point, and the rest of the pointclouds will all get some transformation applied to them, in order to minimized the error across the set of all input scans.

An area of about $350m^2$ composed of four long hallways, took the rover about 5 minutes to collect the laser scans. The sampling rate is about $10Hz$ leading to about 5000 laser scans. The travel speed of the rover is dynamic depending on the amount of free space, but on average is about $0.3m/s$. The pose of the rover is estimated by the Adaptive Monte Carlo Localization (AMCL) at a rate of $10Hz$ and this data is stored as well. Each laser scan and AMCL estimates pose are tagged with the time when the measurement was taken. Typically the two measurements are not synchronized.

Optimizing multiple scans to fit together is typically a very time consuming operation. Instead because the pose approximation from the AMCL is fairly accurate, we sequentially align scans based on that pose estimate. Due to the large sampling rate, in comparison with the speed of the rover, a large amount of overlap is expected between sequential scans. Due to the redundancy of information in the scans, and if registration is not successful, some scans are dropped. `Fitness` is a measure of the overlapping area, in our experiments the scans were registered, and if a scan did not have a mean `fitness` higher than `0.99` with the previous `5` scans, that scan was rejected, indicating the optimization was not successful.

The registration procedure can be summarized by the following steps:

1. Linearly interpolate the poses reported by AMCL at each laser scan time stamp
2. Retain only every 5^{th} scan
3. Apply the pose transformation from AMCL to the scans
4. If the mean `fitness` between the frame, and the previous `5` is less then `0.96`, then drop the scan
5. Sequentially register every two pointclouds
6. If the mean `fitness` between the frame, and the previous `5` is less then `0.99`, then drop the scan

Algorithm 3: Registration summary

During one of our experiments, after running Algorithm 3, only about 2% of the scans were retained. This still left about `115` scans to be analyzed and a floor plan to be extracted.

Chapter 4

Implementation

The system is composed on multiple systems working together in order to collect, analyze it and determine the usefulness of the proposed algorithm. Simulation has been used in order to be able to obtain accurate ground truth measurements of the building, a rover equipped with a Light Detection And Ranging (LIDAR) has also been used in order to collect data from a real world environment.

4.1 Simulation

The first step in the collection of the LIDAR simulation data is to create a 3-Dimensional (3D) Computer Aided Design (CAD) model of an indoor location. The model is of a resolution and size similar to what would be expected in a large building in real world , windows and doors are represented by empty space in the model. During simulation, the LIDAR rays will pass through the empty spaces, the real world experiments show some of LIDAR rays will be reflected back when they hit glass, leading to some discrepancy between simulation and real world when analyzing areas near windows.

The 3D CAD model of the building is made in Autodesk Inventor [4]. After the 3D model is created, a `.stl` file which is then imported into BlenSor [17]. BlenSor is a

simulation software used to create realistic simulations of LIDAR and Kinect sensors, for the purpose of offline simulation. BlenSor is able to simulate measurement errors in the LIDAR as well, the values chosen were consistent with what would be expected for a modern LIDAR, Gaussian noise with 1cm standard deviation. We take advantage of the ability to automatically register the multiple pointclouds in the same coordinate frame, this allows us to forgo the registration step for synthetic data.

Multiple scanning locations are selected in the model, and each scan is saved as a `.pcd` file, along with the location where the scan was taken.

4.2 Data collection

There are many ways to collect the training data. Typically the requirement is to have an adequate coverage of the building. In previous work measurements were made by a human operator. Our goal is to automate the processing pipeline for floor plan creation. In order to do so, we collect the data with a rover platform. In the following sections the software and hardware sections of the platform will be described.

4.2.1 Rover Platform Hardware

The rover is equipped with a LIDAR to perform the localization and mapping. An off-the-shelf laptop is installed on top of the rover where the algorithms are run, and where the sensors are connected to.

The robotic platform is an implementation based on the kit sold by Parallax inc. , the kit is named Arlo Complete Robot System. The assembled rover is about 10 inches tall and 18 inches in diameter. The robot is capable of carrying up to 20 pounds and can travel up to 0.75 m/s. The rover is equipped with two 12VDC lead acid batteries which

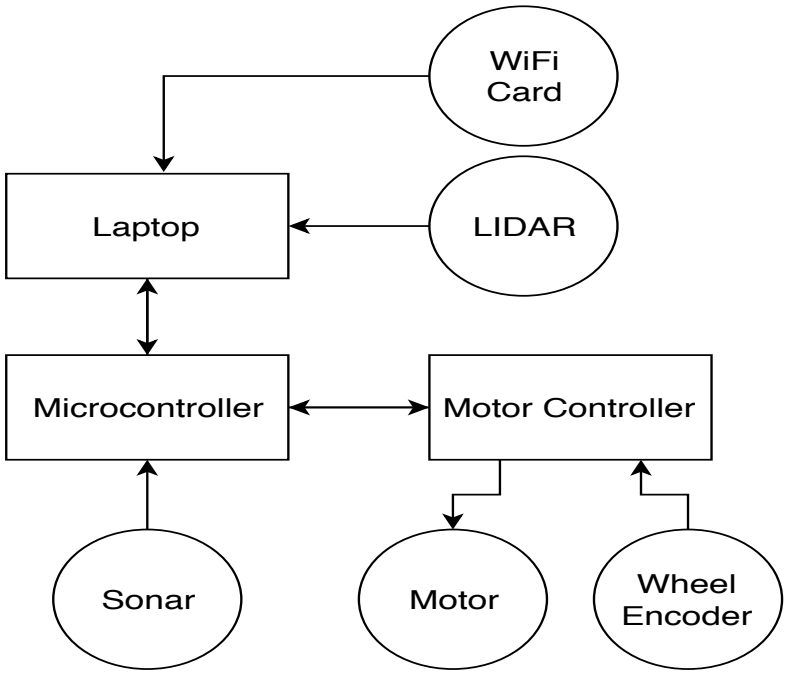


FIGURE 4.1: Rover system architecture.

are used to power two DC motors each equipped with a 36-position disk quadrature encoders resolving to 144 discrete ticks per tire revolution. The rover is equipped with 6 sonars as a safety mechanism in case an obstacle jumps in front of the rover or a faulty navigation command is sent from the navigation algorithm.

The robotic platform is designed in a modular fashion where tasks are segregated on different computing devices based on the computation required and the response time required.

The main components on the system are the Microcontroller (MC) which is connected with the sonar sensors, and motor controller. The laptop is used for it's large computation power to run the Simultaneous Localization and Mapping (SLAM) algorithm and the navigation. A diagram of the system overview can be seen in Figure 4.1.

4.2.2 Rover Platform Software

The MC has been programmed in a proprietary version of C. The laptop is communicating with the MC over Universal Serial Bus (USB), sending the current pose and the desired pose to the MC. The MC is responsible of navigating between the two poses and communicating back to the laptop when it has reached the pose.

The laptop is running Robot Operating System (ROS), an open source software used to manage and integrate different packages in a unified robot. There are two main components for which ROS is reponsible for, the **SLAM** and **navigation**. The SLAM algorithm is using the a LIDAR to create a map in conjecture with the wheel encoder readings. The navigation algorithm used is part of the move_base. The navigation algorithm will output a viable path in a map if the location of the robot is known, and the final destination is indicated.

Sometimes the location of the robot is not known, but a map is available from a previous mapping run. In cases like this, it is valuable to take advantage of the already existing map. In order to do this, just the localization needs to be perform, and the mapping is not necessary anymore. ROS provides a localization library Adaptive Monte Carlo Localization which is a particle filter used to track the position of the robot given an already fabricated map.

During our experiments an initial run of the rover is performed in the desired area, and a map is created. The system is afterwards run in the localization mode, where Adaptive Monte Carlo Localization is used to keep track of the rover. During the second scan, the raw laser scan measurements are recorded in a `.bag` file. The location, as indicated by Adaptive Monte Carlo Localization (AMCL) is recorded as well. Those locations are used as an initial estimation when registering the pointclouds, as described in Section 3.5.

4.3 Software

For the implementation of the algorithms presented we used the Python 3 programming language (Python Software Foundation, https://www.python.org/). To aid with the manipulation of the pointclouds and to extract lines using Random Sample Consensus (RANSAC), the PCL(Point cloud library) library was used [33], along with the python binding python-PCL (http://strawlab.github.io/python-pcl/). Mathematical manipulation on is done using the help of Numpy[30], Pandas[23], and scipy[21]. In order to open the open `.pcd` files, at times it was convenient to use the functions present in pypcd library (https://pypi.org/project/pypcd/).

In order to open the `.stl` files and extract the ground truth locations of the walls, `numpy-stl` was a useful library (https://pypi.org/project/numpy-stl/). The computation of the convex hull and KMeans was performed using open source libraries [15, 32].

Development was performed in the Jupyter notebooks [22] and graphics and graphs have been rendered using Matplotlib [19].

All computations have been run on a 2017 MacBook Pro with a 3.1GHz Intel Core i5 processor and 16GB of memory.

Chapter 5

Evaluation

During our experiments we evaluate the benefit of selecting a subset of the scans in a certain order as opposed to selecting them randomly or in sequential order as they have been collected. We believe the data provided by the scans have a submodular property, and being able to select the most informative subset will provide a significant computation speedup with minimal loss in the final output.

There is currently no well accepted method of measuring the accuracy of the walls extracted, yet we require a metric in order to make any suggestions when the diminishing returns of the scan selection become too great, and it becomes redundant to analyze another scan.

5.1 Metrics

Even when the ground truth synthesized data from simulation is available, calculating the error between the inferred walls and the ground truth is still a challenging problem. Some of those challenges and the solutions we proposed will be presented in this section.

An ideal metric for the measurement of error between the inferred walls and the ground truth would be to match the inferred wall with the walls from the ground truth, and then measure by how much they deviate. In an ideal scenario where the output of the algorithm would provide exactly the same number of walls as the ground truth, and the inferred walls were roughly in the same location, then that would be a feasible metric. Unfortunately the walls are for each scan individually, leading to the case where multiple walls are inferred for the same wall. While this would not cause a problem for the creation of a floor plan, it would cause problems when trying to compute a metric with those walls.

A potential metric would be to use an optimization technique and match the inferred wall candidates to the ground truth ones. Given the noisy nature of the wall extraction algorithm, special care would need to be taken since the optimal of such an algorithm could provide an error which is smaller than the actual error.

Those errors would get emphasized especially when multiple walls are inferred for the same ground truth wall. For this reason we have settled for transforming the walls into a bitmap, and use the bitmap to calculate *Correct match*, *False positive* and *False negative*.

Our goal is to determine a subset of all the scans to use instead of performing computations on the whole data set. The metrics are general enough that any other wall extraction algorithm could be put in place, and as long as the complexity of the algorithm is dependent on the number of scans, the results would still be relevant.

Simulated data is created by making a Computer Aided Design (CAD) model of an indoor floor plan, and then the area is sampled in the same way a Light Detection And Ranging (LIDAR) would behave in that environment. For the simulated data we have the ability to extract the ground truth locations of the walls. This allows us to

analytically measure of how well the algorithm is performing.

Due to the way the data is collected, sometimes multiple walls are present for the same location, leading to an overlap, and also some walls although they are present in the CAD model of the building in simulation, they are never visible from the point of view of the LIDAR. This leads to a situation where there is no one-to-one match between the ground truth walls and the inferred ones. We resort to a binary comparison between the inferred walls and the ground truth walls.

A map of equal size and resolution is created for the inferred and ground truth walls. The map is binary, so multiple walls in the same location will not be counted as multiple errors. If there is one or more wall present, a value of `1` is placed that that location, and if no walls are present, a `0` is placed in the bitmap.

Once the ground truth B_{GT} and inferred B_I maps are created, they can be used to calculate an error map as follows:

$$B_E = B_I - B_{GT}$$

The error map B_E is an matrix with the values $[-1, 0, 1]$. The following are calculated:

$$FalseNegative = \frac{1}{|B_E|} \sum_{i=1}^{|B_E|} [B_{E_i} = -1]$$

$$CorrectMatch = \frac{1}{|B_E|} \sum_{i=1}^{|B_E|} [B_{E_i} = 0]$$

$$FalsePositive = \frac{1}{|B_E|} \sum_{i=1}^{|B_E|} [B_{E_i} = 1]$$

The number of occurrences of each value in the error map is an indication of the

performance of the wall extraction algorithm. During our experiments we use the metrics to compare different sequences of choosing the scans, so the size of the bitmap is consistent. The number of occurrences is normalized by the total area of the map.

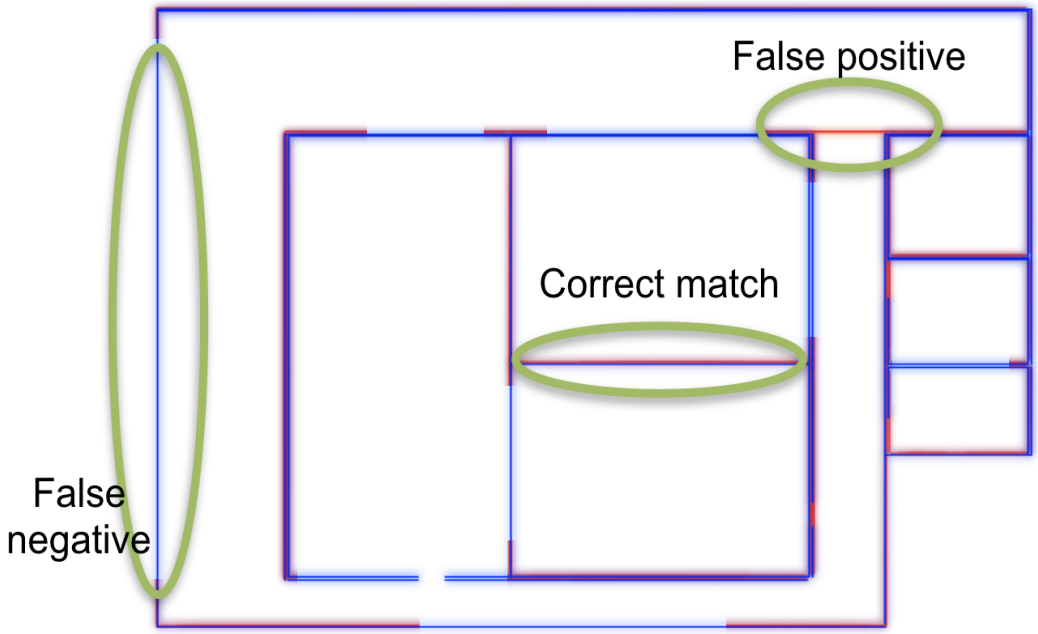


FIGURE 5.1: Overlay of the inferred bitmap and the ground truth bitmap

In Figure 5.1 the inferred wall bitmap (in red) is overlaid on a bitmap of the ground truth wall (in blue). Walls which are present in the ground truth bitmap, but are not extracted by the algorithm (false negative) are shown in blue. Walls which are present in both bitmaps, corresponds to correct matches, and in order to be visible, they are shown in purple in the figure. Artifacts which are extracted by the algorithm but they are not present in the ground truth model (false positive), are shown in red. In order to calculate the metrics for an algorithm the number of bits with each value is summed up and divided by the total area of the bitmap.

5.2 Synthetic data

Working with data generated using a simulation has the benefit of having a model where the exact location of the walls can be extracted from. As explained in section 4, the simulation model is a 3-Dimensional (3D) CAD model. Although in the model itself there is no strict definition of what a wall is, the model can be designed in such a way that taking a vertical slice of that model will be the equivalent of getting a floor plan.

By having the ground truth floor plan, we can compute *Correct match*, *False positive* and *False negative*, see section 5.1 for more information. This information allows us to make an informed decision when processing and extracting data from more scans provides insufficient incentive.

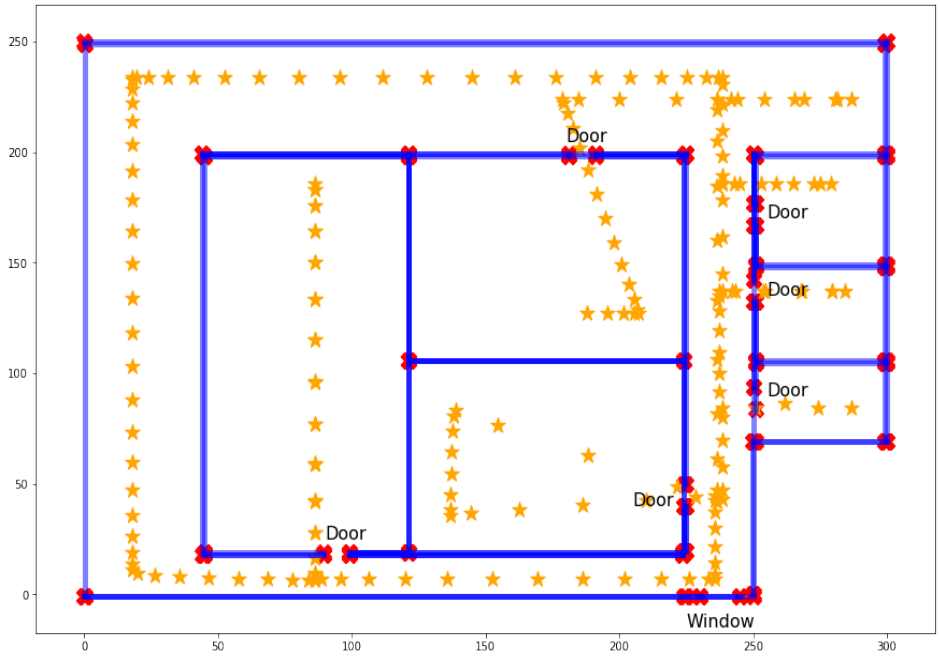


FIGURE 5.2: Ground Truth floor plan with the location of the scans indicated by yellow markers. The endpoints of planes, such as doors and windows are shown by red markers

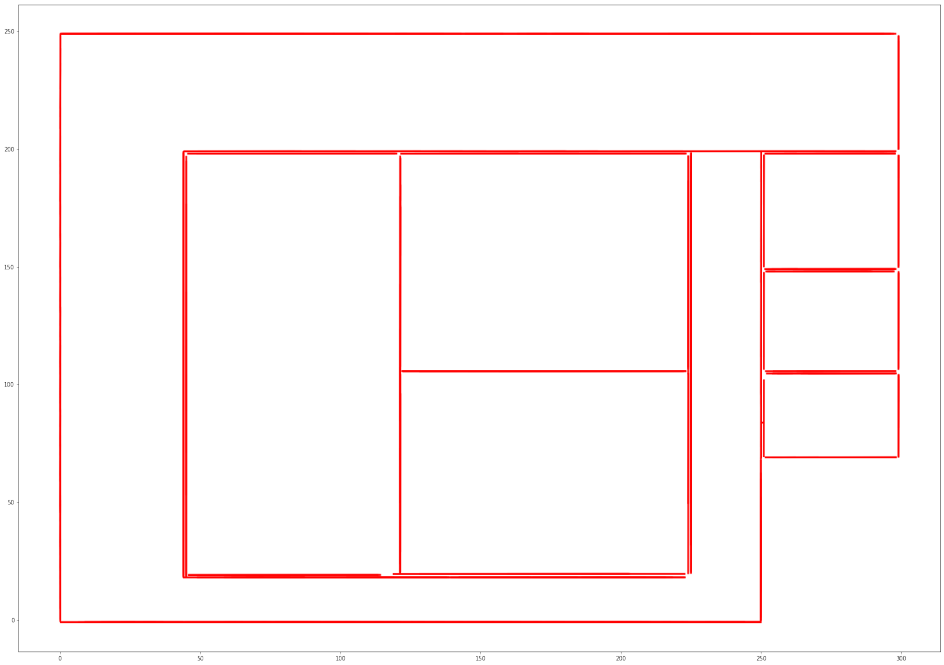


FIGURE 5.3: Walls extracted from the first 60 scans as selected by the subset selection algorithm

The ground truth floor plan of the synthetic data can be seen in Figure 5.3. This floor plan is supposed to be similar in shape to the real data set. Since there was limited accessibility when collecting the data in the real data set, no scans were taken inside the rooms. The synthetic model was sampled extensively inside and outside the rooms. The path along which it was sampled is supposed to imitate the path traveled by a rover with a LIDAR. The sampling rate for the synthetic data is slightly lower than the sampling rate of the real world data. About 150 scans have been collected over this area. In comparison for a similar area in the rover collected about 4000 scans. As described in the registration Algorithm 3, a large portion of the scans will be drooped due during the scan registration phase and the pre-processing. The number of the scans left over is about 130. The scans from the simulation are already registered in a global frame, making the comparison roughly equivalent.

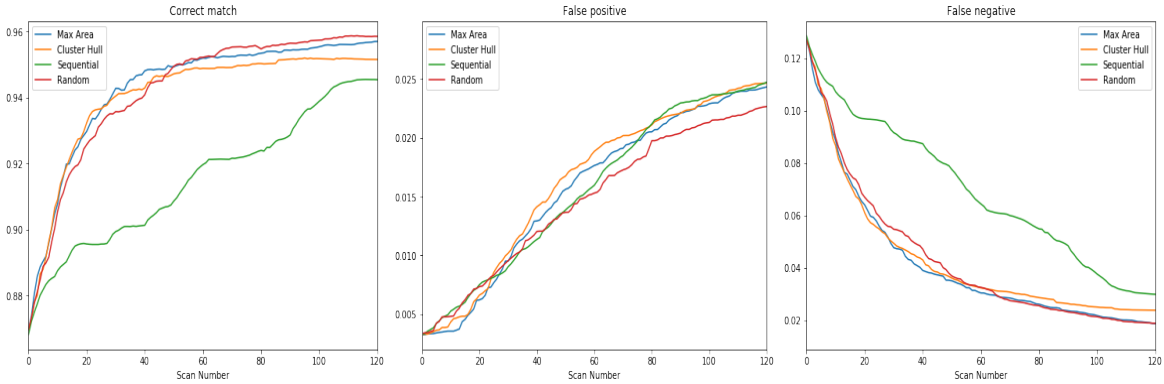


FIGURE 5.4: Correct match, False positive and False negative

Four methods of subset selection have been compared on the synthetic data set. Choosing scans based on the increase in total area covered using the convex hull, based on the area covered using the clustering hull method introduced in Section 3.3, randomly picking scans, and sequentially as they have been collected. As it can be seen in Figure 5.4, the sequential method is performing the poorest. Where the other three perform more similarly.

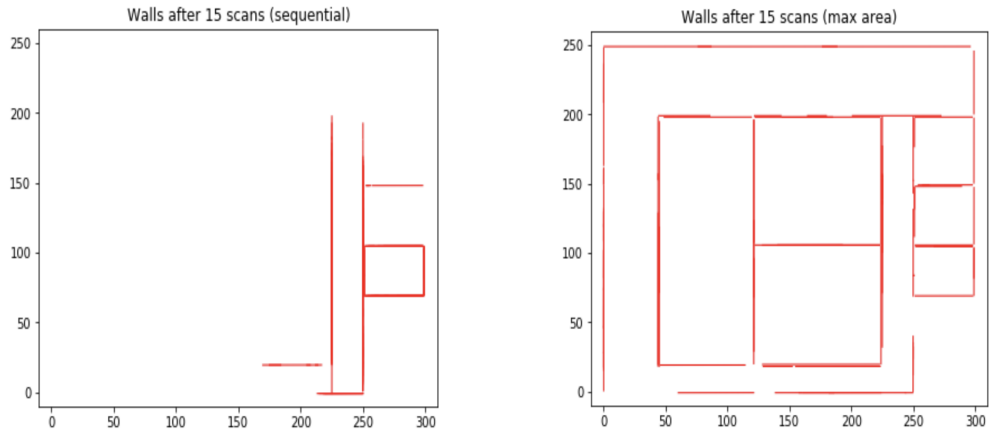


FIGURE 5.5: Floor plan extracted with the first 15 scans if analyzed in the order they were collected, and the order suggested by our algorithm

Figure 5.5 provides a visual comparison between two different orders in which the scans can be analyzed. In the first figure the scans are analyzed in the order the scans

were made. In the second graph the scans are analyzed in the order suggested by our algorithm. As it can be observed by using just a small number of the total scans (15 out of 273) an almost complete floor plan can be extracted. This ability to use just a subset of the scans allows for scalability of the algorithm to larger buildings.

The random scan selection is uniformly choosing scans from all the possible ones. This approach is not taking in consideration the location where the scans were taken. During out simulation all the scans were taken at different locations, in a real world data collection, there are often areas which are sampled more often. This is usually the case when the rover has to go around obstacles or pass though narrow doorways (as highlighted in Figure 5.9). Depending on the severity of the phenomena, this would harm the performance of the random selection, but it would not affect the performance of the area based algorithms.

The following graph shows the area covered by convex hull area, and the cluster area.

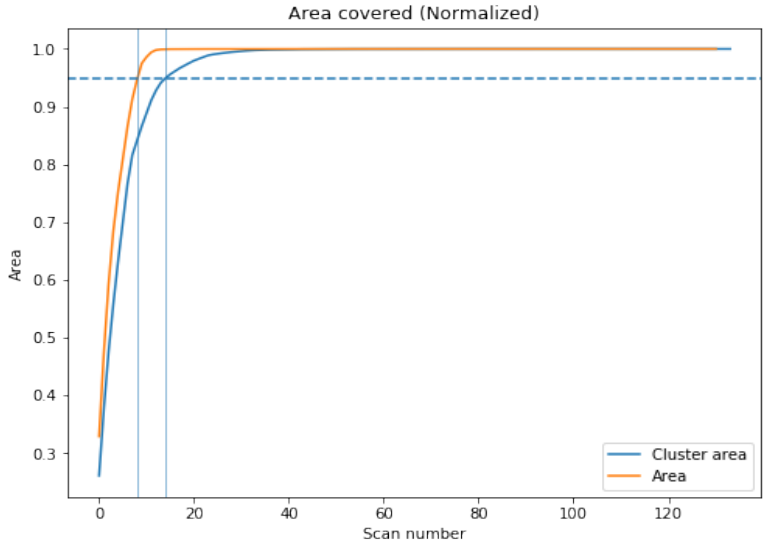


FIGURE 5.6: Convergence of area covered. 99% area covered indicated in the plot.

As it would be expected, by using the convex hull as an area metric will be an

overestimation of the actual usable scan area. By first clustering small parts of the scan, and then using the union of the smaller convex hulls, it will give a more accurate description of the actual area. Quick convergence in this case is not a desirable trait for the algorithm. Once the algorithm believes it covers roughly the whole area, the scan sequence after that point will not be very informative, as each new scan provides very little increase in the area covered.

A slower convergence allows the algorithm to make informed decisions for a larger number of scans. The accuracy of the area estimation in the Cluster area algorithm can be adjusted by changing the number of clusters in KMean function in Algorithm 1. Depending on the area, and the equipment used to make the measurements different number of clusters would be used. As shown in Figure 5.7 using the Area metric will reach 99% area covered after about 10 scans, and the Cluster area metric after about 17 scans.

As it can be see in Figure 5.7 as the number of clusters is increased, the value of the area covered is lower even as the number of scans is increase.

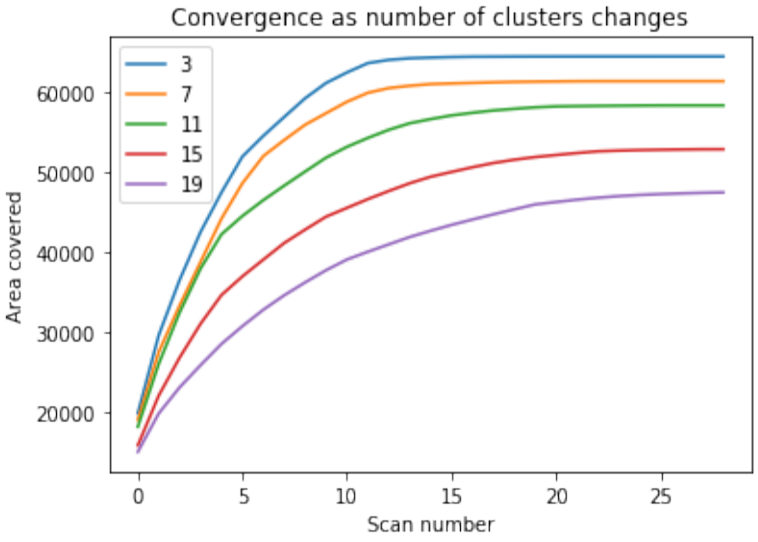


FIGURE 5.7: Relationship between area cover and number of clusters

As the number of clusters is increased, the computation time also increases. Depending on the algorithm used to extract walls, the optimal number of clusters could be identified to minimize the total computation time. Some proposed wall extraction algorithms proposed in literature are very computationally intensive, making it beneficial to start with as few scans as possible.

5.3 Wall extraction

Extracting walls from a pointcloud is typically a problem of detecting and segmenting multiple planes. Algorithms for plane extraction required robustness to outliers. One of the most popular algorithms is Random Sample Consensus (RANSAC), introduced in [11]. RANSAC uses a sampling technique where candidate solutions are generated using the least number of samples. The number of samples is slowly increased while updating the candidate solutions. The RANSAC algorithm can be summarized as:

1. Select a subset of the samples, considered a hypothetical inlier.
2. Generate candidate model parameters for the hypothetical inlier set.
3. Test the hypothetical outliers against the proposed models. Outliers which are well predicted by the model, are considered part of the consensus set.
4. If the consensus set is above a certain size, the model is considered adequate, otherwise step 1-3 is repeated until the consensus set it large enough

Algorithm 4: RANSAC

Several extensions have been made to the RANSAC algorithm including sampling based on maximum likelihood [36] and importance sampling [35] among many others.

Each time a line extraction algorithm is run on a point cloud, a set of inliers and the parameters of a single line is returned. This proved to be an issue in the case where floor plans are to be extracted, because each scan contains multiple planes. A typical approach is to remove the set of inliers from the whole point cloud, and run the algorithm

again, until the number of points left is lower than the threshold required for another line to be found. Based on our experiments using this approach we found multiple wall candidates for each wall in the point cloud scan.

5.4 Real data

During the set of experiments utilizing real data, we are not able to come up with a ground truth model of the walls to compare our inferred walls with, as we have with the synthetic version of the experiments.

Step	Time	# scans at end of step	Data size
Data collection	6min	6070	4.83 GB
Selecting continuous scans	6 min 29s	130	11.5 MB
Registration optimization	40s	90	6.9 MB
Hull creation	10s	90	6.9 MB
Cluster Subset selection	4.5s	15	1.6 MB

TABLE 5.1: Computation time required for each step

For our real data test bed we are using a rover to collect the data on a portion of the second floor of Information Technology Building (ITB) at McMaster University. The time for each of the steps of the process can be found in Table 5.1. The computation time required to process the 'Cluster Subset selection' can be quite lengthy due to the polynomial nature of the hull union algorithm. As it can be observed in Figure 5.7, depending on the number of clusters, the area stops increasing, as more scans are added. This allows us to stop the computation after a desired amount of scans have been selected. For this experiment we chose `10` clusters and chose a subset of 15 scans. The

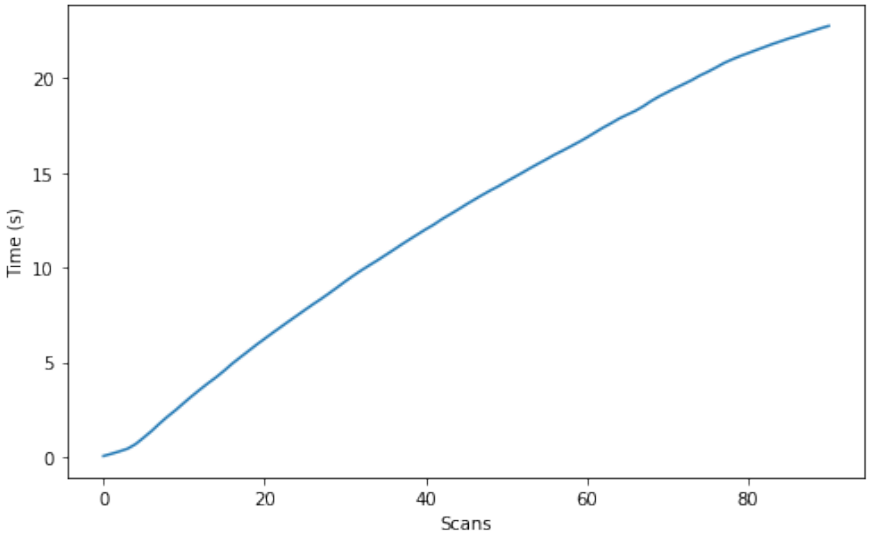


FIGURE 5.8: Time required to compute the subset selection for different number of scans.

computational time required as the number of scans increases can observed in Figure 5.8.

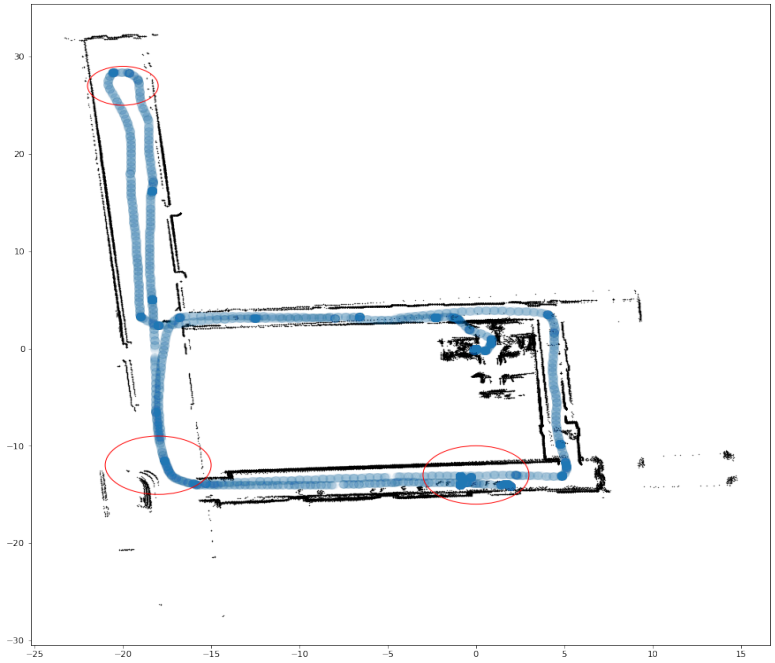


FIGURE 5.9: Registered scans along with the location of each scan. Darker blue indicates more scans at that location. Areas with many scans are circled.

We are also comparing two different methods to extract the walls from the scans. We are looking at comparing an implementation of RANSAC as described in Algorithm 4, with the clustering method described in Section 3.4. In Figure **??** we can see the difference in computation time for different scans. In Figure **??** the cumulative number of walls extracted, and in Figure **??** the number of walls extracted is shown.

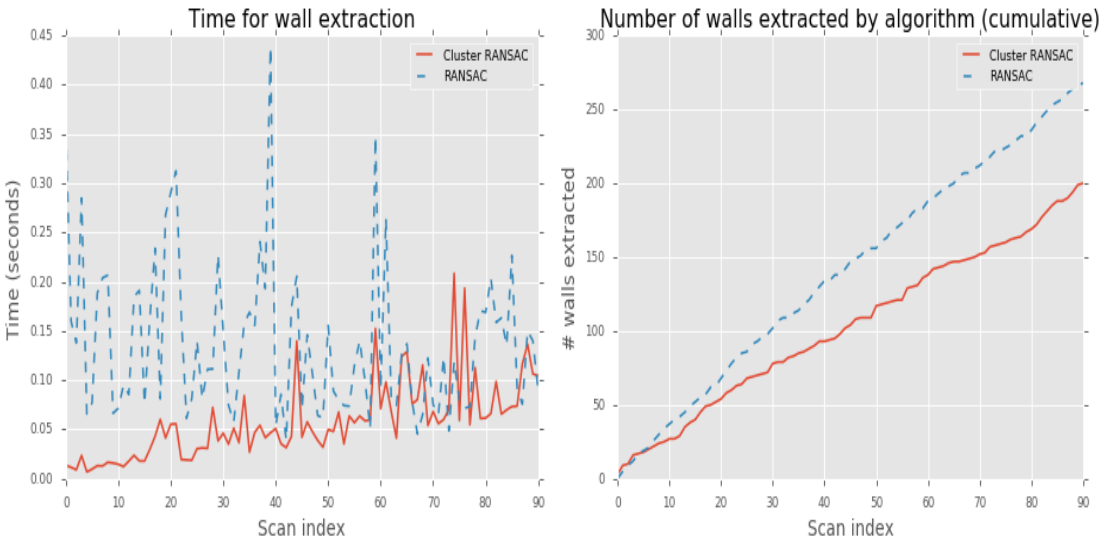


FIGURE 5.10: Time comparison and Cumulative number of walls extracted

As it can be observed in Figure 5.10, RANSAC and the clustering RANSAC algorithm proposed in Section 3.4 are compared in terms of computation time, and number of walls extracted.

RANSAC, described in Algorithm 4, extracts multiple walls from each scan by converging on a line, removing it from the data set, and afterward looking for another one. This iterative process is bounded by a minimum number of points required to be in the point cloud, so the computation time will be quite sporadic, depending on the size of the point cloud. The clustering RANSAC will only iterate depending on the number of clusters found, which is not dependent on the size of the point cloud.

The clustering RANSAC is more restrained towards inferring a wall, leading to less overall walls detected. This can be observed by looking at the overall result of the extracted walls superimposed on the real data in Figure 5.11.

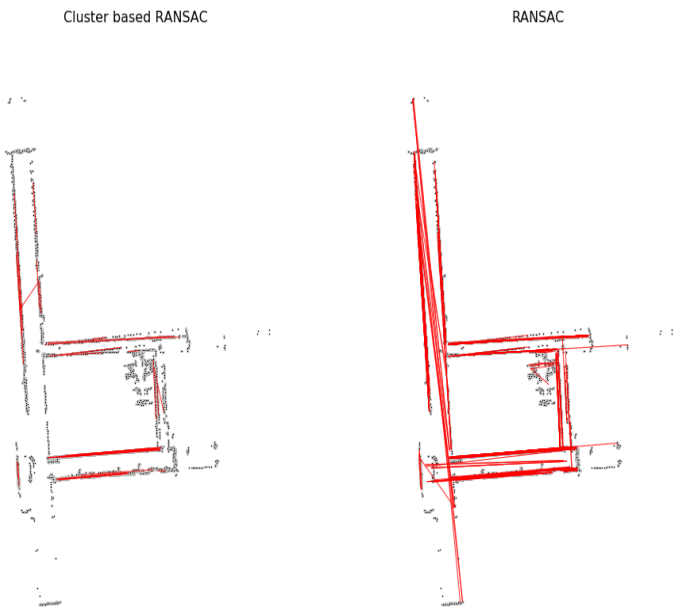


FIGURE 5.11: Walls extracted superimposed on real data, by selecting a subset of 15 scans

The lower number of walls extracted is typically desirable since the computational time for the wall filtering step and step where relationships are found between walls, grows with the number of walls.

In Figure 5.11 the walls extracted after just the first `15` scans have been used. This can be contrasted with the results shown in Figure 5.12 where all the scans have been used. As it can be observed, the performance of the Cluster RANSAC and regular RANSAC are similar although just a fraction of the scans are utilized in the former case.

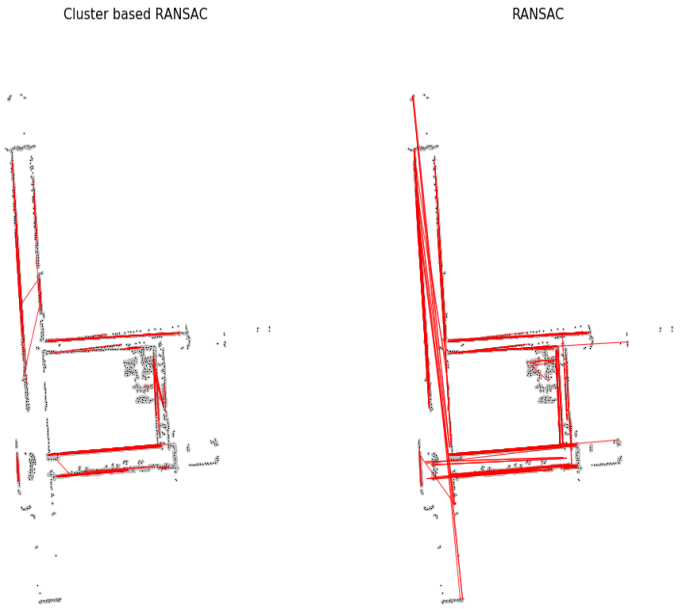


FIGURE 5.12: Walls extracted superimposed on real data, by using all 90 available scans

Chapter 6

Conclusion and Future work

In this thesis, we proposed a method of selecting a subset of scans for the purpose of a floor plan creation. Experiments have been conducted using simulated data, as well as real world data. Multiple metrics are presented for each dataset, based on the available data. We also proposed a method of extracting multiple lines from a single noisy scan. While the experiments show promising results in reducing the number of scans required, there are nevertheless limitations to this approach. Currently the quality of the data is measured by the area coverage, other metrics regarding the data should also be considered, such as the point density. Furthermore the loop between the line extraction and scan selection should be closed, such that the confidence of the line extraction should guide the selection of the scans. Also the current method of choosing scans works offline after a whole data set was collected, literature shows there are algorithms which work on real time data, a different approach would need to be taken to select scans in real time. To compare the proposed algorithmic changes, more robust error metrics for the inferred walls should be implemented.

Bibliography

[1] Icp registration.

[2] Antonio Adan and Daniel Huber. 3d reconstruction of interior wall surfaces under occlusion and clutter. In *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 275–281. IEEE, 2011.

[3] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[4] Autodesk. Autodesk invento.

[5] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

[6] Jan Böhm. Facade detail from incomplete range data. In *Proceedings of the ISPRS Congress, Beijing, China*, volume 1, page 2, 2008.

[7] buildingSMART International Ltd. Industry foundation classes release 4 (ifc4), 2013.

[8] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[9] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5556–5565, 2015.

[10] Satyan Coorg and Seth Teller. Extracting textured vertical facades from controlled close-range imagery. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages 625–632. IEEE, 1999.

[11] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[12] Andrew W Fitzgibbon, David W Eggert, and Robert B Fisher. High-level cad model acquisition from range images. *Computer-Aided Design*, 29(4):321–330, 1997.

[13] Christian Frueh, Siddharth Jain, and Avideh Zakhor. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *International Journal of Computer Vision*, 61(2):159–184, 2005.

[14] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.

[15] Sean Gillies et al. Shapely: manipulation and analysis of geometric objects, 2007–.

[16] Lewis Graham. How dense are you, anyway? *LiDAR News Magazine*, 4(5), 2014.

[17] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: Blender sensor simulation toolbox. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Song Wang, Kim Kyungnam, Bedrich Benes, Kenneth Moreland, Christoph Borst, Stephen DiVerdi, Chiang Yi-Jen, and Jiang Ming, editors, *Advances in Visual Computing*, pages 199–208, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[18] Nouha Hichri, Chiara Stefani, Livio De Luca, Philippe Veron, and Gael Hamon. From point cloud to bim: a survey of existing approaches. In *XXIV International CIPA Symposium*, page na. Proceedings of the XXIV International CIPA Symposium, 2012.

[19] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[20] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *International journal of computer vision*, 97(2):123–147, 2012.

[21] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed <today>].

[22] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[23] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science*

Conference, pages 51 – 56, 2010.

[24] Claudio Mura, Oliver Mattausch, Alberto Jaspe Villanueva, Enrico Gobbetti, and Renato Pajarola. Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics*, 44:20–32, 2014.

[25] Madhav Prasad Nepal, Sheryl Staub-French, Jiemin Zhang, Michael Lawrence, and Rachel Pottinger. Deriving construction features from an ifc model. In *Canadian Society for Civil Engineering 2008 : Partnrship for Innovation*, pages 426–436, Quebec City, Quebec, June 2008. Curran Associates, Inc.

[26] Mariano Martín Nevado, Jaime Gómez García-Bermejo, and Eduardo Zalama Casanova. Obtaining 3d models of indoor environments with a mobile robot by estimating local surface directions. *Robotics and Autonomous Systems*, 48(2-3):131–143, 2004.

[27] Tang-Hung Nguyen, Amr A. Oloufa, and Khaled Nassar. Algorithms for automated deduction of topological information. *Automation in Construction*, 14(1):59 – 70, 2005.

[28] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.

[29] Sebastian Ochmann, Richard Vock, Raoul Wessel, and Reinhard Klein. Automatic reconstruction of parametric building models from indoor point clouds. *Computers & Graphics*, 54:94–103, 2016.

[30] Travis Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006–. [Online; accessed <today>].

[31] Viorica Pătrăucean, Iro Armeni, Mohammad Nahangi, Jamie Yeung, Ioannis

Brilakis, and Carl Haas. State of research in automatic as-built modelling. *Advanced Engineering Informatics*, 29(2):162–171, 2015.

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

[33] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[34] Pingbo Tang, Daniel Huber, Burcu Akinci, Robert Lipman, and Alan Lytle. Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in construction*, 19(7):829–843, 2010.

[35] Philip H. S. Torr and Colin Davidson. Impsac: Synthesis of importance sampling and random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):354–364, 2003.

[36] Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1):138–156, 2000.

[37] Vivek Verma, Rakesh Kumar, and Stephen Hsu. 3d building detection and modeling from aerial lidar data. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2213–2220. IEEE, 2006.

[38] Tomás Werner and Andrew Zisserman. New techniques for automated architectural

reconstruction from photographs. In *European conference on computer vision*, pages 541–555. Springer, 2002.

[39] G Zhang, PA Vela, and I Brilakis. Automatic generation of as-built geometric civil infrastructure models from point cloud data. In *Computing in Civil and Building Engineering (2014)*, pages 406–413. 2014.

[40] Qian Zheng, Andrei Sharf, Guowei Wan, Yangyan Li, Niloy J Mitra, Daniel Cohen-Or, and Baoquan Chen. Non-local scan consolidation for 3d urban scenes. *ACM Trans. Graph.*, 29(4):94–1, 2010.

[41] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.