FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Registration of labeled point clouds
Student:	Tomáš Laurin
Supervisor:	Ing. Jan Glaser
Study program:	Informatics
Branch / specialization:	Artificial Intelligence 2021
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2024/2025

Instructions

A registration of two point clouds is a fundamental task in 3D reconstruction.

The focus of the thesis is to create a pipeline for registration of labeled point clouds, and explore the possibilities of registration using the label information or different distance metrics.

1) survey the topic and existing solutions
2) create a pipeline for segmentation and labeling of point clouds
3) explore the possibilities of registration using the label information of the point clouds
4) experiment with using different distance metrics in the ICP registration algorithm
5) choose an appropriate metric for evaluating these experiments
6) discuss the results of these experiments

References:
1) TRUONG, Giang, et al. Fast point cloud registration using semantic segmentation. In: 2019 Digital Image Computing: Techniques and Applications (DICTA). IEEE, 2019. p. 1-8.
2) YANG, Jiaolong, et al. Go-ICP: A globally optimal solution to 3D ICP point-set registration. IEEE transactions on pattern analysis and machine intelligence, 2015, 38.11: 2241-2254.
3) JOST, Timothée; HÜGLI, Heinz. Fast ICP algorithms for shape registration. In: Pattern Recognition: 24th DAGM Symposium Zurich, Switzerland, September 16–18, 2002 Proceedings 24. Springer Berlin Heidelberg, 2002. p. 91-99.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 22 November 2023 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

4) HUANG, Jing; YOU, Suya. Point cloud labeling using 3d convolutional neural network. In: 2016 23rd International Conference on Pattern Recognition (ICPR). IEEE, 2016. p. 2670-2675.
5) ZHANG, Liqiang, et al. Large-scale urban point cloud labeling and reconstruction. ISPRS Journal of Photogrammetry and Remote Sensing, 2018, 138: 86-100.
6) WIRTH, Florian, et al. Pointatme: efficient 3d point cloud labeling in virtual reality. In: 2019 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2019. p. 1693-1698.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 22 November 2023 in Prague.

Bachelor's thesis

REGISTRATION OF LABELED POINT CLOUDS

Tomáš Laurin

Faculty of Information Technology
Katedra aplikované matematiky
Supervisor: Ing. Jan Glaser
May 12, 2024

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis: Laurin Tomáš. *Registration of labeled Point Clouds.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Contents

List of Figures

List of Tables

List of Algorithms

Firstly, I thank my supervisor, Ing. Jan Glaser, for his advice, guidance, and time spent on this work. Next, I thank my family and girlfriend, who supported me throughout my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Praze on May 12, 2024

Abstract

This thesis delves into developing a pipeline for registering partially overlapping large-scale point clouds and measuring the influence of semantic label information in the registration process. The result of this work is a method that uses the additional information obtained from convolutional neural networks to partition the point clouds and align local structures using principal component analysis. This method was tested against traditional methods on publicly available datasets. Experiments have shown that aligning smaller subsets proved much more stable and accurate than traditional approaches.

Keywords ICP, point cloud, point set registration, principal component analysis, semantic segmentation

Abstrakt

Tato práce se zabývá vývojem metody pro registraci částečně překrývajících se mračen bodů a zjištění vlivu sémantických informací v procesu registrace. Výstupem práce je metoda, která využívá informace získané z konvolučních neuronových sítí pro rozdělení mračen bodů a zarovnání lokálních struktur pomocí analýzy hlavních komponent. Tato metoda byla testována proti tradičním metodám na veřejně dostupných datasetech. Experimenty ukázaly, že registrace menších celků vedla k mnohem stabilnějším a přesnějším výsledkům než tradiční přístupy.

Klíčová slova analýza hlavních komponent, ICP, mračno bodů, registrace mračen bodů, sémantická segmentace

List of abbreviations

LiDAR	Light Detection and Ranging
3D	Three-Dimensional
ICP	Iterative Closest Point
KNN	K-Nearest Neighbors
PCA	Principal Component Analysis
DBSCAN	Density-Based Spacial Clustering of Application with Noise
FFN	Feedforward Neural Network
CNN	Convolutional Neural Network
ReLu	Rectified Linear Unit
MLP	Multi-Layer Perceptron
OA	Overall Accuracy
IoU	Intersection over Union
CPU	Central Processing Unit
GPU	Graphical Processing Unit
RAM	Random Access Memory
RMSE	Root Mean Square Error
RANSAC	Random Sample Consensus
FPFH	Fast Point Feature Histograms

Introduction

The accessibility and affordability of 3D scanning technologies, such as LiDAR sensors, have seen a notable rise [1, p. 3]. This trend has led to their integration into a wide array of applications, including autonomous vehicles [2, 3], robotics [4], archaeology [5], and object scanning for computer software [6]. LiDAR sensors generate data in the form of point clouds, representing real-world 3D objects with numerous data points. Such a 3D scan of a chair is shown in Fig. 1.2.

In many scenarios, multiple scans acquired from different camera angles contribute to the scene reconstruction process [7]. For instance, in object scanning for computer graphics, several scans are taken from various angles and merged to reconstruct the complete object. This process, known as registration, aligns two or more point clouds into a unified coordinate system. However, while effective in many cases, traditional registration methods like Iterative Closest Point (ICP) often struggle with noisy, large-scale real-world data [8, 9], resulting in sub-optimal final point set alignment. Therefore, more robust and efficient approaches are needed.

Segmentation of point clouds is another critical task in 3D data analysis. At its core, segmentation refers to identifying objects within a scene. It is usually done using convolutional neural networks. Previously, when segmenting point clouds, they would often be converted into structured formats, such as 2D images [10] or voxel grids [11], that can be fed into the neural network. However, recent advancements in 3D deep learning have introduced sophisticated architectures capable of directly processing unstructured and unordered point cloud data [12, 13, 14].

This thesis delves into developing a pipeline that utilizes state-of-the-art deep-learning techniques for label acquisition to improve registration accuracy through dataset partitioning. The proposed model is evaluated on large-scale datasets and compared against traditional registration approaches.

Chapter 1
Analysis

This chapter describes the individual parts and technologies needed for our pipeline. First, we discuss scanning technologies and the acquisition of point clouds. Then, we discuss point cloud properties and the essential algorithmic concepts, focusing on processing 3D data and fundamental machine learning algorithms. Lastly, we explore existing solutions to the problem and present our approach.

1.1 LiDAR

LiDAR, which stands for light detection and ranging, is a remote sensing technology that calculates distances by emitting laser beams toward an object and measuring the time it takes for the reflection to return. This time-of-flight data is converted into distance measurement, each corresponding to a specific point in space. A single LiDAR device can emit thousands of pulses per second, enabling it to gather data over a large area quickly. These distance measurements together form a point cloud, representing the 3D model of the scanned area.

LiDAR sensors are commonly mounted on drones or revolving platforms, enabling them to collect data from multiple angles. An example of a LiDAR schematic on a rotating platform is depicted in Fig. 1.1.

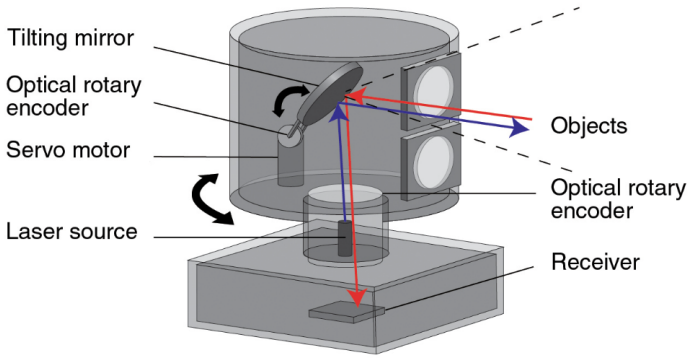


Figure 1.1 Schematic of a LiDAR sensor with its core components labeled: laser source that emits the beam, a receiver that detects the reflected light, a tilting mirror for directing the laser, and two optical rotary encoders connected to a servo motor for angular measurement of the device's orientation [15].

1.2 Point Clouds

Point clouds are a simple data representation that store information about 3D objects as a set of points; an example of a scanned chair can be seen in Fig. 1.2. Each point consists of coordinates (x, y, z) and possibly other features like color, surface normals, or semantic labels.

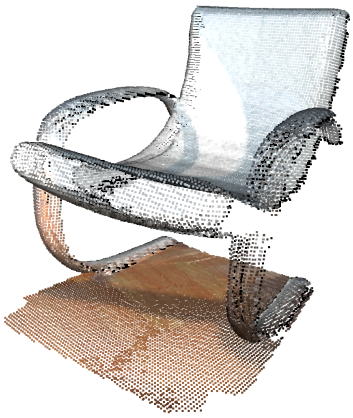


Figure 1.2 Point cloud representation of a chair scan [15].

Point clouds have unique properties that are important to consider during analysis and processing. Among the most prominent properties is that point clouds are unordered, invariant under transformation, and points interact with neighboring points [12]. A brief description of the most prominent properties is provided below:

- **Unordered**: Unlike images, point clouds are a set of points without any specific order. This means the data will remain unchanged if we shuffle the points. As a result, the algorithms cannot assume any order in the representation and must be invariant to every possible permutation.
- **Invariance under transformation**: The point cloud represents the same structure regardless of orientation and position. Rotating, scaling, or moving a point cloud does not change the information inside the point cloud.
- **Interaction among neighboring points**: Similarly to images, the neighborhood of a point in a point cloud holds valuable information for object detection, for example, the structure and shape of parts of the scan. For instance, in a densely populated point cloud representing a forest, the proximity and arrangement of points can help identify individual trees, their canopy sizes, and the underbrush.

1.3 Data Structures and algorithms

The density and size of point clouds vary significantly amongst different datasets. However, larger scans of cities or ground areas have millions of points. When working with such large data, implementing efficient algorithms is paramount. Algorithms for registration often require operations such as finding the nearest neighbors of a point, which would be computationally intensive if we had to check each point due to the unordered property of point clouds. Therefore, various data structures and algorithms have been developed to address these challenges for efficient data processing. We discuss the ones we use in our project.

1.3.1 K-d Tree

A k-d tree, short for k-dimensional tree, is a space-partitioning data structure stored as a binary tree that organizes points in a multidimensional space and is designed to have fast search queries. The data structure works by continuously dividing the space along each axis to halve the number of points. By dividing the points into nested half-spaces, the tree reduces the average search time in the point cloud with n points to $log(n)$ time rather than n time [16]. A 2D visualization of a k-d tree and its appropriate data representation can be seen in Fig. 1.3 and a 3D representation can be seen in Fig. 1.4.

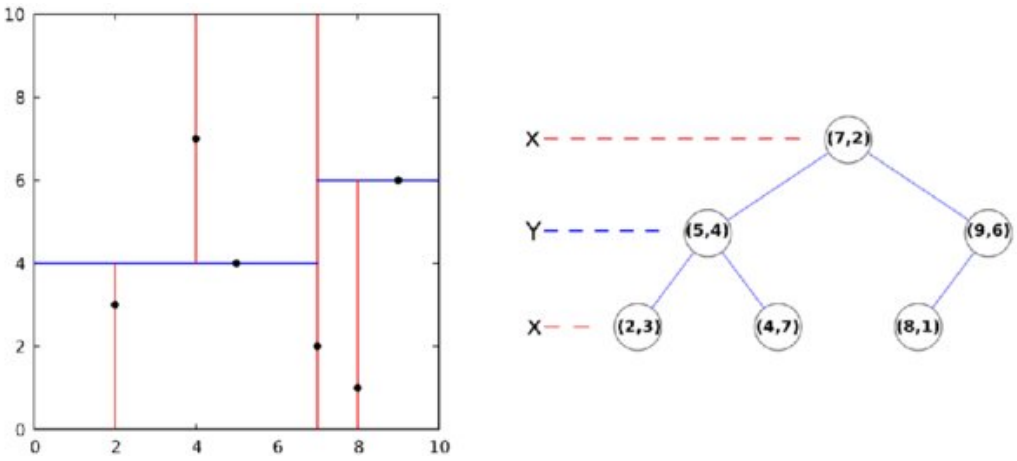


Figure 1.3 Visualisation of a 2D k-d tree showing spatial data partitioning. On the left, a diagram of a k-d tree, showing the space partition with vertical red and horizontal blue lines, and on the right, the corresponding binary tree structure [17, p. 60].

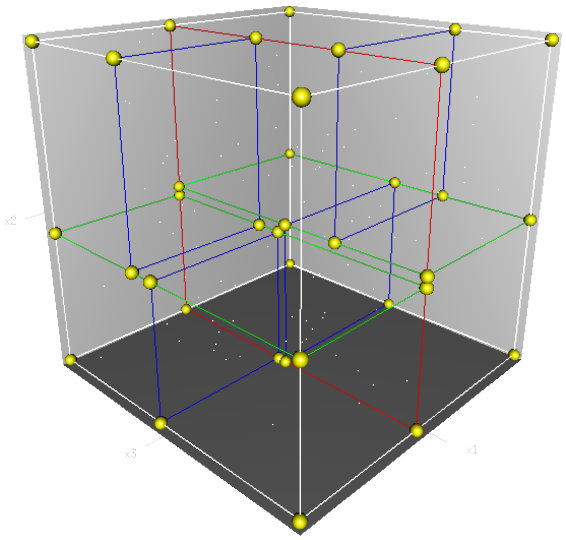


Figure 1.4 Three-dimensional visualization of a k-d tree with points represented by white spheres and partitioning of the space shown by blue, red, and green lines corresponding to divisions along the x, y, and z-axes, respectively [18].

1.3.2 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical method that uses an orthogonal transformation to convert a set of possibly correlated variables into linearly uncorrelated variables called principal components. It is most commonly used in dimensionality reduction, where PCA can be used to identify the main components of the data and remove the less dominant ones [19]. In Fig. 1.5, we can see the principal components of a sample 2D data.

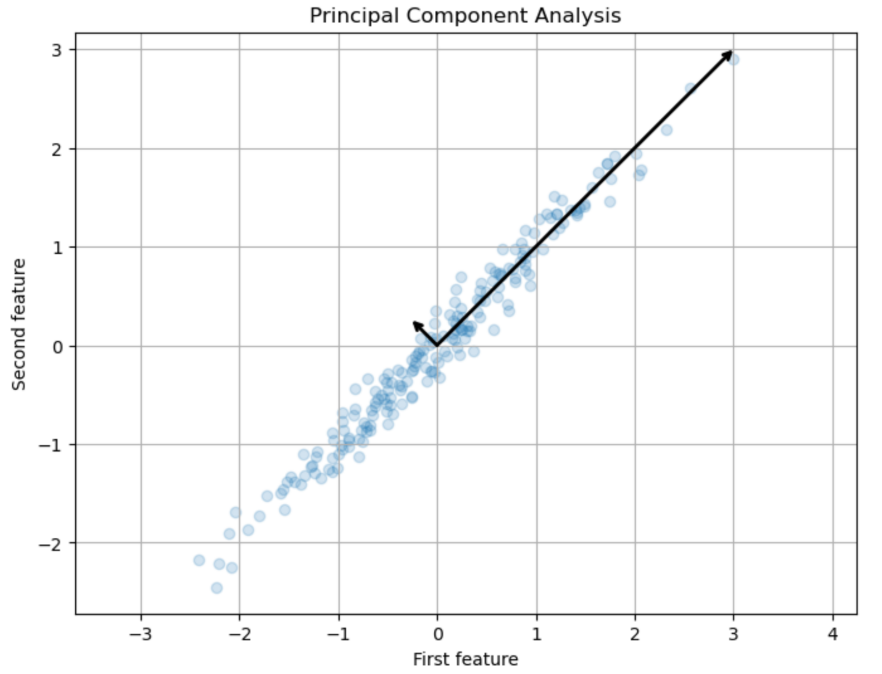


Figure 1.5 A visual representation of PCA, where the blue dots represent a dataset and vectors show the principal components, highlighting the sub-spaces characterized by the least variance within the dataset.

In the context of point clouds, PCA can be used to identify the main directions of data variation, which can help us understand the underlying structure of the point cloud data. By analyzing the principal components, one can infer the dominant spatial orientation [20], which can be critical for alignment and registration. Alignment using PCA is visually represented in Fig. 2.10 – 2.14.

1.3.3 DBSCAN

Density-Based Spatial Clustering of Application with Noise (DBSCAN) is a data clustering algorithm. When given a set of points in space, DBSCAN groups points that are closely packed together and removes points that lack neighbors, marking them outliers. The algorithm identifies points in high-density areas and marks them as core points. These core points are iteratively expanded by including directly reachable points in their neighborhood. If a point lacks neighbors, it is marked as an outlier [21]. The process is visible in Fig. 1.6. The use of DBSCAN on real-world data can be seen in Fig. 1.7.

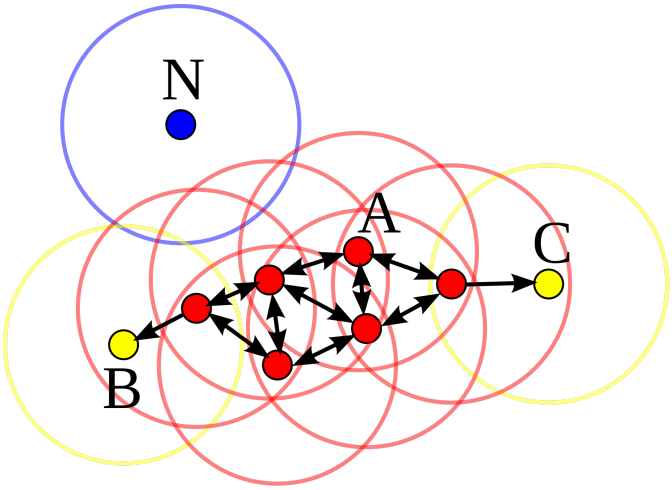


Figure 1.6 Example of a DBSCAN result: core points are colored red, reachable points are yellow, and outliers blue. The circles mark the search radius for each point [22].

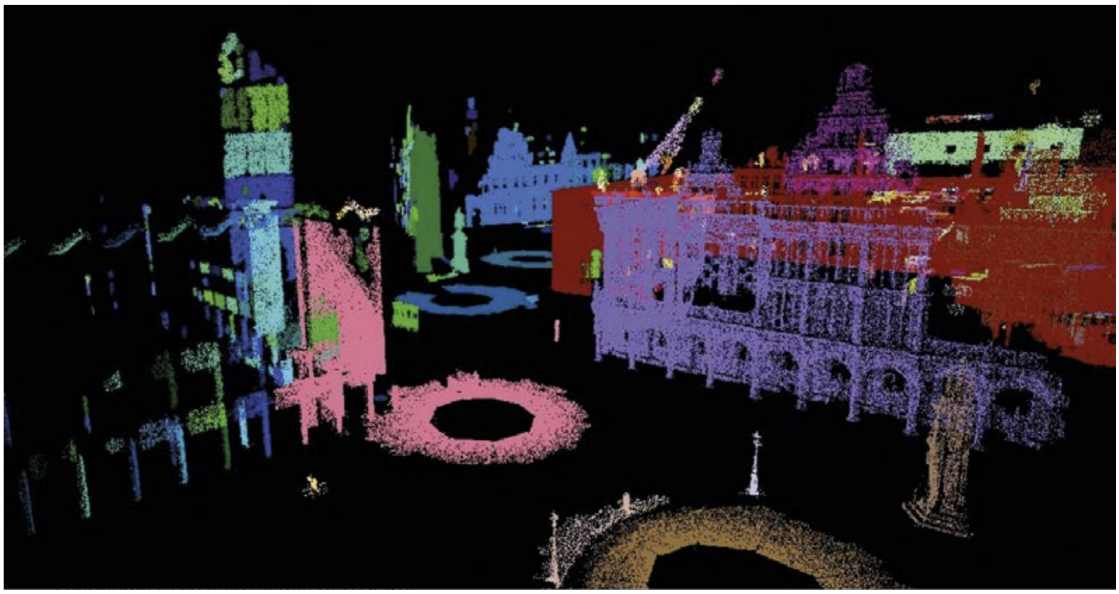


Figure 1.7 A visual representation showcases DBSCAN clustering applied to a real-world outdoor dataset [23]. Each distinct color represents a unique cluster identified by DBSCAN. Points that were marked outliers were removed from the image.

1.3.4 RANSAC

Random sample consensus (RANSAC) is an iterative method used to estimate parameters of a mathematical model from a set of observed data points containing outliers. It is effective in scenarios where a simple model can describe most of the data, but outliers make it hard to create such a model. RANSAC iteratively selects random subsets of data points, fitting the model to each subset and evaluating it based on a predefined threshold. Points consistent with the model within the threshold are considered inliers, while points outside the threshold are considered outliers and are discarded.

A simple example is fitting a line to two-dimensional data. Fig. 1.8 illustrates 2D data to which we want to fit a line. Fig. 1.9 then shows the fitted line using RANSAC. A more complex example can be seen in Fig. 1.10, where RANSAC was used on an indoor point cloud to detect surface planes [24].

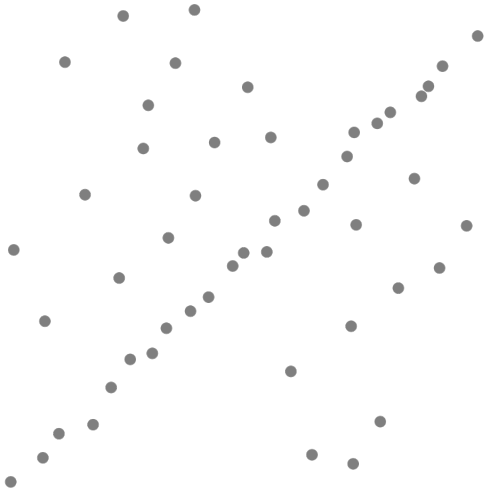


Figure 1.8 A dataset with many outliers for which a line has to be fitted [25].

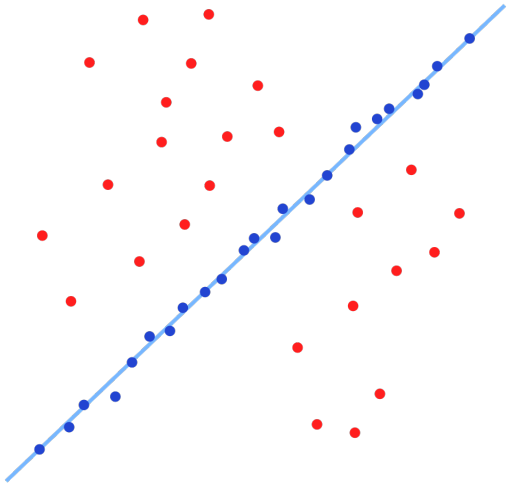


Figure 1.9 Fitted line, which is a result of RANSAC algorithm; The algorithm successfully identifies the inliers (blue points) consistent with the model (blue line) while disregarding the outliers (red points), resulting in an accurate estimation [26].

Figure 1.10 RANSAC used for plane detection on an indoor dataset. A different color marks different planes [24].

1.3.5 Transformation Matrix

Transformation matrix $\mathbb{M}$ is a matrix that transforms one vector space into another. It encapsulates the combined effect of translation vector $\vec{t}$, rotation matrix $\mathbb{R}$, and scaling matrix $\mathbb{S}$ within a single matrix [27]. Such matrix can be seen below:

$$\mathbb{M} = \begin{pmatrix} \mathbb{S}_{00}\mathbb{R}_{00} & \mathbb{R}_{01} & \mathbb{R}_{02} & \vec{t}_0 \\ \mathbb{R}_{10} & \mathbb{S}_{11}\mathbb{R}_{11} & \mathbb{R}_{12} & \vec{t}_1 \\ \mathbb{R}_{20} & \mathbb{R}_{21} & \mathbb{S}_{22}\mathbb{R}_{22} & \vec{t}_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.1}$$

To obtain a transformed point p' in the new coordinate system, we can use the transformation matrix with the point using the following equation:

$$p' = \begin{pmatrix} \mathbb{S}_{00}\mathbb{R}_{00} & \mathbb{R}_{01} & \mathbb{R}_{02} & \vec{t}_0 \\ \mathbb{R}_{10} & \mathbb{S}_{11}\mathbb{R}_{11} & \mathbb{R}_{12} & \vec{t}_1 \\ \mathbb{R}_{20} & \mathbb{R}_{21} & \mathbb{S}_{22}\mathbb{R}_{22} & \vec{t}_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} \tag{1.2}$$

When talking about point sets, we want to find the translation $\vec{t}$ and rotation $\mathbb{R}$ that maps one point cloud to another [28]. Given a source point cloud $A = \{(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n)\}$, target point cloud $B = \{(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_m, y_m, z_m)\}$, rotation matrix $\mathbb{R} \in \mathbb{R}^3$, and translation vector $\vec{t} \in R^3$ we can find the transformation from A to B using the equation:

$$B = \{\mathbb{R} \cdot a + \vec{t} \mid \forall a \in A\} \tag{1.3}$$

Translation vector $\vec{t}$ between set A to B, can be found as: $\vec{t} = \text{centroid}_B - \text{centroid}_A$. The centroid of a set of points is the mean point position, that is, the sum of points divided by the number of points. Given a point cloud $P = \{(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n)\}$ with n points the centroid is given by:

$$(\hat{x}, \hat{y}, \hat{z}) = \left(\frac{1}{n} \sum_{i=0}^{n} x_i, \frac{1}{n} \sum_{i=0}^{n} y_i, \frac{1}{n} \sum_{i=0}^{n} z_i \right) \tag{1.4}$$

Rotation matrix $\mathbb{R}$ can be found using singular value decomposition (SVD) of the familiar covariance matrix of the point clouds. Let A and B represent the source and target point clouds respectively, represented as matrices. We start by centering the point clouds and finding the familiar covariance matrix $\mathbb{H}$:

$$\mathbb{H} = (A - \text{centroid}_A) \cdot (B - \text{centroid}_B)^T \tag{1.5}$$

Next, we can calculate the SVD to decompose the matrix $\mathbb{H}$, with a size of $m \times n$, into a complex unitary matrix $\mathbb{U}$ of size $m \times m$, a rectangular diagonal matrix $\mathbb{S}$ of size $m \times n$, and a complex unitary matrix $\mathbb{V}$ of size $n \times n$.

$$\mathbb{U} \cdot \mathbb{S} \cdot \mathbb{V} = \text{SVD}(\mathbb{H}) \tag{1.6}$$

$\mathbb{U}$, and $\mathbb{V}^T$ represent rotation and reflection of the space, while $\mathbb{S}$ represents the scaling. Finally, the rotation can be found using:

$$\mathbb{R} = \mathbb{V} \cdot \mathbb{U}^T \tag{1.7}$$

1.4 Point Cloud Segmentation

Point cloud segmentation is a part of spatial data analysis. It involves the classification of point clouds into distinct, homogeneous regions, where points within each region share similar properties. This segmentation process plays a pivotal role in automatic scene understanding, enabling the identification and differentiation of various elements within a 3D environment based on shared characteristics. This characteristic, for instance, could be membership in a category such as chair or table, as seen in Fig 1.11 below.

Figure 1.11 Example of semantic segmentation result from a neural network. The input consists of raw indoor point cloud data. The output is a segmented point cloud where different regions within the indoor space are highlighted [12].

1.5 Machine Learning

Machine learning, a subset of artificial intelligence, focuses on developing statistical algorithms that learn from data and make generalizations, allowing them to perform tasks on new, unseen data. In recent years, such algorithms have rapidly gained popularity. They are utilized across various disciplines to recognize objects, particularly in computer vision [29].

1.5.1 Feedforward Neural Network

A feedforward neural network (FNN) is a computational model inspired by biological processes that try to simulate how neurons interact in the human brain. FNN consists of layers of neurons, where information goes from left to right, hence its name [30]. Fig. 1.12 presents a simple neural network outline.

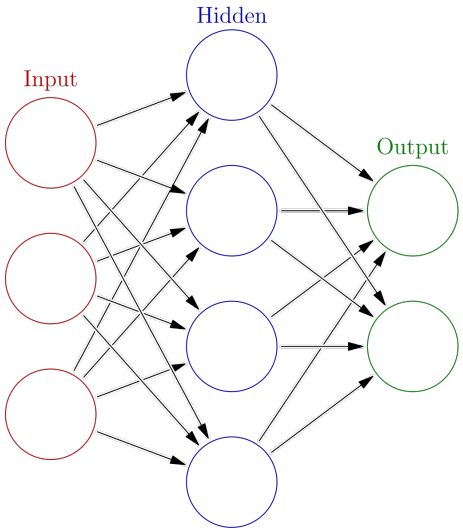


Figure 1.12 Example of a Feedforward Neural Network [31] where the input layer (red) receives the data, the hidden layer (blue) performs the computation, and the output layer (green) produces the final result. Each layer is interconnected, with individual connections characterized by weights representing the significance of their respective interactions. Each circle represents a single neuron.

The structure of a neural network begins with the **input layer**, which receives the raw data. Each neuron in this layer represents a data feature, such as the pixel's intensity in an image. Following the input layer are one or more hidden layers of neurons that carry out computations and feature transformation. Each neuron applies a weighted sum of the input and passes the sum into a nonlinear **activation function**. Activation functions are crucial as they introduce nonlinearity into the network, allowing it to handle linearly non-separable cases. These functions vary, but common examples include **ReLU**, **leaky RELU**, **SELU**, and **tanh** [32]. For their visual representation, refer to Fig. 1.14. Visualization of a single neuron from a neural network can be seen in Fig. 1.13

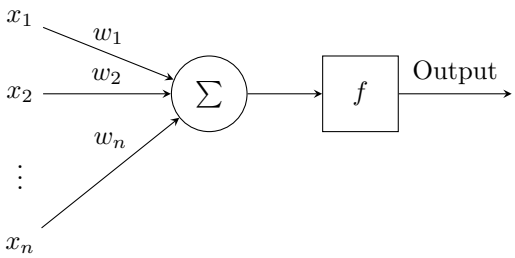


Figure 1.13 Model of a single neuron, also called a perceptron. The inputs marked x are multiplied by weights w and summed together. The result is fed into the activation function f.

.

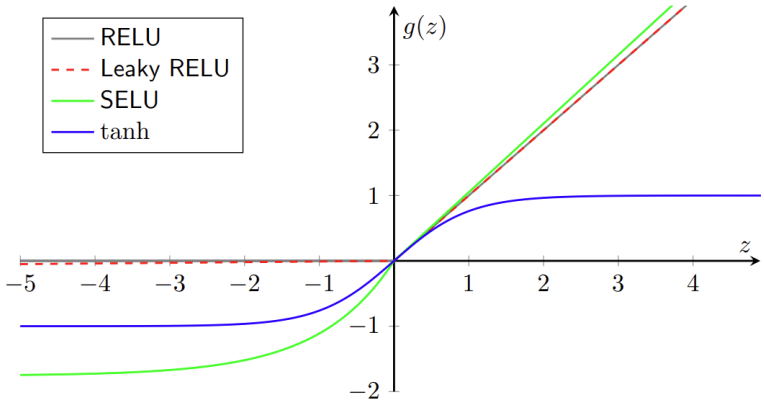


Figure 1.14 Most commonly used activation functions in neural networks [33].

The network ends with an output layer, marked with green color in Fig 1.12, which returns the result. This layer's design and activation function are specific to the network's task. For instance, a softmax activation function might be employed for a multi-class classification problem to produce probabilities that add up to one. In contrast, binary classification might use a single neuron with sigmoid activation.

The **learning process** of a feedforward neural network involves adjusting the weights within the network in such a way that it makes better predictions. This adjustment is typically achieved through backpropagation combined with an optimization algorithm, such as gradient descent. During training, the neural network makes predictions on the input data. It then calculates the error of these predictions against the actual targets. Backpropagation computes the gradient of the loss function concerning each weight using the chain rule [1], effectively propagating the error gradient back through the network. Given a loss function L, the gradient respect to weight w in the network is calculated as follows:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w}. \tag{1.8}$$

Where $\frac{\partial L}{\partial y}$ is the derivative of the loss function concerning the output of the neural network, and $\frac{\partial y}{\partial w}$ is the derivative of the output concerning the weight.

Gradient descent then uses the calculated gradient to update the weights of the neural network:

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w} \tag{1.9}$$

Where $\eta \in (0, 1]$ is the learning rate, a small positive scalar determining the size of the algorithm's step in the negative gradient direction [34].

This iterative process of calculating the error gradient and updating the weights continues until the network's performance meets predetermined stopping criteria, such as a specified number of iterations called **epochs** or a minimum change in the loss between iterations.

[1] The chain rule, describes how we can differentiate composite functions as $h' = (f \circ g)' = (f' \circ g) \cdot g'$, enabling the calculation of derivatives across multiple layers in neural networks.

1.5.2 Convolutional Neural Network

Convolutional Neural Networks (CNN) are a special type of deep neural networks that make use of the convolution operation and are highly effective in analyzing visual imagery [35, p. 811]. A CNN architecture is primarily composed of three-layer types: **convolutional layers**, **pooling layers**, and **fully connected layers**. A typical architecture of a CNN can be seen in Fig. 1.15. The convolutional layer is a CNN core building block, applying kernels to the input to create a feature map that summarizes the presence of detected information in the input [36]. Pooling layers follow the convolutional layers and perform down-sampling operations to reduce the dimensionality of the features, thus decreasing the computational complexity and chances of over-fitting [2]. The effect of convolutional and pooling layers can be seen in Fig. 1.16. Lastly, fully connected layers, which come after several convolutional and pooling layers, perform classification based on the features extracted and down-sampled by the previous layers [37].

The **learning process** in CNNs involves adjusting the weights of the kernels within the network, similarly to FNNs [32]. CNNs show superior results to regular FNNs on visual data, as they have far fewer learnable parameters to adjust, making learning more straightforward on complex data.

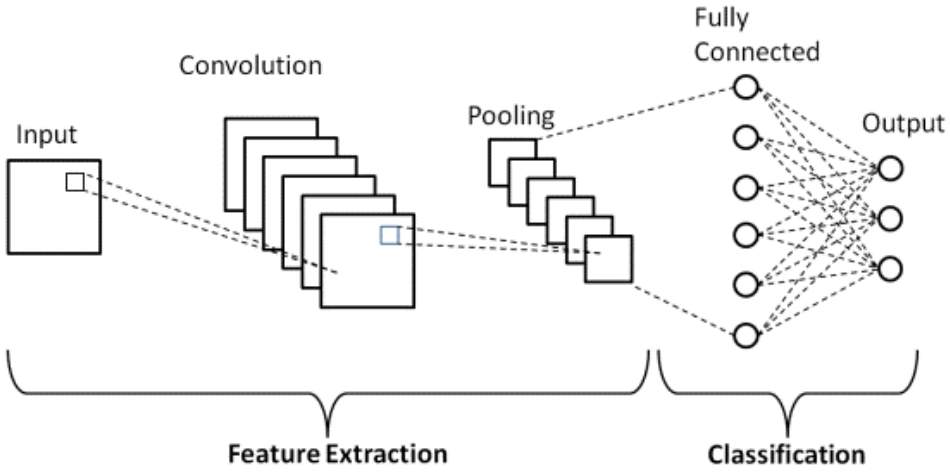


Figure 1.15 Schematic of a convolutional neural network. An input flows through feature extraction with a single convolutional and pooling layer and is interpreted by a fully connected layer [38].

[2] Over-fitting means the model is tuned too much to the training data and is not generalized, resulting in poor performance on new data

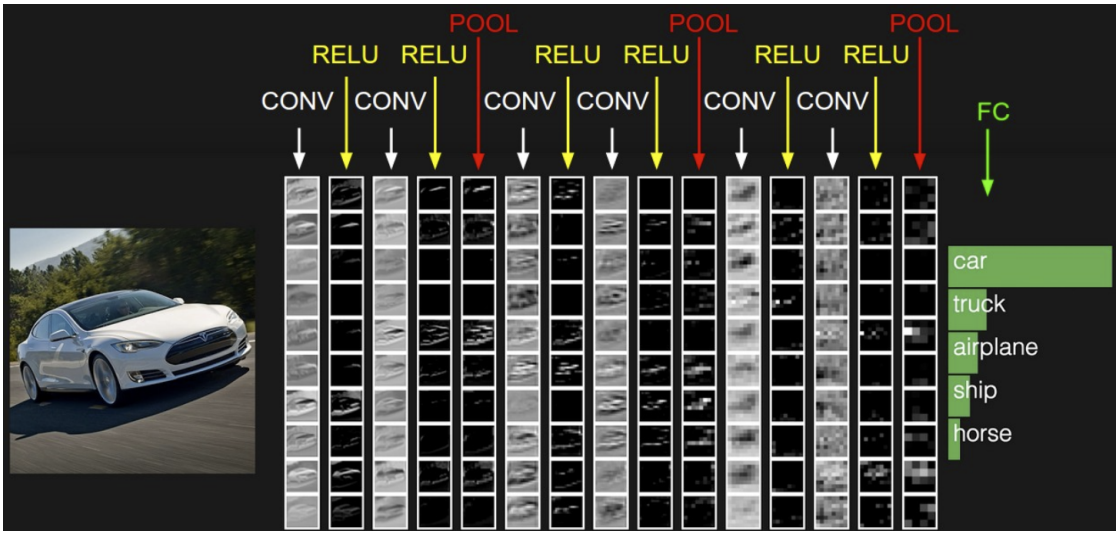


Figure 1.16 Visualization depicting the convolution operation layer by layer. The original image of a car is visible on the left, undergoing successive passes through different layers of the neural network. Convolutional layers detect patterns such as edges and shapes. ReLU layers introduce nonlinearity through the application of a nonlinear function. Pooling layers decrease spatial dimensions while preserving essential features. The gradual reduction in dimensions can be observed in the car, with the final pooling layer containing only a few pixels. Finally, the fully connected layer processes these features, recognizing the image as a car [38].

1.6 Machine Learning on Point Clouds

The unique properties of point clouds discussed in section 1.2, pose difficult challenge for deep learning. Due to their unordered nature, convolution neural networks, which excel with grid-like data, cannot be directly applied. Therefore, most researchers tended to transform point clouds into regular formats, like 3D voxels [11], where a 3D convolutional neural network could be applied or created a set of images (e.g., views), which would be processed by a 2D neural network [10]. However, such representations would significantly increase the memory needed for processing and time spent on data conversion. Advancements in neural network architectures, beginning with pointnet [12] have proved, that using symmetric functions to transform the points into descriptors allows us to apply the neural network on raw point data. We discuss two modern architectures below.

1.6.1 PointNet

PointNet was a revolutionary method introduced in 2016 to classify and segment raw point cloud data [12]. Fig. 1.17 shows the classification and segmentation network diagram. The neural network first learns an affine transformation using a small T-Net neural network to transform the input. The network then tries to understand the local structures using shared MLP. After learning the local features, a max-pooling layer aggregates the points into a global vector, representing the entire point cloud. As the max-pooling function is symmetric, it ensures permutation invariance, meaning the order of input points is irrelevant. The classification network has an output for k classes. The segmentation network is an extension of the classification network. By concatenating global and local features, per-point scores are calculated as output.

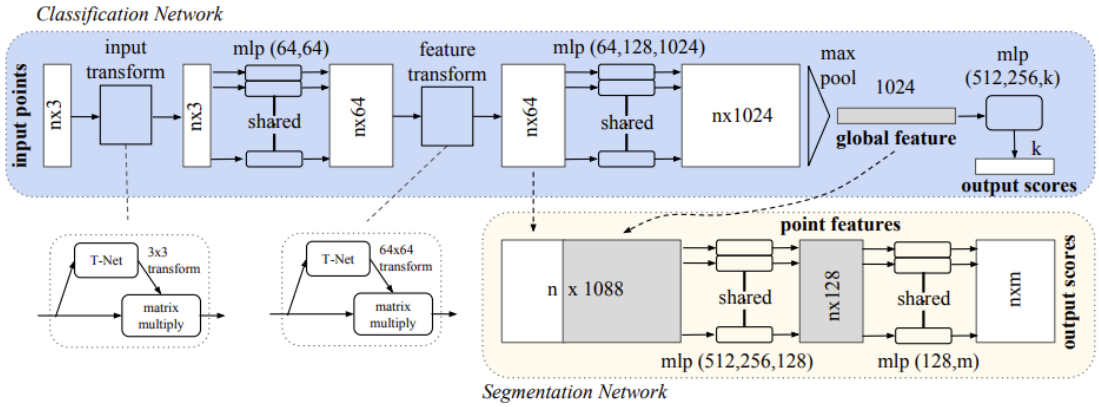


Figure 1.17 PointNet architecture [12]. The numbers in brackets are layer sizes; Batchnorm is used for all layers with ReLU. Dropout layers are used for the last MLP in the classification net

1.6.2 Convpoint

ConvPoint is a more modern approach for larger point clouds [14]. The method uses continuous convolutions for point cloud processing. It applies dynamic kernel convolutions, which capture local geometry. The process begins with extracting local features around each point, utilizing learnable kernels that adapt based on the point's neighborhood. This approach ensures that the model is sensitive to the local context of each point. Following this, ConvPoint employs a hierarchical learning strategy, progressively aggregating these local features to capture higher-level structures within the data.

ConvPoint architecture also ensures the essential properties described in section 1.2, like permutation and transformation invariance.

For classification tasks, ConvPoint produces a global descriptor that summarizes the entire point cloud, enabling the identification of the object or scene category. ConvPoint extends its framework in an encoder-decoder-like structure to combine global and local features in segmentation tasks, providing per-point predictions.

1.7 Point Cloud Registration

The process of finding a spatial transformation that aligns two point clouds is called registration. The fundamental aim of finding such a transformation is to align multiple data sets with overlapping features within a unified coordinate system [39]. Such alignment can be seen in Fig. 2.14. The process is often divided into two separate algorithms. The initial step, known as global registration, focuses on achieving a rough alignment that brings the two datasets in close proximity. Subsequently, a fine registration, often done by algorithms like ICP, is used to refine the fit and achieve a closer alignment.

1.7.1 Registration in CloudCompare

CloudCompare [40] is an open-source 3D point cloud processing software. The program provides most point cloud algorithms with a simple user interface and can serve as a demonstration of how registration is implemented in most 3D software. Registration in CloudCompare is done in two steps. The global registration is calculated using user-specified points. Then, an automatic fine registration algorithm is applied. The workflow is as follows:

First, the user is tasked with finding the corresponding points on both point clouds as in Fig. 1.18.

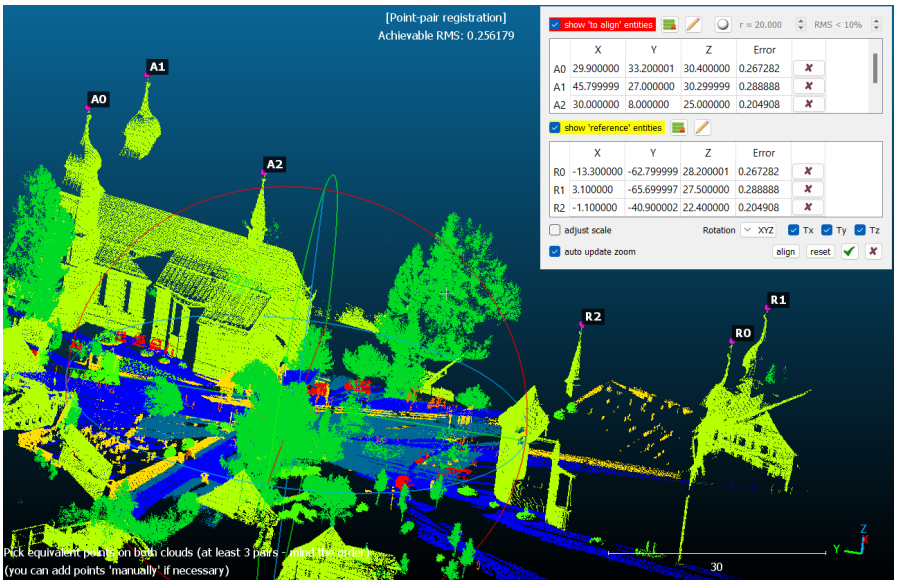


Figure 1.18 CloudCompare interface, illustrating the process of aligning point clouds by selecting corresponding points atop a church tower. Points prefixed with 'A' denote the target point cloud, while those beginning with 'R' represent the source point cloud.

The program calculates the rigid transformation with minimum error on the user-provided points. We then have to apply the transformation manually. The result of the transformation can be seen in Fig. 1.19.

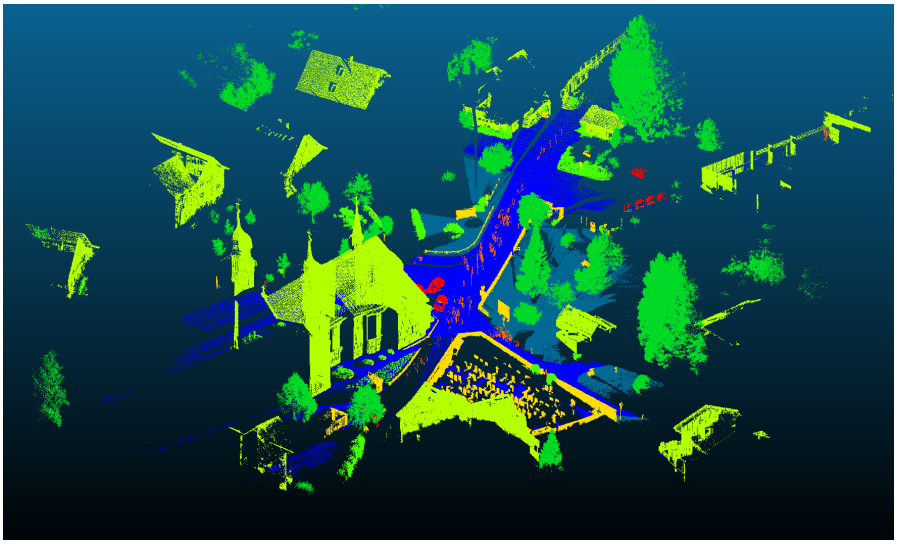


Figure 1.19 Visualization of the result after the rough transformation calculated from the point selection is applied.

For fine registration, we are tasked with filling in a dialog window, see Fig. 1.20, with the theoretical overlap and squared error between the points. After pressing OK, we get the fine

registration result.

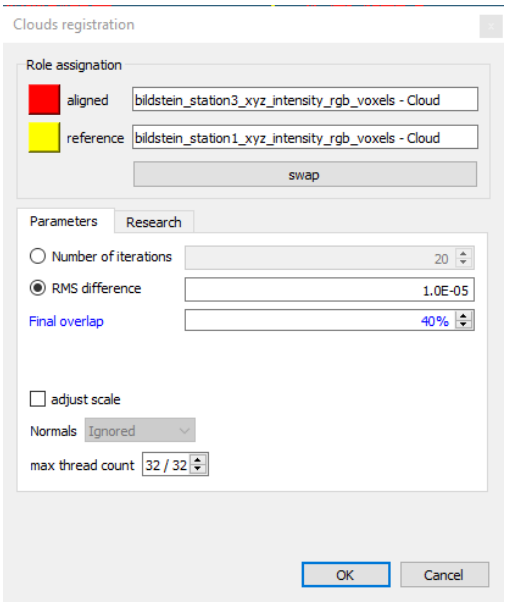


Figure 1.20 A CloudCompare dialog window showing the fine registration settings.

This provides a simple yet effective demonstration of how registration can be achieved with more complex point sets. However, as can be seen, much user input and expertise is required. We must choose the theoretical overlap of the two point clouds and the root mean squared error. For more complex point clouds, this could be very difficult, if impossible, to estimate accurately, leading to incorrect results, see Fig. 1.22. Such an incorrect result can be seen in Fig 1.21, where we have incorrectly estimated the overlap of the point clouds as 60%.

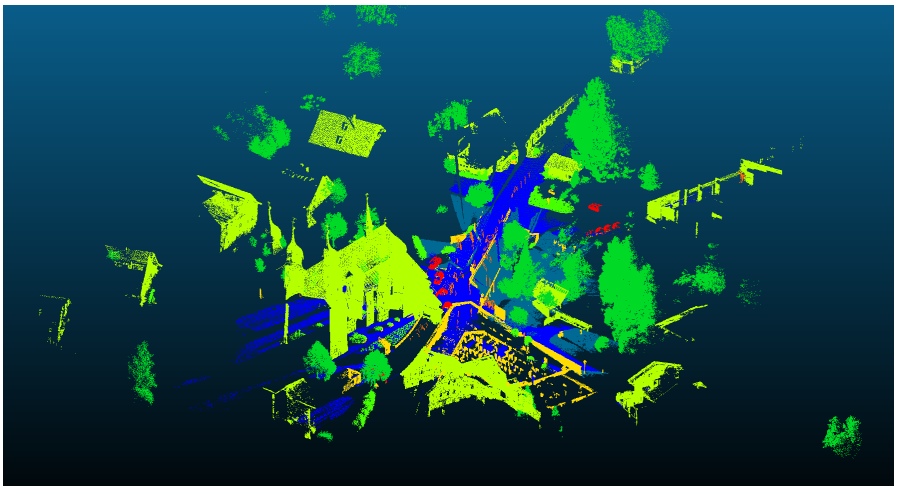


Figure 1.21 Visualization of the sub-optimal result after fine registration, when the overlap was set to 60%.

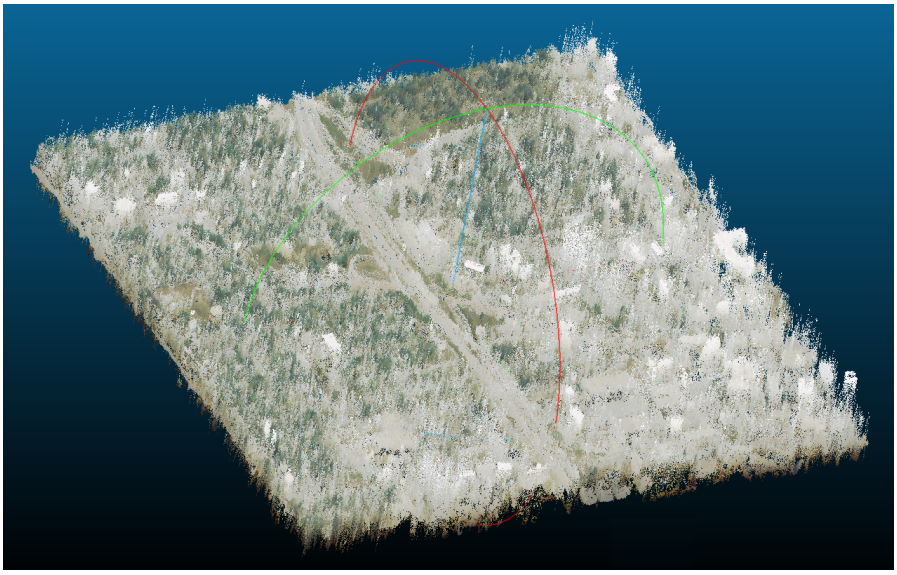


Figure 1.22 Point cloud scan of Des Moines in Iowa with low readability where it is difficult to identify the key points.

1.7.2 Global Registration

An effective and often cited automated method for global registration is using fast point feature histograms (FPFH) [41] for a key-point matching algorithm. FPFH represents each point in a point cloud by its local geometric properties, such as surface normals or curvatures. It computes a feature histogram based on the relationships between neighboring points called the influence region. The influence region of a point can be seen in Fig. 1.23.

This representation gives each point descriptive features, allowing us to find a similar point from the opposing point cloud. After finding the FPFH, RANSAC finds matching correspondences between points, fits a transformation model to each subset, and evaluates its consistency with the best correspondences. As the algorithm uses random searches, the point cloud must be highly sub-sampled to run in an acceptable time.

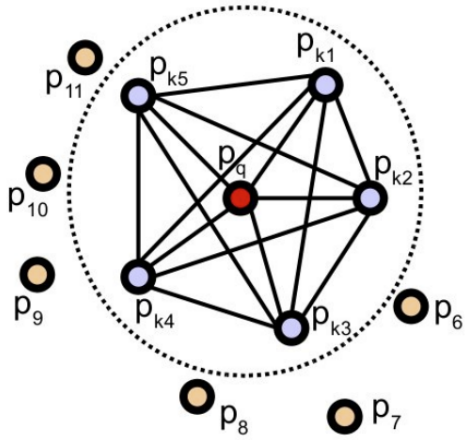


Figure 1.23 The influence region diagram for a point feature histogram. The query point (red) and its k-neighbors (blue) are fully interconnected in a mesh [41].

1.7.3 Iterative Closest Point

Iterative Closest Point (ICP) and its variations are classical approaches to local registration on point clouds. The basic principle is to find a spatial transformation from a source point cloud to a target point cloud; the idea of the transformation is illustrated in Fig. 1.24. The algorithm primarily comprises the following steps: point matching, rotation and translation estimation, point transformation, and an iterative process. Point-to-point ICP [42] matches each point in the source point cloud to its nearest point in the target cloud, minimizing the distance function. Point-to-plane ICP [43] extends this by considering distances from each source point to the tangent plane of its corresponding point in the target cloud, which is especially useful for irregular surfaces. Generalized ICP [44] further extends the algorithm by combining both Point-to-Point ICP and Point-to-Plane ICP into a single probabilistic framework, improving the robustness and accuracy.

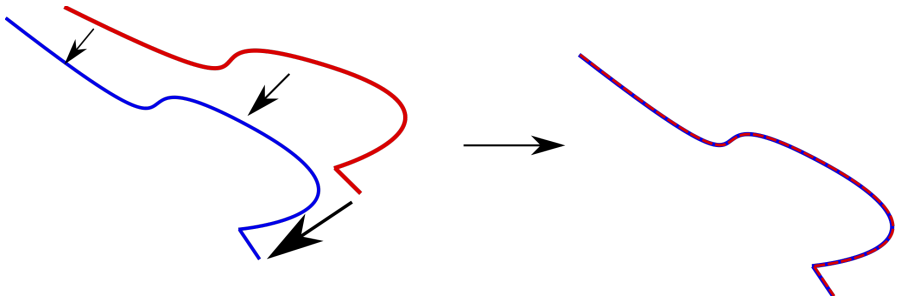


Figure 1.24 Idea behind the iterative closest point algorithm [45]. The source object is colored red, and the target is blue.

The steps of the ICP algorithm are described in more detail below:

1. **Point Matching:** For each point within the source point cloud, the algorithm identifies the nearest point with a maximum radius in the reference point cloud.
2. **Rotation and Translation Estimation:** Using a distance metric minimization technique, the algorithm estimates the optimal rotation and translation. This estimation aims to align each source point with its corresponding match obtained in the previous step.
3. **Point Transformation:** Utilizing the derived transformation from the previous step, the algorithm transforms the source points accordingly.
4. **Iterative Process:** The algorithm iterates through steps 1 – 3, refining the alignment until specific criteria are satisfied, such as reaching a predefined maximum number of iterations or achieving an error less than the previously specified threshold.

In addition, accompanied by the description, a pseudocode representation of the algorithm is provided below.

Algorithm 1 Iterative Closest Point

procedure ICP(S, D, θ)
  Input: source point cloud **A**, destination point cloud **B**, convergence threshold θ
  Output: transformed source point cloud S'
  Initialize transformation **T** with identity matrix
  $error \leftarrow \infty$
  while $error > \theta$ **do**
    Find closest point pairs between **A** and **B**
    Estimate the transformation $\mathbf{T}_{\text{new}}$ to align **A** to **B**
    Apply the transformation: $\mathbf{A}' = \mathbf{T}_{\text{new}} \cdot \mathbf{A}$
    Calculate the error between $\mathbf{A}'$ and **B**
    Update transformation **T**: $\mathbf{T} \leftarrow \mathbf{T}_{\text{new}}$
  end while
  return $\mathbf{A}'$
end procedure

1.8 Evaluation of Registration

The primary objective of the registration process is to ensure a high degree of point overlap between the source point cloud and the target point cloud (fitness, correspondence set) and to minimize the distance between those overlapping points (RMSE).

However, these metrics can be misleading. High overlap and low distance between points do not guarantee successful registration, especially when the characteristics of the registration outcome are unknown. Therefore, visually checking the registration result is also needed.

The metrics are described in more detail below:

- **Visual Check** – A fundamental yet indispensable step is manually visualizing and accessing the registration result. This is because the minimization algorithm attempts to reduce the distance between points. However, these metrics might not yield the best registration, especially when dealing with partially overlapping datasets.

- **Correspondence Set** - refers to a collection of pairs of points, each from a different point cloud. Two points are considered corresponding if the following conditions are met: the distance between them is the smallest among all points within a specified proximity, and this distance is smaller than a predefined threshold. It is believed that the pair represents the same physical point in space. Determining correspondence sets is a critical step in the registration process, and the higher the number of correspondence sets, the more we can trust the algorithm result. Fitness and inlier RMSE metrics are calculated from the correspondence set.

- **Fitness** – Fitness is closely linked to the correspondence set. It measures the proportion of points in one point cloud that have corresponding points in the other point cloud within the specified distance threshold after registration. Fitness is expressed as a value between 0 and 1, where a value closer to 1 indicates a higher percentage of the source points that fit well with the target, suggesting a successful registration. Given a source point cloud A, target point cloud B and a registration result with n number of correspondences, the fitness f is calculated as:

$$f = \frac{n}{|B|}. \tag{1.10}$$

- **Inlier RMSE** – is a metric used to quantify the accuracy of point cloud registration. It is computed on points from the correspondence set and measures the average Euclidean distance

between corresponding points using the equation 1.11. This metric provides insight into how closely the point clouds are aligned. Lower RMSE indicates a higher accuracy.

Given a registration of two point clouds with n corresponding points, for each pair of corresponding points, where $p_i = (x_i, y_i, z_i)$ is from the source point cloud and $\hat{p}_i = (\hat{x}_i, \hat{y}_i, \hat{z}_i)$ is the corresponding nearest point in the target point cloud, the inlier RMSE is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2 + (\hat{z}_i - z_i)^2}\right)^2} \tag{1.11}$$

1.9 Existing Solutions

As described in the previous section, most software working with point clouds implement registration algorithms. These algorithms rely on user input to estimate the initial transformation between the point clouds and then apply algorithmic fine registration to smooth out the result.

Automatic registration processes that detect key points through local feature analysis, like FPFH, are often used in robotics and other sectors where autonomous alignment is needed [46]. Nevertheless, these techniques are not without problems. A significant limitation is their vulnerability to inconsistencies in point cloud densities. Additionally, those algorithms assume that the geometrical structure around a point carries enough information for unique identification. This assumption can be problematic in areas characterized by uniform patterns.

Automatic registration of point clouds using deep learning methods to identify correspondences has been an area of focus for several years [47], and research has focused primarily on the use of neural networks to detect the corresponding parts of point clouds to find an optimal transformation [48, 49]. However, these methods show a significant disparity between synthetic and real-world data [50].

1.10 Our Approach

This work aims to develop a framework that improves the registration of partially overlapping point clouds and investigates the integration of semantic labels in the alignment process. We establish two distinct pipelines: one incorporating semantic labels for dataset partitioning and one using a traditional approach. We do not use neural networks to predict transformations; instead, we use conventional algorithmic approaches and only use deep learning for segmentation, where the networks tend to perform very robustly. We assess the performance of our pipelines on large-scale datasets against traditional global registration using FPFH features and different ICP algorithm variants. We also propose changes to the ICP algorithm by using different distance metrics and the semantic label to perform per-part ICP.

Chapter 2
Implementation

This chapter introduces the programming tools we use for the implementation and explores the datasets we use for testing. Next, we propose the outline of the pipeline, describe the preprocessing methods, and go through the pipeline implementation process.

2.1 Programming Tools and Packages

Machine learning and algorithms working with point clouds are computationally demanding and require large code optimization. Fortunately, open-source libraries and software make this easier by providing us with efficient implementations that we can use as a foundation for this work. Below are the most prominent libraries we use in our implementation:

- Cython is an open-source language that enhances Python by allowing static typing and compiling into C [51]. It allows the call of functions implemented in C/C++ directly from Python. This helps us optimize Python code for performance-critical applications by combining the simplicity of Python with the speed of C/C++.
- Pytorch is an open source, high-performance deep learning library developed by Meta and is intended to be used with Python and C++ [52]. It is one of the most popular frameworks for neural network design due to its simple interface and efficient C++ core. We use this framework for the segmentation network implementation.
- Open3d is a open source library developed to work with 3D data [53]. It provides useful functionality ranging from efficient data structures and algorithms to visualization functions.
- Scikit-Learn is an open-source library containing implementations of a wide array of machine learning algorithms [54]. It offers a high-level interface for preprocessing, model selection, and evaluation tasks.

2.2 Datasets

This section discusses the datasets selected to evaluate our registration method against state-of-art solutions. We specifically chose segmentation benchmarks as they are pre-annotated, simplifying the process of training the neural network for semantic label acquisition.

2.2.1 Stanford 3D Indoor Scene Dataset

The Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS) [55] offers a comprehensive collection of the Stanford campus's indoor point clouds comprising six areas encompassing over 70,000 square meters. The areas are divided by room, where each semantic object is stored in a separate file, together forming the room. The point cloud of each room consists of millions of points and features coordinates (x, y, z), color values (r, g, b), and semantic labels. A schematic of the dataset can be seen in Fig. 2.1.

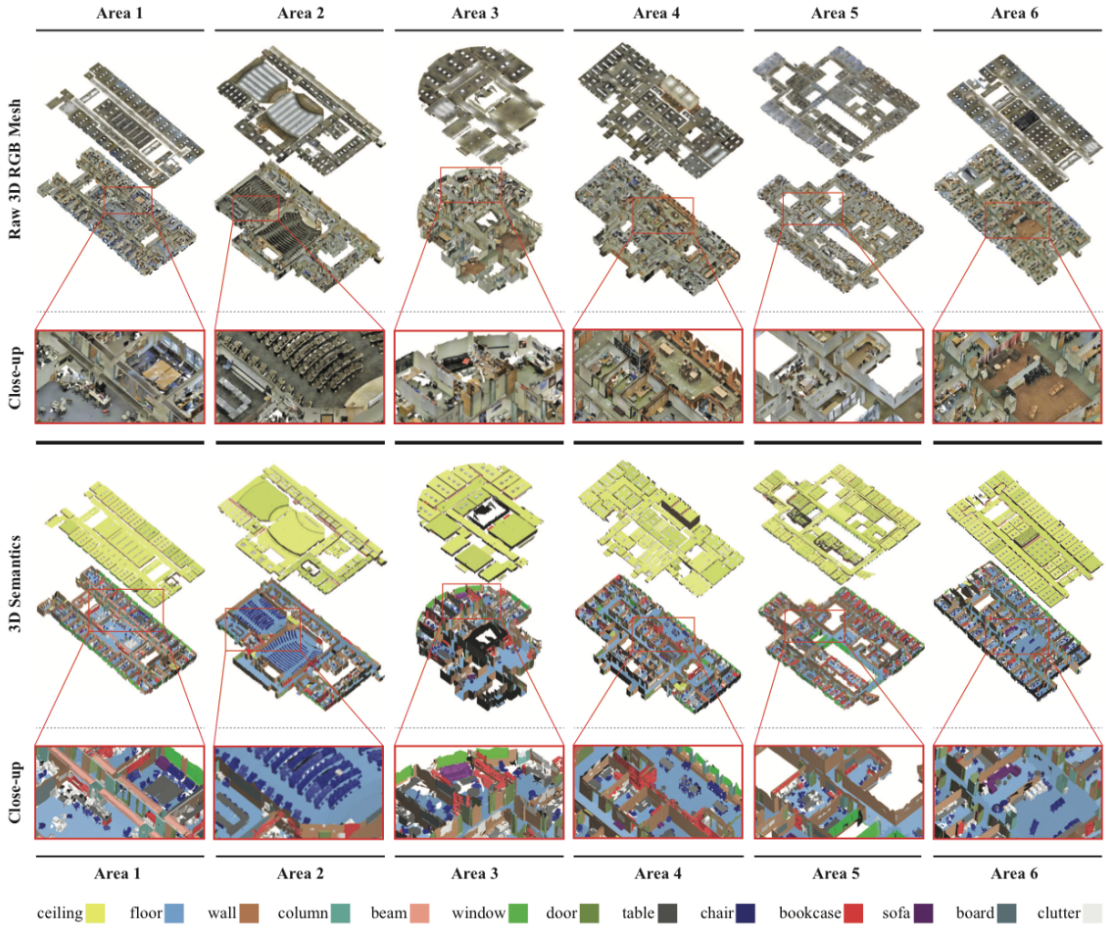


Figure 2.1 Schematic of the S3DIS dataset [55]. It is composed of six areas. At the top, raw scans with color values are displayed. At the bottom, areas are annotated with semantic labels. The color of each respective semantic label is displayed at the bottom.

We use areas 1 – 5 for training the neural network and area 6 for registration. S3DIS alone does not have point clouds with partial overlap, which would be suitable for registration. Therefore, we have to prepare the dataset manually.

Initially, we split each point cloud into two parts based on the mean value of the x-coordinates, with a 30% overlap. This ensures that the two dataset parts have a shared region in the middle. Then, we iteratively swap parts of the halved point clouds by randomly choosing a fixed-size block, moving it to the second half, and leaving random sub-sampled points. This ensures that both point clouds have shared parts different from those in the center. Lastly, rotation and translation are applied to the source point cloud. This simulates real-world scenarios where data might be captured from different viewpoints or at different times. An example of a proceed point cloud pair using the above-described method can be seen in Fig. 2.3.

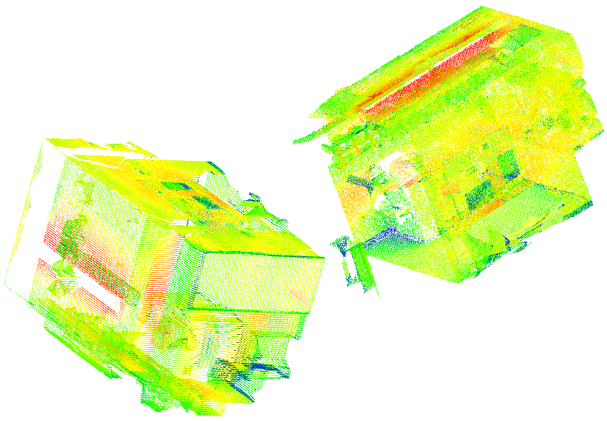


Figure 2.2 Result on the copy room 1 point cloud after the data preparation. The original point cloud was divided into two, translated and rotated.

2.2.2 Nuage de Points et Modélisation 3D

The Nuage de Points et Modélisation 3D (NPM3D) dataset [56] is an outdoor semantic segmentation benchmark. The data was produced by a mobile laser system consisting of six dense, annotated point clouds from two urban locations, Paris and Lille. Three point clouds are used for training the neural network, and three for testing. Individual points have spatial coordinates (x, y, z), reflectance values, and a semantic label.

Figure 2.3 Example from the Lille NPM3D dataset [56]. Colors are semantic labels, where roads are gray, vehicles are red, buildings are beige, and vegetation is green.

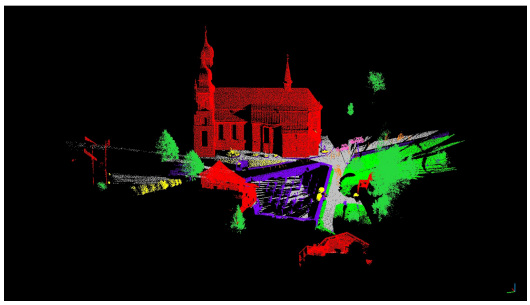
Like the S3DIS dataset, the NPM3D dataset also lacks overlapping point clouds that could be directly registered. As the point clouds were very complex, we manually parted them using CloudCompare.

2.2.3 Semantic3D

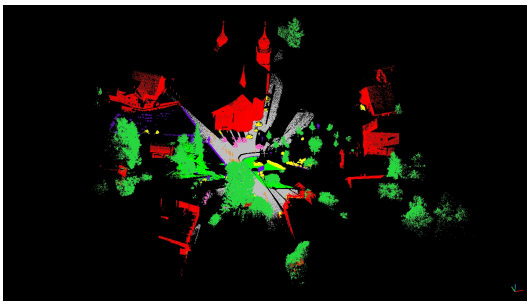
Semantic3D dataset [57] provides us with large-scale, human-annotated, outdoor point clouds. The dataset consists of thirty point clouds, split into training and testing groups, each with fifteen point clouds. Each point cloud has coordinate (x, y, z), intensity, and color (r, g, b) information. Point clouds from the training set also have label information saved in a separate file.

Unlike previous datasets, Semantic3D has multiple scans from the same location, meaning they can be registered as is. Upon inspecting the data, we selected five point cloud pairs with sufficient overlap suitable for registration. These are: Bildstein 1 – Bildstein 3, Untermaederbrunnen 1 – Untermaederbrunnen 3, Domfountain 1 – Domfountain 3, Marketsquarefeldkirch 1 – Marketsquarefeldkirch 4, Stgallencathedral 1 – Stgallencathedral 1. Other point clouds did not have a corresponding pair, and we did not use them further.

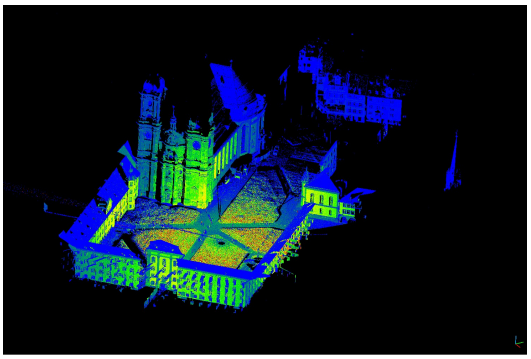
Two pairs from the datasets are illustrated in Fig. 2.4. However, it is essential to note that several datasets, specifically the Bildstein, Untermaederbrunner, and Domfound, are part of the training dataset. We incorporate them into the testing data and disregard the labels associated with these datasets—consequently, the neural network training proceeds with the rest of the training data.



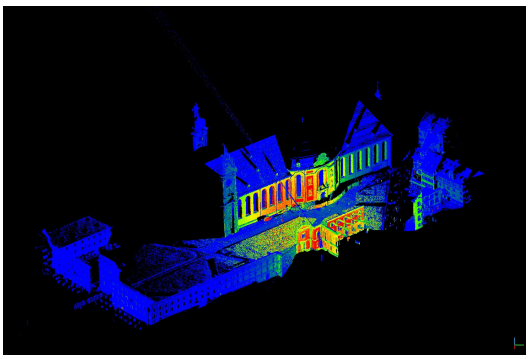
(a) Bildstein 1



(b) Bildstein 3



(c) Stgallencathedral 1



(d) Stgallencathedral 3

Figure 2.4 Example of point cloud pairs for registration Bildstein 1 – Bildstein 3 and Stgallencathedral 1 – Stgallencathedral 3 [58].

2.3 Proposed Method

We propose a registration method utilizing divide-and-conquer paradigm, where we will split the point clouds into logical segments using convolutional neural network, perform registration on the corresponding logical pairs and then choose the best overall transformation on the whole data. We use the label information to divide the dataset and perform principal component alignment with ICP smoothing to align local structures. We start with raw point clouds, apply preprocessing methods, and pass the point clouds along to the segmentation network, which adds label information to each point. We then split the data by the label and pass it to the modified registration algorithm to find the transformation matrix. We named this method Semantic Point Cloud Processing using Principal Component Analysis (SPCR-PCA), and its schema can be seen in Fig. 2.5. In order to properly test whether adding the semantic label into the registration process improves the accuracy, we also implemented a method without the use of semantic labels, called Point Cloud Processing using Principal Component Analysis (PCR-PCA), which works in a similar matter, omitting the first preprocessing step, segmentation network, and batching.

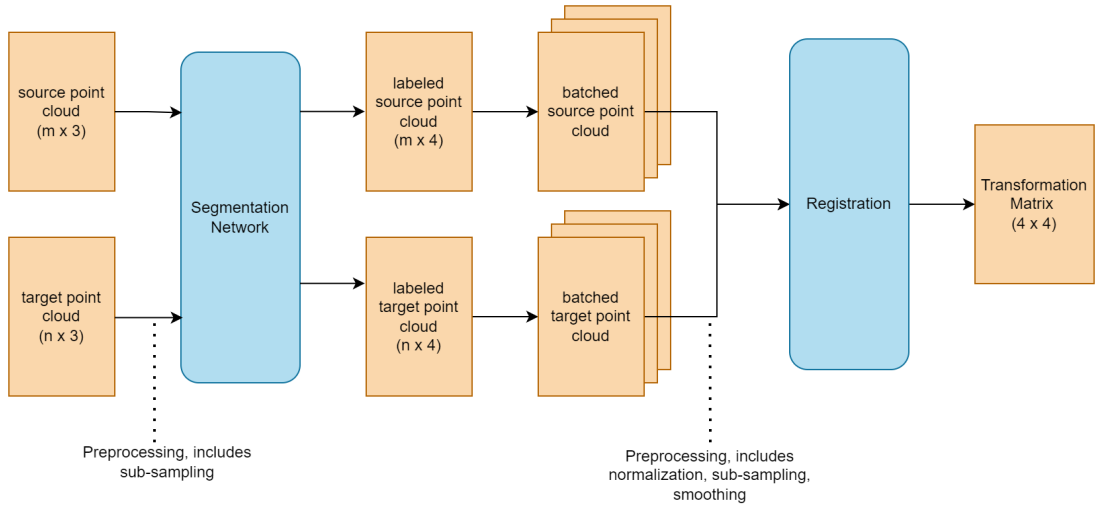


Figure 2.5 Proposed pipeline for labeled point clouds. The segmentation network separately takes two point clouds and the predicted labels. The registration algorithm is discussed in section 2.6.

2.4 Segmentation Network

In this section, the implementation of the segmentation network is discussed. For this purpose, we use the ConvPoint architecture and its corresponding implementation [14, 59].

2.4.1 Model Training and Evaluation

The training process was carried out in the same way, as described in section 1.6.2. A random point is selected at training time, and all points are extracted in an infinite vertical column centered around the point. The column is eight meters wide. 8192 points are randomly selected for each column and fed into the neural network. We use the Adam optimizer to determine the step size and cross-entropy as the loss function. Training is carried out throughout 100 epochs.

The ConvPoint method uses three metrics to evaluate the precision of the neural network. They are listed below:

- Overall Accuracy: Describes the overall accuracy of the model predictions. It can be expressed as:
$$\text{OA} = \frac{\text{Sum of True Positives across all classes}}{\text{Total Number of Observations}}$$
- Average Accuracy Per Class: Describes the mean of per class accuracy, where the class accuracy is computed as:
$$\text{Accuracy}(i) = \frac{\text{True Positives of Class } i}{\text{Total Samples of Class } i}$$
- Intersection Over Union (Jaccard index): measures the similarity between sample sets, defined as the intersection's size divided by the union's size. The equation is provided below:
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$
where $|S|$ denotes the cardinality of a set S.

We use all those metrics to evaluate the neural network. The change in the values of the metrics on the Semantic3D dataset can be seen in Fig. 2.6. The final overall accuracy, per class accuracy, and IoU of the model on all the datasets can be seen in Table 2.1.

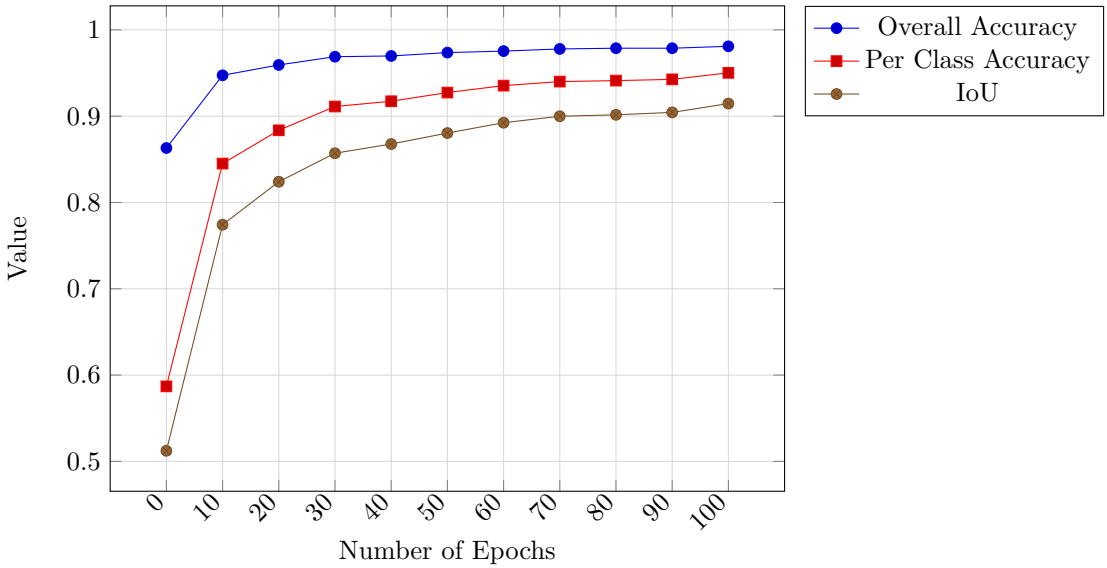


Figure 2.6 Change in metrics on the Semantic3D training dataset over 100 epochs.

Table 2.1 Model accuracy on the training set. The number of epochs for each of the datasets is written in brackets.

Metric	Overall Accuracy	Per Class Accuracy	IoU
S3DIS (250)	0.95726	0.93094	0.88003
NPM3D (100)	0.99330	0.94647	0.91402
Semantic3D (100)	0.98096	0.95024	0.91459

2.4.2 Model Predictions

Making predictions using the trained model is similar to the training process. A 2D occupancy pixel map is computed with a pixel size of 0.5 meters by projecting vertically on the horizontal plane. For each column, 8192 points are randomly selected and fed into the network. The output scores are summed at a point level. The points that were not selected and labeled receive the label of their nearest neighbor. Prediction on the Bildstein 1 dataset can be seen in Fig. 2.7 and the same image split into individual labels in Fig. 2.8.

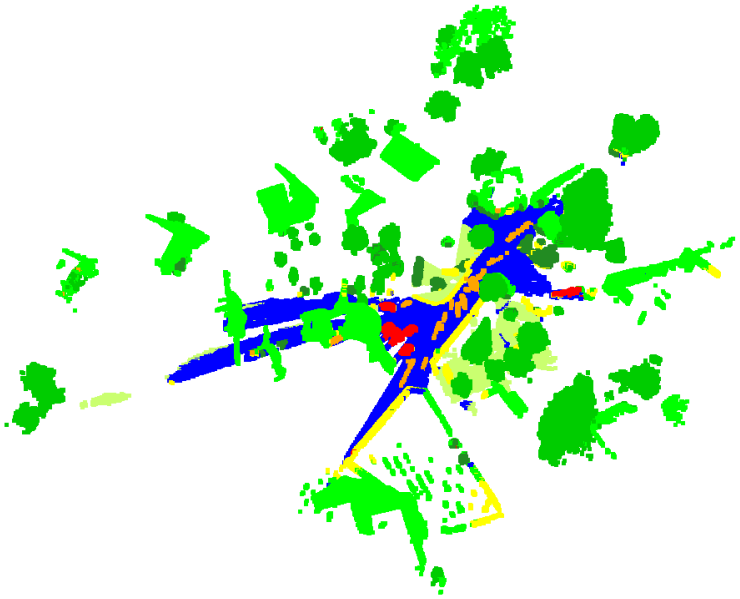
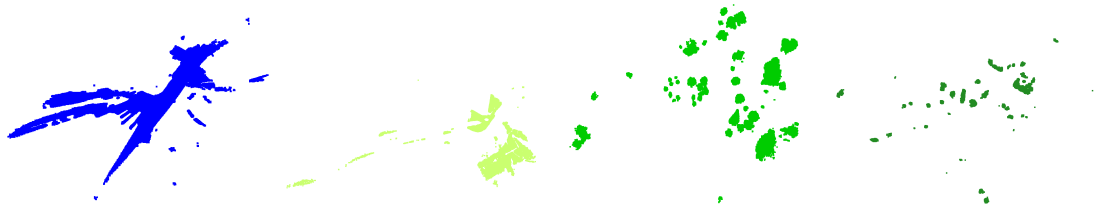


Figure 2.7 Prediction of the neural network on Bildstein 1 dataset. Different Colors are different predicted labels.



(a) man-made terrain **(b)** natural terrain **(c)** high vegetation **(d)** low vegetation

(e) buildings **(f)** hard scape **(g)** scanning artefacts **(h)** cars

Figure 2.8 Individual predicted labels on the Bildstein 1 dataset.

2.5 Data Preprocessing

In this section, we discuss the preprocessing methods from the above-described pipeline. The preprocessing steps are applied on each batch from the neural network separately.

2.5.1 Data Reduction

The point clouds from the datasets are too dense to be used in an iterative algorithm, as it would take too long to run. To eliminate this, we subsample each data set, reducing the number of points but keeping the essential structure. We chose to use a voxel grid filter, as it is known for efficient downsampling while preserving the core structure of the data [60].

The process of voxel down-sampling can be broken down into two steps. The points are divided into a regular voxel grid in the first step, creating a structured data representation. In the second step, each voxel generates a single point by averaging all the points it contains. This averaging process also extends to the point properties, such as color or label. For a visual representation of this process, refer to Fig. 2.9.

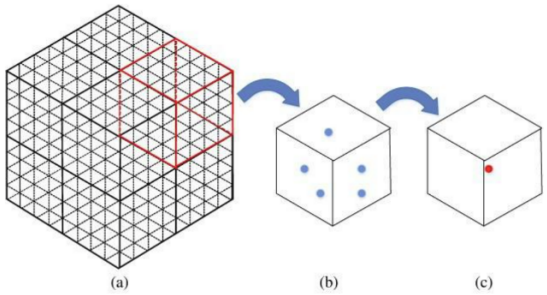


Figure 2.9 Illustration of voxel-grid filtering: (a) voxel grid; (b) unfiltered unit voxel; and (c) the center of mass of the unit voxel [61].

While subsampling does not alter the algorithm's output, as the transformation matrix can be applied to the original dense point cloud, it is essential to note that it does introduce a trade-off between speed and precision resulting from information loss.

Data reduction proved necessary for efficient implementation. We have experimented with many sizes and have experimentally deduced that the best trade-off between speed and accuracy was to choose a voxel size for clustering of 0.005. This value splits the unit ball into a space of 400x400 voxels. This leaves enough information inside the dataset but dramatically increases the algorithm's speed.

2.5.2 Normalization

Normalization is an important preprocessing step in many methods. It encloses all points in a common boundary, typically within the (-1, 1) range, without sacrificing any information. The process can be visualized as scaling the point cloud within a unit sphere [1]. Given a point cloud P, centroid of the point cloud $\hat{p}$, the normalized point cloud C can be found using:

$$C = \left\{ \frac{p - \hat{p}}{\max\limits_{p' \in P} (||p' - \hat{p}||)} \mid \forall p \in P \right\} \tag{2.1}$$

[1] A unit sphere is a sphere with a diameter equal to one

The data values fluctuate highly between different point cloud pairs. We use normalization to remove this inconsistency. To uniformly scale both clouds by the same proportion, we have modified the normalization equation to take the maximum of both point clouds. Given point clouds A and B, their respective centroids $\hat{a}$, $\hat{b}$, we define a distance function M that returns the maximum distance, calculated from each point to its respective centroid. This equation can be seen below:

$$M(A, B) = \max\left(\max_{a \in A}(\|a - \hat{a}\|), \max_{b \in B}(\|b - \hat{b}\|)\right) \tag{2.2}$$

The normalized point cloud A' is calculated using the equation 2.3 and point cloud B' using the equation 2.4.

$$A' = \left\{ \frac{a - \hat{a}}{M(A, B)} \mid \forall a \in A \right\} \tag{2.3}$$

$$B' = \left\{ \frac{b - \hat{b}}{M(A, B)} \mid \forall b \in B \right\} \tag{2.4}$$

This method ensures that both A' and B' are scaled by the same proportion, ensuring the relative dimensions between the point clouds remain unchanged.

Given the significant variations in the data, normalization proved crucial. Choosing and calculating parameters, such as the epsilon value for noise reduction and threshold for the ICP algorithm, was much more straightforward.

2.5.3 Noise Reduction

Noise reduction is a common preprocessing step. It refines the point cloud by removing outliers that do not contribute to the primary structure of the data, cleaning the point cloud, and ensuring it consists only of relevant points.

We use the DBSCAN algorithm for effective noise reduction. DBSCAN distinguishes between core points (central points), border points (at the edges of a cluster), and noise (isolated points not part of a cluster) as in Fig. 1.6. The algorithm is explained in detail in section 1.3.3.

Unlike other clustering algorithms that require parameter tuning, DBSCAN requires minimal hyperparameter tuning, as it can run entirely without hyperparameters. We utilize the scikit-learn implementation of DBSCAN [62], which offers a straightforward and efficient approach to applying the algorithm to our data.

When DBSCAN was applied without parameters, it did not filter any data, resulting in the entire point cloud being in a single cluster. Therefore, we set the epsilon value, which determines the distance within which each point considers its neighborhood, to $2 \cdot$voxel size. This decision was made after experimental adjustments to optimize the balance between sensitivity and specificity in the clustering process. By setting epsilon to twice the voxel size, we ensure that the neighborhood of each point encompasses a sufficiently large area to include directly adjacent points while filtering out outliers.

2.6 Registration

The initial alignment of the point clouds is the biggest precondition for a successful fine registration using ICP. If the point clouds are roughly aligned, the ICP algorithm likely succeeds in finding the close-to-optimal solution. If the initial position of the point clouds is off, the algorithm likely converges toward local minima far away from a successful registration.

Similarly to most software solutions, we have divided the registration process into two steps: a global registration, which aims to roughly align the two overlapping point clouds, and a fine

registration using ICP, which smooths the registration. We also proposed changes to the ICP by adding different distance metrics to the algorithm. Additionally, we implemented a per-part ICP using label information.

We have created two distinct approaches, one incorporating label information (SPCR-PCA) and one without the information (PCR-PCA). However, before discussing the registration process, it is essential to look into the calculation of the ICP threshold and explore methods for estimating the best registration results.

2.6.1 ICP Threshold Calculation

An important step when using ICP is to estimate a reasonable ICP threshold that allows the registration to converge successfully. We use the calculations from this section in both the global registration and fine registration process. Firstly, we tried to calculate the threshold t by finding the mean of the minimum Euclidean distances between each point in the source point cloud A with n points and its closest counterpart in the target point cloud B. Mathematically, this process can be represented as:

$$t = \frac{1}{n} \sum_{a \in A} \min_{b \in B} \|a - b\| \tag{2.5}$$

This approach has, however, returned large values, as the point clouds overlap only partially, meaning distant points strongly influence this calculation.

Next, we utilized the fact that we have already defined a neighborhood in the noise reduction section 2.5.3 as $2 \cdot v$, where v is the voxel size, which is essentially the resolution of the point clouds. We used this to calculate the threshold as the mean value of points that have corresponding points in their neighborhood.

$$\mathrm{t} = \frac{1}{|A|} \sum_{a \in A} \left(\min_{b \in B} \left\{ \begin{cases} ||a - b|| & \text{if } ||a - b|| < 2 \cdot v \\ 0 & \text{otherwise} \end{cases} \right\} \right) \tag{2.6}$$

This approach gave excellent threshold approximation for both rough and fine registration.

2.6.2 Choosing the best Registration

As the pipeline requires automatic selection of the best registration, we had to create a scoring function for them. Choosing a single metric proved to be inaccurate in many circumstances. We combined all three metrics, as it was the most robust method. Given a registration with fitness f, correspondence set c and RMSE r, each registration is given a score of:

$$b_i = \frac{f_i}{f_{max}} + \frac{c_i}{c_{max}} - \frac{r_i}{r_{max}} \tag{2.7}$$

f_{max}, c_{max}, and r_{max} are the largest values observed in their respective fields compared to the registration being evaluated. The registration with the highest score b_i is then the best one.

2.6.3 Global Registration

This section explains the global registration method incorporating label information (SPCR-PCA). The method working on the entire point cloud (PCR-PCA) is implemented similarly, omitting batching.

Firstly, we split both point clouds by the label information. We run the following algorithm separately for each corresponding pair of labels. We demonstrate the algorithm on the Bildstein 1 point cloud from the Semantic3D dataset with the ”buildings” batch:

1. Firstly, we find the principal axes of each point cloud, as seen in Fig. 2.10, showing us the subspace with the most significant amount of variance in the data.

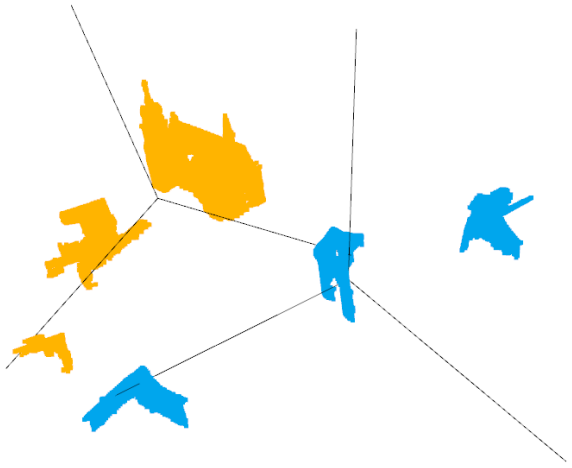


Figure 2.10 Three most prominent principal axes on the subsampled dataset. The source point cloud is yellow, and the target point cloud is blue.

2. Next, we align the principal axes by finding the translation and rotation between the principal vectors, using the equations 1.3 – 1.7. The result of this transformation aligns the axes, as in Fig. 2.11.

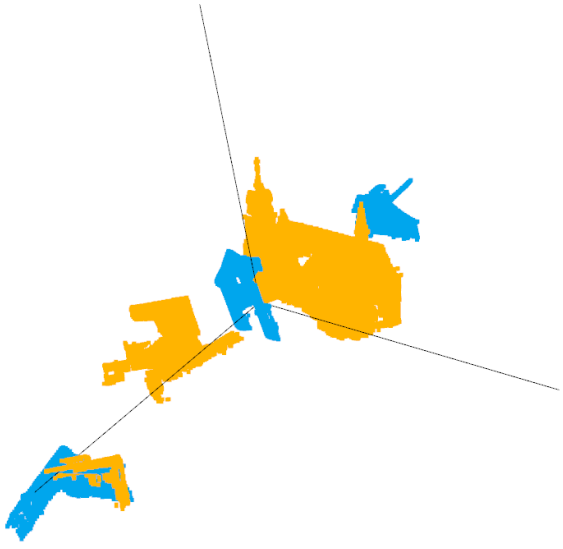


Figure 2.11 Aligned principal components after finding the transformation between them.

3. Because PCA provides us with a sub-space (plane in case of a 3D point cloud) with the most significant variance, not vectors, we should also consider vectors going in the opposite directions. Therefore, we append all possible rotations around the principal axes. Fig. 2.12 below shows rotation around one principal component.

Figure 2.12 Example of a rotated point cloud around one of the principal axes.

4. For each rotation, we run the ICP algorithm to find a better correspondence and refine the alignment. Such alignment refinement from Fig. 2.12 can be seen in Fig. 2.13.

Figure 2.13 Point cloud from Fig. 2.12 after ICP registration was completed.

5. Finally, from all the refined alignments, we choose the one that fits the original dataset the best. The alignment from Fig. 2.13 on the entire dataset is visualized in Fig. 2.14.

Figure 2.14 The best transformation from Fig. 2.13, shown on the entire point cloud.

After we have completed this for each batch of labels, we can estimate the best rough transformation, either by programmatically choosing the best transformation or averaging the results.

2.6.4 Distance functions in ICP

We took two approaches to improve the fine registration process. We have modified the open3d [53] implementation of point-to-point ICP to accept additional distance metrics and allow per-part ICP. We examine whether using different distance metrics or per-part registration can affect the process.

We have implemented the distance metrics listed below. Let $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ represent two points in a 3D point cloud.

- Euclidean Distance is widely used for point cloud registration and represents the straight line distance between two points. It is defined as:

$$D_E = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{2.8}$$

- Chebyshev Distance is the maximum absolute difference along any coordinate dimension between two points. For 3D points, it is defined as:

$$D_C = \max\left(|x_2 - x_1|, |y_2 - y_1|, |z_2 - z_1|\right) \tag{2.9}$$

- Minkowski Distance generalizes the Euclidean and Manhattan distances. For 3D point clouds, it is given by:

$$D_M = \left(|x_2 - x_1|^p + |y_2 - y_1|^p + |z_2 - z_1|^p\right)^{\frac{1}{p}}, \tag{2.10}$$

where p is a hyperparameter that adjusts the metric. Specifically, $p = 2$ yields the Euclidean distance, and $p = 1$ results in the Manhattan distance.

We used a similar method described in the initial alignment section 2.6.3 for the per-part registration. We used the label information to create batches, which we then registered separately. However, we do not apply any preprocessing steps.

After applying these above-described changes, we have deduced that the changes did not produce significant improvements. The per-part registration proved to be slightly more accurate regarding fitness and RMSE, but there were problems with merging the parts, as some batches would collapse in entirely wrong directions. Different distance metrics also did not provide significant improvements; an overview of the mean percentage improvement on the Semantic 3D dataset can be seen in Table 2.2. As the original ICP algorithm was very effective after the global registration itself and inaccuracies were made by subsampling rather than inaccurate fine registration, we did not look deeper into this problem.

Table 2.2 Average improvement of different distance metrics compared to the Euclidean distance on the Semantic3D dataset.

Metric	Fitness increase	RMSE increase	Correspondence set increase
Euclidean	0%	0%	0%
Chebyshev	-1.60%	0.41%	-1.60%
Minkowski (p=3)	-0.23%	0.01%	-0.23%

2.7 Final Pipeline

Based on the experimental results, the final registration pipeline preserves the outline presented in the implementation chapter 2, with the preprocessing steps discussed above. It uses the ConvPoint neural network architecture [59] and per-part registration, which was more robust in complicated data sets.

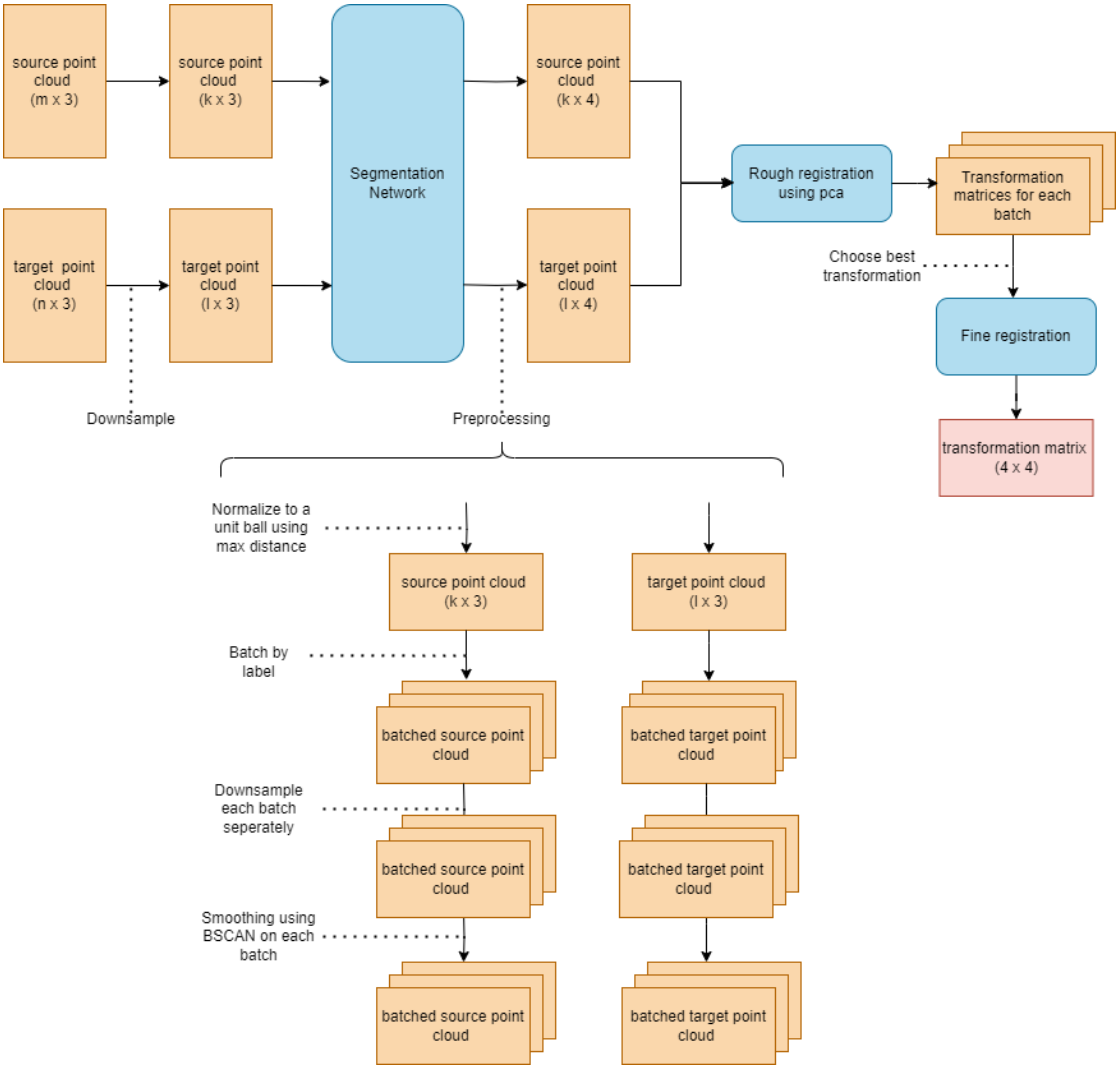


Figure 2.15 Illustration of the final pipeline, the downsampling uses a voxel size of 0.005, and DBSCAN uses a search radius twice that. Choosing the best registration is done using the formula described in section 2.6.2

Chapter 3
Results

This chapter explores how adding label information to SPCR-PCA impacted the results compared to PCR-PCA. We also compare our methods against global registration using FPFH with different ICP algorithm variants implemented in Open3D. We use all the metrics described in section 1.8 to evaluate the results. These are visual check, fitness, correspondence set size, and RMSE.

All algorithms ran on a computer with the specifications listed below.

- **CPU:** Amd Ryzen 9 5950X 16-core
- **GPU:** Nvidia RTX 4070, 12 Gb.
- **RAM:** 128 Gb.

3.0.1 Experimental Results on S3DIS

Overall, 48 point clouds from the S3DIS dataset were used to evaluate our proposed method against traditional approaches. The average fitness and RMSE of the tested algorithms can be seen in Table 3.1. SPCR-PCA performed the best, having the highest average fitness and RMSE, finding correct alignment on all tested point clouds. Methods utilizing feature histograms with ICP smoothing also performed well on most data.

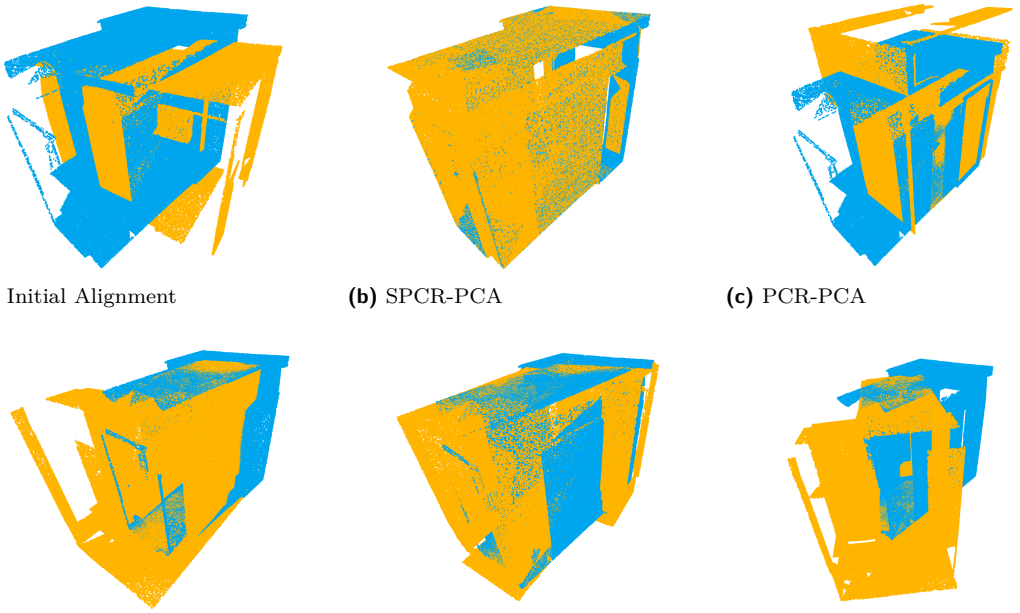
Table 3.1 Average results of the methods on S3DIS dataset. Higher fitness and lower RMSE is better.

Method	Fitness	RMSE	Num. of correspondences
SPCR-PCA	0.74782	0.00102	504 820
PCR-PCA	0.48820	0.00216	339 998
FPFH + Point2Point ICP	0.64201	0.00137	460 018
FPFH + Point2Plane ICP	0.67368	0.00123	473 165
FPFH + Generalized ICP	0.67758	0.00124	479 693

The decrease in fitness across all algorithms using FPFH is because of inconsistency in data with many points with similar features like hallways or pantries, where the RANSAC algorithm needed to match the corresponding points correctly. This can be seen in Fig. 3.1 – 3.2.

This dataset has shown that SPCR-PCA is more stable than traditional approaches, especially on point clouds that lack significant characteristics. PCR-PCA performed the worst of all methods, only working on approximately half of the tested point clouds and showing the worst overall fitness and RMSE.

Hallway 3

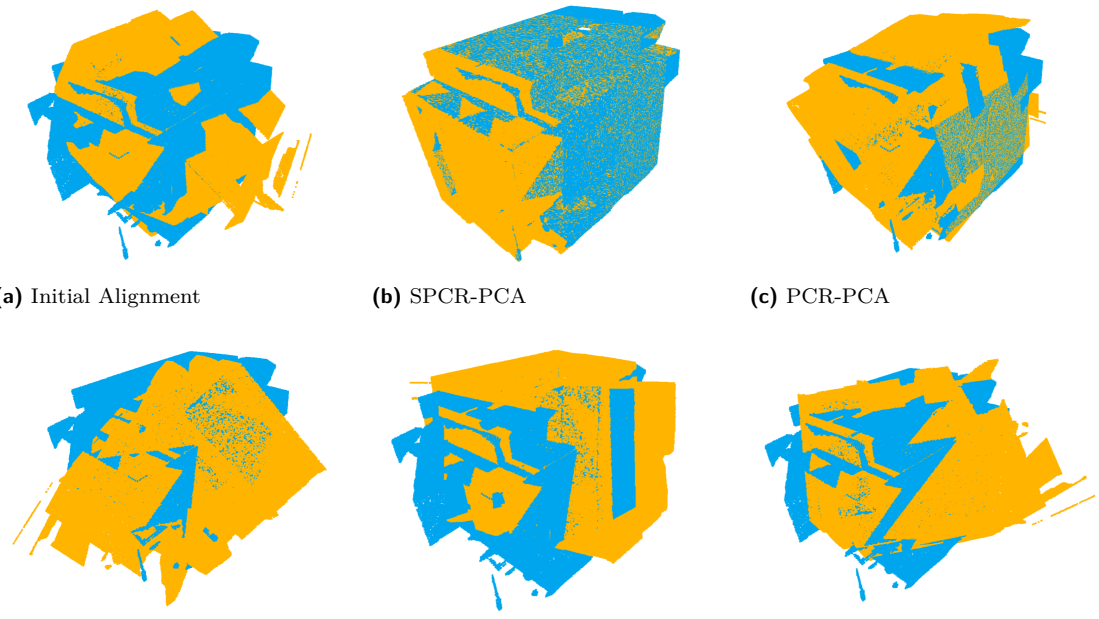


(a) Initial Alignment **(b)** SPCR-PCA **(c)** PCR-PCA

(d) FPFH + Point2Point ICP **(e)** FPFH + Point2Plane ICP **(f)** FPFH + Generalized ICP

Figure 3.1 Results of different registration algorithms. Only SPCR-PCA found the correct alignment.

Pantry 1



(a) Initial Alignment **(b)** SPCR-PCA **(c)** PCR-PCA

(d) FPFH + Point2Point ICP **(e)** FPFH + Point2Plane ICP **(f)** FPFH + Generalized ICP

Figure 3.2 Results of different registration algorithms. SPCR-PCA found the correct alignment.

3.0.2 Results on NPM3D

Three manually parted point clouds were used from the NPM3D dataset, and the average result of fitness and RMSE can be seen in Table 3.2. All tested methods except PCR-PCA performed exceptionally, finding the correct alignment for all point clouds.

Table 3.2 Average results of the methods on NPM3D dataset. Higher fitness and lower RMSE is better.

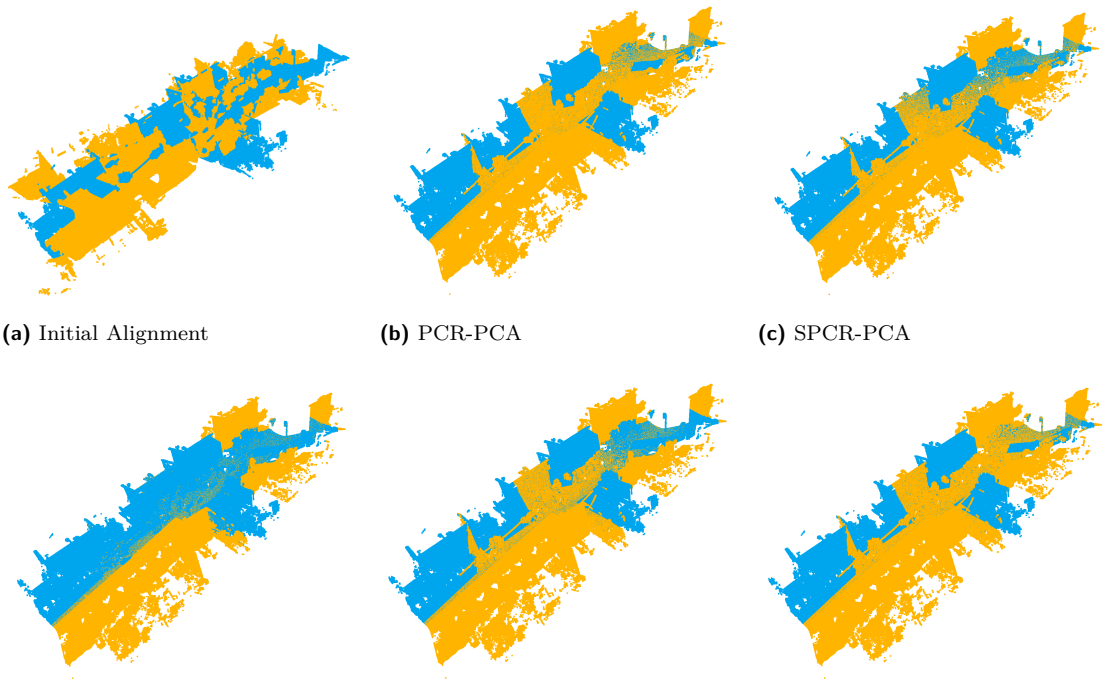
Method	Fitness	RMSE	Num. of correspondences
SPCR-PCA	0.65017	0.00041	4 774 984
PCR-PCA	0.49463	0.00147	3 627 842
FPFH + Point2Point ICP	0.65037	0.00041	4 776 499
FPFH + Point2Plane ICP	0.65014	0.00041	4 774 738
FPFH + Generalized ICP	0.65017	0.00041	4 774 981

Results on each of the datasets separately can be seen in Table 3.3. We can see that PCR-PCA failed in two of three cases, ending in a local minima. The visualization of the registration result from the Table 3.3 can be seen in Fig. 3.3 – 3.5. The differences in metrics between FPFH and SPCR-PCA methods are insignificant and result from the different fine alignment metrics used in the ICP.

Table 3.3 All results on the NPM3D dataset. Higher fitness and number of correspondences are better, and lower RMSE is better.

Method	Fitness	RMSE	Num. of correspondences
Ajaccio 2			
SPCR-PCA	0.50408	0.00066	3 994 296
PCR-PCA	0.50407	0.00066	3 994 276
FPFH + Point2Point ICP	0.50457	0.00066	3 998 162
FPFH + Point2Plane ICP	0.50396	0.00066	3 993 385
FPFH + Generalized ICP	0.50406	0.00066	3 994 129
Ajaccio 57			
SPCR-PCA	0.69953	0.00024	6 462 677
PCR-PCA	0.44708	0.00189	4 130 338
FPFH + Point2Point ICP	0.69957	0.00025	6 463 044
FPFH + Point2Plane ICP	0.69956	0.00024	6 462 901
FPFH + Generalized ICP	0.69955	0.00024	6 462 872
Dijon 9			
SPCR-PCA	0.74690	0.00033	3 867 980
PCR-PCA	0.53274	0.00186	2 758 911
FPFH + Point2Point ICP	0.74696	0.00033	3 868 290
FPFH + Point2Plane ICP	0.74689	0.00033	3 867 927
FPFH + Generalized ICP	0.74690	0.00033	3 867 941

Ajaccio 2

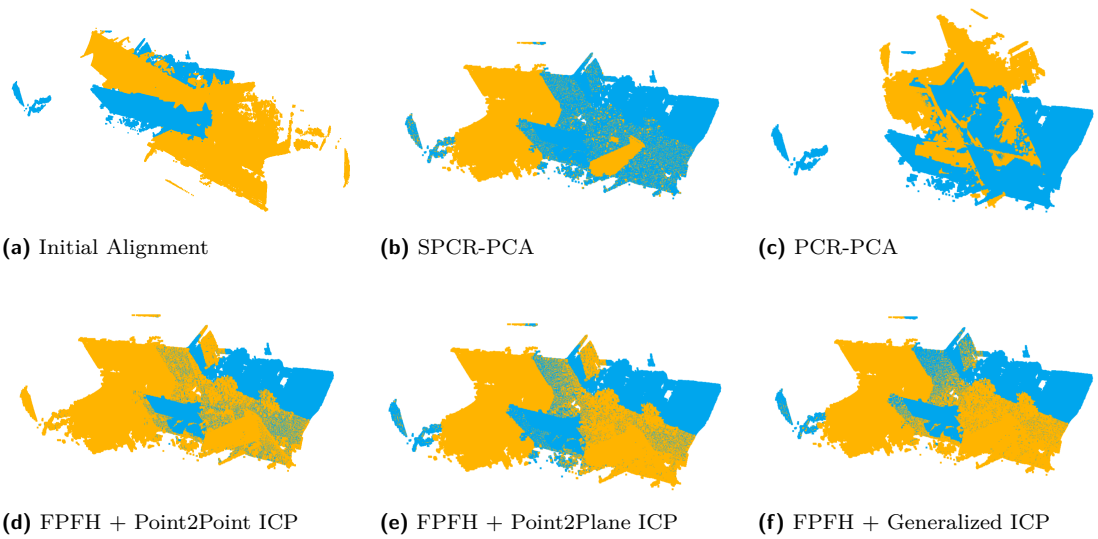


(a) Initial Alignment **(b)** PCR-PCA **(c)** SPCR-PCA

(d) FPFH + Point2Point **(e)** FPFH + Point2Plane **(f)** FPFH + Generalized

Figure 3.3 Result of the tested registration algorithms. All methods found a correct alignment.

Ajaccio 57

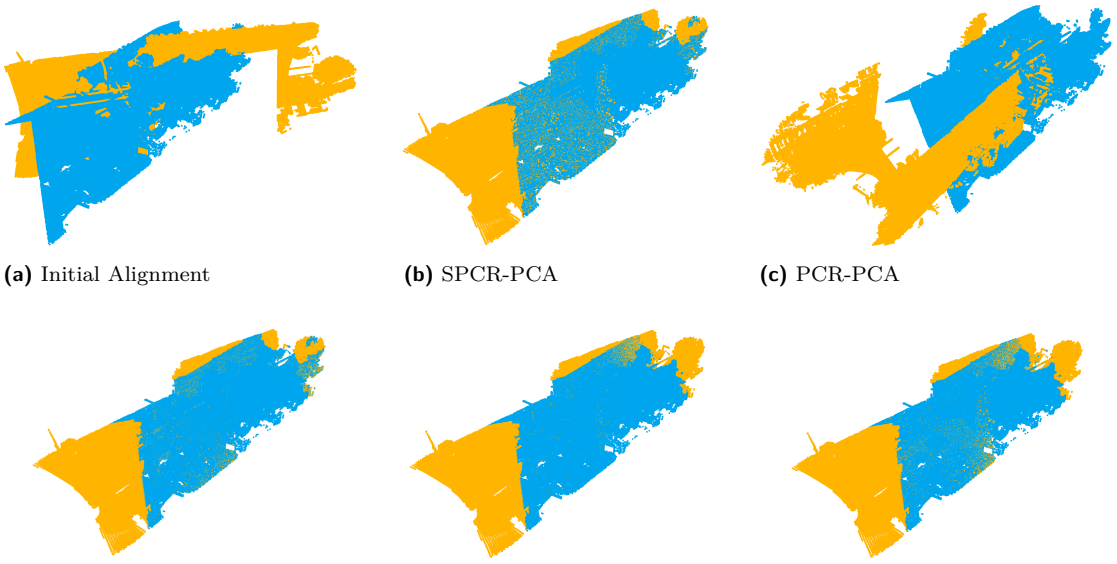


(a) Initial Alignment **(b)** SPCR-PCA **(c)** PCR-PCA

(d) FPFH + Point2Point ICP **(e)** FPFH + Point2Plane ICP **(f)** FPFH + Generalized ICP

Figure 3.4 Result of the tested registration algorithms. All methods except PCR-PCA found the correct alignment. PCR-PCA got stuck in a local minima

Dijon 9



(a) Initial Alignment **(b)** SPCR-PCA **(c)** PCR-PCA

(d) FPFH + Point2Point ICP **(e)** FPFH + Point2Plane ICP **(f)** FPFH + Generalized ICP

Figure 3.5 Result of the tested registration algorithms. All methods except PCR-PCA found the correct alignment. PCR-PCA got stuck in a local minima

3.0.3 Results on Semantic3D

Five point cloud pairs were used from the Semantic3D dataset, and the average fitness and RMSE of the tested methods can be seen in Table 3.4. Again, we can see similar results in fitness and RMSE for all FPFH methods. SPCR-PCA has better average fitness, resulting from finding a better fit on the Marketplace Feldkirch 1–4 point clouds. However, It did not find the correct alignment; this is most likely because this dataset consists mainly of building, meaning the label distribution is not widespread enough for the semantic part to play a significant role.

Table 3.4 Average results of the methods on Semantic3D dataset. Higher fitness and lower RMSE is better.

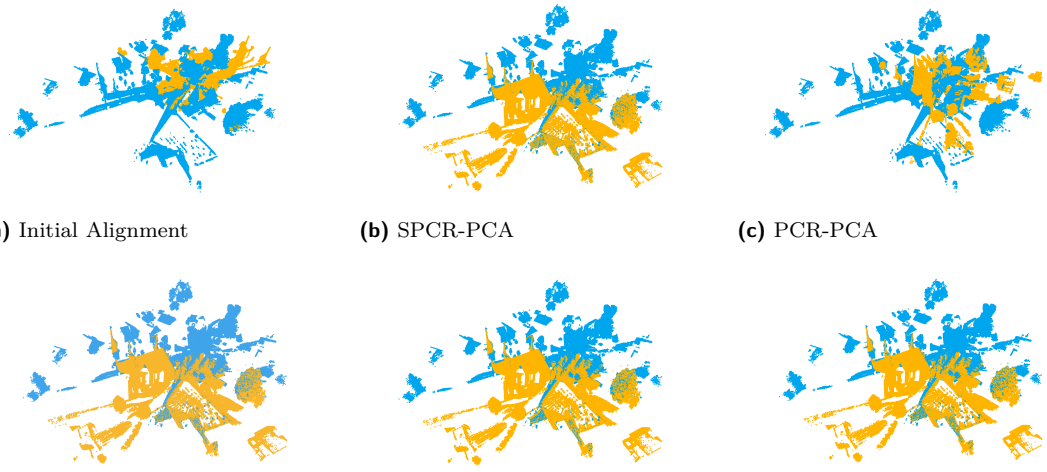
Method	Fitness	RMSE	Num. of correspondences
SPCR-PCA	0.58478	0.00124	474 004
PCR-PCA	0.34507	0.00168	241 432
FPFH + Point2Point ICP	0.53773	0.00128	447 272
FPFH + Point2Plane ICP	0.53547	0.00126	446 111
FPFH + Generalized ICP	0.52181	0.00135	438 446

Table 3.5 shows the numerical results on individual point cloud pairs. And the corresponding images in Fig. 3.6 – 3.10. Like in previous datasets, the results are similar on most point clouds. However, SPCR-PCA performs better on the Marketplace Feldkirch 1–4 point clouds. Even though it is not a perfect alignment, it roughly aligned the point clouds into close proximity. Other methods failed.

Table 3.5 Numerical results from the registration process on the Semantic3D dataset. Higher fitness and num. correspondences are better, and lower RMES is better.

Method	Fitness	RMSE	Num. of Correspondences
Bildstein 1–3			
SPCR-PCA	0.49758	0.00167	261 354
PCR-PCA	0.29649	0.00256	155 731
FPFH + Point2Point ICP	0.49855	0.00167	261 865
FPFH + Point2Plane ICP	0.49942	0.00167	262 320
FPFH + Generalized ICP	0.49779	0.00167	261 466
Untermaederbrunnen 1–3			
SPCR-PCA	0.54095	0.00096	259 840
PCR-PCA	0.21710	0.00177	104 280
FPFH + Point2Point ICP	0.54122	0.00098	259 969
FPFH + Point2Plane ICP	0.54150	0.00097	260 105
FPFH + Generalized ICP	0.54104	0.00097	259 885
Domfountain 1–3			
SPCR-PCA	0.76144	0.00092	301 073
PCR-PCA	0.76138	0.00092	301 051
FPFH + Point2Point ICP	0.76328	0.00099	301 801
FPFH + Point2Plane ICP	0.76280	0.00094	301 610
FPFH + Generalized ICP	0.76191	0.00094	301 260
Marketplace Feldkirch 1–4			
SPCR-PCA	0.41898	0.00181	236 562
PCR-PCA	0.14792	0.00197	83 521
FPFH + Point2Point ICP	0.18097	0.00190	102 178
FPFH + Point2Plane ICP	0.16858	0.00185	95 185
FPFH + Generalized ICP	0.10324	0.00236	58 294
St. Gallen Cathedral 1–3			
SPCR-PCA	0.70497	0.00084	1 311 190
PCR-PCA	0.30247	0.00115	562 577
FPFH + Point2Point ICP	0.70462	0.00086	1 310 545
FPFH + Point2Plane ICP	0.70505	0.00084	1 311 337
FPFH + Generalized ICP	0.70504	0.00084	1 311 326

Bildstein 1 – 3

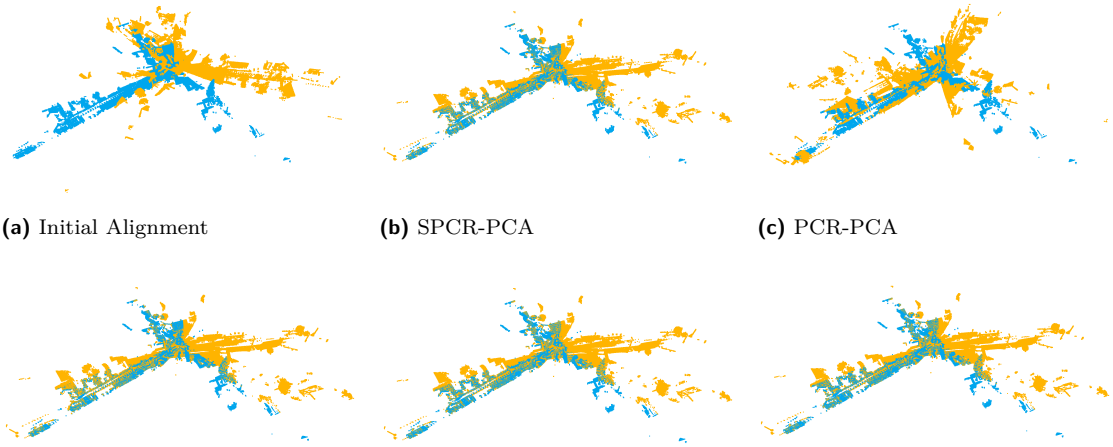


(a) Initial Alignment **(b)** SPCR-PCA **(c)** PCR-PCA

(d) FPFH + Point2Point ICP **(e)** FPFH + Point2Plane ICP **(f)** FPFH + Generalized ICP

Figure 3.6 Result of the tested registration algorithms. All methods except PCR-PCA found the correct alignment. PCR-PCA got stuck in a local minima.

Untermaederbrunnen 1 – 3

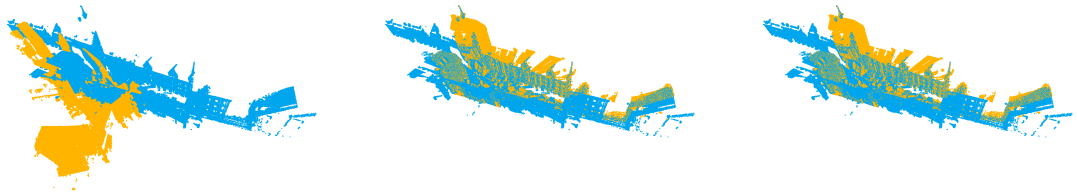


(a) Initial Alignment **(b)** SPCR-PCA **(c)** PCR-PCA

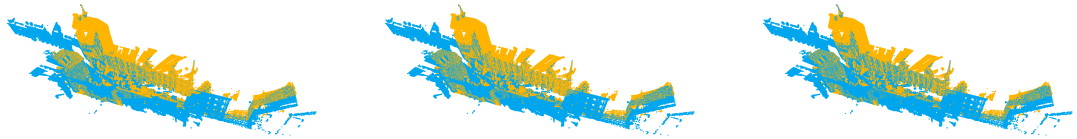
(d) FPFH + Point2Point ICP **(e)** FPFH + Point2Plane ICP **(f)** FPFH + Generalized ICP

Figure 3.7 Result of the tested registration algorithms. All methods except PCR-PCA found the correct alignment. PCR-PCA got stuck in a local minima.

Domfountain 1 – 3



(a) Initial Alignment **(b)** SPCR-PCA **(c)** PCR-PCA



(d) FPFH + Point2Point ICP **(e)** FPFH + Point2Plane ICP **(f)** FPFH + Generalized ICP

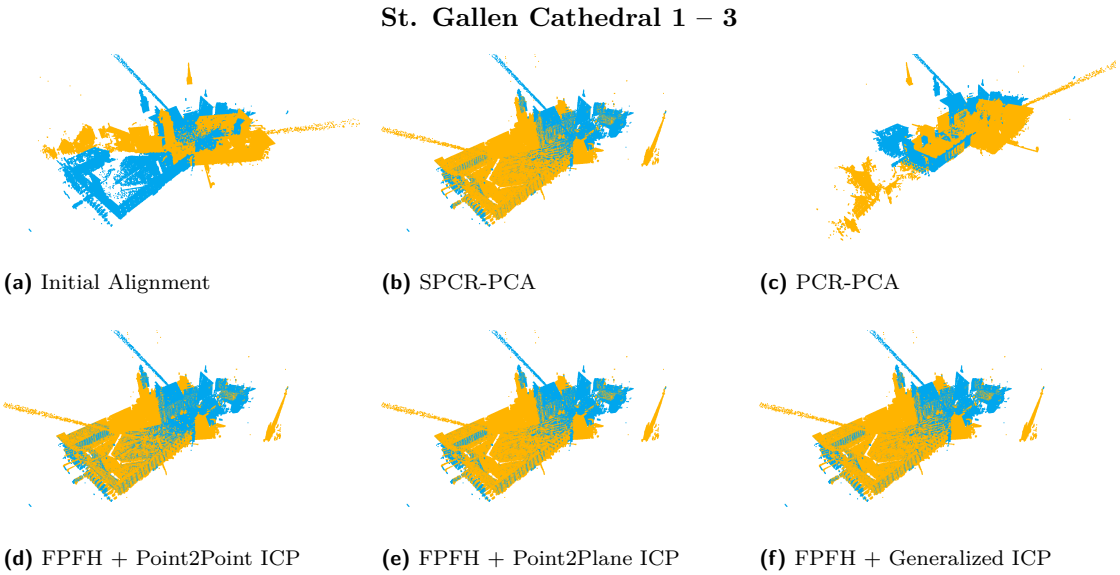
Figure 3.8 Result of the tested registration algorithms. All methods found the correct alignment.

Marketplacefeldkirch 1 – 4

(a) Initial Alignment **(b)** SPCR-PCA **(c)** PCR-PCA

(d) FPFH + Point2Point ICP **(e)** FPFH + Point2Plane ICP **(f)** FPFH + Generalized ICP

Figure 3.9 Result of the tested registration algorithms. No method had found the correct alignment, however SCPR-PCA found the best alignment.



(a) Initial Alignment **(b)** SPCR-PCA **(c)** PCR-PCA

(d) FPFH + Point2Point ICP **(e)** FPFH + Point2Plane ICP **(f)** FPFH + Generalized ICP

Figure 3.10 Result of the tested registration algorithms. All methods except PCR-PCA found the correct alignment. PCR-PCA got stuck in a local minima.

3.1 Summary of the Results

Integrating semantic labels into the SPCR-PCA significantly improved alignment accuracy and stability compared to the label-less PCR-PCA across all datasets. The addition of semantic labels helps SPCR-PCA distinguish and align local structures rather than focusing on the entire complex environment.

All algorithms using FPFH showed good accuracy on most available data. However, SPCR-PCA exhibited better stability on point clouds that lacked distinct features, such as alleys and hallways, where FPFH failed to match corresponding points. This enhanced stability of SPCR-PCA explains the observed differences in average fitness and RMSE on the S3DIS and Semantic3D datasets. The change was made because several point clouds were completely misaligned by the FPFH algorithm, significantly reducing the average performance metrics.

This experiment shows the potential of semantic label integration in the point cloud registration process, improving the accuracy and reliability of 3D data alignment across diverse environments and datasets.

Chapter 4
Discussion

This work proposed two methods to register partially overlapping large-scale point clouds. SPCR-PCA, which utilizes label information, performs a per-part global registration, and PCR-PCA performs the global registration on the entire point clouds. We predicted that the SPCR-PCA would be superior, as it would align local structures rather than focus on the entire point cloud, which is a more complex task. This hypothesis was experimentally proven to be correct, as the label method performed exceptionally well on most datasets, while the traditional approach only worked on significantly directionally oriented data. Additionally, compared to traditional registration approaches using FPFH, our method proved to be more robust, finding correct alignment on a more significant number of point clouds.

We thoroughly surveyed the topic in the analysis chapter 1. We discussed and evaluated the creation of a segmentation pipeline in section 2.4. We successfully explored and implemented the registration algorithm in the initial alignment section 2.6.3. The implementation of distance metrics was discussed in section 2.6.4, where we concluded that they do not significantly affect the results. We presented the evaluation metrics in section 1.8 and the results in section 3. Therefore, we can confidently state that we have completed all the assigned tasks.

4.1 Contributions

This work presents a promising new method for registering large-scale, partially overlapping point clouds. The proposed methods, focusing on aligning local structures and utilizing label information, hold great potential for improving the accuracy and robustness of point cloud registration.

4.2 Limitations

The pipeline demonstrated robust performance when datasets, comprised of indoor and outdoor scans with a wide label distribution, were applied. However, we have seen the method perform suboptimally on dataset where one label group had significant representation and others carried little to no information. It is also important to note that the pipeline's reliance on centralizing point clouds may lead to suboptimal outcomes in scenarios where the overlap between datasets is minimal, further complicating the registration task.

4.3 Future Work

In ongoing work, we are committed to further testing and refining the proposed methods. We plan to evaluate the pipeline on various indoor and outdoor datasets, including those with imbalanced label representation. Additionally, we aim to enhance the initial alignment method to effectively handle cases with a strong representation of points within a single-label group.

Chapter 5

Conclusion

We have successfully created a pipeline for the automatic registration of partially overlapping point clouds and completed all points from the assignment. We have experimentally deduced that incorporating semantic labels into the alignment phase has significantly improved the precision of the initial registration step. By leveraging these labels, our method has shifted the focus towards localized structural features, facilitating a more nuanced understanding of the point cloud data rather than treating it as a whole. This has drastically improved the overall accuracy on all evaluated point clouds.

Bibliography

1. WANG, Xin; PAN, HuaZhi; GUO, Kai; YANG, Xinli; LUO, Sheng. The evolution of LiDAR and its application in high precision measurement. *IOP Conference Series: Earth and Environmental Science* [online]. 2020, vol. 502, no. 1, p. 1208. ISSN 1755-1315. Available from DOI: 10.1088/1755-1315/502/1/012008.
2. LI, You; IBANEZ-GUZMAN, Javier. Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems. *IEEE Signal Processing Magazine* [online]. 2020, vol. 37, no. 4, pp. 50–61. ISSN 1053-5888. Available from DOI: 10.1109/MSP.2020.2973615.
3. GÖHRING, Daniel; WANG, Miao; SCHNÜRMACHER, Michael; GANJINEH, Tinosch. Radar/lidar sensor fusion for car-following on highways. In: *The 5th International Conference on Automation Robotics and Applications* [online]. IEEE, 2011, pp. 407–412. Available from DOI: 10.1109/ICARA.2011.6144918.
4. YANG, Tao; LI, You; ZHAO, Cheng; YAO, Dexin; CHEN, Guanyin; SUN, Li; KRAJNÍK, Tomás; YAN, Zhi. 3D ToF LiDAR in Mobile Robotics: A Review. *ArXiv* [online]. 2022, vol. abs/2202.11025. Available from DOI: 10.48550/arXiv.2202.11025.
5. CHASE, Arlen; CHASE, Diane; WEISHAMPEL, John; DRAKE, Jason; SHRESTHA, R.; SLATTON, K.; AWE, Jaime; CARTER, William. Airborne LiDAR, archaeology, and the ancient Maya landscape at Caracol, Belize. *Journal of Archaeological Science* [online]. 2011, vol. 38, pp. 387–398. ISSN 0305-4403. Available from DOI: 10.1016/j.jas.2010.09.018.
6. WANG, Ruisheng. 3D building modeling using images and LiDAR: a review. *International Journal of Image and Data Fusion* [online]. 2013, vol. 4, no. 4, pp. 273–292. ISSN 1947-9832. Available from DOI: 10.1080/19479832.2013.811124.
7. KÜHNER, Tilman; KÜMMERLE, Julius. Large-Scale Volumetric Scene Reconstruction using LiDAR. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)* [online]. IEEE, 2020, pp. 6261–6267. Available from DOI: 10.1109/ICRA40945.2020.9197388.
8. ZHANG, Juyong; YAO, Yuxin; DENG, Bailin. Fast and Robust Iterative Closest Point. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2022, vol. 44, no. 7, pp. 3450–3466. ISSN 0162-8828. Available from DOI: 10.1109/TPAMI.2021.3054619.
9. BOUAZIZ, Sofien; TAGLIASACCHI, Andrea; PAULY, Mark. Sparse iterative closest point. In: *Computer graphics forum* [online]. Wiley Online Library, 2013, vol. 32, pp. 113–123. No. 5. ISSN 1467-8659. Available from DOI: 10.1111/cgf.12178.

10. BOULCH, Alexandre; GUERRY, Joris; LE SAUX, Bertrand; AUDEBERT, Nicolas. SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. *Computers & Graphics* [online]. 2018, vol. 71, pp. 189–198. ISSN 0097-8493. Available from DOI: 10.1109/ICCVW.2017.85.

11. MATURANA, Daniel; SCHERER, Sebastian. Voxnet: A 3d convolutional neural network for real-time object recognition. In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 922–928. ISSN 2153-0858. Available from DOI: 10.1109/IROS.2015.7353481.

12. QI, Charles R; SU, Hao; MO, Kaichun; GUIBAS, Leonidas J. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition* [online]. 2017, pp. 652–660. ISSN 1063-6919. Available from DOI: 10.1109/CVPR.2017.16.

13. QI, Charles Ruizhongtai; YI, Li; SU, Hao; GUIBAS, Leonidas J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems* [online]. 2017, vol. 30. ISSN 1049-5258. Available from DOI: 10.48550/arXiv.1706.02413.

14. BOULCH, Alexandre. ConvPoint: Continuous convolutions for point cloud processing. *Computers & Graphics* [online]. 2020, vol. 88, pp. 24–34. ISSN 0097-8493. Available from DOI: 10.48550/arXiv.1904.02375.

15. OPEN3D DEVELOPMENT TEAM. *open3d::basic::PointCloud Class Reference* [online]. Open3D Documentation, 2022. Available also from: https://www.open3d.org/docs/latest/tutorial/Basic/pointcloud.html. [Accessed 2024-04-03].

16. BENTLEY, Jon Louis. Multidimensional binary search trees used for associative searching. *Communications of the ACM* [online]. 1975, vol. 18, no. 9, pp. 509–517. ISSN 0001-0782. Available from DOI: 10.1145/361002.361007.

17. GARCIA-GARCIA, Alberto. *Towards a real-time 3D object recognition pipeline on mobile GPGPU computing platforms using low-cost RGB-D sensors* [Bachelor Thesis]. University of Alicante, 2015.

18. BTYNER. *3dtree* [online]. Wikimedia Commons, 2006. Available also from: https://commons.wikimedia.org/wiki/File:3dtree.png. [Accessed 2024-01-30].

19. MAĆKIEWICZ, Andrzej; RATAJCZAK, Waldemar. Principal components analysis (PCA). *Computers Geosciences* [online]. 1993, vol. 19, no. 3, pp. 303–342. ISSN 0098-3004. Available from DOI: https://doi.org/10.1016/0098-3004(93)90090-R.

20. LI, Wei; LI, Zhibin; ZHU, XinKai; CHANG, JiaWei. Point Cloud Registration Algorithm Fusing PCA and NDT. In: *2023 38th Youth Academic Annual Conference of Chinese Association of Automation (YAC)* [online]. 2023, pp. 75–80. ISSN 2691-1242. Available from DOI: 10.1109/YAC59482.2023.10401822.

21. SCHUBERT, Erich; SANDER, Jörg; ESTER, Martin; KRIEGEL, Hans Peter; XU, Xiaowei. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. In: *ACM Transactions on Database Systems* [online]. New York, NY, USA: Association for Computing Machinery, 2017, vol. 42. No. 3. ISSN 0362-5915. Available from DOI: 10.1145/3068335.

22. CHIRE. *DBSCAN illustration* [online]. Wikimedia Commons, 2011. Available also from: https://commons.wikimedia.org/wiki/File:DBSCAN-Illustration.svg. [Accessed 2024-01-21].

23. C., Bodenstein. *Analysis of 3D Point Clouds using a Parallel DBSCAN Clustering Algorithm* [online]. Jülich Supercomputing Centre (JSC), Germany, 2015. Available also from: https://juser.fz-juelich.de/record/276079/files/inside_autumn15-32.pdf.

24. LI, Lin; YANG, Fan; ZHU, Haihong; LI, Dalin; LI, You; TANG, Lei. An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells. *Remote Sensing* [online]. 2017, vol. 9, no. 5. ISSN 2072-4292. Available also from: https://www.mdpi.com/2072-4292/9/5/433.

25. MSM. *Line with outliers* [Online]. Wikimedia Commons, 2007. Available also from: https://commons.wikimedia.org/wiki/File:Line_with_outliers.svg. [Accessed 12-04-2024].

26. MSM. *Fitted line* [Online]. Wikimedia Commons, 2007. Available also from: https://commons.wikimedia.org/wiki/File:Fitted_line.svg. [Accessed 12-04-2024].

27. OPEN3D DEVELOPMENT TEAM. *Transformation* [online]. Open3d, 2020. Available also from: https://www.open3d.org/docs/latest/tutorial/Basic/transformation.html. [Accessed 27-03-2024].

28. NGHIAHO. *Finding optimal rotation and translation between corresponding 3D points* [online]. 2011. Available also from: https://nghiaho.com/?page_id=671. [Accessed 01-04-2024].

29. KHAN, Asharul Islam; AL-HABSI, Salim. Machine learning in computer vision. *Procedia Computer Science* [online]. 2020, vol. 167, pp. 1444–1451. ISSN 1877-0509. Available from DOI: 10.1016/j.procs.2020.03.355.

30. MAIND, Sonali B; WANKAR, Priyanka, et al. Research paper on basic of artificial neural network. *International Journal on Recent and Innovation Trends in Computing and Communication* [online]. 2014, vol. 2, no. 1, pp. 96–100. ISSN 2321-8169. Available from DOI: 10.1002/agj2.21185.

31. GLOSSER. *Colored neural network* [online]. Wikimedia Commons, 2013. Available also from: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg.

32. SHARMA, Sagar; SHARMA, Simone; ATHAIYA, Anidhya. Activation functions in neural networks. *Towards Data Science* [online]. 2017, vol. 6, no. 12, pp. 310–316.

33. VAŠATA, Daniel. *Vícevrstvé neuronové sítě* [Handout presentation]. Czech Technical University, 2023. Lecture notes for Machine Learning 2.

34. JAIN, Anil K; MAO, Jianchang; MOHIUDDIN, K Moidin. Artificial neural networks: A tutorial. *Computer* [online]. 1996, vol. 29, no. 3, pp. 31–44. ISSN 0018-9162. Available from DOI: 10.1109/2.485891.

35. NORVIG, P Russel; INTELLIGENCE, S Artificial. A modern approach. *Prentice Hall Upper Saddle River, NJ, USA: Rani, M., Nayak, R., & Vyas, OP (2015). An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. Knowledge-Based Systems.* 2002, vol. 90, pp. 811–813. ISBN 978-0136042594.

36. LE, Quoc V et al. A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. *Google Brain* [online]. 2015, vol. 20, pp. 1–20. Available also from: https://cs.stanford.edu/~quocle/tutorial2.pdf.

37. ALBAWI, Saad; MOHAMMED, Tareq Abed; AL-ZAWI, Saad. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)* [online]. 2017, pp. 1–6. Available from DOI: 10.1109/ICEngTechnol.2017.8308186.

38. BALAJI, Sai. *Binary Image classifier CNN using TensorFlow* [online]. Wikimedia Commons, 2019. Available also from: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg. [Accessed 2024-07-04].

39. PROCHÁZKOVÁ, Jana; MARTIŠEK, Dalibor. Notes on Iterative Closest Algorithm. In: *17th Conference on Applied Mathematics Aplimat 2018 Proceedings* [online]. Institute of Mathematics and Physics, Faculty of Mechanical Engineering, Slovak University of Technology in Bratislava, 2018, pp. 876–884. ISBN 978-80-227-4765-3. Available also from: http://hdl.handle.net/11012/70943. [Accessed 2024-04-08].

40. GIRARDEAU-MONTAUT, Daniel. *CloudCompare*. Vol. 11 [online]. EDF R&D Telecom ParisTech, 2016. No. 5. Available also from: https://www.cloudcompare.org.

41. RUSU, Radu Bogdan; BLODOW, Nico; BEETZ, Michael. Fast point feature histograms (FPFH) for 3D registration. In: *2009 IEEE international conference on robotics and automation* [online]. IEEE, 2009, pp. 3212–3217. ISSN 1049-3492. Available from DOI: 10.1109/ROBOT.2009.5152473.

42. ARUN, K. S.; HUANG, T. S.; BLOSTEIN, S. D. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 1987, vol. PAMI-9, no. 5, pp. 698–700. ISSN 0162-8828. Available from DOI: 10.1109/TPAMI.1987.4767965.

43. BESL, P.J.; MCKAY, Neil D. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 1992, vol. 14, no. 2, pp. 239–256. ISSN 0162-8828. Available from DOI: 10.1109/34.121791.

44. SEGAL, Aleksandr; HAEHNEL, Dirk; THRUN, Sebastian. Generalized-icp. In: *Robotics: science and systems* [online]. Seattle, WA, 2009, vol. 2, p. 435. No. 4. Available from DOI: 10.15607/RSS.2009.V.021.

45. BIGGERJ1. *Illustration of the idea behind the Iterative Closest Point Algorithm* [Online]. Wikimedia Commons, 2020. Available also from: https://commons.wikimedia.org/wiki/File:Idea_closest_point_algorithm.svg. [Accessed 02-04-2024].

46. GAWEL, Abel; DUBÉ, Renaud; SURMANN, Hartmut; NIETO, Juan; SIEGWART, Roland; CADENA, Cesar. 3D registration of aerial and ground robots for disaster response: An evaluation of features, descriptors, and transformation estimation. In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)* [online]. IEEE, 2017, pp. 27–34. Available from DOI: 10.1109/SSRR.2017.8088136.

47. DONG, Zhen; LIANG, Fuxun; YANG, Bisheng; XU, Yusheng; ZANG, Yufu; LI, Jianping; WANG, Yuan; DAI, Wenxia; FAN, Hongchao; HYYPPÄ, Juha; STILLA, Uwe. Registration of large-scale terrestrial laser scanner point clouds: A review and benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing* [online]. 2020, vol. 163, pp. 327–342. ISSN 0924-2716. Available from DOI: https://doi.org/10.1016/j.isprsjprs.2020.03.013.

48. DENG, Haowen; BIRDAL, Tolga; ILIC, Slobodan. PPFNet: Global Context Aware Local Features for Robust 3D Point Matching. *CoRR* [online]. 2018, vol. abs/1802.02669. Available from arXiv: 1802.02669.

49. ZENG, Andy; SONG, Shuran; NIESSNER, Matthias; FISHER, Matthew; XIAO, Jianxiong. 3DMatch: Learning the Matching of Local 3D Geometry in Range Scans. *CoRR* [online]. 2016, vol. abs/1603.08182. Available from arXiv: 1603.08182.

50. ZHANG, Zhiyuan; DAI, Yuchao; SUN, Jiadai. Deep learning based point cloud registration: an overview. *Virtual Reality Intelligent Hardware* [online]. 2020, vol. 2, no. 3, pp. 222–246. ISSN 2096-5796. Available from DOI: https://doi.org/10.1016/j.vrih.2020.05.002. 3D Visual Processing and Reconstruction Special Issue.

51. BEHNEL, Stefan; BRADSHAW, Robert; CITRO, Craig; DALCIN, Lisandro; SELJEBOTN, Dag Sverre; SMITH, Kurt. Cython: The best of both worlds. *Computing in Science & Engineering* [online]. 2010, vol. 13, no. 2, pp. 31–39. ISSN 1521-9615. Available from DOI: 10.1109/MCSE.2010.118.

52. PASZKE, Adam; GROSS, Sam; MASSA, Francisco; LERER, Adam; BRADBURY, James; CHANAN, Gregory; KILLEEN, Trevor; LIN, Zeming; GIMELSHEIN, Natalia; ANTIGA, Luca; DESMAISON, Alban; KÖPF, Andreas; YANG, Edward; DEVITO, Zach; RAISON, Martin; TEJANI, Alykhan; CHILAMKURTHY, Sasank; STEINER, Benoit; FANG, Lu; BAI, Junjie; CHINTALA, Soumith. PyTorch: An Imperative Style, High-Performance Deep Learning Library [online]. 2019. Available from DOI: 10.48550/arXiv.1912.01703.

53. ZHOU, Qian-Yi; PARK, Jaesik; KOLTUN, Vladlen. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847* [online]. 2018. Available from DOI: 10.1145/3386569.3392393.

54. PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* [online]. 2011, vol. 12, pp. 2825–2830. ISSN 1533-7928. Available from DOI: 10.48550/arXiv.1201.0490.

55. ARMENI, I.; SAX, A.; ZAMIR, A. R.; SAVARESE, S. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints* [online]. 2017. Available from arXiv: 1702.01105 [cs.CV].

56. ROYNARD, Xavier; DESCHAUD, Jean-Emmanuel; GOULETTE, François. Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *The International Journal of Robotics Research* [online]. 2018, vol. 37, no. 6, pp. 545–557. ISSN 0278-3649. Available from DOI: 10.1177/0278364918767506.

57. RUSU, Radu Bogdan. Semantic 3D object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz* [online]. 2010, vol. 24, pp. 345–348. ISSN 0933-1875. Available from DOI: 10.1007/s13218-010-0059-6.

58. HACKEL, Timo; SAVINOV, N.; LADICKY, L.; WEGNER, Jan D.; SCHINDLER, K.; POLLEFEYS, M. *Lare-Scale Point Cloud Classification Benchmark* [online]. 2017. Available also from: https://www.semantic3d.net. [Accessed 07-04-2024].

59. BOULCH, Alexandre. *Convpoint* [online]. GitHub, 2019. Available also from: https://github.com/aboulch/ConvPoint. [Accessed 2024-04-07].

60. HAN, Xian-Feng; JIN, Jesse S.; WANG, Ming-Jie; JIANG, Wei; GAO, Lei; XIAO, Liping. A review of algorithms for filtering the 3D point cloud. *Signal Processing: Image Communication* [online]. 2017, vol. 57, pp. 103–112. ISSN 0923-5965. Available from DOI: https://doi.org/10.1016/j.image.2017.05.009.

61. CHEN, Liang-Chia; HUANG, Sheng-Hao; HUANG, Bo-Han. Precise 6DOF Localization of Robot End Effectors Using 3D Vision and Registration without Referencing Targets. In: 2022. ISBN 978-1-83769-987-2. Available from DOI: 10.5772/intechopen.107968.

62. SCIKIT-LEARN DEVELOPMENT TEAM. *sklearn.cluster.DBSCAN - scikit-learn Documentation* [online]. Scikit-Learn, 2024. Available also from: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html. [Accessed on 2024-03-15].

Attachments

```
README.md ..... brief description of the media content
src
    impl ..... source files of the implementation
    thesis ..... source files of the thesis in LaTeX
text ..... text
    thesis.pdf ..... thesis in the form of a PDF file
```