ABSTRACT

NIMAWAT, JATIN. Automatic Point Cloud Registration using Visual SLAM and Iterative Closest Point Algorithms. (Under the direction of Dr. Kevin Han and Dr. Abhinav Gupta)

360 cameras have a lot of potential in the construction industry. When used with BIMs, 360 cameras can be used for Augmented Reality (AR) based applications, construction robotics, indoor navigation, etc. However, due to the lack of any automatic registration algorithm which can align a BIM and the 360 video, the potential is largely untapped. In this study, we have developed a system for automatic registration which generates a 3D point cloud from given 360 videos using a visual Simultaneous Localization and Mapping (SLAM) algorithm called StellaVSLAM. The point cloud is then registered with the BIM using the Iterative Closest Point (ICP), a popular algorithm used for fine registration. To avoid convergence at a local minimum solution from ICP, a series of optimizations are being executed before ICP to ensure reliability in registration. The developed system has been tested against six datasets. Automatic registration works well on half of the test cases and for others, it is not achieved reliably on each dataset. We have discussed reasons why it does not work for certain datasets and recommendations have been made to improve its reliability and widen its applicability.

Automatic Point Cloud Registration using Visual SLAM and Iterative Closest Point Algorithms.

by
Jatin Nimawat

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Master of Science

Civil Engineering

Raleigh, North Carolina
2023

APPROVED BY:

Kevin Han
Committee Co-Chair

Abhinav Gupta
Committee Co-Chair

Saran Srikanth Bodda

DEDICATION

To my parents and my brother,

for their constant love and support.

And, to my friends for always being there.

BIOGRAPHY

Jatin Nimawat was born in Nimbahera, Rajasthan on September 3, 1999. He completed most of his schooling in Nimbahera at Kailash Vidya Vihar. In 2014, he moved to Kota for the preparation of Joint Entrance Exam (JEE) and studied there for three years. He started his college studies in Civil Engineering at the Indian Institute of Technology Mandi in August 2017. In 2021, he finished his undergraduate degree and joined North Carolina State University, Raleigh, NC to pursue a Master’s degree in Civil Engineering with focus in computing and systems.

ACKNOWLEDGEMENT

This dissertation would not have been possible without the immense help and support of many others. Most of all, I would like to express my deepest gratitude towards my advisor Dr. Kevin Han, for supporting, guiding, and encouraging me throughout my journey at NC State. His direction and ideas have been invaluable in understanding the problem of point cloud registration and coming up with a solution. I would also like to thank Dr. Abhinav Gupta for his support and guidance in academics as well as outside academics. His outstanding work ethic has been a great source of inspiration to me, and his mentorship has helped me decide and work towards a life that I would like to pursue. Also, I cannot thank him enough for his support in providing me Graduate Student Support Plan along with Dr. Han throughout my Masters's.

I would also like to thank Dr. Saran Srikanth Bodda for providing support outside of academics and helping me keep going in tough times. I am grateful to Dr. Doyun Lee for all the help he provided me in writing this thesis and expressing my work in a clear and concise manner.

I am thankful to my parents, Mr. Sohan Das Vairagi and Mrs. Madhu Vairagi for their unconditional love, and endless support. I cannot express enough gratitude for the selfless sacrifices they have made for my education and happiness. I am grateful to my brother, Mr. Nikhil Nimawat for his support and guidance in striving for greater heights in life.

I am thankful to my friends, Ms. Stuti Garg, Mr. Sudhir Kumar Gupta, Mr. Antony John Edathattil, Ms. Mega Viswanathan, Ms. Varsha Verma, and Mr. Aswin Chembu Keezhedathil for being there, and providing support to navigate my life here in the United States. They are a big part of my life and I cherish the memories I made with each of them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Problem Statement

In recent years, the construction and civil industry has been changing rapidly. The rise of BIM has made collaboration between architects, engineers, and construction professionals quite seamless. With its inter-organizational and cross-disciplinary capabilities, BIM has become ubiquitous with 99% of the industry aware of BIM and about 73% of the construction industry already using BIM according to a 2020 study [1]. Similarly, the usage of panoramic cameras is also on the rise. Tasks like progress tracking [2], project documentation [3], safety training [4], and site inspections [5] are more often being done with the usage of panoramic or omnidirectional cameras. Panoramic cameras provide a "sense of presence, of being there" [6] unlike any other technology used for these applications. Panoramic cameras have a lot of potential when used with BIM. Omnidirectional images, when used registered alongside BIM unlock a vast range of applications including virtual tour creation, photogrammetric measurements, indoor navigation, other AR-based applications, and many more. However, despite having such vast potential, due to the lack of any automatic registration algorithm, the usage of 360 images or video with BIM is rarely seen.

The main challenges in creating an automatic registration system for BIM and 360 images are:

1. For registration, the coordinate systems of the BIM and the images need to be aligned. However, images, especially, a set of images, do not have any 3D coordinate system to perform any alignment.
2. If we introduce a coordinate system in all the images, by using a visual SLAM algorithm and develop a point cloud out of the images, since the point cloud has been generated

independently of the BIM, there is no correspondence information between the two to perform registration on (Figure 1).

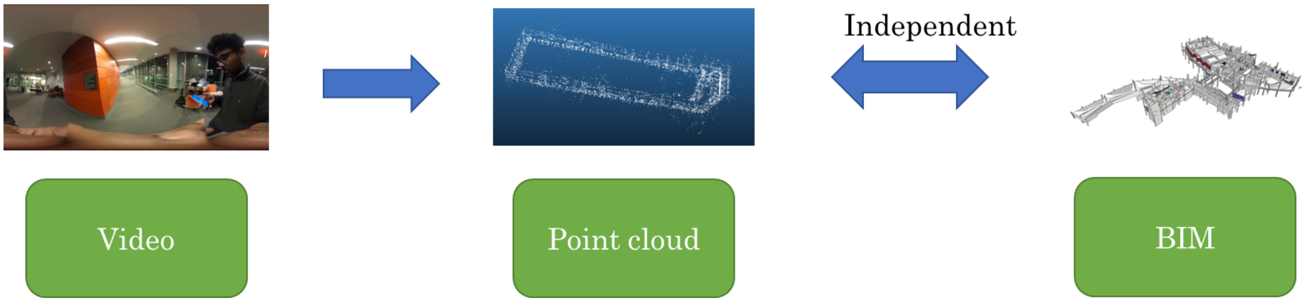


Figure 1: Point cloud generated from images has no correspondence information to register with BIM.

Various studies have explored the idea of using panoramic images registered with BIM for construction but most of them rely on manual registration. For example, a study by Ghesari and Sabzevar in 2016 [7] and also [8] in 2011 developed an augmented panoramic environment that overlaid a panoramic image over the BIM but in both the studies, the registration was performed manually. Additionally, in the latter, the registration was done on a static image with a fixed camera angle.

Studies with the aim to develop automatic registration systems more often use dense laser-scanned point clouds (generated by very expensive sensors costing tens of thousands of dollars) instead of either panoramic images or visual SLAM-generated point clouds.
Iterative Closest Point (ICP) is one of the most common methods of automatic point cloud registration. However, ICP is usually used for fine corrections in registration. In ICP, the registration configuration is obtained by iteratively going through various orientation, translation, and scale configurations of the first point cloud and finding the minimal RMS distance of the second point cloud and the first point cloud.

In this study, we will explore the problem of automatic point cloud registration of panoramic video and a BIM using a visual SLAM algorithm. We also explore the potential of the Iterative Closest Point (ICP) algorithm [9][10] for fully automatic point cloud registration rather than just for fine corrections.

1.2 Research Goals and Overview

The ultimate objective of this research is to develop an automatic BIM-point cloud registration system where the point cloud is generated from a set of images. The study also aims to assess the potential or use of ICP for automatic registration algorithms. Here, we developed a registration system that takes a BIM and a walkthrough video of a floor for a given building as an input and provides registered point cloud-BIM configuration along with the transformation matrix as an output. At the core of this system, floor plans are extracted or generated from the two inputs, and then iteratively compared using the ICP algorithm on various configurations to find the best fit.

Floor plan extraction from the BIM is relatively straightforward as a simple section cut on the 3D model provides the necessary information for an accurate floor plan. On the other hand, the generation of a floor plan from a video needs to be performed in multiple steps. First, a point cloud is generated using a visual SLAM algorithm called StellaVSLAM. This point cloud is then cleaned, either automatically or manually, and a boundary is traced over the cleaned point cloud, generating an approximate floor plan of the building. The two-floor plans are then compared iteratively using ICP on various initial configurations, and the best fit from these iterations is returned as the final output. The workflow of the system is depicted in Figure 2.

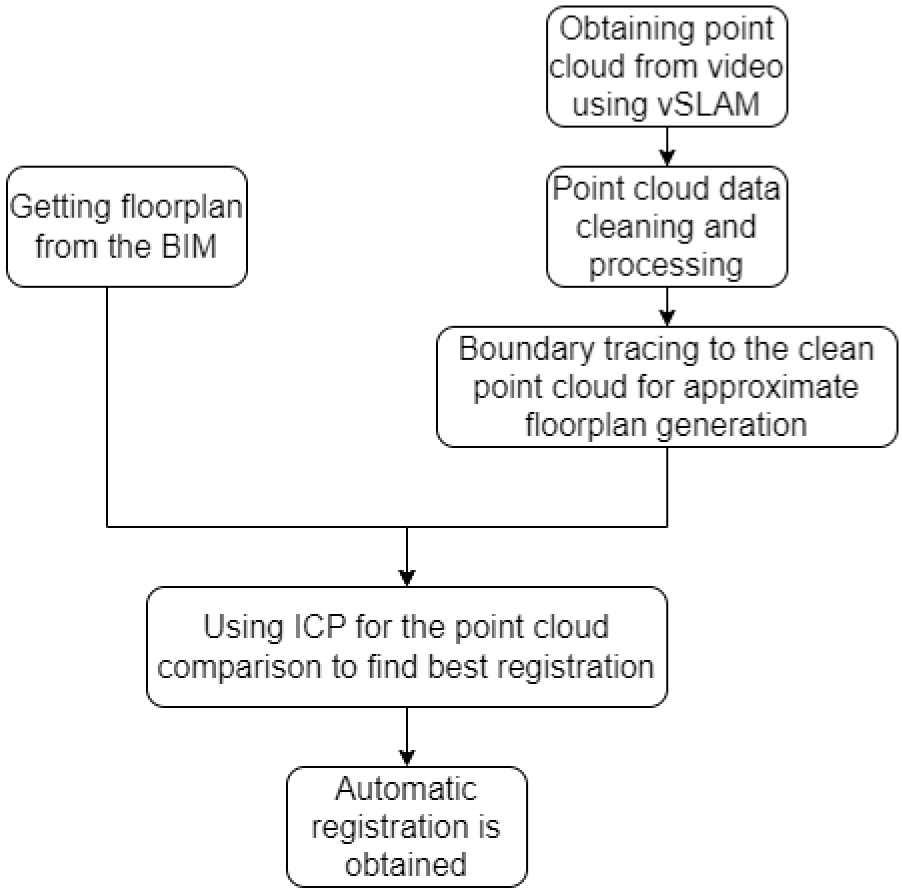


Figure 2: Workflow of the registration system.

The accuracy of a video-generated floor plan highly depends on how clean the supplied point cloud is. Inaccuracies in cleaning lead to inaccurate video-obtained floor plans. This study has also assessed how these inaccuracies affect ICP results' accuracy. For each test dataset, registration results with manually and algorithmically cleaned point clouds are compared and discussed.

CHAPTER 2: BACKGROUND AND RELATED STUDIES

2.1 BIM-Point Cloud Registration

BIM-Point cloud registration is a problem of relating entities from different domains, namely, a Building Information Model (BIM) and a Point Cloud. The idea is to develop a transformation matrix that aligns the point clouds' coordinate system with that of the BIMs. As mentioned previously, one of the main solution algorithms for this problem is ICP. The ICP starts with two-point clouds, a query point cloud, and a reference point cloud. On the query point cloud, several scales, rotation, and translation transformations are applied iteratively. At each iteration, the average Root Mean Square (RMS) distance between this transformed query point cloud and the reference point cloud is calculated. When the RMS distance drops down below a threshold RMS value, registration is achieved. The threshold RMS is determined on a case-by-case basis. Figure 3 shows the workflow of ICP algorithms.

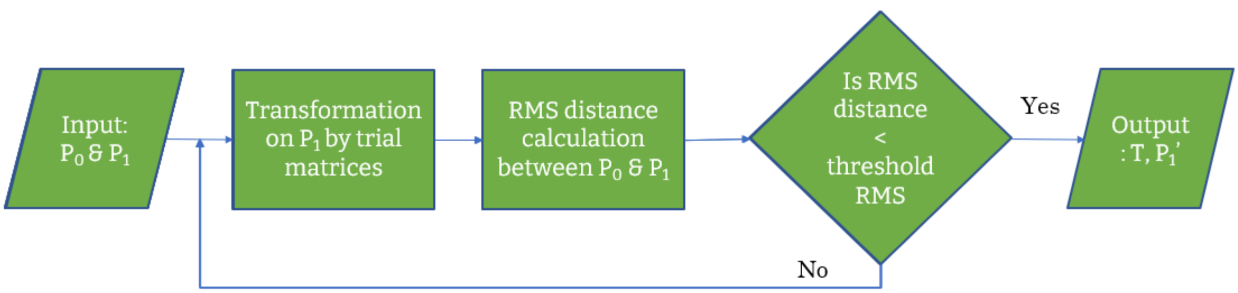


Figure 3: Algorithmic flowchart of the Iterative Closest Point (ICP) algorithm.

There are multiple variants of the ICP algorithm, each with modifications customized for their use cases. For example, Sparse ICP [11] expresses ICP into sparse optimization problems or Efficient Sparse ICP [12], which is based on Simulated Annealing [13] and Alternating Direction Method of Multipliers based ICP. There is also a Generalized ICP [14] along with various other variants of ICP [15], which are commonly used for point cloud registration.

Coherent Point Drift (CPD) [16] has recently emerged as another popular method for point cloud registration, where this problem is posed as a probability density estimation problem. In this method, a point cloud geometry is created on a Euclidean space of the reference point cloud using a GMM, which is made by introducing isotropic covariances at each point. In contrast, the query point cloud is fitted with a Maximum Likelihood Estimator (MLE).

One big drawback of CPD is that it requires extreme computation power. To address this issue, many variants have been developed of the CPD. Accelerated CPD has been developed [17], accelerating the CPD approach by more than an order of magnitude. LSG-CPD [18] is another improvement on the base CPD algorithm incorporating local surface geometry components for a better fit. However, in all the variants of CPD, high computation power requirements restrict its omnipresence in point cloud registration.

Another popular method of BIM-point cloud registration is by identification of objects from the point cloud and alignment of the corresponding objects in the BIM [19]-[21]. In these approaches, image databases are required to make sense of the point cloud. Recent studies [22][23] have used synthetic BIM-generated images to avoid such hassle for reference object identification. Photogrammetry and Structure-from-Motion (SfM) are developing fast, but these methods require relatively high computing power and leave information from BIM unexploited. Additionally, since SfM is not real-time, SLAM turns out as the preferred alternative.

Although not as popular as previously mentioned approaches, Hausdorff distances can replace RMS distances in ICP for BIM-point cloud registration. Hausdorff distances [24] represent the maximal distance between any point in one shape to its closest point in the second point cloud. However, focusing on minimizing only one point's distance during ICP can lead to final registration with substantially inaccurate orientation and scale errors.

2.2 SLAM: StellaVSLAM

Simultaneous Localization and Mapping (SLAM) algorithms are quite useful in cases where there is a requirement to generate point clouds out from images. For registration, it is essential to introduce a sense of coordinate system in the images. Visual SLAM is one of the best methods to map out a 3D environment from the given visual data. Although most SLAM algorithms are mainly developed for LiDAR, or Visual-Inertial cameras, odometry information, mapping, and localization can be also performed using just the visual information. In visual SLAM, visual features are identified and tracked frame-by-frame for mapping and localization. Since panoramic cameras, if used standalone, can be only used with visual SLAM algorithms, we will focus our discussion on visual SLAM algorithms.

Most visual SLAM algorithms are developed to run with various camera configurations, including stereo, monocular, and even depth camera configurations. Each configuration has a further distinction based on the situation; the monocular cameras can be omnidirectional, fisheye, or standard/perspective. A plethora of SLAM algorithms have been developed over the years for each type of sensor setup like ORB SLAM [25], EKF SLAM [26], LSD SLAM [27], RTABMAP [28], FastSLAM [29], GraphSLAM [30], etc.

Because of the ubiquity of cameras in our life nowadays and the proliferation of cameras, we have chosen to work with Visual SLAM with a monocular camera setup. The added flexibility of not relying on a particular type of sensor makes the algorithm developed here more accessible. Although the algorithms and programs developed here can work with any commonly used camera, we have used an omnidirectional camera setup in the study. Various visual SLAM algorithms have been developed to work with omnidirectional monocular camera configuration, for example, LSD-SLAM [31] developed by the Technical University of Munich, SVO [32] from the University of Zurich, RSLAM [33] from Oxford University, etc. The best choice among all the options, because of its solid documentation on using omnidirectional (and fisheye) cameras with this algorithm, is the ORB SLAM algorithm-based open-source visual SLAM algorithm called StellaVSLAM [34]. The home page of its website is populated with a point cloud generated from the fisheye camera lens as shown in Figure 4. The tutorial is quite easy-to-follow and comes with its sample dataset to test (Figure 5). Use of ORB SLAM or ORB SLAM-derived studies in construction is plenty, like [35][36] but most of the work done in this area does not deal with registration using visible imaging cameras.

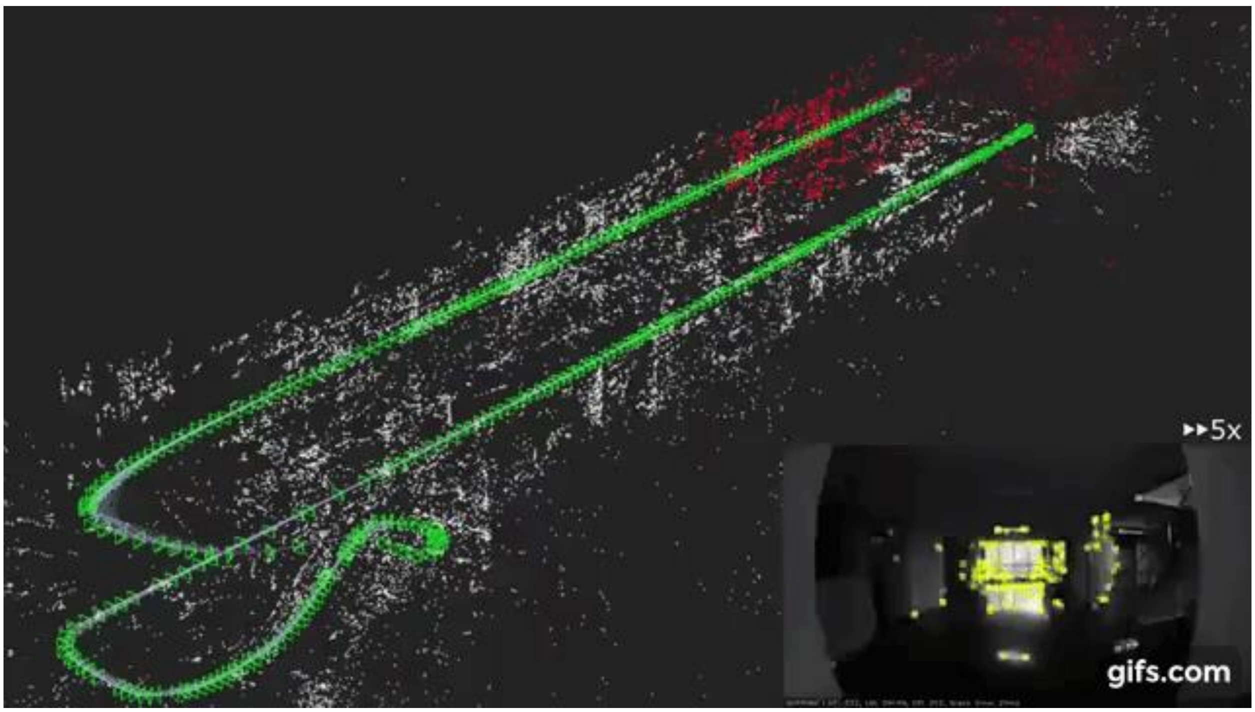
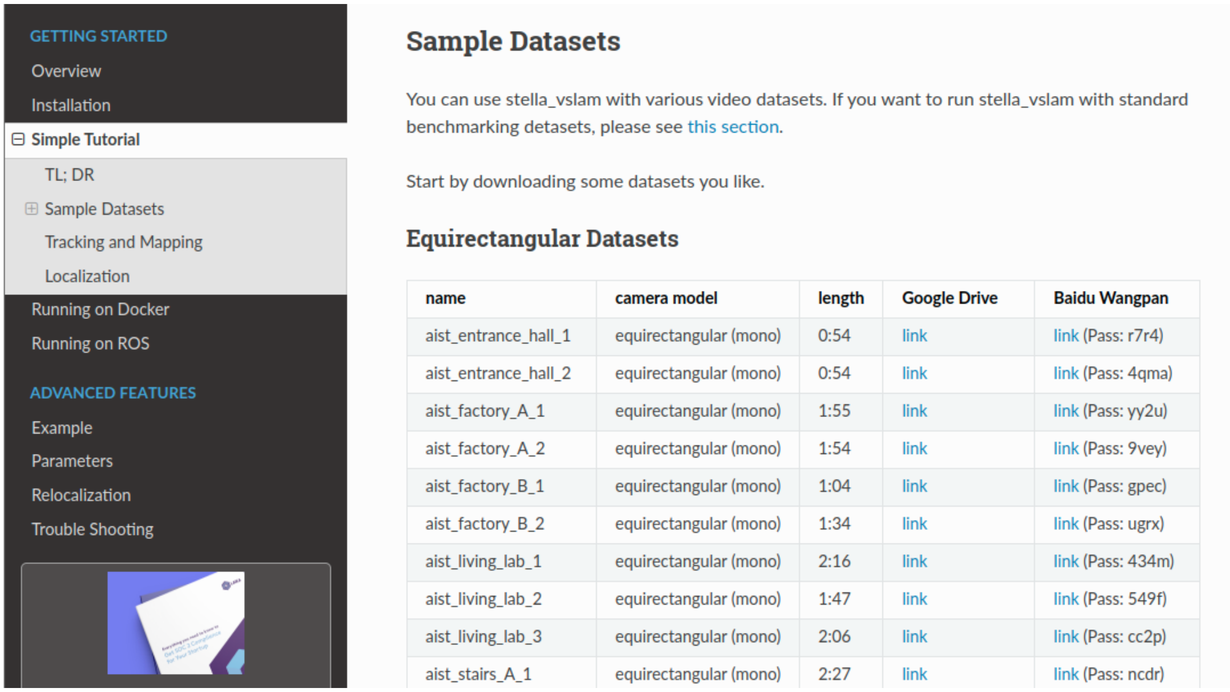


Figure 4: Point cloud generated by StellaVSLAM from a sample fisheye camera dataset. The green squares represent keyframes.


GETTING STARTED
Overview
Installation
Simple Tutorial
TL; DR
Sample Datasets
Tracking and Mapping
Localization
Running on Docker
Running on ROS
ADVANCED FEATURES
Example
Parameters
Relocalization
Trouble Shooting

Sample Datasets

You can use stella_vslam with various video datasets. If you want to run stella_vslam with standard benchmarking detasets, please see this section.

Start by downloading some datasets you like.

Equirectangular Datasets

name	camera model	length	Google Drive	Baidu Wangpan
aist_entrance_hall_1	equirectangular (mono)	0:54	link	link (Pass: r7r4)
aist_entrance_hall_2	equirectangular (mono)	0:54	link	link (Pass: 4qma)
aist_factory_A_1	equirectangular (mono)	1:55	link	link (Pass: yy2u)
aist_factory_A_2	equirectangular (mono)	1:54	link	link (Pass: 9vey)
aist_factory_B_1	equirectangular (mono)	1:04	link	link (Pass: gpec)
aist_factory_B_2	equirectangular (mono)	1:34	link	link (Pass: ugrx)
aist_living_lab_1	equirectangular (mono)	2:16	link	link (Pass: 434m)
aist_living_lab_2	equirectangular (mono)	1:47	link	link (Pass: 549f)
aist_living_lab_3	equirectangular (mono)	2:06	link	link (Pass: cc2p)
aist_stairs_A_1	equirectangular (mono)	2:27	link	link (Pass: ncdr)

Figure 5: The tutorial on StellaVSLAM directly contains documentation on the use of omnidirectional cameras.

Since StellaVSLAM is based on ORB SLAM, it shares most of the inner workings of ORB SLAM, making it a fast, robust, and accurate SLAM system. StellaVSLAM detects and tracks the Orientated FAST and rotated BRIEF (ORB) [37] features as map points throughout the frames of the supplied set (a sample can be seen in Figure 6) and plots them into a 3D world space. The calculation and matching of ORB features are fast and invariant to viewpoints [38].

Figure 6: A sample image was supplied image to StellaVSLAM. The diamond-shaped red-orange dots represent the ORB feature. These are tracked frame by frame and plotted in 3D space.

An omnidirectional video can be directly supplied to the StellaVSLAM algorithm; however, there are two prerequisites. A file containing ORB vocabulary is required, and a file containing relevant computer vision, system & intrinsic camera parameters is needed.

For ORB vocabulary, StellaVSLAM comes with a default ORB vocab file. This file has been developed over a large variety of data. Thus, it is usable for general test case scenarios like the one in this study. A custom ORB vocabulary file can be used here if the algorithm is used with specific cases or if higher accuracy in feature detection is needed.

Various parameters are needed for the camera configuration prerequisite file, such as the camera setup, camera model, fps, resolution, color_order, mask_rectangles, feature properties, etc. These need to be modified according to the experimental setup and the video on which SLAM is applied. In addition, StellaVSLAM comes with a sample file for equirectangular and fisheye use cases containing default values for system and computer vision parameters.

Once all the prerequisites are met, StellaVSLAM is ready to be executed. Based on the length of the video, StellaVSLAM can take a few minutes to finish. Once done, StellaVSLAM saves its output in a .msg file, a highly compressed file format [39]. This output file contains all the information about the point cloud, the camera trajectory, and the camera's extrinsic parameters. However, to use any of the information, data needs to be extracted from the output .msg file using a C++ library called msgpack.

CHAPTER 3: METHOD

In this chapter, we will go over the workflow of the automatic registration system. Then, I will look at each component's role and high-level details on the execution.

3.1 Workflow for Automatic Registration

Although we touched upon the workflow of the registration algorithm in the introduction, a detailed discussion is necessary to understand the logic of the automatic registration system. For successful registration, we generate floor plans from the visual information provided by the video and from the BIM. Then, we perform various fitting comparisons on several orientation transformations, translation transformations, and scale transformations. Figure 7 shows the workflow of the registration system we developed here.

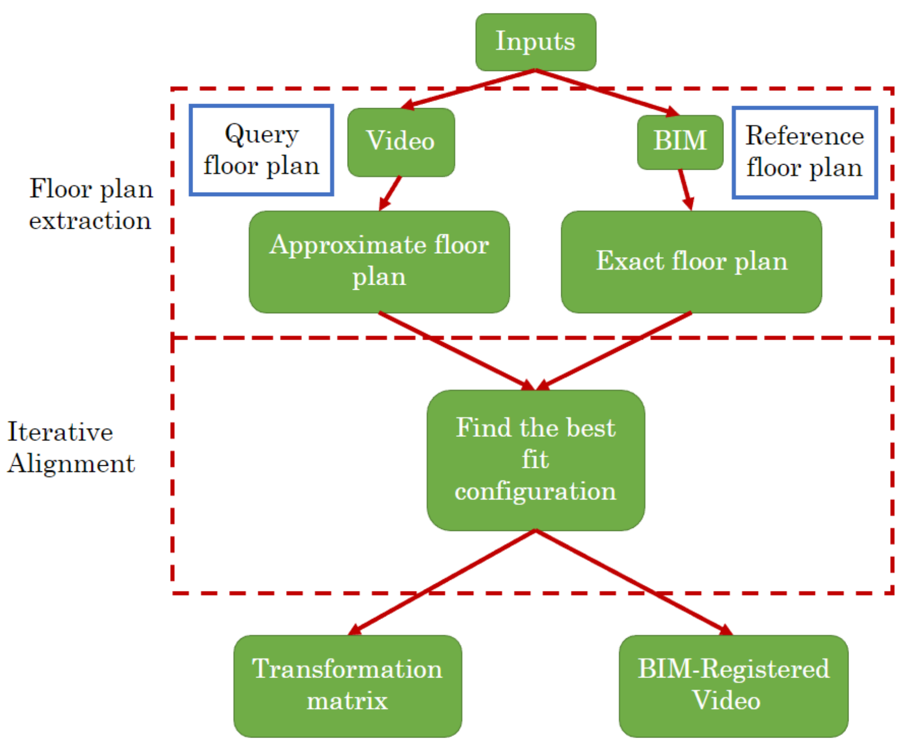


Figure 7: A workflow diagram of the automatic registration system developed in this thesis.

As seen in the flowchart diagram, the code system can be divided into two phases, Floor plan extraction, and Iterative Alignment.

In the first phase, we extract the floor plan from the two inputs, the BIM and the panoramic video. Extracting a floor plan from a given BIM is quite straightforward. A section cut across the BIM can generate a floor plan. Here, our code does exactly that. It creates a section cut across the BIM and records all the intersection lines in an SVG file.

The next step is to obtain the floor plan from the omnidirectional video. Using a visual SLAM algorithm called StellaVSLAM, a sparse point cloud is generated from the handheld video. After some cleaning and processing (done either manually or algorithmically), the point cloud can give out an approximate floor plan shape. A boundary is drawn around the clean point cloud using the concepts of convex and concave hull, and an approximate floor plan is generated from a video.

Once we have the two-floor plans, one exact (obtained from the BIM) and one approximate (obtained from the video), we apply ICP to find the best-fitting configuration. These fitting comparisons are made by calculating the RMS distances of the first-floor plan to the other. The comparisons are performed until the RMS distance value decreases below a threshold. Once the best fit is obtained, the transformation matrix is stored along with the transformed point cloud.

With this overview of the code, let’s try to dive deeper into each component of the automatic registration system we developed here.

3.2 Floor Plan Extraction

3.2.1 From the BIM

As mentioned above, the first step is to obtain the floor plan from the given BIM file. There are several ways to obtain the floor plan for a given building. For example, it can be extracted from the RVT or IFC files, obtained separately along with BIM, or generated by taking a section cut of a 3D model.

However, in all these formats except the last one, the obtained floor plan would be in a different coordinate system than the building model itself. This would necessitate a transformation matrix that relates the 3D model to the obtained floor plan. Determining such a transformation matrix is not straightforward and can be a research problem. However, with the latter approach, the coordinate system will remain the same.

So, in this study, we have chosen the latter, taking a direct section cut of the 3D model to obtain the floor plan. The developed system takes an IFC file as an input, converts it to an OBJ file, a popular file format for 3D models, using IFCOpenShell [40], and takes a section cut of the 3D model at a user-defined height. All the information from the section cut is then stored in a Scalable Vector Graphics (.svg) file format.

Examples of the input and the output for the floor plan extraction are shown in Figure 8. Figure 8 shows the input (on the left) and output (on the right) of the simplistic test cases where the developed system is run on a simple house. Figures 9 (a) and (b) are the test cases run on a more complicated building.

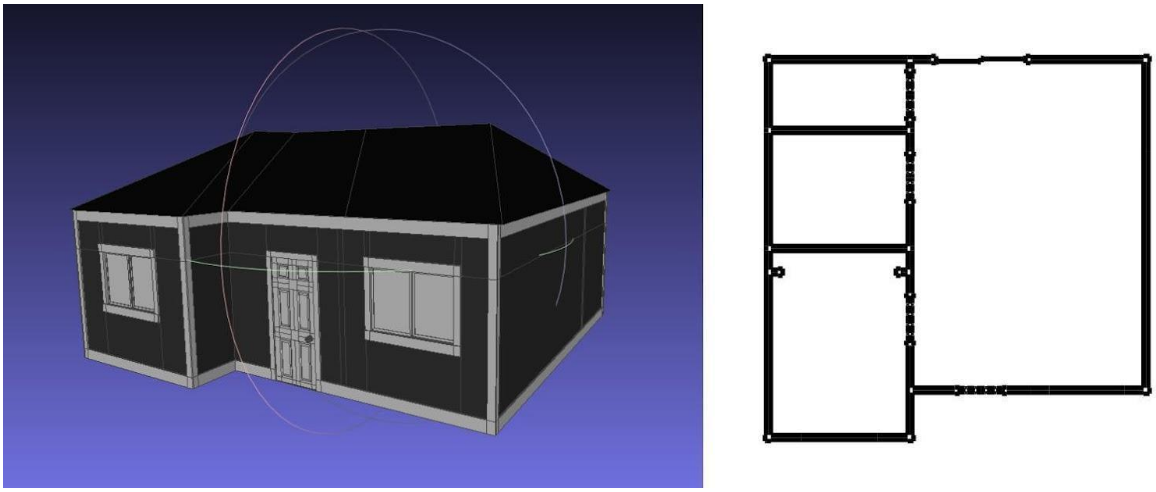
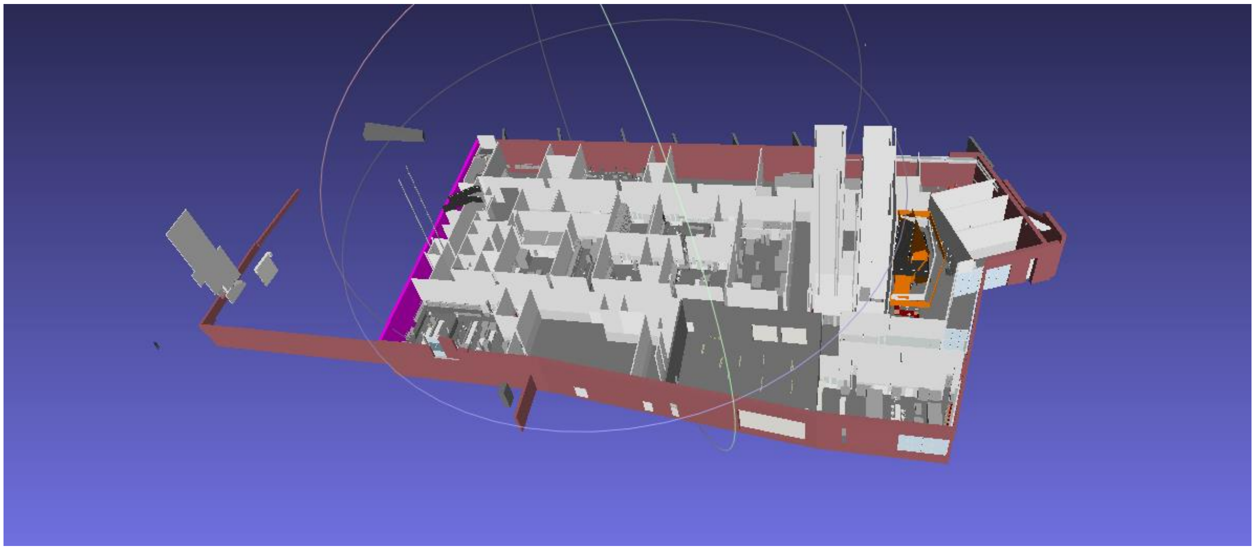


Figure 8: 3D model of a simple house and its floor plan when sectioned at 1 foot above the ground.



a)

b)

Figure 9: a) 3D model of Fitts Woolard Hall, a building at NC State University b) Output of floor plan extraction code on the same 3D model.

Once we have the BIM's floor plan, the workflow's first branch will be completed. we can now move to the other branch of the workflow.

3.2.2 From Panoramic Video: Using StellaVSLAM

The next step is to obtain a 3D point cloud from an omnidirectional video. As mentioned in the Background chapter, we have used StellaVSLAM. A video is supplied to the SLAM algorithm, ORB vocabulary, and camera configuration to obtain the point cloud. Since StellaVSLAM is an ORB SLAM-derived algorithm, it first detects the ORB features, tracks their motion frame-by-frame, and locates it in a 3D space.

More information about StellaVSLAM can be found in [41]. The obtained point cloud is stored in the msg format, which needs to be extracted and exported. With minor changes in the code provided by the StellaVSLAM documentation, point cloud information is extracted into more common formats, e.g., CSV or PLY.

The obtained point cloud maps ORB features in a 3D space. Figure 10 shows a sample point cloud loaded in an open-source point cloud modification and manipulation program called CloudCompare.

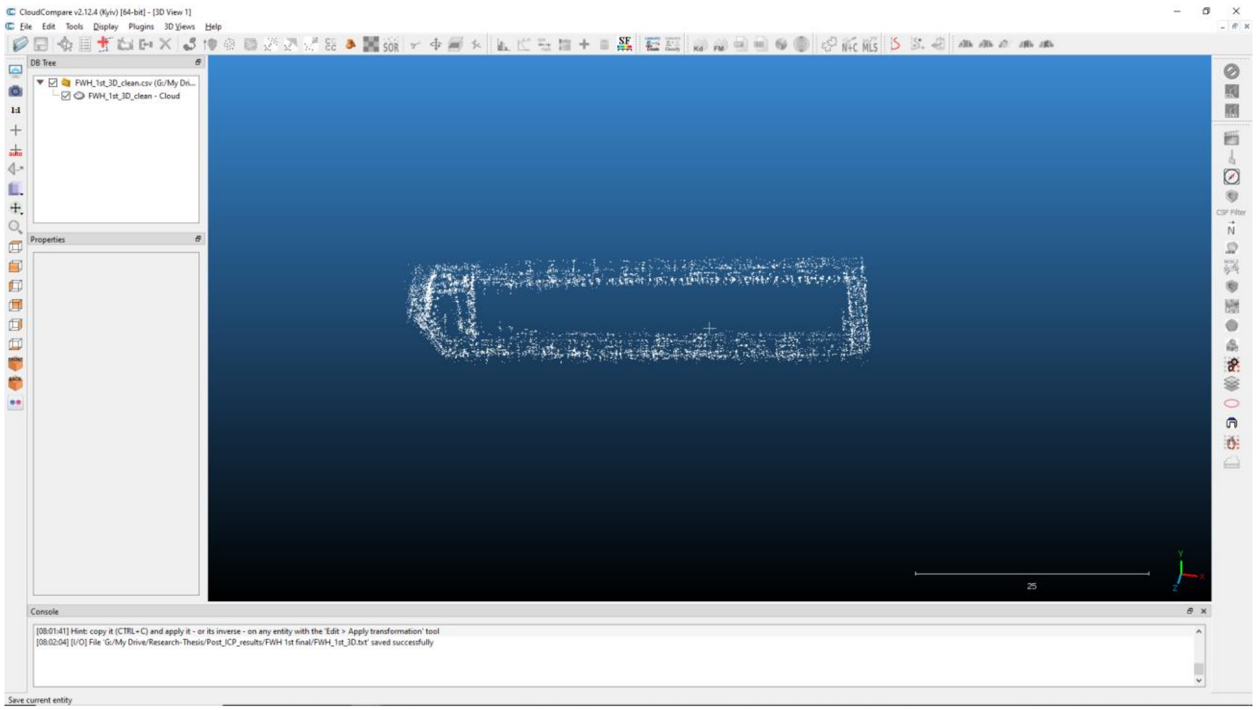
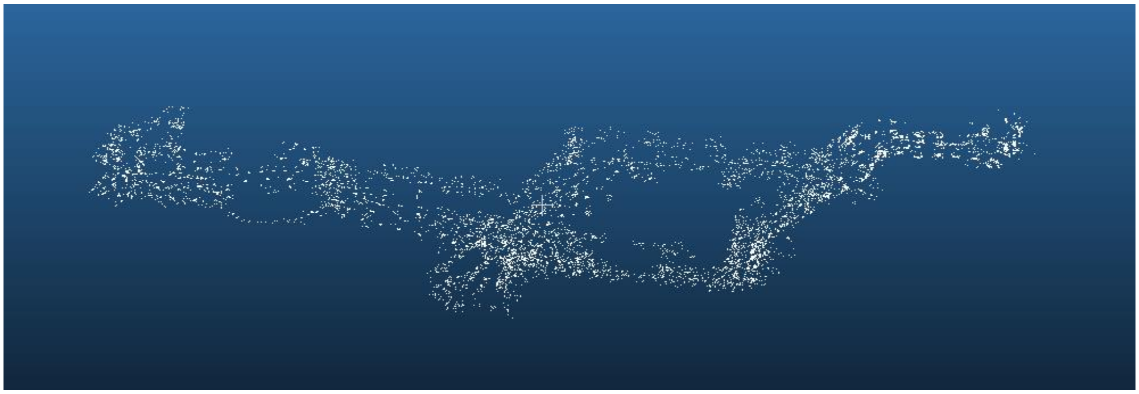


Figure 10: A sample point cloud obtained using StellaVSLAM is loaded on CloudCompare. An omnidirectional video of the first floor of Fitts Woolard Hall building, North Carolina State University.

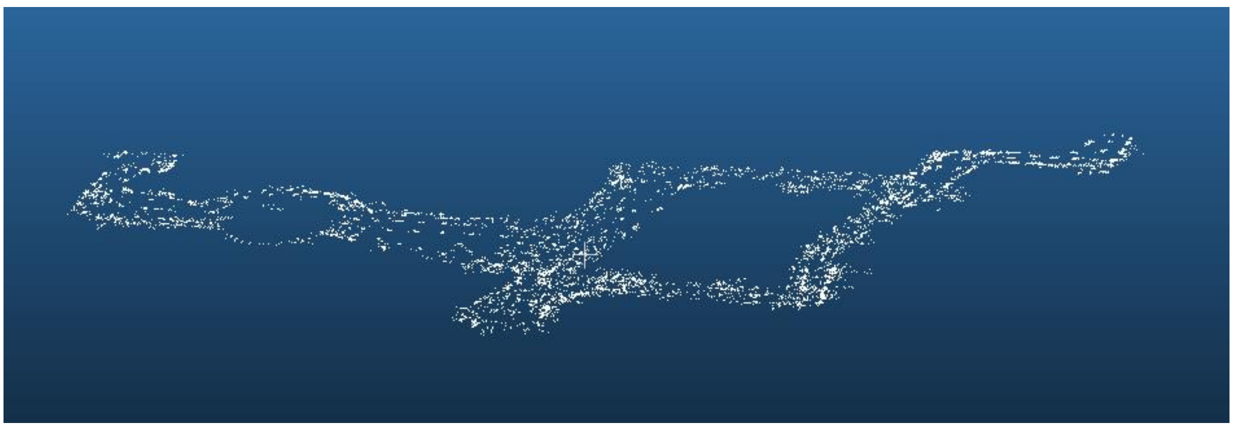
3.2.3 Data Processing on Point Clouds

The raw point cloud obtained from StellaVSLAM cannot be used directly to generate a floor plan. Instead, the data needs to be cleaned, outliers must be removed, and the data needs to be converted into two dimensions. There are two options for data cleaning and outlier removal. The first method manually removes the outliers and cleans up the point cloud. This can be performed on CloudCompare using its segment tool. Otherwise, any of the data-cleaning algorithms can be used. In this thesis, we have used the Statistical Outlier Removal (SOR) method implemented in CloudCompare for automatic data cleaning. In the SOR method, the mean distance of all points from their nearest neighbors is calculated, and the points that are further than a certain number of standard deviations from the mean of mean distances, classified as outliers, are removed [42].

Once the data is clean, the next step is to generate a 2D projection of this clean point cloud. This is created by removing the axis perpendicular to the plan view of the building. Figure 11 demonstrates this step, a) a manually cleaned 3D point cloud versus its b) 2D projection generated by removing the third axis from the 3D point cloud file.



a)



b)

Figure 11: a) 3D point cloud obtained from StellaVSLAM b) 2D point cloud generated by removing one axis from a) point cloud.

Tracing the clean 2D point cloud's outline can generate a part of the floor plan. This outline represents the outer walls of a hallway in which the omnidirectional video has been recorded. The outline is obtained here using MATLAB's boundary function. The polygon generated here is one of the alpha shapes [43][44] or the point cloud. However, using just the outer boundary for floor plan comparison can lead to false positive registration results. So, we generated a floor plan for the inner hallway walls from the recorded video as well. For that, we created an "inverse" of the original point cloud. An “inverse” point cloud has data points in the empty region of the original point cloud and does not have any data points in regions that were populated with points in the original point cloud. This "inverse" point cloud is generated by discretizing the area inside the outer walls into smaller cells. As shown in Figure 12, a small window is traversed through all these cells. We are interested in the empty cells for the "inverse" point cloud, not containing any data point from the point cloud. All the empty cells are marked to generate a uniform "inverse" point cloud. Creating a boundary to this uniform inner point cloud results in the floor plan of the inner hallway which is then added to the outer hallway floor plan to create the approximate floor plan.

Figure 12: A small window moving on the 2D point cloud to create an "inverse" point cloud.

Once a plan for inner and outer walls is obtained, this information is combined to generate an approximate floor plan. A sample approximate floor plan can is shown in Figure 13.

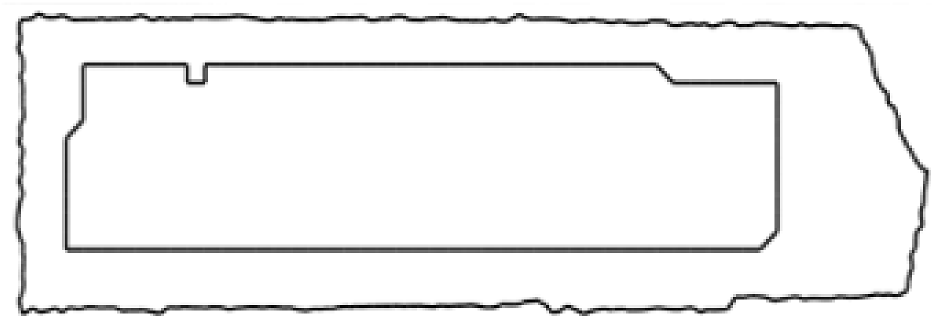


Figure 13: An approximate floor plan obtained from the panoramic video.

At this point, we have obtained information about both the floor plans, from the BIM and the point cloud. Now, we move to the next phase of our method, the iterative alignment.

3.3 Iterative Alignment

As mentioned in the previous chapter, ICP is one of the most popular algorithms for fine corrections in automatic registration. In ICP, given two point clouds, one point cloud iteratively undergoes various transformations to fit the second point cloud. Fitness at each transformation is recorded. When a threshold fitting parameter is reached, the transformation for such fit is recorded.

ICP works well for fine corrections in registration but cannot be reliably used for registration. ICP has a tendency to converge on local minima. It is very sensitive to the initial conditions; it can yield very different outputs with a slight change in initial conditions. Figure 14 demonstrates the sensitivity of the ICP on the initial conditions. Figure 14 a) and b) show the initial conditions before ICP and c) d) show corresponding registration results for a) and b) after ICP is executed.

Even with such slight changes in the initial conditions (as seen in Figures 14 a) and 14 b), the output has varied quite substantially (Figures 14 c) and 14 d). This is one of the main reasons ICP is only used for fine corrections.

Figure 14: ICP results with marginal changes in initial conditions.

For iterative alignment, we propose a method where the initial conditions are approximately registered and then we apply ICP. ICP executed after that would be expected to have reliable automatic registration and be called a Guided ICP. A schematic workflow for Guided ICP is shown in Figure 15.

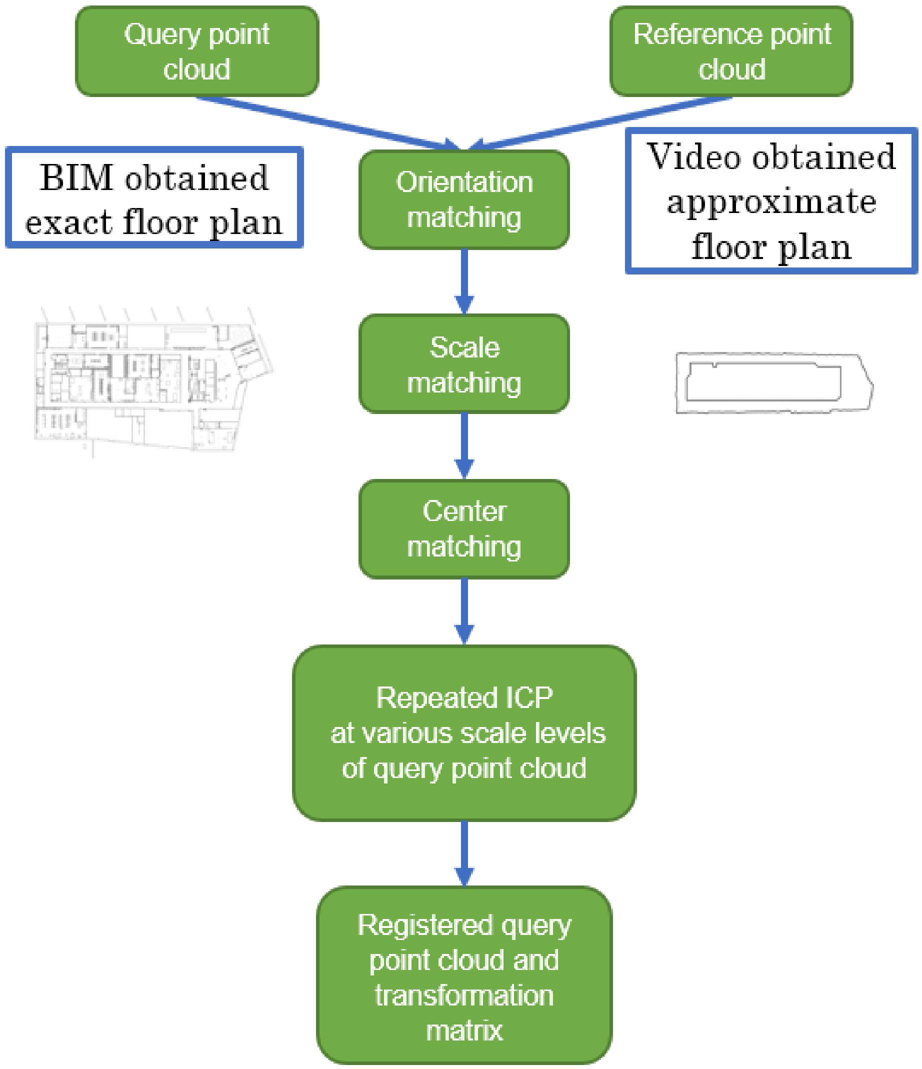


Figure 15: Guided ICP algorithmic flow chart.

In Guided ICP, we match the orientation of the query point cloud to the reference point cloud. For orientation matching, the mean angle each point subtends from its centroid is calculated for each input point cloud. Here, the input point cloud is the BIM obtained exact floor plan, and the panoramic video obtained the approximate floor plan. The query point cloud is then rotated to match its mean angle to that of the reference point cloud. Figure 16 shows a flowchart of the working of orientation matching and the pseudo-code for orientation matching is as follows:

```
for each point cloud:
    x_centroid = average(x)
    y_centroid = average(y)

    for every point p in the point cloud:
        x_diff = p_x - x_centroid
        y_diff = p_y - y_centroid
        angle = atan(y_diff/x_diff)

    mean_angle = average(angle)

rotation = mean_angle_reference − mean_angle_query
query_pc_oriented = Apply_rotation(rotation, query_pc)
```

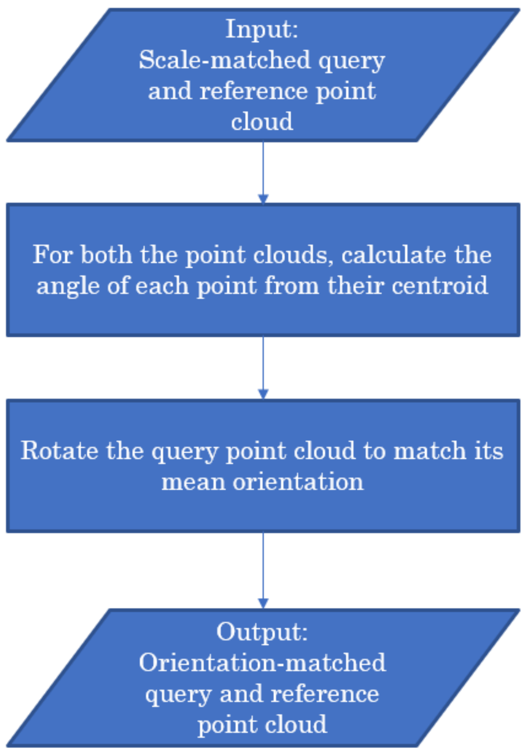


Figure 16: Flow chart for orientation matching.

Next, we perform scale matching on the orientation-matched query point cloud. First, we calculate bounding boxes for both point clouds. A bounding box is the smallest rectangle that completely surrounds all the points in a point cloud with all its sides parallel to the coordinate axes. Then, the biggest dimension of the bounding box of the query point cloud is scaled to match that of the reference point cloud. Figure 17 shows the flowchart of the scale-matching algorithm. The pseudo-code for scale-matching looks as follows:

```
for each point cloud:
    x_min = min(x)
    x_max = min(x)
    y_min = min(y)
    y_max = min(y)

    bounding_box = ((x_min, y_min),
                    (x_max, y_min),
                    (x_max, y_max),
                    (x_min, y_max))

    x_side = x_max - x_min
    y_side = y_max - y_min
    max_side = max(x_side, y_side)

scale = max_side_reference/max_side_query
T = Apply_transform(scale, query_pc)
```

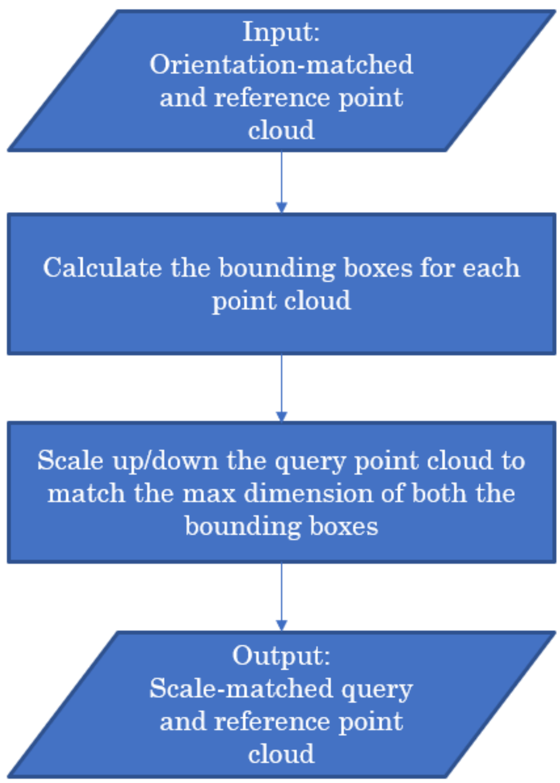


Figure 17: Algorithmic flowchart of scale matching.

In scale matching, the objective is not exactly to scale match the query point cloud with the reference point cloud, but rather to bring them in a similar range. The exact scale matching is performed by the following ICP procedure.

Finally, we move to center matching. In center matching, the centroids of the orientation and scaled-matched query point cloud are translated to match that of the reference point cloud. Figure 18 shows the schematic for the flow of center matching.

```
for each point cloud:
    x_centroid = average(x)
    y_centroid = average(y)

translation = [(x_centroid_reference - x_centroid_query),
               (y_centroid_reference - y_centroid_query)]

T = Apply_transform(translation)
query_pc_centered = T*query_pc_orientated
```

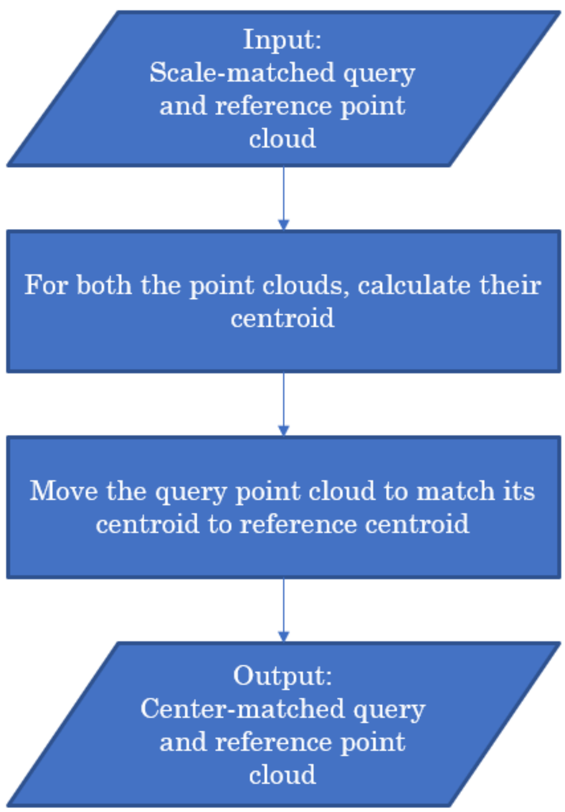


Figure 18: Algorithmic flowchart of the center matching algorithm.

At this point, the query point clouds, and the reference point clouds are approximately matched. However, the floor plan obtained from the video only contains plan information of hallways or corridors in which a panoramic video is taken, the query point cloud, when registered, is expected to be smaller than the reference point cloud and, in scale-matching, the query point cloud was scaled to match the reference point cloud. So, at these initial conditions, the query point cloud is bigger than it should be. So, now, we iteratively decrease the scale of the query point cloud and at each iteration, ICP is applied. The scale level at which the RMS distance is least, we report in the registration result.

The main focus of guided ICP is to perform horizontal registration between the BIM-obtained point cloud and the video-obtained point cloud. However, when the center matching is performed, the vertical alignment automatically gets registered as it also consists of vertical coordinate information.

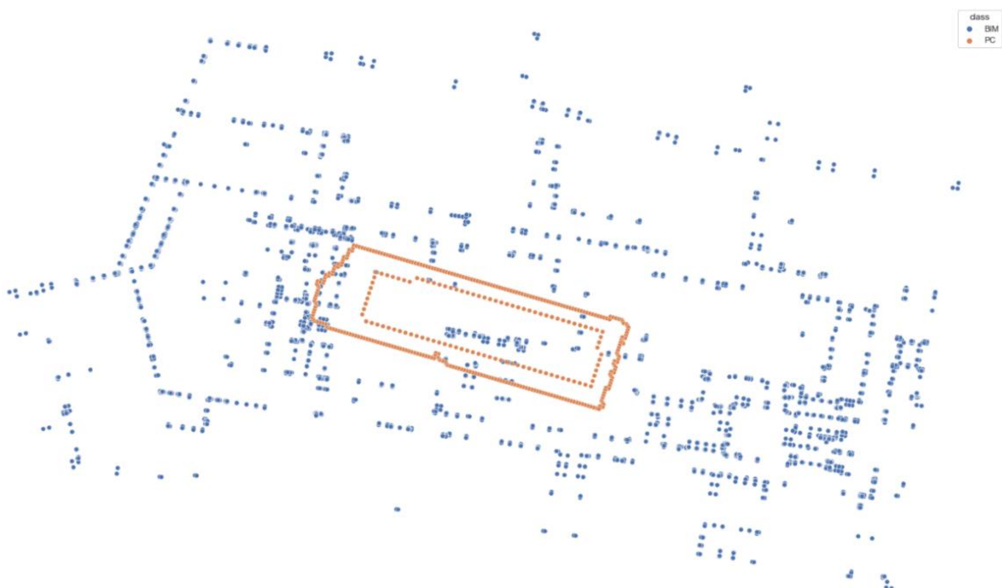
There are various implementations of ICP in Python, but the best easy-to-use ICP implementation comes in CloudCompare when running in the headless mode. It provides plenty of parameters to tweak for the best results, including the number of iterations, RMS difference, % overlap, etc.

An illustration example can help understand the whole process of Guided ICP. The automatic registration starts with the two-point clouds loaded in the same 3D space as shown in Figure 19 a), in all the figures, the reference point cloud, the BIM-obtained point cloud is shown in blue coloar and the query point cloud, the video obtained point cloud is shown in orange. First, the query point cloud is rotated such that both the point clouds have the same mean angle from their centroid (Figure 19 b). Then, the results maximum dimension of their bounding box is matched (Figure 19 c). Finally, the centers of both point clouds are matched as can be seen in Figure 19 d). The query point cloud is moved to the center of the reference point cloud.

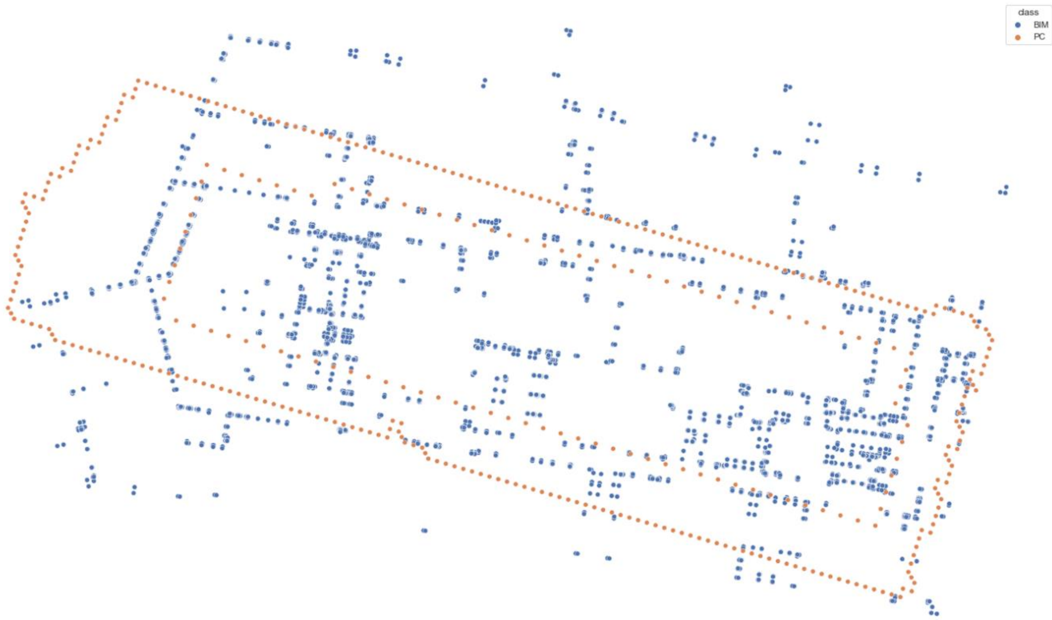
Now the ICP is executed iteratively with each iteration decreasing the scale of the query point cloud, and the RMS error is recorded in each iteration (Figure 19 e).

Figure 19: Steps of transformations applied to the query point cloud a) Original configuration, b) After scaling, c) orienting point clouds together, d) bringing their center together e) final configurations after ICP is executed.

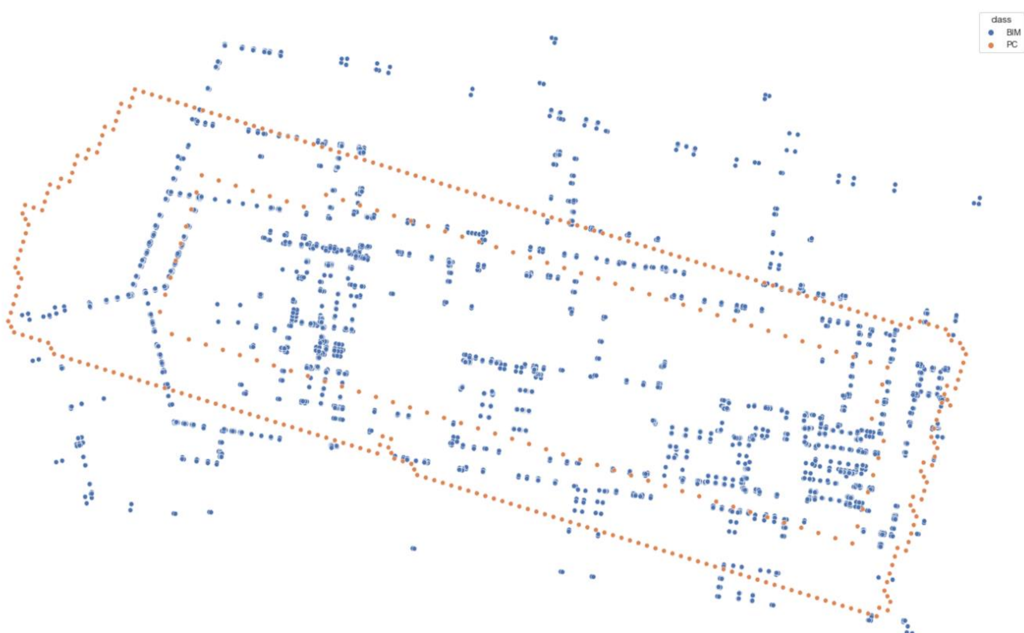
a)



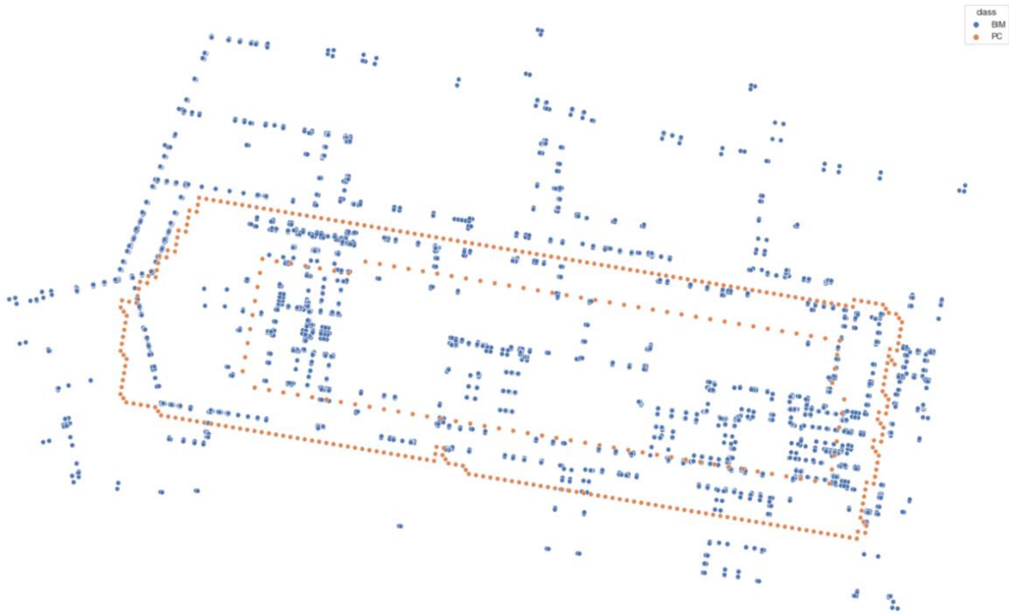
b)



c)



d)



e)

It can be inferred that the RMS error (RMSE) is used to assess the accuracy of the alignment. This calculation is performed by searching for the nearest point in the second point cloud for each point in the first point cloud. This way of RMSE calculation can work reasonably well for quantifying the fitness of the alignment. Still, the true method for assessing the alignment would be to identify and locate a few unique features common in both floor plans and find the distance between these common features after the ICP has been implemented. However, the SLAM algorithm used here, StellaVSLAM, cannot mark a set of specific ORB features as "special," thus precluding true assessment of the alignment in this project.

CHAPTER 4: EXPERIMENTAL SETUP AND RESULTS

In this chapter, we will discuss the equipment and testing setup, including the building used for testing, its characteristics, and its oddities. We will also talk about the characteristics of a building that influence the registration system's performance, looking at scenarios where the system works well and fails. We will also distinguish the differences in registration accuracy with manual and automatic cleaning of the query point cloud. The later part of the chapter will also go through possible reasons why the system works or fails to work.

4.1 Equipment setup

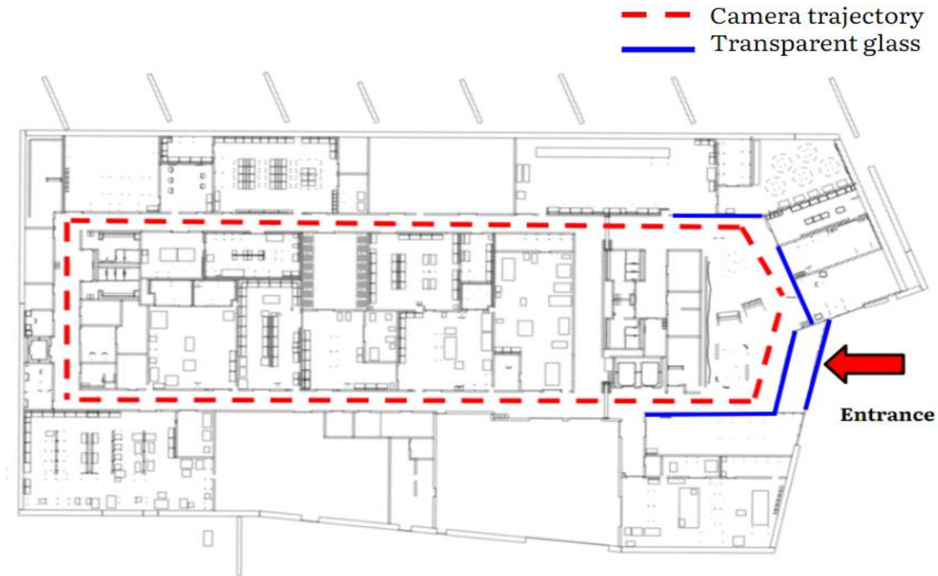
The omnidirectional camera used in this study is RICOH THETA S. All the computing work has been done on a desktop computer with an Intel Core i7 6800K processor and 64 GB of RAM running a dual boot Ubuntu 18.04 and Windows 10 as its operating system.

4.2 Testing setup

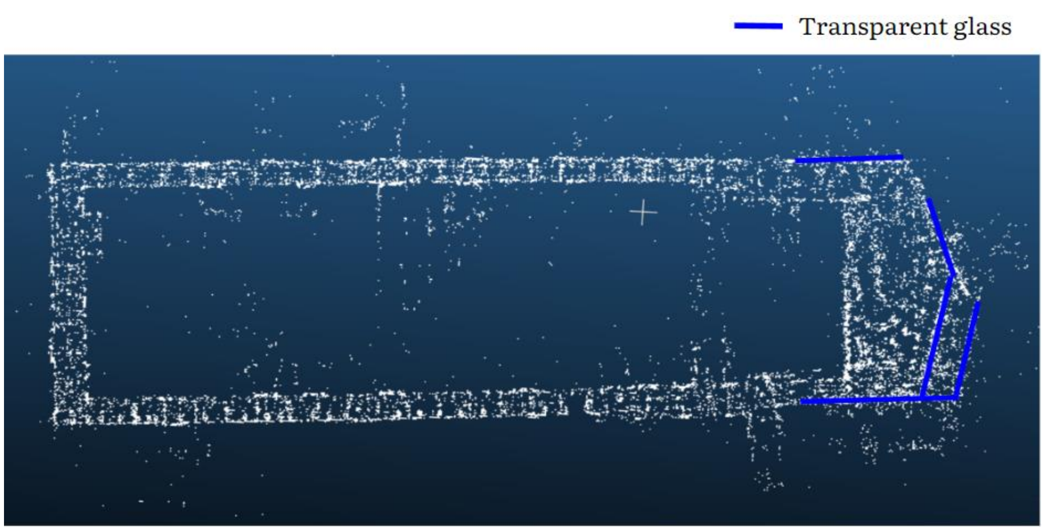
The registration system is tested on six datasets collected from two buildings (three floors from each building) on the North Carolina State University campus. The first building is the newly constructed Fitts Woolard Hall (FWH) at the Centennial campus, while the second is the Student Talley Center on the Main campus. For FWH, we have collected datasets from the first, third, and fourth floors. Each dataset has its own characteristics, which deserve its own section.

4.2.1 Fitts Woolard Hall

A) Fitts Woolard Hall: First floor



a)



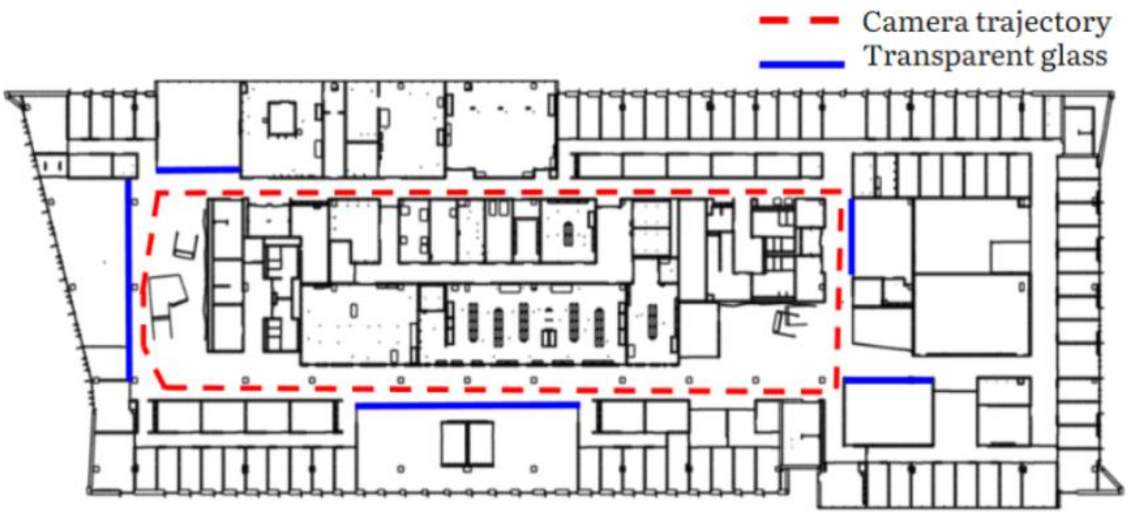
b)

Figure 20: a) First floor Fitts Woolard Hall with a trajectory for data collection (marked in red), transparent glass walls are marked in blue; b) point cloud obtained on the first floor by StellaVSLAM.

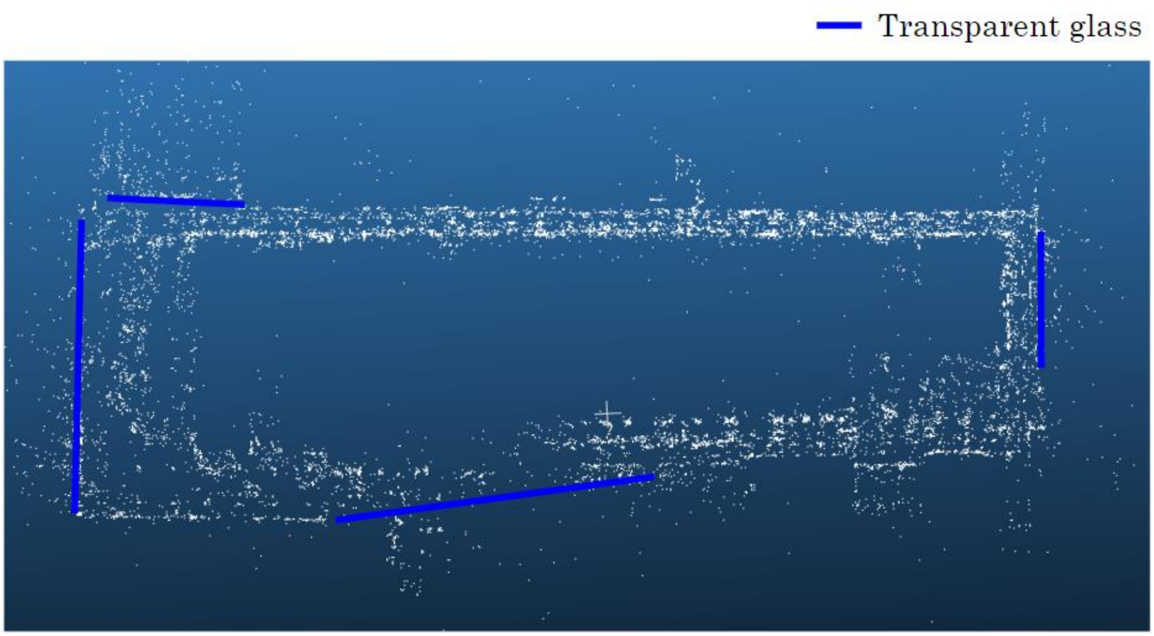
The first floor of FWH consists of long straight hallways along its length with rooms/laboratories on either side. Most rooms have opaque walls made from concrete. Additionally, none of the rooms have any windows restricting visibility of the inside of these labs. Since this is the first floor, it has the main entrance to the building, and a few rooms near it have transparent glass walls making their insides visible from the hall (marked blue in Figure 20 a)). Also, there is a set of stairs near the entrance. The hallway closely resembles the shape of an irregular pentagon. The trajectory followed for data collection is highlighted in red dashed lines in Figure 20 a).

The point cloud generated by the StellaVSLAM is shown in Figure 20 b). The point cloud is noisier towards the entrance due to transparent objects. Since StellaVSLAM is a visual SLAM algorithm, objects which are transparent to visible light inhibit the accuracy of their corresponding location in the point cloud. It is further supported by a much well-defined (comparatively) point cloud farther from the entrance, where most of the walls are opaque.

B) **Fitts Woolard Hall: Third floor**



a)



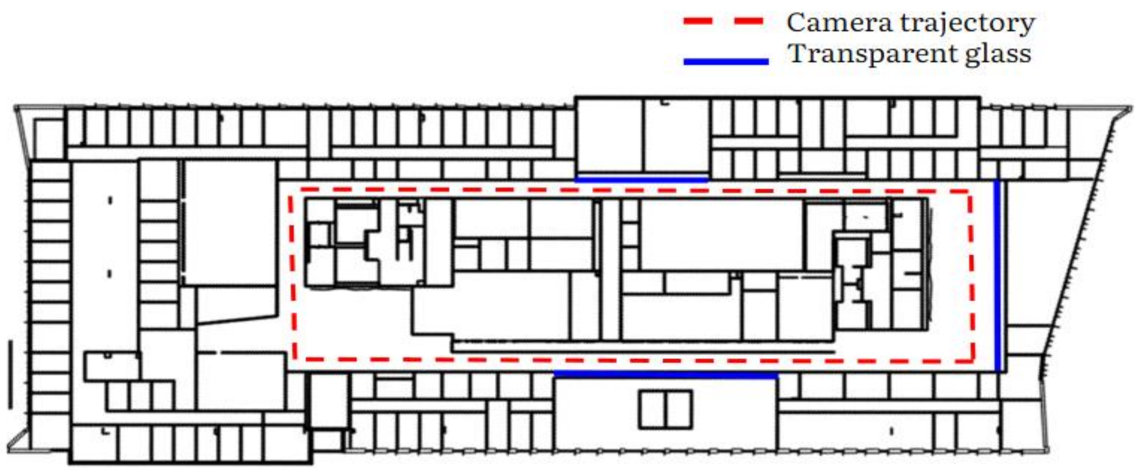
b)

Figure 21: a) Floor plan of Fitts Woolard Hall, third floor, b) Obtained point cloud of Fitts Woolard Hall, third floor.

The third floor has offices, office spaces, laboratories, classrooms, and conference rooms. Similar to the first floor, it has a main hallway with rooms on either side. However, unlike the first floor, there is a fair amount of branching out from the main corridor into "mini" hallways. The main hallways are predominantly opaque with concrete walls. However, the portion of glass walls is significantly higher than that on the first floor, with around one-third of the total area covered with glass. The solid blue lines in Figure 21 a) represent the transparent glass walls visible from the main hallway. Concrete walled sections are relatively plain, providing a limited number of candidates for the ORB features.

Because of the higher portion of glass walls and relatively plain wall sections, the point cloud from the third floor is significantly noisier than from the first floor. This is especially pronounced in the sections with glass walls. In addition, the loop closure is not achieved as well as the first floor because of increased transparent objects on this floor.

C) Fitts Woolard Hall: Fourth floor



a)

b)

Figure 22: a) Floor plan of fourth floor Talley b) Obtained point cloud of Fitts Woolard Hall, fourth floor.

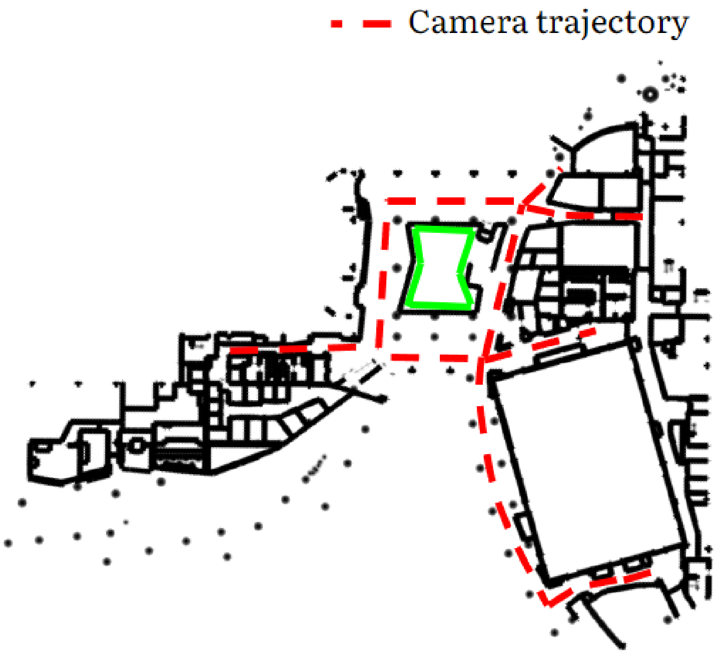
The fourth floor is very similar to the third floor. It has many offices, office spaces, study rooms, and classrooms. The main hallway is still running along the building's length, with rooms on either side with a few "mini" corridors branching out. The proportion of opaque concrete and transparent glass walls (marked blue in Figure 22) in a walkthrough of the main hallway is still the same, with around two-thirds concrete and one-third clear glass.

As expected, the point cloud is much noisier than the first floor, with more inaccuracies in the sections with higher glass objects.

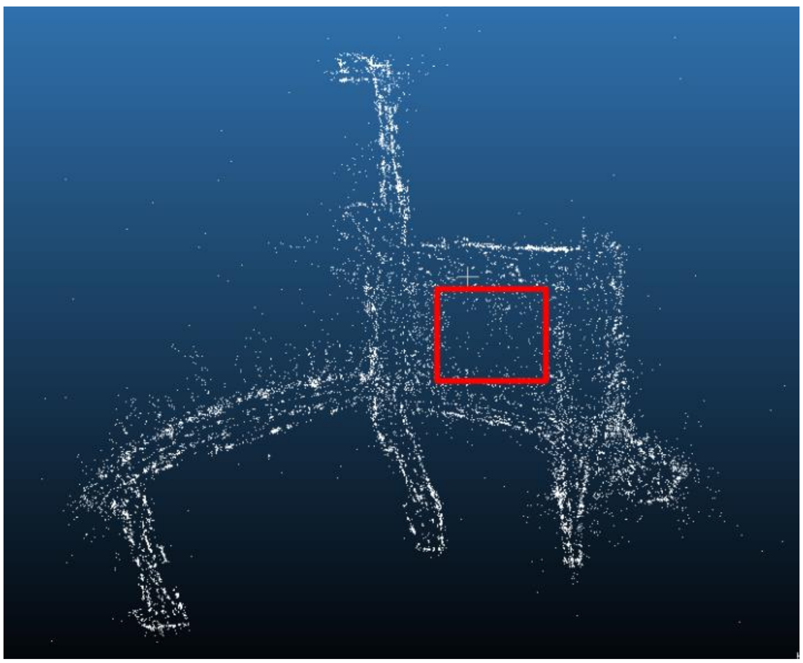
4.2.2 Student Talley Center

Student Talley Center is quite a different building than the Fitts Woolard Hall. It has many open spaces, big halls, curved hallways, offices, classrooms, and study areas. Its third, fourth, and fifth floors were scanned in this study. In Student Talley Center, referred to as Talley for the rest of this thesis, a significant section of the second floor is visible from the higher floors. Hence, ORB features of second floors get included in the point clouds of higher floors, resulting in a noisier point cloud. Figure 23 a) shows the floor plan obtained from the BIM.

A) **Student Talley Center: Third floor**



a)

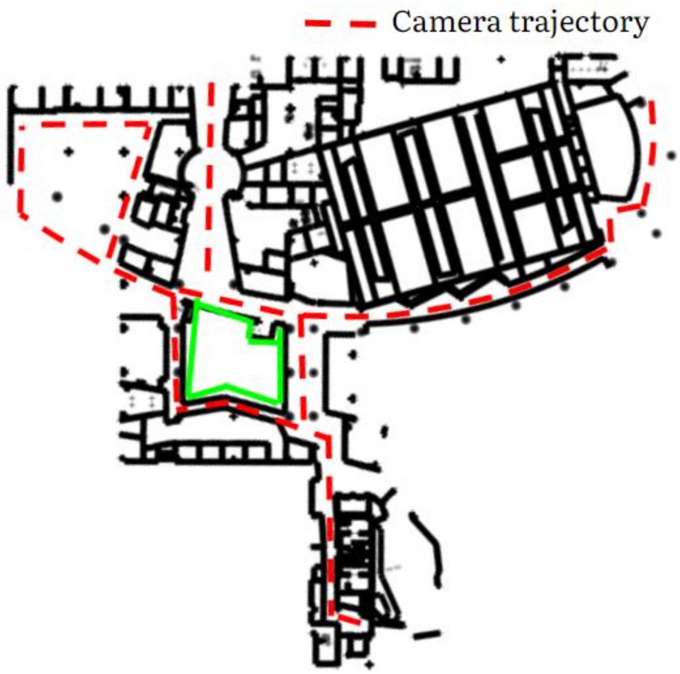


b)

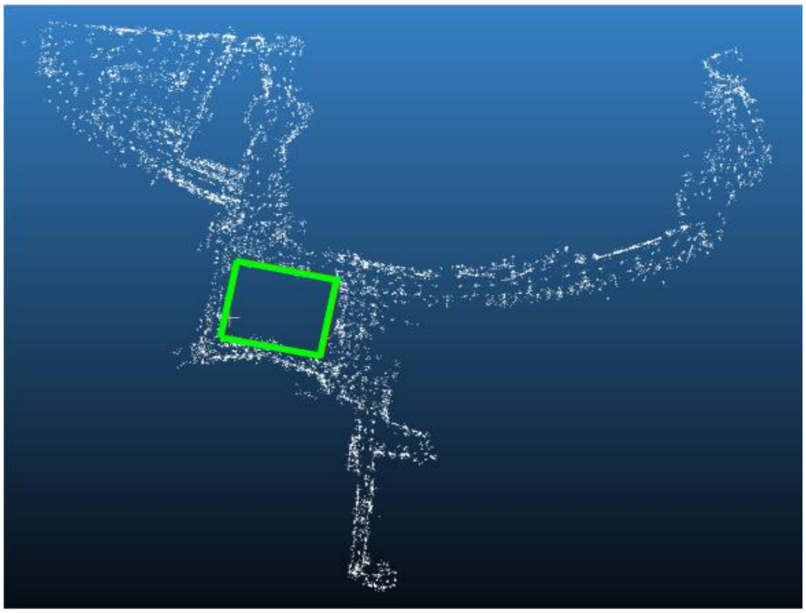
Figure 23: a) Talley third floor plan, the red box is the area where the second floor is visible, b) Point cloud obtained from StellaVSLAM on the third floor Talley.

The third floor of Talley consists of a few classrooms, a hall, a corridor of offices, and some open space. The second floor is quite visible towards the center of the floor. Almost all walls are opaque concrete, resulting in well-defined point clouds (see red box in Figure 23 b).

B) Student Talley Center: Fourth floor



a)

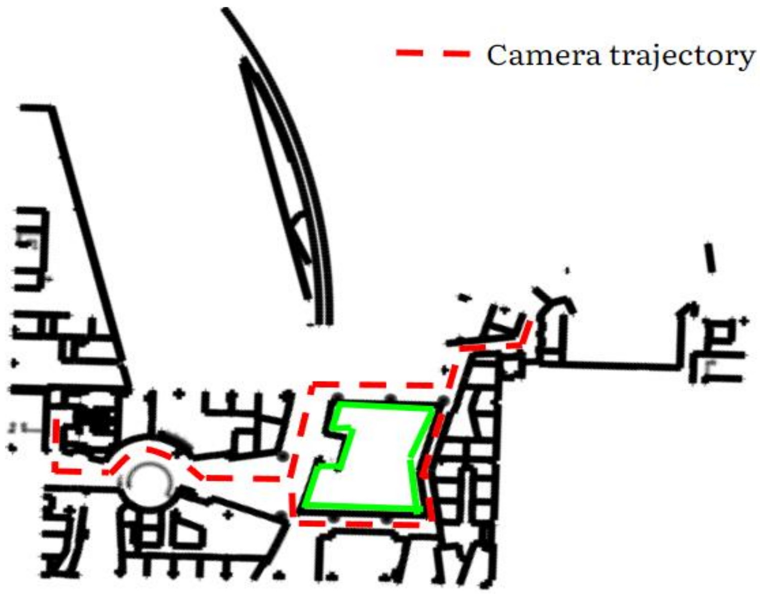


b)

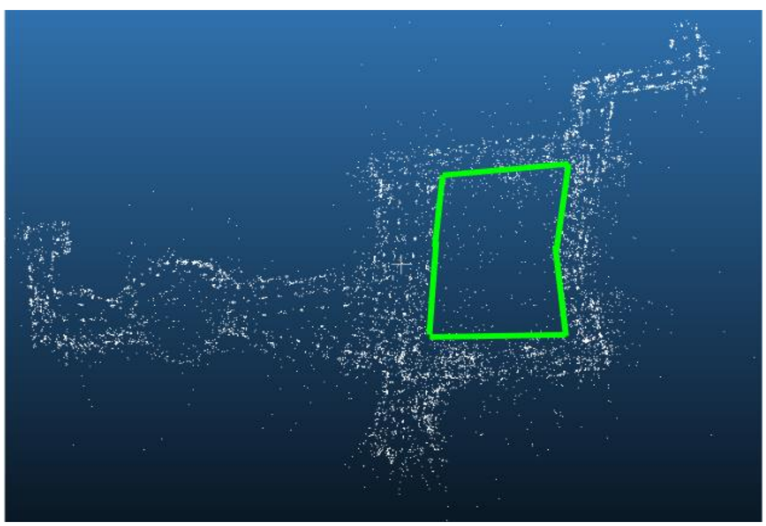
Figure 24: a) Floor plan obtained from BIM. The area where the second floor is visible is marked in red b) Obtained point cloud on Talley, fourth floor.

The fourth floor is very similar to the third floor. It primarily consists of office spaces, classrooms, and an open area. However, unlike the third floor, there is no hall on this floor. Instead of the hall, there is a big-open study area with no wall in-between. Nevertheless, the second floor is still quite visible in an omnidirectional record in its main corridor (marked by a green box in Figure 24 a) b). Because of the predominant opaque walls and feature-rich environment, the point cloud generated on this floor is well-defined.

C) Student Talley Center: Fifth floor



a)



b)

Figure 25: a) Floor plan obtained from BIM section cut b) Obtained point cloud on Talley, fifth floor. The area where the second floor and its ORB features are visible is marked by a green box.

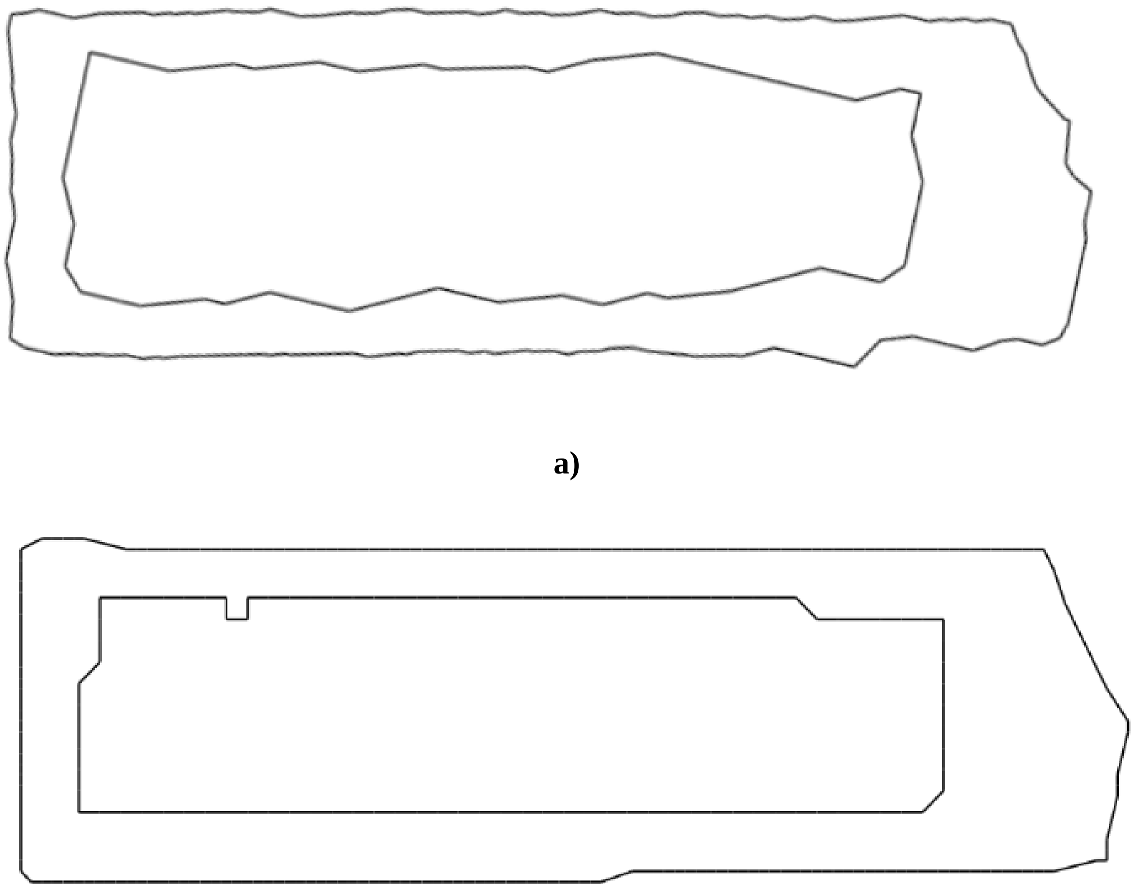
The fifth floor is smaller than the third and fourth floors, but the arrangement is similar: a hallway with offices on either side, an open space area, and a few classrooms. Again, like the other two floors, the point cloud generated on this floor is well-defined due to the feature-rich environment with relatively fewer transparent objects to interact with.

4.3 Results and Analysis

To generate an approximate floor plan from a point cloud, you must first obtain the point clouds from the video. Once you have the point clouds, you can manually or automatically clean the data. After cleaning the data, you can draw an outer and inner boundary (one of the alpha shapes). This will generate an approximate floor plan out of a point cloud.

As mentioned in the previous chapter, the generated point cloud is cleaned manually or algorithmically on each dataset. A boundary is drawn on the clean point cloud to generate an approximate plan of the main hallway on which ICP-based point cloud registration is applied. In this section, we will look into the registration accuracy of each dataset and compare it with the actual registration. Furthermore, manual and algorithmic cleaning results are also compared and analyzed. For brevity, an algorithmically cleaned query point cloud dataset is referred to as an algorithmic dataset or algorithmic point cloud, and their manual counterparts are referred to as a manual dataset or manual point cloud from here on.

4.3.1 Fitts Woolard Hall: First floor



a)

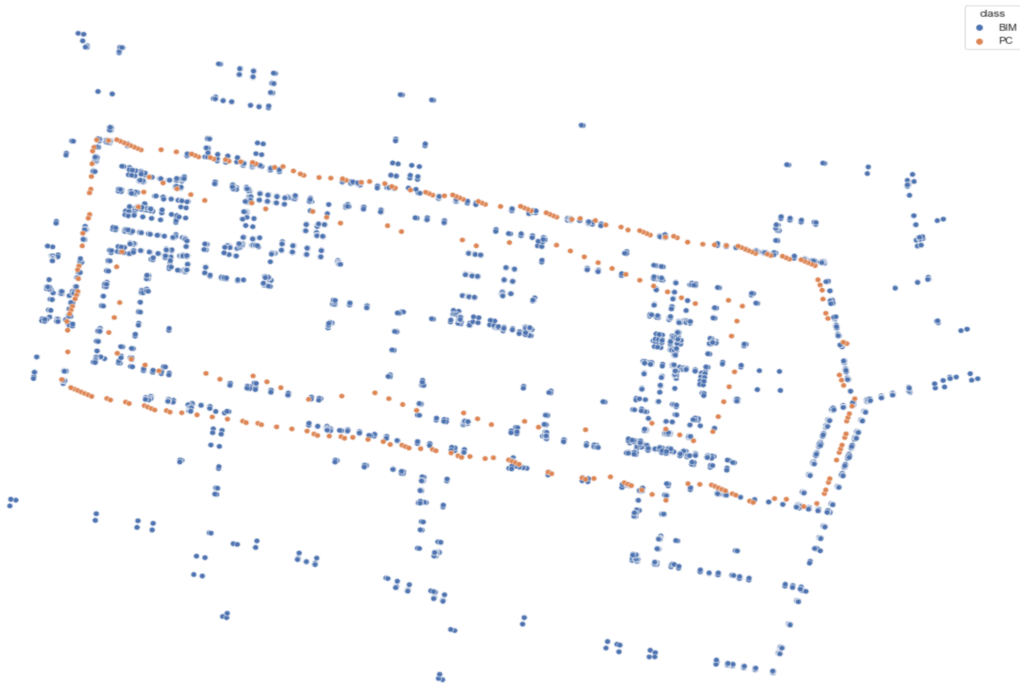
b)

Figure 26: Hallway floor plans generated from a) automatically, and b) manually cleaned point cloud.

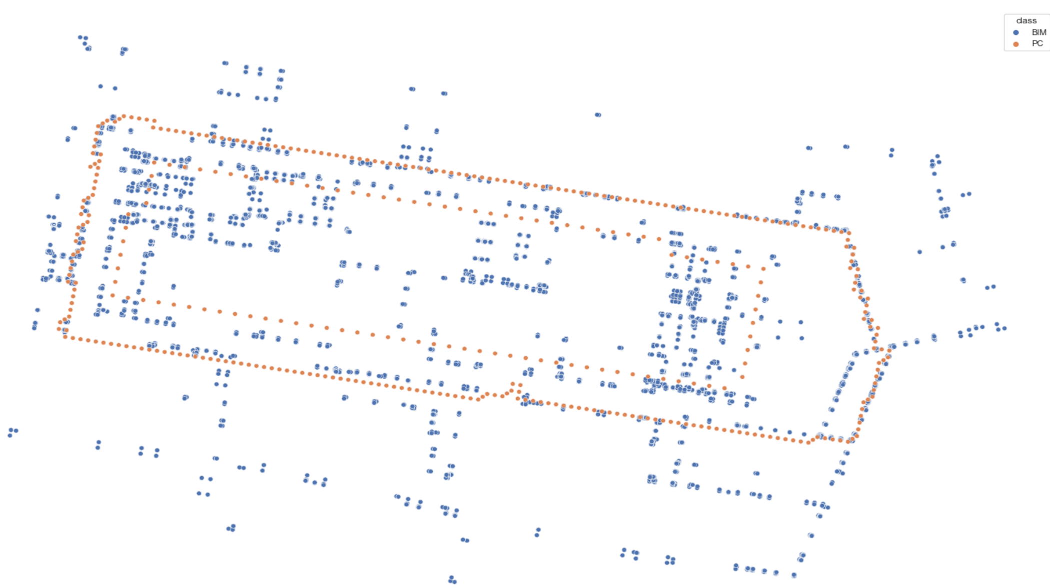
Figure 26 a) shows the hallway on the floor plan obtained from the algorithmic dataset, while b) shows the hallway of the manual dataset. Results of ICP obtained on algorithmically cleaned point cloud yield a mean distance of 0.527798 meters between the query point cloud and the reference point cloud as shown in Figure 27 a). The maximum distance between the two point clouds is around 1.96252 meters. Compared with true registration, the deviation has been quantified in Table 1 and visualized in Figure 28 a). On average, the resultant registration is about 0.388458 meters, deviating from a standard deviation of 0.254902 meters. The maximum deviation is around 0.877186 meters, and the minimum is 0 meters.

Similarly, on average, the final configuration is around 0.568894 meters away for manually cleaned query point clouds shown in Figure 27 b). The maximum distance between the query and reference point cloud is 2.775420 meters, and the minimum is 0. Comparing registration results from the developed system with manual registration, we obtained a mean deviation of around 0.141422 meters, a maximum deviation of 0.452354 meters, and a standard deviation of 0.052926 meters. The deviation is visualized in Figure 28 b), and the values are shown in Table 1.

The results represent the accuracy of the automatic registration algorithm is independent of whether the query cloud is obtained from an algorithmic cleaning or a manual one.

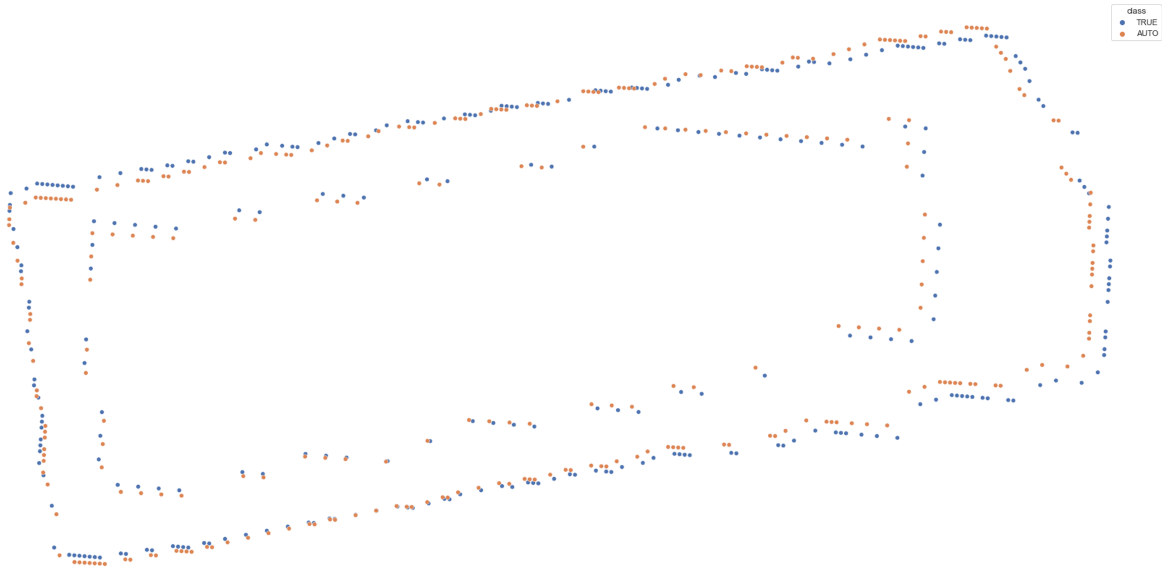


a)

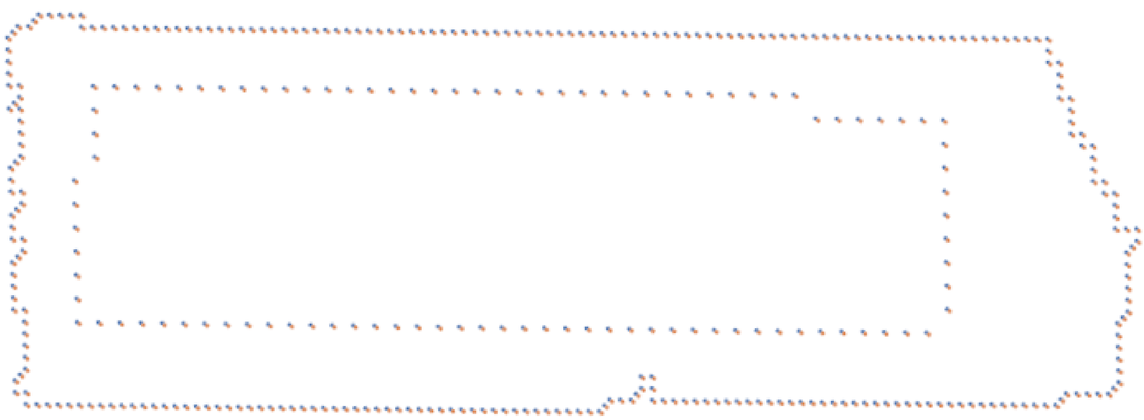


b)

Figure 27: Registration results where blue points represent BIM and orange points represent floor plan point cloud of a) algorithmic point cloud and b) manual point cloud.



a)



b)

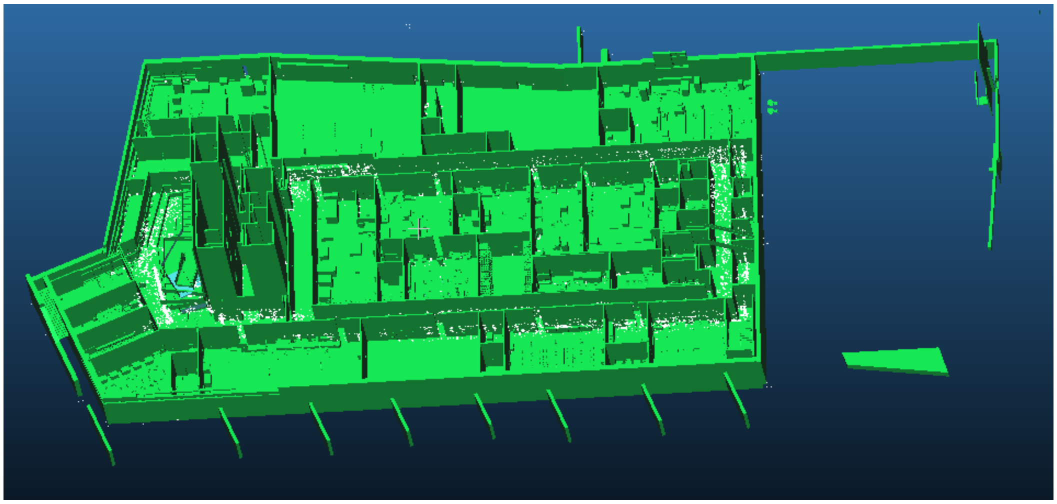
Figure 28: Deviation of ICP results visualized a) Algorithmically cleaned (orange) vs True (blue) b) Manually cleaned (orange) vs True (blue).

Table 1: Deviations from true registration for FWH first floor.

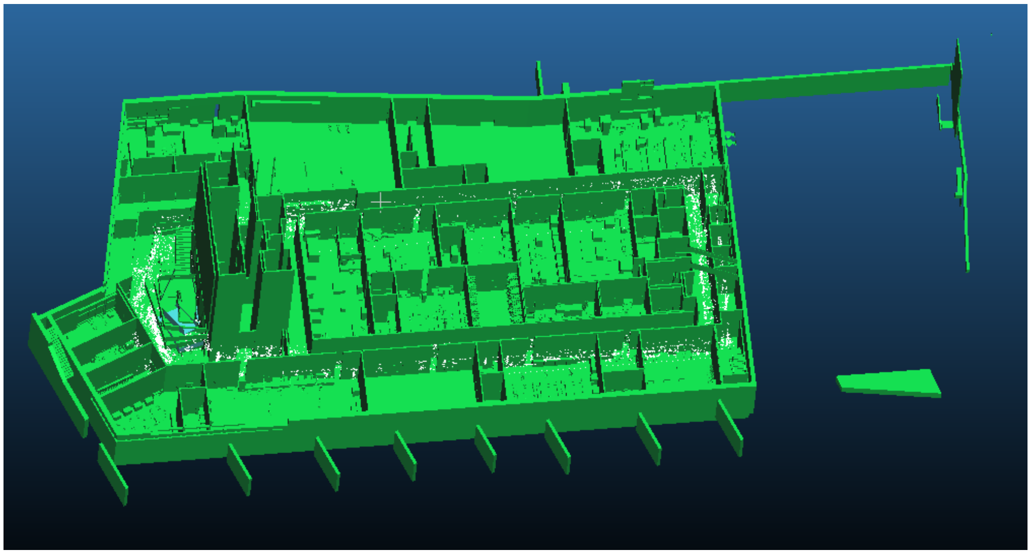
Statistics of deviation from true registration	**Automatically cleaned (meters)**	**Manually cleaned (meters)**
Mean	0.388458	0.141422
Minimum	0	0
Maximum	0.877186	0.452354
Standard deviation	0.254902	0.052926

As can be observed from the results, as shown in Table 1 and Figure 28, a simple ICP works well for registration in this case. The resultant configuration, especially for the case of the manually cleaned point cloud, yields a reasonably accurate fit after registration.

The variability of deviation from the true registration also remains quite low. Guided ICP also takes care of vertical registration between the BIM and the video obtained point cloud. The center matching step in guided ICP brings the two point clouds to the same vertical level, resulting in quite accurate vertical registration. Figures 29 a) and b) show the final configuration after registration. Both vertical and horizontal registration has been achieved at a reasonable accuracy.



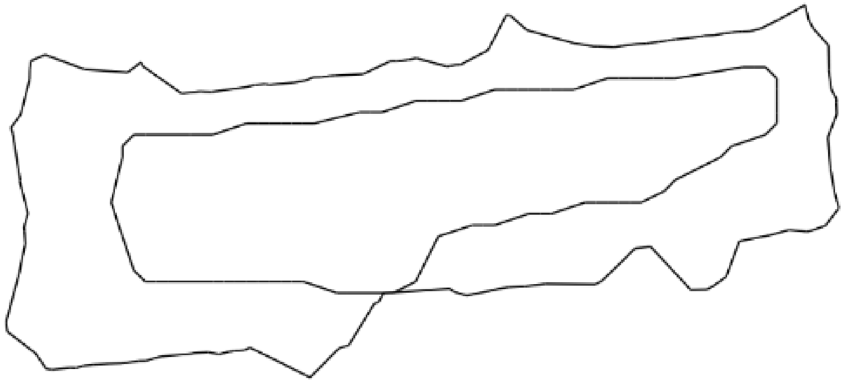
a)



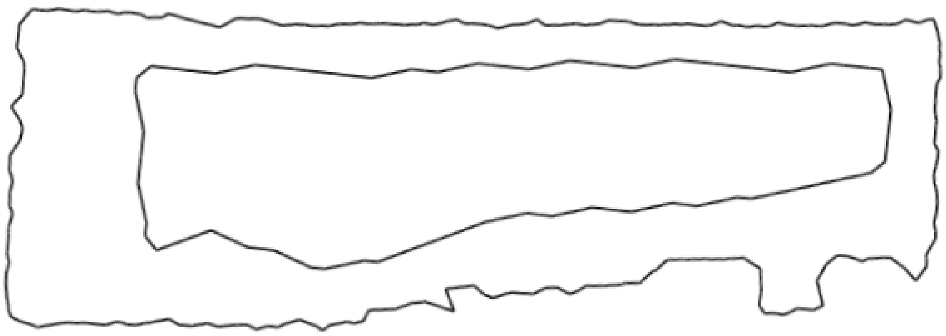
b)

Figure 29: Final registration configuration for FWH first floor.

4.3.2 Fitts Woolard Hall: Third floor



a)



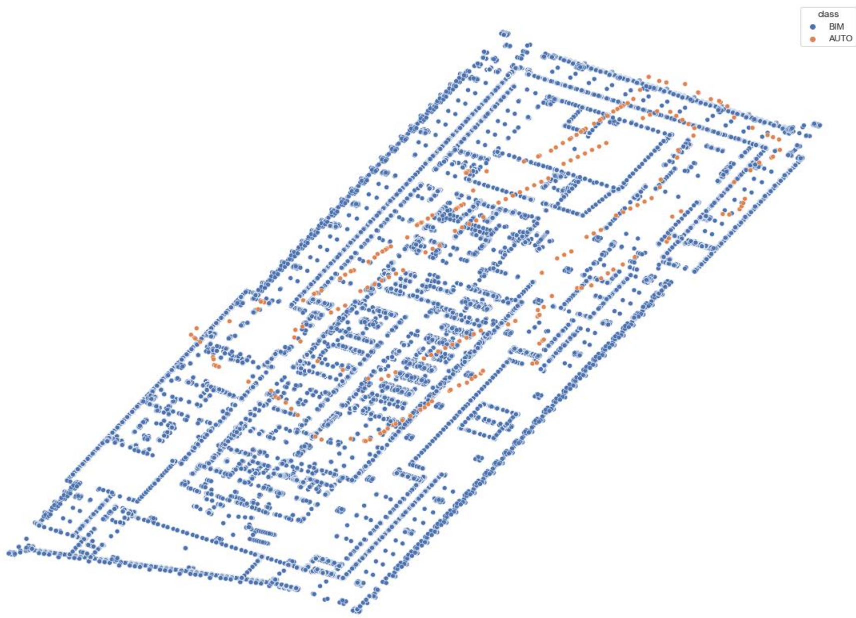
b)

Figure 30: Hallways on the floor plan obtained from a) algorithmic point cloud and b) manual point cloud.

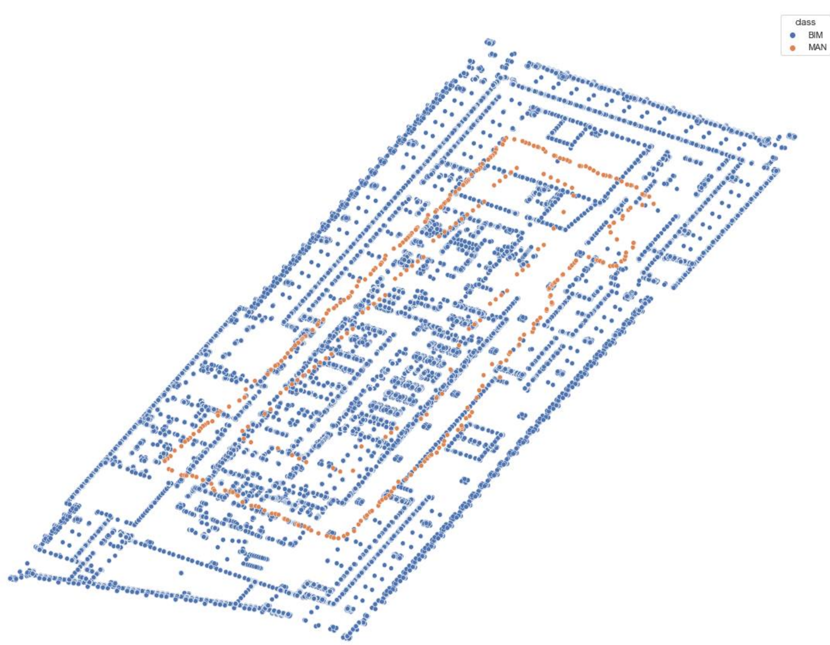
The floor plans obtained from the query point cloud for the third floor of the FWH building are not clean compared to the first floor of the same building as visualized in Figure 30. However, the results of automatic registration are similar. For the algorithmic dataset, the mean distance between the BIM and the point cloud is around 0.2752 and a standard deviation of 0.1888 meters. Figure 31 a) shows the final configuration after automatic registration.

Similarly, the automatic registration on the manual dataset yields a mean distance of 0.2665 meters and a standard deviation of 0.2118 meters, as visualized in Figure 31 b).

The final configuration of both point clouds tested their true registration. The algorithmic point cloud is off by 9.5710 meters on average, with a maximum and minimum deviation of around 11.1156 meters and 8.1238 meters, respectively. Similarly, for manual point cloud, the mean deviation is 4.3443 meters, with a maximum and minimum of 4.8636 and 3.8878 meters.



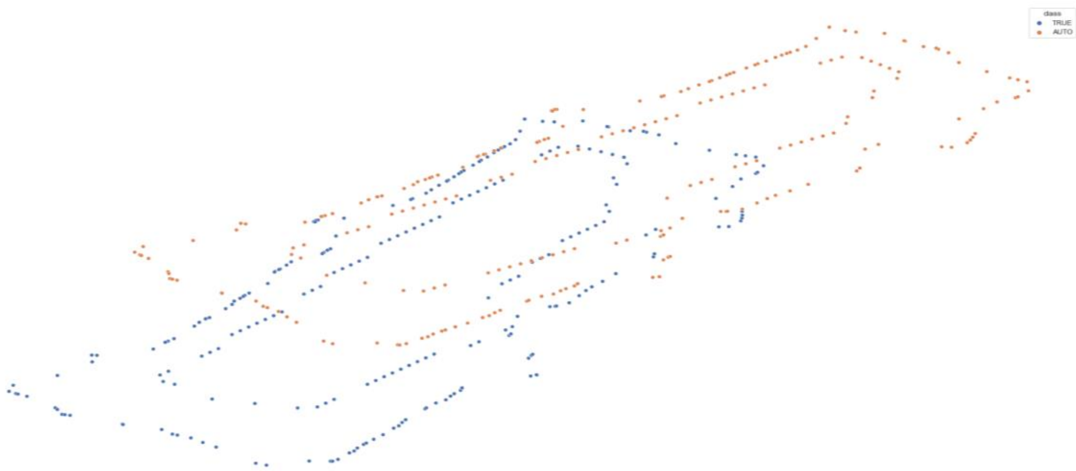
a)



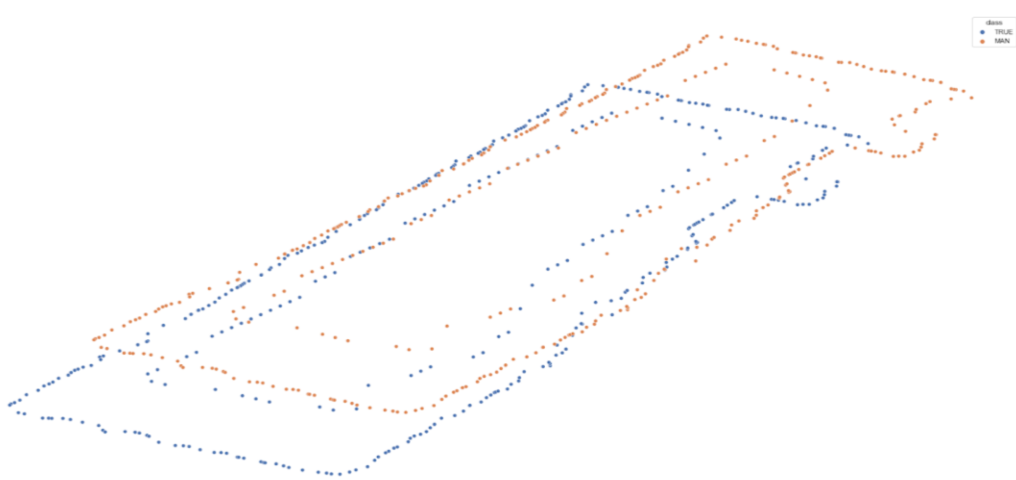
b)

Figure 31: Registration results of a) algorithmic point cloud b) manual point cloud where blue points represent BIM and orange points represent floor plan point cloud.

The deviations have been visualized in Figure 32 a) and b), and the deviation statistics are recorded in Table 2. Unlike the first floor, the deviation of the result of automatic registration is primarily translational, the orientation and scale parameters are fairly close to that of the true registration. This translational offset is also reflected by a relatively slim deviation spread observed by its standard deviation.



a)



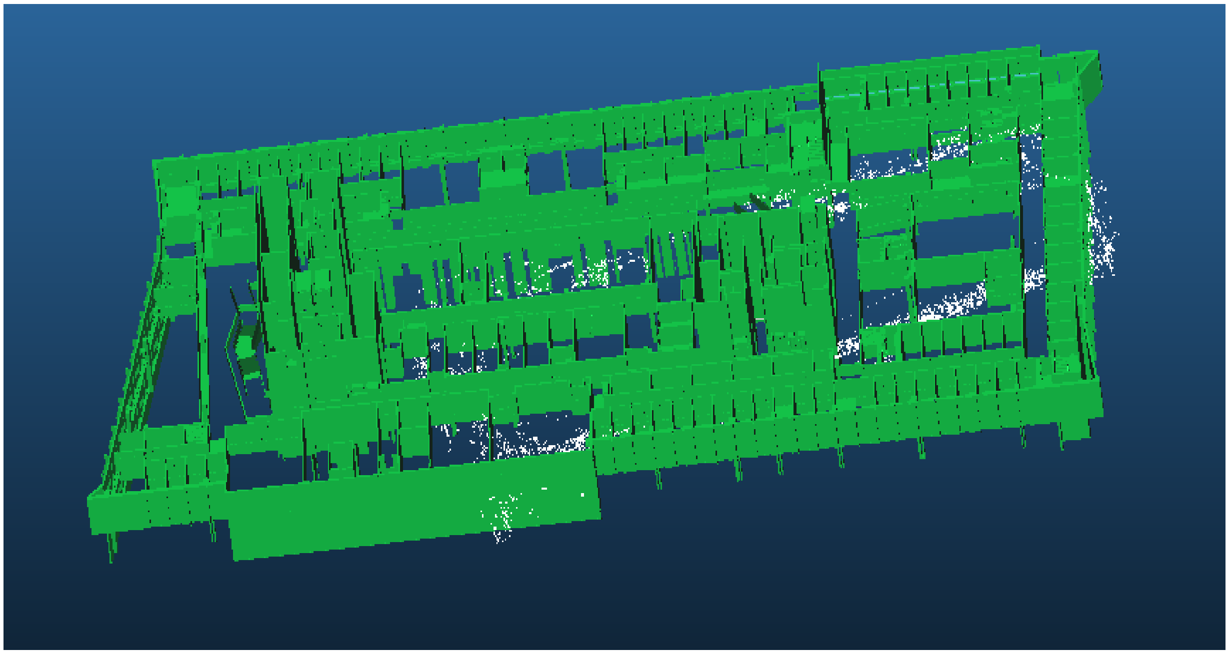
b)

Figure 32: The deviation of automatic registration (orange) with true registration (blue) on a) algorithmic b) manual point cloud.

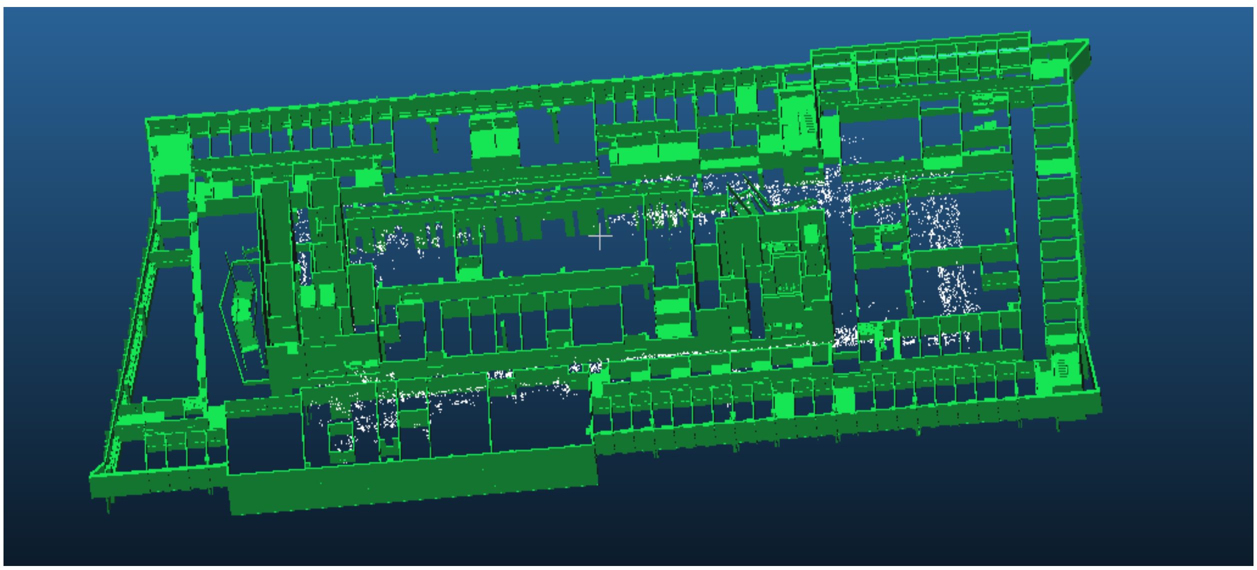
The vertical registration resulting from performing center matching has worked quite well. Figure 33 a) and b) show the vertical position of the point cloud after registration for both algorithmic query point clouds and the manual query point clouds. The point cloud obtained from the panoramic video is in white and the BIM is in green.

Table 2: Deviations from true registration for FWH third floor.

Statistics of deviation from true registration	**Algorithmically cleaned**	**Manually cleaned**
Mean	9.571069793	4.344359121
Minimum	8.123878272	3.887818358
Maximum	11.11560239	4.863601153
Standard deviation	0.763002317	0.253657325



a)

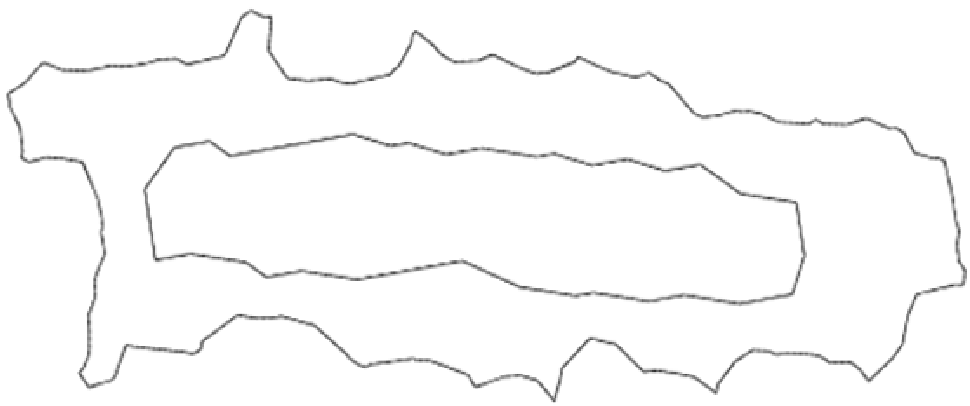


b)

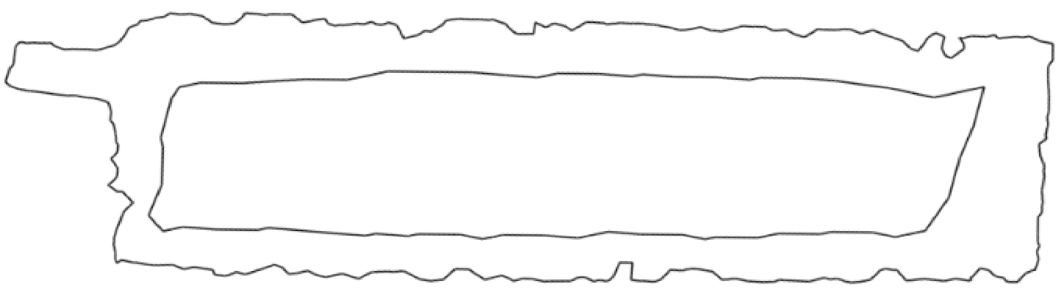
Figure 33: Final registration configuration for FWH third floor a) algorithmic b) manual point cloud.

Overall, the automatic registration algorithm does not work as well as it worked for the first floor FWH. The main reason behind poor registration can be attributed to the much noisier point cloud of the FWH third floor, resulting in a poor approximation of the floor plan from the video. Also, the difference in the accuracy of point cloud performance is much bigger than that in the FWH first-floor dataset.

4.3.3 Fitts Woolard Hall: Fourth floor



a)

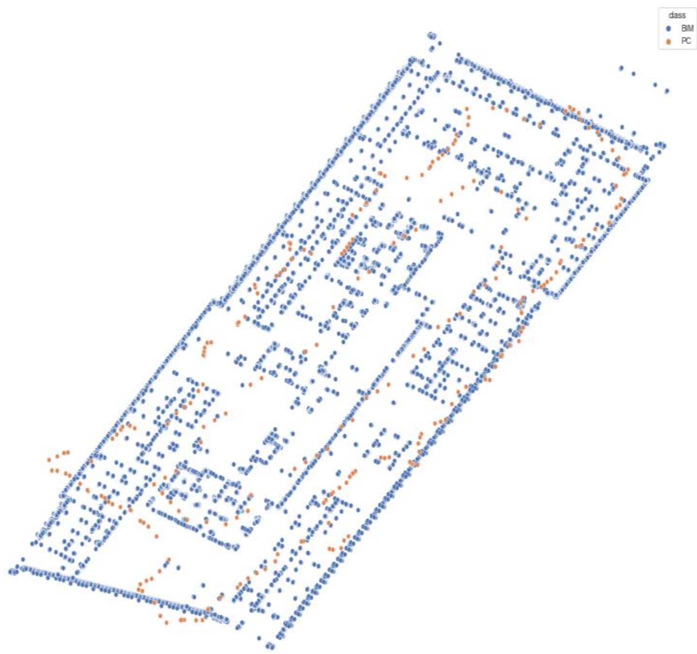


b)

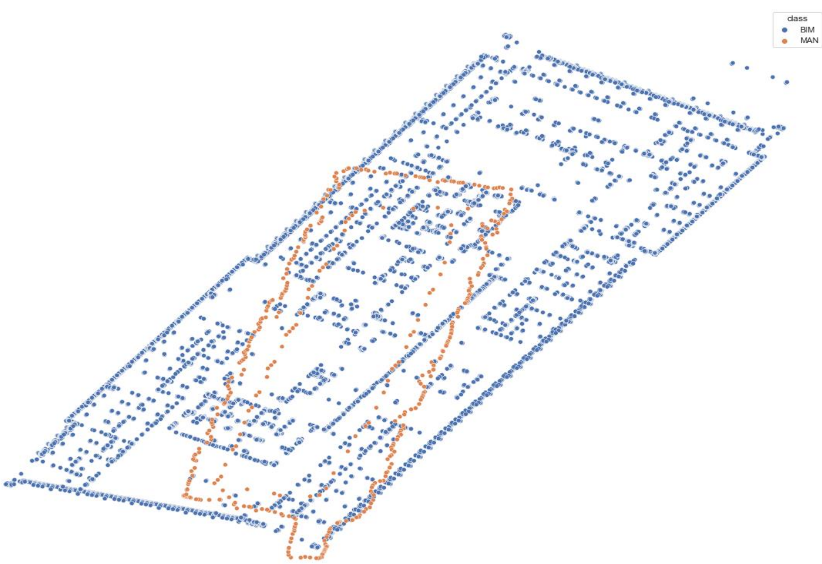
Figure 34: Hallways of FWH fourth floor obtained from a) algorithmic point cloud b) manual point cloud.

For the fourth floor of the FWH building, the approximate floor plans obtained from the point clouds (as shown in Figure 34 a) and b) are not as well-defined as compared to those of the first floor, especially in the case of algorithmic cleaning (Figure 34 a). This is attributed to a relatively higher amount of transparent glass objects and a noisier point cloud. Running automatic registration for algorithmic point cloud yields a mean distance of 1.14937 meters and a standard deviation of 1.0532 meters. The final configuration and fitting are shown in Figure 35 a). Similarly, the mean distance for the manual dataset is around 1.31667, with a standard deviation of 0.9541.

When the results are compared with the true registration, the first dataset's mean deviation is very large compared to the previous two datasets at around 59.7125 meters. The standard, minimum, and maximum deviations are also large at 32.9330 meters, 1.076 meters, and 123.9304 meters, respectively. This is because the automatic registration yields a 180-degree flipped configuration than the ground truth. Because of the primarily rectangular symmetry of the hallways, the code finds the lowest RMS distance at a flipped configuration. The flip is quite evident in Figure 36 a).



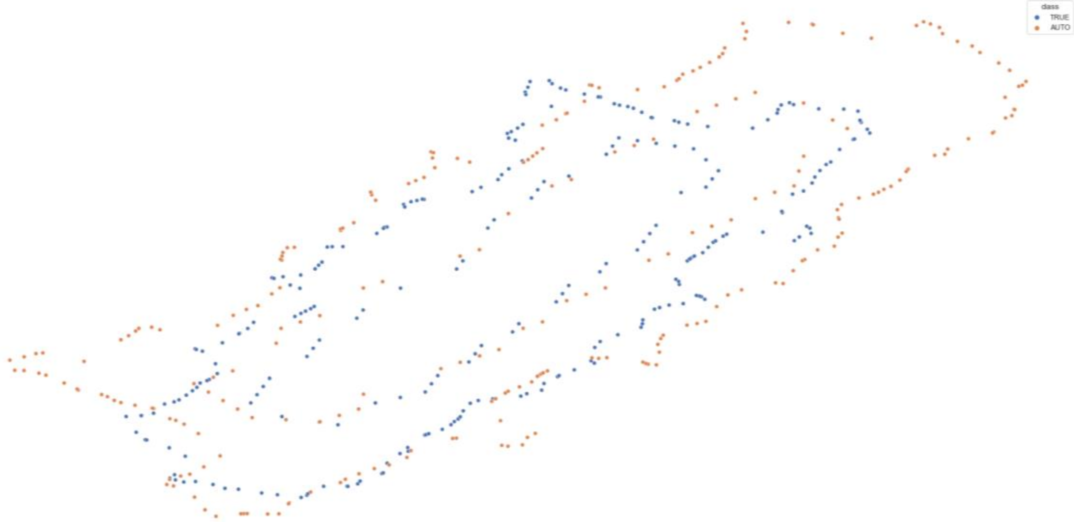
a)



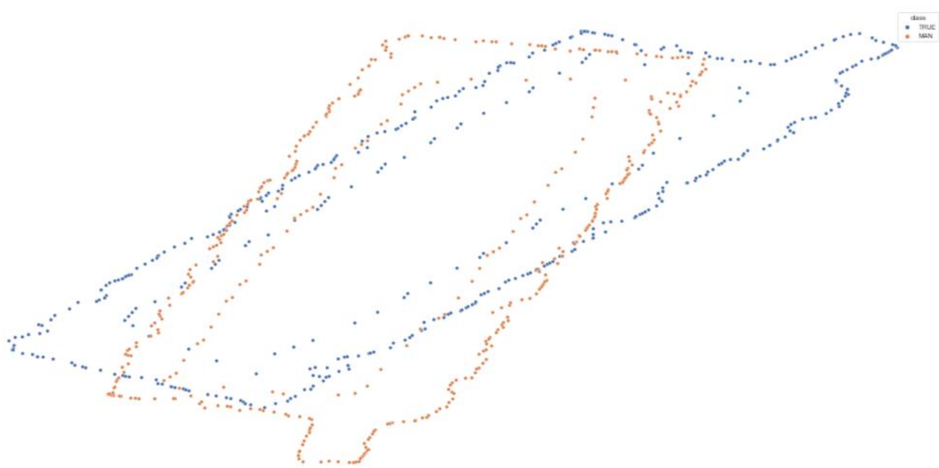
b)

Figure 35: Final configuration after automatic registration a) algorithmic point cloud b) manual point cloud where blue points represent BIM and orange points represent floor plan point cloud.

This is also observed when the developed code is executed on the manual dataset (Figure 36 b). With a mean deviation of 49.6144 meters, a maximum deviation of 92.2697 meters, and a standard deviation of 23.7368 meters, it is obvious that the resultant configuration is flipped. Figure 36 b) very clearly visualizes this flip. Also, the deviation statistics are shown in Table 3.



a)



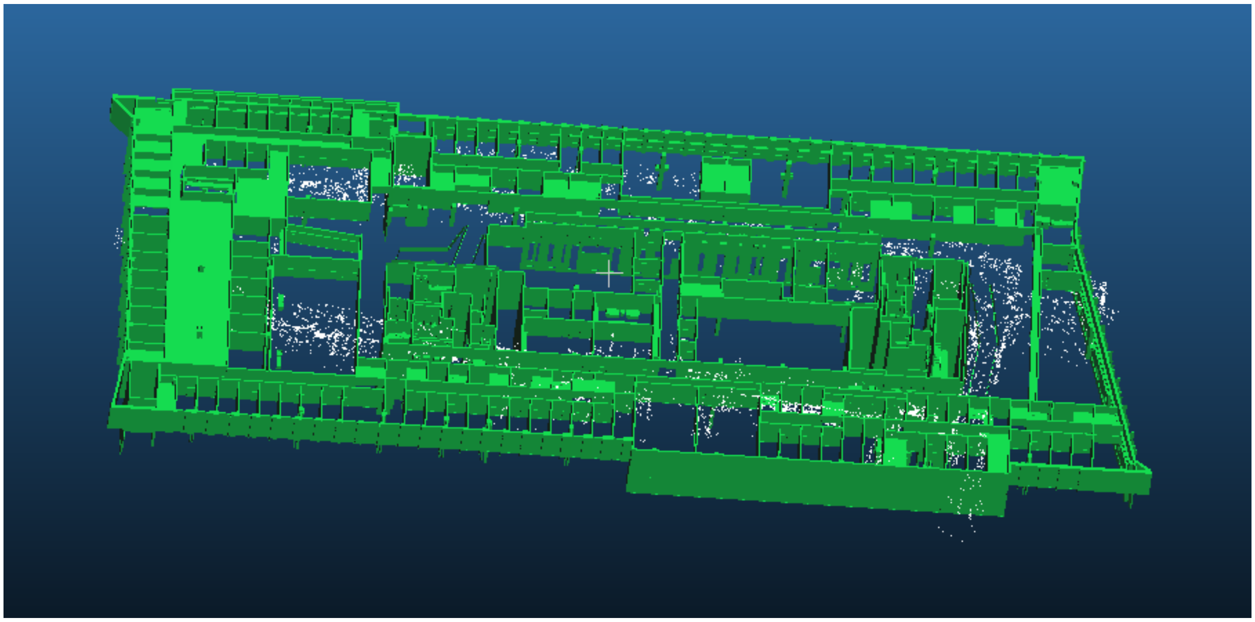
b)

Figure 36: The deviation from true registration is visualized for each dataset a) algorithmic and b) manual.

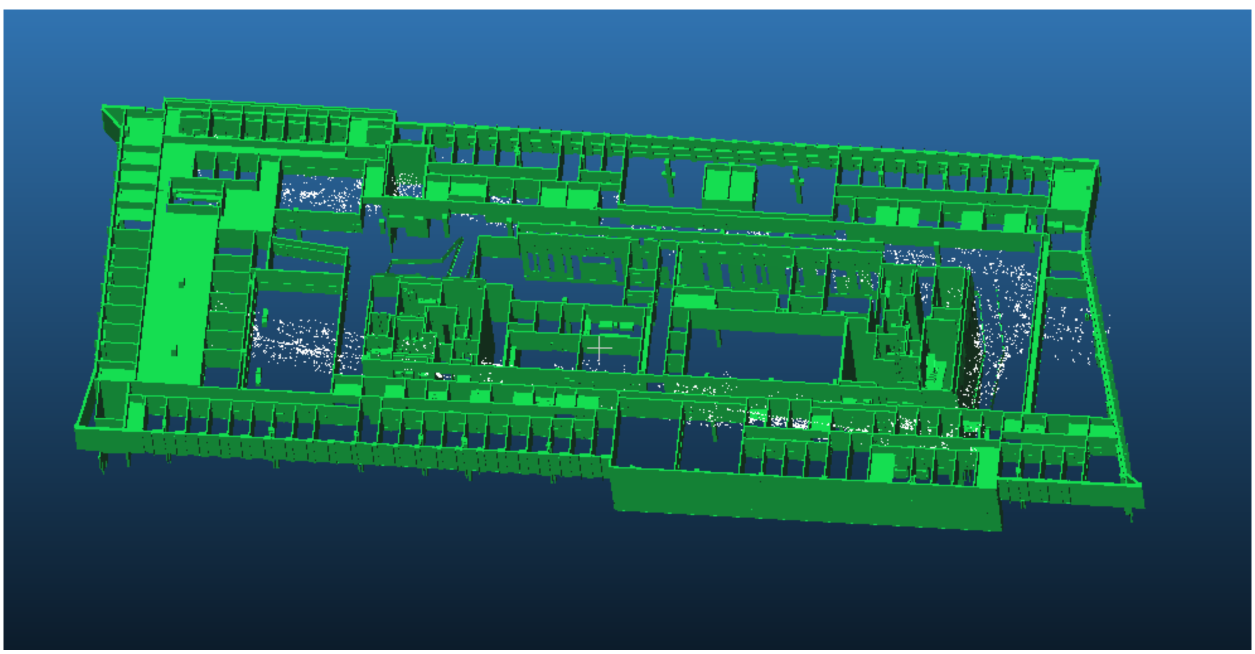
Even though the horizontal registration is flipped, the vertical registration of FWH's fourth floor with the developed automatic registration works well. Figure 37 a) and b) shows the vertical registration for the algorithmic point cloud and the manual point cloud.

Table 3: Deviations from true registration for FWH fourth floor.

Statistics of deviation from true registration	**Algorithmically cleaned**	**Manually cleaned**
Mean	59.712535	49.61444486
Minimum	1.076029	10.11336826
Maximum	123.930489	92.2697007
Standard deviation	32.933046	23.73683388



a)

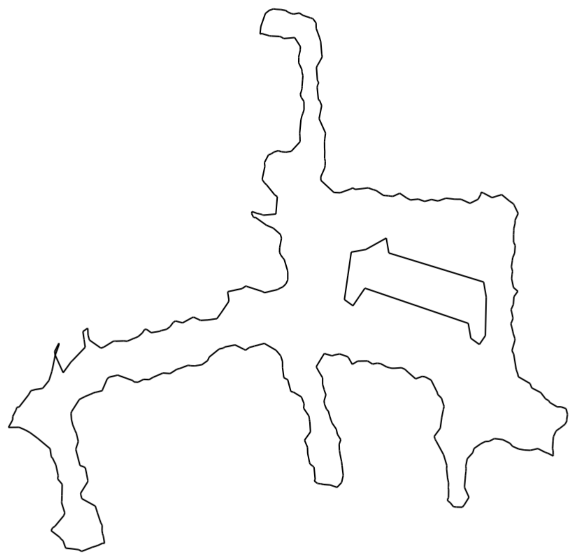


b)

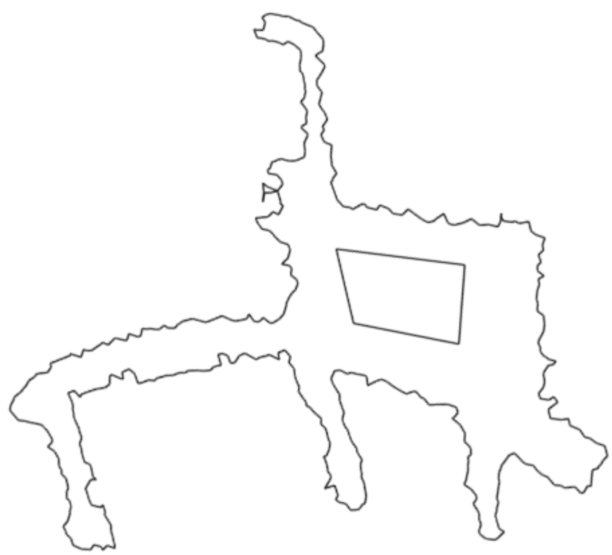
Figure 37: Final registration configuration at FWH fourth floor.

Out of all the datasets, automatic registration performs the worst here. It completely misses out on true registration and outputs a flipped configuration. These results highlight the problem with using RMS distance to measure registration accuracy.

4.3.4 Student Talley Center: Third floor



a)



b)

Figure 38: The floor plans obtained from the query point cloud obtained after a) algorithmic cleaning and b) manual cleaning.

Since the point cloud obtained for Talley is more well-defined than FWH, the floor plan generated by creating alpha shapes around a clean point cloud will more closely resemble the actual floor plan obtained from the BIM (thus, it will be called reference point cloud). This effect is more pronounced in the algorithmically cleaned datasets. The floor plans obtained from the query point cloud are shown in Figure 38 a) and b), where a) is cleaned algorithmically and b) is cleaned manually.

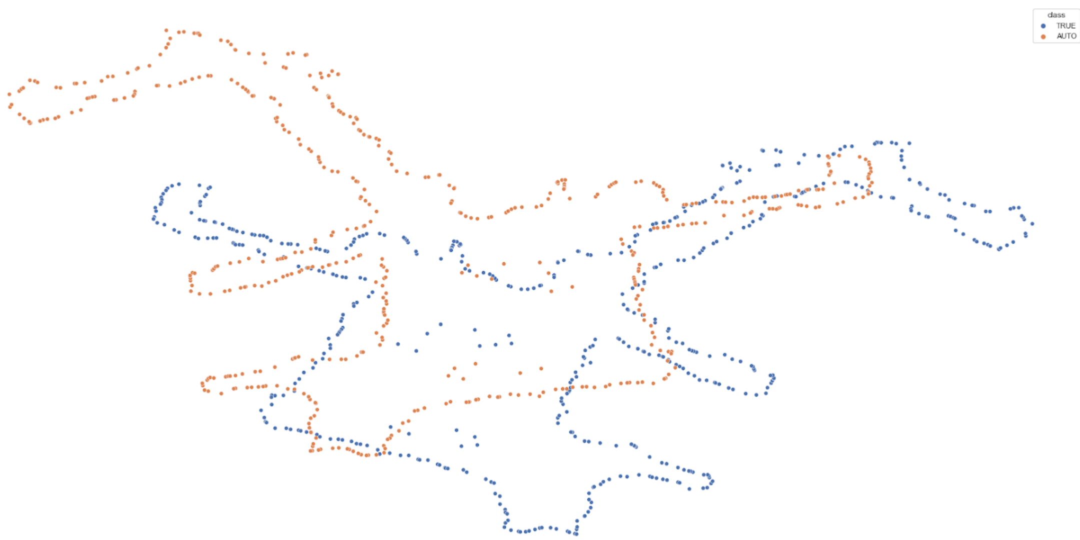
When the automatic registration algorithm is applied, the generated configurations for both the point clouds are shown in Figure 39 a) and b). The final configurations yield a mean distance of 2.8080 meters with a standard deviation of 2.4716 for the algorithmic one while 2.3470 meters mean distance and 2.4085 meters standard deviation for the manual dataset.

As done previously, the resultant registered configuration is also compared with the actual registration. The mean deviation in the algorithmic query point cloud is around 47.5162 meters. With standard deviation, maximum deviation, and minimum deviation being 23.9809 meters, 114.2337 meters, and 14.3044 meters, the automatic registration works well. Similarly, for the manual query point cloud, the mean, standard deviation, and maximum and minimum deviation are 5.0370 meters, 2.4975 meters, 9.1385 meters, and 0.1189 meters. The fit and deviation are visualized in Figure 40 a) and b). Also, the statistics of deviation from actual registration can be found in Table 4.

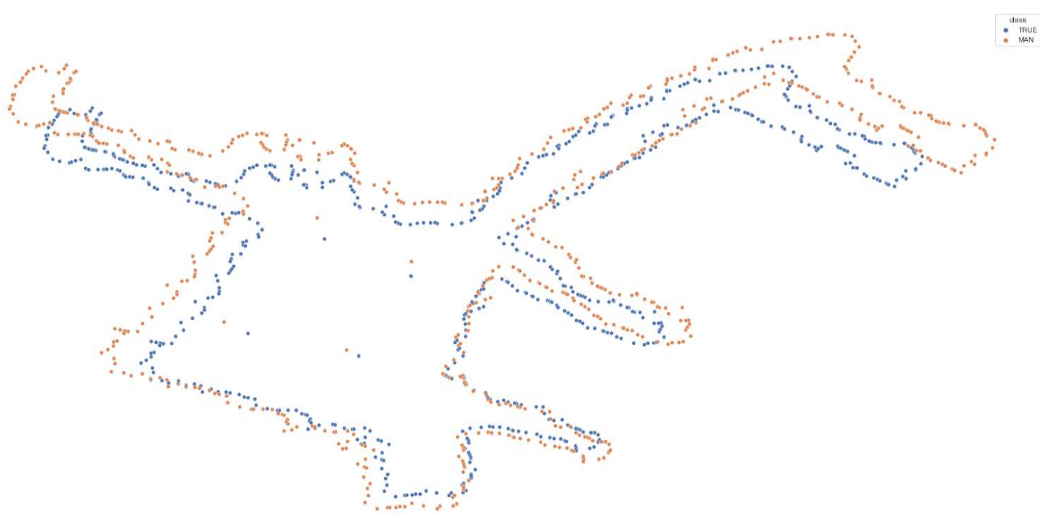
a)

b)

Figure 39: Final configuration of point clouds after automatic registration a) algorithmic point cloud b) manual point cloud where blue points represent BIM and orange points represent floor plan point cloud.



a)



b)

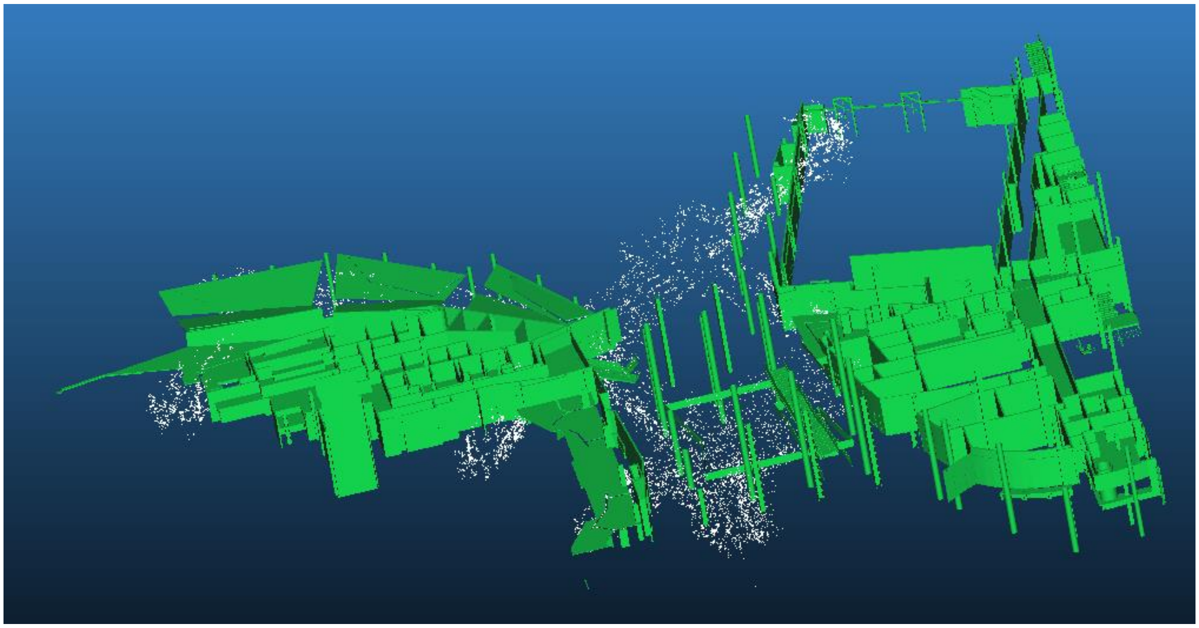
Figure 40: The deviation from true registration (blue) is visualized for each dataset a) algorithmic (orange) b) manual (orange).

The automatic registration works reasonably well for the manual dataset. However, ICP gets stuck on a local minimum configuration of RMS distance for the algorithmically clean dataset which is quite off than the true registration.

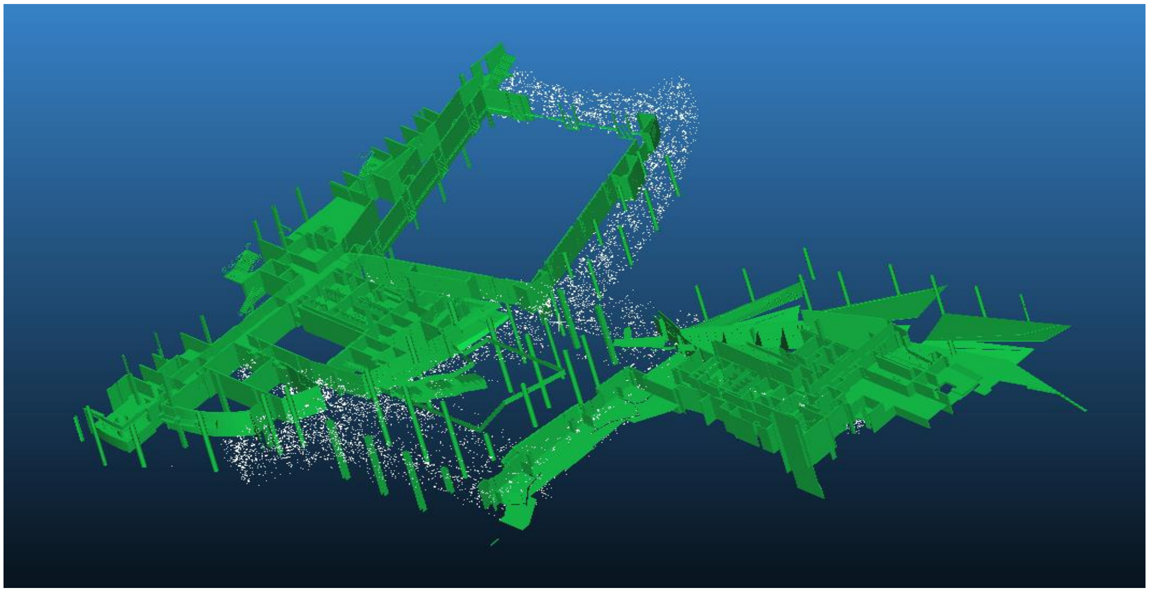
Table 4: Deviations from true registration for Talley's third floor.

Statistics of deviation from true registration	**Algorithmically cleaned**	**Manually cleaned**
Mean	47.51627484	5.037072125
Minimum	14.30439877	0.118962327
Maximum	114.2336547	9.138556758
Standard deviation	23.98091006	2.497572869

For the vertical registration, the results are shown in Figure 41 a) and b). The vertical registration performs quite well.



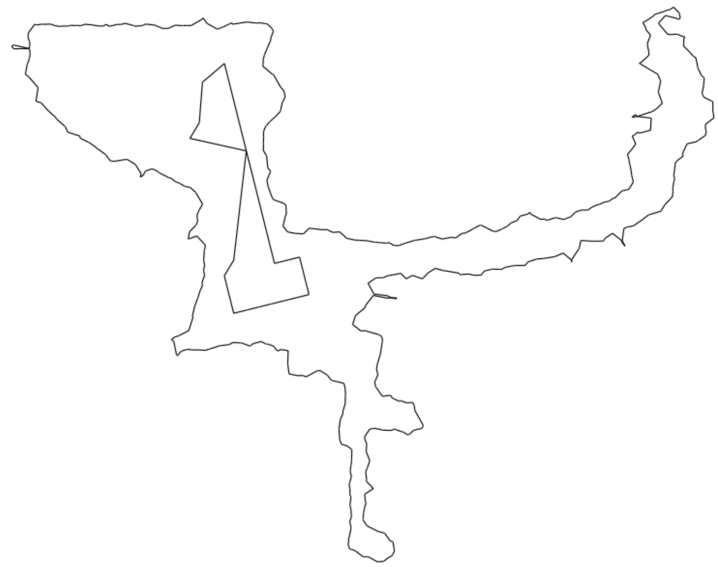
a)



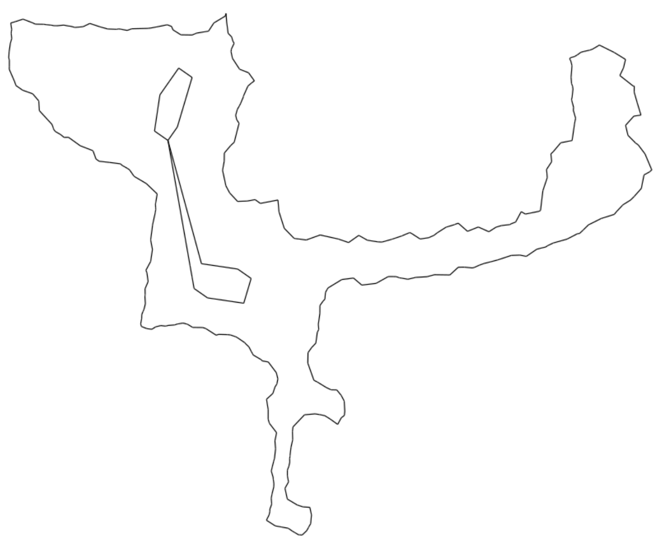
b)

Figure 41: Final registration of Talley's third floor for a) algorithmically and b) manually cleaned dataset.

4.3.5 Student Talley Center: Fourth floor



a)



b)

Figure 42: The floor plans obtained from the query point cloud obtained after a) algorithmic cleaning and b) manual cleaning.

The approximate floor plan obtained from the query point cloud works reasonably well for generating the outer walls of the hall (as seen in Figure 42). However, the algorithm generating the plan for the inner walls fails here. This is because Talley's fourth floor consists of two different and separate inner hallways. So, when the code generates an alpha shape to the inner point cloud, it joins the two separate partitions. The method of cleaning does not help here. Going ahead with this floor plan, the automatic registration algorithm developed here yields a mean distance of 1.4332 meters and a standard deviation of 1.1755 meters where the manual cleaning method is used on the query point cloud (Figure 43 a). The same values of the algorithmically clean dataset are 1.3103 meters and 0.9708 meters (Figure 43 b).

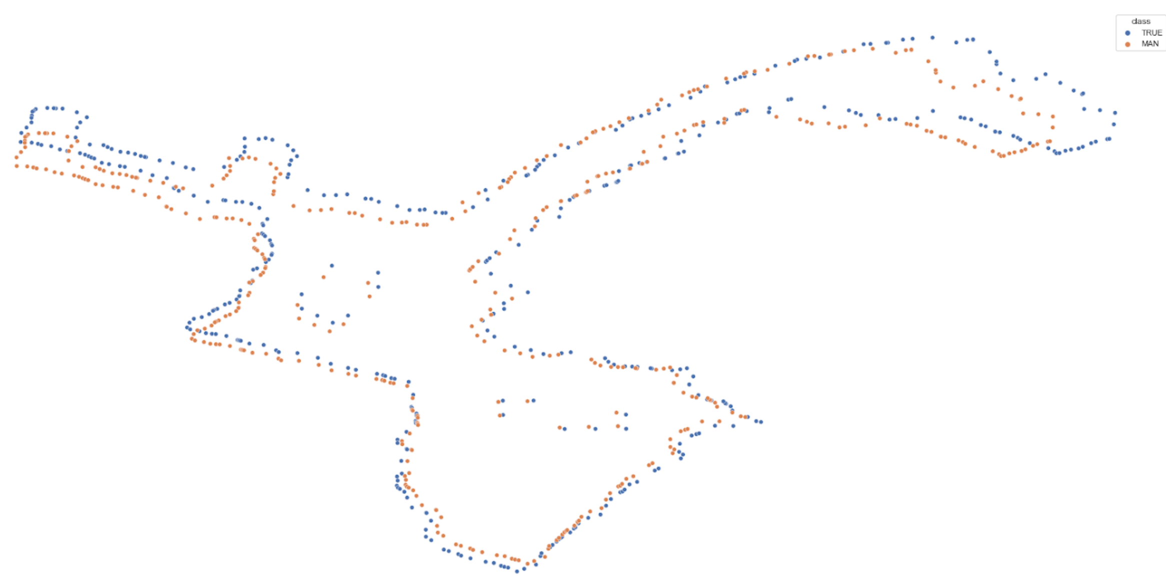
Compared with actual registration, automatic registration performs excellently (Figure 44 a) and b), particularly the one cleaned algorithmically. Its mean deviation from truth sits at 0.6142 meters, the maximum deviation at 1.2914 meters, the minimum deviation at 0.1040 meters, and the standard deviation at 0.2880 meters. The manually cleaned also yields a reasonably registered configuration having a mean deviation of 3.3916 meters, a minimum and a maximum of 0.3426 meters, and 6.5892 meters.

a)

b)

Figure 43: Automatic registration for datasets cleaned by a) algorithmic methods and b) manually.

a)



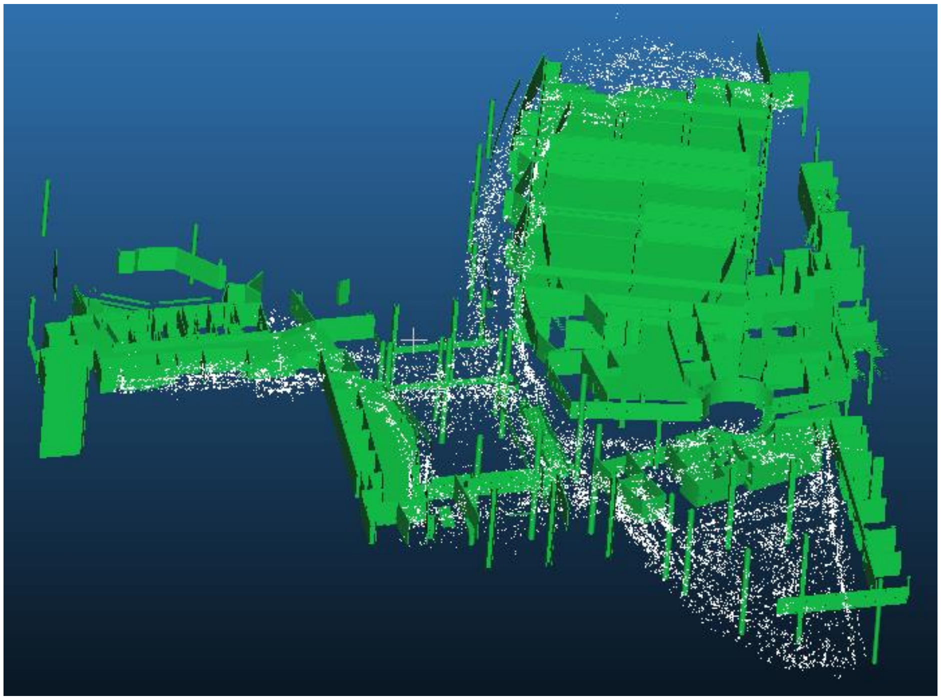
b)

Figure 44: Registration results compared against actual registration a) algorithmic point cloud b) manual point cloud.

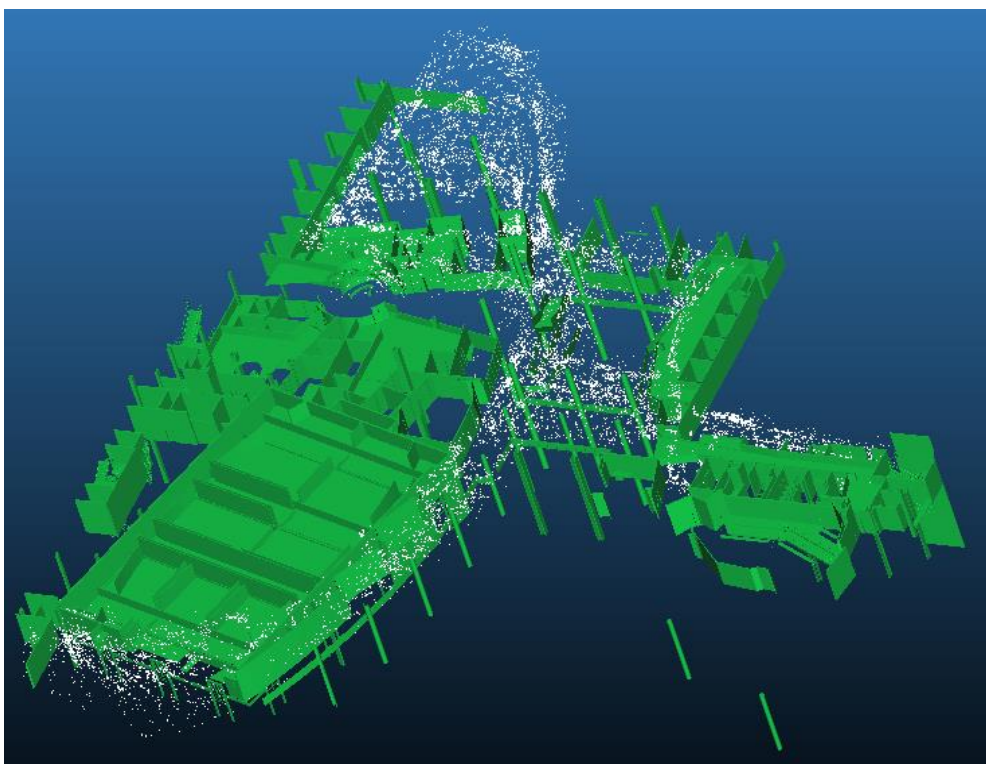
Table 5: Deviations from true registration for Talley's fourth floor.

Statistics of deviation from true registration	**Algorithmically cleaned**	**Manually cleaned**
Mean	0.614254091	3.3916348
Minimum	0.104076387	0.342639258
Maximum	1.29141692	6.589237044
Standard deviation	0.288017418	1.768835947

The automatic registration works with a high degree of accuracy for this dataset. Vertically, as well, the registration is quite well as we see in Figure 45.



a)

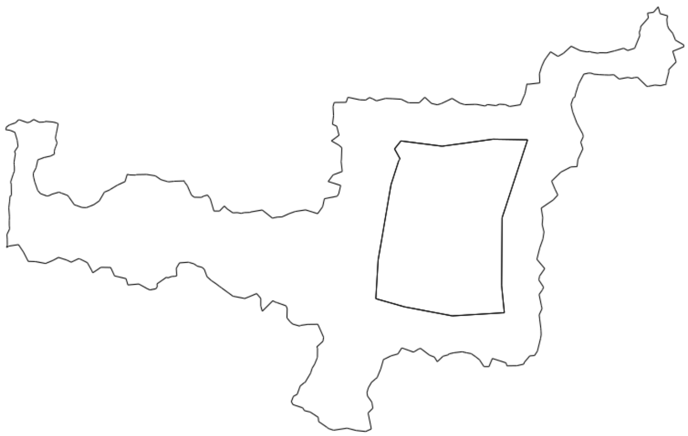


b)

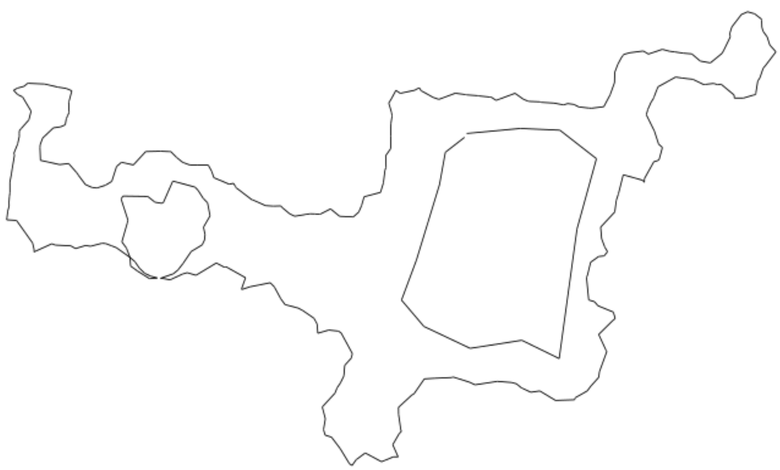
Figure 45: Final registration of Talley's fourth-floor a) algorithmically, and b) manually.

The main reason registration works well here is because SLAM obtained point cloud is well-defined (reducing ill-shaped query point clouds). Additionally, the overall building is an unsymmetrical geometry of the floor (reducing chances of flipping) and it has a large number of points for ICP to find a minimum.

4.3.6 Student Talley Center: Fifth floor



a)



b)

Figure 46: The floor plans obtained from the query point cloud obtained after a) algorithmic cleaning and b) manual cleaning.

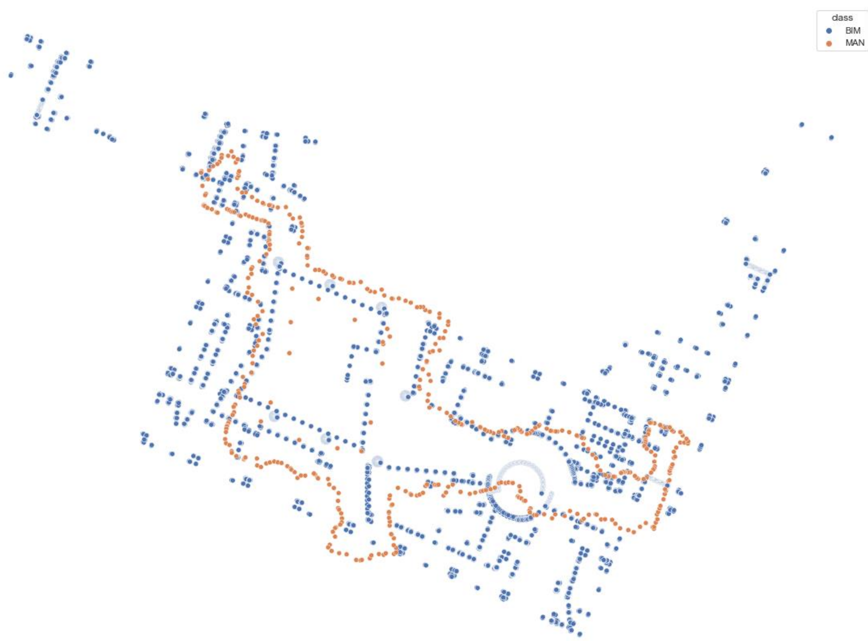
The floor plan generation of the fifth floor of Talley from the query point cloud is an interesting case. Similar to the fourth floor of the Student Talley building, the fifth floor consists of 2 separate and independent hallways disconnected from one another. One is a circular inner hallway (a circular staircase towards the lower floor), while the other is of a conventional polygonal shape (as seen in Figure 46). We have already seen how the code struggles with multiple hallways. But this time, the plan for inner walls generated from outlining the "inverse" point cloud (explained in the previous chapter) does not have linked polygons. In both cases, the inner walls in the floor plan come exclusively from the polygon-shaped inner corridor. This can be explained for each case separately:

1. In the case of algorithmic cleaning, because there are plenty of ORB features in a circular inner hallway/staircase, the code for generating an "inverse" point cloud does not find enough space to consider it an inner hallway. Hence, the code only generates one inner wall polygon.
2. For the case of manual cleaning, the alpha shape for the outer polygon is created in a way that covers the circular inner hallway.

Automatic registration works quite well with Talley's fifth floor (Figure 47). The mean distance between the query point cloud (after automatic registration) and the reference point cloud is around 1.0989 meters with a standard deviation of 0.7797 meters for the algorithmic dataset and around 0.9017 meters with a standard deviation of 0.6054 meters for the manual dataset.

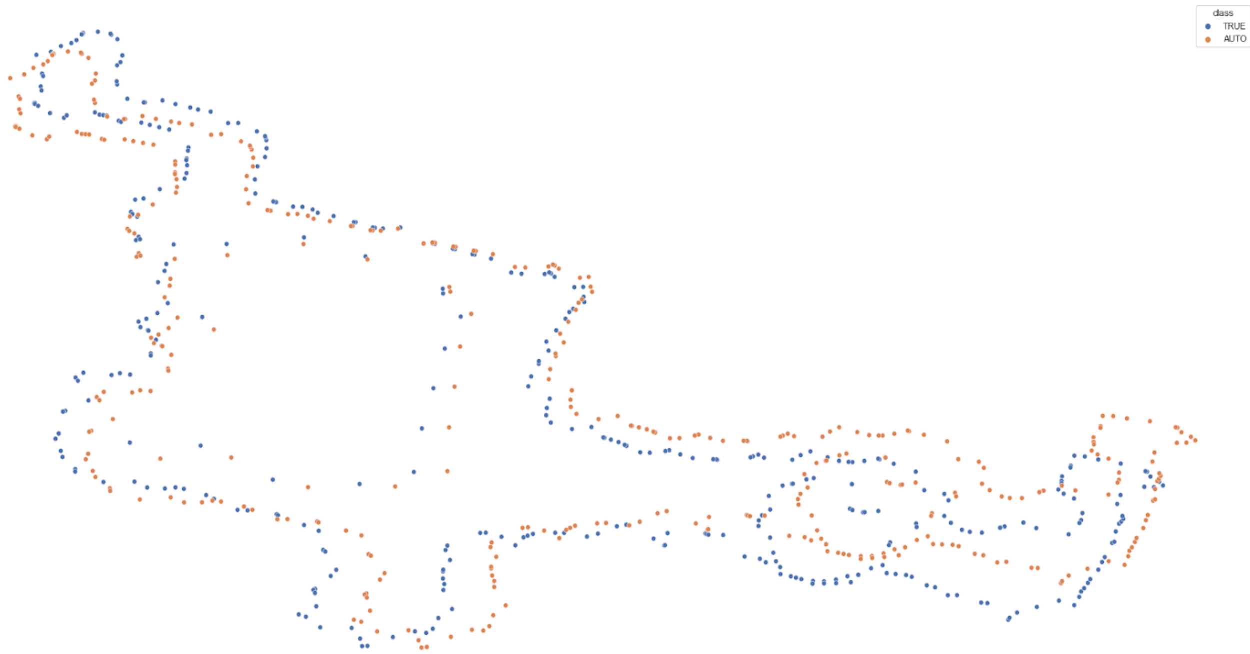
Even when compared with actual registration, the results of automatic registration look good. The mean deviation is about 2.7765 meters with a standard deviation of 0.6602 meters, a maximum deviation of 3.888 meters, and a minimum deviation of 1.0696 meters for the manual dataset. Similarly, the algorithmic dataset's mean deviation is 1.9920 meters, 0.9052 meters standard deviation, 0.0269 meters minimum deviation, and 3.6716 meters maximum deviation. Values can be found in Table 6 and results are visualized in Figure 48.

a)

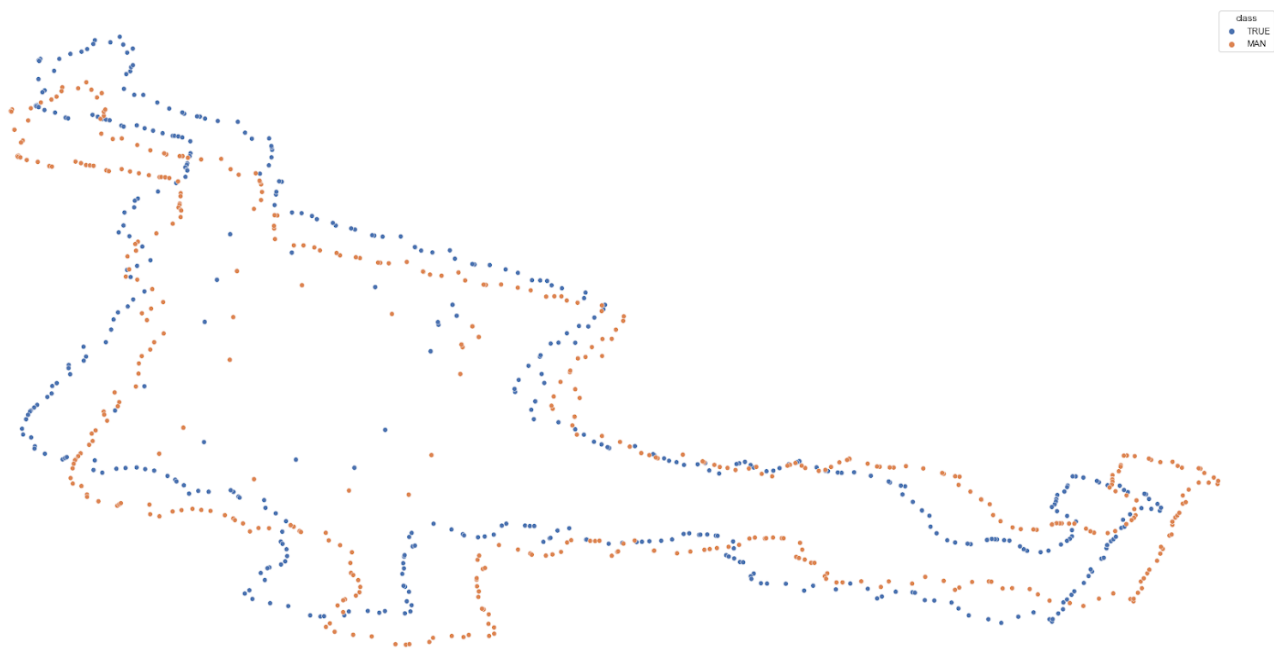


b)

Figure 47: Automatic registration for datasets cleaned by a) algorithmic methods and b) manually.



a)



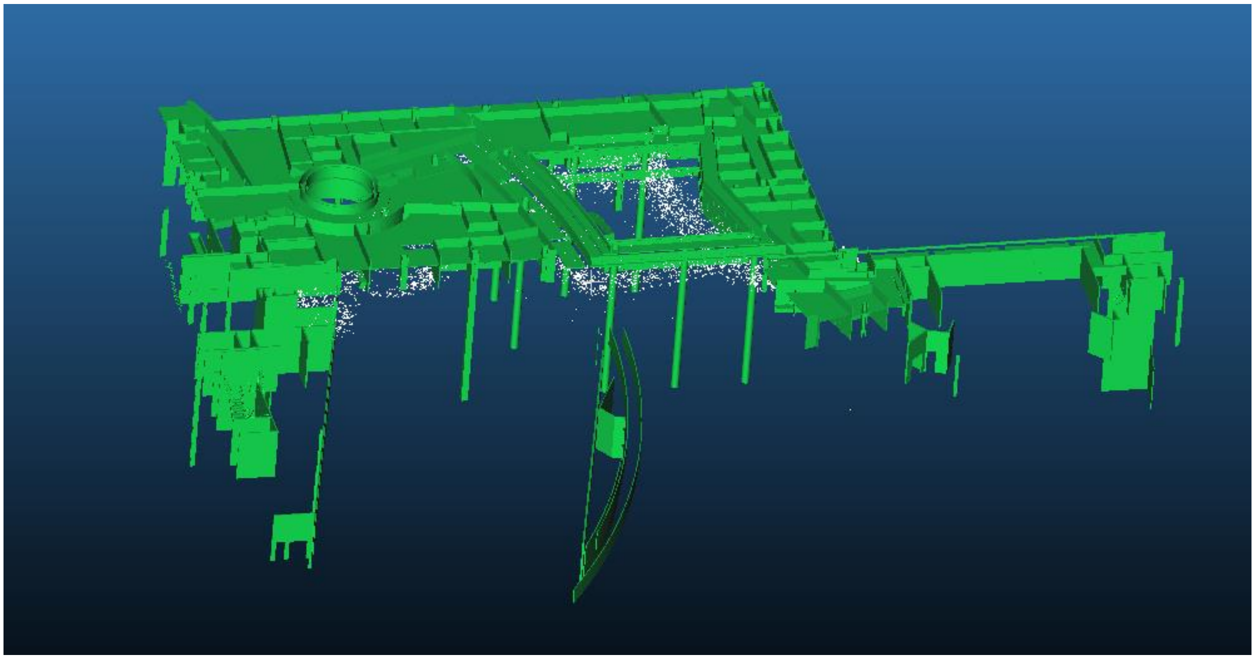
b)

Figure 48: Registration results compared against actual registration a) algorithmic point cloud b) manual point cloud.

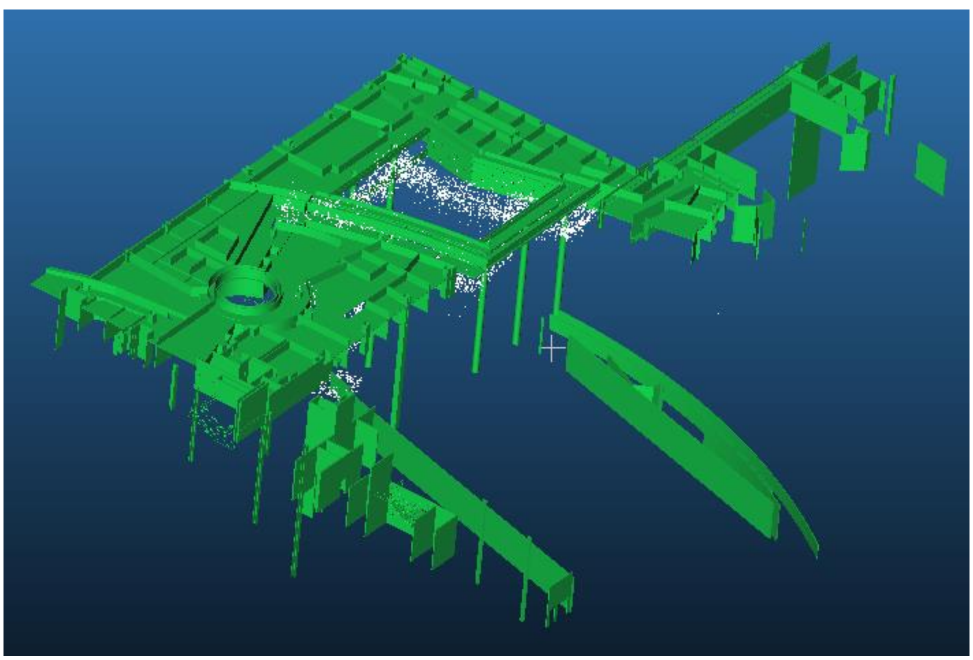
Table 6: Deviations from true registration for Talley's fifth floor.

Statistics of deviation from true registration	**Algorithmically cleaned**	**Manually cleaned**
Mean	1.992040647	2.776586403
Minimum	0.026965694	1.069666111
Maximum	3.67166113	3.888363701
Standard deviation	0.905285425	0.660289406

Because of the same reasons for the fourth floor of Talley, the automatic registration works well in this case. Similar to the fourth floor of Talley, the vertical registration of SLAM obtained point cloud with BIM is quite accurate (Figure 49).



a)



b)

Figure 49: Final registration configuration of Talley's fifth floor on a) algorithmically cleaned floor plan and b) manually cleaned floor plan.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

Based on the results and pattern of the testing datasets, the developed registration system works well for Student Talley Center-type buildings that have a high number of opaque walls with feature-rich environments but fails where there are plenty of transparent glass walls (like FWH third-floor type scenarios) are present. More opaque walls result in more well-defined query point clouds (generated from visual SLAM algorithm) which indicates that the more well-defined a point cloud is, the more accurate the results of automatic registration will be.

Additionally, the developed automatic registration system creates a reasonably approximate floor plan if provided with a clean point cloud and no separate inner hallway. However, if there are multiple inner hallways, the code does not perform well and joins the separate floor plans into one (like it did in the Talley fourth floor). To tackle the cases with multiple inner hallways (like Talley's fourth and fifth floor), one recommendation is to identify if the "inverse" point cloud is completely uniformly spread or not. In the process of "inverse" point cloud generation, the empty cells into which the point cloud was divided were marked uniformly. So, any point with non-uniform distances from its neighbors should be either noise or signify the presence of another independent inner hallway. In a clean "inverse" point cloud, each cluster with uniform point cloud density represents an inner hallway and its boundary, the inner walls of a hallway.

The automatic registration using ICP with RMS distances is not very reliable, it fails quite significantly with FWH third, FWH fourth floor, and Talley third floor. One of the main reasons it fails is because it focuses only on minimum RMS distance configuration, which is not a good measure to assess true point cloud registration. Moreover, the RMS distance used by ICP does not represent the RMS distance between corresponding points of the query and the reference point clouds, nor does it work well as a proxy of the same measure. Since, in a truly registered configuration, the RMS distance between the points of query and reference point clouds are fairly consistent, if the variability of distances between the point clouds is also taken into account, the measure would become a better proxy to the true registration RMS distance.

Secondly, ICP tends to get stuck in the local minima. This causes significantly off-resultant configurations with relatively high RMS distances at times, like in the case of the third floor of Talley, and the fourth floor of FWH. Also, this tendency makes ICP highly dependent on the initial configuration of the query dataset, and results vary vastly with slight changes in the initial configuration.

A modified version of ICP taking the variance of distances between the points along with mean RMS distance can act as a better proxy for calculating RMS distance between the query and reference point clouds. Another idea to develop this model into a robust automatic registration system is by marking a few ORB features "special" in the point cloud and the BIM. The "special " points represent corresponding points in the two coordinate systems. The special points can be aligned first, providing a reasonably good registration. Running ICP over this would make the registration extremely accurate. The "special" points we have suggested above can be any object, such as a QR code sticker placed on each corner of the building that SLAM recognizes and marks as special. If the SLAM algorithm cannot recognize an object, some augmentation work would be required to add a feature of object recognition. The specific location of these "special" can be marked in the BIM/3D model of the building.

The automatic registration system can also utilize the camera trajectory already provided by StellaVSLAM for more accuracy. For example, a camera trajectory representing a hallway loop can be compared with all possible hallway routes on a floor plan. Likewise, possible routes of a building can be calculated from the family of maze-solving algorithms. The configuration where one of the routes aligns the best with the camera trajectory could provide a reasonably good starting point for this automatic point cloud registration system.

REFERENCES

1. NBS Organization. (2020). National BIM Report. Natl. BIM Rep., 1-17.
2. Barbosa, A. S., & Costa, D. B. (2022, November). Use of BIM and visual data collected by UAS and 360° camera for construction progress monitoring. In IOP Conference Series: Earth and Environmental Science (Vol. 1101, No. 8, p. 082007). IOP Publishing.
3. Zollmann, S., Hoppe, C., Kluckner, S., Poglitsch, C., Bischof, H., & Reitmayr, G. (2014). Augmented reality for construction site monitoring and documentation. Proceedings of the IEEE, 102(2), 137-154.Leite, F., Cho, Y., Behzadan, H. A., Lee, S., Choe, S., Fang, Y., Akhavian, R., & Hwang, S. (2016). Visualization, information modeling, and simulation: grand challenges in the construction industry. Journal of Computing in Civil Engineering, 30(6).
4. Eiris, R., Gheisari, M., & Esmaeili, B. (2019). PARS: Using Augmented 360-Degree Panoramas of Reality for Construction Safety Training. Applied Sciences, 9(7), 1379. https://doi.org/10.3390/app9071379
5. Halder, S.; Afsari, K.; Serdakowski, J.; DeVito, S. A Methodology for BIM-Enabled Automated Reality Capture in Construction Inspection with Quadruped Robots. In Proceedings of the International Symposium on Automation and Robotics in Construction, Bogotá, Colombia, 13–15 July 2021; pp. 17–24.
6. Bourke, P. (2014). Sense of presence and its impact on emotional responses to immersive virtual reality environments in panoramic images. In Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry (pp. 81-88). ACM.

7. Gheisari, M., Foroughi Sabzevar, M., Chen, P., & Irizzary, J. (2016). Integrating BIM and panorama to create a semi-augmented-reality experience of a construction site. *International Journal of Construction Education and Research*, *12*(4), 303-316.
8. Côté, S. (2011). Augmented reality for infrastructure: a first step. *Published electronically on BE Communities*.
9. Besl, P., & McKay, N. (1992). A method for registration of 3-D shapes. IEEE Transactions on pattern analysis and machine intelligence, 14(2), 239-256.
10. Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. International journal of computer vision, 13(2), 119-152.
11. Fuentes-Perez, J. F., & Martinez-Carranza, J. (2013). A review of image segmentation approaches: evolution, recent trends, and comparative analysis. arXiv preprint arXiv:1305.4969.
12. Schmidt, A., Kaess, M., & Burgard, W. (2014). Robust 3D Mapping with Multi-Resolution Surfel Trees. In Proceedings of Robotics: Science and Systems Conference.
13. Bertsimas, D., & Tsitsiklis, J. (1993). Simulated annealing. Statistical science, 8(1), 10-15.
14. Segal, A., Haehnel, D., & Thrun, S. (2009). Generalized-ICP. In Robotics: Science and Systems Conference.
15. D. Fox, W. Burgard, and S. Thrun, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," in Proceedings of the 2004 IEEE International Conference on Robotics and Automation, 2004, pp. 403-408. doi: 10.1109/ROBOT.2004.1308895

16. J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in Proceedings of the 2014 IEEE International Conference on Computer Vision, 2014, pp. 1778-1785. doi: 10.1109/ICCV.2013.221

17. Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," in Proceedings of the 2018 International Conference on 3D Vision, 2018, pp. 1-10. doi: 10.1109/3DV.2018.00007

18. L. Liu, Y. Lu, J. Ye, and Y. Wang, "LSG-CPD: Coherent Point Drift with Local Surface Geometry for Point Cloud Registration," in Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision, 2021, pp. 12855-12864. doi: 10.1109/ICCV51895.2021.01287

19. S. Agarwal, et al., "Building Rome on a Cloudless Day," in Proceedings of the 12th European Conference on Computer Vision, 2012, pp. 368-381. doi: 10.1007/978-3-642-33718-5_27

20. L. Li, H. Li, and J. Zhao, "The Robustness of Slam Algorithms in Mobile Robots," in Journal of Field Robotics, vol. 32, no. 1, pp. 1-15, 2015. doi: 10.1002/rob.21551

21. J. Klein and M. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007, pp. 225-234. doi: 10.1109/ISMAR.2007.4538852

22. R. Gao, et al., "Visual SLAM algorithms: A survey from 2010 to 2016," in Journal of Intelligent & Robotic Systems, vol. 88, no. 1, pp. 35-65, 2017. doi: 10.1007/s10846-017-0559-x

23. R. A. Carino, J. J. Mendoza, and F. C. Somera, "Enhanced Online Visual SLAM for Dynamic Scenes via Adaptive Feature Management," in Journal of Field Robotics, vol. 37, no. 2, pp. 335-350, 2020. doi: 10.1002/rob.21925
24. Duke University: D. Carlson, "Feature-Based Visual SLAM," in CPS 296.2: Mobile Robotics, Duke University, 2007. [Online]. Available: https://courses.cs.duke.edu/spring
25. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system [Preprint]. arXiv:1502.00956
26. Smith, J. D., & Jones, R. (2018). Simultaneous localization and mapping: Part I. IEEE Journals & Magazine, 36(5), 12-25. https://doi.org/10.1109/MRA.2018.2841238
27. Engel, J., Sturm, J., & Cremers, D. (2014). Photometric 3D mapping with RGB-D cameras. International Journal of Robotics Research, 33(12), 1733-1752. https://doi.org/10.1177/0278364914534255
28. Labbé, M., Michaud, F., & Dudek, G. (2014). RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. Journal of Field Robotics, 31(4), 550-566. https://doi.org/10.1002/rob.21525
29. Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In AAAI/IAAI (pp. 593-598).
30. Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based SLAM. IEEE Intelligent Transportation Systems Magazine, 2(4), 31-43. https://doi.org/10.1109/MITS.2010.938298

31. Sturm, J., Engel, J., & Cremers, D. (2012). LSD-SLAM: Large-scale direct monocular SLAM. Computer Vision–ECCV 2012, 782-795. https://cvg.cit.tum.de/research/vslam/lsdslam

32. Forster, C., & Carlone, L. (2017). SVO: Fast semi-direct monocular visual odometry. Robotics and Automation (ICRA), 2017 IEEE International Conference on, 15-21. https://rpg.ifi.uzh.ch/svo_pro.html

33. Cummins, M., & Newman, P. (2011). Appearance-only SLAM at large scale with FAB-MAP 2.0. The International Journal of Robotics Research, 30(9), 1100-1123. https://www.robots.ox.ac.uk/~mjc/Papers/RSLAMLargeScaleMapping.pdf

34. Stella-CV. (n.d.). Overview. Read the Docs. https://stella-cv.readthedocs.io/en/latest/overview.html

35. Lv, Q., Lin, H., Wang, G., Wei, H., & Wang, Y. (2017). ORB-SLAM-based tracing and 3D reconstruction for robot using Kinect 2.0. In 2017 29th Chinese Control And Decision Conference (CCDC) (pp. 3319-3324). IEEE. https://doi.org/10.1109/CCDC.2017.7979079

36. Kopsida, M., & Brilakis, I. (2017). BIM registration methods for mobile augmented reality-based inspection. In Proceedings of the 34th International Symposium on Automation and Robotics in Construction (ISARC 2017) (pp. 1-8). Taylor & Francis. https://doi.org/10.1201/9781315386904-31

37. Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In Proceedings of the 2011 International Conference on Computer Vision (ICCV) (pp. 2564-2571). IEEE. https://doi.org/10.1109/ICCV.2011.6126544

38. Forster, C., Lynen, S., Kümmerle, R., & Scaramuzza, D. (2015). Monocular visual-inertial SLAM with points and lines. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (pp. 2607-2614). IEEE. https://doi.org/10.1109/ICRA.2015.7139638

39. "MessagePack: It's like JSON. but fast and small." (n.d.). Retrieved February 11, 2023, from https://msgpack.org/

40. "IfcOpenShell," BuildingSMART International Ltd. [Online]. Available: https://github.com/IfcOpenShell/IfcOpenShell. [Accessed: 11-Apr-2023]

41. Stella, C. V. (n.d.). Stella: Computer vision made easy. Simple tutorial. Retrieved April 11, 2023, from https://stella-cv.readthedocs.io/en/latest/simple_tutorial.html

42. CloudCompare. (n.d.). Statistical outlier removal filter. CloudCompare Wiki. Retrieved April 11, 2023, from https://www.cloudcompare.org/doc/wiki/index.php/SOR_filter

43. MathWorks. (n.d.). alphashape. MATLAB documentation. Retrieved April 11, 2023, from https://www.mathworks.com/help/matlab/ref/alphashape.html

44. Edelsbrunner, H., & Mücke, E. P. (2010). Alpha shapes. In M. G. De Berg, M. van Kreveld, M. Overmars, & O. Schwarzkopf (Eds.), Computational geometry: Algorithms and applications (3rd ed., pp. 137-163). Springer-Verlag. Retrieved April 11, 2023, from http://graphics.stanford.edu/courses/cs268-14-fall/Handouts/AlphaShapes/2010-B-01-AlphaShapes.pdf