Chair of Computational Modeling and Simulation
TUM School of Engineering and Design
Technical University of Munich

Graph-based Entity Alignment: Adapting SGAligner for Point Cloud to BIM Alignment

Scientific work to obtain the degree

Master of Science (M.Sc.)

at the TUM School of Engineering and Design
of the Technical University of Munich.

Supervised by	Prof. Dr.-Ing. Andre Borrmann Fiona Collins M.Sc. Sebastian Esser M.Sc. Chair of Computational Modeling and Simulation
Submitted by	Binod Singh ([redacted]) [redacted] e-Mail: binod.singh@tum.de
Submitted on	26. Jan 2024

Abstract

Integration of the as-built reality represented using point cloud and as-designed generally represented using Building Information Modeling is an integral process in progress monitoring and more general Digital Twin (DT) activities. An up-to-date as-built model helps save time, and resources and allows for informed decision-making. Alignment helps update the old or outdated as-designed model to the current standards. But due to lack of digitization and BIM not being readily available, comparing point clouds taken at different phases during the building lifecycle provides an alternative approach for progress monitoring and compliance checking. Traditionally, this process involves overlaying recent point cloud data with the designed model or previous point cloud data, identifying common elements, and analyzing the differences. However, this method has limitations, especially when elements are far apart or have different shapes.

This study utilized a graph-based learning method to link semantic instances between two point clouds of a scene. Graph-based representations were derived and enriched, and multi-modal encoders were leveraged to link two semantically same building elements. Subsequently, this method was employed to investigate PCD-BIM integration.

The trained multi-modal architecture showed promising results with an Mean Reciprocal Rank (MRR) score of 0.99 when normal scenes were analyzed and an MRR score of 0.97 when deviated scenes were fed to the model. The base case had an MRR score of 0.79, showing promising results of graph-enriching methods, an increase of 25%, and demonstrating that the method is robust to positional deviations.

Using PCD-PCD to generalize PCD-BIM did not provide sufficient insights but highlighted the need for its training module.

The results showcase the proposed solution's potential for integrating PCD-PCD in the built environment. The best-performing model finds links between the same building elements without processing individual points, significantly reducing the computing time. This leads to the downstream task of finding node correspondences and registration being computationally cheaper and faster. **Keywords: Point Cloud, BIM, Entity Alignment, Graph Attention Network (GAT), Multi-Modal Encoder, Semantic, MRR**

Acknowledgement

I would very much like to extend my gratitude to Fiona for her support, guidance, and mentorship she has provided. It was very instrumental in seeing this thesis to its destination. From being ready for instantaneous meetings to wanting to go through the steps together so we could come up with ideas to solve the problem, it was a pleasure to have had the opportunity to work under her guidance.

I also would like to express my appreciation to Sebastian. This marks our second project together, and I am highly thankful for his insights and the unwavering support that he offered throughout the journey. Working with him was undoubtedly an enriching experience. A big thank you to Prof. Borrmann and the Chair of Computational Modeling and Simulation

for providing me with the invaluable resources that have been crucial to the success of my research.

Finally, to my family and friends, who have been a cornerstone of everything I have been a part of. I am deeply indebted to them.

Contents

List of Figures

List of Algorithms

List of Tables

Acronyms

2D	Two Dimensional
3D	Three Dimensional
AI	Artifical Intelligence
BIM	Building Information Modeling
CAD	Computer Aided Design
CD	Chamfer Distance
EA	Entity Alignment
FMR	Feature Matching Recall
GAT	Graph Attention Network
GCN	Graph Convolution Network
GIS	Geographic Information System
GNN	Graph Neural Network
IAL	Inter-modal Alignment Loss
ICL	Intra-modal Contrastive Loss
ICP	Iterative Closest Point
IFC	Industry Foundation Classes
IR	Inliner Ratio
KG	Knowledge Graph
KL	Kullback-Leibler
KPConv	Kernel Point Convolution
LiDAR	Light Detection and Ranging
LLMs	Large Language Models
MCLEA	Multi-modal Contrastive Learning based Entity Alignment
MCS	Maximum Common Subgraph
MRR	Mean Reciprocal Rank
NLP	Natural Language Processing
PCA	Principal Component Analysis
PCD	Point Cloud
QL4BIM	Query Lanuguage for Building Information Modeling
RANSAC	Random Sample Consensus

Continued on next page

Table 0: (Continued)

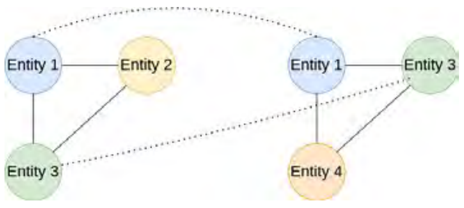
RR	Registration Recall
RRE	Relative Rotation Error
RTE	Relative Translation Error
SE	Special Euclidean group
SGAligner	Scene Graph Aligner
SO	Special Orthogonal group
SVD	Singular Value Decomposition

Chapter 1

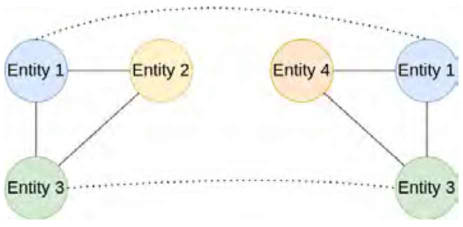
Introduction

Building Information Modeling and Point Cloud play pivotal roles in construction and built environment sectors. BIM, a digital representation of the physical and functional characteristics of a building (LEE et al., 2006), serves as a comprehensive digital representation or as-designed geometry. PCD, usually derived from Light Detection and Ranging (LiDAR) sensors offer a nuanced three-dimensional depiction of the as-built environment. Information flow between a physical building and its digital twin lies at the core of improving transparency and informed decision-making in the built environment (KRITZINGER et al., 2018). Progress monitoring during the construction phase of a building leverages both data streams and provides real-time insights into progress, quality assurance, and compliance with the design specifications (BRAUN et al., 2020). It also allows for timely intervention and adjustments to maintain the desired standards. This enables model updates and compliance checks, making progress monitoring a proactive tool to identify discrepancies between the digital model and physical structure.

Progress monitoring is achieved by integrating the point cloud of an area of interest with its digital model and analyzing the potential differences(COLLINS et al., 2023). The integration process combines data streams to establish connections between heterogeneous representations of the same building elements (SACKS et al., 2020). Traditionally, data are integrated using an iterative spatial-based method. The point cloud is acquired, pre-processed, and integrated with its digital counterpart, where the task is to create a link to the as-built point cloud representation with its historic element representation (COLLINS et al., 2023). This task of matching same entities across data streams is known as alignment (Z. WANG et al., 2018). The conceptual diagram is shown in fig. 1.1.



(a) Misaligned Data



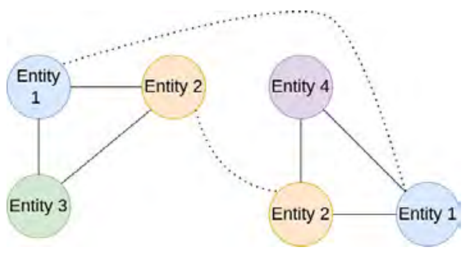
(b) Aligned Data

Figure 1.1: Alignment Conceptual Diagram. Dotted lines represent the same entity

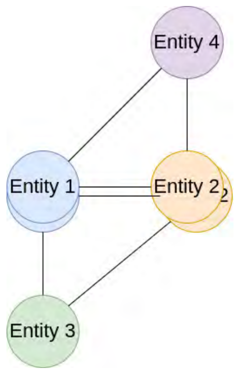
Point Cloud plays a vital role in data alignment as it is a means of documenting the changing appearance of a physical building and it allows for capturing data as the process is going on. While the scanning process is mainly manual, the captured current data can be integrated with the potentially outdated digital model to analyze potential differences (BRAUN et al., 2020).

However, this alignment approach is not without challenges. Due to the lack of digitization in the construction industry, BIM in most cases is not readily available. Usually, Two Dimensional (2D) floor plans are available and integration in such cases suffers from limitations common to Direct Visual Odometry (DVO) (KAMINSKY et al., 2009), and model updates and compliance checks are limited in such cases. Additionally, digital twins, a virtual representation of a complex physical asset in the digital space to closely characterize the operations of the original physical process or system (GRIEVES, 2015), are not necessarily based on BIM models since they are just one of the many representations. So in most cases, point clouds taken at different timesteps during the building lifecycle offer a way to track an everchanging environment (MEYER et al., 2022). As data is acquired continually throughout the building lifecycle, aligning point-cloud-to-point-cloud from different time steps presents an alternative approach to gaining valuable insights into the project's progress.

Integrating these data sources allows for concluding the timeline and compliance with regulations related to a particular building element's placement or product specifications (COLLINS et al., 2023). Furthermore, alignment can be used as a coarse initializer for registration instead of computing 3D correspondences directly on the entire point cloud (SARKAR et al., 2023). Registration involves finding the spatial transformation that aligns two point sets (LI et al., 2021). This allows for tracking the progress visually as well as analyzing further differences. The conceptual diagram for registration is shown in fig. 1.2



(a) Unregistered, aligned Data



(b) Registered, aligned Data

Figure 1.2: Registration Conceptual Diagram

This process of matching at the point level is time-consuming and technically challenging (COLLINS et al., 2023). Research has shown that graph-based techniques provide a structured and rich way to represent the built environment (SARKAR et al., 2023). Additionally, research shows that graph-based representations of digital twins are beneficial for various practical purposes (DROBNYI et al., 2024). Connectivity between elements is a feature of the graph. There also have been suggestions for digital twins and as-built reality to be represented as entity graphs where nodes denote an object. So, this thesis aims to utilize the graph-based learning approach to tackle this alignment task, where nodes are semantic entities with relationships between them. On top of that, as semantic and instance segmentation is becoming mature (Collins et al., 2023), this thesis work aims to utilize that information to find a link between the exact semantic element representation in two point clouds of a scene.

It also aims to facilitate the integration process using graph-based deep learning to fuse data and create links between building elements. First, this work investigates a deep learning network from literature (Sarkar et al., 2023) to find links between semantic elements in two point clouds of a scene. Then, the downstream registration task is investigated after the node correspondences are found. Finally, after carefully examining the network, this work explores the application of a network trained on PCD and PCD in integrating BIM and point cloud.

1.1 Background

Point cloud alignment is an ongoing field of research. Martens and Blankenbach (2018) use density-based point histograms to align point clouds which they also integrated into existing BIM modelling software. Sarode et al. (2019) use PointNet (Qi et al., 2017) encoding to align point clouds and perform registration. Qiao et al. (2023) proposed a semantic-geometric graph-theoretic method in low overlap scenarios. Two-point cloud pairs have low overlapping areas in a low overlap scenario due to occlusions or viewpoint change (Qiao et al., 2023). Sarkar et al. (2023) align scene graphs and the result is then used for registering the point sets. This is also discussed in detail in section 3.4.2. Most of these tasks are computer-vision-centric, and limited research has been done on using deep learning with graphs to align building elements.

Aligning BIM models to their point cloud counterparts is also an actively explored area of research. Rausch and Haas (2021) uses Iterative Closest Point (ICP) method to register point cloud to BIM. A vast amount of literature uses the spatial proximity method for registering BIM and point clouds. However, Chuang and Yang (2023) showed that matching based on spatial proximity poses technical complexities in case of poor registration results and in cases where the positions of the elements haven't been documented properly. Therefore, a new area of research is gaining traction where instead of point-based linking, semantic information is used, (Collins et al., 2023). This method does not deal with vast amounts of points; it focuses only on a subset of points and relies more on semantic information, making it much more computationally cheaper.

One such method is developed by Collins et al. (2023), where they use a graph-based method to link point cloud and BIM using geometrical and topological relations (Hu & Brilakis, 2024), which is one of the bases of this research work. They showed that elements can be linked despite positional and geometric deviations. However, one of the limitations that they highlighted is that most connections between the building elements should be present in both graphs.

1.2 Motivation

Apart from the general motivation behind the topic as discussed in the previous section, the technical motivation behind this thesis stems from rapid advancements in entity alignment using Deep Learning techniques, mostly in the domain of Natural Language Processing (NLP) as well as the limitations highlighted in COLLINS et al., 2023 This thesis adopts a graph-based learning method to abstract the high resolution and data volume.

SARKAR et al., 2023, developed a framework that showed how techniques developed for alignment in the text domain can be leveraged to align 3D scene graphs that have applications in robotics, localization, and navigation.

By adopting the techniques and insights from these sources, this thesis aims to develop novel methodologies that perform alignment between building elements using deep learning and address the limitations identified by COLLINS et al., 2023, ultimately enhancing the alignment accuracy in the construction domain.

1.3 Research Objective

The objective of this thesis can be subdivided into three main categories.

1. Investigate the use of graph-based deep learning methods for linking building elements between point cloud and point cloud
2. Study the effect of graph-enriching methods on graph alignment and registration task
3. Examine the performance of PCD-PCD based method on BIM-PCD alignment

1.4 Structure of thesis

Chapter 2 touches upon the fundamental concepts crucial to the thesis. It details the data types, graph data structure, entity alignment, and the various graph neural architectures.

Chapter 3 dives deeper into the topics used as a basis for this thesis work. It highlights the state-of-the-art registration technique using deep learning, and the graph formulation of point clouds and touches on the work of (COLLINS et al., 2023). Additionally, it emphasizes the importance of contrastive learning for entity alignment and provides insights into the workings of (SARKAR et al., 2023). The discussion extends to the different modalities used, and how the results are then utilized for registration. Also, different metrics related to alignment and registration are explained that are later used in this thesis work to measure the model performance.

Chapter 4 dives deeper into the specifics of the thesis work. The dataset used, algorithms used to preprocess along with the detailed architecture of the individual modalities. Finally, the full network architecture is explained.

Chapter 5 focuses on the outcomes, providing a comprehensive analysis of alignment results and their performance on registration. The results of different models are analyzed, and compared, and the best-performing model is highlighted. It emphasizes the results of graph-enriching methods and showcases the results of different ablation studies.

Chapter 6 discusses the results and the insights obtained. It also talks about the limitations of the approach.

Finally, Chapter 7 summarizes key findings and offers suggestions for future research.

Chapter 2

Theoretical Background

This research work extends and builds upon the theoretical and practical frameworks established by prior research, which will be discussed in the next chapter. This section serves as a theoretical framework for the entire research work.

2.1 Point Cloud

Point clouds are unstructured, unordered points representing an object or Three Dimensional (3D) space. The concept of point cloud is not new and has been around since 1960 since the invention of the laser (MAIMAN, 1960). Each data point in a PCD consists of spatial coordinates X,Y and Z and might include color information in R,G and B values, as well as normal values n$_x$, n$_y$, n$_z$ of a particular object. They are commonly used in computer vision, computer graphics, remote sensing, and Geographic Information System (GIS). Normally, point clouds are generated through 3D Scanning, LiDAR or photogrammetry.

2.1.1 Point Cloud Registration

Registration of point clouds is a fundamental task in 3D computer vision and photogrammetry (PAN et al., 2018). The objective is to find correspondences between two distinct point sets and determine the transformation that aligns one with the other, (MYRONENKO & SONG, 2010). Several different iterative techniques are present for performing registration.

1. **Iterative Closest Point**
   Initially introduced in the paper by (BESL & MCKAY, 1992), the objective is to iteratively find a transformation matrix that maps a given point cloud to a reference point(can be a surface as well). It achieves this by minimizing the square errors among the corresponding points. Given two point sets, ICP finds the best-fit transformation between that maps point set in A to point set in B. As the name suggests, the algorithm proceeds in an iterative manner. It applies the transformation to A, calculates the error between the transformed set A and B, and stops if the error is less than a tolerance value. Set A is updated every iteration.

   Algorithm 2.1, presents the basic version of ICP algorithm, as described by (Q. ZHOU et al., 2022). Many other famous variations of ICP like point-to-plane and point-to-point exist, (POMERLEAU et al., 2015).

2. **Random Sample Consensus**
   RANSAC is a widely used algorithm for fitting models with outliers. Introduced by

Algorithm 2.1: ICP Algorithm

```
Given reference P and data point Q,
    Repeat until the objective < threshold
        For points $\in$ Q find corresponding nearing points in P
           Objective Function $\leftarrow$ minimize the euclidean distance
              between the matching pairs
           find R,T that minimizes the Objective Function
           Q_new  $\leftarrow$ R (Q) + T
```

(Fischler & Bolles, 1987) as a robust method for fitting straight lines to 2D images with outliers. The idea is to select a subset of data points, fit a model to this subset and evaluate the model on the remaining data points. Data points that are consistent with the model are considered inliers, while points that are not are considered outliers. It is also an iterative process.

While ICP is a local registration method because tit relies on rough alignment as initialization, RANSAC performs global registration which does not require any initial transformation.

The only pre-requisite for using RANSAC is that number of data points should outnumber the minimum number required for determining the model parameters. Algorithm for calculating RANSAC is shown in 2.2, (Derpanis, 2010).

Algorithm 2.2: RANSAC Algorithm

```
Randomly select minimum points required for model parameter
   determination.
Solve for the model parameters
Identify points fitting the model with tolerance $\tau_1$
If inliner ratio > threshold $\tau_2$, estimate parameters using identified
   inliers and stop
Repeat steps 1-4 N times if inliner ratio < $\tau_2$
```

2.1.2 Deep Learning in Point Clouds

There exist numerous architectures that deal with point clouds. Various techniques exist to deal with the unstructured nature of point clouds. Voxel-based, (Maturana & Scherer, 2015), set-based, (Qi et al., 2017), graph-based, (Y. Wang et al., 2018), multi-view-based,(W. Wang et al., 2022) are some of the methods that address the unique challenges posed by point cloud data. These network can be used for downstream tasks of semantic segmentation or instance segmentation or feature extraction.

In this section, the focus is on set-based approach, particularly PointNet, (Qi et al., 2017). Main reason being, its feature extraction component is used as a module, which is explained in chapter 4.

2.1.2.1 PointNet

PointNet directly takes point clouds as input, without the need for intermediate representations such as voxels or meshes. The architecture consists of three main components: a shared multi-layer perceptron (MLP) network, a max pooling layer and a fully connected layer. The share MLP network is applied to each point independently, and produces a point wise feature vector for each point independently. PointNet is able to capture local features of each point while preserving its permutation invariance. This allows PointNet to work well with unstructured point clouds that do not have a fixed topology.

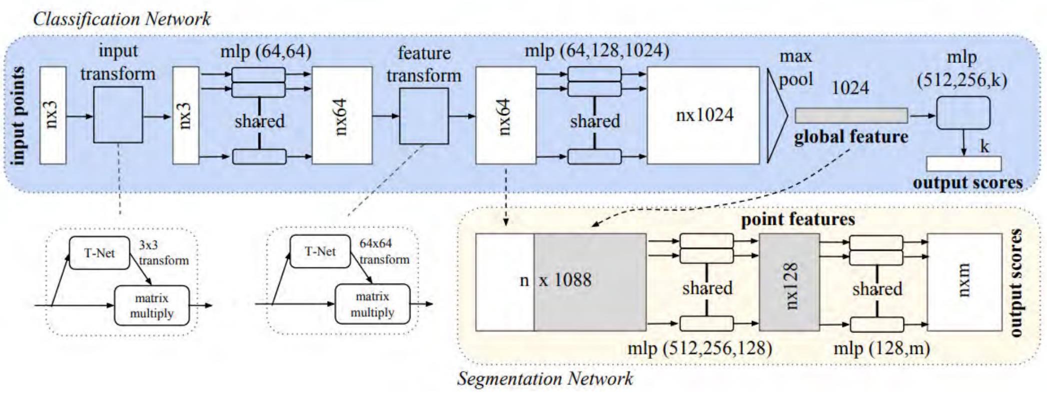


Figure 2.1: PointNet Architecture, (Qi et al., 2017)

2.2 Building Information Modeling

Building Information Modeling is a process that encompasses creating and managing a digital model representing the physical and functional characteristics of a structure or a location (*Building Information Modeling: Technology Foundations and Industry Practice*, 2018). It can also be called the digital twin of the physical structure that it represents (Collins et al., 2021). It creates an accurate 2D or 3D geometry and adds semantic information about the built asset in question (Braun et al., 2018). This includes various relationship types about the buildings, properties of components, and other data relevant to construction, operation, and maintenance. Also enables real-time collaboration and interoperability.

BIM is designed in a way to cover the entire life cycle of a building, from its inception to its construction and to its operation. It also incorporates parametric modeling, that are not automatically turned on in traditional Computer Aided Design (CAD) software, such that changes in one element is reflected across the entire model (Eastman, 2009).

2.3 Graph Data Structure

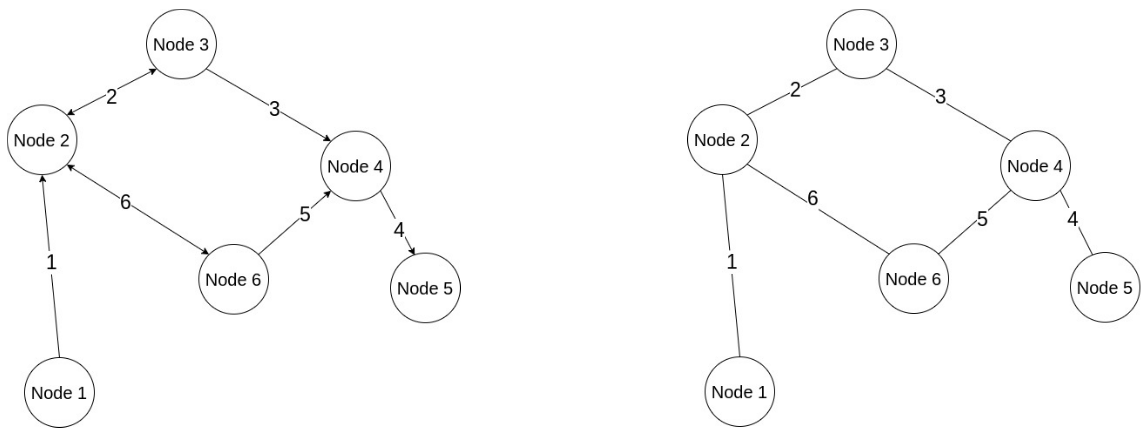
In the most basic sense, it is a mathematical structure consisting of a set of vertices (V), colloquially called nodes, and sets of edges (E). It is a mathematical abstraction that helps model objects and relationships between them (CHERNOSKUTOV, 2021). Generally, a graph is denoted as G = (V, E).

Graph structures can be further divided into two subcategories.

1. Directed Graphs
   A directed graph or digraph, (BENDER & WILLIAMSON, 2010), is a pair G = (V,E) comprising:
   - **V:** Set of vertices.
   - **E:** Set of directed edges, are ordered pairs of distinct vertices. Each edge connects a source vertex or a target vertex. If there is a directed edge from A to B, then it is denoted as (A, B) and not (B, A) unless the relation is bidirectional. Here A and B are

$$E = \{(x, y) | (x, y) \in \mathbf{V}^2\}$$

2. Undirected Graphs
   Graph where the edges do not have a direction. It indicates a two way relationship, that the relationship between the two edges is mutual. In such cases a edge can be traversed in both direction. Here an edge **E**: (A,B) is equivalent to **E**: (B,A).



(a) Directed Graph (b) Undirected Graph

Figure 2.2: Types of Graph Data Structures

2.3.1 Knowledge Graph (KG)

A knowledge graph is a data structure that includes information about the entities and the relationships linking them, (HOGAN et al., 2021). Even though they have been historically present, they quickly gained momentum following the introduction of Google Knowledge Graph (SINGHAL, 2012) in 2012.

Knowledge graphs are graph-structured knowledge bases. Usually, a digraph, is assumed. It represents semantics by describing entities and their relationships with each other. The main components of KGs are: nodes, edges, and labels. Labels define the relationship between the nodes in an edge.

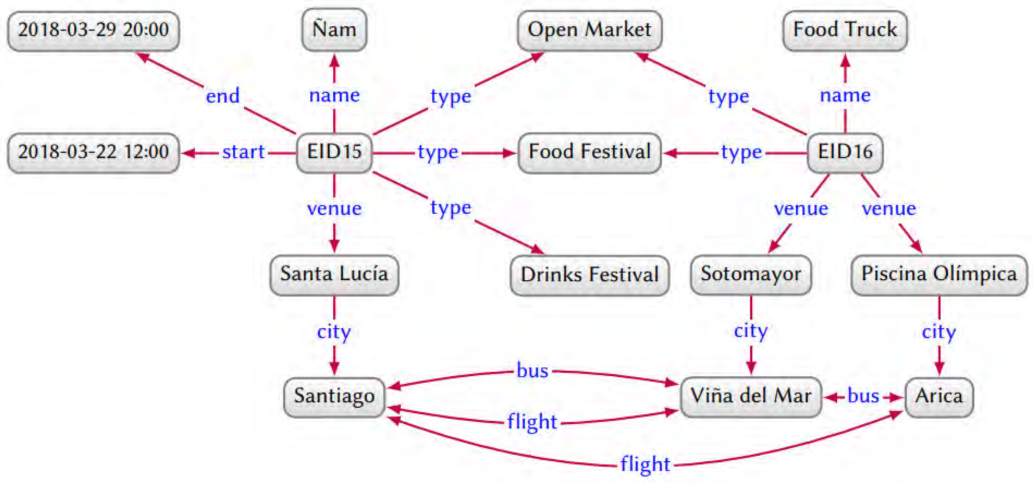


Figure 2.3: Example of a Knowledge Graph, (HOGAN et al., 2021)

Currently, KGs are extensively used with Machine Learning, Deep Learning, to reason over the vast amount of stored data. Due to the vastness of the field and contexts, same entities or similar entities appear in multiple knowledge graphs. The task of aligning two different KGs based on the same or similar entity present in both is called knowledge graph entity alignment. It is a huge and active area of research. (BERRENDORF et al., 2020)

Mostly, some variation of Graph Neural Network (GNN)s are employed to extract the information of KGs in terms of embeddings and features. Due to the current advancement in Large Language Models (LLMs), KGs have become an important basis of many Artifical Intelligence (AI) and NLP tasks, (Z. WANG et al., 2018).

2.3.1.1 KG in 3D Data

Works by WALD et al., 2020, and ARMENI et al., 2019, show that knowledge graphs can be used to represent 3D data (built environment) as they provide a rich and structured way to represent the data. WALD et al., 2020, uses a graph prediction network that can semantically segment the objects and define a relation between the elements of the built environment. LANDRIEU and SIMONOVSKY, 2018a, uses super point graph formulation for the downstream semantic segmentation task. This is a huge area of research where knowledge graphs are used to understand 3D data.

2.3.2 Entity Alignment (EA)

As discussed in the previous section 2.3.1 entity alignment refers of aligning two different KG containing contain information from the same modality.

2.3.2.1 Multi-Modal Alignment

Multi-Modal Knowledge Graph Alignment, (GUO et al., 2021) refers to aligning multiple knowledge graphs of different modalities (eg. languages, texts, images). A certain degree of overlap must be present in the target and reference KGs. Research by Entity Visual Alignment (LIU et al., 2021) uses visual and other knowledge to achieve multi-modal KG alignment. There are various approaches to learning information from different modalities. Work by (CHEN et al., 2020), (LIN et al., 2022) use common embedding space for all the modalities. They employ different individual encoders depending upon the type of information to obtain modality-specific representation.

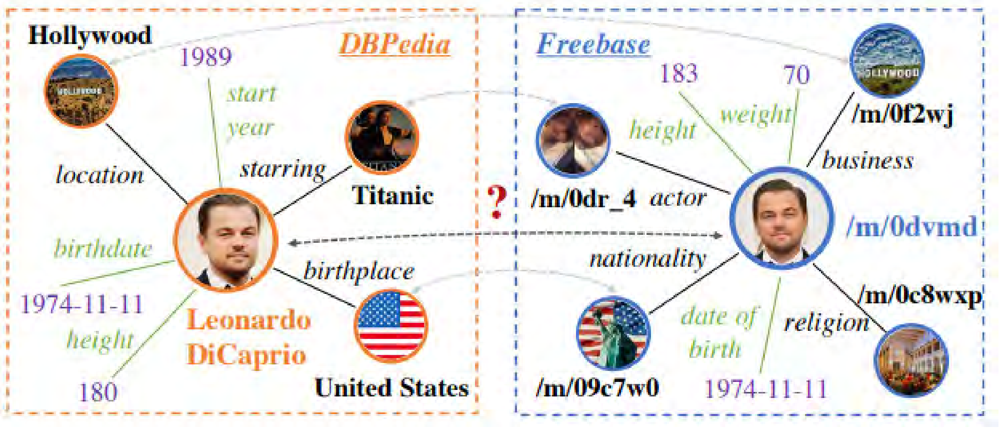


Figure 2.4: Example of a multi-modal entity alignment, (ZHU et al., 2023)

2.3.2.2 Multi-Modal Alignment in 3D Data

Even though most of the KG alignment work has been particularly in the domain of NLP, research work by (LANDRIEU & SIMONOVSKY, 2018b), (SARKAR et al., 2023) has shown that KGs also facilitates the handling of 3D data.

Information-rich 3D data can be abstracted using KG to deal with the high resolution and large data volume. The underlying geometry represented in the point cloud and BIM can be encoded as a feature in the graph.

Work by (SARKAR et al., 2023) also shows how one can leverage the individual modalities, with contrastive learning introduced by (LIN et al., 2022), in 3D alignment. This work is the current state of the art in aligning 3D point cloud data represented in graphs, which is one of the main backbones of this research work. Chapter 3 talks about the inner workings of this research work.

2.4 Deep Learning Architectures for Graph Data

Different network architectures exist to work with structured graph data.

2.4.1 Graph Neural Network (GNN)

GNNs were introduced by (J. Zhou et al., 2020), as a method to solve problems in graph domains. They can be directly applied to graphs. Generally GNN are used in link-prediction, classification, NLP tasks.

Given a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges, the working of GNN can be divided into three steps.

1. **Neural Message Passing:**
   Each node i in the graph will receive information about their state. Neighbour of a node i is defined as $N_i = \{j : e_{ij} \in E\}$.

2. **Aggregation:** The received information is aggregated together. Message parsing and aggregation can be defined mathematically as,

$$\hat{\mathbf{m}}_i = \square_{\mathbf{j} \in \mathbf{N_i}}(\mathbf{h^j_{t-1}}, \forall \mathbf{j} \in \mathbf{N_i}) \tag{2.1}$$

   $\square$ is an aggregation function like mean, max, sum, a neural network invariant to permutability, and h^j_{t-1} is the message incoming from node j.

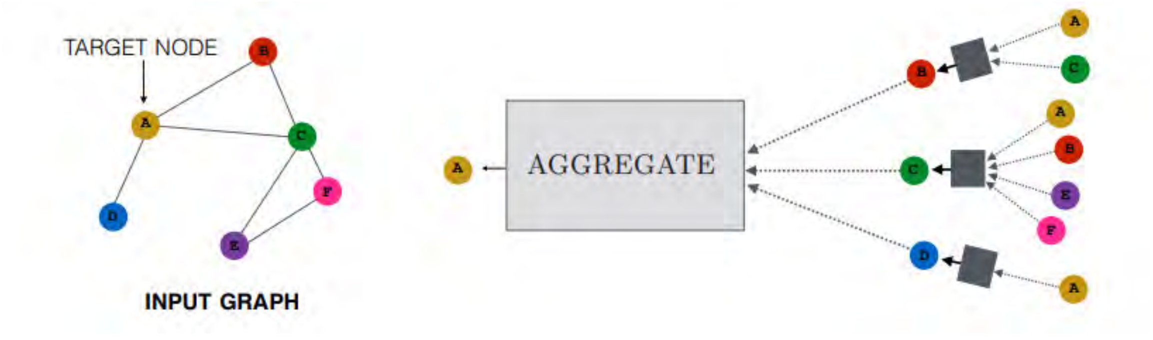


Figure 2.5: Overview of message aggregation from node's local neighborhood, (Hamilton, 2020)

3. **Update:**
   After the information has been aggregated, the current state of node i is updated.

$$\mathbf{h^i_t} = \sigma(\mathbf{h^i_{t-1}}, \hat{\mathbf{m}}_\mathbf{i}) \tag{2.2}$$

   where σ is a differentiable update function.

At each time step t, for a given node i, the aggregation function gathers information from its neighborhoods, $\mathbf{N_i}$ and generates a message $\hat{\mathbf{m}}_\mathbf{i}$. The update function then combines this message with the embedding from the previous time step $\mathbf{h^i_{t-1}}$ to generate the updated embeddings $\mathbf{h^i_t}$. This happens simultaneously for all the nodes at the same time. After the first time step, each node will have information about the features of its immediate neighbors. After the second time step, each node will contain information from its 2-hop neighborhood, (Hamilton, 2020).

Three phases described above form the framework of a basic GNN. A trainable GNN can be defined as,

$$h_t^i = \sigma\left(W_{t-1}^{self}\ h_{t-1}^i + W_{t-1}^{neigh}\ \square_{j \in N_i}\ h_{t-1}^j \right) \tag{2.3}$$

where $\mathbf{W_{t-1}^{self}}, \mathbf{W_{t-1}^{neigh}} \in \mathbb{R}^{d_t \times d_{t-1}}$ are trainable parameters. Equation 2.3 represents the message-passing and update at the node level, the graph-level passing and update can be defined as,

$$H_t = \sigma\left(AH_{t-1}W_t^{neight}\ +\ H_{t-1}W_t^{self} \right) \tag{2.4}$$

where $H_t \in \mathbb{R}^{|V| \times d}$, denotes node representation at layer/time-step t, A is the graph adjacency matrix.

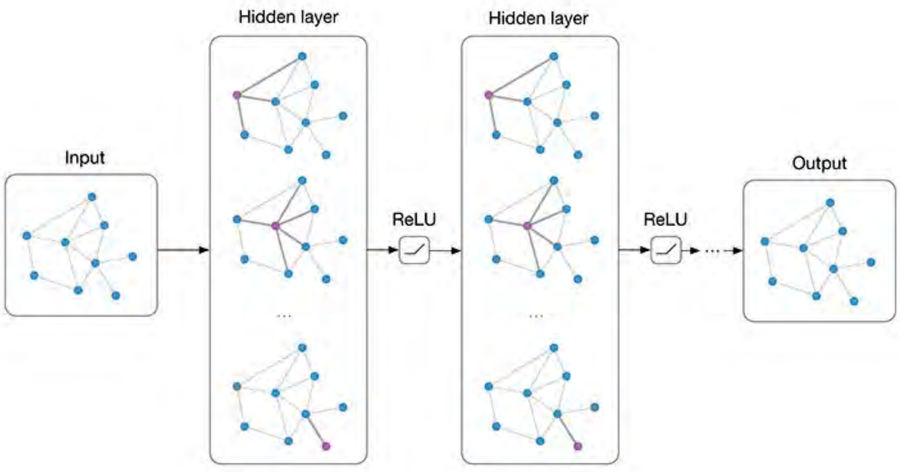


Figure 2.6: Basic Layout of Graph Neural Network, (T. KIPF, 2016)

2.4.2 Graph Convolution Network (GCN)

GCN is one of the most popular variant of GNN. It utilizes symmetric normalization for aggregation and incorporates a self-loop update approach, (HAMILTON, 2020).

Given a graph $G = (V, E)$ with N nodes $n_i \in V$, edges $(e_i, e_j) \in E$, adjacency matrix $A \in \mathbb{R}^{N \times N}$, and a degree matrix $D_{ii} = \sum_j A_{ij}$, the layer wise propagation rule in Graph Convolution Network, (T. N. KIPF & WELLING, 2017) can be obtained as,

$$H^{l+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^lW^l) \tag{2.5}$$

where $\hat{A} = A + I_N$ is adjacency matrix of undirected graph G with added self connections, I_N is the identity matrix,

$\hat{D}_{ii} = \sum_j \hat{A}_{ij}$,
W^l represents layer specific trainable weight matrix,
$\sigma(\cdot)$ denotes an activation function and,
$H^l \in \mathbb{R}^{N \times N}$ is activation in the l^{th} layer.

2.4.3 Graph Attention Network (GAT)

(VASWANI et al., 2017) introduced attention networks which showed that self-attention can improve a method and is sufficient for constructing a model obtaining maximum performance on machine learning tasks. The benefit of the attention mechanism is that it allows for variable-sized input. And when it is used to compute a representation of a single sequence, it is termed self-attention.

Inspired by this work, (VELIČKOVIĆ et al., 2018) introduced attention-based architecture to perform node classification based on graph-structured data. It computes hidden representations of each node in the graph by attending using a self-attention strategy over its neighbors.

Given a graph $G = (V, E)$ with N number of nodes, with each node having a feature of $h_i, i \in V$, which can be written as,

$$\mathbf{h} = \{h_1, h_2, ..., h_N\}, h_i \in \mathbb{R}^F \tag{2.6}$$

where F is the number of features in each node. The attention layer acts on these inputs to produce new feature sets,

$$\mathbf{h}' = \{h'_1, h'_2, ..., h'_N\}, h'_i \in \mathbb{R}^{F'} \tag{2.7}$$

To transform the input features into high-level features, a shared linear transformation is applied to every node, represented by $\mathbf{W} \in \mathbb{R}^{F' \times F}$. Then self-attention mechanism is applied on the nodes, $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ to compute attention coefficients e_{ij}. a is just a single-layer feed-forward neural network.

$$e_{ij} = a(\mathbf{W}h_i, \mathbf{W}h_j) \tag{2.8}$$

Attention coefficients indicate the importance of node $j^{'}$s features to node i. e_{ij} only for nodes $j \in N_i$, where N_i are the neighbouring nodes of i, is computed. The coefficients are then normalized across all neighbors using the softmax function.

$$\alpha_{ij} = softmax_j(\sigma(e_{ij})) = \frac{exp(\sigma(e_{ij}))}{\sum_{k \in N_i} exp(\sigma(e_{ik}))} \tag{2.9}$$

In the paper, LeakyRELU was used as a non-linearity, so 2.10 can be written in its expanded form as,

$$\alpha_{ij} = \frac{exp(LeakyRELU(a^T[\mathbf{W}h_i||\mathbf{W}h_j]))}{\sum_{k \in N_i} exp(LeakyRELU(a^T[\mathbf{W}h_i||\mathbf{W}h_k]))} \tag{2.10}$$

where || denotes concatenation.

The new set of feature vectors for each node can be computed as,

$$h_i' = \sigma\left(\sum_{j \in N_i} \alpha_{ij}\mathbf{W}h_j\right) \tag{2.11}$$

On running experiments, this research found that multi-head attention to be beneficial. K different independent attention mechanisms are executed on 2.13 and the features are concatenated to get a final representation.

$$h_i' = \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in N_i} \alpha_{ij}^k \mathbf{W}^k h_j\right) \tag{2.12}$$

where || represents concatenation, α_{ij}^k represents normalized coefficients of k-th attention mechanism and $\mathbf{W}^k$ represents transformation matrix. Instead of F' features, multi-headed attention will now contain KF' features.

If multi-headed attention is applied to the final layer, instead of concatenation averaging is employed.

$$h_i' = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in N_i} \alpha_{ij}^k \mathbf{W}^k h_j\right) \tag{2.13}$$

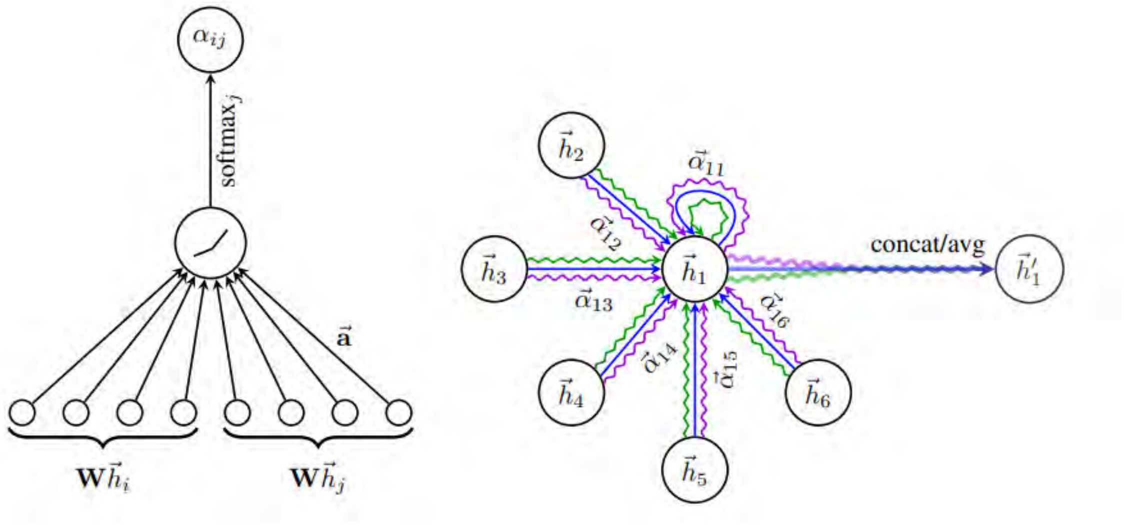


Figure 2.7: **Left**: Attention Mechanics employed by GAT, **Right**: Illustration of multi-headed attention (K=3) by node 1 on its neighborhood. (VASWANI et al., 2017)

Chapter 3

Related Work

This chapter discusses research work that forms the basis of this research work. Studies related to registration, contrastive learning, PCD-BIM alignment, and PCD-PCD alignment are discussed in detail. Section 3.1 dives deeper into the current learning based state-of-the-art registration method, section 3.2 outlines the methodology of the PCD-BIM alignment. Since *contrastive learning* is a key part of this thesis, section 3.3 dives deeper into how contrastive learning can be used in the case of multi-modal encoders, although the study being in the domain of NLP. Finally, in section 3.4 the workings of the current state-of-the-art architecture in PCD-PCD (scene) alignment is discussed.

3.1 GeoTransformer

This paper by Qin et al. (2022) deals registration of point clouds. They focus primarily on finding accurate correspondences using learning-based approach. Given two sets of point clouds $\mathcal{P} = \{p_i \in \mathbb{R}^3 | i = 1, ..., N\}$ and $\mathcal{Q} = \{q_i \in \mathbb{R}^3 | i = 1, ..., M\}$, the process involves estimating a rigid transformation **T** = {**R, t**}. This transformation aligns the two point sets through a 3D rotation $R \in SpecialOrthogonalgroup(\mathrm{SO})(3)$ and a translation **t** $\in \mathbb{R}^3$. The objective function is defined as,

$$\min_{R,t} \sum_{(p^*_{x_i}, q^*_{x_i}) \in C^*} \|R \cdot p^*_{x_i} + t - q^*_{y_i}\|_2^2 \tag{3.1}$$

where C^* denotes the set of ground truth correspondences between $\mathcal{P}$ and $\mathcal{Q}$. As C^* is unknown, correspondences between the point sets is first established and an alignment transformation is then calculated.

Correspondences are found hierarchically and go from coarse to fine. The method of registration can be sub-divided into four parts.

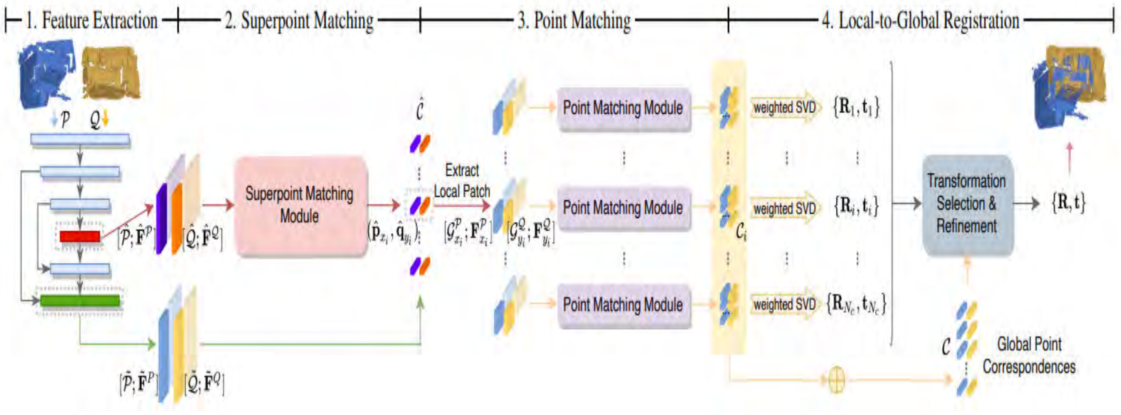

1. Feature Extraction
2. Superpoint Matching
3. Point Matching
4. Local-to-Global Registration
Superpoint Matching Module
Extract Local Patch
Point Matching Module
weighted SVD
Transformation Selection & Refinement
Global Point Correspondences

Figure 3.1: Backbone of GeoTransformer, (Qin et al., 2022)

3.1.1 Feature Extraction

Kernel Point Convolution (KPConv)-FPN is used to extract features from the given point sets. The point sets get down-sampled in the process. The down-sampled points, also called superpoints, correspond to the coarsest resolution, denoted by $\hat{\mathcal{P}}$ and $\hat{\mathcal{Q}}$, and the learnt features denoted by $\hat{F}^{\mathcal{P}} \in \mathbb{R}^{|\hat{\mathcal{P}}|\times\hat{d}}$ and $\hat{F}^{\mathcal{Q}} \in \mathbb{R}^{|\hat{\mathcal{Q}}|\times\hat{d}}$. The dense correspondences are computed at half of the original resolution, denoted by $\widetilde{\mathcal{P}}$ and $\widetilde{\mathcal{Q}}$.

3.1.2 Superpoint Matching

Geometric Transformer encodes global contextual information and explicitly captures geometric structures within each point cloud and ensures geometric consistency across point clouds. GeoTransformer consists of a self-attention module to learn intra-point cloud features and a feature-based attention module to model inter-point cloud consistency. The two different modules are interleaved for N times to extract hybrid features $\hat{\mathbf{H}}^{\mathcal{P}}$ and $\hat{\mathbf{H}}^{\mathcal{Q}}$.

The hybrid features exhibit invariance to transformation and robustness for correspondence. The features are first normalized onto a unit hypersphere and then Gaussian correlation matrix is computed as $\mathbf{S} \in \mathbb{R}^{|\hat{\mathcal{P}}|\times|\hat{\mathcal{Q}}|}$ with $s_{i,j} = exp(-||\hat{h}_i^{\mathcal{P}} - \hat{h}_j^{\mathcal{Q}}||_2^2)$. Since some patches of point clouds are less discriminative geometrically and can have multiple patches in another point set, dual-normalization on $\mathbf{S}$ is performed to suppress ambiguous matches.

$$\overline{s}_{i,j} = \frac{s_{i,j}}{\sum_{k=1}^{|\hat{\mathcal{Q}}|} s_{i,k}} \cdot \frac{s_{i,j}}{\sum_{k=1}^{|\hat{\mathcal{P}}|} s_{k,j}} \tag{3.2}$$

Finally, the largest N_c entries in $\overline{\mathbf{S}}$ are chosen as superpoint correspondences.

3.1.2.1 Point Matching Module

The point correspondences are from superpoint correspondences. For each superpoint correspondence $\hat{C}_i = (\hat{p_{xi}}, \hat{q_{yi}})$, a transport layer, (Sarlin et al., 2020), is used to extract local dense point correspondences. Computed point correspondences from each superpoint are collected to form a final global dense point correspondences: $\mathcal{C} = \bigcup_{i=1}^{N_c} \mathcal{C}_i$.

3.1.3 Point Cloud Registration

They developed a registration scheme that operates in a local-to-global manner. It comprises a local phase that generates transformation candidates and a global phase for selecting transformation. In the local phase, a transformation is solved for each superpoint match using its local correspondence.

$$R_i, t_i = \min_{R,t} \sum_{(\widetilde{p}_{x_j}, \widetilde{q}_{y_j}) \in \mathcal{C}_\rangle} w_j^i \| R \cdot \widetilde{p}_{x_j} + t - \widetilde{q}_{y_j} \|_2^2 \tag{3.3}$$

This is solved using weighted Singular Value Decomposition (SVD) (Besl & McKay, 1992). The transformation obtained in this phase is very close to accurate. In the global phase, a transformation that admits the highest number of inlier matches over the entire global point correspondences is selected.

$$R, t = \max_{R,t} \sum_{(\widetilde{p}_{x_j}, \widetilde{q}_{y_j}) \in \mathcal{C}_\rangle} [\![\| R_i \cdot \widetilde{p}_{x_j} + t - \widetilde{q}_{y_j} \|_2^2 < \tau_a]\!] \tag{3.4}$$

3.1.4 Loss Functions

Loss function $\mathcal{L} = \mathcal{L}_{oc} + \mathcal{L}_p$ is composed of an overlap aware circle loss $\mathcal{L}_{oc}$ for superpoint matching and a point matching loss $\mathcal{L}_p$ for point matching.

The modules in this section 3.1 only provide a basic overview. To get a detailed overview of GeoTransformer, regarding all the methods, and loss functions used one should refer to (Qin et al., 2022).

3.2 Graph-based linking of Point Cloud and BIM

This study by Collins et al. (2023) focuses on integrating semantic elements in PCD to BIM. Geometric features and topological relationships in the building are used.

Two different data streams are used and formulated in a graph as $G = (V, W)$, where V represents the nodes and W represents the distance-weighed adjacency of the nodes. A node is one building element. A set of geometric features F, including high-level geometric information about the element's shape, is attached to the node.

The feature includes a semantic label, two principal components obtained using Principal Component Analysis (PCA), and the extent of the shapes in the direction of principal components.

$$F = \{semantic\_label, PC_1, PC_2, extent_1, extent_2\} \tag{3.5}$$

The methodology of this research can be subdivided into three categories.

3.2.1 BIM to G_{DT}

Node and its features are extracted using a model parsing library, IfcOpenShell. Building elements and their features are then calculated by sampling 1000 points on its surface.

Spatial-topological Query Lanuguage for Building Information Modeling (QL4BIM), developed by DAUM and BORRMANN (2014), extracts adjacent building elements. The element connectivities are then weighted by the distance between the centroids between the elements which form a matrix W.

3.2.2 Point Cloud to G_{PCD}

COLLINS et al. (2022) in their work in finding geometric and topological similarities in building elements, suggest G_{PCD} formulation method using superpoint graph approach formulated by LANDRIEU and SIMONOVSKY (2018a). A node/superpoint is one semantic instance in a given point cloud. Feature matrix F, for point clouds, is calculated similarly as described in 3.2.1.

Edges are generated using the 3D Delaunay Triangulation approach. Edges larger than 0.5m are discarded, and the distance of the remaining edges forms the matrix W.

3.2.3 Graph Matching

Geometric similarities between all nodes of the same semantic class of G_{PCD} and G_{DT} are calculated using cosine similarity. This results in a list that contains ranked candidates with a matching score M. The matching score ranges from the highest matching geometric score to the least matching geometric score.

The highest-ranked candidate pairs are then taken and Maximum Common Subgraph (MCS) is computed within 2-hop induced subgraphs. The next node pair is evaluated if multiple structurally equivalent graphs are found. If no such equivalence is found, rewards are calculated and added to the matching score, M. This process is iterative, and the candidate pairs are re-ranked after each iteration. The calculation continues until no further re-ranking of the candidate pairs occurs.

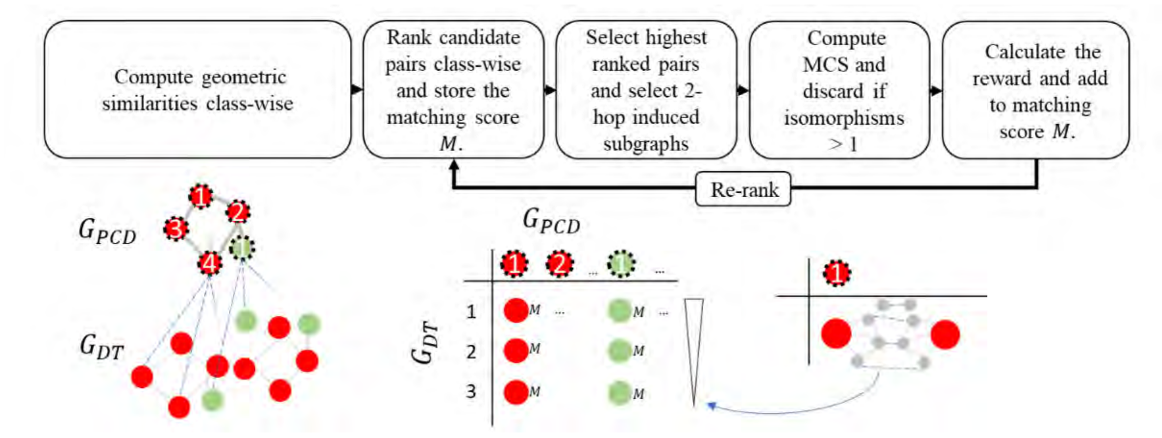

Compute geometric similarities class-wise
Rank candidate pairs class-wise and store the matching score M.
Select highest ranked pairs and select 2-hop induced subgraphs
Compute MCS and discard if isomorphisms > 1
Calculate the reward and add to matching score M.
Re-rank
G_{PCD}
G_{DT}

Figure 3.2: Graph-based Linking Methodology, (COLLINS et al., 2023)

3.3 Contrastive Learning For Entity Alignment

Contrastive learning is a semi-supervised machine learning technique where the model learns to pull similar entities closer and dissimilar entities are pushed away in the embedding space, (HADSELL et al., 2006).

Work by LIN et al. (2022) is in the domain of Natural Language Processing, where they use multi-modality for alignment. This paper proposes Multi-modal Contrastive Learning based Entity Alignment (MCLEA), where information from uni-modal is integrated into joint representations for entity alignment. Multiple individual encoders are utilized to get modality-specific embeddings for each entity. Their modal encodes information related to neighborhood structures, relations, attributes, surface forms, and images.

They also introduced contrastive learning into Entity alignment, which serves as a foundation for this research work. Two different losses were proposed, Intra-modal Contrastive Loss (ICL) and Inter-modal Alignment Loss (IAL).

3.3.1 Intra-modal Contrastive Loss

The objective of ICL is to differentiate the embeddings of equivalent entities from the other entities within each modality. It ensures that the representations of similar entities are closer in the embedding space than to representations of non-equivalent entities within the same modality.

In MCLEA, ICL enables the model to differentiate between the embeddings of identical entities belonging to different Knowledge graphs within a given modality.

If S is regarded as a positive sample and any other non-aligned pair is regarded as a negative sample, then for *i*-th entity $e_1^i \in E_1$ of minibatch *B*, the positive set is defined as,

$$P_i = \{e_2^i \mid e_2^i \in E_2\}$$

where (e_1^i, e_2^i) is an aligned pair.

The negative set is comprised of inner graph unaligned pairs from source KG G_1 and cross-graph unaligned pairs from target KG G_2, defined as,

$$N_1^i = \{e_1^j \mid \forall e_1^j \in E_1, j \neq i\} \quad \text{and} \quad N_2^i = \{e_2^j \mid \forall e_2^j \in E_2, j \neq i\}$$

N_1^i, N_2^i both come from minibatch *B* and are designed to constrain the joint embedding space. The alignment probability distribution $q_m(e_1^i, e_2^i)$ for modality *m* for each positive pair (e_1^i, e_2^i) is defined as,

$$q_m(e_1^i, e_2^i) = \frac{\delta_m(e_1^i, e_2^i)}{\delta_m(e_1^i, e_2^i) + \sum_{e_1^j \in N_1^i} \delta_m(e_1^i, e_1^j) + \sum_{e_2^j \in N_2^i} \delta_m(e_1^i, e_2^j)} \tag{3.6}$$

where $\delta_m(u, v) = \exp\left(\frac{f_m(u)^T f_m(v)}{\tau_1}\right)$ is the encoding of the modality *m* and τ_1 is a temperature parameter.

Since the distribution in Eq.3.6 is directional and asymmetric, ICL is defined as,

$$L_{\mathsf{ICL}}^m = -\mathbb{E}_{i \in B} \log\left(\frac{1}{2}(q_m(e_1^i, e_2^i) + q_m(e_2^i, e_1^i))\right) \tag{3.7}$$

ICL is applied separately on each modality as well as also on the joint embedding space.

3.3.2 Inter-modal Alignment Loss

Embeddings of different modalities are trained separately on ICL. And the complex interaction between modalities can't be modeled just using the fusion module. To solve this problem, MCLEA introduced inter-modal alignment loss, which aims to minimize the gap between the output distribution across different modalities. It helps to capture the inter-modal interaction and obtain more meaningful representations.

Bi-directional Kullback-Leibler (KL) divergence is minimized over the output distribution between joint embedding and uni-modal embedding. This facilitates the transfer of knowledge from joint embedding to uni-modal embedding so that the uni-modal embeddings could better utilize the information from others.

IAL is defined as,

$$\begin{aligned} L_{\mathsf{IAL}}^m = \mathbb{E}_{i \in B} \frac{1}{2} \Big[& \mathsf{KL}\big(q_o^{'}(e_1^i, e_2^i) || q_m^{'}(e_1^i, e_2^i)\big) \\ & + \mathsf{KL}\big(q_0^{'}(e_2^i, e_1^i) || q_m^{'}(e_2^i, e_1^i)\big)\Big] \end{aligned} \tag{3.8}$$

where $q_o^{'}(e_1^i, e_2^i)$, $q_0^{'}(e_2^i, e_1^i)$ and $q_m^{'}(e_1^i, e_2^i)$, $q_m^{'}(e_2^i, e_1^i)$ represents output predictions with two directions of joint embedding and the uni-modal embedding of modality m.

They only back-propagate through $q_m^{'}(e_1^i, e_2^i)$, $q_m^{'}(e_2^i, e_1^i)$ in Eq.3.8 as knowledge distillation (HINTON et al., 2015).

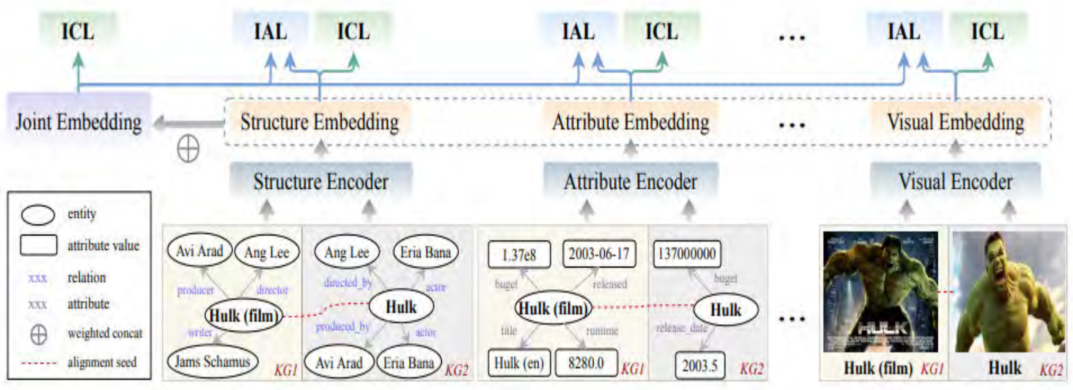


Figure 3.3: Architrecure of MCLEA that learns through IAL and ICL, (LIN et al., 2022)

3.3.3 Objective Function

MCLEA in their work defined the optimization objective as,

$$L = L_{\text{ICL}}^{o} + \sum_{m \in M} \alpha_m L_{\text{ICL}}^{m} + \sum_{m \in M} \beta_m L_{\text{IAL}}^{m} \tag{3.9}$$

where L_{ICL}^{o} is ICL operates on joínt embedding, α_m and β_m are hyperparameters. This Loss L is further modified into,

$$L = L_{\text{ICL}}^{o} + \sum_{m \in M} \left(\frac{1}{\alpha_m^2} L_{\text{ICL}}^{m} + \frac{1}{\beta_m^2} L_{\text{IAL}}^{m} + \log \alpha_m + \log \beta_m\right) \tag{3.10}$$

where α_m and β_m are learned during training.

3.4 Scene Graph Aligner (SGAligner)

SARKAR et al. (2023) work concentrates on aligning 3D scene graphs. Leveraging the concepts of multi-modality and contrastive learning inspired from LIN et al. (2022) and the work by CHEN et al. (2020). It employs embeddings learned in a joint embedding space to determine alignment and estimate transformation between pairs of 3D scenes. This thesis mostly follows the framework set up by this referenced research.

This method developed by SGAligner is the first method for aligning pairs of 3D scene graphs. Information contained in 3D scene graphs: semantic entities, their geometry, and relationships between these entities are encoded independently to learn a joint embedding space that reasons whether or not the two given nodes are similar or not. Cosine similarity is used to get a list of matched entities and alignment is performed using the one with the highest similarity.

The scene alignment method can further be employed for 3D point cloud registration. The alignment result is used for coarse initialization for registration, which is further refined by computing 3D correspondences, (QIN et al., 2022). Then, the rigid point cloud transformation is estimated using the computed correspondences.

3.4.1 3D scene graphs

A 3D scene graph provides a structured approach to representing detailed descriptions of real-world scenes, (KIM et al., 2020). The structure of scene graphs employed in this work aligns with the 3D scene graph framework introduced by WALD et al. (2020). This thesis work also leveraging that information, uses the same graph structure.

3.4.2 3D Scene Graph Alignment

3D scene graph $\mathcal{G}$ is defined as a pair of sets of $(\mathcal{N}, \mathcal{R})$, where $\mathcal{N}$ represents node and $\mathcal{R}$ represents edges. A node represents a 3D object $\mathcal{O}$ in a given scene. Point clouds $\mathcal{P}$ and the edges $\mathcal{R}$ define the semantic relationship between the nodes. The network architecture is derived from LIN et al. (2022), which was modified from the language domain to the task of 3D scene alignment.

Four different modules for encoding information are used, as explained in 3.4.3. Object embedding that encodes point cloud information $\mathcal{P}$, structure embedding that encodes $\mathcal{R}$ in the form of a structured graph, and two meta modalities that encode $\mathcal{A}$ and $\mathcal{R}$. In the end, these individual embeddings are combined in a weighted manner. A joint optimization is performed using knowledge distillation across all embeddings, as explained in section 3.3.

3.4.3 Uni and Multi-Modal Embeddings

1. **Object Embedding:** To capture the geometric information about objects, PointNet (QI et al., 2017) is used. Each individual point $\mathcal{P}_i$ of $\mathcal{O}_i$ serves as an input to the model and the visual features $\phi_i^{\mathcal{R}}$ for each node are then extracted.

2. **Structure Embedding:** This module encodes objects' layout in a given scene $\int$. This information is represented in the form of a structured graph. Node features include the relative translation between object instances with the highest number of relationships and any other object instance in the scene. GAT (VELIČKOVIĆ et al., 2018) is used to model this information. They limit the weight matrix to a diagonal

matrix, to minimize computations and improve the model's scalability, as suggested by (LIN et al., 2022).

3. **Meta Embeddings:** Attributes and relationships per object $\mathcal{O}_i$ are modeled as two separate embeddings using feed-forward network (BEBIS & GEORGIOPOULOS, 1994). Relationships of $\mathcal{O}_i$ with other objects are regarded as bag-of-words feature vectors and used to obtain relational embedding $\phi_i^{\mathcal{R}}$. They use the same approach for getting $\phi_i^{\mathcal{A}}$.

4. **Joint Embedding:** The individual embeddings are concatenated to a single compact representation $\hat{\phi}_i$ for each object $\mathcal{O}_i$ as,

$$\hat{\phi}_i = \bigoplus_{k \in K} \left[\frac{exp(w_k)}{\sum_{j \in K} exp(w_j)} h_i^m \right] \tag{3.11}$$

where $\bigoplus$ denotes concatenation, $\mathcal{K} = \{\mathcal{P}, \mathcal{S}, \mathcal{R}, \mathcal{A}\}$ represents each individual modality. Variable w_m represents a trainable attention weight for each modality. L2 normalization is applied to each uni-modal feature before the final concatenation.

To model the interaction between modalities, a contrastive loss function is employed, which brings aligned pairs closer and pushes misaligned pairs further in the joint embedding space. This approach incorporates contrastive learning, mainly Intra-modal Contrastive Loss (ICL) and Inter-modal Alignment Loss (IAL), which are described in section 3.3, 3.3.1 and 3.3.2. The optimization function used in this method is described in section 3.3.3.

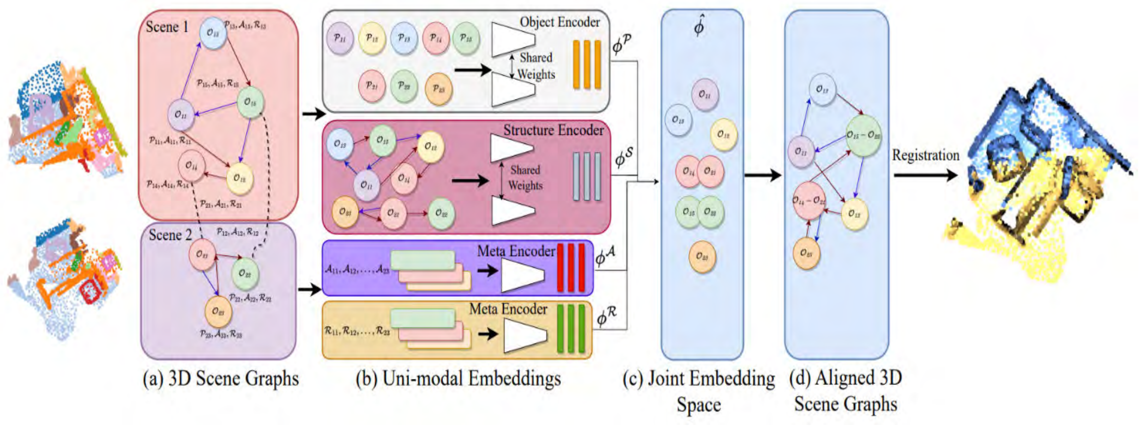


Figure 3.4: SGAligner Network Architecture, (SARKAR et al., 2023)

3.4.4 Point Cloud Registration

The output of scene alignment is the set of matches entity pairs n_1 and n_2. 3D correspondences from $\mathcal{P}_1^i$ and $\mathcal{P}_2^i$ for each entity pair is extracted. Then correspondences are estimated by an off-shell correspondence extraction algorithm, which runs on node pairs independently. After collecting the correspondences across all matched entities robust estimator, RANSAC (FISCHLER & BOLLES, 1987) is used to get the transformation matrix **T** $\in$Special Euclidean group (SE)(3) between the PCD of the two scenes.

3.4.5 Evaluation Metrics

This work also introduces alignment and point cloud registration metrics, inspired by works of LIN et al. (2022) and QIN et al. (2022). The metrics introduced here are later used by this thesis work as well.

3.4.5.1 Alignment Metrics

1. **Hits@K**
   Indicates the fraction of true anchor entities present within the top k predictions.

$$H_k(r_1, ..., r_n) = \frac{1}{n}\sum_{i=1}^{n} I[r_i \leq k] \tag{3.12}$$

   where $I[x \leq y] = 1$ when $x \leq y$ else 0 and $k \in [1, 2, 3, 4, 5]$.

2. **MRR**
   It corresponds to the average of the reciprocals of the rank of true triples.

$$MRR(r_1, ..., r_n) = \frac{1}{n}\sum_{i=1}^{n} r_i^{-1} \tag{3.13}$$

3.4.5.2 Registration Metrics

1. **Registration Recall (RR)**
   Fraction of registered point cloud pairs where transformation error is less than a specified *tolerance*. Transformation error is defined as the root mean squared error of the true correspondences after applying the transformation **T**.

   If C is the total number of true correspondences and P and Q are the two point sets, the transformation error between them is given by,

$$RMSE = \sqrt{\frac{1}{|C|}\sum_{(P_{Xi}, Q_{Yi}) \in C} ||T_{P \to Q}(P_{Xi}) - Q_{Yi}||_2^2} \tag{3.14}$$

$$RR = \frac{1}{M}\sum_{i=1}[RMSE_i < tolerance] \tag{3.15}$$

   Where M is the number of all point cloud pairs.

2. **Relative Rotation Error (RRE)**
   It measures the angular difference between the estimated rotation and real rotation

matrices. If a point set A is rotated from point set B using rotation matrix $\hat{R}$, if R is the rotation matrix calculated by the iterative process as defined earlier,

$$RRE = arccos\frac{trace(R\hat{R} - 1)}{2} \tag{3.16}$$

3. **Relative Translation Error (RTE)**
   Measures the euclidean distance between estimated and real translation vectors.

$$RTE = ||t - \hat{t}|| \tag{3.17}$$

4. **Chamfer Distance (CD)**
   It evaluates the quality of registration. Modified chamfer distance, following (Yew & Lee, 2020), is given as,

$$\begin{aligned} CD(P, Q) = &\frac{1}{|P|}\sum_{p \in P} min_{q \in Q_{raw}} ||T_P^Q(p) - q||_2^2 \\ &+ \frac{1}{|Q|}\sum_{q \in Q} min_{p \in P_{raw}} ||q - T_P^Q(p)||_2^2 \end{aligned} \tag{3.18}$$

5. **Inliner Ratio (IR)**
   It measures the proportion of inlier matches among all point matches. A match is classified as an inlier if the distance between two points is smaller than a defined threshold τ_1.

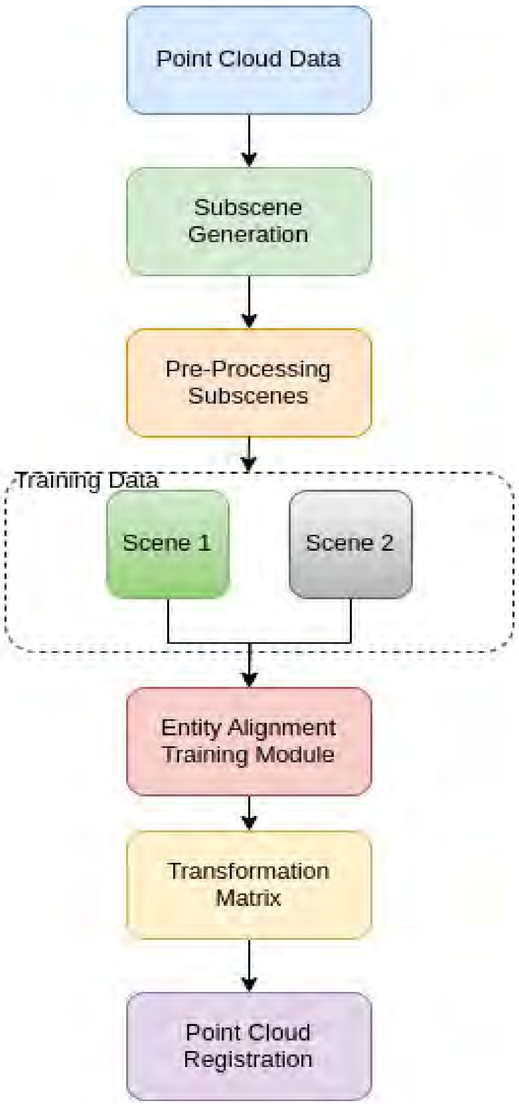
$$IR = \frac{1}{|C|}\sum_{(P_{Xi}, Q_{Yi}) \in C} I\Big[||T_{P \to Q}(P_{Xi}) - Q_{Yi}||_2 \le \tau_1\Big] \tag{3.19}$$

6. **Feature Matching Recall (FMR)**
   Fraction of point cloud pairs with an IR above a specified threshold τ_2. It measures the potential success during the registration.

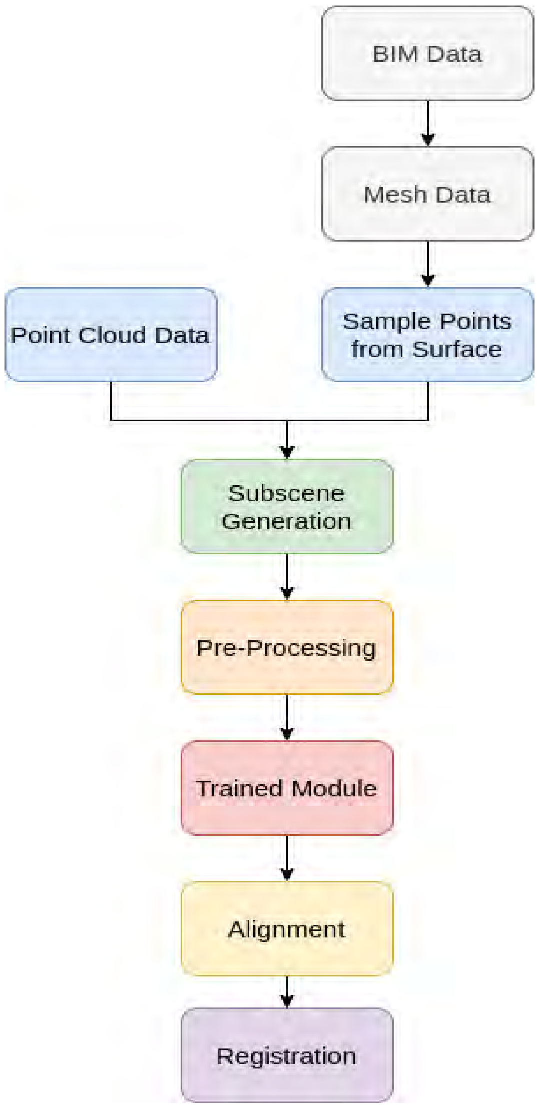
$$FMR = \frac{1}{M}\sum_{i=1}^{M} I\Big[IR_i > \tau_2\Big] \tag{3.20}$$

Chapter 4

Methodology



(a) Alignment of Point Cloud to Point Cloud



(b) Alignment of Point Cloud to BIM

Figure 4.1: Thesis Methodology

The work structure of this research work can be divided into two different pipelines. The first pipeline deals with point cloud to point cloud alignment, whereas the second pipeline deals with aligning the point cloud to BIM. Most structures are identical in both pipelines, except at the beginning. The point cloud- BIM has some intermediate step where points are sampled from the surface of the BIM. The sampled point cloud represents the BIM model and alignment of point cloud data happens with this sampled data.

After the initial dataset has been defined, subscenes are generated. A subscenes consist of two scenes, from either point cloud-point cloud or point cloud-BIM, with some overlap. Subscenes with overlap information between 0.1 and 0.9 are taken into consideration. Then they are sent to the preprocessing module whose algorithm is detailed in 4.6. An entity alignment model is trained on the subscene data and inference is performed on the validation set. The transformation matrix for the validation set is calculated and their results are visualized. The steps of the methodology are explained in detail in the following sections.

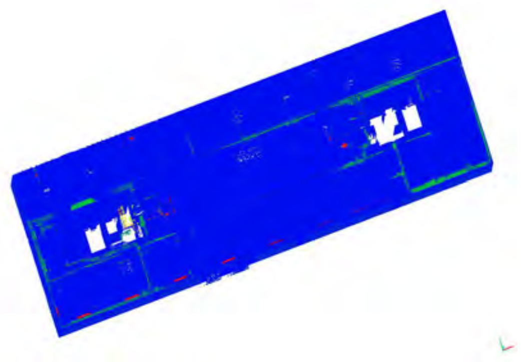
4.1 Dataset

This section 4.1 is the first step in the research process. Point Cloud dataset and BIM dataset are provided. Normally, this dataset is of an entire floor, as shown in fig 4.2.

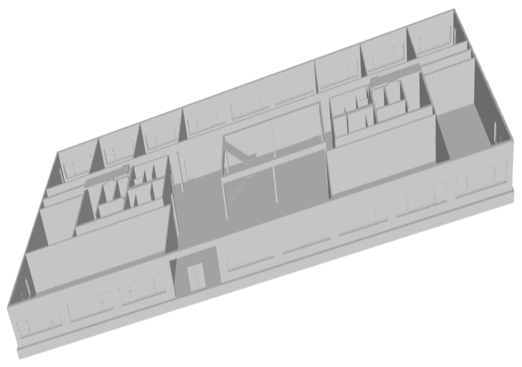
4.1.1 Point Cloud Dataset

LANDRIEU and SIMONOVSKY (2018a) introduced a point graph formulation called super point graph in 2018. It is a deep learning framework designed to tackle the problem of semantic segmentation of millions of points. On top of that, this framework offers a concise representation of contextual relationships between the objects (LANDRIEU & SIMONOVSKY, 2018a).

The point cloud dataset is then passed through this formulation to get a project element map that has updated superpoint IDs, which contain semantic labels of the object as well as their instance labels. This one dataset is then further divided into multiple datasets to train the model, which are also called legs. The legs are normally just individual rooms in a given floor and are termed as **scenes**.

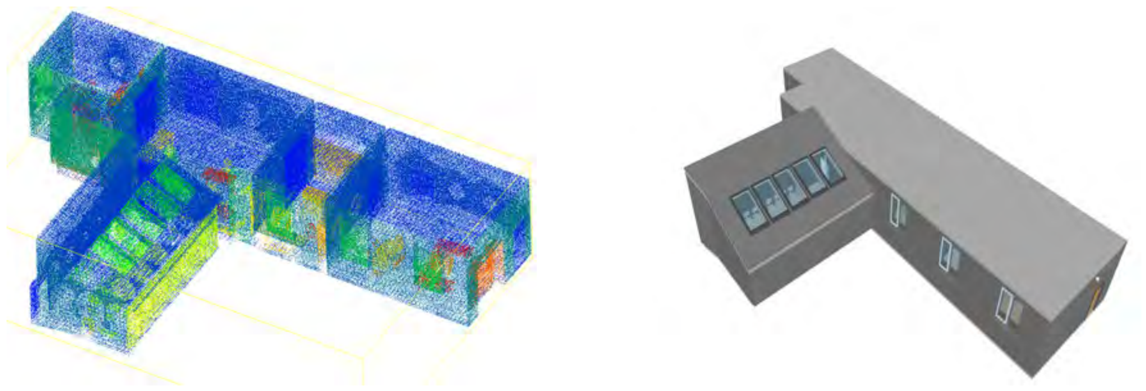


(a) Point Cloud Dataset without Furniture Elements, 75M points



(b) BIM Dataset

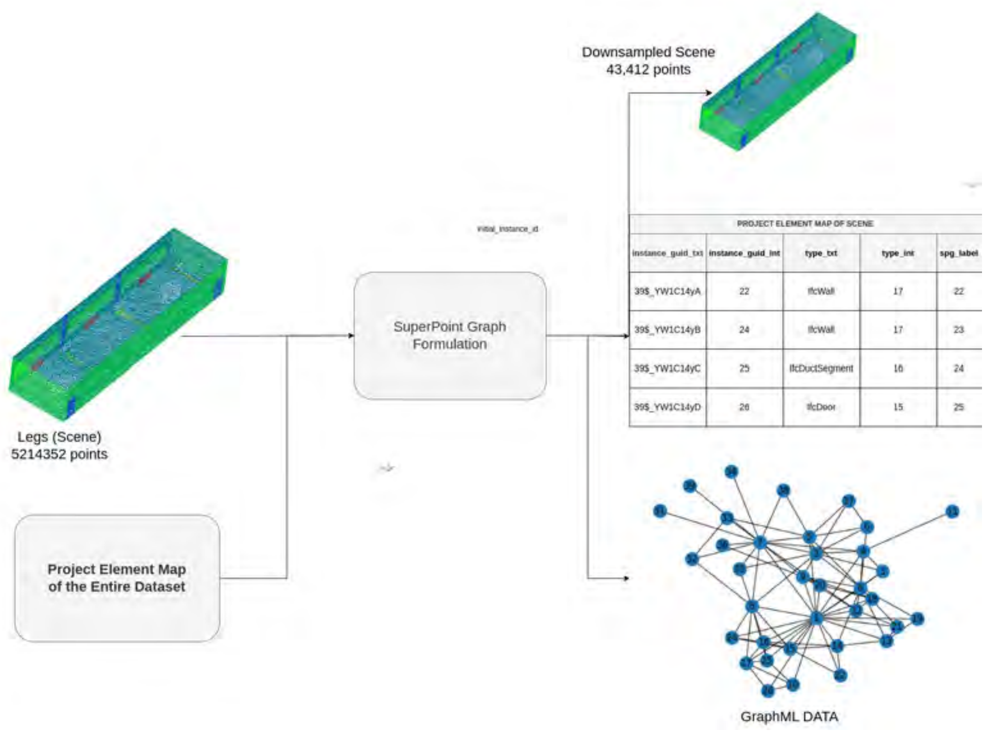
Figure 4.2: Point Cloud and BIM Dataset without Furniture


(a) Point Cloud Dataset with Furniture Elements, 66M points

(b) BIM Dataset

Figure 4.3: Point Cloud and BIM Dataset with Furniture

The individual scenes are then again passed through the super point cloud formulation to get their own individual project element map as well as a super point graph, that contains properties about the nodes (objects) in the scene and edges which contain the relationship between the objects in a given scene as shown in fig 4.6. The scene also gets down-sampled in this process. Usually, the graph is in hp5 formulation, but a pickle file of the graph formulation is also obtained.



PROJECT ELEMENT MAP OF SCENE

instance_guid_txt	instance_guid_int	type_txt	type_int	spg_label
39$_YW1C14yA	22	IfcWall	17	22
39$_YW1C14yB	24	IfcWall	17	23
39$_YW1C14yC	25	IfcDuctSegment	16	24
39$_YW1C14yD	26	IfcDoor	15	25

Figure 4.4: Scene to Graph along with its project element map

4.1.2 BIM Dataset

Industry Foundation Classes (IFC) files are normally used to represent the BIM data. To obtain a BIM graph, individual IfcSpaces with elements inside and adjacent to it are extracted using IfcOpenshell. Then the space-level data is converted into a mesh file (obj) using IfcConvert. All the elements inside of it are parsed and 5000 points are sampled from the surface of the elements. Geometric properties are calculated for each entity, similar to what is given by super-point graph formulation (LANDRIEU & SIMONOVSKY, 2018a). The algorithm for extracting the BIM-graph from a BIM dataset is shown in Algorithm 4.1, Algorithm 4.2 and Algorithm 4.3.

Algorithm 4.1: Space Information Extraction

```
Open bim_ifc_file
for space in bim_ifc_file["IfcSpace"]
    elements ← select(space, completely_within=True, extend=0.7)
    ind_file ← add(elements)
    save ind_file as space_guid.ifc
```

Afterward, elements inside and adjacent to each space are saved as a separate IFC file. The individual elements are parsed from a file extracted from the previous step and converted into a mesh separately using IfcConvert. IfcConvert is an open-source command line application that lets one convert files from one type to another. Here, obj

Algorithm 4.2: Conversion to Mesh

```
Open all space.ifc_files saved using the previous step
for space_guid.ifc in all_space.ifc files
    for element in space_guid.ifc["IfcProduct]
        save each element as an .ifc file
        IfcConvert --use-world-coords --use-element-guid element.ifc
```

files for each element related to an IfcSpace are obtained. Then these individual obj files are parsed to sample points using Open3D (Q.-Y. ZHOU et al., 2018). Then points are uniformly sampled from each element and bim graph is extracted. The schematic flow of

Algorithm 4.3: Point Sampling and BIM Graph Extraction

```
bim_graph ← {}
combined_points ← np.empty()
for id, element.obj related to space.ifc
    label ← name_of(element.obj)
    mesh ← Open3d.io.read_tringle_mesh(element.obj)
    points ← sample_points_uniform(mesh, number_of_points=5000)
    geometric_properties ← calculate_pca_and_pca_extent(points)
    combined_points ← append(combined_points, points)
    bim_graph ← add id, label, geometric_properties
```

information can be seen in fig. 4.5 BIM graph is obtained at the Space level. Its equivalent in the point cloud dataset would be the formulation at a scene level, which is explained in the section 4.1.1. No further subscenes are obtained for the BIM dataset since an entity aligner trained on point cloud to point cloud is used to study the alignment of BIM and PCD. But a similar algorithm described in section 4.2 can be employed.

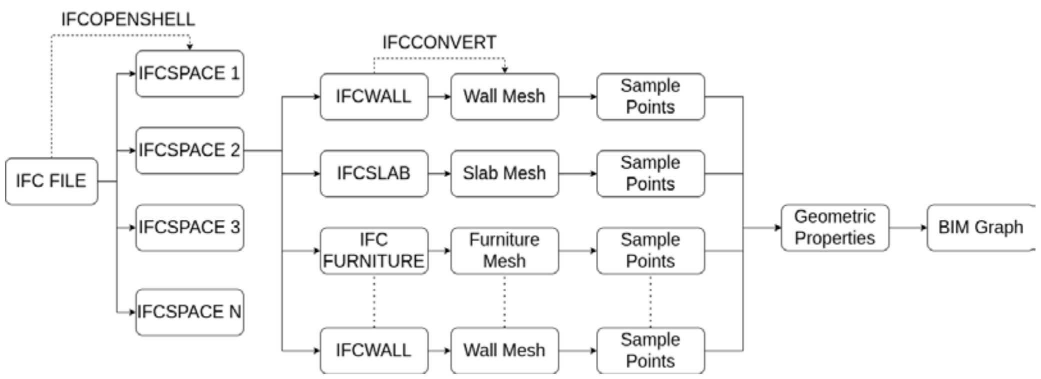


Figure 4.5: Schematic Diagram of BIM to BIM Graph

4.2 Sub-Scenes Generation

Entity alignment requires two different graph formulations with some overlap between the Knowledge graphs. Scenes produced in the earlier section are an entity of their own, with their knowledge graphs, and have no relationship with other scenes. So, the scenes are further broken down into sub-scenes. The algorithm is shown in 4.4

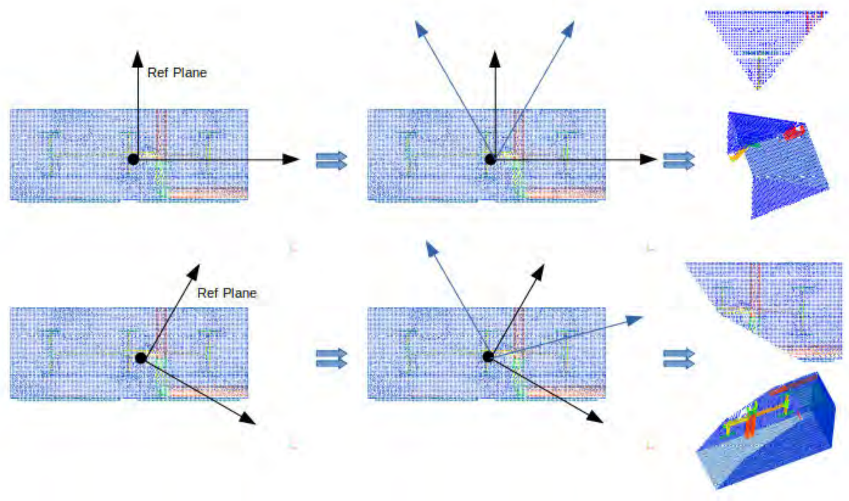


Figure 4.6: Overview of Subscene Generation. **Top**: Initialization of a ref plane, **Bottom**: Rotation of ref plane

For a given scene, viewpoint is calculated, which is the mean of the points. Then a reference plane is taken and two symmetric planes at an angle of α around the reference plane are considered. Then the points between these planes are taken. If the number of points inside these two planes satisfies the stopping criteria, a sub-scene is generated from these points, and information regarding its objects and relationships is saved. Then the reference plane rotates by a predefined angle β and the same process is repeated

Algorithm 4.4: Subscene Generation Algorithm

```
Take a Scene
    max points per subscene ← Random (20−50)% of scene points
    view point ← mean(Scene points)
    ref plane rotation ← Initialize in degrees
    start swiping angle ← Initialize
    swiping angle increment ← Initialize
    current mask ← False values with shape scene_points.shape[0]
    subscan points ← scene_points[current\_mask]
    for starting ref plane in planes parallel to [yz, zx, xy] passing
       through view point
        ref plane ← starting ref plane
        current angle ← Initialize to 0 degrees
        while current angle < 360
            while subscan points.shape[0] < max points per subscene
                plane 1 ← plane at −start swiping angle from ref plane
                plane 2 ← plane at +start swiping angle from ref plane
                current mask ← points between plane 1 and plane 2
                subscan points ← scene_points[current_mask]
                start swiping angle ← start swipping angle + swiping
                   angle increment
            object and relationship data ← subscan points
            save subscan points, object, and relationship data
            ref plane ← rotate by ref plane rotation
            reset current mask ← False values with shape scene\_points
               .shape[0]
            reset start swiping angle ← Initial value
            current angle ← current angle + ref plane rotation
```

until the reference plane does a complete sweep of 360 degrees. The reference plane passes through the viewpoint and is sequentially aligned with the three principal directions.

4.2.1 Overlap Information

Algorithm 4.5: Overlap Information

```
Take a Scene
    all subscenes ← Related to the scene
    subscan id pairs ← combinations(all subscenes, 2)
    subscan ply data all ← Append ply data of all subscenes
    for id pair in subscan id pairs
        src ply data ← subscan ply data all[id pair[0]]
        ref ply data ← subscan ply data all[id pair[1]]
        src points ← src ply[x,y,z]
        ref points ← ref ply[x,y,z]
        overlap r, common ids src ← compute overlap(src points, ref
           points)
        if 0.1 < overlap r < 0.9
            common points instance label ← src ply data["objectid"][
               common pts ids in src]
            overlap_data ← id pair, overlap ratio, common points
               instance label
            save overlap_data
```

This process is a subpart of subscene generation. After the individual subscenes are generated, information about overlap - pertaining to common elements, the extent of overlap - is extracted, whose algorithm is shown in 4.5.

4.3 Pre-processing

As discussed previously, point clouds and BIM are both information-rich resources. Information regarding structure, objects, their attributes, and the relationship between them presents a situation where multiple knowledge graphs are produced. Since this research follows the framework introduced by Sarkar et al., 2023, the multiple knowledge graphs must be processed by a dedicated modality. Before the information is passed to the dedicated module, it should be pre-processed so that information from all different sub-scenes has a concurrent pattern. **Preprocessing methods for point clouds and BIM are the same since they have the same graph structure.**

Pre-processing of subscenes can be broadly divided into four steps. The first step deals with processing structural information encompassing nodes, node features, edges, and if available, edge features. The second step is the processing of object points. In this stage, the points undergo downsampling to a predefined resolution for every object in the subscene. This is crucial as processing object points requires significant processing power. They are downsampled so that essential information is retained and computational efficiency is optimized in the process. In the third step, object features which include geometric features are processed. The final step consists of edge feature processing. Edge features include the semantic relationships between the node entities.

The pre-processing algorithm for each of these steps for each sub-scene is shown in section 4.3.1, section 4.3.2 and section 4.3.3. These algorithms are designed to organize and standardize information from different subscenes systematically.

4.3.1 Processing Structural Information

The algorithm here deals with the structure of the graph that is required by the structural encoder. This portion deals with the structure of the knowledge graph.

Center of each subscene is determined by computing the mean of its convex hull. A root node is then identified. Root node has the highest number of outgoing relationships among all the obejcts in the scene. The relative translation between the root nodes' barry center and the center of other objects within the scene is then calculated. These computed values serve as the node features in the graph structure.

For the information to flow within the graph, edges are essential along with node features. All the edges existing within the subscene are extracted. The algorithm to extract the node features and edges is shown in Algorithm 4.6.

Algorithm 4.6: Subscenes Preprocessing

```
for subscene in all subscenes generated
    object ids ← {}
    global object ids ← {}
    object atrributes ← {}
    barry centers ← {}

    object data ← objectdata[subscene]
    relation data ← relationdata[subscene]
    ply data ← load ply data(subscene)
    points ← ply data["x","y","z"]
    resolution ← Initialize

    for id, object in enumerate(object data)
        object attribute ← object["atributes"]
        object id = object["id"]

        global object id ← object["global id"]
        object pts ← where(ply data["objectId"] == object id)
        object pcl ← points[object pts]
        if object pcl.shape[0] < read_from_config("min_obj_points")
            continue
        hull ← ConvexHull(object pcl)
        cx, cy, cz ← mean(hull)
        # Decrease the point density of resolution for each object
        object points ← pcl_farthest_sample(object pcl, resolution)

        barry center ← cx, cy, cz
        object ids ← append(object id)
        global object ids ← append{global object id}
        barry centers ← append(barry center)
        object attributes ← aapend(object arribute)

    triples ← {}
    pairs ← {}
    edges ← {}
    for id, obj id1, obj id2, relation in enumerate(relation data)
        if obj id1 and obj id2 in object ids
        triples ← append(obj id1, obj id2)
        edges ← append(call REL2ID(relation))

        if obj id1 and obj id2 not in pairs
            pairs ← append(obj id1, obj id2)

    # get obj with high number of relation
    root obj idx ← maxrelation(pairs)
    relative translation ← {}
    for barry center in barry centers
        relative translation ← append(barry center[root obj idx],
            barry center)

    for id1 in object ids:
        for id2 in object ids:
            if (i,j) not in pairs and if i != j
            triples ← append(id1, id2, REL2ID("none"))
            pairs ← append(id1, id2)
            edges ← append(call REL2ID("none"))

    save data_dict[subscene]
```

4.3.2 Processing Attribute Information

The attributes associated with an object in a given subscene undergo processing to facilitate their integration into the attribute encoder.

Three types of attribute information are considered, object attributes calculated by the super point graph formulation (LANDRIEU & SIMONOVSKY, 2018a), ifc embeddings proposed by KAYHANI et al. (2023), and a combination of both.

The algorithm for processing geometric features from super point graph formulation is not shown here because it has already been saved into a dictionary file while processing structural information in 4.3.1

The algorithm shown in 4.7 adds global information, one hot encoding of object labels, to the attribute feature.

Algorithm 4.7: IFC2VEC Embeddings Processing

```
for subscene in all subscenes
    data_dict ← load data_dict[subscene]
    ifc2vecemb ← {}
    for object in data_dict["objects"]
        ifc2vecemb ← append(ifc2vecembeddings(object["label"]))
    save ifc2vecemb in data_dict[subscene]
```

The algorithm shown in 4.8 combines global information in ifc2vec embeddings to local-level object features calculated using super point graph formulation.

Algorithm 4.8: Combining geometric and ifc2vec embeddings

```
for subscene in all subscenes
    data_dict ← load data_dict[subscene]
    combined attributes ← {}
    for object in data_dict["objects"]:
        combined attributes ← append(data_dict["object attributes],
            data_dict["ifc2vecemb"]
    save in the same data dict[subscene]
```

4.3.3 Processing Relation Information

Super Point graph formulation provides bidirectional relational information. There's no semantic information about the label that could improve the alignment metric. Both the approaches are tried and the results are shown in chapter 5. The edge feature for each node in a sub-scene is converted into a feature vector using the bag-of-words (BOW) (QADER et al., 2019) technique. For each object, the frequency of occurrence of a type of relation is used as a feature.

Algorithm for processing relation information is shown in 4.9

Algorithm 4.9: Processing Relation

```
for subscene in all subscenes
    data_dict ← load data_dict[subscene]
    word2Idx ← load_from(REL2IDX.keys)
    edges, triples ← data_dict["edges"], data_dict["triples"]
    entities edge names ← [None] * len(data_dict["object_ids"])
    for idx in len(edges)
        edge = edges[idx]
        entity = edge[0]
        rel name = IDX2REL(triples[idx][2])
        entities edge names ← append(rel name)
    entity edge feat ← {}
    for entity edge name in entities edge names
        entity edge feat ← append(makefeaturevector(entity edge name))
```

Super point graph formulation only provides adjacency information. So, two building elements are either "connected" or not. Elements having no adjacency are considered to have a relation of type "none". This information is then used to get a two-dimensional feature vector, where the first dimension represents bag-of-words for "connected" and the second dimension represents bag-of-words for "none" for each object.

Besides this, another formulation is also used. Using the information obtained from the super point graph formulation, a semantic label is added according to the relative position of one building element with the other. The algorithm to process this information is also the same as shown in Algorithm 4.9.

4.4 Multi-Modal Encoder

There are four different individual encoders as a part of a multi-modal encoder. They each work separately on different types of data. Individual encoders can be switched on and off depending on requirements. All encoders work simultaneously and embeddings in joint space are produced which is then used to check how the model performs with alignment.

1. **Structure Encoder** $(\phi^{\mathcal{S}})$

   Multi-headed Graph Attention Network followed by a linear layer form structure encoder. A linear layer is needed to get the embeddings down to a specific size, which is a hyper-parameter.

   GATConv module from PyTorch geometric is used. A subscene is taken and a root node is obtained. In a subscene, each node represents an object, and the root node is the one that has the highest number of connections among all nodes in the subscene. Node features are relative translations of a given node with respect to the root node, which is done during the preprocessing step, described in 4.3.1. So each node will have a feature size of 3.

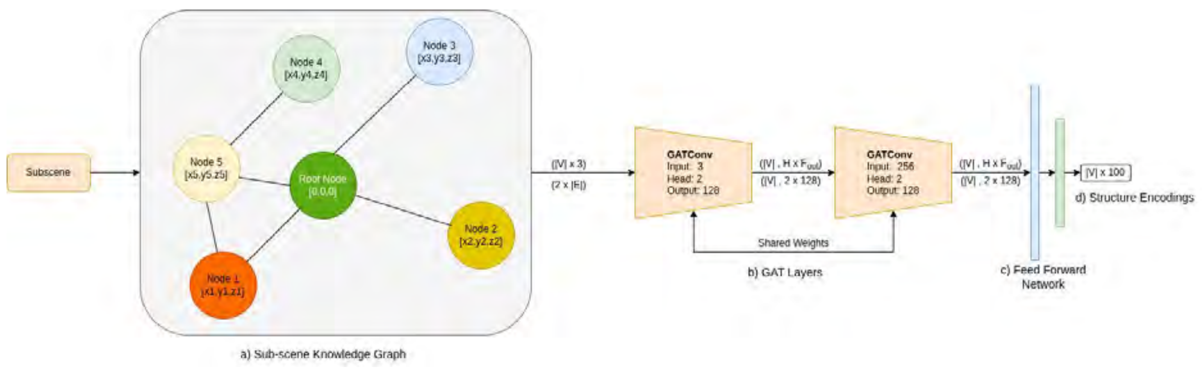


Figure 4.7: Structure Encoder

|V| represents the cardinality of objects in the subscene. Two GATConv layers are used, and the output is fed to a feed-forward network which down-samples the embeddings from **256** to **100**. The final output embedding size is $|\mathbf{V}| \times \mathbf{100}$.

2. **Object Encoder** $(\phi^{\mathcal{P}})$

   PointNet, (QI et al., 2017) architecture encodes the geometric information in the object points. This module is computationally expensive, so the points are down-sampled before extracting the features. (SARKAR et al., 2023) demonstrated state-of-the-art performance with 512 points per object per scene. They also obtained similar performance metrics using only 64 points, with a marginal drop of 3% on MRR.

   The dataset used by this thesis work undergoes downsampling during the preprocessing phase while undergoing super point graph formulation,(LANDRIEU & SIMONOVSKY, 2018a). Therefore, the choice is made to retain 512 points per object per scene as used by SARKAR et al. (2023).

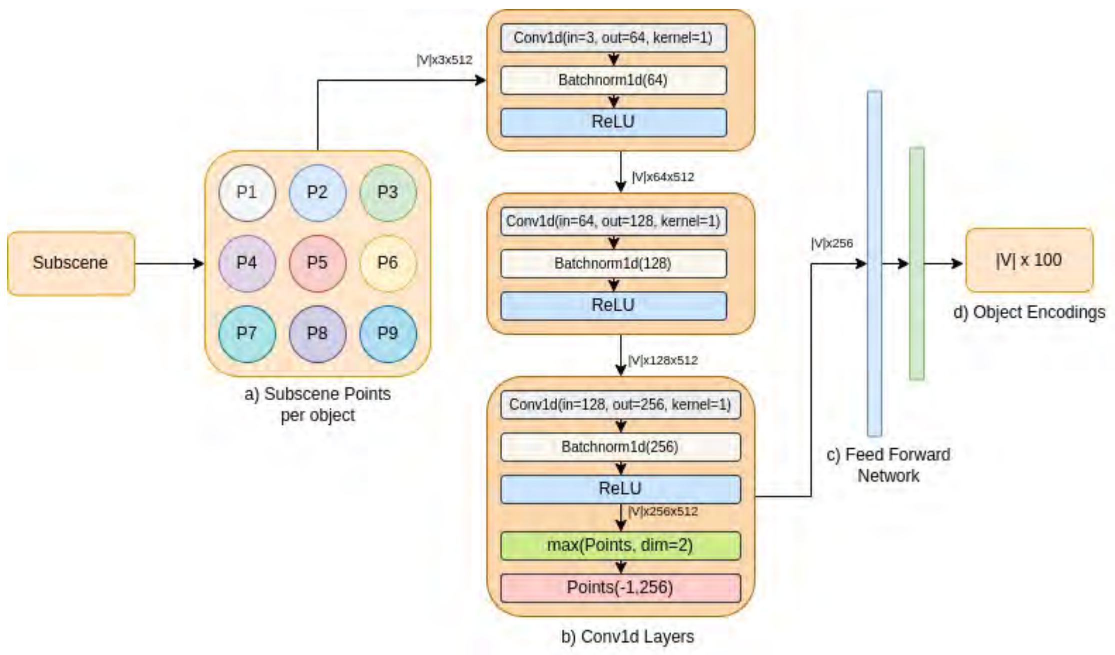


Figure 4.8: Object Encoder

Instead of the entire PointNet architecture, only a subset of the network encodes the point features. Three Conv1d layers, batchnorm1d layers with relu as an activation function are used with feed-forward network to reduce the feature from $|\mathbf{V}| \times \mathbf{3} \times \mathbf{512}$ to final output object embedding size of $|\mathbf{V}| \times \mathbf{100}$.

3. **Attribute MetaEncoder** $(\phi^{\mathcal{A}})$

   A feed-forward network is used to map the features to a specific dimension. Three different types of attributes are used to see which results in the best alignment results. Geometric features, IFC embeddings, and a combination of both were used. This encoder maps from $|\mathbf{V}| \times \mathbf{Attribute\_type\_size}$ to final attribute embeddings of size $|\mathbf{V}| \times \mathbf{100}$. Attribute size for **12** for geometric embeddings, **182** for IFC embeddings, and **194** for combined embeddings. Geometric features as discussed in section 3.2 are used.

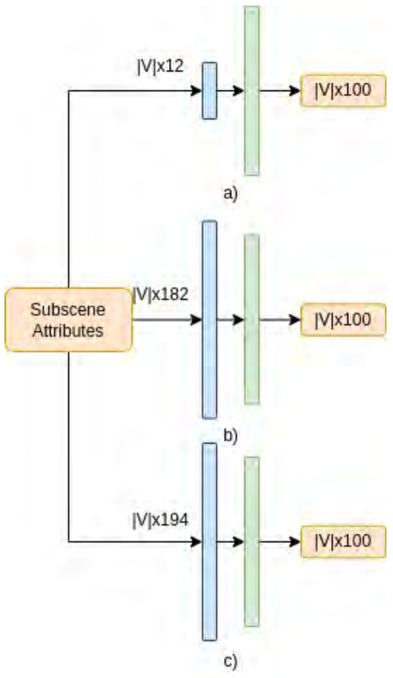


Figure 4.9: Attribute Encoder. Fig a) Geometric Attribute Encoder $(\phi^{\mathcal{A}}{}_{g})$. Fig b) IFC Embeddings Attribute Encoder $(\phi^{\mathcal{A}}{}_{i})$. Fig c) Combined geometric and IFC Embeddings Attribute Encoder $(\phi^{\mathcal{A}}{}_{g+i})$

   These attribute encoders are not used in parallel or conjunction. Fig 4.10 shows the available attribute encoders. Only one of them can be used at a time. No matter which type of attribute encoder has been used, the final output embedding size for all the cases is $|\mathbf{V}| \times \mathbf{100}$.

4. **Relation MetaEncoder** $(\phi^{\mathcal{R}})$

   A feed-forward network is employed to get relational embeddings like an attribute encoder. Two different relational encoders are formulated according to the type of data. First, relational information between nodes as discussed in section 3.2.2 of chapter 3, using delauny triangulation method, is used. This information only gives us bidirectional information, whether two elements are touching. So the relational dimension in this case is **2**. This information is encoded and the final embedding size of $|\mathbf{V}| \times \mathbf{100}$ is outputted.

Semantic meaning is added to the relation using "connected" or "none" information between the nodes for the second one. Semantic meaning is derived depending upon the relative position of one object with respect to another object. So, now the relation dimension changes from the previous value of 2 to 11. Relational information is changed to one of **behind, front, left, right, above, down, standing on, embedded, attached, connected, same object type, none**. Then this relation information is first converted into bag-of-words (BOW) feature vector as discussed in section 4.3.3 and passed to the feed-forward network to get a final embedding size of $|\mathbf{v}| \times \mathbf{100}$.

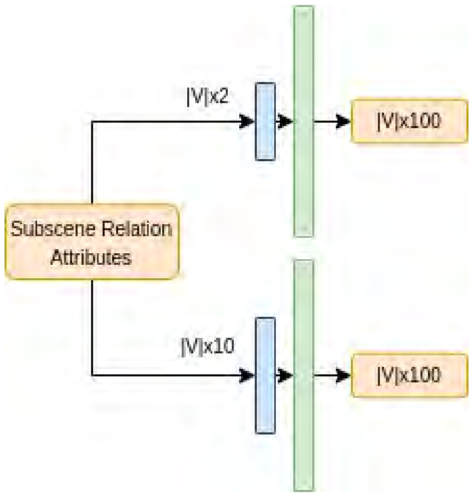


Figure 4.10: Relational Encoder. Top arrow represents embeddings obtained using 2 features. Bottom arrow represents embeddings obtained using 10 features

4.5 Network Architecture

The previous section detailed the available encoders and how they encode information from a subscene. However, one data instance consists of two different sub-scenes with some overlap. So the information from both the subscenes needs to be concatenated to get embeddings in the joint space. This network is termed as **TUMAligner**.

Figure 4.11 shows the overview of TUMAligner. Embeddings from all the individual models are combined to form joint embeddings. $|\mathbf{V_{s1}}|$ represents the cardinality of objects in subscene 1 and $|\mathbf{V_{s2}}|$ in subscene 2. If all the modalities are used then the total size of embeddings produced in the joint embedding space is $(|\mathbf{V_{s1}}| + |\mathbf{V_{s2}}|) \times \mathbf{400}$.

The generation process of joint embeddings is explained in section 3.4.3. In the network, a **fusion layer**, takes the embeddings produced by individual modalities, normalizes them using L-2 normalization and assigns learnable weights to each modality. These weights represent the importance of each modality in the joint embedding space. Then, they are

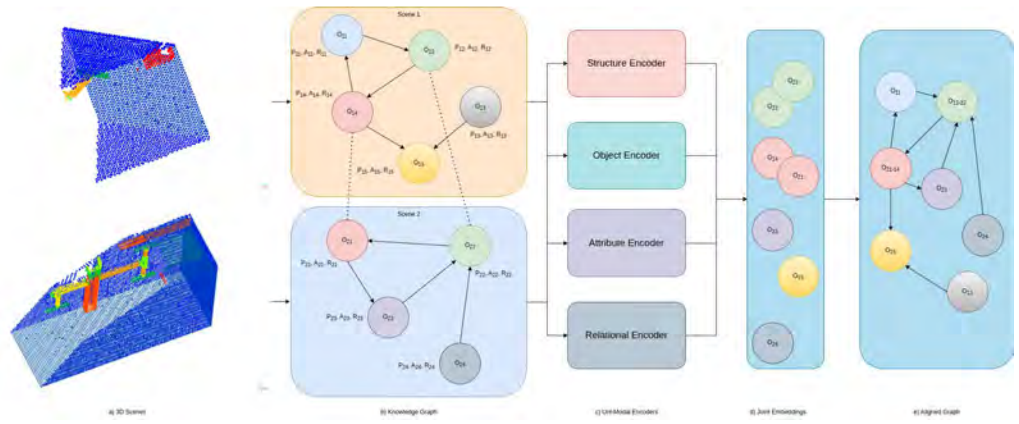


Figure 4.11: Overview of TUM Aligner

concatenated to form joint embeddings, which are used downstream for finding node correspondences and finding the transformation matrix between the subscenes.

4.6 Node Correspondences

After the normalized joint embeddings ($\hat{\phi}_i$) is obtained, a distance matrix is calculated, which contains the list of node ids. Then it is sorted from highest to lowest to get the ranked order. The algorithm to obtain node correspondences is shown below.

Algorithm 4.10: Node Correspondences

```
Get Joint Embedding
k ← initialize
dist matrix ← 1 - $\hat{\phi}_i\hat{\phi}_i^T$
ranked_list ← sort(dist_matrix)
node_corrs = []
for id in src_objects
    entity_rank ← ranked_list[id]
    remove(element at entity_rank[id])
    get top k predictions ← entity_rank[:k]
    for k_id in k:
        if not entity_rank[k_id] < src_obj_count
            node_corrs ← append(id, entity_rank[k_id])
get object ids corresponding to node correspondences
```

4.7 Registration

FindRidigTransform function from pygcransac library is utilized to get an estimated transformation between the source points and reference points. After the node correspondences are obtained, points corresponding to ids in both source and reference are extracted. Then point correspondences are obtained using a method implemented by QIN et al. (2022). These correspondences are then passed to findrigidtransform method to obtain the transformation.

Algorithm 4.11: Estimating Transformation

```
Get ground truth node corres points in both src and ref
Initialize Ransac parameters
for node in node_corrs:
    points_src ← src_points[objectId == node_corr[0]]
    points_ref ← ref_points[objectId == node_corr[1]]
    ref_corr_points, src_corr_points ← append(performregistration(
        points_src, points_ref) from Geotransformer)
src_corr ← concatenate(src_corr_points)
ref_corr ← concatenate(ref_corr_points)
corrs_ransac ← concatenate(src_corr, ref_corr)
est_transformation ← findRigidTransform(corrs_ransac, ransac_
    parameters)
```

Chapter 5

Results

Different experiments were done with different modules and attributes. Different types of datasets were used. Point clouds with just architectural information and with architectural and furniture information. Both alignment results and registration results for all considered cases will be discussed in this chapter in great detail. The first two sections deal with graph alignment of point cloud and point cloud and the third section deals with leveraging that result to align a point cloud and its bim counterpart.

5.1 Architectural Data

Data containing only the information of architectural elements were used in this experiment. The number of different semantic classes in this case was 16.

Table 5.1: List of Architectural Classes

IfcBeam	IfcColumn	IfcDoor	IfcSlab
IfcWall	IfcWindow	IfcAirTerminal	IfcDuctFitting
IfcDuctSegment	IfcSpaceHeater	IfcPipeSegment	IfcSanitaryTerminal
IfcUnitaryEquipment	IfcPipeFitting	IfcStairFlight	IfcDamper

In the sections following this, only the overall loss is shown. All the other losses are shown in appendix B. Every scenario listed in the sections below shows a comprehensive loss graph alongside alignment results. Subsequently, the registration results are computed and compared with those obtained from GeoTransformer, a state-of-the-art registration method. Listing the registration result of every scene is avoided and a handful of the results since registration metrics for different cases are similar, are listed in B.1.6.1.

5.1.1 Case 1: Structure and Object Encoder

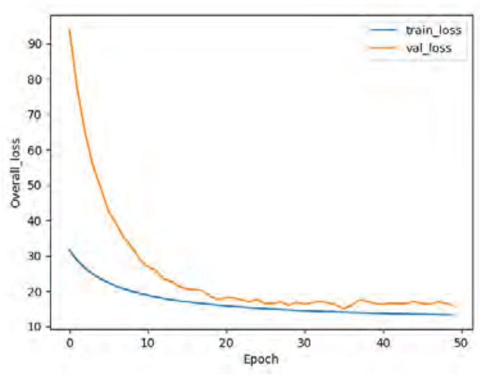


Figure 5.1: Overall Loss of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$

Metric	TUMAligner
MRR	0.8748
hits@1	0.8195
hits@2	0.8787
hits@3	0.9167
hits@4	0.9345
hits@5	0.9469

Table 5.2: Alignment Metric of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$

The loss graph shows that the model has converged and is not overfitting. An alignment score of 0.87 is observed when structure and object information is incorporated into the model. The registration results for this case are shown here in table 5.3.

Table 5.3: Comparison of Registration results of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$ with GeoTransfomer

Metric	TUMAligner ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$)	GeoTransformer
CD	0.0040	0.0228
RTE	0.0028	0.0088
RRE	0.0821	0.2975
RR	1.00	1.00

5.1.2 Case 2: Structure, Object and Relational Encoder

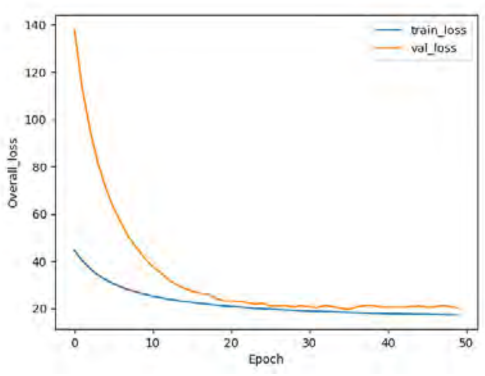


Figure 5.2: Overall Loss of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}}$

Metric	TUMAligner
MRR	0.8868
hits@1	0.8275
hits@2	0.8995
hits@3	0.9374
hits@4	0.9561
hits@5	0.9650

Table 5.4: Alignment Metric of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}}$

A relational encoder where relation information is a relation feature vector of 2 is incorporated into the model along with structure and object information. Alignment scores improve slightly by 1% from the base case.

Table 5.5: Comparison of Registration results of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}}$ with GeoTransfomer

Metric	TUMAligner ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + +\phi^{\mathcal{R}}$)	GeoTransformer
CD	0.0040	0.0238
RTE	0.0035	0.0090
RRE	0.0330	0.3092
RR	1.00	1.00

A relational encoder with a relation feature vector of 11 was also incorporated. The experiment was first conducted on a different model, shown in table 5.12. Since no significant deviations were observed in the results when new semantics were fed to the model, therefore, a decision was made to keep the original bidirectional semantics.

5.1.3 Case 3: Structure, Object, Relational and Attribute (ifc2vec) Encoder

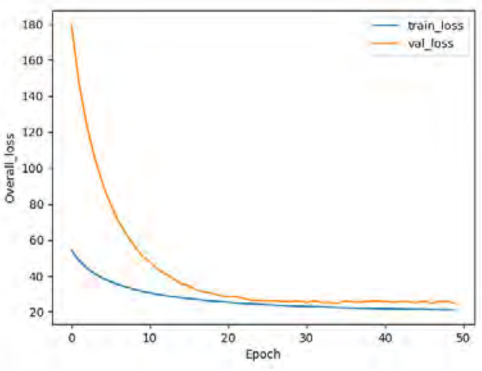


Figure 5.3: Overall Loss of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_i^{\mathcal{A}}$

Metric	TUMAligner
MRR	0.9336
hits@1	0.8933
hits@2	0.9484
hits@3	0.9709
hits@4	0.9807
hits@5	0.9854

Table 5.6: Alignment Metric of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_i^{\mathcal{A}}$

In this case, along with the existing information, a global-level feature in the form of a semantic label of the object is included in the model to study its effects. Introducing this global-level feature results in an enhancement of the alignment metric from 0.87 to 0.93. Improvements in **hits@k** are also noticeable.

Table 5.7: Comparison of Registration results of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_i^{\mathcal{A}}$ with GeoTransformer

Metric	TUMAligner ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_i^{\mathcal{A}}$)	GeoTransformer
CD	0.0031	0.0242
RTE	0.0036	0.0093
RRE	0.0682	0.3272
RR	1.00	1.00

Registration results also demonstrate improvement in comparison to GeoTransformer. All the metrics used for registration show improvement as seen in table 5.7.

5.1.4 Case 4: Structure, Object, Relational and Attribute (geometric) Encoder

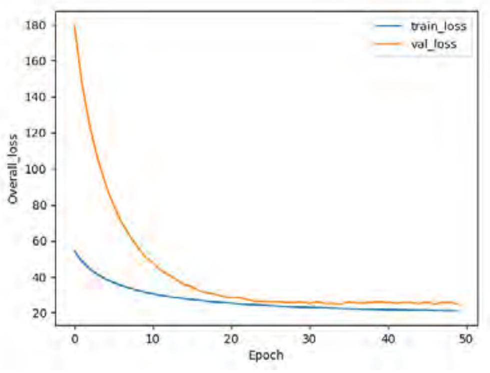


Figure 5.4: Overall Loss of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_g^{\mathcal{A}}$

Metric	TUMAligner
MRR	0.9341
hits@1	0.8959
hits@2	0.9502
hits@3	0.9641
hits@4	0.9739
hits@5	0.9807

Table 5.8: Alignment Metric of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_g^{\mathcal{A}}$

Geometric features of objects were obtained when using super point graph formulation on the point sets. These instance-level features were introduced to the attribute encoder to investigate its effects.

Table 5.9: Comparison of Registration results of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_g^{\mathcal{A}}$ with GeoTransfomer

Metric	TUMAligner ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_g^{\mathcal{A}}$)	GeoTransformer
CD	0.0041	0.0230
RTE	0.0037	0.0098
RRE	0.0244	0.3055
RR	1.00	1.00

5.1.5 Case 5: Structure, Object, Relational and Attribute (combined) Encoder

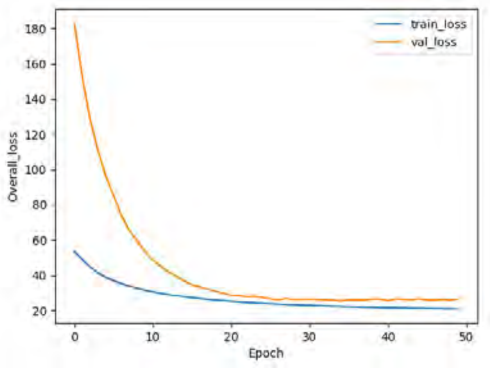


Figure 5.5: Overall Loss of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_{i+g}^{\mathcal{A}}$

Metric	TUMAligner
MRR	0.9712
hits@1	0.9522
hits@2	0.9789
hits@3	0.9899
hits@4	0.9943
hits@5	0.9964

Table 5.10: Alignment Metric of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_{i+g}^{\mathcal{A}}$

In this experiment, the global and local level features were combined to see if that would improve the model's performance. All the information available about a subscene was fed to the model.

Table 5.11: Comparison of Registration results of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$ with GeoTransfomer

Metric	TUMAligner ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$)	GeoTransformer
CD	0.0018	0.0246
RTE	0.0016	0.0096
RRE	0.0085	0.3253
RR	1.00	1.00

5.1.6 Case 6: Structure, Relation and Attribute Encoder(combined)

Here, the object encoder was dropped as the network encodes information about the geometric characteristics of point sets. This decision was taken as the geometric information was already available from the superpoint graph formulation for each object.

At first, a run with newly defined relationships was conducted for this case, with 11 different types of relations. The result can be observed in table 5.12.

Table 5.12: Alignment Metric of $\phi^{\mathcal{S}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$ with changed relationship semantics

Metric	TUMAligner
MRR	0.9953
hits@1	0.9912
hits@2	0.9981
hits@3	0.9983
hits@4	0.9988
hits@5	0.9992

Then the semantics were reverted to the original to study the effects.

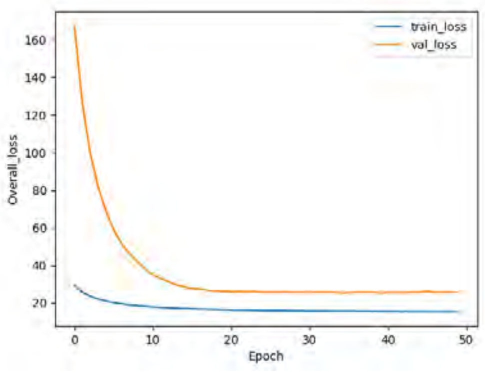


Figure 5.6: Overall Loss of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$

Metric	TUMAligner
MRR	0.9975
hits@1	0.9959
hits@2	0.9985
hits@3	0.9985
hits@4	0.9988
hits@5	0.9994

Table 5.13: Alignment Metric of $\phi^{\mathcal{S}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$

This experiment combined the global and local level features to see if that would improve the model's performance. All the information available about a subscene was fed to the model.

Table 5.14: Comparison of Registration results of $\phi^{\mathcal{S}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$ with GeoTransfomer

Metric	TUMAligner ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$)	GeoTransformer
CD	0.0047	0.0276
RTE	0.0030	0.0120
RRE	0.0952	0.3701
RR	1.00	1.00

The alignment result demonstrates drastic improvement compared to all the cases.

5.1.7 Results Comparison

Table 5.15: Comparison of Alignment Results between different models

Method	MRR	hits@1	hits@2	hits@3	hits@4	hits@5
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$	0.8748	0.8195	0.8787	0.9167	0.9345	0.9469
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}}$	0.8868	0.8275	0.8995	0.9374	0.9561	0.9650
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i}$	0.9336	0.8933	0.9484	0.9709	0.9807	0.9854
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{g}$	0.9341	0.8959	0.9502	0.9641	0.9739	0.9807
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$	*0.9712*	*0.9522*	*0.9789*	*0.9899*	*0.9943*	*0.9964*
$\phi^{\mathcal{S}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$	**0.9975**	**0.9959**	**0.9985**	**0.9985**	**0.9988**	**0.9994**

Table 5.16: Comparison of Registration Results between different models

Method	CD	RTE	RRE
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$	0.0040	0.0028	0.0821
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}}$	0.0040	0.0035	0.0330
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i}$	0.0031	0.0036	0.0682
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{g}$	0.0041	0.0037	0.0244
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$	*0.0018*	*0.0016*	*0.0085*
$\phi^{\mathcal{S}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$	0.0047	0.0030	0.0952

Table 5.15 and Table 5.16 highlight the alignment and registration metrics of the best-performing and the second-best-performing model. At every level, as new information related to the subscenes was fed, the alignment metric and the registration metric went up. It can also be observed that the alignment metric went up when the point encoder was dropped.

5.1.8 Ablation Study

A series of ablation studies on various cases were conducted to assess the robustness of the model. The investigation involved three cases: changing node semantics, removing nodes, and removing edges.

For each case, two distinct scenarios were run to understand how changes to the training and validation sets affect the model's performance. In the first scenario, only the validation set was modified keeping the training set intact. The validation set was kept intact in the second case, and only the training set was altered.

5.1.8.1 Changing Node Semantic

The semantics of nodes were changed using the IFC2VEC embeddings, (KAYHANI et al., 2023). Initially, the label of a randomly selected node was obtained, and its corresponding IFC2VEC embeddings was calculated. Using cosine similarity, another label with the highest similarity with the node in the embedding space was assigned as the new label. Between 15% to 41% of the nodes in a given subscene were randomly selected and their semantics changed.

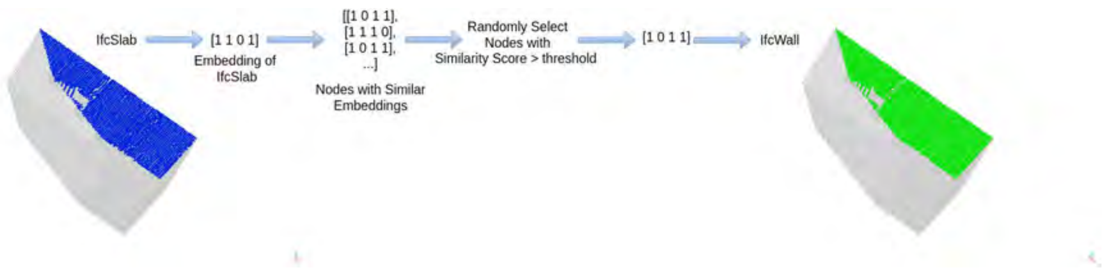


Figure 5.7: Altering Node Semantic Example

Table 5.17: Node Semantic Changed

Dataset	MRR	hits@1	hits@2	hits@3	hits@4	hits@5
Val Changed	0.7186	0.6742	0.6978	0.7178	0.7389	0.7546
Train Changed	0.9791	0.9648	0.9866	0.9940	0.9940	0.9943

Here, it is observed that changing node semantics in the validation degrades the model's performance. But, when semantics are altered in training data, the model's performance almost stays the same, indicated by the metric MRR. This phenomenon can be attributed to the inclusion of altered data in the training process. Since the model dynamically learns and adjusts the weight of each modality in the final joint embeddings, no significant performance degradation is observed.

5.1.8.2 Removing Edges

Out of all the edges defined in the preprocessing step, between 15% to 41% of the edges are selected randomly and removed. The result is tabulated below. Removing edges did not impact the performance of the model by much.

Table 5.18: Removing Edges

Dataset	MRR	hits@1	hits@2	hits@3	hits@4	hits@5
Val Changed	0.9951	0.9922	0.9951	0.9982	1.0000	1.0000
Train Changed	0.9988	0.9979	0.999	1.0000	1.0000	1.0000

5.1.8.3 Removing Nodes

Similar to removing edges, around 15% to 41% of the nodes in a subscene is selected and removed and the performance is tabulated below.

Table 5.19: Removing Nodes

Dataset	MRR	hits@1	hits@2	hits@3	hits@4	hits@5
Val Changed	0.9958	0.9923	0.9982	1.000	1.0000	1.0000
Train Changed	0.9941	0.9892	0.9971	1.000	1.000	1.000

Compared to the base model, the model's performance improves when random nodes are removed from the training set.

5.2 Architectural Data with furniture Elements

Data containing information on architectural elements and furniture elements were used in this experiment. Similar experiments as in the section 5.1 were also performed in this case. In this case, the number of semantic classes was 7. ***Unlike the previous section, here subscenes are transformed randomly and included in both the training and validation sets***.

Only the detail results of $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$ and the best-performing model, $\phi^{\mathcal{S}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$ are listed down in the following sections. It is done to avoid listing every attempted model in detail. However, in section 5.2.3 the final results of all tried models are tabulated.

Table 5.20: List of Architectural and Furniture Classes

IfcDoor	IfcRoof	IfcSlab	IfcWall
IfcFurniture	IfcWindow	IfcSystemFurnitureElement	

5.2.1 Case 1: Structure, Object, Relation, and Attribute (combined) Encoder

Metric	TUMAligner
MRR	0.9388
hits@1	0.9307
hits@2	0.9507
hits@3	0.9714
hits@4	0.9767
hits@5	0.9814

Table 5.21: Alignment Metric

Metric	TUMAligner	GeoTransformer
CD	0.0860	0.0967
RTE	0.2154	0.2189
RRE	1.8218	1.8218
RR	1.00	1.00

Table 5.22: Comparison of Registration results of with GeoTransfomer

5.2.2 Case 2: Structure, Relation, and Attribute (combined) Encoder

Metric	TUMAligner
MRR	0.9718
hits@1	0.9608
hits@2	0.9709
hits@3	0.9804
hits@4	0.9820
hits@5	0.9851

Table 5.23: Alignment Metric

Metric	TUMAligner	GeoTransformer
CD	0.0487	0.0611
RTE	0.1220	0.1264
RRE	0.9563	1.1128
RR	1.00	1.00

Table 5.24: Comparison of Registration results of with GeoTransfomer

5.2.3 Results Comparison

Table 5.25: Comparison of Alignment Results between different models for data with furniture

Method	MRR	hits@1	hits@2	hits@3	hits@4	hits@5
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$	0.7958	0.7100	0.8005	0.8545	0.8862	0.9095
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}}$	0.8064	0.7328	0.8037	0.8555	0.8761	0.8994
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i}$	0.8541	0.7931	0.8550	0.8936	0.9169	0.9418
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{g}$	0.8981	0.8566	0.9052	0.9243	0.9386	0.9481
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$	<u>*0.9388*</u>	<u>*0.9037*</u>	<u>*0.9507*</u>	<u>*0.9714*</u>	<u>*0.9767*</u>	<u>*0.9814*</u>
$\phi^{\mathcal{S}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$	**0.9718**	**0.9608**	**0.9709**	**0.9804**	**0.9820**	**0.9851**

Table 5.26: Comparison of Registration Results between different models for data with furniture

Method	CD	RTE	RRE
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$	0.0991	0.2434	1.6379
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}}$	**0.0200**	**0.0245**	**0.1280**
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i}$	0.0671	0.1686	1.073
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{g}$	0.0494	0.1234	0.9690
$\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$	0.0860	0.2154	1.6860
$\phi^{\mathcal{S}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$	0.0487	0.1220	0.9563

As observed in the previous section, the model where structure, relation, and combined attributes were fed, has the best alignment metric out of all the models.

5.2.4 Ablation Study

A similar study is performed as described in section 5.1.8. The results for each of the cases are tabulated in table 5.27, table 5.28 and table 5.29

5.2.4.1 Changing Node Semantic

Table 5.27: Node Semantic Changed for data with furniture

Dataset	MRR	hits@1	hits@2	hits@3	hits@4	hits@5
Val Changed	0.6253	0.5608	0.5920	0.6211	0.6460	0.6603
Train Changed	0.9656	0.9555	0.9619	0.9687	0.9746	0.9793

5.2.4.2 Removing Edges

Table 5.28: Removing Edges for data with furniture

Dataset	MRR	hits@1	hits@2	hits@3	hits@4	hits@5
Val Changed	0.9598	0.9370	0.9703	0.9804	0.9820	0.9846
Train Changed	0.9718	0.9608	0.9709	0.9798	0.9820	0.9851

5.2.4.3 Removing Nodes

Table 5.29: Removing Nodes for data with furniture

Dataset	MRR	hits@1	hits@2	hits@3	hits@4	hits@5
Val Changed	0.9699	0.9558	0.9721	0.9779	0.9849	0.9895
Train Changed	**0.9780**	**0.9677**	**0.9777**	**0.9888**	**0.9904**	**0.9915**

5.3 Point Cloud and BIM Data

Leveraging the result of the previous section, a study was carried out to see how well a BIM model would match its point cloud counterpart. The best-performing model was used. The results are tabulated in table 5.30.

Table 5.30: Alignment of Point Cloud to BIM

MRR	hits@1	hits@2	hits@3	hits@4	hits@5
0.5368	0.4545	0.4545	0.5454	0.5454	0.5454

The MRR score sharply declined from 0.97 to 0.53, a drop of 45%. Not only that, but the hits@K also shows a noticeable decline.

Chapter 6

Discussion and Limitations

In this section results of the previous chapter 5 are discussed and the objectives set at the section 1.3 are addressed.

6.1 Discussion

6.1.1 Architectural Data

In the context where only architectural elements were considered, the alignment score, MRR increased as more information about the scene was given to the model. With the addition of global-level and local-level features, the MRR score went to **0.97** from **0.87**. Just considering hits@1, where K=1, which went from 0.82 to 0.95, shows that the model found the correct entity pair (matched nodes) 95% of the time. In the case of hits@5, where K=5, shows the top 5 predicted entity pairs for a given source object, the probability of finding the correct node pair went up to almost 100%. It is also observed that using all modalities at K=3 outperforms all the other combinations, as seen in fig. 6.1 Hence, enriching the graph certainly increased the graph alignment score. Despite all this improvement in alignment, this was the second-best performing model.

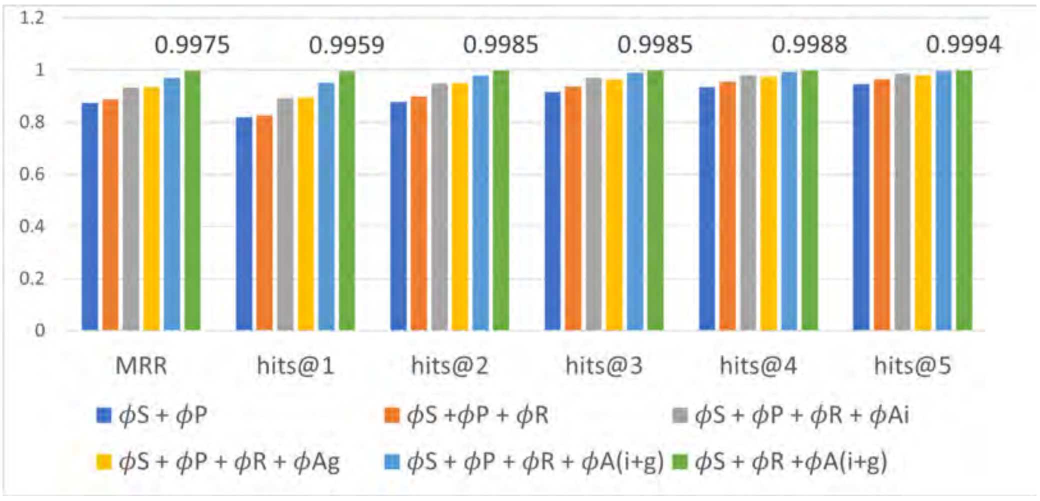


Figure 6.1: Architectural Data Alignment Metric Comparison

Another highlight of this thesis work is that, if geometric features are available then the point feature extractor module can be dropped with no drop in performance. This leads to significantly faster computation time as no points are processed during training or validation. So, for only graph-linking tasks geometric properties and object information regarding the relative position are enough to find node correspondences.

After the node correspondences are found, the source and reference points are required, but processing them is unnecessary. As no downsampling is done to the points, finding rigid transformation between the source and the target pcd might take extra time, but IR will increase as the points are densely packed.

The ablation study showed that in cases where the node semantics were changed, having no information about the object label is better than having wrong semantic labels in the validation set. If there's a scenario where semantic information is only available for the training set, and almost little or no semantic information is available for the validation set, then the study shows that it is better to exclude that information from everywhere and train the entity alignment module. Including semantic information in training but having wrong information in validation time shows a steep decrease in alignment score from 0.97 to 0.79, a drop of 18%.

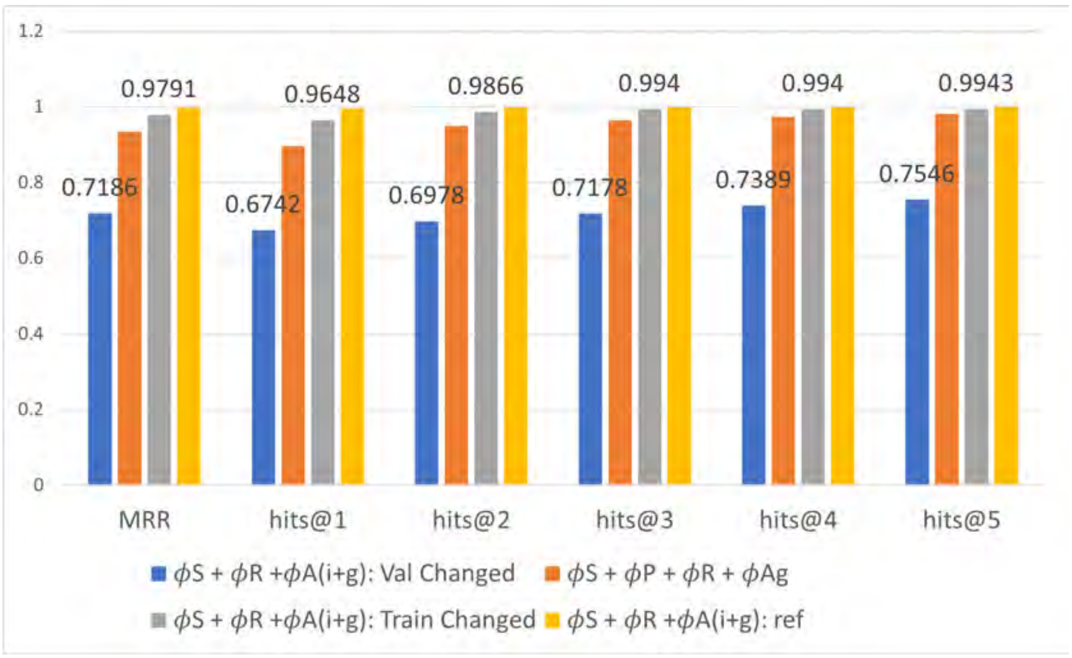


Figure 6.2: Node Semantic Changed, Alignment Metric Comparison, Architectural Data

Whereas, removing edges did not significantly affect the model's performance. In either case, where it was removed from the training or validation set, no more than 1% deviation from the best scenario was observed.

6.1.2 Architectural Data with Furniture Elements

In this scenario, along with architectural elements furniture elements were also present in both the training and validation set. To emulate real-world scenarios where the data streams are not of similar dimensions, a subscene and the scene where the subscene was derived from were taken as source and target. Here, the overlap is essentially 1. Also, two subscenes are not always aligned perfectly in the real world, so random transformations were added to the source, which randomly rotates between 1 and 7 degrees in all directions. In addition to that, random translation was applied as well. Subscenes with overlap greater than 0.1 were considered in this case.

Since randomly transformed subscenes were added, a drop in alignment metric is observed in the base model. A drop from 0.87 to 0.79 in the alignment metric is observed. However, similar to the previous section, as the complexity of the model increased, an increase in the model's overall performance was observed. The second-best model previously had an alignment score of 0.97 but in this scenario, an alignment metric of 0.93 was observed.

An alignment score of 0.97 is observed in the best-performing model as shown in fig. 6.3. The corresponding values are presented in fig. 6.3. Even after the transformed subscenes and full scenes were modeled in the data, TUMAligner showed that it could learn features and retain a high degree of alignment. Despite differences in datasets used, SARKAR et al., 2023 in their best-performing model had an alignment score of 0.95 which did not include any transformed subscenes. This showcases the potential of TUMAligner in scene alignment and downstream registration tasks.

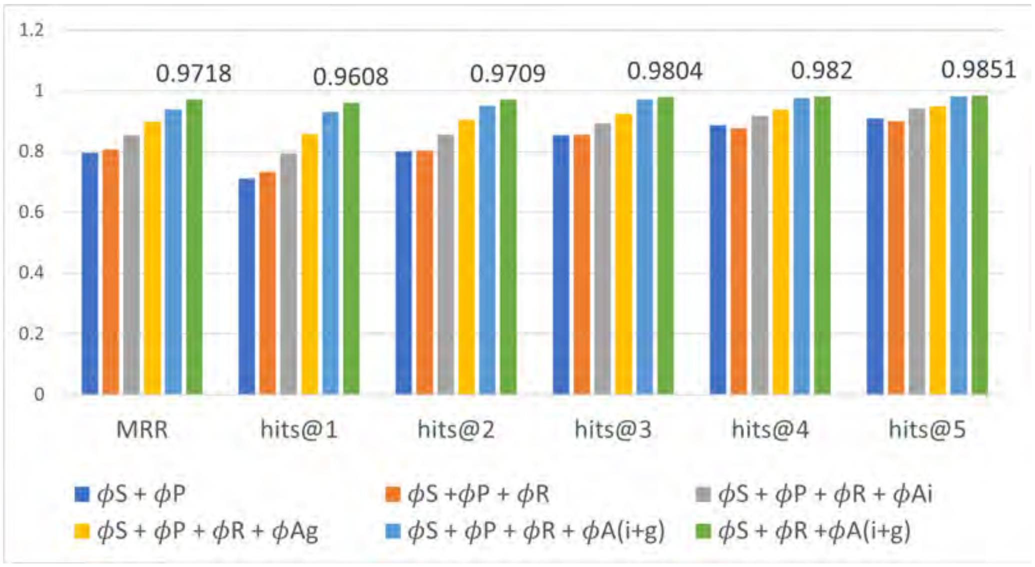


Figure 6.3: Furniture Data Alignment Metric Comparison

In the case of the ablation study, results similar to those of the previous section were observed. A drop of 36% in performance was observed when trying to validate the data with the wrong semantic level as seen in fig. 6.4, from a reference value of 0.97 to 0.62. The data labels are shown in the figure when node semantics were changed for train data as well as validation data. It can be seen that the model performs better when label information is not included in comparison to when wrong semantics information is added to the validation set. No significant changes were observed when edges and nodes were removed.

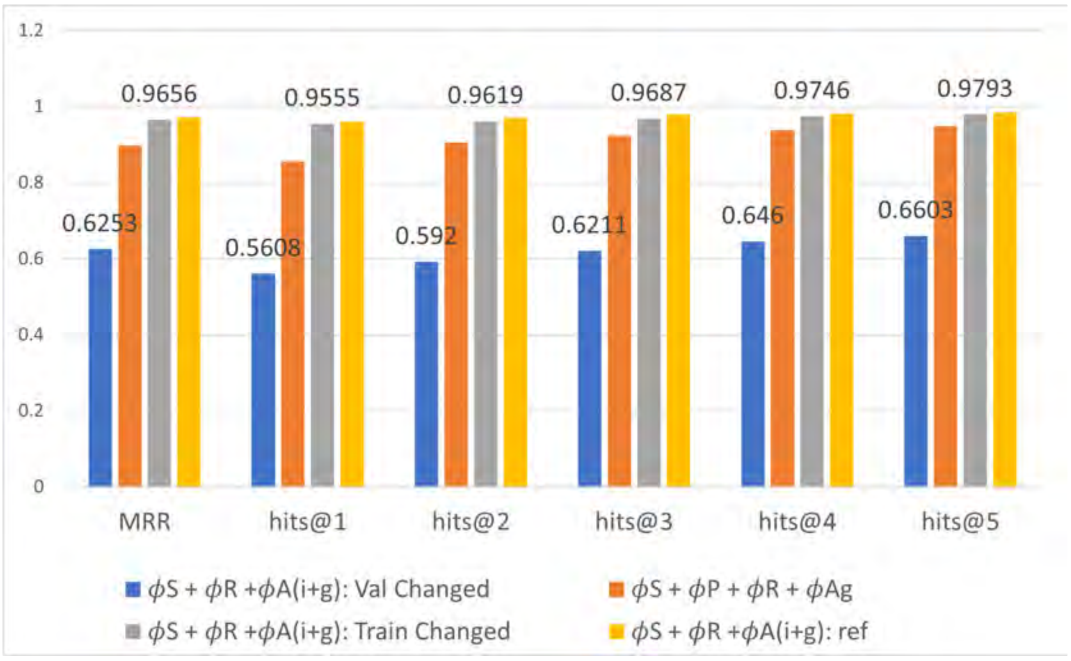


Figure 6.4: Node Semantic Changed, Alignment Metric Comparison

Interestingly, when nodes were removed randomly from the train set, an increase in the model's performance was observed. Even though the increase in performance is about 1%, it is hypothesized that this enhancement can be attributed to a phenomenon akin to dropout regularization.

Therefore, an experiment was conducted where a fixed number of nodes were removed for each subscene to see the change in MRR score. A similar experiment was done in the case of edges as well. The results are shown below.

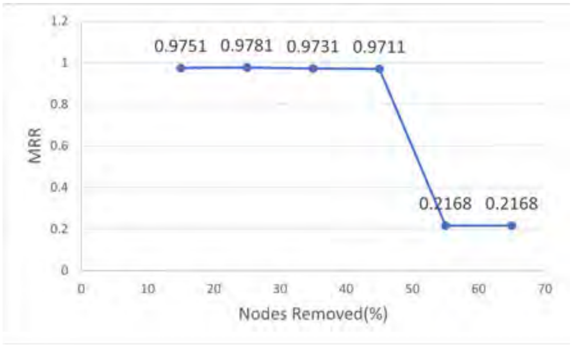


Figure 6.5: MRR vs % of Nodes Removed

In the case of node removal, a drop in performance is observed when at least 55% of the nodes were removed from each subscene. Whereas, in the case of edge removal, no effect on performance was seen even when removing 65% of the relationships, showcasing that the semantics of the relationship do not matter as long as there is a mechanism for information flow in the network.

6.1.3 Point Cloud and BIM

The alignment result obtained in table 5.30 shows the importance of a separate entity alignment module for BIM and point cloud.

A thing to consider is that the alignment result was obtained on one data instance. Even though the model's performance cannot be generalized after running an experiment on one instance, the low alignment metric emphasizes the need for a separate training module for such cases.

Due to low node correspondence, registration results could not be obtained.

6.1.4 Use Case

In this section, the use cases of the results are discussed.

6.1.4.1 Label Information

An ablation study regarding wrong label information was done to study the effects of noise on label accuracy. Labels on both PCD and BIM are obtained after passing the individual data sets through a **semantic segmentation model**. As deep learning models are not 100% accurate, noises are introduced in the labels. The model can assign a label to similar structures. For example, a wall and a slab generally have a similar structure from the models' point of view. So a wall might be labelled as slab and vice-versa. This was modeled by taking the most similar label in ifc2vec embedding space.

As previously stated, a drop of 35% was observed in the alignment score when wrong label information was incorporated into the validation set. Intriguingly, the model showcased robust performance even when wrong label information was incorporated into training. As a learnable parameter weights individual embeddings from modalities during training, the model learns to disregard the noisy inputs in the learning process.

Furthermore, it was observed that the models' performance was better when label information was omitted compared to instances where wrong information was introduced in the validation set.

6.1.4.2 Transformed Subscenes

The difficulties in integrating (aligning) two data streams were discussed in the chapter 1. They do not have the same coordinate system, and even if they do, due to scanning machine settings or some inherent error present, data streams do not perfectly align with each other. Some degree of transformation is present between source and reference data streams. This is why source scenes were transformed randomly, to mimic this transformation observed in real life. When such scenes were incorporated into the learning process, from table 5.25 it is observed that the alignment scores go down but when geometric features are incorporated along with it an alignment score of 0.97 is achieved.

This shows that the model learns even when scenes are transformed and can find accurate node correspondences, which is further used to find the transformation matrix between the source and the reference scene. This way the two data streams can be aligned with each other.

The main assumption is that data streams are already aligned to some degree, i.e., some degree of overlap between the scenes.

6.1.4.3 Fully Overlapping Scene

Even though scenes with an overlap of 0.1 and 0.9 were only selected in the first case which only had architectural elements. But in the case of elements with furniture, a scene, and its full scene were also incorporated where the scene overlaps fully with its full scene. The purpose was to study how well a small piece of a larger scene aligns.

6.1.4.4 Importance of adjacent elements

During the preprocessing phase, a link between all elements in a scene is created. This way the model learns to prioritize the links that bring the similar elements closer and push the dissimilar elements away. So, not knowing adjacent elements does not pose a disadvantage when training an entity learning module.

6.1.4.5 Importance of Semantics in Relationship

Table 5.25 shows that knowing the semantics of relationships between the elements does not significantly change the models' performance. MRR score went from 0.79 to 0.80 between model $\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$ and $\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}}$. Hits@K are very much comparable between the two models.

6.2 Limitations

The entity alignment module does not work if two similar scenes do not overlap at all. It is assumed that some degree of alignment is already present. Even if two of the same scenes are taken and a large amount transforms one, node alignment is not possible. This is one of the major limitations of this method. Another iterative process must be used first to bring the scenes together so there is some overlap.

Also, the two subscenes, that are to be aligned and registered need to be of similar size. As for registration, indexes of the common elements in both the subscenes are required, if one of the subscenes is very large compared to the other, $scene\_1/scene\_2 \rightarrow 0$ as scene_2 becomes larger and larger. This way the overlap tends to go to zero, and no common elements are found in the smaller scene and hence no alignment and registration will happen.

The relation semantic changer also does not work on objects not aligned with one of the principal axes. So if slanted roofs and windows are present, it will give the wrong relation semantics.

There are instances where the aligner does not produce registration results. The existence of the same exact object in multiple replicas can explain such failure. Even with instance-level geometric features, the registration cannot be recovered. When a scene is more descriptive, accurate matching is recovered.

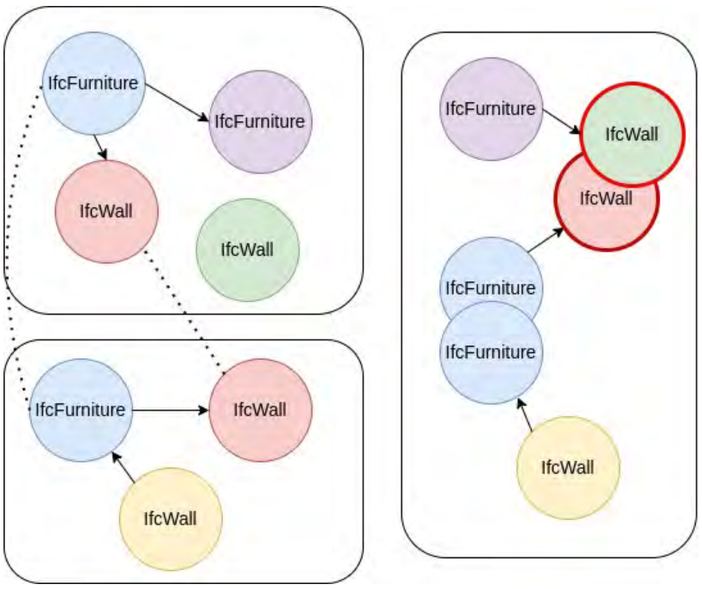


Figure 6.6: Failure Case

Information from BIM is extracted using IfcOpenshell. All the elements associated with a space are extracted and visualized in Solibri for verification. Usually in BIM models slab and wall geometry are modeled to span multiple rooms, which might or might not reflect the as-built span of the concrete slab. Even though entity alignment is possible during this case, this can be further broken down into slabs and walls of individual spaces for accurate alignment.

The BIM Graph formulation did not have relationship information. So relationship information from its point cloud counterpart was used. For objects belonging to BIM, it was checked if the corresponding object was in the point cloud and relationships were imported.

One major limitation was that only one data instance was used for BIM-point cloud. The approach was very manual as compared to point cloud-point cloud.

Chapter 7

Future Work and Conclusion

7.1 Future Work

The primary focus of this thesis work has been in the direction of integration of point cloud data. Even though different studies were conducted, there are venues where this work can be extended and improved. One interesting area would be the study of the removal of links that are created during the pre-processing phase. This area was not fully explored within the scope of this research. Different amounts of links can be removed and their effect on the MRR score can be studied.

Also for Graph Attention Network, no features along the links were defined. A separate relational encoder was used that defined the semantics of the links. An idea for future implementation would be to implement the semantic feature inside as the link feature in the network.

Another area that this thesis work does not explore is the scaling of the scenes. The point cloud-to-point cloud integration happened at the subscene level. It would be interesting to see how this module would on a formulation that is obtained at a floor level or multi-floor level.

Since point cloud integration happened at the subscene level, reconstructing the whole 3D scene from partial point clouds is another direction worth exploring. Essentially, studying the problem of 3D point cloud mosaicking.

Whereas, not much was explored in integrating BIM and point cloud. The BIM components like slab and wall span across multiple rooms so that they can be broken down into space levels and alignment can be studied. It would also be exciting to see if the point cloud-to-point cloud method would produce a better alignment score than what it currently outputs.

The methodology used in this work is very manual in the BIM-PCD direction. This is where further significant improvements can be made.

The alignment result showed the necessity of a separate entity alignment module, that captures the complexity of the data coming from BIM. A separate entity alignment module that works on BIM and PCD at subscene and scene levels can be a possible avenue for furthering this work. Then the scaling of this new integration method can be studied at a floor or multi-floor level.

7.2 Conclusion

This work mainly focused on the integration of point-cloud-to-point-cloud of a built reality between two different time steps, which has its use case in progress monitoring. Graph enriching methods were developed to enrich the point-cloud graph and to study its effect on the alignment task. The methods developed showed that feeding the information-rich graph to the Multi-Modal Encoder improved the model's performance. The current state-of-the-art SGAligner had the best alignment metric of 0.95 when used on normal scenes, whereas the method developed here even with transformed subscenes had an alignment metric of 0.97. Despite the differences in the datasets, the base-case model had an alignment metric of 0.79 whereas SGAligner's was 0.89. This bolsters the efficacy of TUMAligner.

One of the major findings is that pointnet feature encoder can be dropped without loss of performance. Out of the various enriching methods, geometric properties affected the alignment metric the most. A byproduct of this outcome was that the training and inference processing time was much faster as points did not have to be processed.

This research work shows that learning methods can be used to align 3D-built environments. However, it is important to note that while the methods developed demonstrate improvements in aligning and registering 3D-built environments, it is also essential to acknowledge the limitations of this approach. Some degree of overlap is assumed to be a pre-existing knowledge.

While the result of the trained module was inefficient in point-cloud-BIM integration, it certainly showed the need for its training pipeline. The limitations identified in this work pave the way for potential avenues of research.

The method developed showed that TUMAligner is robust to varying degrees of overlaps. It was also observed that aligning entities on the graph level can improve downstream tasks. Even though the research focused on built environments, the method and the improvement in registration show that the method can be leveraged for mapping, automation, robotics, and medical imaging.

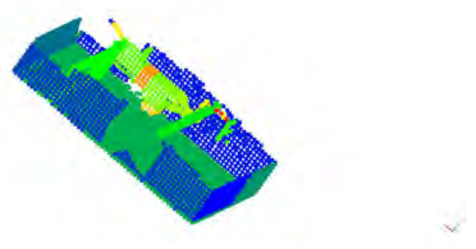
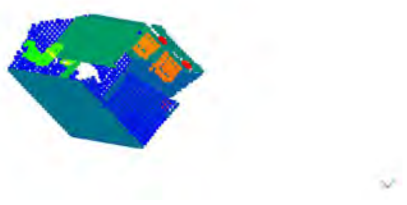
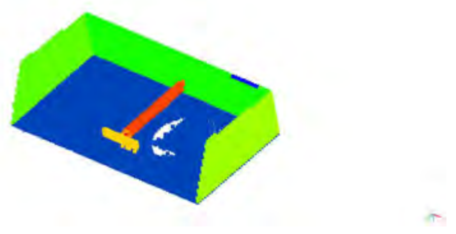
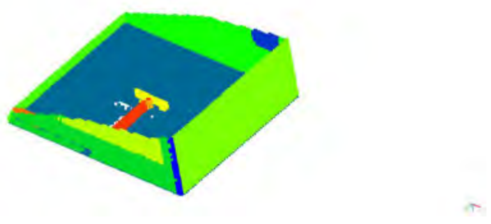
As decisions about the current state of the building are tied to integrating data from a previous model, this work serves as a link between the two. Leveraging the learning-based approach in built environments not only helps refine existing tools and methods but also opens up new avenues to add substantial value.

Appendix A

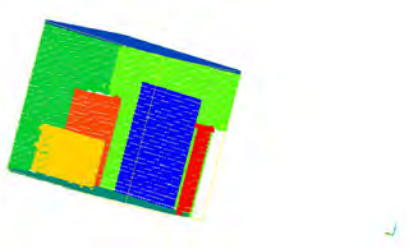
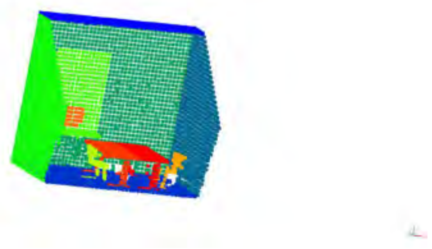
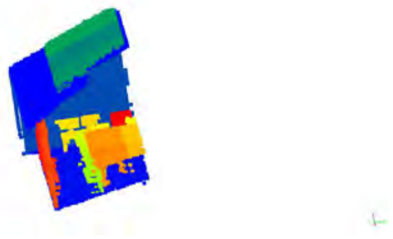
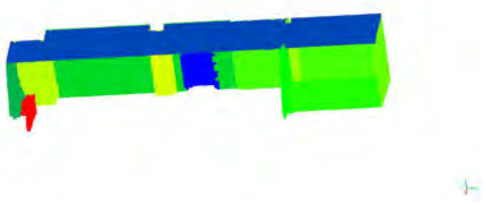
Dataset

A.1 Subscenes

A.1.1 Dataset without Furniture

A.1.2 Dataset With Furniture

Appendix B

Results

B.1 Architectural Data

B.1.1 Case 1: Structure Encoder and Object Encoder

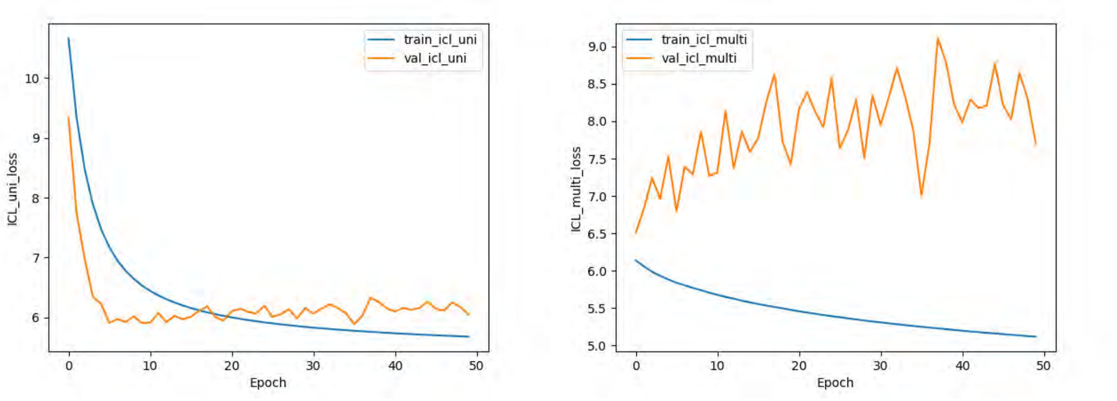


Figure B.1: ICL uni-modal and multi-modal Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$)

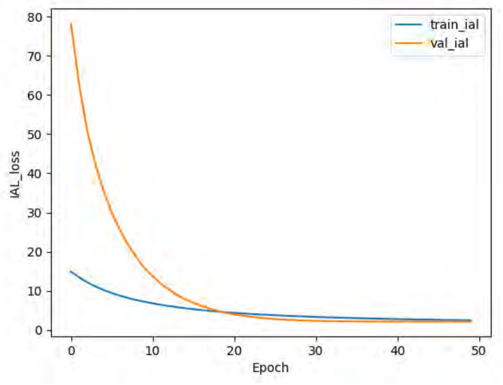
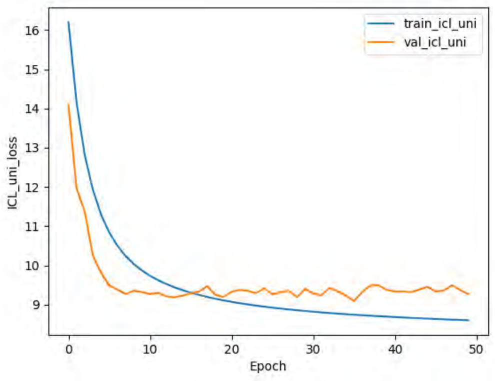


Figure B.2: IAL Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}}$)

B.1.2 Case 2: Structure, Object and Relational Encoder



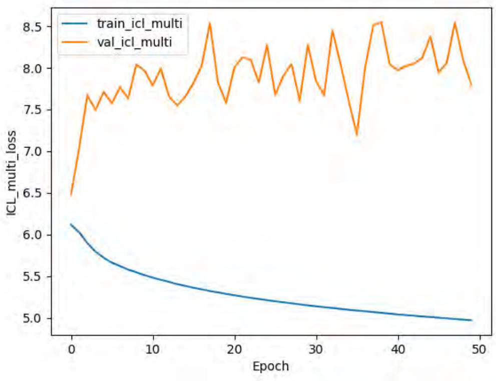


Figure B.3: ICL uni-modal and multi-modal Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}}$)

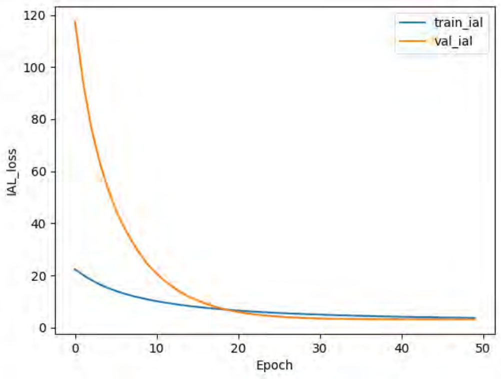
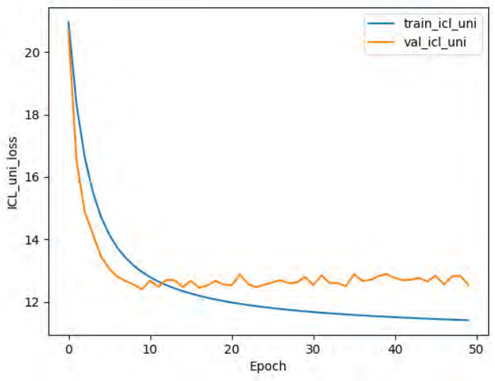


Figure B.4: IAL Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}}$)

B.1.3 Case 3: Structure, Object, Relational and Attribute(ifc2vec) Encoder



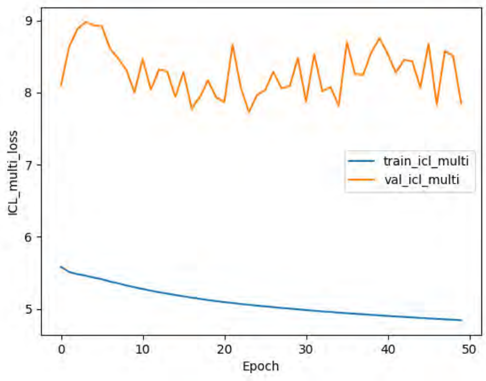


Figure B.5: ICL uni-modal and multi-modal Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_i^{\mathcal{A}}$)

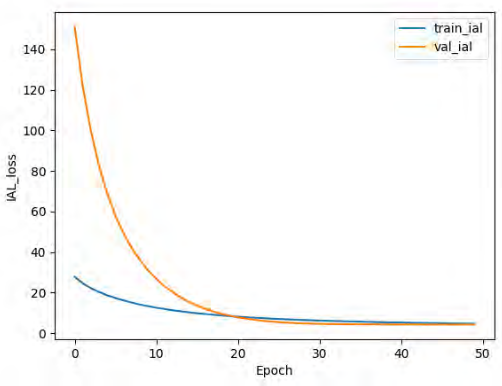


Figure B.6: IAL Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_i^{\mathcal{A}}$)

B.1.4 Case 4: Structure, Object, Relational and Attribute(geometric) Encoder

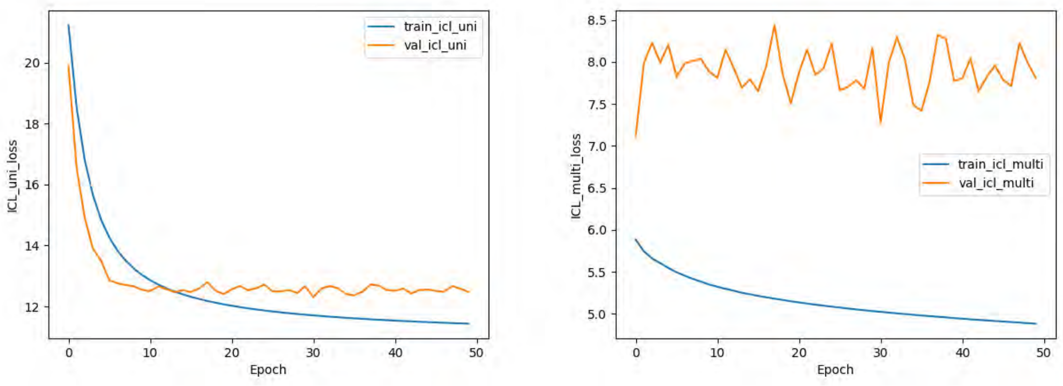


Figure B.7: ICL uni-modal and multi-modal Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_g^{\mathcal{A}}$)

0.5

Figure B.8: IAL Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi_g^{\mathcal{A}}$)

B.1.5 Case 5: Structure, Object, Relational and Attribute(combined) Encoder

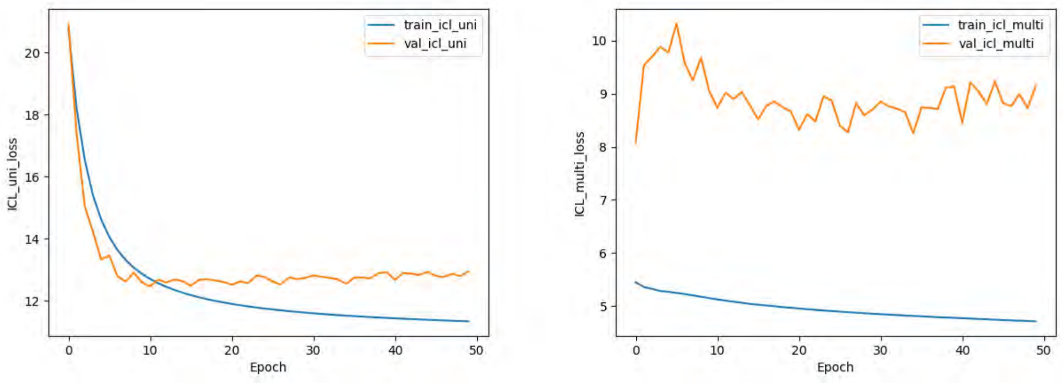


Figure B.9: ICL uni-modal and multi-modal Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$)

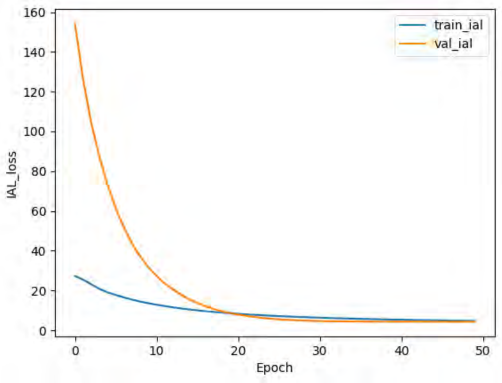


Figure B.10: IAL Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{P}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$)

B.1.6 Case 6: Structure, Relational and Attribute(combined) Encoder

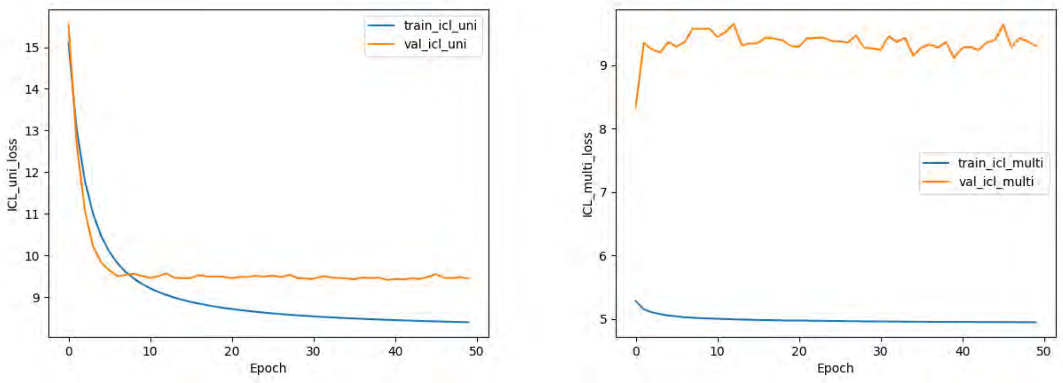


Figure B.11: ICL uni-modal and multi-modal Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$)

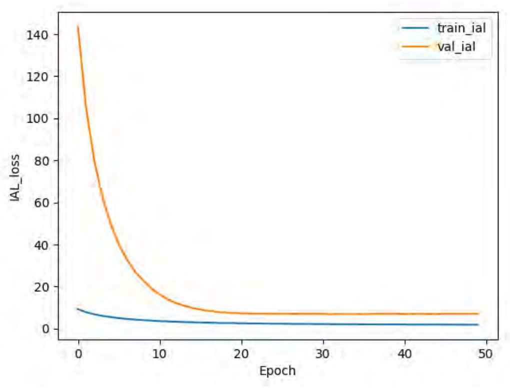
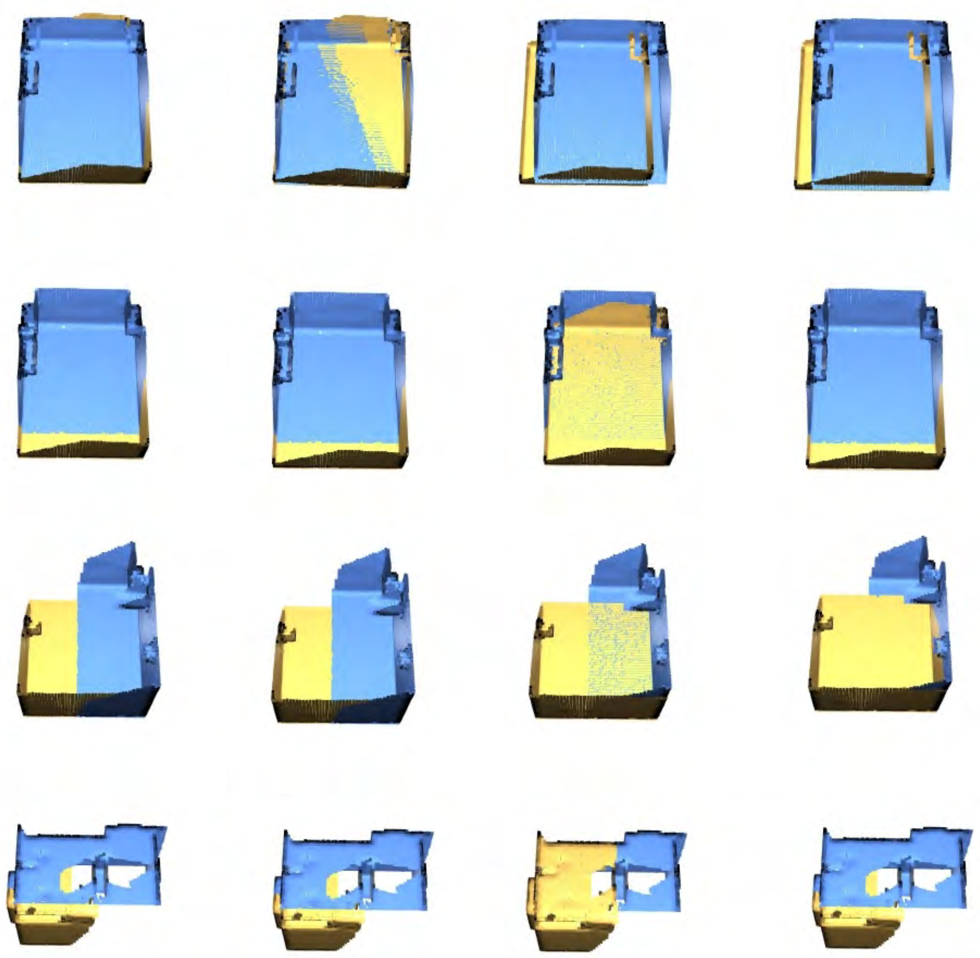


Figure B.12: IAL Loss ($\phi^{\mathcal{S}} + \phi^{\mathcal{R}} + \phi^{\mathcal{A}}_{i+g}$)

B.1.6.1 Registration Results

The order of Images are

a) Ground Truth b) Transformation_applied c) GeoTransformer d) TUMAligner



Bibliography

MAIMAN, T. (1960). Stimulated optical radiation in ruby. *Nature*.

ARMENI, I., HE, Z.-Y., GWAK, J., ZAMIR, A. R., FISCHER, M., MALIK, J., & SAVARESE, S. (2019). 3d scene graph: A structure for unified semantics, 3d space, and camera.

BEBIS, G., & GEORGIOPOULOS, M. (1994). Feed-forward neural networks. *IEEE Potentials*, *13*(4), 27–31. https://doi.org/10.1109/45.329294

BENDER, E. A., & WILLIAMSON, S. G. (2010). *Lists, decisions and graphs. with an introduction to probability*.

BERRENDORF, M., FAERMAN, E., MELNYCHUK, V., TRESP, V., & SEIDL, T. (2020). Knowledge graph entity alignment with graph convolutional networks: Lessons learned. *Advances in Information Retrieval*.

BESL, P. J., & MCKAY, N. D. (1992). Method for registration of 3-d shapes. *Sensor Fusion IV: Control Paradigms and Data Structures*, *1611*, 586–606.

BRAUN, A., TUTTAS, S., STILLA, U., & BORRMANN, A. (2018). Bim-based progress monitoring. In A. BORRMANN, M. KÖNIG, C. KOCH, & J. BEETZ (Eds.), *Building information modeling - technology foundations and industry practice* (pp. 463–476). Springer. https://doi.org/10.1007/978-3-319-92862-3_28

BRAUN, A., TUTTAS, S., BORRMANN, A., & STILLA, U. (2020). Improving progress monitoring by fusing point clouds, semantic data and computer vision. *Automation in Construction*, *116*, 103210. https://doi.org/https://doi.org/10.1016/j.autcon.2020.103210

Building information modeling: Technology foundations and industry practice. (2018). Springer International Publishing. https://doi.org/10.1007/978-3-319-92862-3

CHEN, L., LI, Z., WANG, Y., XU, T., WANG, Z., & CHEN, E. (2020). Mmea: Entity alignment for multi-modal knowledge graph. *Knowledge Science, Engineering and Management: 13th International Conference, KSEM 2020, Hangzhou, China, August 28–30, 2020, Proceedings, Part I*, *13*, 134–147. https://arxiv.org/abs/2209.06111

CHERNOSKUTOV, M. (2021). Data structure for faster graph processing. *2021 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, 0297–0300. https://doi.org/10.1109/USBEREIT51232.2021.9454964

CHUANG, T., & YANG, M. (2023). Change component identification of bim models for facility management based on time-variant bims or point clouds [Publisher Copyright: © 2022 Elsevier B.V.]. *Automation in construction*, *147*. https://doi.org/10.1016/j.autcon.2022.104731

COLLINS, F., MAFIPOUR, M., NOICHL, F., PAN, Y., & VEGA TORRES, M. A. (2021). Towards applicable scan-to-bim and scan-to-floorplan: An end-to-end experiment. In *Proc. of the 32th forum bauinformatik*. https://doi.org/10.26083/tuprints-00019496

COLLINS, F., BRAUN, A., & BORRMANN, A. (2022). Finding geometric and topological similarities in building elements for large-scale pose updates in scan-vs-bim. *Proc. of the Int. Conf. on Computing in Civil and Building Engineering (ICCCBE)*.

COLLINS, F., BRAUN, A., & BORRMANN, A. (2023). Graph-based linking of point cloud and bim elements for extended dt integration. In *Proc. of the 30th int. conference on intelligent computing in engineering (eg-ice)*.

DAUM, S., & BORRMANN, A. (2014). Processing of topological bim queries using boundary representation based methods. *Advanced Engineering Informatics*, *28*(4), 272–286. https://doi.org/https://doi.org/10.1016/j.aei.2014.06.001

DERPANIS, K. G. (2010). *Overview of the ransac algorithm*.

DROBNYI, V., LI, S., & BRILAKIS, I. (2024). Connectivity detection for automatic construction of building geometric digital twins. *Automation in Construction*, *159*, 105281. https://doi.org/https://doi.org/10.1016/j.autcon.2024.105281

EASTMAN, C. (2009). What is bim?

FISCHLER, M. A., & BOLLES, R. C. (1987). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision* (pp. 726–740). Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-08-051581-6.50070-2

GRIEVES, M. (2015). Digital twin: Manufacturing excellence through virtual factory replication.

GUO, H., TANG, J., ZENG, W., ZHAO, X., & LIU, L. (2021). Multi-modal entity alignment in hyperbolic space. *Neurocomputing*, *461*, 598–607. https://doi.org/https://doi.org/10.1016/j.neucom.2021.03.132

HADSELL, R., CHOPRA, S., & LECUN, Y. (2006). Dimensionality reduction by learning an invariant mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, *2*, 1735–1742. https://doi.org/10.1109/CVPR.2006.100

HAMILTON, W. L. (2020). *Graph representation learning* (Vol. 14).

HINTON, G., VINYALS, O., & DEAN, J. (2015). Distilling the knowledge in a neural network.

HOGAN, A., BLOMQVIST, E., COCHEZ, M., D'AMATO, C., MELO, G. D., GUTIERREZ, C., KIRRANE, S., GAYO, J. E. L., NAVIGLI, R., NEUMAIER, S., NGOMO, A.-C. N., POLLERES, A., RASHID, S. M., RULA, A., SCHMELZEISEN, L., SEQUEDA, J., STAAB, S., & ZIMMERMANN, A. (2021). Knowledge graphs. *ACM Computing Surveys*, *54*(4), 1–37. https://doi.org/10.1145/3447772

HU, Z., & BRILAKIS, I. (2024). Matching design-intent planar, curved, and linear structural instances in point clouds. *Automation in Construction*, *158*, 105219. https://doi.org/https://doi.org/10.1016/j.autcon.2023.105219

KAMINSKY, R. S., SNAVELY, N., SEITZ, S. M., & SZELISKI, R. (2009). Alignment of 3d point clouds to overhead images. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 63–70. https://doi.org/10.1109/CVPRW.2009.5204180

KAYHANI, N., MCCABE, B., & SANKARAN, B. (2023). Semantic-aware quality assessment of building elements using graph neural networks. *Automation in Construction*, *155*, 105054. https://doi.org/https://doi.org/10.1016/j.autcon.2023.105054

KIM, U.-H., PARK, J.-M., SONG, T.-j., & KIM, J.-H. (2020). 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE*

Transactions on Cybernetics, *50*(12), 4921–4933. https://doi.org/10.1109/tcyb.2019.2931042

Kipf, T. (2016). *Graph convolutional networks*. https://tkipf.github.io/graph-convolutional-networks

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks.

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital twin in manufacturing: A categorical literature review and classification [16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018]. *IFAC-PapersOnLine*, *51*(11), 1016–1022. https://doi.org/https://doi.org/10.1016/j.ifacol.2018.08.474

Landrieu, L., & Simonovsky, M. (2018a). Large-scale point cloud semantic segmentation with superpoint graphs.

Landrieu, L., & Simonovsky, M. (2018b). Large-scale point cloud semantic segmentation with superpoint graphs.

Lee, G., Sacks, R., & Eastman, C. M. (2006). Specifying parametric building object behavior (bob) for a building information modeling system [Knowledge Enabled Information System Applications in Construction]. *Automation in Construction*, *15*(6), 758–776. https://doi.org/https://doi.org/10.1016/j.autcon.2005.09.009

Li, L., Wang, R., & Zhang, X. (2021). A tutorial review on point cloud registrations: Principle, classification, comparison, and technology challenges. *Mathematical Problems in Engineering*, *2021*, 9953910. https://doi.org/10.1155/2021/9953910

Lin, Z., Zhang, Z., Wang, M., Shi, Y., Wu, X., & Zheng, Y. (2022). Multi-modal contrastive representation learning for entity alignment.

Liu, F., Chen, M., Roth, D., & Collier, N. (2021). Visual pivoting for (unsupervised) entity alignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(5), 4257–4266.

Martens, J., & Blankenbach, J. (2018). An automated approach for point cloud alignment based on density histograms. *Intelligent Computing in Engineering and Architecture*. https://api.semanticscholar.org/CorpusID:197537469

Maturana, D., & Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 922–928. https://doi.org/10.1109/IROS.2015.7353481

Meyer, T., Brunn, A., & Stilla, U. (2022). Change detection for indoor construction progress monitoring based on bim, point clouds and uncertainties. *Automation in Construction*, *141*. https://doi.org/https://doi.org/10.1016/j.autcon.2022.104442

Myronenko, A., & Song, X. (2010). Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(12), 2262–2275. https://doi.org/10.1109/TPAMI.2010.46

Pan, Y., Yang, B., Liang, F., & Dong, Z. (2018). Iterative global similarity points : A robust coarse-to-fine integration solution for pairwise 3d point cloud registration.

Pomerleau, F., Colas, F., & Siegwart, R. (2015). A review of point cloud registration algorithms for mobile robotics [ffhal-01178661f]. *Foundations and Trends in Robotics*, *4*(1), 1–104. https://doi.org/10.1561/2300000035

Qader, W. A., Ameen, M. M., & Ahmed, B. I. (2019). An overview of bag of words;importance, implementation, applications, and challenges. *2019 International Engineering Conference (IEC)*, 200–204. https://doi.org/10.1109/IEC47844.2019.8950616

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation.

Qiao, Z., Yu, Z., Yin, H., & Shen, S. (2023). Pyramid semantic graph-based global point cloud registration with low overlap.

Qin, Z., Yu, H., Wang, C., Guo, Y., Peng, Y., & Xu, K. (2022). Geometric transformer for fast and robust point cloud registration.

Rausch, C., & Haas, C. (2021). Automated shape and pose updating of building information model elements from 3d point clouds. *Automation in Construction*, *124*, 103561. https://doi.org/10.1016/j.autcon.2021.103561

Sacks, R., Girolami, M., & Brilakis, I. (2020). Building information modelling, artificial intelligence and construction tech. *Developments in the Built Environment*, *4*, 100011. https://doi.org/10.1016/j.dibe.2020.100011

Sarkar, S. D., Miksik, O., Pollefeys, M., Barath, D., & Armeni, I. (2023). Sgaligner : 3d scene alignment with scene graphs.

Sarlin, P.-E., DeTone, D., Malisiewicz, T., & Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks.

Sarode, V., Li, X., Goforth, H., Aoki, Y., Dhagat, A., Srivatsan, R. A., Lucey, S., & Choset, H. (2019). One framework to register them all: Pointnet encoding for point cloud alignment.

Singhal, A. (2012). *Introducing the knowledge graph: Things, not strings* [Google Blog]. https://www.blog.google/products/search/introducing-knowledge-graph-things-not/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks.

Wald, J., Dhamo, H., Navab, N., & Tombari, F. (2020). Learning 3d semantic scene graphs from 3d indoor reconstructions. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, W., Wang, T., & Cai, Y. (2022). Multi-view attention-convolution pooling network for 3d point cloud classification. *Applied Intelligence*, *52*, 14787–14798. https://doi.org/10.1007/s10489-021-02840-2

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2018). Dynamic graph CNN for learning on point clouds. *CoRR*, *abs/1801.07829*. http://arxiv.org/abs/1801.07829

Wang, Z., Lv, Q., Lan, X., & Zhang, Y. (2018). Cross-lingual knowledge graph alignment via graph convolutional networks. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 349–357). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1032

Yew, Z. J., & Lee, G. H. (2020). Rpm-net: Robust point matching using learned features. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11821–11830. https://doi.org/10.1109/CVPR42600.2020.01184

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57–81. https://doi.org/https://doi.org/10.1016/j.aiopen.2021.01.001

Zhou, Q.-Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.

Zhou, Q., Shen, Z., Xu, X., Zhi, P., & Zhao, R. (2022). Theories and practices of self-driving vehicles. In Q. Zhou, Z. Shen, B. Yong, R. Zhao, & P. Zhi (Eds.), *Theories and practices of self-driving vehicles* (pp. 27–62). Elsevier. https://doi.org/https://doi.org/10.1016/B978-0-323-99448-4.00002-3

Zhu, B., Liu, X., Mao, X., Chen, Z., Guo, L., Gui, T., & Zhang, Q. (2023). Universal multi-modal entity alignment via iteratively fusing modality similarity paths.

Declaration

I hereby affirm that I have independently written the thesis submitted by me and have not used any sources or aids other than those indicated.

Munich, 26.01.2024, [illegible]

Location, Date, Signature