Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Registering and visualizing point cloud data with existing 3D CityGML Models

Alan Orlando Gomez Torres

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul G. Plöger
Dr. Stefan Rilling
Alex Mitrevski

January 2021

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Alan Orlando Gomez Torres

Abstract

Rescue services put their lives at risk trying to save other lives. Therefore, robots and other new technologies are being used to help them do their meaningful work. Point clouds can be employed to monitor the current state of a danger area, and CityGML models to depict its ideal state, supporting rescue services in designing an incident action plan. The point cloud and the CityGML model registration is, therefore, of great importance.

The existing methods to register a point cloud with a CityGML model do not leverage the CityGML model's information or only work when the point cloud and the CityGML model are already coarsely registered.

In this work, the 3D registration problem was converted into a 2D registration problem by projecting the point cloud and the CityGML model's terrain intersection into the xy-plane. In the resulting projection, the lines represented the walls of the 3D space, and the intersections of the lines and their angle were utilized as features to perform the registration. A Mixed Integer Linear Program was then utilized to compute the correspondences between the features and a transformation that aligns them. The obtained transformation was finally extended to be applied in the 3D space. The proposed method successfully performs a coarse registration of a point cloud with a CityGML model. Moreover, it can be extended to work with other features to increase its accuracy.

Acknowledgements

I would like to thank my supervisor Prof. Dr. Paul G. Plöger, for helping me with his “Robot Perception” class to understand essential concepts for accomplishing this project. I would also like to thank my supervisor Dr. Stefan Rilling and Julian Theis, for giving me an introduction to the A-DRZ project at the Fraunhofer IAIS and supporting me with all the necessary information from the A-DRZ project, especially the data needed to evaluate this work. And thanks to my supervisor Alex Mitrevski who always answered my theoretical and technical questions.

And last but not least, thanks to all my family and my friends, who support me emotionally and economically during this project.

Contents

List of Figures

1

Introduction

1.1 Motivation

Every day, people are exposed to some kind of accident, and the worst kind of accidents are those that put their lives at risk. Fortunately, for those kinds of accidents, we have rescue services like firefighters. Nevertheless, firefighters are humans and put their lives at risk trying to save other people. According to the Deutscher Feuerwehrverband (German Firefighters Association), the number of firefighters missions in Germany in 2017 reached more than 3 million, and the number of deaths in fires in 2016 was approximately 350 [3].

What if the firefighters are supported by rescue robots? With the use of rescue robots that explore the accident areas, we can help firefighters do their job more safely. Rescue robots can explore buildings to find entrances, exits, and other key areas that help firefighters rescue people faster and more efficiently. However, robots cannot find their exact position inside a building with a single GPS because this kind of technology is unreliable under certain conditions (e.g., inside buildings).

In this work, we try to enable rescue robots to explore buildings using LiDAR systems and 3D models of the buildings. With the use of LiDAR systems, we can obtain representations of the current state of buildings at risk without endangering any lives. These kinds of representations are known as point clouds.

With the use of CityGML models [22], we have access to the 3D representation of many buildings in Germany. CityGML models can be presented in different Levels of Detail (LOD), from the coarsest level LOD0 to the most refined level LOD4. In LOD0, buildings can be represented by a simple footprint. While in LOD4, buildings contain information about windows, doors, stairs, and other internal structures. Unfortunately, not all models are LOD4. Even if they were, the information might be outdated. For example, it may be that due to some collapse, some door or window is blocked.

Using modern registration techniques, one can find a match between the point cloud given by the LiDAR system and the 3D model of the building. The registration result can then be used to find the robot's exact position with respect to the building. The registration result can also help locate relevant parts of the building, like windows, doors, stairs, or corridors. This information can be presented to the firefighters in a way that enables them to design a rescue plan, with the only objective of supporting firefighters in their mission to rescue lives and safeguard theirs.

1.2 Challenges and Difficulties

Most of the registration algorithms are focused on working with two point clouds. Just a few registration algorithms address the registration of 3D models with point clouds. Moreover, between those registration algorithms, only a couple are created to work with CityGML models, as described in chapter 2.

The registration process is usually done by hand because it is a simple task for humans since we all learn to match patterns when we are just children. Automatic registration is complicated because it is difficult for a computer to find features and match them between two different types of 3D data.

An additional challenge is the data itself. The available point clouds are partial, they do not represent the whole 3D model, and they usually just capture part of the walls of the buildings and the surface outside the buildings. In contrast, the 3D models do not have information about the surface outside the buildings. The only information available in both the 3D models and the point cloud is some of the walls of the buildings.

1.3 Problem Statement

This work intends to implement an automated registration method for a point cloud with a CityGML model. This automated registration method will then be used as part of the A-DRZ (Aufbau des Deutschen Rettungsrobotik-Zentrums) [1] project of the Fraunhofer IAIS. Figure 1.1 shows the system architecture of the A-DRZ project. The final objective of the automated registration in the A-DRZ project is to look inside the CityGML models to find the location and state of relevant parts of the buildings (e.g., doors, windows, stairs, and corridors) that can be used for rescue tasks.

The current automated approaches are mainly focused on point-to-point registration. Therefore, the exploration of point-to-model automated registration methods is necessary to implement a solution that aims to work in real-time. Moreover, there is no automated process implemented in the A-DRZ project yet. However, as an assistance system, the A-DRZ project should avoid human interaction for performing the registration.

Figure 1.2 shows the general workflow of the required method. The method's input is a point cloud and the corresponding CityGML model of a building (the process to obtain this input is outside of the scope of this work). The provided input is preprocessed and used for the registration algorithm. The output of the registration algorithm is a transformation that maps the point cloud to the CityGML model. Finally, the output transformation and the input data are used to visualize the point cloud on the CityGML model.

First, the resulting algorithm will be tested manually. If the results are the expected ones, the resulting algorithm will be integrated into a visualization software from the Fraunhofer IAIS. At the moment, this visualization software is, in general, a 3D rendering of CityGML models limited to a particular geographical area.

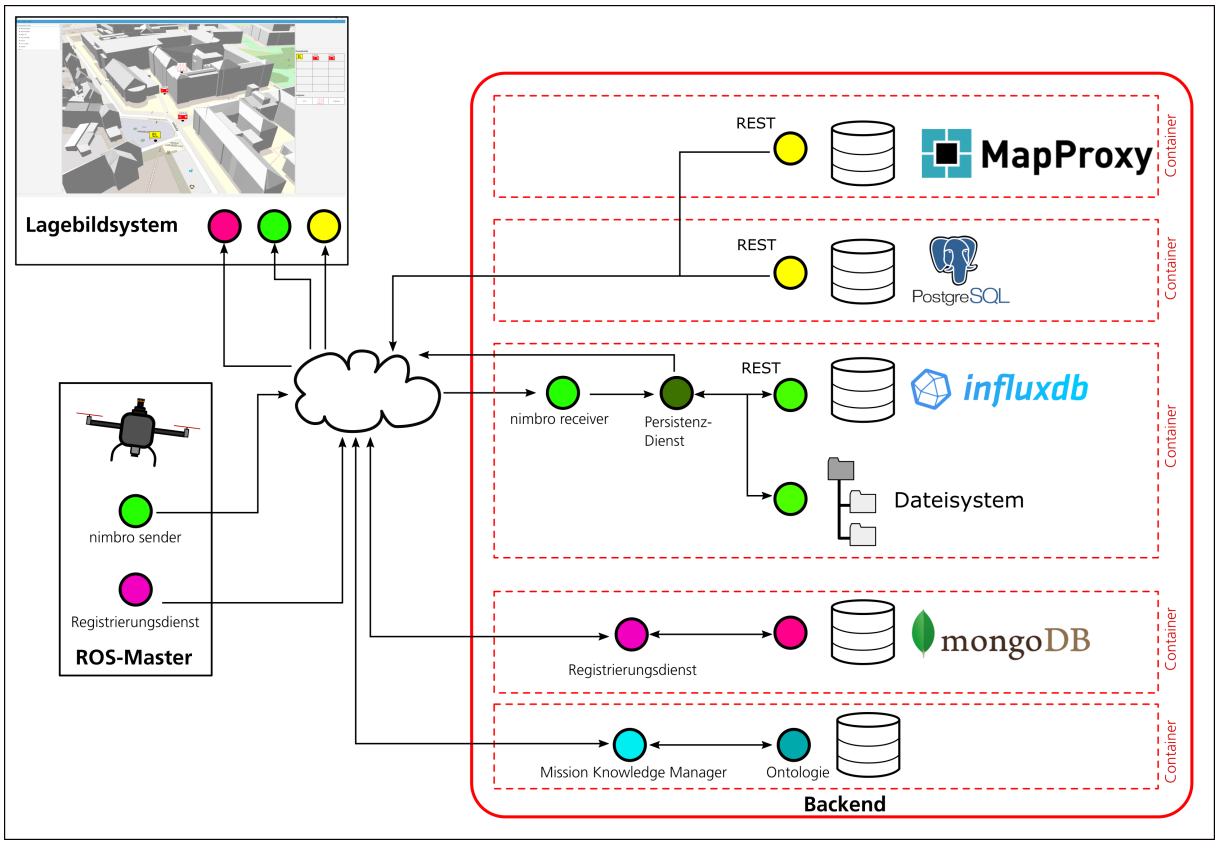


Figure 1.1: A-DRZ architecture system (Image provided by the Fraunhofer IAIS).

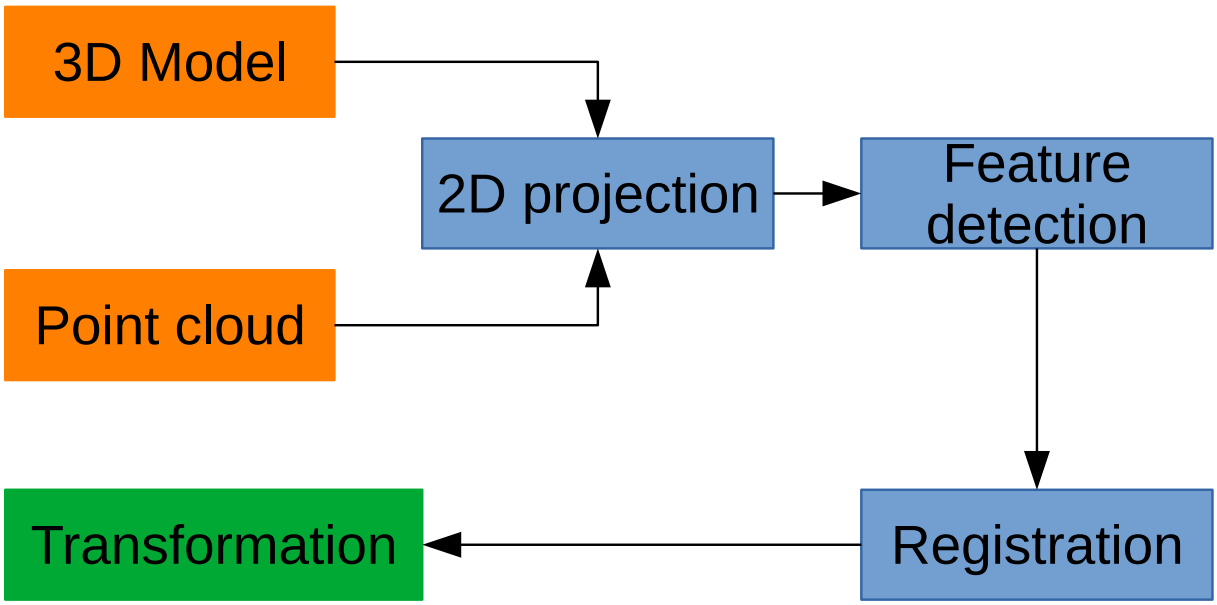


Figure 1.2: System diagram.

2

State of the Art

Point cloud registration algorithms are typically classified into two major categories: global (coarse) registration and local (fine) registration [36, 26]. Coarse registration algorithms attempt to find a transformation that approximately maps the source point cloud to the target point cloud. Fine registration algorithms attempt to refine an initial transformation that maps the source point cloud to the target point cloud. Some approaches combine both of these methods; they first apply a global method and then refine the results with the help of a local method. We can classify these approaches into a third category called hybrid methods.

For the purpose of this work, we classify point cloud registration algorithms into two other major categories: point-to-point and point-to-model. Point-to-point registration algorithms attempt to align two point clouds, while point-to-model registration algorithms attempt to align a point cloud with a 3D model.

The approaches that do not use point cloud data or 3D models, but are still related to the registration problem, are classified into the category of others. Figure 2.1 shows an overview of the classification of the papers reviewed.

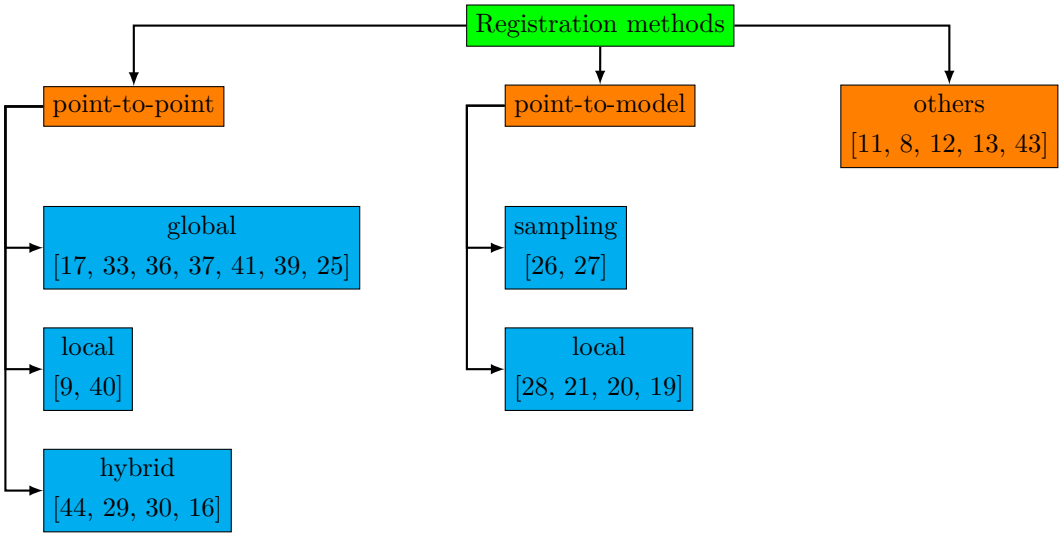


Figure 2.1: Classification of the State of the Art.

2.1 Point-to-point registration

2.1.1 Global methods

RANSAC [17] is a well-known algorithm that can be used to perform the registration task. Generally speaking, given a set of already computed correspondences, the algorithm consists of 2 main steps that are repeated iteratively. In the first step, a number of correspondences, depending on the degrees of freedom of the required transformation, are randomly selected and used to compute a perfect transformation. In the second step, the number of outliers is counted. If the value is under a given percentage, then the algorithm terminates. Otherwise, it is repeated from the first step. Another stop condition is the number of iterations. If the value is above a given limit, then the algorithm stops, and the transformation with the lower percentage of outliers is selected.

Unfortunately, RANSAC is time-consuming due to its randomness. Therefore, Pankaj et al. [33] present a guided sampling improvement to RANSAC to reduce the registration time, sorting the correspondences obtained from a previous 3D feature matching according to their quality.

Following the same line, S. Quan and J. Yang [36] propose a method that helps the RANSAC algorithm sample the best point correspondences. Their approach first detects keypoints using Harris3D, and then it matches the keypoints by using the Local Voxelized Structure (LoVS) descriptor. After that, the set of correspondences is reduced using the Nearest Neighbor Similarity Ratio (NNSR) and the compatibility score of the remaining correspondences. The compatibility score is a function of two constraints: the rigidity constraint and distance between salient points (DSP) constraint, both of them defined by the authors. Finally, the correspondences with the best compatibility scores are used to generate transformations, and the best one (the one with the maximum number of inliers) is the final result. Unfortunately, the RANSAC-based approaches are sensitive to outliers.

Another less common approach for coarse registration is proposed by Sakakubara et al. [37]. This coarse registration approach uses Mixed Integer Linear Programming to find pairs of corresponding points used to find a coarse transformation that aligns two different point clouds. This approach finds a global optimal result without using the values of invariant features. It adjusts the error tolerance depending on the accuracy of the given data and finds the best correspondences between the points. Nevertheless, Mixed Integer Linear Programming problems are NP-hard. Therefore, this approach is excessively time-consuming.

Others have proposed the use of neural networks to solve the global registration problem. Wang and Solomon [41] propose a learning-based method called Deep Closest Point (DCP). The approach proposed first uses PointNet [34] or DGCNN [42] to embed the point clouds. Then, it uses an attention-based module to encode the output of the first part. Finally, it estimates the transformation using a differentiable SVD layer.

Sarode et al. [39] present a registration framework that also uses PointNet to extract features from the point clouds. The PointNet features are then aligned using another network. The process is then repeated a given number of times to improve the registration result.

Ding and Feng [15] present another deep learning method called DeepMapping, whose main structure is constituted of two deep neural networks. The first network is a localization network, whose output is the pose estimation for the input point clouds, and the second network is a map network that estimates the occupancy status of the input locations to compute a loss function that reveals the registration quality.

More recently, Huang et al. [25] propose a learning method that can be trained in a semi-supervised or unsupervised way. The proposed framework is composed of an encoder that extracts features from the input point clouds and a decoder that interprets the extracted features to compute a projection error that serves as input for an algorithm that estimates a transformation increment. This increment is then employed to update the parameters of the transformation and transform one of the point clouds. The whole process is repeated until the loss computed is less than some desired value.

2.1.2 Local methods

Besl and N. D. McKay [9] propose the Iterative Closest Point (ICP) algorithm to perform registration between point sets, line segments, implicit curves, parametric curves, triangles, implicit surfaces, and parametric surfaces. ICP iteratively finds and updates correspondences between the desired sets based on an initial approximation to the solution and the spatial distance.

According to Segal et al. [40], the ICP algorithm can be summarized in two main steps: the computation of correspondences between the two scans and the computation of a transformation that minimizes the distance between the corresponding points. Segal et al. present a generalized version of the ICP algorithm by attaching a probabilistic model to the second step. This generalization makes the algorithm more robust to outliers without modifying the simplicity. However, it does not improve the speed of the original ICP.

2.1.3 Hybrid methods

The global methods results can be refined using a local method, most of the time using a variation of the popular ICP. In this section, some of these methods that combine a global method with a local one can be found.

Go-ICP is a branch-and-bound (BnB) based approach proposed by Yang et al. [44] to address the susceptibility of ICP to local minima. The main idea of Go-ICP is that BnB and ICP work iteratively until a global minimum is reached: BnB finds a solution, which is then refined by ICP. According to the authors, Go-ICP guarantees global optimality regardless of the initialization, and it is recommended for scenarios where real-time performance is not critical.

The approach proposed by Jun Lu et al. [29] extracts keypoints from the point clouds, then it obtains four optimal point correspondences from the key points by using Super4PCS [32], i.e., the four-point correspondences that minimize the number of outliers. Afterward, it creates a neighborhood ball with each point as its center and obtains these neighborhood balls' overlapping regions. Finally, it applies ICP to the overlapping regions. The main drawback of this method is that overlapping regions are needed to perform the registration task.

Due to the increasing success of Deep Learning in 2D applications, there have been attempts to use deep neural networks to solve the whole point cloud registration problem (global and local registration). DeepICP [30] is, according to its creators, the first end-to-end learning-based point cloud registration framework. DeepICP makes use of other networks to build a complete framework. The first step of DeepICP is the Deep Feature Extraction (PointNet++ [35]), which extracts features from the point clouds. In the second step, the Pont Weighting (3DFeatNet [45]) selects the relevant points. In the third step, the selected points of the target point cloud are sampled using the initial transformations. In the fourth step, Deep Feature Embedding is used to obtain more specific descriptors of the points. In the final step, the corresponding pairs of points are generated with a 3D Convolutional Neural Network to compute an improvement of the initial transformation.

Other approaches do not use a neural network to solve the whole point cloud registration problem but part of it. LORAX is an algorithm proposed in [16] that selects some sets of points called super-points and uses a low-dimensional descriptor generated by an autoencoder to describe the structure of each super-point. The sets of super-points from both point clouds are then matched using the euclidean distance between the descriptors. Finally, a coarse registration is computed using a RANSAC procedure, followed by a fine registration using ICP. The main contribution of LORAX is the use of an autoencoder to extract features from 3D point clouds. Nevertheless, the implementation is not optimized for real-time performance, and it is not proved to be fast for large point clouds.

2.2 Point-to-model registration

2.2.1 Sampling methods

The most straightforward idea to register a 3D model with a point cloud is to sample points from the 3D model to generate a second point cloud and then perform one of the point-to-point registration methods.

Kim et al. [26] register a 3D CAD model of a construction site with a point cloud by sampling points from the 3D model and applying a point-to-point registration algorithm. The registration consists of three main steps. First, the point clouds are re-sampled to the same resolution. Second, PCA is used to identify the principal axes of the data to obtain a coarse registration based on these axes. Third, fine registration is performed with the Levenberg-Marquardt iterative closest point (LM-ICP) algorithm [18].

Kim et al. [27] is an extension of their previous work [26]. In this approach, they just add a noise filter step to the pre-processing of the point cloud to improve the registration results. The rest of the steps remain the same. However, using PCA for coarse registration will not assure a good performance when the building's form resembles a cube.

2.2.2 Local methods

Other approaches work directly with the information provided in the 3D models to perform the registration task.

Li and Song [28] propose two extensions to the most popular local registration method, the ICP algorithm, to use it with an STL-file (3D model) and a point cloud. The first of these extensions is called ICP-STL, and its main modification consists of finding the corresponding triangle mesh of the STL file to the points in the point cloud, and then the closest point in the triangle mesh to each point of the point cloud. The rest of the ICP algorithm remains the same. The second modification is called ICP-DAF (ICP dynamic adjustment factor), and it aims to reduce the time of the ICP-STL method. ICP-DAF adds a dynamic adjustment factor to adjust the transformation parameters dynamically. The disadvantage of this method is the same disadvantage present in the ICP algorithm, i.e., to perform well, it requires a good initial approximation of the solution [37].

In the specific case of CityGML, Goebbels et al. [21] propose an extension of the well-known ICP to perform point-to-model registration based on point-to-plane registration. This approach only considers points on the perimeter of the buildings. The main contribution of this approach is the speed since point-to-model ICP is faster than the point-to-plane ICP. The CityGML models used, however, did not contain any terrain information. Therefore, according to the authors, the ground plane is the lowest point of the building, and some walls might intersect with the real terrain, increasing the algorithm's error. An additional disadvantage of this method is that it expects the point cloud to be coarsely registered with the model.

Another approach by Goebbels et al. [20] explores the possibility of finding features that help to perform the registration. The approach first rotates the point cloud to align the walls to the z-axis. Then, it projects all the points onto the xy-plane to obtain a 2D image of the point cloud viewed from above. After that, it detects the corners in the 2D image. Having the CityGML model's corners, it applies a Mixed Integer Linear Program to perform the registration. Finally, a Linear Program relaxation is used to fine-tune the coarse result. Nonetheless, the point cloud has to be already coarsely registered to obtain a successful and fast result.

An extension of the work in [20] has been made by Goebbels et al. [19], where the steps to perform the registration are the same, but instead of corners, lines are used in the Mixed Integer Linear Program. Furthermore, no fine registration is performed at the end. Unfortunately, it has the same disadvantage as the previous approach: the point cloud has to be already coarsely registered to obtain a successful and fast result. However, this is not a disadvantage if the desired method will be just used to perform a fine registration.

2.3 Others

Other approaches attempt to solve different matching problems with or without correspondences. These approaches cannot be classified in the previous categories because their input data is not composed of two point clouds or one point cloud and a 3D model. Some of these approaches are collected into this category.

Breuel [11] proposes the use of matchlist-based BnB techniques to solve geometric matching problems without correspondences. For example, point and line segment matching in 2D and 3D.

Bazin et al. [8] present an approach that combines Linear Programming and BnB procedures to

achieve global optimality. The proposed algorithm receives two different sets of features extracted from two images and outputs the correspondences between the features, verifying the appearance similarity and geometric constraints. In the end, it not only computes the feature correspondences but also computes an estimation of a geometric transformation between the features and identifies the inliers/outliers.

Brown et al. [12] extend the idea of the two previous approaches and present a 2D-3D registration method that does not need correspondences by using a BnB algorithm. They also propose a deterministic annealing algorithm to reduce computation time. The proposed method works with points or lines, but not with a combination of both. Therefore, Brown et al. [13] propose a framework to solve the 2D-3D registration problem without feature correspondences that is able to use points, lines, or a combination of both.

Windheuser et al. [43] propose the use of Integer Linear Programming to find correspondences between non-rigid 3D shapes. They use an Integer Linear Program to minimize the elastic thin-shell energy required to deform one shape into the other one. Their method can be improved by a relaxation of the Integer Linear Program and the inclusion of feature descriptors.

2.4 Limitations of previous work

ICP and its variants provide simple and easily-implemented iterative methods, but these algorithms can converge to false local optima [41]. Furthermore, they do not scale well with the number of points, and they are not differentiable. Therefore they cannot be integrated into end-to-end deep learning pipelines [39]. Moreover, the principal disadvantage of these methods is that they require a good initial approximation to the solution to perform well [37].

RANSAC and its variations cannot guarantee any global optimality in their final results [8]. According to [36], RANSAC has two main issues. The first one is its computational complexity, which makes it time-consuming. The second one is its randomness during sampling, which can lead to inaccurate results. These two disadvantages worsen with the number of outliers.

The globally optimal methods, such as Linear programming and BnB based methods, are very time-consuming. Therefore, their application in real-time tasks is limited [39].

In the last years, there has been a boom in the use of neural networks and deep neural networks to solve 2D problems. Therefore, their use in 3D tasks has been explored too. Unfortunately, methods based on deep learning generally suffer from issues regarding the generalization ability and the requirement of massive amounts of training data [36].

In general, none of these methods offer a straightforward solution for the coarse registration of a model with a point cloud. The only solution available for this is to sample a point cloud from the model and apply a global method for two point clouds. Unfortunately, this solution does not leverage the complete information contained in the model. However, some of the ideas of the approaches presented in this chapter can be used to implement alternative coarse registration methods of a point cloud with a CityGML model.

3

Methodology

In this chapter, a brief description of the Data and the definition of the Registration Problem will be given. The structure of a CityGML model is complex, and its complete explanation is out of the scope of this project. For detailed information about CityGML please refer to Gröger et al. [22] or the official websites.

3.1 Data

The CityGML models are public data in some states in Germany, for example, in Nordrhein-Westfalen [5]. Other states of Germany, like Hessen [4] or Bayern [2], charge money for their Models. That is not relevant for this project because the CityGML Models are received from the server of the A-DRZ project.

Besides the representation of the building, CityGML models provide other interesting information about the buildings. For this work, the intersection of the model with the terrain is the most relevant information. The terrain intersection is provided as an array of 3D points representing some points on the walls that intersect the ground.

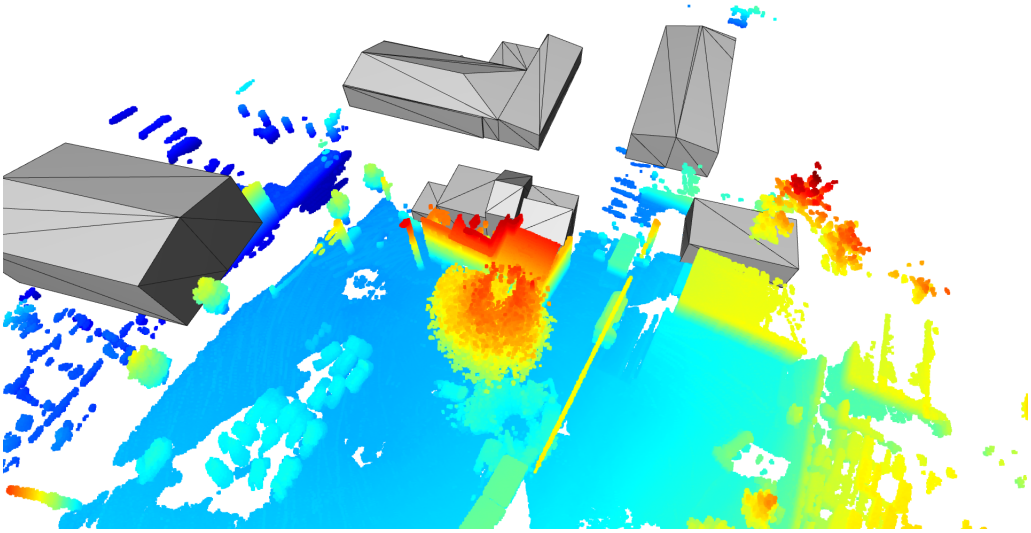
The point cloud data is collected and provided by the A-DRZ project partners that develop and operate the actual Robots. These are mainly the University of Bonn [7] and the Technical University of Darmstadt [6], which have integrated corresponding laser scanners and LiDAR sensors on their platforms.

Figure 3.1a and Figure 3.1b show a CityGML model and its corresponding point cloud before being registered. The pose of the point cloud is different from the pose of the CityGML model.

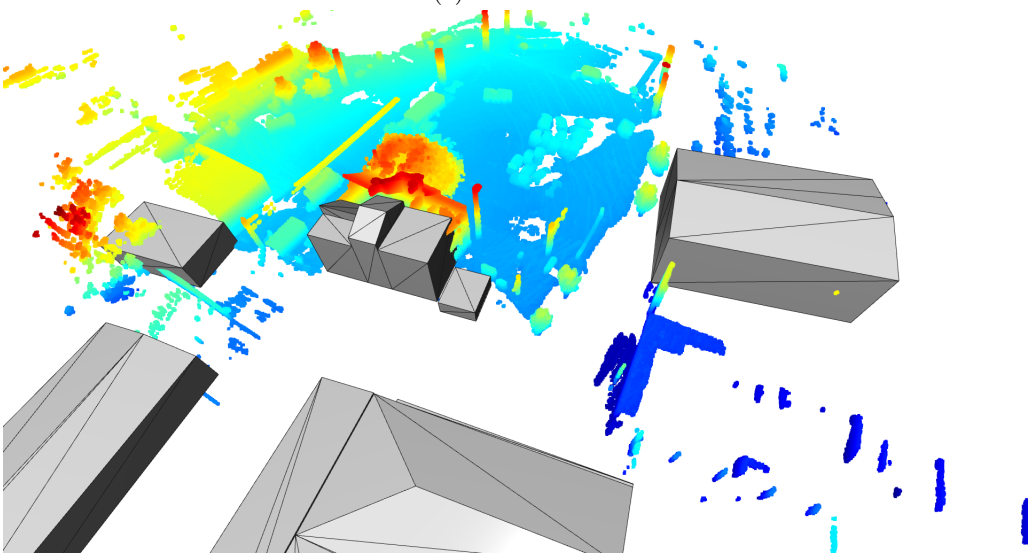
Unfortunately, there is no ground truth for the registration of the data available. Therefore, the results are evaluated manually by a person who decides whether the registration was successful or not.

3.2 Setup

The main setup is depicted in Figure 1.2. The input of the system is a CityGML model and its corresponding point cloud. The output of the registration algorithm is a transformation that aligns the point cloud with the CityGML model. This transformation is then used to visualize the alignment of the input data for its proper evaluation.



(a) Front view.



(b) Back view.

Figure 3.1: CityGML model and its corresponding point cloud before registration.

3.3 Registration Problem in 3D

In general, the registration task consists of finding the transformation that aligns the input CityGML model with the input point cloud. That means a transformation that modifies the point cloud's pose to bring it close to the CityGML model or vice versa.

More specifically, given a source point set $\boldsymbol{P} = \{p_i\}_{i=1}^{N}$, where $p_i \in \mathbb{R}^4$, and a target point set $\boldsymbol{Q} = \{q_j\}_{j=1}^{M}$, where $q_j \in \mathbb{R}^4$, the registration task consists of finding a transformation $\boldsymbol{T} \in SE(3)$ that minimizes:

$$\sum_{i=1}^{N} \min_{j \in 1 \dots M} d(\boldsymbol{T}p_i, q_j) \tag{3.1}$$

where $d(\boldsymbol{T}p_i, q_j)$ is the Euclidian distance between $\boldsymbol{T}p_i$ and q_j. The points p_i and q_j are points in homogeneous coordinates. Therefore, their fourth element is always one.

In this case, the point set $\boldsymbol{P}$ corresponds to the point cloud, while the point set $\boldsymbol{Q}$ corresponds to the CityGML model. It is clear that $\boldsymbol{P}$ is formed by the same points that form the point cloud, but the set $\boldsymbol{Q}$ could be a sampled point cloud from the CityGML model or other characteristic points of the CityGML model.

The difficulty of this task is to find correspondences between $\boldsymbol{P}$ and $\boldsymbol{Q}$ that minimizes Equation 3.1. If the correspondences were defined, the registration would be pretty straightforward and correct.

3.4 Registration Problem in 2D

To simplify the problem, one can transform the 3D registration problem into a 2D registration problem by projecting the point cloud and the CityGML model onto the xy-plane. Both projections can be imagined as an image representing the point cloud's observation or the CityGML model's observation from the same point of view, for example, from above or below.

The registration problem in 2D remains the same as in 3D, but $p_i \in \mathbb{R}^3$, $q_j \in \mathbb{R}^3$, and $\boldsymbol{T} \in SE(2)$. As in the 3D registration problem, p_i and q_j are points in homogeneous coordinates. Therefore, their third element is always one.

3.5 Registration Problem as a Mixed Integer Linear Program

The 3D registration problem defined in section 3.3 can be redefined as a Mixed Integer Linear Program. The exact definition is given by Sakakubara et al. [37]. Furthermore, the 2D registration problem defined in section 3.4 can be redefined as a Mixed Integer Linear Program in the same way.

There is a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times M}$ that represents the correspondences between $\boldsymbol{P}$ and $\boldsymbol{Q}$. That means that $x_{ij} = 1$ if the point p_i in the source image corresponds to the point q_j in the target image, otherwise $x_{ij} = 0$.

Then, there should be a homogeneous transformation $\boldsymbol{T}$ that transforms the point p_i into the point q_j, for all i, j where $x_{ij} = 1$. That means there is a loss function $L = [l_0, l_1, l_2]^T = (q_i - \boldsymbol{T}p_i)$ that should be less than a given threshold ϵ_d for all i, j where $x_{ij} = 1$. The constant ϵ_d can be seen as a threshold that removes outliers. In other words, ϵ_d is the maximum distance allowed between a target point and its corresponding source point after transformation.

More formally, the objective of the Mixed Integer Linear Program is to maximize:

$$\sum_{i}^{N}\sum_{j}^{M} x_{ij} \tag{3.2}$$

subject to:

$$\sum_{i}^{N} x_{ij} \leq 1 \tag{3.3}$$

$$\sum_{j}^{M} x_{ij} \leq 1 \tag{3.4}$$

$$\sum_{i}^{N}\sum_{j}^{M} l_0 \leq \epsilon_d + b(1 - x_{ij}) \tag{3.5}$$

$$\sum_{i}^{N}\sum_{j}^{M} -l_0 \leq \epsilon_d + b(1 - x_{ij}) \tag{3.6}$$

$$\sum_{i}^{N}\sum_{j}^{M} l_1 \leq \epsilon_d + b(1 - x_{ij}) \tag{3.7}$$

$$\sum_{i}^{N}\sum_{j}^{M} -l_1 \leq \epsilon_d + b(1 - x_{ij}) \tag{3.8}$$

$$\sum_{i}^{N}\sum_{k}^{N}\sum_{j}^{M}\sum_{l}^{M} x_{ij} + x_{kl} \leq 1 \quad for \quad \mid d(p_i, p_k) - d(q_j, q_l) \mid \geq \epsilon_l \tag{3.9}$$

$$\begin{aligned} \delta_1^- &\leq t_{11} \leq \delta_1^+ \\ \delta_2^- &\leq t_{12} \leq \delta_2^+ \\ t_x^- &\leq t_{13} \leq t_x^+ \\ t_y^- &\leq t_{23} \leq t_y^+ \\ t_{11} &= t_{22} \\ t_{21} &= -t_{12} \end{aligned} \tag{3.10}$$

Equation 3.3 and Equation 3.4 constraint to one the number of correspondences that a point can have, i.e., a point p_i can only correspond to one point q_j and vice versa. The number on the equation's right-hand side can be increased to allow multiple correspondences between points.

Equation 3.5, Equation 3.6, Equation 3.7, and Equation 3.8 are the implementation of the loss function. If $x_{ij} = 1$, i.e., p_i corresponds to q_j, then the equation's right-hand side is ϵ_d. On the contrary, if $x_{ij} = 0$, i.e., p_i does not correspond to q_j, then the equation's right-hand side is $\epsilon_d + b$. To ensure that the pairs of

points that do not correspond to each other comply with these constraints, the constant b should be some large number. For example, b can be two times the maximum distance between points of the source image or points of the target image [20].

Equation 3.9 is used to preserve lines between points after the transformation. If $\mid d(p_i, p_k) - d(q_j, q_l) \mid \geq \epsilon_l$, then either p_i corresponds to q_j or p_k corresponds to q_l. The constant ϵ_l depends on ϵ_d and according to [37], $\epsilon_l \geq 2\sqrt{3}\epsilon_d$ needs to be satisfied because ϵ_d is the allowed rigid transformation error.

The elements of $\boldsymbol{T}$ are constrained by lower (δ_1^-, δ_2^-, t_x^-, and t_y^-) and upper (δ_1^+, δ_2^+, t_x^+, and t_y^+) boundaries in Equation 3.10. While the rest of the elements of $\boldsymbol{T}$ are constant: $t_{31} = 0$, $t_{32} = 0$, and $t_{33} = 1$. The last two constraints in Equation 3.10, $t_{11} = t_{22}$ and $t_{21} = -t_{12}$, ensure that the transformation is a similarity transformation.

4

Solution

Due to causes unrelated to this project, the available data was just one CityGML model and its corresponding point cloud. Therefore, a solution was designed based on this data and its characteristics. Moreover, at the time this work was started, there was no GPS information of the point clouds used in the registration, just of the CityGML model. Therefore, it was not possible to perform a coarse registration with GPS information. All of this together made it difficult to decide which approach to use.

Later, in the middle phase of the project, another point cloud was available, but its corresponding CityGML model was not. However, instead of a CityGML model, a 3D model of the site was available in Polygon File Format (ply). Because of the different information contained in the PLY file, a different preprocessing was needed.

Although (deep) learning methods are very attractive due to their boom in many 2D and 3D applications in recent years, they are unthinkable to solve our problem. The main reason for this is the data needed to train such an approach. Therefore, and because of the time needed to collect enough data to think about a (deep) learning method, a more traditional method needs to be used.

The first thought was to apply a built-in registration method of Opend3D. Therefore, Open3D's implementations of point-to-point ICP, point-to-model ICP, and RANSAC were tried, unfortunately without success. The reason was probably the difference of information contained in the point clouds and the 3D models. The 3D models contain complete buildings but do not contain any information about the ground or objects outside the buildings. In contrast, the point clouds mainly contain information about the ground and the objects outside the buildings, but they just contain partial information about the buildings, usually a couple of walls.

Goebbels et al. [19, 20] face a similar problem with the difference of information in the data and successfully propose using Mixed Integer Linear Programming to register a point cloud with a CityGML model based on lines and points. Mixed Integer Linear Programming has been used to solve other registration problems like the registration between two point clouds [37], registration between 3D shapes [43], and 2D registration [8]. Therefore, the solution proposed is to detect the walls in the point cloud and the 3D model by projecting both onto the xy-plane, and then use the angles of the corners of the walls (where the walls intersect) in a Mixed Integer Linear Program to find correspondences and a transformation that aligns the point cloud with the 3D Model at the same time. The exact details are given in the next sections.

4.1 Proposed algorithm

The main steps of the proposed algorithm are listed as follows:

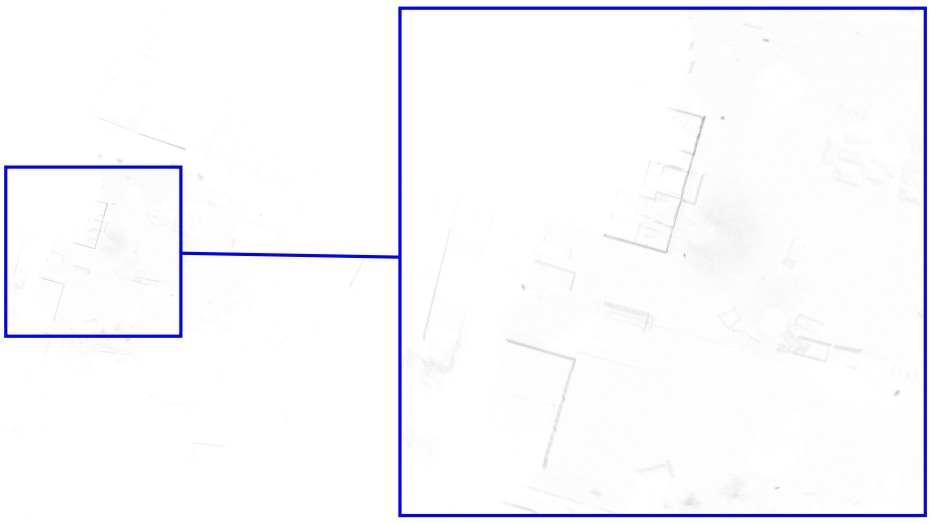
1. Projection of the point cloud onto the xy-plane.
2. Projection of the 3D model onto the xy-plane.
3. Detection of line segments.
4. Detection of line intersections and their angles.
5. Identification of possible correspondences between angles detected.
6. Alignment of both projections onto the xy-plane.
7. Alignment on the z-axis.

The point cloud was downsampled before the first step to speed up the solution. However, this step is optional. From now on, the point cloud will be referred to as the source, and the 3D model as the target.

4.1.1 Projection of the point cloud onto the xy-plane

The registration task is transported from the 3D space to the 2D space in order to simplify it. As we can see in Figure 3.1a and Figure 3.1b, the source's z-axis and the target's z-axis are already correctly aligned, which allows for a direct projection of the source onto the xy-plane to detect walls.

To project the source onto the xy-plane, one can create an empty image with enough size for the source and the target to fit. Then, one can sum the number of points in the source with the same x- and y-coordinates and write the result in the corresponding pixel in the image. In this way, we can obtain a view of the source from below, where line segments depict walls. Figure 4.1 shows the image obtained after the projection of the source.



(a) (b)

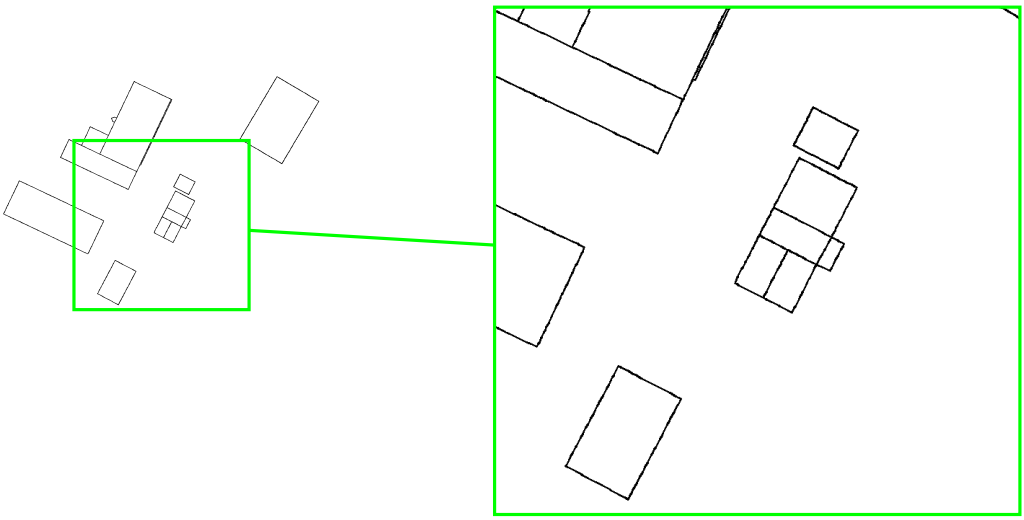
Figure 4.1: Projection of the source onto the xy-palne.

4.1.2 Projection the 3D model onto the xy-plane

The difference between using CityGML Models and the PLY models relies on their projection onto the xy-plane. From the PLY model, a point cloud is sampled and projected onto the xy-plane as described in subsection 4.1.1.

Fortunately, the CityGML model already provides a terrain intersection. From this terrain intersection, only the x- and y-coordinates are used to make the projection. In an empty image, line segments are drawn from point to point to obtain a footprint.

Figure 4.2 shows the image obtained after this process.



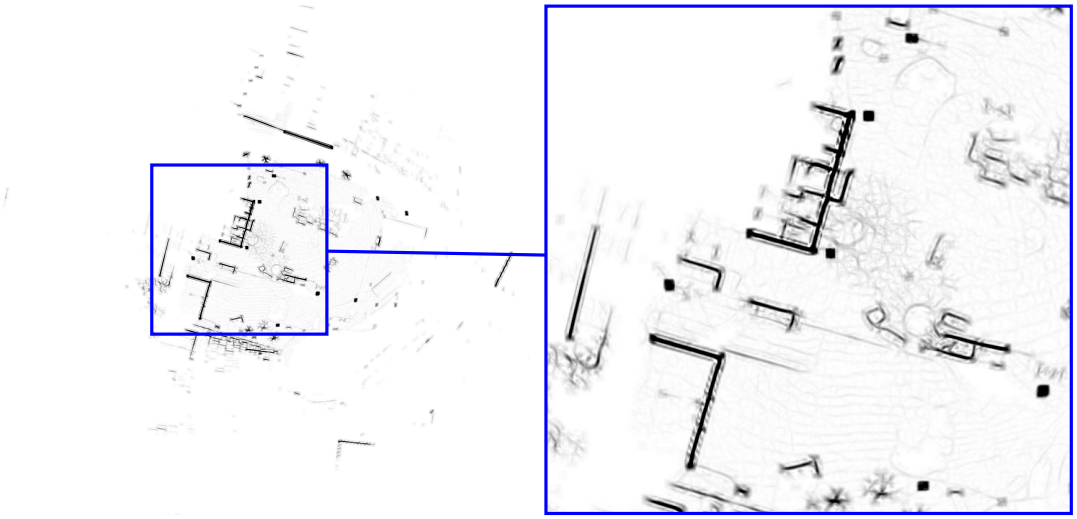
(a) (b)

Figure 4.2: Projection of the target onto the xy-palne.

4.1.3 Detection of line segments

The projections of the source and the target are now 2D images. The source image could be noisy, but the target is clear, and the line segments that represent the walls are very well defined. Therefore, to deal with the noise and remark the line segments of the source image, an implementation of the technique proposed by Chaudhuri et al. [14] provided by DIPlib [31], to detect blood vessels in retinal images, together with a binary threshold is used. Figure 4.3 shows the result after the filter is applied to Figure 4.1.

Line segments in both images are detected using the probabilistic hough line transform of OpenCV [10]. The number of line segments is reduced in four steps. Firstly, short line segments are deleted, i.e., line segments with lengths lower than a given threshold. Secondly, line segments with similar slopes are identified. Then, from this set of line segments, subsets of line segments with close starting and ending points are created, and only the longest one of each set is preserved. The rest of the line segments are



(a) (b)

Figure 4.3: Source image after filter.

deleted. Thirdly, line segments with similar inclination angles are detected, and the same as in the second step is done. Finally, line segments that are too close and have similar inclination angles are merged by eliminating or extending one of the line segments.

4.1.4 Detection of line intersections and their angles

The line segments are converted to their homogeneous representation, and the intersections between them are calculated using the cross product as described in [24]. Only the intersections that are in the line segments are taken into account. Then, the angles of the intersections are computed.

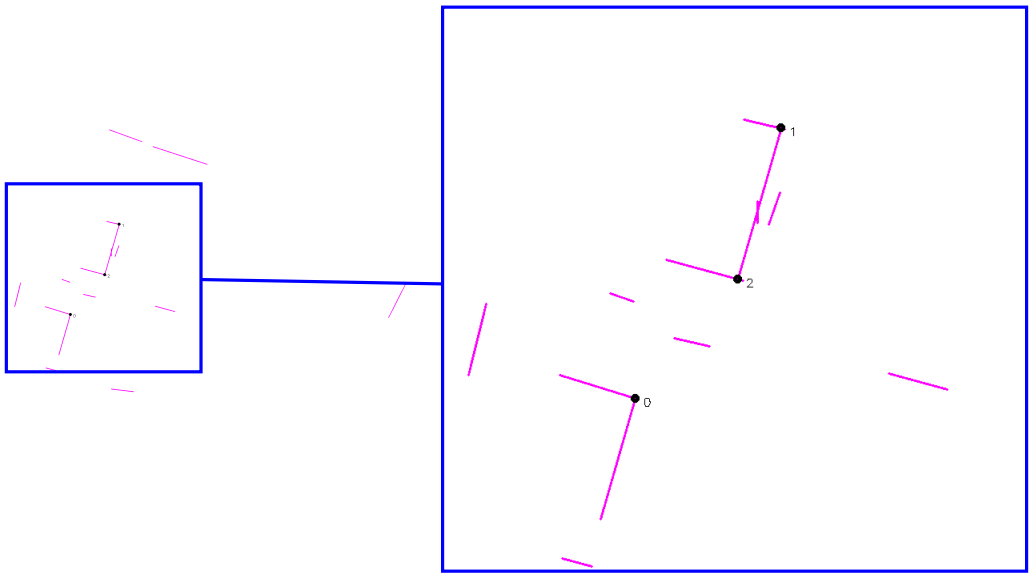
The intersections detected in the source image and the target image are now the set of points $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively, as described in section 3.4. Figure 4.4 shows $\boldsymbol{P}$, and Figure 4.5 shows $\boldsymbol{Q}$.

4.1.5 Identification of possible correspondences between angles detected

The angles of the intersections in the source image and the target image are compared, and if they are similar to each other, they represent a possible match.

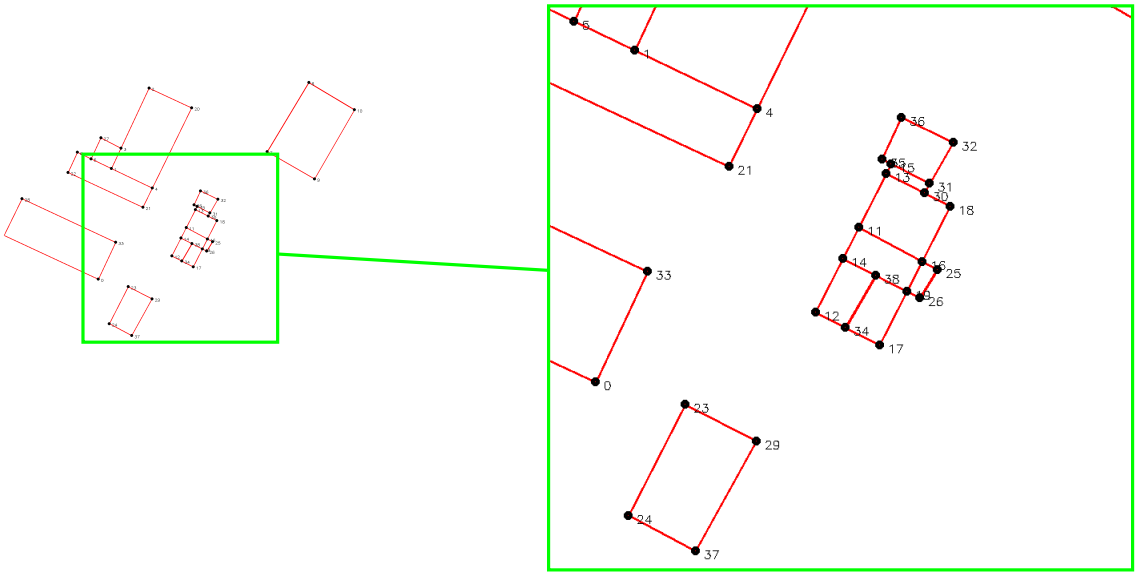
More formally, there is a matrix $\boldsymbol{C} \in \mathbb{R}^{N \times M}$ that represents the possible correspondences. That means that $c_{ij} = 1$ if the angle i in the source image is similar to the angle j in the target image. Otherwise, $c_{ij} = 0$.

If the point cloud and the model are supposed to be close to each other (not necessarily coarsely registered), the possible matches could be further reduced by setting $c_{ij} = 0$ when the euclidean distance between p_i and q_j is greater than a selected threshold.



(a) (b)

Figure 4.4: Source lines and intersections.



(a) (b)

Figure 4.5: Target lines and intersections.

4.1.6 Alignment of both projections onto the xy-plane

The Mixed Integer Linear Program of section 3.5 is then solved using the information contained in $\boldsymbol{C}$, i.e., the possible correspondences between angles. The new objective is to maximize:

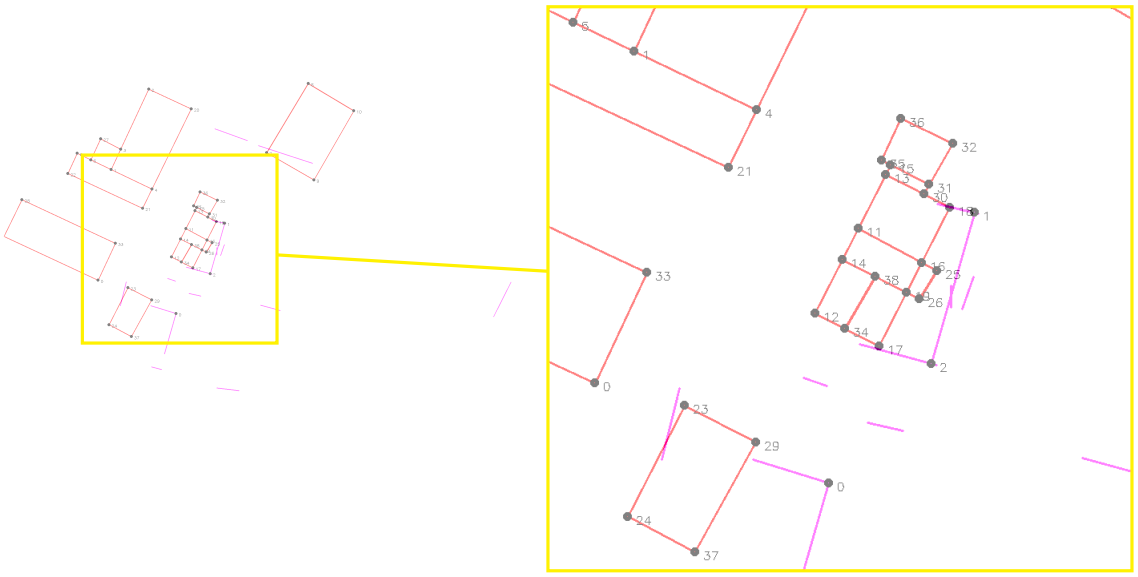
$$\sum_{i}^{N}\sum_{j}^{M} x_{ij}c_{ij} \tag{4.1}$$

subject to the constraints presented in section 3.5. With this change in the objective, correspondences of intersections with different angles are excluded from the Mixed Integer Linear Program, i.e., the number of possible correspondences between intersections is reduced. Therefore, the computation time is significantly reduced too.

The result of the Mixed Integer Linear Program is a transformation $\boldsymbol{T}$ that aligns the source image with the target image and a matrix $\boldsymbol{X}$ representing the correspondences between $\boldsymbol{P}$ and $\boldsymbol{Q}$.

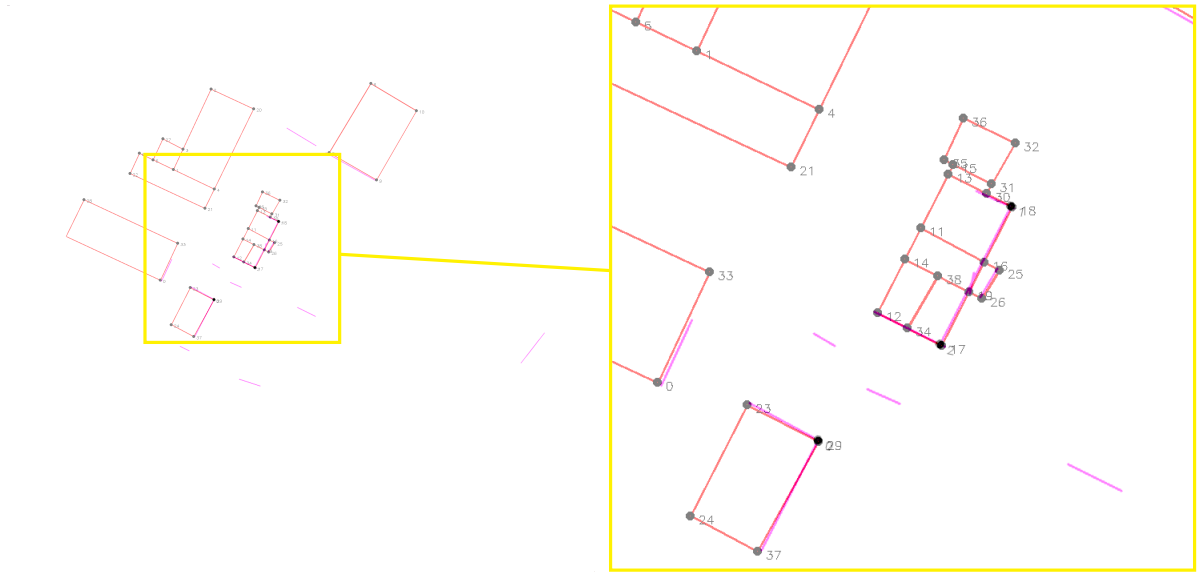
Figure 4.6 shows the initial position of the source image and the target image. Figure 4.7 shows the result after the transformation of the source image with $\boldsymbol{T}$.

The source image and the target image are the source's projection and the target's projection onto the xy-plane, respectively. Therefore, $\boldsymbol{T}$ is used to align the source and the target on the x-axis and the y-axis by expanding it into the $SE(3)$.



(a) (b)

Figure 4.6: Initial position of the source image and the target image.



(a) (b)

Figure 4.7: Source image and target image aligned.

4.1.7 Alignment on the z-axis

Sometimes a translation across the z-axis is needed to align the source and the target completely. The correspondences found in subsection 4.1.6 can be used to find the appropriate translation across the z-axis.

If $x_{ij} = 1$ then the points p_i and q_j are used in the following way. The point p_i is used to find the lowest point p_i^l in the source with x- and y-coordinates similar to those of p_i. And the point q_j is used to find the lowest point q_j^l in the source with x- and y-coordinates similar to those of q_j. Then, the difference along the z-axis between p_i^l and q_i^l is computed. Finally, an average of all the differences computed is the translation t_z, which is then integrated into the transformation $\boldsymbol{T}$.

4.2 Implementation details

The solution was implemented with Python. Therefore, the standard library "json" was used to read the CityGML models. Other third-party libraries were used too. Numpy [23] was employed to compute array operations, Open3d [46] for the visualization of 3D models and point clouds, OpenCV [10] to process images, PyDIP to filter the 2D projection of the model and make the lines more visible, and Python-MIP [38] to solve the Mixed Integer Linear Program described in subsection 4.1.6.

The parameter ϵ_d used in Equation 3.5, Equation 3.6, Equation 3.7, and Equation 3.8 was set to 20 cm, and the parameter b was set to two times the maximum distance between points of the source image and points of the target image. Moreover, the parameter ϵ_l was set to $\epsilon_l \geq 2\sqrt{3}\epsilon_d$.

The upper and lower bounds of $\boldsymbol{T}$ showed in Equation 3.10 were set as follows: $\delta_1^- = 0.7$, $\delta_1^+ = 1.3$, $\delta_2^- = -0.3$, $\delta_2^+ = 0.3$, $t_x^- = -b/2$, $t_x^+ = b/2$, $t_y^- = -b/2$, and $t_y^+ = b/2$. However, these bounds can

be changed to allow a different transformation (different rotation or different translation). Because the universities that provide the point cloud data agreed to send it in a position and orientation close to the model's position, the chosen parameters allow only a small rotation and a small translation.

5

Results and Evaluation

In this chapter, the results of the registration of the available data using the solution presented in chapter 4 will be presented and discussed. The data was visualized and evaluated with the use of the 3D visualization provided by Open3D.

The running times presented in Table 5.1 were taken on an Intel Core i7-1065G7 3.9GHz with 16 GB RAM. The most time-consuming task with the CityGML model is the detection of line segments. In comparison, the most time-consuming task with the PLY model is the alignment of both projections onto the xy-plane (the Mixed Integer Linear Program).

These running times are obtained with the assumption that the position of the point cloud is not far away from the position of the 3D model, as described in subsection 4.1.5. If this assumption is not considered, then the Mixed Integer Linear Program's running time is increased to 6.682 seconds for the CityGML model and 32.906 seconds for the PLY model.

Task \ Data	CityGML model	PLY model
Projection onto the xy-plane	0.607 s	1.33 s
Detection of line segments	3.198 s	3.662 s
Detection of line intersections and their angles	0.226 s	0.229 s
Identification of possible correspondences	0.005 s	0.017 s
Alignment of both projections onto the xy-plane	2.904 s	14.359 s
Alignment on the z-axis	0.421 s	1.859 s

Table 5.1: Running time of the registration.

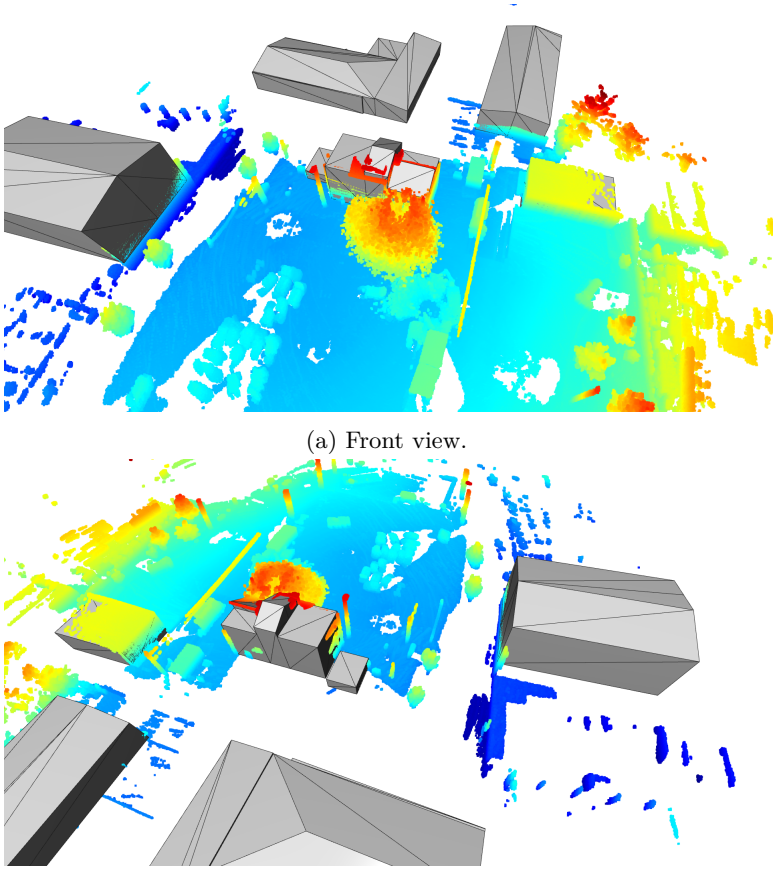
5.1 Registration of a CityGML model with a point cloud

The initial pose of the CityGML model and the point cloud can be found in Figure 3.1. Figure 3.1a shows a front view of the fire exercise building in Dortmund, while Figure 3.1b shows a back view. The point cloud is just rotated around the z-axis and translated across the x- and y-axis. It is also slightly translated across the z-axis, but this is not easy to visualize.

After the registration process, the resulting transformation is

$$\boldsymbol{T_1} = \begin{bmatrix} 0.987 & -0.197 & 0.0 & -4.994 \\ 0.197 & 0.987 & 0.0 & -2.636 \\ 0.0 & 0.0 & 1.0 & -0.207 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

and the results after applying this transformation to the point cloud are shown in Figure 5.1. In the transformation, one can observe a rotation around the z-axis and a translation across the x-, y-, and z-axis.



(a) Front view.

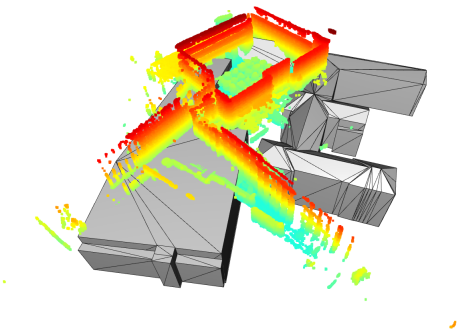
(b) Back view.

Figure 5.1: CityGML model and its corresponding point cloud after registration.

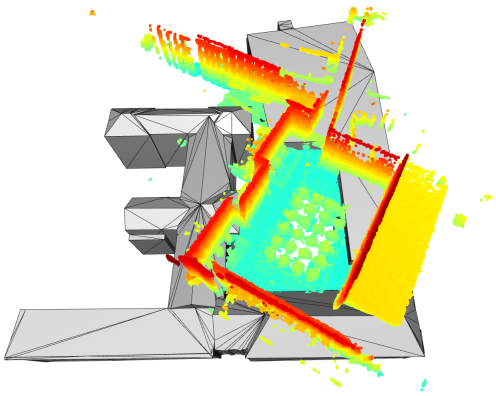
The alignment of the CityGML model with the point cloud can be seen in the fire exercise building (the one in the center of the image) and the right building in Figure 5.1a. The walls and the ceilings are correctly overlapped in both buildings. There is only a slight misalignment in the center part of the fire exercise building's ceil because the CityGML model is outdated.

5.2 Registration of a PLY model with a point cloud

The second available data represents a part of the Technical University of Darmstadt. The 3D model is given in ply format, but this does not make any difference in the visualization. Figure 5.2 shows two different views of the initial pose of the 3D model and its corresponding point cloud. This time, the point cloud is translated across the x-, y-, and z-axis and rotated around the z-axis.



(a) Frist view.



(b) Second view.

Figure 5.2: PLY model and its corresponding point cloud before registration.

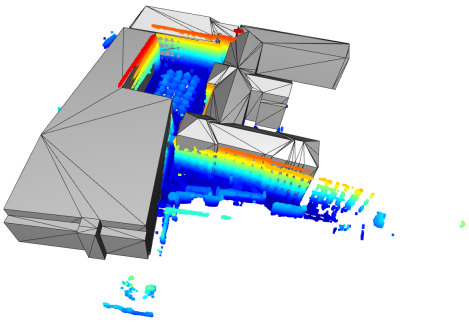
After the registration process, the resulting transformation is

$$\boldsymbol{T_2} = \begin{bmatrix} 0.882 & -0.455 & 0.0 & 20.504 \\ 0.455 & 0.882 & 0.0 & -21.645 \\ 0.0 & 0.0 & 1.0 & -19.254 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

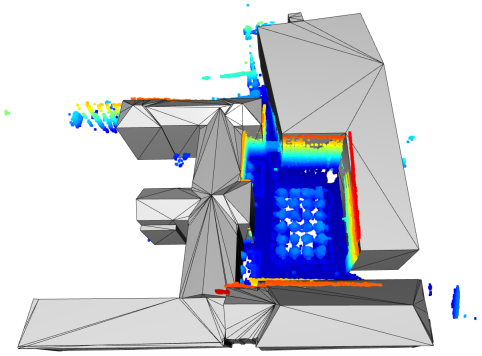
and the results after applying this transformation to the point cloud are shown in Figure 5.3.
In the transformation, one can observe a rotation around the z-axis and a translation across the x-, y-,

and z-axis. This time, the translation across the z-axis is considerably larger than the one in $\boldsymbol{T_1}$.

In Figure 5.3, one can see that the walls are correctly aligned. Unfortunately, the point cloud does not contain any scans of the ceilings, and it is not clear whether the alignment in the z-axis is correct or not. However, the point cloud contains the ground information of the center area between the buildings, and one can see that it is aligned with the floor of the buildings.



(a) First view.



(b) Second view.

Figure 5.3: PLY model and its corresponding point cloud after registration.

5.3 Discussion

The results presented in the two past sections highly depend on the step described in subsection 4.1.4, i.e., on detecting lines and their intersections. If the number of intersections detected is poor, then the registration results could be wrong. Furthermore, the accuracy of the registration result depends on the accuracy with which the position of the intersections is detected. Therefore, if no lines and no intersections are detected, then no registration is possible.

The different running times presented in Table 5.1 could be due to the size of the point clouds. The point cloud corresponding to the CityGML model contains 1178841 points before being downsampled and

1159064 points after being downsampled. In contrast, the point cloud corresponding to the PLY model contains 3724578 points before being downsampled and 2805065 points after being downsampled.

Furthermore, the number of intersections detected in the projections onto the xy-plane, both of the models and the point clouds, differs too. The number of intersections detected in the projection of the CityGML model and its corresponding point cloud is 3 and 39, respectively. In comparison, the number of intersections detected in the projection of the PLY model and its corresponding point cloud is 10 and 43, respectively. Therefore, the Mixed Integer Linear Program takes longer for the PLY model.

6

Conclusions

The results obtained from the proposed solution are successful in the provided data. Although the proposed method provides only a coarse registration, it is significant for the A-DRZ project's proposes. The registration method is ready to be included in the A-DRZ software of the Fraunhofer IAIS, and whenever there are new point clouds, the method could be further tested and improved.

6.1 Contributions

This work's main contribution is a coarse registration method for a 3D model and a point cloud that converts the 3D registration task into a 2D registration task and works based on the intersection of lines and their angle information. Furthermore, the solution presented can be adapted to be used with different kinds of 2D features.

In addition to this work's scientific contributions, it also contributes to the A-DRZ project as an automatic registration method for their visualization system, helping rescue services to obtain an overview of the current state of possible dangerous situations.

6.2 Future work

The detection of lines and their intersection is the most crucial part of the solution proposed since these are the features used for the finding of correspondences conducted by the Mixed Integer Linear Program. Therefore, improving this task's performance will benefit the registration's global performance, although it is not easy.

Another improvement that could be made is to increase the information in the angles detected. Not just the position of the angle can be used in the Mixed Integer Linear Program, but also the orientation of the angle, i.e., the orientation of the sum of the line segments that form the angle. Moreover, other kinds of features can be included. For example, the lines or even other corners that are not necessarily intersections of lines.

Two additional options could be tried to improve the system's performance in the future. Firstly, one could work with 3D features and find directly a 3D transformation that aligns the CityGML model and the point cloud. Secondly, a Branch-and-Bounding approach [8, 11, 12, 13] could be applied to search for a transformation that aligns the projections of the CityGML model and the point cloud onto the xy-plane or to search such a transformation directly with 3D features such as planes, lines, or corners.

Finally, the registration results can be refined by applying a local registration method. The local registration can be done with a Linear Program relaxation as presented by Goebbels et al. [20, 19] or with the ICP variation presented by Goebbels et al. [21], which makes direct use of the CityGML model's surfaces. This refinement could bring the point cloud closer to the CityGML model.

References

[1] Kompetenzzentrum für Rettungsrobotik. URL https://rettungsrobotik.de/.

[2] Bayerische Vermessungsverwaltung - Produkte - 3D-Produkte - 3D-Gebäudemodell. URL https://www.ldbv.bayern.de/produkte/3dprodukte/3d.html.

[3] Deutscher Feuerwehrverband: Feuerwehr-Statistik. URL https://www.feuerwehrverband.de/presse/statistik/. Accessed on 05/31/2020.

[4] Cp_Landesvermessung_3DDaten — Hessische Verwaltung für Bodenmanagement und Geoinformation. URL https://hvbg.hessen.de/geoinformation/landesvermessung/geotopographie/3d-daten.

[5] 3D-Gebäudemodell LoD2 - NRW. URL https://www.opengeodata.nrw.de/produkte/geobasis/3dg/lod2{_}gml/.

[6] SIM.tud : FG Simulation, Systemoptimierung und Robotik / FB Informatik / TU Darmstadt. URL https://www.sim.informatik.tu-darmstadt.de/index/.

[7] University of Bonn, Computer Science VI, Autonomous Intelligent Systems. URL http://www.ais.uni-bonn.de/.

[8] Jean Charles Bazin, Hongdong Li, In So Kweon, Cédric Demonceaux, Pascal Vasseur, and Katsushi Ikeuchi. A branch-and-bound approach to correspondence and grouping problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1565–1576, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2012.264.

[9] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, feb 1992. ISSN 1939-3539. doi: 10.1109/34.121791.

[10] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[11] Thomas M. Breuel. Implementation techniques for geometric branch-and-bound matching methods. *Computer Vision and Image Understanding*, 90(3):258–294, jun 2003. ISSN 10773142. doi: 10.1016/S1077-3142(03)00026-2.

[12] Mark Brown, David Windridge, and Jean Yves Guillemaut. Globally optimal 2D-3D registration from points or lines without correspondences. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 2111–2119, 2015. ISBN 9781467383912. doi: 10.1109/ICCV.2015.244.

[13] Mark Brown, David Windridge, and Jean Yves Guillemaut. A family of globally optimal branch-and-bound algorithms for 2D–3D correspondence-free registration. *Pattern Recognition*, 93:36–54, sep 2019. ISSN 00313203. doi: 10.1016/j.patcog.2019.04.002. URL `https://doi.org/10.1016/j.patcog.2019.04.002`.

[14] Subhasis Chaudhuri, Shankar Chatterjee, Norman Katz, Mark Nelson, and Michael Goldbaum. Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE Transactions on Medical Imaging*, 8(3):263–269, sep 1989. ISSN 02780062. doi: 10.1109/42.34715.

[15] Li Ding and Chen Feng. DeepMapping: Unsupervised Map Estimation From Multiple Point Clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[16] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3D Point Cloud Registration for Localization Using a Deep Neural Network Auto-Encoder. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] 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. ISSN 0001-0782. doi: 10.1145/358669.358692. URL `https://doi.org/10.1145/358669.358692`.

[18] Andrew W. Fitzgibbon. Robust registration of 2D and 3D point sets. *Image and Vision Computing*, 21(13):1145–1153, 2003. ISSN 0262-8856. doi: https://doi.org/10.1016/j.imavis.2003.09.004. URL `http://www.sciencedirect.com/science/article/pii/S0262885603001835`.

[19] Steffen Goebbels and Regina Pohle-Froehlich. Line-based Registration of Photogrammetric Point Clouds with 3D City Models by Means of Mixed Integer Linear Programming. In *Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, 2018.

[20] Steffen Goebbels, Regina Pohle-Fröhlich, and Philipp Kant. A Linear Program for Matching Photogrammetric Point Clouds with CityGML Building Models. In Natalia Kliewer, Jan Fabian Ehmke, and Ralf Borndörfer, editors, *Operations Research Proceedings 2017*, pages 129–134, Cham, 2018. Springer International Publishing. ISBN 978-3-319-89920-6.

[21] Steffen Goebbels, Regina Pohle-Fröhlich, and Philipp Pricken. Iterative Closest Point Algorithm for Accurate Registration of Coarsely Registered Point Clouds with CityGML Models. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 201–208, 2019.

[22] Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, and Karl-Heinz Häfele. *OGC City Geography Markup Language (CityGML) Encoding Standard*. Open Geospatial Consortium, 2.0.0 edition, 2012.

[23] Charles R. Harris, K. Jarrod Millman, Stefan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández

del Río, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, sep 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

[24] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[25] Xiaoshui Huang, Guofeng Mei, and Jian Zhang. Feature-metric Registration: A Fast Semi-supervised Approach for Robust Point Cloud Registration without Correspondences. *arXiv preprint arXiv:2005.01014*, 2020.

[26] Changmin Kim, Joohyuk Lee, Minwoo Cho, and Changwan Kim. Fully automated registration of 3D CAD model with point cloud from construction site. In *28th International Symposium on Automation and Robotics in Construction, Seoul, Korea*, pages 917–922, 2011.

[27] Changmin Kim, Hyojoo Son, and Changwan Kim. Fully automated registration of 3D data to a 3D CAD model for project progress monitoring. *Automation in Construction*, 35:587–594, 2013. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2013.01.005. URL `http://www.sciencedirect.com/science/article/pii/S0926580513000071`.

[28] Weimin Li and Pengfei Song. A modified ICP algorithm based on dynamic adjustment factor for registration of point cloud and CAD model. *Pattern Recognition Letters*, 65:88–94, 2015. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2015.07.019. URL `http://www.sciencedirect.com/science/article/pii/S0167865515002287`.

[29] Jun Lu, Wei Wang, Hongxu Shao, and Li Su. Point Cloud Registration Algorithm Fusing of Super 4PCS and ICP Based on the Key Points. In *2019 Chinese Control Conference (CCC)*, pages 4439–4444, 2019. doi: 10.23919/ChiCC.2019.8866059.

[30] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. DeepICP: An End-to-End Deep Neural Network for 3D Point Cloud Registration. *CoRR*, abs/1905.0, 2019. URL `http://arxiv.org/abs/1905.04153`.

[31] Cris Luengo and Wouter Caarls. DIPlib: Quantitative Image Analysis in C++, MATLAB and Python. URL `https://github.com/DIPlib/diplib`. version 2.9.

[32] Nicolas Mellado, Dror Aiger, and Niloy J. Mitra. SUPER 4PCS Fast Global Pointcloud Registration via Smart Indexing. *Computer Graphics Forum*, 33(5):205–215, 2014. doi: 10.1111/cgf.12446.

[33] Dhanya S. Pankaj and Rama Rao Nidamanuri. A robust estimation technique for 3D point cloud registration. *Image Analysis and Stereology*, 35(1):15–28, 2015. ISSN 18545165. doi: 10.5566/ias.1378.

[34] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. Technical report, 2017.

[35] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5099–5108. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7095-pointnet-deep-hierarchical-feature-learning-on-point-sets-in-a-metric-space.pdf`.

[36] Siwen Quan and Jiaqi Yang. Compatibility-Guided Sampling Consensus for 3-D Point Cloud Registration. *IEEE Transactions on Geoscience and Remote Sensing (2020)*, pages 1–13, 2020. ISSN 1558-0644. doi: 10.1109/TGRS.2020.2982221.

[37] Shizu Sakakubara, Yuusuke Kounoike, Yuji Shinano, and Ikuko Shimizu. Automatic range image registration using mixed integer linear programming. In *Asian Conference on Computer Vision*, pages 424–434. Springer, 2007.

[38] Haroldo Gambini Santos and Túlio Toffolo. Python MIP (Mixed-Integer Linear Programming) Tools. URL `https://github.com/coin-or/python-mip`. version 1.13.0.

[39] Vinit Sarode, Xueqian Li, Hunter Goforth, Yasuhiro Aoki, Animesh Dhagat, Rangaprasad Arun Srivatsan, Simon Lucey, and Howie Choset. One Framework to Register Them All: PointNet Encoding for Point Cloud Alignment. *arXiv preprint arXiv:1912.05766*, 2019.

[40] Aleksandr V. Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. *Robotics: science and systems*, 2:435, 2009.

[41] Yue Wang and Justin M. Solomon. Deep Closest Point: Learning Representations for Point Cloud Registration. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[42] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5):Article 146, 2019. ISSN 15577368.

[43] Thomas Windheuser, Ulrich Schlickwei, Frank R. Schimdt, and Daniel Cremers. Large-Scale Integer Linear Programming for Orientation Preserving 3D Shape Matching. *Computer Graphics Forum*, 30 (5):1471–1480, 2011. doi: 10.1111/j.1467-8659.2011.02021.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2011.02021.x`.

[44] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2241–2254, nov 2016. ISSN 1939-3539. doi: 10.1109/TPAMI.2015.2513405.

[45] Zi Jian Yew and Gim Hee Lee. 3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *European Conference on Computer Vision (ECCV) 2018*, pages 630–646, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01267-0.

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