Merging of Topometric Maps of Indoor Environments

Extraction, matching and fusion

Maximiliaan van Schendel
student #4384644
m.vanschendel@tudelft.nl

1st supervisor: Edward Verbree
2nd supervisor: Pirouz Nourian
external supervisor: Robert Voûte

31/10/2022

Contents

abstract The merging of multiple partial maps of indoor environments created by teams of human or robot agents into a single global map is a key problem that, when solved, can improve mapping speed and quality. Existing map merging approaches generally depend on external signals which are not available indoors or only use the geometric properties of an environment. Inspired by the human understanding of environments in relationship to their context we propose a map merging system that extracts and uses topometric maps, a map representation containing both the geometric and topological characteristics of an environments, to solve the map merging problem in indoor spaces. In this research we demonstrate an intuitive approach to extracting topometric maps of 3D, multi-floor, indoor environments and use both the topological and geometric characteristics contained in the topometric maps to perform context-aware map matching and fusion.

Glossary

global descriptor An n-dimensional vector representing the properties of a whole point cloud which can be used to compare the similarity of point clouds.. 8

global map A single, complete map constructed by merging multiple partial maps.. 4

local descriptor An n-dimensional vector representing the properties of a single point in a point cloud that can be used to find corresponding points between point clouds.. 8

map A symbolic representation of an environment which contains information about its characteristics.. 1, 4

map extraction The problem of converting one map representation to another. For the purposes of this research this mostly refers to the extraction of topometric maps from voxel grids.. 5

map fusion The problem of combining multiple partial maps into a single global map based on their overlapping areas.. 4

map matching The problem of identifying overlapping areas between partial maps.. 4

map merging The problem of identifying overlapping areas between partial maps and using these to combine the partial maps into a global map.. 4

partial map A collection of maps without a common coordinate frame that each represent a part of the environment.. 4

point cloud An unordered collection of points representing the geometry of an object or environment in 3D euclidean space (Volodine, 2007).. 5

registration The problem of finding a transformation that optimally aligns two point clouds.. 41

topological map Topological maps are a graph representation of an environment's structure, where vertices represent locally distinctive places, often rooms, and edges represent traversable paths between them (Thrun, 1998; Kuipers & Byun, 1988). Topological maps are inspired by the fact that humans are capable of spatial learning despite limited sensory and processing capability and only having partial knowledge of the environment. This is based on observations that cognitive maps, the mental maps used by humans to navigate within an environment, consist of multiple layers with a topological description of the environment being a fundamental component (Kuipers & Byun, 1988).. 20

topometric map A hybrid map representation combining both the topological and metric characteristics of the environment. This map representation allows the end-user to use either topological or metric information depending on the needs of the situation, e.g. the topological layer can be used for large-scale navigation and abstract reasoning while the metric layer can be used for landmark detection or obstacle avoidance.. 5

voxel grid Also known as an occupancy grid, a voxel grid is a "multi-dimensional (typically 2D or 3D) tesselation of space into cells, where each cell stores a probabilistic estimate of its state." (Elfes, 1990). In practice this probabilistic estimate is often a binary value that represents whether a cell is occupied or not.. 16

1 Introduction

3D maps of indoor environments are used for a wide range of applications, ranging from domestic to industrial, and may be used to provide comprehension of the environment to humans through cartography or for automated scene understanding that enables complex robot behaviour (Chen & Clarke, 2020; C. Wang et al., 2019; Hermann et al., 2016). In many cases it is advantageous for multiple mapping agents to collaboratively create a map, such as during rescue operations where the location of the subject(s) is unknown (Queralta et al., 2020). By working together larger areas can be mapped in a shorter amount of time (Lajoie et al., 2022). Furthermore, each agent may perceive the environment differently, resulting in a more complete whole (Schuster et al., 2020).

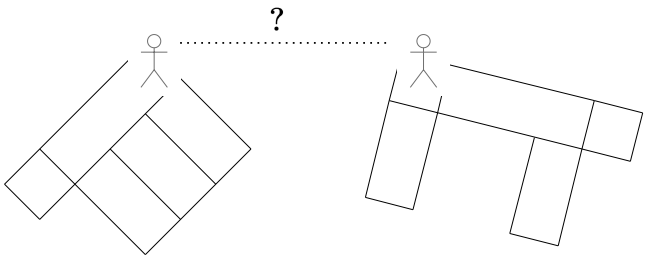

Figure 1: The map merging problem. How do you combine partial maps if relative positions and orientations are unknown?

The problem of creating a single global map from multiple partial maps is called map merging. Map merging is challenging because the relative positions and orientations (and sometimes scale) of agents within the environment are unknown. In indoor environments where external positioning signals like GNSS are highly attenuated map merging can only rely on the properties of the partial maps themselves. Figure 1 illustrates the map merging problem, showing two agents with unknown relative positions and orientations within the same environment. Map merging can be subdivided into two subproblems. The first is map matching, the identification of overlapping areas between partial maps. The second is map fusion, the alignment and combination of the partial maps based on the overlapping areas.

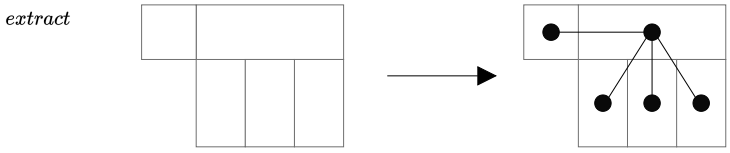

Figure 2: Extraction of topometric maps from raw data.

Map matching is essentially a problem of place recognition, or identifying

the same place in different maps despite differences in appearance. Human place recognition uses a combination of the visible properties of that place and the structure, or topology, of its environment (Kuipers, 1978). To illustrate, when asked to describe a room, someone may answer that it has an L-shape but also that the room has two neighbouring rooms. In this thesis we propose an approach that uses both the geometric and the topological properties of indoor environments to solve the map matching problem. For this purpose we also propose an approach for extracting topometric maps, which represent both the environment's geometry and its room-level topological structure, from purely geometric point cloud maps. We further propose an approach to fuse the geometry and topology of the partial topometric maps into a global topometric map based on the identified matches. Figures 2, 3 and 4 respectively illustrate the problems of map extraction, matching and fusion.

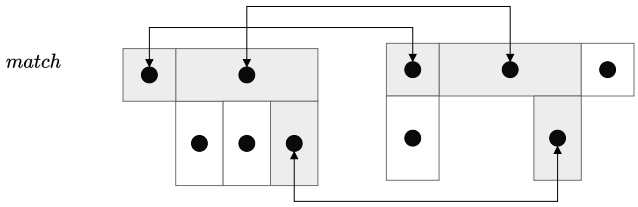

Figure 3: Map matching using both geometric and topological properties.

We hypothesize that the topological structure of rooms within the environment are consistently identifiable between partial maps. We further hypothesize that using the metric characteristics of the environment in conjunction with its topological characteristics will improve identification of overlapping areas over a purely geometric approach. Finally, we hypothesize that the identified matches can be used to fuse both the geometry and topology of the partial topometric maps into a global topometric map.

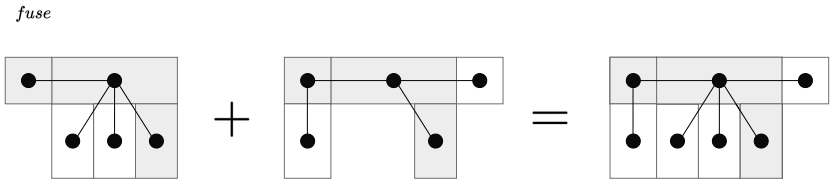

Figure 4: Fusion of partial topometric maps into a global topometric map.

The most important contributions of our work are:

1. Map extraction of 3D topometric maps from point clouds.
2. Map matching using both the geometry and topology of topometric maps.
3. Map fusion of topometric maps.

2 Research Questions

2.1 Main question

How can we apply topometric representations of indoor environments to solve the map merging problem?

2.2 Subquestions

1. In what way can partial topometric maps be extracted from partial point cloud maps?
2. What approach is best suited for identifying matches between partial topometric maps?
3. How can the identified matches be used to fuse two or more partial topometric maps into a global topometric map?

2.3 Scope

During this thesis the following will be created.

1. A program that is capable of topometric map extraction from point clouds and topometric map merging.
2. An analysis of the program's performance on publically available standard datasets.
3. Reports containing documentation and background research.

To better delineate the scope of the thesis we provide several aspects that will **not** be researched or discussed.

1. Map merging using known relative poses between agents or meeting strategies. Agent behaviour is assumed to be independent and agents are not able to sense eachother.
2. Map merging using observations unrelated to the environment's geometric or topological characteristics. E.g. the environment's colour or actively transmitted beacon signals.
3. Map merging assisted by a priori knowledge of the environment. E.g. building information models or floor plans.
4. Map merging using the pose graphs of agents. Agent poses are assumed to be unknown.
5. Achieving near real-time performance.

3 Related work

Previous research on mapping and map merging has considered various map representations (Tomatis et al., 2003; Huang & Beevers, 2005; Bonanni et al., 2017; Gholami Shahbandi & Magnusson, 2019). According to Andersone (2019) and Yu et al. (2020) these representations can be subdivided into three types: metric-, feature-, and topological maps. Hybrid maps that are combinations of two or more map types also exist, such as topometric maps, which are a combination of metric and topological maps (Yu et al., 2020). Map types that are not one of the three main types or a hybrid are rarely used but do exist (Yu et al., 2020). In this section we will discuss the work that has been done on extracting and merging metric and topometric maps. Figure 5 shows a diagram of the fields of research that are relevant for this thesis.

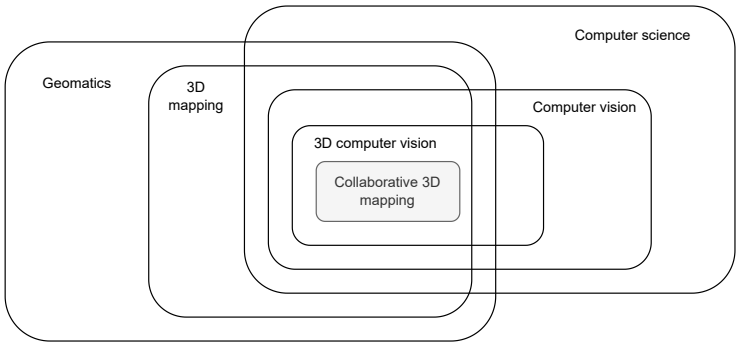

Figure 5: Euler diagram showing the overlapping fields of research that are relevant for this thesis.

3.1 Metric Maps

In this section we give an overview of the existing research into metric map extraction and map merging.

3.1.1 Metric map extraction

Metric maps are a map representation that represent the geometry of an environment. Two common metric map types that are relevant for our research are point clouds and voxel grids. Point clouds are usually the direct output of 3D mapping sensors and algorithms so it is not necessary to extract them from another map representation (Rusu & Cousins, 2011). Volodine (2007) gives an overview of point clouds and how to process them. Elfes (1990) gives a description of voxel grids and how to extract them from point cloud maps.

3.1.2 Metric map merging

Metric map matching is the problem of recognizing overlapping areas between partial maps based purely on their geometry. Metric map matching is a mature area of research that has applications for 3D mapping and place recognition. Some approaches use local descriptors that describe the properties of each point in the point cloud to identify corresponding points between point clouds. Descriptors may include corners, lines, planes or other points of interest, e.g. SIFT, SURF, FPFH or Harris points (Andersone, 2019; Rusu et al., 2009). Some descriptors are better suited for different kinds of input data. For example, the descriptor approach by Li & Olson (2010) is scale-independent and the approach of Yang et al. (2016) is able to deal with differences in resolution. Recent research into using deep learning for local descriptor extraction has also shown great potential, examples include PointNet and DGCNN (Qi et al., 2017; Phan et al., 2018).

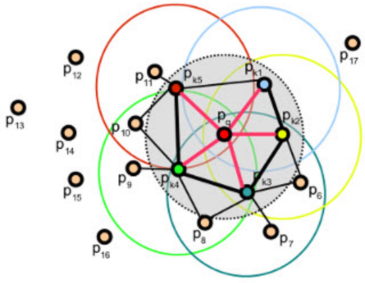

Figure 6: Illustration of FPFH local descriptor (Rusu et al., 2009).

Another approach to metric map matching uses global descriptors that describe the properties of segments of the point cloud, overlapping segments can then be identified based on the similarity of their global descriptor (Dubé et al., 2017). A large number of global descriptors have been proposed which describe the point cloud based on various properties, such as its volume, planarity or roughness (Han et al., 2018). Another approach to global descriptor extraction aggregrates the local descriptor of each point into a global descriptor, examples include DBoW and VLAD (Shan et al., 2021; Arandjelovic & Zisserman, 2013). Global descriptors can also be based on the spectral characteristics of graphs derived from the point cloud, including approaches such as ShapeDNA and heat kernel signatures (Reuter et al., 2006; Bronstein & Kokkinos, 2010). An advantage of these approaches is that they do not require manual selection of relevant descriptors. Finally, recent research has shown that deep learning can be used to extract global descriptors, with approaches such as PointNetVLAD, LPDNet and MinkLoc3D (Uy & Lee, 2018; Z. Liu et al., 2019; Komorowski, 2021). Although these approaches claim to have better resilience against differences between partial maps than other approaches they often require large

amounts of training data and may not be able to deal with environments that are not similar to the training data. Note that the above deep learning approaches often use a trainable variation of the local descriptor aggregration described above (Arandjelovic et al., 2016).

Metric map fusion is a mature area of research and various approaches have been proposed. The problem of metric map fusion comes down to finding a transformation between partial maps that brings the geometry of their overlapping areas into alignment, this is called registration. Most approaches to registration include a variation of the iterative closest point (ICP) algorithm. This algorithm finds the transformation between two point clouds by iteratively applying rigid transformations that minimize the distance between the points in one point clouds and their closest points in the other (Rusinkiewicz & Levoy, 2001). Since its first introduction a number of variations on the ICP algorithm have been proposed that improve its accuracy and performance. An example of this is the normal iterative closest point algorithm, which improves data-association by taking into account the normal vector of a point's neighbourhood and the gravity-aligned ICP algorithm, which constrains the transformation to use with maps that have a consistent up direction (Serafin & Grisetti, 2015; Kubelka et al., 2022).

The ICP algorithm and its variations are not guarantueed to find a globally optimal transformation and they are sensitive to the initial transformation between point clouds. To mitigate this Yang et al. (2016) proposes a two step algorithm that first performs a rough, global registration followed by a precise, local registration (see figure 7). The function of the global registration step is to find a good initialization that will allow the local registration step to find the global optimum. Various approaches to global registration have been proposed, with most depending on the detection of correspondences between point clouds using one of the descriptor approaches described above. A global transformation can then be estimated using RANSAC (Koguciuk, 2017).

Figure 7: Illustration of two-step approach to registration (Yang et al., 2016).

More recently, deep learning approaches for point cloud registration have been proposed, such as PointNetLK and DCP (Aoki et al., 2019; Y. Wang & Solomon, 2019). However, these methods require training and their results can not be constrained without retraining the model.

3.2 Topometric Maps

In this section we discuss the existing research into the extraction and merging of topometric maps.

3.2.1 Topometric map extraction

Various approaches for extracting topological maps from raw sensor data or metric maps have been proposed. Bormann et al. (2016) and Pintore et al. (2020) both give a review of different methods for extracting structured maps of indoor spaces in which different room segmentation methods are discussed and compared.

Kuipers & Byun (1991) first proposes identifying distinctive places, nodes of the topological map, directly from sensor data by finding local maxima of a distinctiveness measure within a neighbourhood. Its edges are identified by having the robot try to move between the identified nodes, if this is possible an edge is created. Note that this approach is dependent on the mapping agent's exploration strategy.

Thrun (1998) extracts a 2D topological map from a 2D metric map by identifying narrow passages. They then partition the metric map into areas divided by passages which are respectively the vertices and the edges of the graph. This approach is not able to deal with 3D environments with multiple storeys and it assumes that rooms are always separated by narrow passages.

Mura et al. (2014) proposes an approach to room segmentation that divides the ground plane of the environment into polygons which are then iteratively clustered to form rooms. Mura et al. (2016) avoids this clustering step by directly clustering scanner positions using a graph clustering approach. This approach is then further improved by Ambrus et al. (2017) by generating synthetic scanning positions along the medial axis of the map. Figure 8 illustrates the concept behind visibility clustering-based room segmentation.

Ochmann et al. (2014) describe extracting hierarchical topometric maps directly from point clouds. The hierachy is divided into four encapsulating layers, building - storey - room - object. Entities within the graph are represented as vertices, with edges representing the topological and spatial relationships between entities. Each vertex is linked to a local metric map of the entity's geometry.

Gorte et al. (2019) provide a novel approach for extracting the walkable floor space from a voxel grid across multiple storeys (see figure 9). They do so by first applying a 3D convolution filter using a stick-shaped kernel to extract the parts of the floor without obstructions. They then apply an upwards dilation to connect steps of stairways into a connected volume. Because the topology of the environment depends on the traversability between spaces the extraction of navigable floor space is essential for the extraction of topological maps.

Bot et al. (2019) propose a graph matching approach to the map merging problem. They use spectral graph matching on dual graphs of a mesh representation of the environment to localize an agent within a pre-existing BIM map

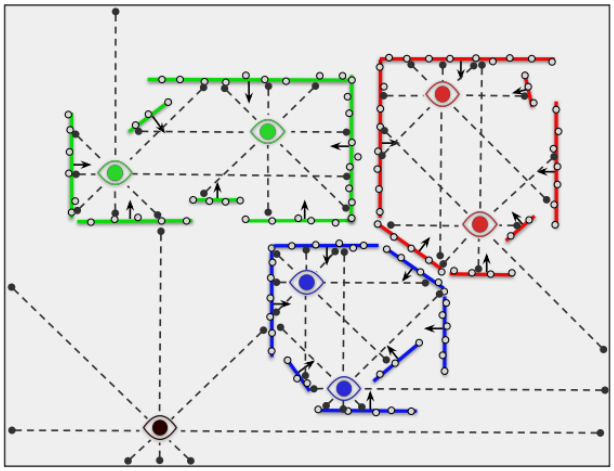

Figure 8: Illustration of visibility clustering-based room segmentation (Pintore et al., 2020).

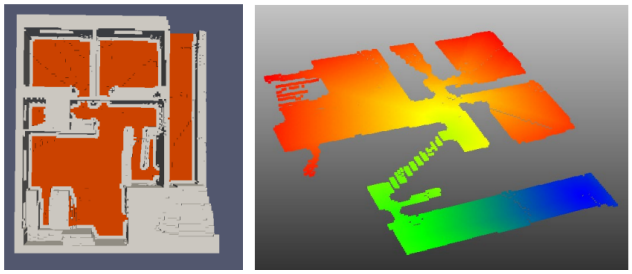

Figure 9: Illustration of walkable floor space extraction(Gorte et al., 2019).

(see figure 10). Although their approach is not specifically proposed for map merging without prior maps their work should be well-suited for this purpose too. Additionally, the graphs used in this approach do not represent the connectivity of places within the environment, Nevertheless, their approach could also be applied to topometric maps. Finally, the graph clustering approach proposed in this research could also be applied to room segmentation.

He et al. (2021) describes an approach for extracting a hierarchical topological-metric map with three layers: storey - region - volume, from a voxel grid map. To extract the map they use a novel approach to room segmentation using ray-casting. A downside of their methodology is that it depends on the presence of ceilings in the metric map, which are often not captured in practice when using handheld scanners.

Ma et al. (2020) and Tang et al. (2022) both propose a deep learning approach to semantic segmentation of indoor spaces. Despite the fact that the

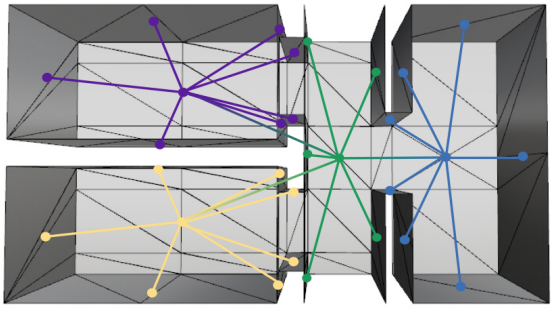

Figure 10: Dual graph of a mesh representation of the environment coloured according to its spectrum ((Bot et al., 2019)).

primary goal of these methods is not room segmentation but instance segmentation (splitting the map into walls, floor, furniture, etc.) their findings can also be applied to room segmentation. In contrast to previous approaches these approaches require training a model on a labelled dataset.

3.2.2 Topometric map merging

In comparison to topometric map extraction relatively little work has been done on the subject of topometric map merging. Dudek et al. (1998) first proposes an approach to topological map merging which depends on a robot meeting strategy to merge partial maps created by each robot. When new distinctive places are recognized at the frontier of the global map the other robots will travel towards it and synchronize their maps. As with other early approaches to map extraction and map merging this approach depends on a coordinated exploration strategy.

The work of Huang & Beevers (2005) is a significant milestone in topological map merging, demonstrating that topological maps can be merged using both map structure and map geometry. They first identify vertex matches by comparing the similarity of their attributes, such as their degree, and the spatial relationships of incident edges. Vertex matches are then expanded using a region growing approach, where every added edge and vertex is compared for similarity and rejected if too dissimilar (see figure 11). The results are multiple hypotheses for overlapping areas between partial maps. They then estimate a rigid transformation between the partial maps for each hypothesis. Afterwards, hypotheses that result in similar transformations are grouped into hypothesis clusters. They then select the most appropriate hypothesis cluster by using a heuristic that includes the number of vertices in the cluster, the error between matched vertices after transformation and the number of hypotheses in the cluster.

Bonanni et al. (2017) provides a unique approach to topometric map merging

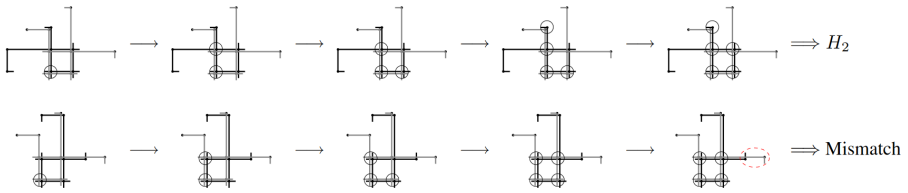

Figure 11: Illustration of hypothesis growing approach from (Huang & Beevers, 2005).

using the pose graph of mapping agents as the topological component and the point cloud captured at each node as the metric component. Matches between nodes are identified by computing the similarity of their associated point cloud. In comparison to most other map merging approaches they fuse the maps using a non-rigid transformation, meaning the partial maps are deformed to improve their alignment.

Garcia-Fidalgo & Ortiz (2017) proposes a hierarchical approach for place recognition in topological maps in which images of the environment are grouped by similarity and described by both a local descriptor of their properties and a global descriptor of their grouping's properties. This approach reduces the search space when recognizing places. Note that this approach does not use a 3D metric component but an image-based one.

Rincon & Carpin (2019) proposes an approach to topological map merging that is based on both the similarity of nodes and their context within the graph based on a model of human object recognition. This approach depends on the previous alignment of partial maps.

Rozemberczki et al. (2021) propose an approach to feature embedding in graphs that combines each node's associated attribute with the distribution of the attributes of its neighbourhood over multiple scales. While they do not use this for 3D mapping their approach can be applied to topometric maps.

4 Methodology

In this section we will describe our methodology for solving the map merging problem. We divide our methodology into three major components: map extraction, map matching and map fusion, which correspond with the three research subquestions. This section follows this division, with an added subsection describing the different map representations that we use. Figure 12 shows the steps of our methodology. Refer to the specific subsections for each step for a further description of the algorithms and notation. We will now give a short summary of each of the steps of our methodology.

Map extraction For each input partial map, a point cloud, we create a voxel grid with a given cell size. Within these voxel grids we detect which voxels could feasibly be used to navigate (walk) through the environment. Using this information we segment the voxel grids into submaps which closely match a human interpretation of how an indoor environment can be divided into rooms. We do so by finding areas with many common viewpoints. By combining the room submaps with the navigable voxels we can extract the environment's topological graph. We then create a topometric map for each input point cloud by merging the topological graph with the segmented voxel grid into a single map.

Map matching In the second part of our methodology, map matching, we identify matches between the rooms of the partial topometric maps with the purpose of detecting overlapping areas. We do this by first generating a global descriptor for each room that captures its geometric features and those of its context, the rooms that lie within a number of steps in the topological graph. We then find a matching by growing multiple matching hypotheses along the topological graphs in a constrained manner and selecting the one that contains the largest number of matches.

Map fusion In the final part of our methodology, map fusion, we find the transformation that aligns the partial maps' geometry and use it to create a single, global topometric map. We do so by finding the optimal transformation between each pair of matched rooms. We then cluster the transformations based on similarity and select the cluster whose mean transformation best aligns the partial maps as the most likely correct transformation. After using the mean transformation to align the geometry of the partial maps into a global voxel grid map we extract a topometric map from it to create the global topometric map.

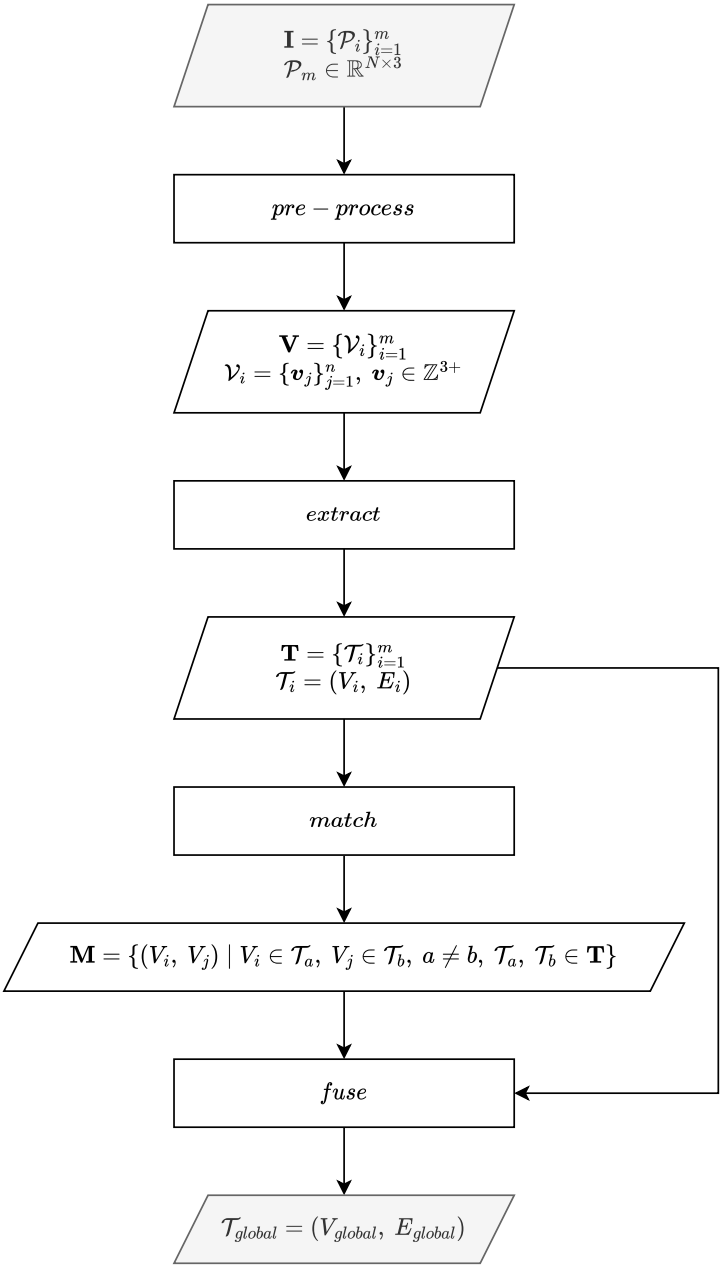

Figure 12: Diagram showing overview of methodology.

4.1 Map Representations

In this section we will give a description of the different kinds of map representation that are used in this research, their mathematical notation, and the operations that we perform on them.

4.1.1 Point Cloud

An unordered collection of points representing the geometry of an object or environment in 3D euclidean space (Volodine, 2007).

$$\mathcal{P} = \{p_i\}_{i=1}^{m}, p_i \in \mathbb{R}^3 \tag{1}$$

Where $\mathcal{P}$ denotes the point cloud and n the number of points that it contains.

4.1.2 Voxel Grid

A voxel is the 3D equivalent of a pixel. A voxel represents a single cell in a bounded 3D volume divided into a regular voxel grid. A voxel represents information about its volume, such as whether it is occupied, what color it is, or any other property. Unless otherwise specified, voxels in this research represent whether a given volume is occupied by any kind of obstruction, such as an object or the environment. A voxel can be represented by a three-dimensional vector containing its coordinates along the x, y and z axes of the voxel grid, as shown in equation 2.

$$\boldsymbol{v} = (x, y, z)^T \in \mathbb{N}^3 \tag{2}$$

We define a voxel grid as a set of voxels with an associated size e_l, as shown in equation 3. Unoccupied voxels are not present in the set, making it a sparse representation. Figure 13 shows an example voxel grid and its components.

$$\mathcal{V} = \{\boldsymbol{v}_i\}_{i=1}^{n} \tag{3}$$

To generate a voxel grid we divide a 3D axis-aligned volume bounding box defined by minimum and maximum bounds $\mathbf{b}_{min}, \mathbf{b}_{max} \in \mathbb{R}^3$ into a grid of cubic cells with distance $e \in \mathbb{R}^+$ between their centers. A voxel represents a subvolume of the bounding box bounded by a single cell. A voxel coordinate only consists of integer values that represent the position of the voxel in the grid along each axis. Voxel $\boldsymbol{u} = (0, 0, 0)^T$ represents the first cell along each of the voxel grid's axes and the minimum of the volume's bounds, voxel $(0, 1, 1)^T$ represents the first cell along the x- and the second along the y- and z-axes, etc. We also restrict voxel coordinates to only be positive as negative coordinates would fall outside of the bounds of the volume. For the same reason a voxel's coordinates can not be larger than that of the voxel representing the volume's maximum bounds $\boldsymbol{w} = (\mathbf{b}_{max} - \mathbf{b}_{min}) \rfloor e$, where $\rfloor$ denotes floor division.

$$\boldsymbol{v}_{min} = \mathbf{b}_{min} + \boldsymbol{v}e \tag{4}$$

$$\boldsymbol{v}_{max} = \boldsymbol{v}_{min} + e \tag{5}$$

The minimum and maximum bounds of this subvolume are given by equation 4 and 5. The voxel's centroid is given by equation 6. Given a point $\boldsymbol{p}$ within the bounds of V, the corresponding voxel is given by equation 7 .

$$\boldsymbol{v_c} = (\boldsymbol{v_{min}} + 0.5\boldsymbol{v_{max}}) \tag{6}$$

$$\boldsymbol{v_p} = (\boldsymbol{p} - \mathbf{b}_{min}) \rfloor e_l \tag{7}$$

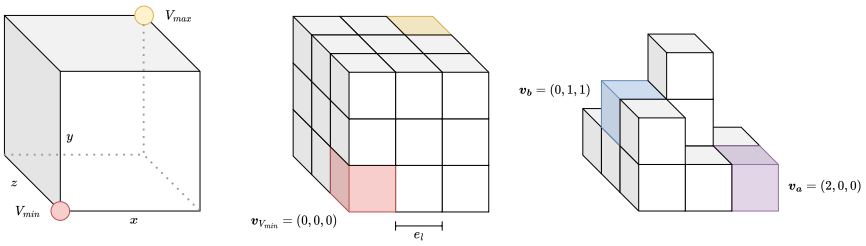

Figure 13: A voxel grid and its components.

Sparse Voxel Octree Several operations on voxel grids benefit from using a spatial index, including radius searching and level of detail generation. We use a data structure called a sparse voxel octree (SVO) to achieve this. A normal octree recursively subdivides a volume into 8 cells, called octants. This operation results in a tree data structure, with nodes representing octants at a certain level of subsidivision. The root node of the tree structure represents the entire volume while the leaf nodes represent batches of 1 or more data points. In the case of a sparse voxel octree the leaf nodes represent individual voxels, with only the octants containing an occupied voxel represented in the tree.

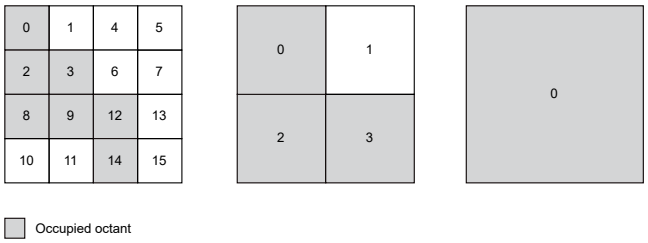

Figure 14: 2D example of morton codes.

To generate the SVO we first create a Morton order for the voxel grid, this is illustrated in figure 14. A Morton order maps the three-dimensional coordinates of the voxels to one dimension while preserving locality. It does by interleaving the binary representation of the voxel's coordinates into a single binary string

which is interpreted as a positive integer, a Morton code. The ascending sorted vector of Morton codes gives us the Morton order. We divide the Morton order into buckets with size 8, such that each bucket contains at most 8 Morton codes, with a maximum difference of 8. Each non-empty bucket represents a parent node of at most 8 child nodes in the octree. By recursively performing this step until only one bucket remains, the root node, we can construct an SVO. The SVO corresponding to the Morton code in figure 14 is shown in figure 15.

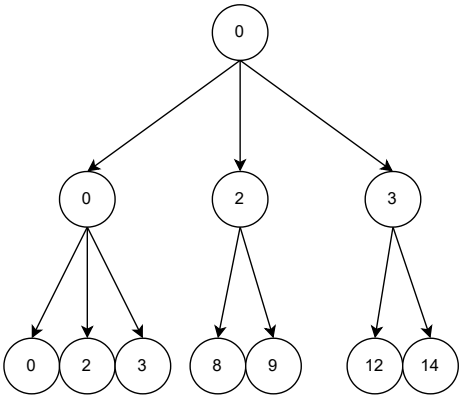

Figure 15: Example of a sparse voxel octree generated from the above Morton codes.

We denote the function that returns all n voxels within range r of a voxel as follows.

$$radius : \mathbb{N}^3,\ \mathbb{R} \mapsto \mathbb{Z}^{n\times 3} \tag{8}$$

The voxels within a sphere around a point can be found by recursively intersecting the sphere with the octants of the SVO. If the sphere does not intersect with an octant then none of its leaf nodes do and the corresponding voxels are not within the sphere. If an octant does intersect with the sphere then its children are tested for intersection. The algorithm returns all leaf nodes that intersect with the sphere.

Levels of detail can be generated by using the occupied octants at the levels above leaf nodes as a simplified voxel grid. This can enable certain operations that are not computationally feasible at the original level of detail.

Voxel convolution Voxel convolution involves moving a sliding window, or kernel, over each voxel in the grid to retrieve its neighbourhood and then computing a new value for the voxel based on the weighted sum of its neighbours. We can define a kernel $\mathcal{K}$ as a voxel grid with an associated weight for each voxel and an origin voxel $\boldsymbol{o}_{\mathcal{K}}$.

$$weight : \mathbb{N}^3 \mapsto \mathbb{R} \tag{9}$$

$$\boldsymbol{o}_{\mathcal{K}} \in \mathbb{N}^3 \tag{10}$$

To apply a kernel to a voxel we first translate the kernel so that its origin lies on the voxel.

$$\mathcal{K}_t = \{\boldsymbol{v}_{\boldsymbol{\mathcal{K}}} + (\boldsymbol{v} - \boldsymbol{o}_{\boldsymbol{\mathcal{K}}}) \mid \boldsymbol{v}_{\boldsymbol{\mathcal{K}}} \in \mathcal{K}\} \tag{11}$$

We then get the property which we wish to convolve of each neighbour, multiply it by the neighbour's weight and sum it.

$$\mathcal{K}_{property}(\boldsymbol{v}) = \sum \{weight(\boldsymbol{v})\mathcal{V}_{property}(\boldsymbol{v}) \mid \boldsymbol{v} \in \mathcal{K}_t \cap \mathcal{V}\} \tag{12}$$

We denote the convolution of a property of every voxel in $\mathcal{V}$ with $\mathcal{K}$ as follows.

$$\mathcal{V}_{property,\ \mathcal{K}} = \mathcal{V} * \mathcal{K}_{property} = \{\mathcal{K}_{property}(\boldsymbol{v}) \mid \boldsymbol{v} \in \mathcal{V}\} \tag{13}$$

Neighbourhood graph The neighbourhood graph of a voxel grid represents the connectivity between voxels as undirected, unweighted graph. The nodes of the neighbourhood graph correspond to individual voxels and the edges to whether two voxels can be considered neighbours. Whether two voxels are neighbours is defined by a kernel which has only 1 or 0-valued weights. If, when applying the kernel to a voxel, another voxel within that kernel is occupied and the kernel's weight for that position is 1 then the two voxels are neighbours. The neighbourhood graph allows us to perform graph operations, such as identifying connected components, on voxel grids. Figure 16 shows two commonly used kernels for constructing neighbourhood graphs, the Von Neumann and Moore neighbourhoods, also respectively known as the 6-neighbourhood and the 26-neighbourhood.

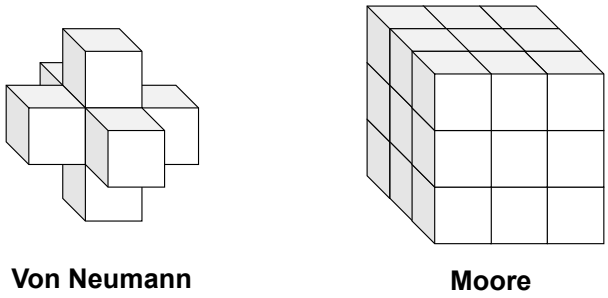

Figure 16: Two common connectivity kernels. Note that the center voxel's weight is 0, as a voxel does not neighbour with itself.

4.1.3 Topological map

Topological maps are graph representations of an environment's structure, where nodes represent locally distinctive places and edges represent traversable paths between them (see figure 17) (Thrun, 1998; Kuipers & Byun, 1988). Topological maps are based on observations that cognitive maps, the mental maps used by humans to navigate within an environment, consist of multiple layers with a topological description of the environment being a fundamental component (Kuipers & Byun, 1988; Kuipers, 1978).

We denote a topological map as shown in equations 14, 15 and 16. The topological map consists of a graph G, where nodes N represent distinctive places n_i and edges E represent the presence of a navigable path between neighbouring pairs of places (n_j, n_k) that does not pass through any other places. Whether a path is navigable depends on who or what is navigating. For the purpose of this thesis a navigable path is a path that can be reasonably used by humans to walk from one room to another. Following this definition, only a part of the environment can be used as a navigable path. This includes the parts of the floor, stairs or ramps that are at sufficient distance from a wall, the ceiling or other obstructions.

$$G = (N, E) \tag{14}$$

$$N = \{n_i\}_{i=1}^{k} \tag{15}$$

$$E = \{(n_j, n_k)_i\}_{i=1}^{m},\ n_j \in N,\ n_k \in N,\ n_j \neq n_k\} \tag{16}$$

Figure 17 shows an example topological map of a house with five rooms and their connectivity.

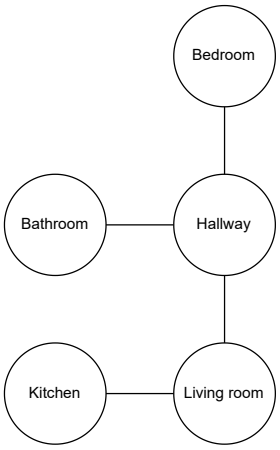

Figure 17: Example of a topological map.

4.1.4 Topometric map

A hybrid map representation combining both the topological and geometric characteristics of the environment. This map representation allows the end-user to use either topological or metric information depending on the needs of the situation, e.g. the topological layer can be used for large-scale navigation and abstract reasoning while the metric layer can be used for place recognition and obstacle avoidance. In the context of this thesis a topometric map refers to a graph representation of an indoor environment where the nodes represent rooms and their associated geometry as a voxel grid and the edges represent the navigability relationship between them. It is thus a hybrid representation of the environment that combines the properties of the voxel grid and the topological map which we described above. We denote a topometric map $\mathcal{T}$ as shown in equation 17, where $\mathcal{V}$ represents the complete geometry of the environment and G the topological graph.

$$\mathcal{T} = (\mathcal{V},\ G),\ \mathcal{V} = \{\boldsymbol{v}_i\}_{i=1}^{n} \tag{17}$$

The topological graph, which we denote as shown in equation 18, consists of a set of nodes N and a set of edges E. Each node $n \in N$ represents a room and contains a subset of $\mathcal{V}$ that describes the geometry of that room. The nodes' subsets of $\mathcal{V}$ are not allowed to overlap, which means they represent a segmentation of $\mathcal{V}$.

$$G = (N,\ E),\ N = \{n_i\}_{i=1}^{k},\ n \subset \mathcal{V} \tag{18}$$

Figure 18 shows an example of a topometric map of an indoor environment.

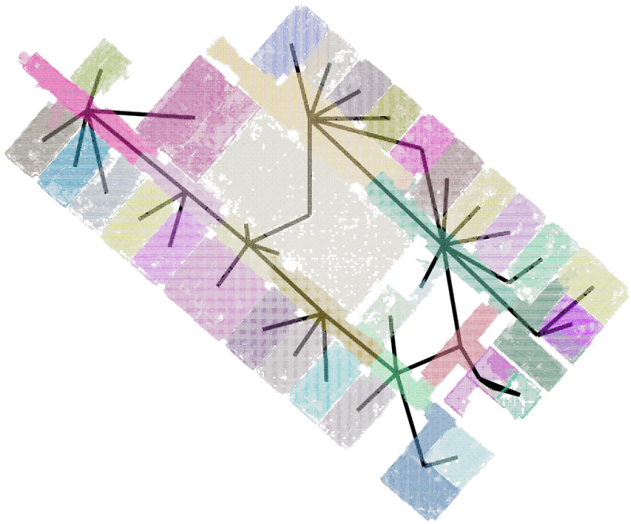

Figure 18: Example of a topometric map.

4.2 Map Extraction

The first step of our approach is topometric map extraction. The purpose of this step is to transform a partial voxel grid map of an indoor environment, denoted by $\mathcal{V}$, into a topometric map, denoted by $\mathcal{T}$. In this section we propose an algorithm to achieve this goal. In an overview, it works as follows.

4.2.1 Overview

We first extract a navigation graph $\mathcal{G}_{navigation}$, the neighbourhood graph of all voxels which a hypothetical human agent could use to move through the environment, from $\mathcal{V}$. Using $\mathcal{G}_{navigation}$ we compute points where the hypothetical agent would have an optimal view of the environment. We then find the visible voxels for each of these points. By clustering the resultant visibilities based on similarity we segment $\mathcal{V}$ into submaps that align closely with how humans may divide indoor environments into rooms. As such, we refer to the submaps of $\mathcal{V}$ when segmented using visibility clustering as 'rooms'. We then construct the topological graph of the environment by finding which rooms have adjacent voxels in $\mathcal{G}_{navigation}$. Finally, we fuse the topological graph with the segmented map to construct the topometric map $\mathcal{T}$.

Figures 19 and 20 show an overview of our map extraction algorithm, its input, and intermediate outputs. In the rest of this subsection we will discuss the algorithm in detail.

4.2.2 Navigation graph

We first extract a navigation graph $\mathcal{G}_{navigation}$. The navigation graph is a connected graph which tells us how a theoretical agent in the environment could move through the environment from one point to another. In practice, assuming a human agent, this means the areas of the floor, ramps and stairs that are at a sufficient distance from a wall, the ceiling or any other obstruction. We compute $\mathcal{G}_{navigation}$ using a three step algorithm which we describe below.

Convolution The first step of navigation graph extraction uses voxel convolution with a stick-shaped kernel $\mathcal{K}_{stick}$ (shown in figure 21) to find all voxels that are unobstructed and may thus be used to navigate the environment. This approach is based on Gorte et al. (2019). Each voxel in the kernel has a weight of 1, except the origin voxel which has weight 0. Convolving the voxel grid's occupancy property with the stick kernel gives us each voxel's obstruction property, which has a value of 0 when no other voxels are present in the stick kernel. This indicates that these voxels have enough space around and above them to be used for navigation. We then filter out all obstructed voxels leaving only the voxels that could be used for navigation.

$$\mathcal{V}_{unobstructed} = \{\boldsymbol{v} \mid \boldsymbol{v} \in \mathcal{V},\ \mathcal{K}_{stick} * \boldsymbol{v} = 0\} \tag{19}$$

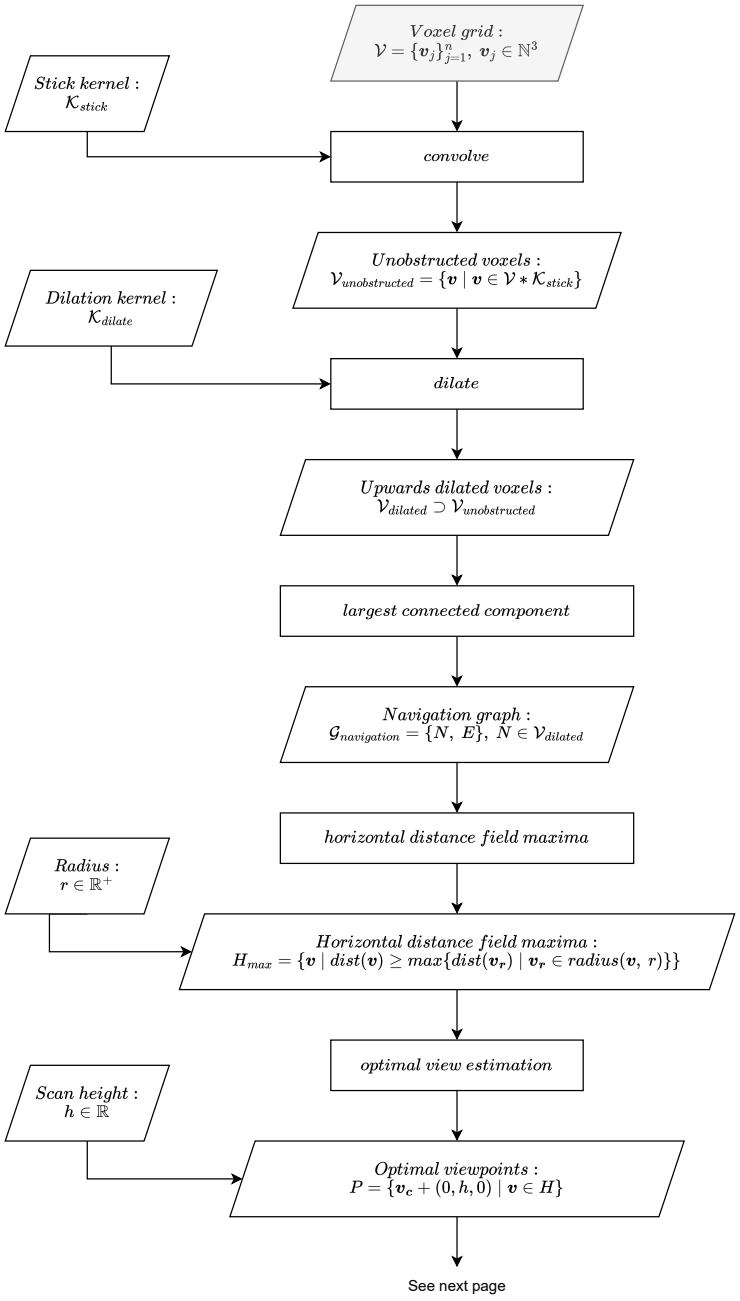

Figure 19: Diagram showing map extraction processes and intermediate data (part 1).

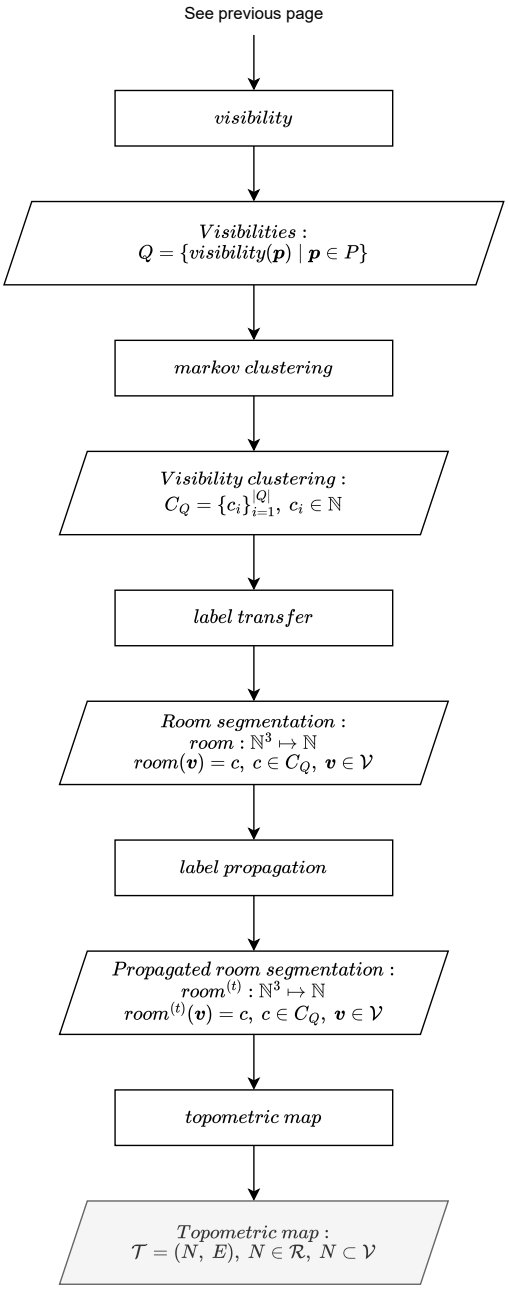

Figure 20: Diagram showing map extraction processes and intermediate data (part 2).

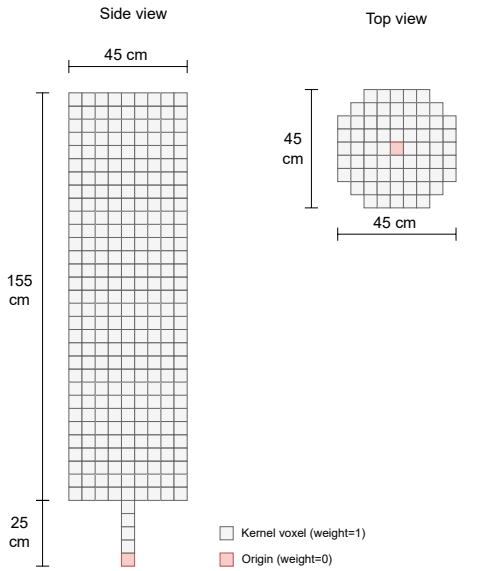

Figure 21: Illustration of stick kernel with top and side views.

Upwards dilation The next step of the algorithm is to dilate the unobstructed voxels upwards by a distance d_{dilate}. This connects the voxels separated by a small height differences into a connected volume, which is necessary for the navigation graph to be able to connect stairs. The value of d_{dilate} depends on the expected differences in height between the steps of stairs in the environment. Typically we use a value of 0.2m. The result is a new voxel grid $\mathcal{V}_{dilated}$. Note that the dilation step may create new occupied voxels that are not in the original voxel grid which means that $\mathcal{V}_{dilated}$ is not necessarily a subset of $\mathcal{V}$.

Connected components The final step of the algorithm is to split $\mathcal{V}_{dilated}$ into one or more connected components. A connected component $\mathcal{V}_i$ of a voxel grid is a subset of $\mathcal{V}$ where there exists a path between every voxel in $\mathcal{V}_i$. We denote the set of all connected components as $\mathcal{C} = \{\mathcal{V}_i\}_{i=1}^{n}$. To find the connected components we first find the neighbourhood graph of $\mathcal{V}_{dilated}$ using the Von Neumann neighbourhood kernel $\mathcal{K}_6$, which is shown in equation 20.

$$\mathcal{G}_{\mathcal{K}_6} = (N,\ E),\ N = \mathcal{V}_{dilated} \tag{20}$$

We then find the connected components of the neighbourhood graph using the below algorithm. After doing so we find the connected component with the largest amount of voxels and use it as the navigation graph $\mathcal{G}_{navigation}$. We denote the intersection of the voxels in the navigation graph with $\mathcal{V}$ as $\mathcal{V}_{navigation}$.

Algorithm 1 Region growing connected components

Input Dilated voxel grid $\mathcal{V}_{dilated}$
Input Von Neumann Connectivity kernel $\mathcal{K}_6$
Output Connected components $\mathcal{C}$

$\mathcal{G}_{\mathcal{K}_6} = (N,\ E),\ N = \mathcal{V}_{dilated}$ ▷ Convert voxel grid to neighbourhood graph
$N_{unvisited} = N$
$\mathcal{C} = \{\}$
while $|N_{unvisited}| \neq 0$ **do**
  Select random node n from $N_{unvisited}$
  Remove $DFS(n)$ from $N_{unvisited}$ ▷ Depth-first search to find connected nodes
  Add $DFS(n)$ to $\mathcal{C}$
end while

4.2.3 Room segmentation

The next step of our approach is to segment the complete voxel grid map $\mathcal{V}$ into non-overlapping rooms. We do so by using a visibility clustering approach. We will now describe the algorithm that we use to achieve this.

Maximum visibility estimation To segment the map into rooms using visibility clustering it is first necessary to identify the viewpoints that will be used to compute the visibilities. Ideally, we want viewpoints that maximize the view of the environment. This is related to the skeletonization problem, which aims to find a version of a shape that is equidistant to its boundaries, and for which approaches such as the medial axis transform and the grassfire transform exist (Bonnassie et al., 2003; L. Liu et al., 2011). The reasoning behind this is that the points that maximize the view of the environment should be equally spaced and as far away from the walls and obstructions as possible. As we only need the nodes of the skeleton we can use an approach which is less complex and more computationally efficient. This approach works by finding the points that are at a maximum distance from the boundary of the navigation graph within their local neighbourhood (Mille et al., 2019). It works as follows. For each voxel in the navigation graph we compute the horizontal distance to the nearest boundary voxel. A boundary voxel is a voxel for which not every voxel in its Von Neumann neighbourhood is occupied. To compute this value we iteratively convolve the voxel grid with a circle-shaped kernel on the X-Z plane, where the radius of the circle is expanded by 1 voxel with each iteration, starting with a radius of 1. When the number of voxel neighbours within the kernel is less than the number of voxels in the kernel a boundary voxel has been reached. The number of radius expansions that were performed tells us the distance to the boundary of a particular voxel. We denote the horizontal distance of a voxel to its boundary as $dist : \mathbb{N}^3 \mapsto \mathbb{N}$. Computing the horizontal distance for every voxel in $\mathcal{V}$ gives us the horizontal distance field (H), as shown in equation 21.

$$H = \{dist(\boldsymbol{v}) \mid \boldsymbol{v} \in \mathcal{V}\} \tag{21}$$

$$H_{max} = \{\boldsymbol{v} \mid dist(\boldsymbol{v}) \geq max\{dist(\boldsymbol{v_r} \mid \boldsymbol{v_r} \in radius(\boldsymbol{v},\ r))\}\} \tag{22}$$

We denote the horizontal distance of a given voxel $\boldsymbol{v}$ as $d_{\boldsymbol{v}}$. We implement this using the following algorithm.

Algorithm 2 Horizontal distance field

Input Navigation voxel grid $\mathcal{V}_{navigation}$
Output Horizontal distance field $H = \{dist(\boldsymbol{v}) \mid \boldsymbol{v} \in \mathcal{V}_{navigation}\}$

```
H = (h_i)_{i=1}^{|V_navigation|}
j = 0
for each v ∈ V_navigation do
    r = 1
    Create K_circle with radius r
    while K ∗ v = |K_circle|) do
        r++
        Expand K_circle with new radius r
    end while
    h_j ← r                ▷ Add voxel's radius to horizontal distance field
    j++
end for
```

We then find the maxima of the horizontal distance field within a given radius $r \in \mathbb{R}^+$. The local maxima of the horizontal distance field are all voxels that have a larger or equal horizontal distance than all voxels within r, such that equation 22 follows. Increasing the value of r reduces the number of local maxima and vice versa. All voxels in H_{max} lie within the geometry of the environment, which means the view of the environment is blocked by the surrounding voxels. To solve this, we take the centroids of the voxels in H_{max} and translate them upwards to a reasonable scanning height h for a human agent, we use 1.8m. We denote these positions as:

$$P = \{\boldsymbol{v_c} + (0, h, 0) \mid \boldsymbol{v} \in HDF_{max}\} \tag{23}$$

Figure 22 shows an illustration of the horizontal distance field computation and the identification of its local maxima. Figure 23 shows a real example of the horizontal distance field extracted from a small two-storey environment in grayscale and the associated P in red.

Visibility computation The next step in the room segmentation algorithm is to compute the visibility from each position in P. We denote the set of voxels that are visible from a given position as:

$$visibility : \mathbb{R}^3,\ \mathbb{Z}^{n\times 3} \mapsto \mathbb{N}^{m\times 3},\ n \geq m \tag{24}$$

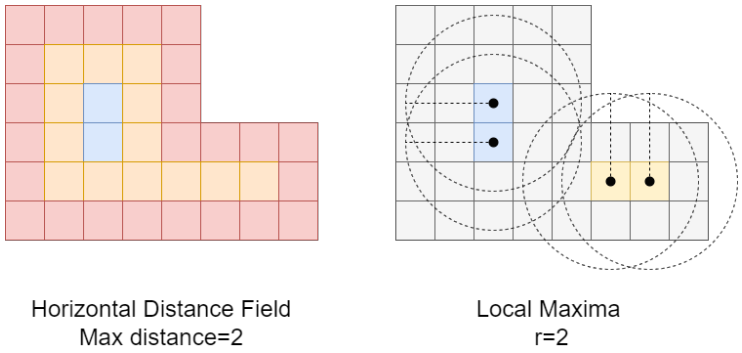

Figure 22: Illustration of horizontal distance field computation and extraction of local maxima.

A target voxel is visible from a position if a ray cast from the position towards the centroid of the voxel does not intersect with any other voxel. To compute this we use the fast voxel traversal (FVT) algorithm to rasterize the ray onto the voxel grid in 3D (see algorithm 3) (Amanatides & Woo, 1987). We then check if any of the voxels that the ray enters, except the target voxel, is occupied. If none are, the target voxel is visible from the point. Figure 24 shows a 2D representation of how the FVT algorithm works. Figure 25 shows a 2D representation of a visibility computation.

We perform this raycasting operation from every position in P towards every voxel within a radius r_v of that position. Only taking into account voxels within a radius speeds up the visibility computation, and is justifiable based on the fact that real-world 3D scanners have limited range. We denote the set of visibilities from each point in P as:

$$Q = \{visibility(\boldsymbol{o}) \mid \boldsymbol{o} \in P\} \tag{25}$$

,

Where a single visibility consists of all of the voxels for which a ray cast towards its origin from a point in P does not hit any other voxels along its way, as shown in equation 26.

$$visibility(\boldsymbol{o}) = \{\boldsymbol{t} \mid \boldsymbol{t} \in radius(\boldsymbol{o}, r_v),\ FVT(\mathcal{V}, \boldsymbol{o}, \boldsymbol{t}) = \boldsymbol{t}\} \tag{26}$$

Visibility clustering After computing the set of visibilities from the estimated optimal views we apply clustering to group the visibilities by similarity. Remember that each visibility is a subset of the voxel grid map. To compute the similarity of two sets we use the Jaccard index, which is given by equation 27.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{27}$$

Algorithm 3 FVT (Fast Voxel Traversal)

Input Voxel grid $\mathcal{V}$
Input Ray origin $\boldsymbol{o} \in \mathbf{R}^3, \boldsymbol{o} \in [\mathcal{V}_{min}, \mathcal{V}_{max}]$
Input Ray target $\boldsymbol{t} \in \mathbf{R}^3$
Output $hit \in \mathbb{N}^3$ ▷ First encountered collision

$\boldsymbol{p}_{current} = \boldsymbol{o}$
$\boldsymbol{v}_{\boldsymbol{o}} = (\boldsymbol{o} - \mathcal{V}_{min})//\mathcal{V}_e$
$\boldsymbol{v}_{current} = \boldsymbol{v}_{\boldsymbol{o}}$
$\boldsymbol{d} = (\boldsymbol{t} - \boldsymbol{o})$
$heading = \boldsymbol{d} \odot abs(\boldsymbol{d})^{-1}$ ▷ Determine if ray points in positive or negative direction for every axis
while $(\boldsymbol{v}_{current} \notin \mathcal{V} \vee \boldsymbol{v}_{current} = \boldsymbol{v}_{\boldsymbol{o}}) \wedge \boldsymbol{p}_{current} \in [\mathcal{V}_{min}, \mathcal{V}_{max}]$ **do**
    $\boldsymbol{c} = centroid(\boldsymbol{v}_{current})$
    $d_{planes} = \boldsymbol{c} + heading * \mathcal{V}_e/2$
    $d_{min} = \infty$
    $axis = 1$
    for each $d \in d_{planes}$ **do**
        $t = \frac{d - \boldsymbol{n} \cdot \boldsymbol{p}_{current}}{\boldsymbol{n} \cdot (\boldsymbol{t} - \boldsymbol{p}_{current})}$
        $\boldsymbol{i} = \boldsymbol{p}_{current} + t(\boldsymbol{t} - \boldsymbol{o})$
        if $d_{min} \geq t$ **then**
            $\boldsymbol{p}_{current} = \boldsymbol{i}$
            $\boldsymbol{v}_{current,\ axis} += heading_{axis}$
        end if
        $axis = axis + 1$
    end for
    $hit = \boldsymbol{v}_{current}$
end while

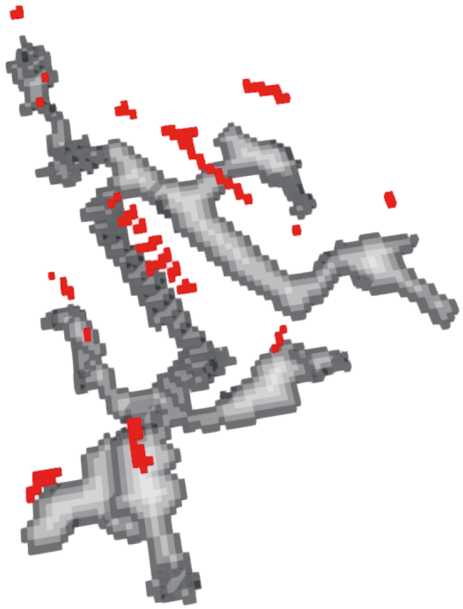

Figure 23: Resulting horizontal distance field of partial map, with resultant optimal view points shown in red.

Computing the Jaccard index for every combination of visibilities gives us a visibility overlap matrix $\boldsymbol{S} \in 0, 1^{n \times n}$. An example visibility overlap matrix is shown in figure 26. We can also consider $\boldsymbol{S}$ as an undirected weighted graph $\mathcal{G}_S$, where every node represents a viewpoint and the edges the degree of overlap between what's visible from the two viewpoints, as illustrated in figure 27. This means we can treat visibility clustering as a weighted graph clustering problem. To solve this problem we used the Markov Cluster (MCL) algorithm (van Dongen, 2000). The main parameter of the MCL algorithm is inflation. By varying this parameter between an approximate range of $[1.2, 2.5]$ we get different clustering results. We find the optimal value for inflation within this range by maximizing the clustering's modularity. This value indicates the difference between the fraction of edges within a given cluster and the expected number of edges for that cluster if edges are randomly distributed. We denote the clustering of Q that results from the MCL algorithm as:

$$C_Q = \{c_i\}_{i=1}^{|Q|}, \; c_i \in \mathbb{N} \tag{28}$$

Where the ith element of C_Q is the cluster that the ith element of Q belongs to, such that for a given value of c the elements in Q for which the corresponding c in C_Q have the same value belong to the same cluster. As each visibility is a subset of the map, each cluster of visibilities is also a subset of the map. We

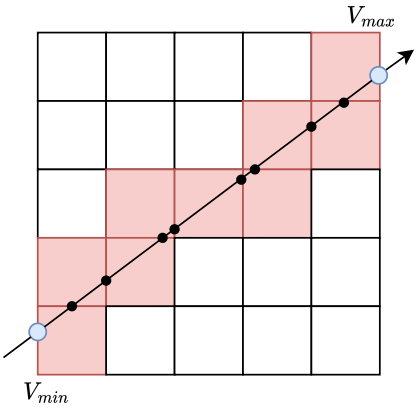

Figure 24: 2D representation of voxel raycasting.

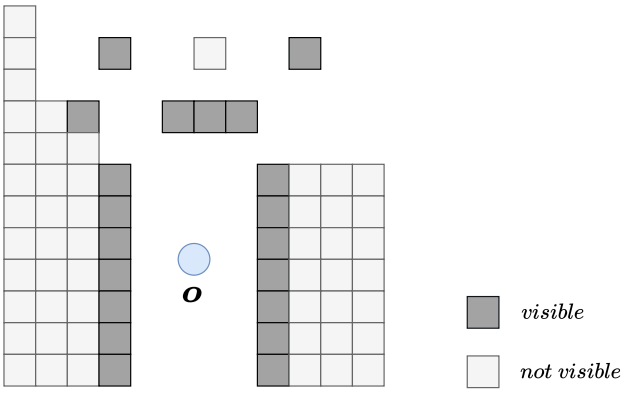

Figure 25: 2D representation of visibility computation.

denote the union of the visibilities belonging to each cluster as $\mathcal{V}_c$.

Label propagation It is possible for visibility clusters in $\mathcal{V}_c$ to have overlapping voxels. This means that each voxel in the partial map may have multiple associated visibility clusters. However, the goal is to assign a single cluster to each voxel in the map to create a non-overlapping segmentation. To solve this we assign to each voxel the cluster which contains the most visibilities that include that voxel. The result is a mapping from voxels to visibility clusters, which we will from now on refer to as rooms, as shown in equations 29 and 30.

$$room : \mathbb{N}^3 \mapsto \mathbb{N} \tag{29}$$

$$room(\boldsymbol{v}) = c,\ c \in C_Q,\ \boldsymbol{v} \in \mathcal{V} \tag{30}$$

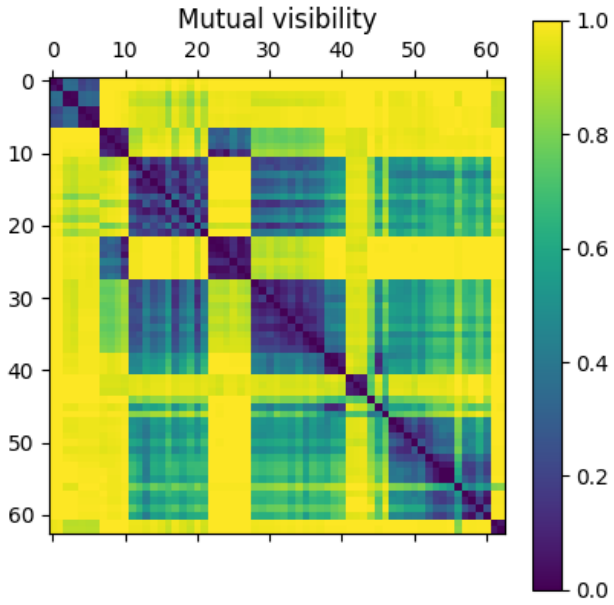

Figure 26: Similarity matrix extracted from set of visibilities. Each value represents the Jaccard index of two visibilities.

This often results in noisy results, with small, disconnected islands of rooms surrounded by other rooms. Intuitively, this does not correspond to a reasonable room segmentation. To solve this we apply a label propagation algorithm, meaning that for every labelled voxel we find the labelled voxels within a neighbourhood as defined by a convolution kernel. We then assign to the voxel the most common label of its neighbourhood if that label is more common than the current label. We iteratively apply this step until the assigned labels stop changing. Depending on the size of the convolution kernel the results are smoothed and small islands are absorbed by the surrounding rooms. Algorithm 4 shows our approach to label propagation.

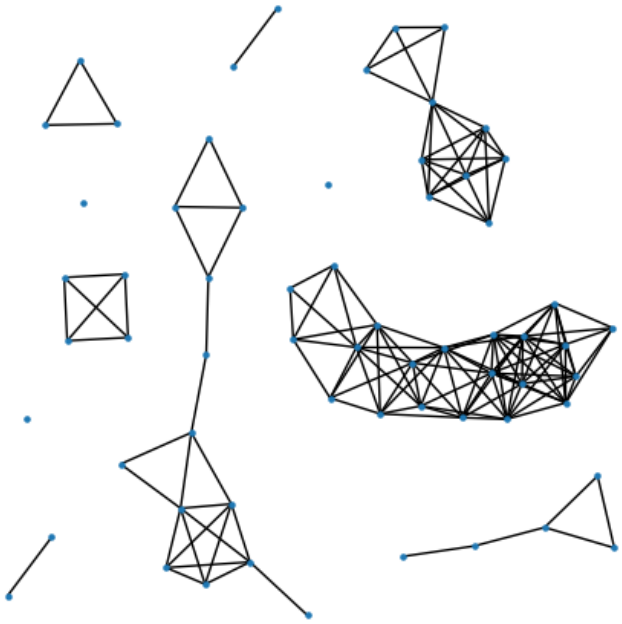

Figure 27: Graph representation of the similarity matrix, edges under a threshold similarity are removed. Nodes represent visibilities.

4.2.4 Topometric map extraction

The above steps segment the map into multiple non-overlapping rooms using visibility clustering. In the next step we extract the topometric representation $\mathcal{T} = (\mathcal{G}, \mathcal{V})$, which consists of a topological graph $\mathcal{G} = (N, E)$ and a voxel grid map $\mathcal{V}$. Each node in $\mathcal{G}$ represents a room and also has an associated voxel grid which is a subset of $\mathcal{V}$ and represents the geometry of that room as a voxel set. Edges in $\mathcal{G}$ represent navigability between rooms, meaning that there is a path between them on the navigable volume that does not pass through any other rooms. This means that for two rooms to have a navigable relationship they need to have adjacent voxels that are both in the navigable volume. To construct the topometric map we thus add a node for every room in the segmented map with its associated voxels, we then add edges between every pair of nodes that satisfy the above navigability requirement.

Algorithm 4 Label propagation

Input Voxel grid $\mathcal{V}$
Input Initial labeling $label^{(0)} : \mathbb{N}^3 \mapsto \mathbb{N}$
Input Kernel $\mathcal{K}$
Output Propagated labeling after t steps $label^{(t)} : \mathbb{N}^3 \mapsto \mathbb{N}$

$t = 0$
while $label^{(t)} \neq label^{(t+1)}$ **do** ▷ Keep iterating until labels stop changing
  for each $\boldsymbol{v} \in \mathcal{V}$ **do**
    $L = \{label^{(t)}(\boldsymbol{v}_{nb}) \mid \boldsymbol{v}_{nb} \in neighbours(\boldsymbol{v}, \mathcal{K})\}4$
    $l_{max} = argmax_l \ |\{l \mid l \in L\}|$ ▷ Most common label in neighbourhood
    $l_{current} = label^{(t)}(\boldsymbol{v})$ ▷ Label of current voxel
    if $|\{l \mid l \in L \wedge l = l_{max}\}| > |\{l \mid l \in L \wedge l = l_{current}|$ **then**
      $label^{(t+1)}(\boldsymbol{v}) = l_{max}$
    else
      $label^{(t+1)}(\boldsymbol{v}) = label^{(t)}(\boldsymbol{v})$
    end if
  end for
  $t = t + 1$ ▷ Use propagated labeling as input for next iteration
end while

4.3 Map Matching

4.3.1 Overview

The process of identifying overlapping areas between partial maps is called map matching. In the case of topometric map matching this refers to identifying which nodes represent the same rooms between two partial maps. We denote our two partial topometric maps as:

$$\mathcal{T}_a = (\mathcal{G}_a,\ \mathcal{V}_a),\ \mathcal{G}_a = (N_a, E_a) \tag{31}$$

$$\mathcal{T}_b = (\mathcal{G}_b,\ \mathcal{V}_b),\ \mathcal{G}_b = (N_b, E_b) \tag{32}$$

The goal of map matching is to find a one-to-one mapping between the rooms of both partial maps which corresponds to the real world and is robust to differences in coordinate system, resolution and quality between partial maps. To identify matches between rooms we need to be able to compute their similarity. To do so, we first transform each room into a descriptor, an n-dimensional vector, which represents both the geometry of the room. The descriptor of two nodes with similar geometry should be close to eachother in feature space, meaning the distance between their vectors should be small. Conversely, the descriptors of two dissimilar rooms should be far away from eachother in feature space. We then use the topological properties of the topometric maps to improve map matching in two ways. The first is contextual embedding. This means that we combine the descriptor of each room with the descriptor of its neighbourhood

in the topological graph. This improves matching because multiple rooms may have similar geometry but not necessarily similar neighbourhoods. The second is hypothesis growing, which means that we grow multiple matching hypotheses along the topological graph in a constrained manner and only use the hypothesis that contains the most matches. Figure 28 shows an overview of the steps described above. In the rest of this section we will describe the aforementioned steps in depth.

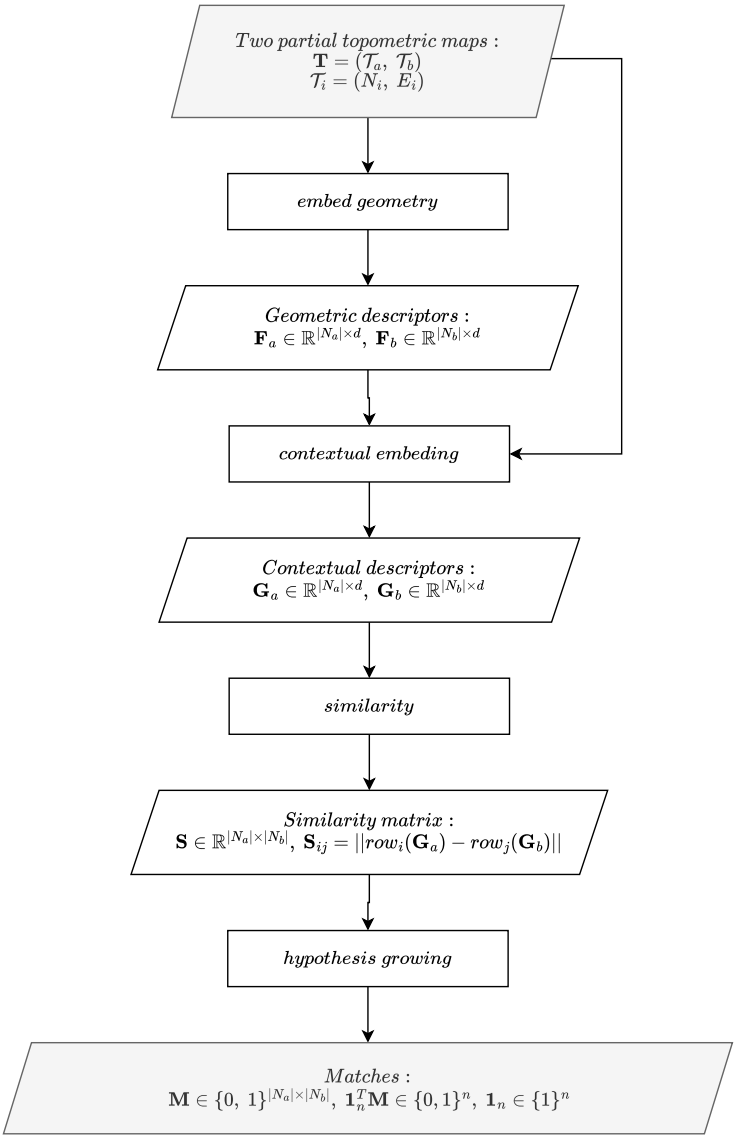

Figure 28: Diagram showing map matching methodology.

4.3.2 Geometric descriptor

Geometric feature embedding transforms a geometric object, in our case a voxel grid, into an m-dimensional vector, a descriptor, such that objects with similar geometry have similar descriptors (their Euclidean distance is small) and vice versa. We implement geometric feature embedding using two different approaches, which we discuss below. The concept of geometric feature embedding is illustrated in figure 29.

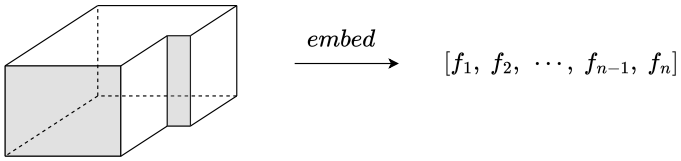

Figure 29: Diagram showing geometric feature embedding.

Spectral Features Our first approach to geometric feature embedding uses ShapeDNA (Reuter et al., 2006). This approach uses the first n sorted, non-zero eigenvalues of the graph Laplacian, in our case of the neighbourhood graph of a room's geometry, as a geometric descriptor. To compute this we first convert each room's neighbourhood graph $\mathcal{G}$ to an adjacency matrix A and a degree matrix D. We then find the Laplacian matrix of the neighbourhood graph by subtracting its adjacency matrix from its degree matrix, as shown in equation 33.

$$L = D - A \tag{33}$$

After computing the Laplacian matrix we find its eigenvalues, sort them in ascending order and use the first 256 non-zero values as the descriptor.

Deep Learning Our second approach to geometric feature embedding uses deep learning. Specifically, we use the LPDNet neural network architecture. This architecture is used for place recognition, it does so by learning descriptors, typically 2048 or 4096-dimensional, of point clouds that are theoretically independent of transformation, perspective and completeness. It does so by computing a local descriptor for every point in the point cloud and aggregrating them into a global descriptor. The LPDNet model we use is trained on outdoor maps which have different characteristics from indoor maps. However, the authors of LPDNet claim that a model trained on outdoor data can also effectively be used for indoor data. Figure 30 shows the network architecture of LPDNet.

4.3.3 Contextual Embedding

After computing a descriptor for each individual room we augment them by taking into account the descriptor of the neighbourhood. For every room we

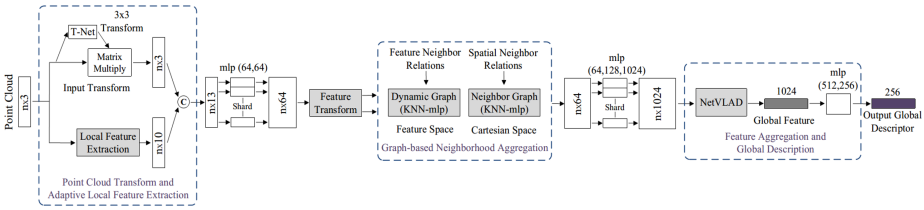

Figure 30: Diagram showing LPDNet network architecture (Z. Liu et al., 2019).

find their neighbours and merge their geometry into one voxel grid, for which we compute a new descriptor. We do this step multiple times for neighbours that are at most one or multiple steps away from the room. We then append the descriptors of the neighbourhood to the room's descriptor. By doing so we can distinguish between rooms with similar geometry but dissimilar neighbourhoods, which are often present in indoor environments. The concept of contextual embedding is illustrated in figure 31.

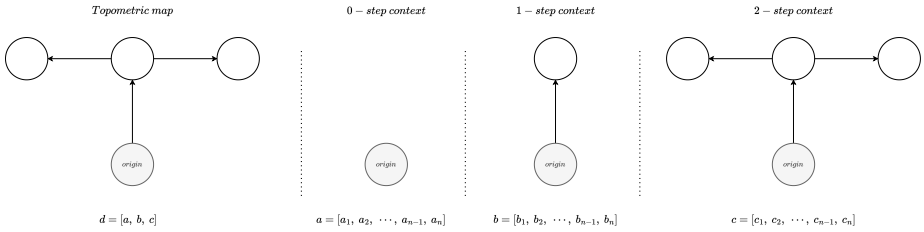

Figure 31: Diagram showing contextual embedding.

4.3.4 Initial Matching

The above steps are applied to both partial maps. This gives us two sets of descriptors $\mathbf{G}_a$, $\mathbf{G}_b$ representing the contextual embedding of the rooms of both topometric maps. To identify the most likely overlapping rooms between the partial maps we find the one-to-one mapping between the sets of descriptors that maximizes the similarity (or minimizes the distances) between the chosen pairs. This is an example of the unbalanced assignment problem, which consists of finding a matching in a weighted bipartite graph that minimizes the sum of its edge weights. It is unbalanced because there may be more nodes in one part of the bipartite graph than the other, which means it is not possible to assign every node in one part to a node in the other. This is illustrated in figure 32.

To construct the weighted bipartite graph we first find the Cartesian product of the feature vectors.

$$\mathbf{G}_{ab} = \mathbf{G}_a \times \mathbf{G}_b = \{(a, b) \mid a \in \mathbf{G}_a,\ b \in \mathbf{G}_b\} \tag{34}$$

We then compute the Euclidean distance in feature space between every pair of nodes in $\mathbf{G}_{ab}$, creating the cost matrix that represents the weighted bipartite graph.

$$\mathbf{S} \in \mathbb{R}^{|N_a| \times |N_b|},\ \mathbf{S}_{ij} = ||row_i(\mathbf{G}_a) - row_j(\mathbf{G}_b)|| \tag{35}$$

We can then find unbalanced assignment using the Jonker-Volgenant algorithm (Jonker & Volgenant, 1987). We denote the resulting matching between the nodes of both partial maps and their distance in feature space as a matrix:

$$\mathbf{M} \in \{0,\ 1\}^{|N_a| \times |N_b|},\ \mathbf{1}_n^T \mathbf{M} \in \{0, 1\}^n,\ \mathbf{1}_n \in \{1\}^n \tag{36}$$

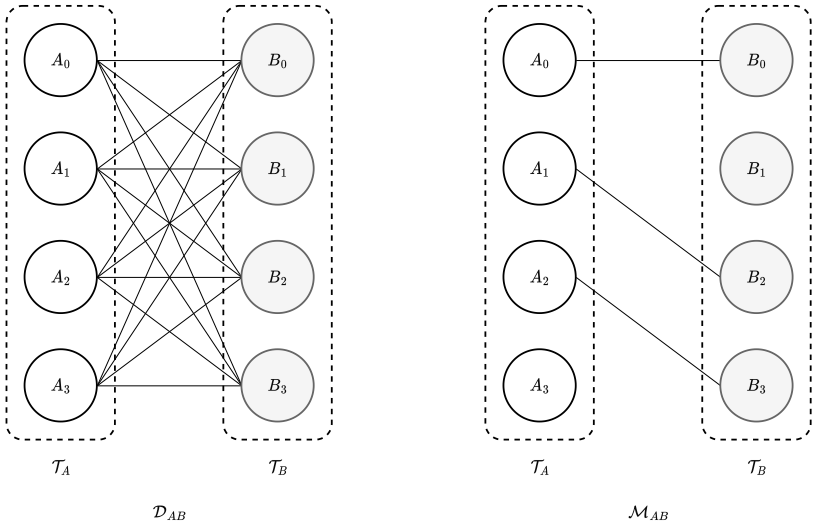

Figure 32: Illustration of unbalanced assignment problem.

4.3.5 Hypothesis growing

In practice it is unlikely that every match in $\mathbf{M}$ is correct. However, we can use them as seeds to generate hypotheses similar to the approach described in Huang & Beevers (2005). Starting at each initial match we get the neighbourhood of both nodes. We then construct a new cost matrix from the Euclidean distance between the embeddings of both neighbourhoods, again creating a weighted bipartite graph for which we can solve the assignment problem. By doing this we identify which neighbours of the nodes in the match are most likely to also match. We recursively apply this step to the matching neighbours to grow our initial matches into hypotheses. To decrease the risk of incorrectly identifying neighbourhood matches we constrain hypothesis growing in two ways. First, the cost of two potential matches must be below a given threshold c_{max}. Second, a newly identified match may not bring the existing matching too much out of

alignment. To check this, we perform a registration (see next section) between the centroids of the geometry of the identified matches at every step of the hypothesis growing. If the error increases between steps, and the increase is too large such that $\triangle e \geq \triangle e_{max}$, then the matching is rejected. By adjusting the values of c_{max} and $\triangle e_{max}$ more or less uncertainty is allowed when growing hypotheses. Hypothesis growing is illustrated in figure 33

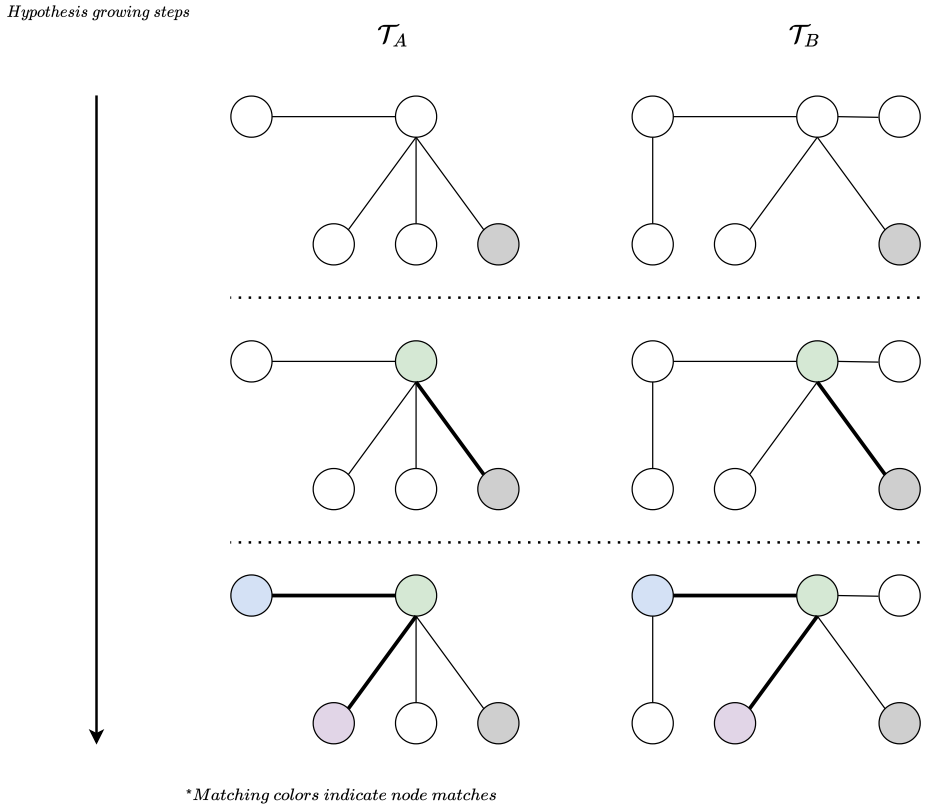

Figure 33: Illustration of hypothesis growing.

4.4 Map Fusion

4.4.1 Overview

The final step of the map merging process is map fusion, in general this means the problem of combining multiple partial maps into a global map. In our case, it specifically refers to the fusion of two partial topometric maps at the geometric and topological level to produce a global topometric map.

To achieve geometric fusion we designate one partial map as the source and the other as the target and find a transformation that aligns the source map with the target map. We constrain the transformation to a rotation around the y-axis and a translation because most 3D scans of indoor environments are gravity-aligned, which means that it is not necessary to consider rotations around the x- or z-axis.

To find the transformation between the partial maps we first find the transformation between each pair of matched rooms. We do so by using RANSAC to find a global alignment which we then refine using the iterative closest point algorithm. Afterwards, we cluster the transformations and find the mean transformation of each cluster. We use the mean transformation which leads to the smallest difference between partial maps as our final rigid transformation. We then apply this transformation to the source map and fuse the geometry of the two maps into a global voxel grid map. Finally, we extract a new, global topometric map from the fused geometry. Figure 34 shows an overview of our map fusion approach. In the rest of this section we will describe our map fusion approach in detail.

4.4.2 Registration

The goal of registration is to find a rigid transformation τ between two point clouds that minimizes the error, as defined by an error function e, between them. For the error function we use point-to-plane distance, as shown in equation 38. In the context of indoor mapping data is usually aligned to gravity, with the direction of gravity being equal to the direction of the negative y-axis. We take this into account by constraining the transformation between partial maps to a translation along all three axes and a rotation around the y-axis. The rigid transformation can thus be expressed as a 4-dimensional vector, as shown in equation 37. Reducing the degrees of freedom of the problem from 6 to 4 can improve the alignment (Kubelka et al., 2022).

$$\tau = \underset{\tau}{argmin}\ e(\mathcal{P}, \mathcal{Q}) = \begin{bmatrix} \gamma \\ \mathbf{t} \end{bmatrix} = \begin{bmatrix} \gamma \\ t_x \\ t_y \\ t_z \end{bmatrix} \tag{37}$$

$$e = \sum_{k=1}^{K} ||((R(\gamma)\mathbf{p_k} + \mathbf{t}) - \mathbf{q_k}) \cdot \mathbf{n_k}||,\ \mathbf{p_k} \in \mathcal{P},\ \mathbf{q_k} \in \mathcal{Q} \tag{38}$$

Kubelka et al. (2022) gives a method for restating gravity-constrained alignment between two sets of points as a system of equations, which is shown in equations 39 and 40 (adapted to use y-axis instead of z-axis as the gravity direction). We can then solve this system of equations using least squares adjustment to find the optimal transformation τ.

$$c_k = (\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \mathbf{p_k}) \cdot \mathbf{n_k} \tag{39}$$

$$\sum_{k=1}^{K} \begin{bmatrix} c_k \\ n_k \end{bmatrix} \begin{bmatrix} c_k & n_k \end{bmatrix} \tau = \sum_{k=1}^{K} \begin{bmatrix} c_k \\ n_k \end{bmatrix} (\mathbf{d_k} \cdot \mathbf{n_k}) \rightarrow \mathbf{A}\tau = \mathbf{b} \tag{40}$$

$$transform : (\mathbb{R}^{m \times 3},\ \mathbb{R}^{m \times 3}) \rightarrow \mathbb{R}^4,\ transform(\mathcal{P},\ \mathcal{Q}) = \tau \tag{41}$$

The above is able to align two point clouds optimally if the correspondence between points is known exactly, meaning that the k-th point of both $\mathcal{P}$ and $\mathcal{Q}$ correspond to the same point in the real world. This is, however, usually not the case in real world scenarios. This means that registration requires another two steps, global and local registration, which we will discuss below.

Global registration The purpose of global registration is to find a rough alignment between point clouds, which can then be further refined in the local registration step. To do so we use a RANSAC based approach based on the work of Koguciuk (2017). This algorithm works as follows. For every point in the point clouds $\mathcal{P}$ and $\mathcal{Q}$ compute a feature embedding that can be used to find similar points in the other point clouds. We use Fast Point Feature Histograms (FPFH) features, which are commonly used for this purpose (Rusu et al., 2009). We then randomly select 3 points from $\mathcal{P}$ and find the corresponding points in $\mathcal{Q}$ which have the smallest distance in feature space. We then compare the triangles formed by both sets of points. If the edge lengths of both triangles are too dissimilar, meaning that the ratio of their lengths is outside of a predetermined range, then the selected points are discarded and new ones are selected. If not, we find the gravity-constrained rigid transformation between the 3 pairs. We then evaluate how well the rigid transformation aligns the two point clouds by finding the mean distance of every point in $\mathcal{P}$ to its nearest neighbour in $\mathcal{Q}$. If the mean distance is smaller than the previous smallest mean distance then we store the transformation. We repeat this for a set amount of iterations or until a mean distance threshold is reached. The rigid transformation with the smallest mean distance is then used for the next step, local registration.

Local registration The purpose of local registration is to refine the alignment found in the global registration step. We use the iterative closest point (ICP) algorithm to achieve this. The only major modification is that we use the gravity-aligned transformation with the point-to-plane error function described above at each iteration.

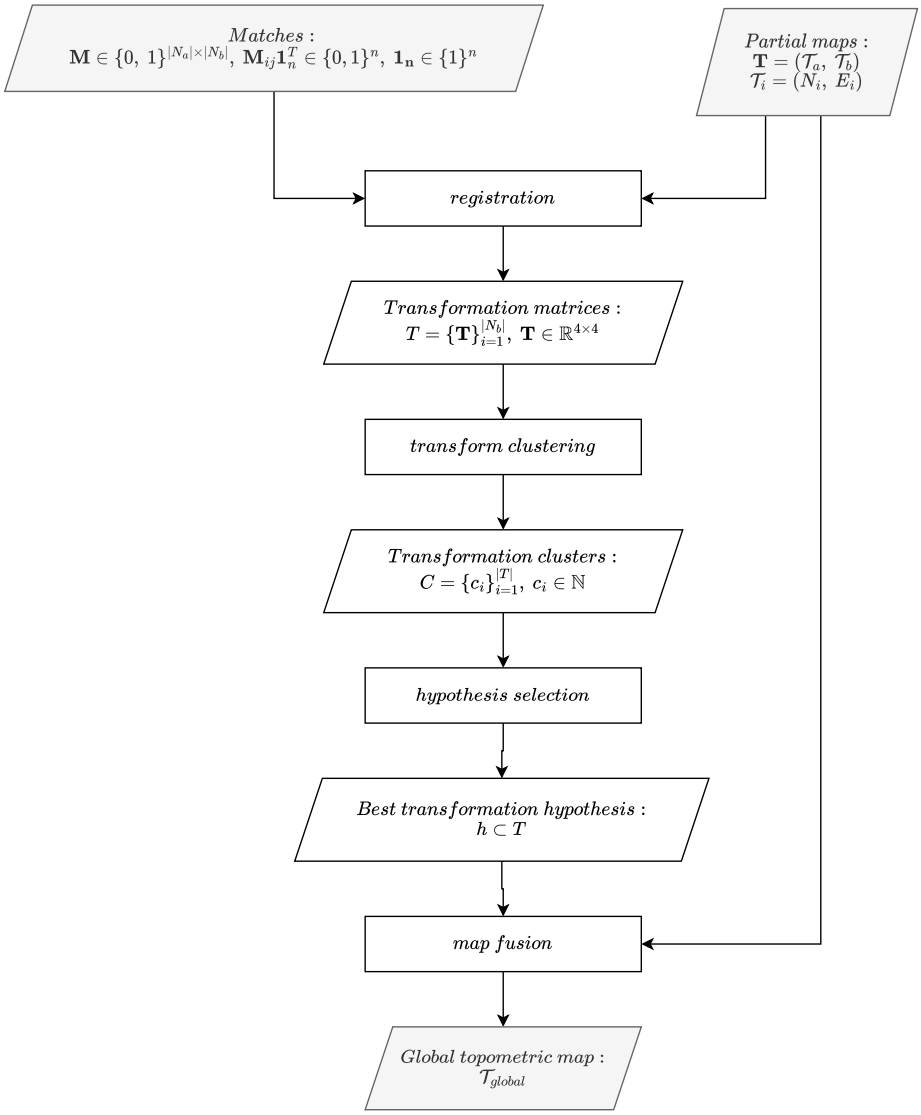

Figure 34: Diagram showing overview of map fusion methodology.

Transform concatenation After finding the global and the local transformations we can find the final transformation τ by multiplying their by 4 by 4 transformation matrices. We denote the function that converts τ to a 4 by 4 transformation matrix as in equation 42. Equation 43 shows how to compute the final transformation matrix, note that the order of multiplication is important, as matrix multiplication is not commutative.

$$T : \mathbb{R}^4 \rightarrow \mathbb{R}^{4\ times4} \tag{42}$$

$$T(\tau) = T(\tau_{local})T(\tau_{global}),\ T : \mathbb{R}^4 \rightarrow \mathbb{R}^{4\times4} \tag{43}$$

4.4.3 Geometric fusion

We apply the above steps to the geometry of each room and its match if their distance in feature space is below a threshold. This gives us a 4 by 4 transformation matrix for each match.

$$\mathbf{T} = \{T(\tau_i)\}_{i=1}^{|\mathbf{M}|},\ T(\tau_i) \in \mathbb{R}^{4\times4} \tag{44}$$

We discard the matches where the registration error is too high and cluster the remaining transformations using the DBSCAN algorithm (Schubert et al., 2017). This gives us multiple clusters of transformations that, within a cluster, result in a similar alignment between partial maps (see equation 45).

$$\mathbf{C} = \{c_i\}_{i=1}^{|\mathbf{T}|},\ c_i \in \mathbb{N} \tag{45}$$

For each of these clusters we find the mean transformation, apply it to the partial map and compute the point-to-plane error between the partial maps. We then select the cluster h with the lowest error and use its mean transformation to align the geometry of the partial topometric maps.

4.4.4 Topological fusion

After the geometric fusion step the geometry of the topometric maps is brought into alignment but their graphs haven't been fused yet. Fusing the graphs directly using the identified matches is possible but does not guarantuee a good global topometric map. This is because the geometric fusion may have changed the partial maps in a way that the global geometric map's topology is greater than the sum of the partial maps' topology. As a result, we have to reextract the topology from the results of the geometric fusion. To do so, we reuse the navigation graphs of the partial maps as the navigation graph of the global map. We also reuse the optimal viewpoints from both partial maps. This means that only the room segmentation step needs to be redone. The result is a global topometric map created from the fusion of two partial topometric maps.

5 Results

We evaluated the approach described in the methodology section on several datasets. In this section we describe these datasets and show the results we achieved with them. The shown results are divided into three sections based on the major components of our methodology: map extraction, map matching and map fusion.

5.1 Datasets

In this section we describe the datasets that we used to test our algorithm. We also describe how we prepared the ground truth datasets so as to be able to compare our results to them and objectively measure their performance.

5.1.1 Simulated Scan

To objectively evaluate the performance of our methodology it is necessary to compare the results to a ground truth. This ground truth must contain the following elements: room labels and their topological relationships to evaluate map extraction, room matches between partial maps to evaluate map matching, and the transformations between partial maps to evaluate map fusion. When capturing real-world data, the second and third elements are especially difficult to determine. Furthermore, little pre-existing data is available that contains exactly these elements. To solve this problem we simulate partial maps from annotated global maps. The annotations are integer labels for every point representing the ground truth room segmentation and a graph representing the ground truth topological map.

We create the partials maps by manually defining a trajectory for each desired partial map and simulating an agent moving along them that scans the global map at a regular interval. We do this by using the FVT algorithm described in the methodology section. This allows us to objectively evaluate our approach for a large number of different scenarios at the cost of missing some of the subtleties inherent in non-simulated partial maps, such as changes in the environment and measurement error. To mitigate this, we apply pre-processing steps to the global map. These pre-processing steps are: random removal of points, adding noise to points and random rotation and translation of the point cloud. The latter's purpose is to simulate the unknown transformation between real-world partial maps. By comparing the results of our approach to the annotations in the ground truth global map we can objectively measure the performance of map extraction, matching and fusion. Figure 35 shows an example global map with simulated viewpoints. Figure 36 shows two simulated partial maps created from a single map.

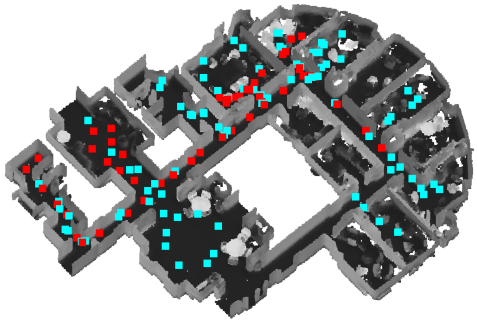

Figure 35: Global map with simulated viewpoints. Each coloured dot represents a viewpoint, the union of all views with the same color forms a partial map.

5.1.2 Stanford 3D Indoor Scene Dataset

The Stanford 3D Indoor Scene Dataset (S3DIS) consists of a collection of 3D scans of six different indoor environments (Area 1 through 6) and the raw measurements used to create them (Armeni et al., 2016). The environments all span a single floor. Each environment scan in its original state consists of a number of point clouds, one for each room in the building. We merge these separate point clouds into a single cloud but retain the room labels as point attributes as we use these as our ground truth room labels for the map extraction step. We manually created the topological graph of each area.

5.1.3 Collaborative SLAM Dataset

The Collaborative SLAM Dataset (CSLAMD) is a dataset meant specifically for collaborative SLAM. It consists of three environments (house, flat and lab), each consisting of multiple partial maps and their ground truth transformations. Two of the environments consist of multiple storeys. The partial maps were captured using a low-end 3D scanner and thus has a low point density. We merge the partial maps of each environment into a single global map to create the simulated partial maps described above. We then manually annotate each environment with our interpretation of an appropriate room segmentation and manually create the topological graph. Due to problems with the quality of this dataset we only use it to demonstrate that our map extraction approach is capable of extracting topometric maps from multi-storey environments.

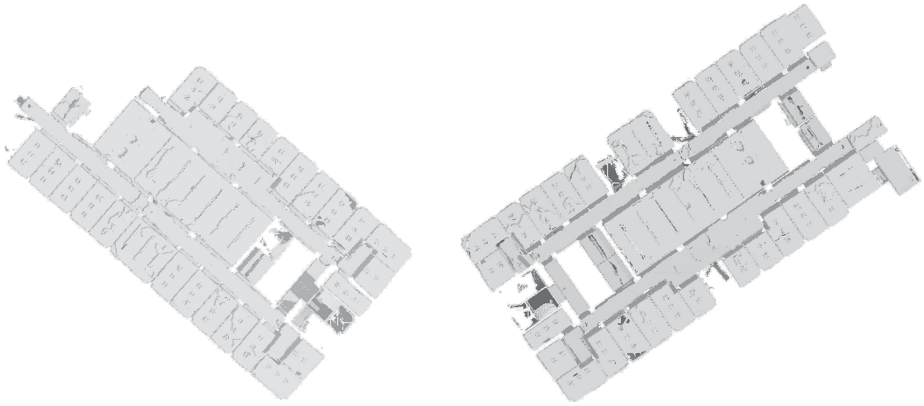

Figure 36: Simulated partial maps extracted from S3DIS area 1 dataset.

5.2 Map Extraction

In this section we show the results of our map extraction approach. To evaluate the results of our map extraction approach we compare the extracted topometric partial maps to the ground truth global map. For each node's geometry in a topometric map we find its Jaccard index with every node in the global map's geometry. This gives us a weighted bipartite graph, one part being the nodes in the partial map and the other being the nodes in the global map, with the weight representing the Jaccard index between nodes' geometry. We assign each node in the partial map to a single node in the global map using linear assignment, as is described in the map matching section. This gives us the correspondence between nodes in the partial map and nodes in the global map.

Room segmentation After finding the correspondences between the partial maps and the ground truth global map we compute the mean Jaccard index of the correspondences. This metric is called Mean Intersection over Union (MIoU) and it measures the quality of our room segmentation. Figure 37 shows the MIoU for each of the partial maps in the S3DIS dataset. Partial maps where map extraction has completely failed are indicated by a dash pattern.

Sparse voxel octree To measure the impact that the sparse voxel octree data structure has on ball query performance we compare its computation time for balls of different radii versus using a hash table to check each if each voxel in the ball is occupied. The results of this measurement are shown in figure 38.

Result maps The results of our map extraction approach for area 1 from the S3DIS dataset are plotted as top-down views in figure 39. Each room is coloured using a unique random colour. The topological edges between rooms are shown as solid black lines. Figure 40 shows a topometric map extracted

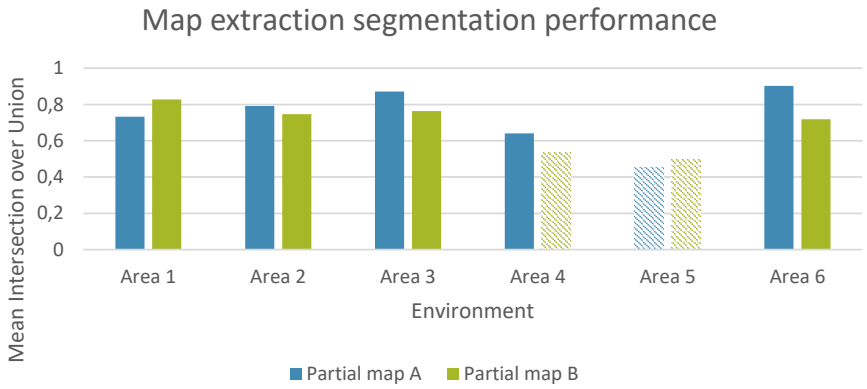

Figure 37: Comparison of map extraction results for the different areas of the S3DIS dataset.

from a multi-storey environment from the CSLAM dataset.

5.3 Map Matching

In this section we show the results of our map matching approach. Using the partial map to global map node correspondences described in the previous section we can distinguish incorrect matches from correct matches; a match between two partial maps is correct if both nodes in the match correspond to the same node in the global map. This is illustrated in figure 41.

Matching precision Figure 42 shows the accuracy of our map matching approaches divided by descriptor type, the number of steps included in the contextual embedding and whether hypothesis growing was used.

Matching results Figure 43 shows the matches between the partial maps of area 1 of the S3DIS dataset, the same partial maps as in the previous sections. Each match is given a unique random colour, with a room in the first partial map having the same colour as its match in the second. These matches were generated using hypothesis growing and a one step contextual embedding with ShapeDNA descriptors

5.4 Map Fusion

In this section we show the results of our map merging approach.

Transformation error We evaluate the performance of our map merging approach by finding the distance between every point in the target map to its closest point in the result map. The results of this are shown in figure 47.

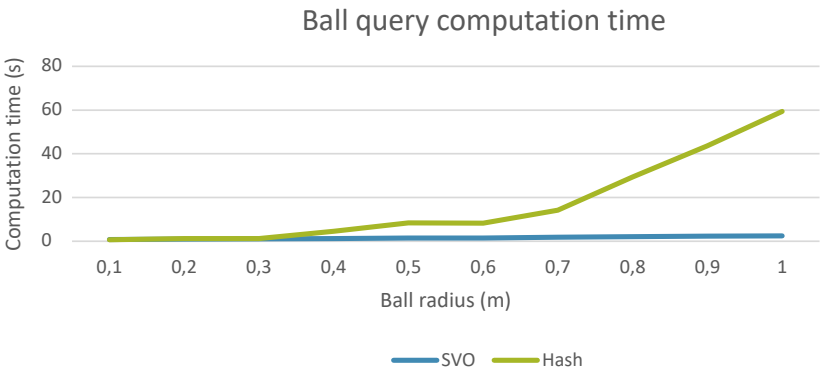

Figure 38: Comparison of SVO radius search performance versus voxel hashing.

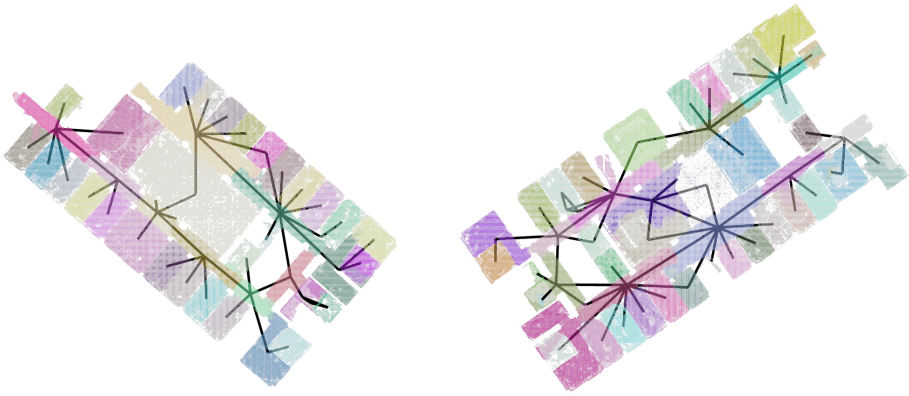

Figure 39: Topometric maps extracted from S3DIS area 1 dataset.

Relationship between descriptor and registration error Figure 44 shows the relationship between the distance in feature space between two rooms and their point-to-plane distance error after registration. This measures the degree to which descriptor similarity predicts how well two rooms align. Both results use two step context embedding.

RANSAC Figure 45 shows the number of optimal transformations that were found by RANSAC within a range of iterations.

ICP Figure 46 shows the relationship between the number of iterations of the iterative closest point algorithm and the point-to-plane error for a large number of matches. Individual iterative closest point runs are coloured in grey, the mean of all runs is coloured in red.

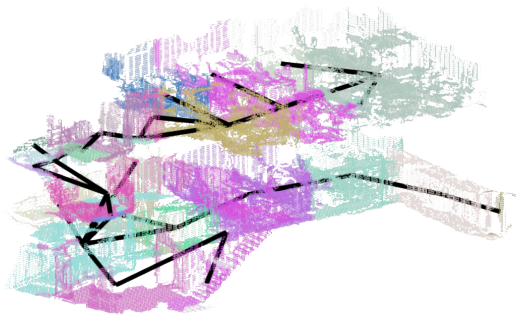

Figure 40: Topometric map extracted from the lab environment from the CSLAM dataset.

Fuse results Figure 48 shows the global map created by fusing the partial maps of area 1 of the S3DIS dataset.

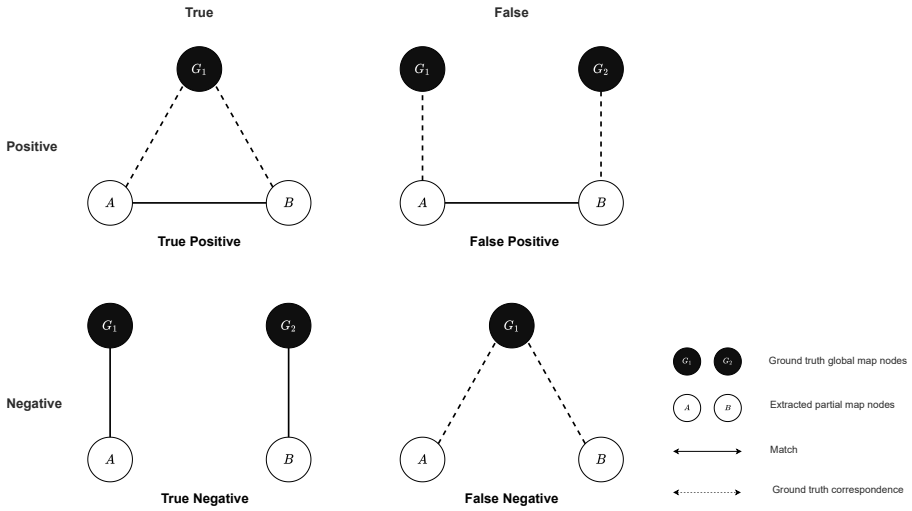

Figure 41: Illustration of map matching results comparison to ground truth.

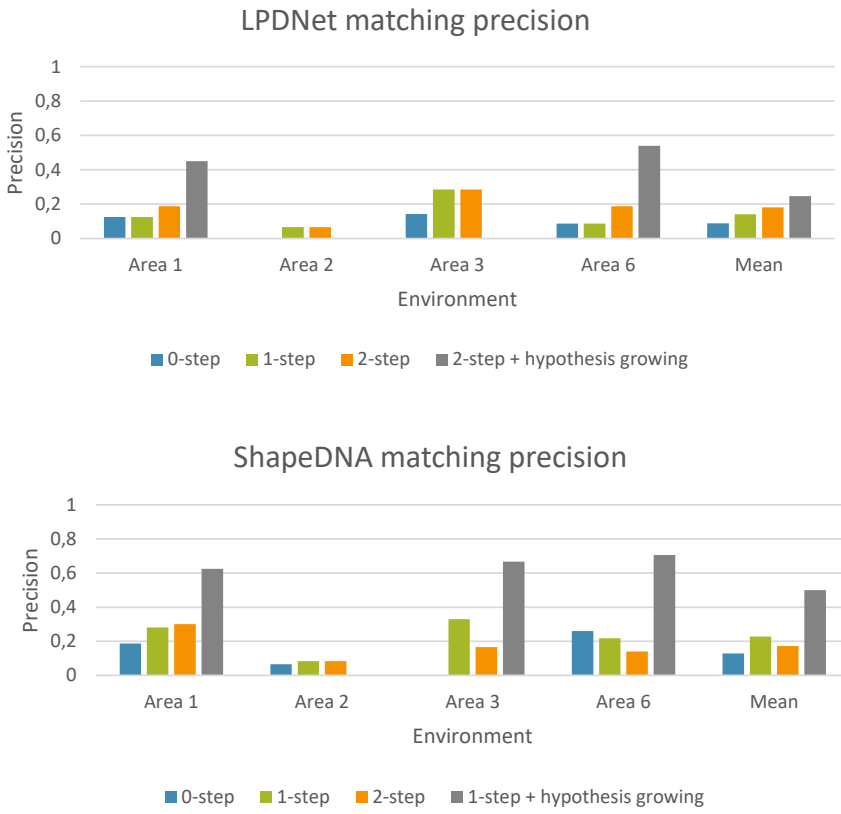

Figure 42: Map matching results for the two descriptor types with different levels of contextual embedding and hypothesis growing.

Figure 43: Identified matches between S3DIS area 1 partial maps using ShapeDNA with one-step contextual embedding and hypothesis growing.

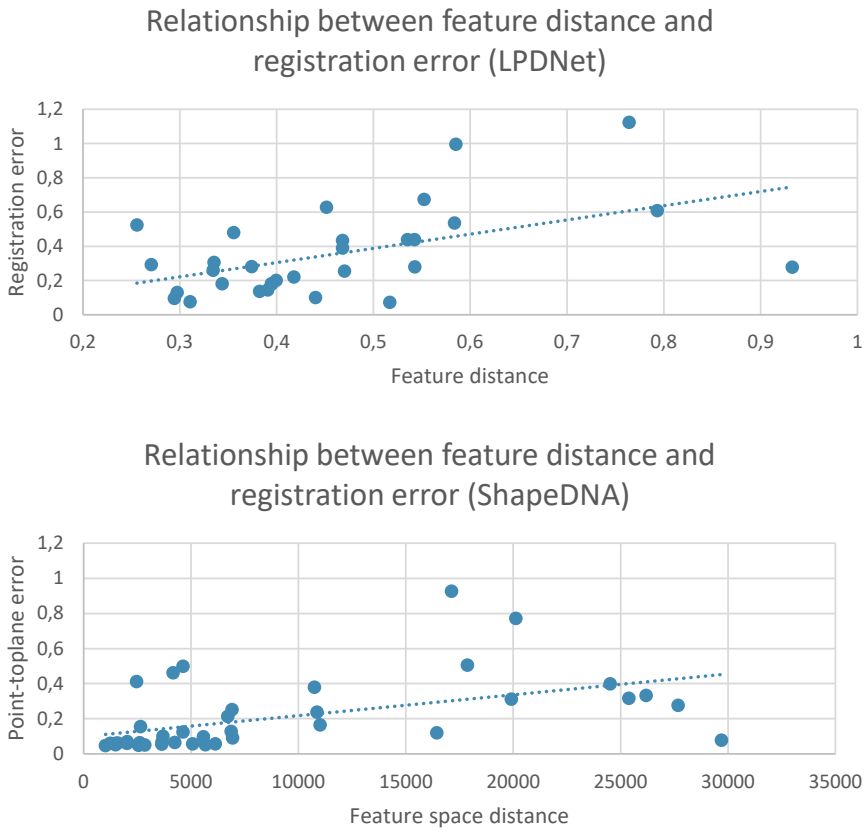

Figure 44: Relationship between descriptor distance in feature space and point-to-plane error after registration.

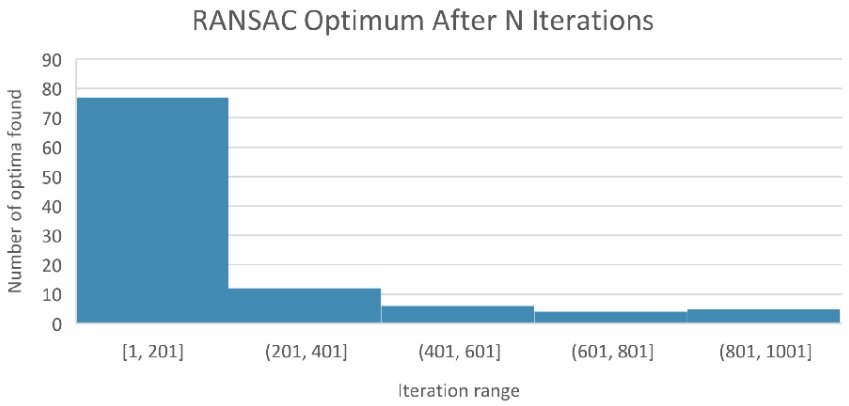

Figure 45: Iteration in which RANSAC finds optimal global registration.

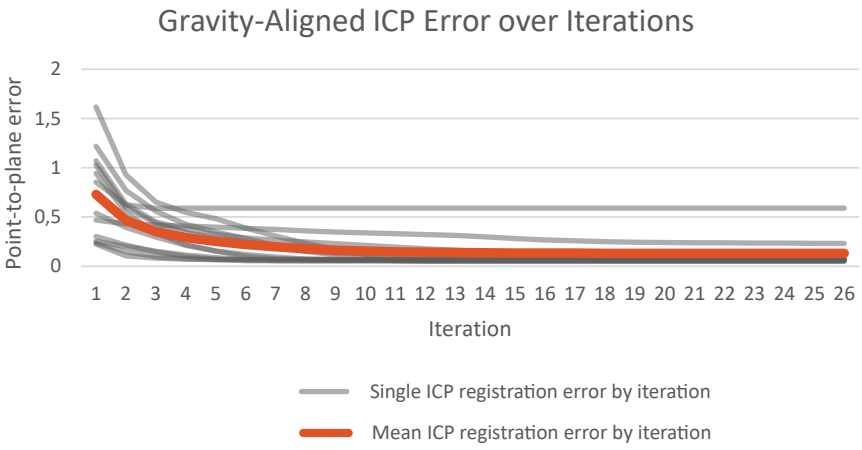

Figure 46: Convergence of ICP algorithm over multiple iterations.

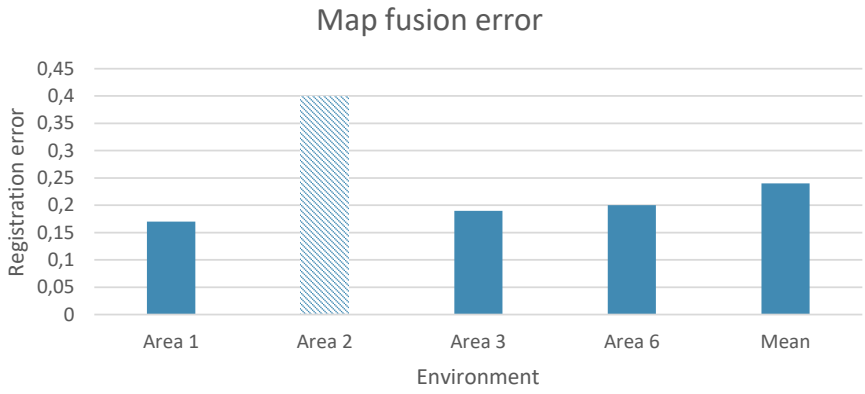

Figure 47: Comparison of point-to-plane error for each area in S3DIS dataset after registration.

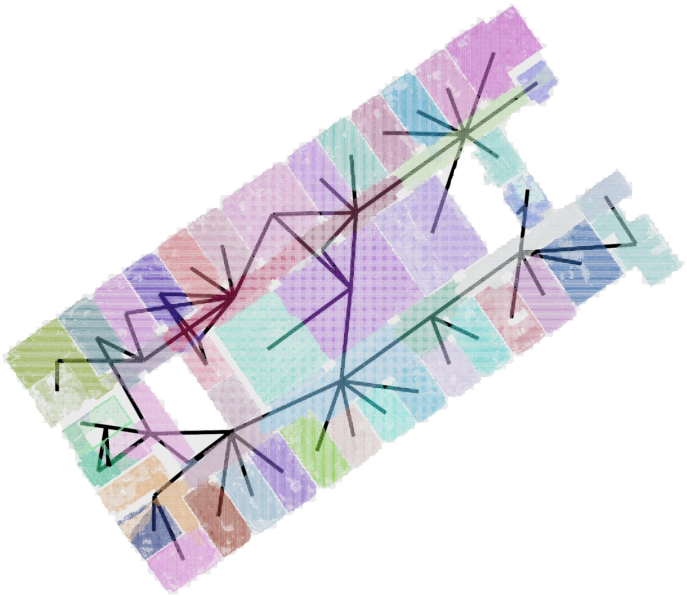

Figure 48: Topometric map created from fusion of the two simulated partial maps of area 1 of the S3DIS dataset.

6 Discussion

In this section we will discuss the results of each of the steps of our methodology.

6.1 Map Extraction

In the previous section we showed the results of our map extraction approach. In this section we will discuss these results. For the majority of tested environments the resultant room segmentation closely matches the ground truth room segmentation. This is especially the case in environments where rooms have clear delineations; environments where rooms have walls between them and are only connected by small openings. The results of our room segmentation approach match less closely in environments where this is not the case. Our approach often splits single rooms that are large in one or multiple dimensions, such as hallways or auditoriums, into multiple rooms. Additionally, rooms where there are obstructions to the view from inside the room are also split into multiple parts. The opposite also occurs, where two or more rooms that are separate in the ground truth data are not separate in the room segmentation. This mostly occurs when there are no obstructions between two adjacent rooms. These effects do not necessarily indicate a failure of our approach. The segmentation in the ground truth data is based on human intuition about what separates a room from its neighbours. Although room segmentation based on visibility clustering often closely matches this intuition it is inherently different as it does not take into account the intended use of rooms. Where humans might recognize that a long hallway or a large hall serves a single purpose, and should therefore be considered as the same room, visibility clustering fails to take this subjective interpretation of purpose into account. Nevertheless, the objective visibility clustering approach comes remarkably close to the subjective human understanding.

A common failure mode of our approach, which causes map extraction to fail completely, is when the input point cloud data is of insufficient density to construct a connected navigation graph. In this case, the voxels belonging to the navigation graph are identified correctly but there are gaps between voxels. This can be solved by increasing the size of the kernel used for constructing the neighbourhood graph of the navigable voxels. However, this has the side effect that voxels that are not actually navigable are added to the navigation graph. The result is that low elevated surfaces with sloping sides, beds for example, are added to the navigation graph. While this usually does not have a large effect on map extraction in extreme cases it can also cause the ceiling to become part of the navigation graph. This will usually cause significant errors in room segmentation, as the view from above the ceiling towards the rest of the map is often completely unobstructed.

Another way that room segmentation may fail is when stairs have very shallow treads and steep rises (respectively the horizontal and vertical part of its steps). This causes the stick kernel approach to fail to label the stairs' voxels as navigable, which means it will not be included in the navigation graph. This is

because the wide part of the stick kernel placed on one step may intersect with the next step. If there are no other connections between two storeys then this will cause one or multiple storeys to become disconnected from the navigation graph, excluding it from the extracted topometric map. This problem can be resolved by changing the dimensions of the stick kernel. However, this may in turn cause other problems. Increasing the height of the thin part of the stick kernel causes low elevated surfaces with sloping sides to be included in the navigation graph, as described in the previous paragraph. Decreasing the radius of the stick kernel's wide part will include parts of the map that are not actually navigable in the navigation graph. The problem can also be solved by increasing the size of the kernel used to construct the navigable voxels' neighbourhood graph to force the stairs' voxels to become connected even though some are missing but this causes the same issues as described in the previous paragraph.

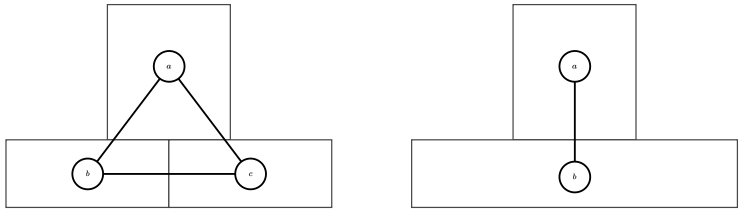

Figure 49: Diagram showing segmentation failure where a hallway and a room meet.

Differences between the ground truth topological graph and the extracted topological graph are often caused by differences in room segmentation. One such case is when a hallway connected to a room is split into multiple rooms around the opening towards the connected room. This will result in a triangular subgraph between the two parts of the hallway and the connected room, which in reality should just be a single edge between the hallway and the room.

6.2 Map Matching

In this section we will discuss the results of our map matching approach.

In the results section we show a comparison between the two different descriptor types. When looking at the baseline map matching precision for both descriptors, meaning no contextual embedding and no hypothesis growing, the ShapeDNA descriptor performs significantly better than LPDNet. However, the accuracy is quite low for both descriptors. In the case of LPDNet this could be caused by a difference between the training dataset and our dataset. Our LPDNet model was trained on point clouds of outdoor environments which have a significantly different appearance than those of indoor environments. Furthermore, each point cloud in the training data is captured from a single point while our data is captured from multiple viewpoints.

We also measured the difference in map matching precision when using contextual embedding versus without. Both descriptors show an improvement in precision when using one-step contextual embedding. LPDNet descriptors also benefit from two-step contextual embedding but ShapeDNA descriptors do not. In both cases two-step contextual embedding still performs better than no contextual embedding.

Based on our results we find that hypothesis growing has the potential to significantly improve map matching performance. However, its performance greatly depends on the quality of the descriptor. If the initial matching is completely incorrect then hypothesis growing also fails. Even if some initial matches are correct, the performance of hypothesis growing still depends on the quality of the descriptor. An exception to this is when the initial match used to grow a hypothesis is at the end of a linear chain of rooms (see figure 50). In this case the growing step will succesfully match all rooms in the chain given that the descriptor between two matches does not fall under the similarity threshold.

We also find that the transformation estimation step is able to constrain region growing to give more reasonable results by preventing matches from being made that would bring the existing matching out of alignment. Adjusting the transformation difference threshold upwards allows the region growing to grow further while increasing the risk that an incorrect match is made. In reverse, adjusting it downwards makes region growing more restrained and decreases the risk of incorrect matches. In the ideal case with no differences between partial maps and their segmentation the threshold could be set to zero as any correct match would align perfectly with the existing matches. However, incompleteness of data and error between partial maps causes the centroids of two matching rooms to be in different positions, introducing error into the alignment. From this it follows that incompleteness, and to a lesser degree error, also has a significant impact on the performance of hypothesis growing.

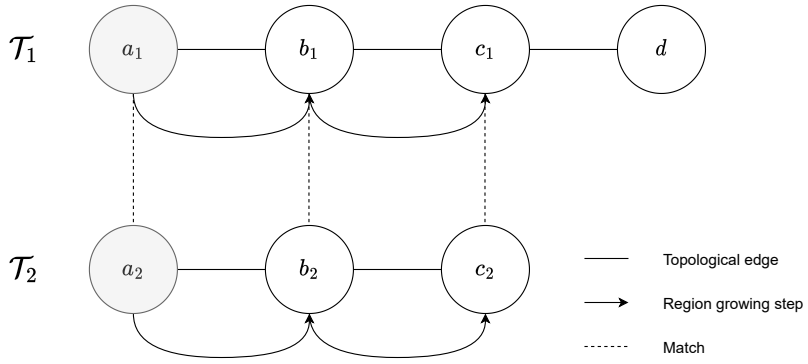

Figure 50: Illustration of hypothesis growing in case of linear chain of rooms.

6.3 Map Fusion

Our results show that our approach to registration is able of finding a good alignment between room matches. We find that there is a positive relationship between the similarity of room descriptors and the final registration error. We further find that many incorrect matches can be filtered out based on their registration error. We also find that most RANSAC and ICP optima are found within 200 and 15 iterations respectively.

Based on the results our approach for registration is able to succesfully find a transformation between partial maps by clustering the transformations between matches. It is able to do so across a wide range of overlaps between partial maps. When no correct matches are identified map fusion fails, but this failure is detectable by its relatively high registration error. Our results also show that our approach is able to fuse the geometry of the partial maps with sufficient accuracy to be able to re-extract a new topometric map from it.

7 Conclusion

In our research, we tried to answer how the properties of 3D topometric maps of indoor environments can be applied to the map merging problem. In this section we will give our conclusion to each of our subquestions that together will answer our main question.

Our first subquestion asked how to extract these topometric maps from point clouds. We find that visibility clustering of synthetic scanning positions using the MCL algorithm, combined with some post-processing steps, can be used for room segmentation. We also find that plausible synthetic scanning positions can be found by computing the local maxima of the environment's navigation graph's horizontal distance field. We further find that this navigation graph can be extracted for multi-storey environments by applying voxel convolution using a stick kernel and that the room segmentation can be combined with the navigation graph to create a topometric map. However, this approach is currently sensitive to map quality, with failures occuring when the map of the environment is incomplete.

Our second subquestion asked how to find matches between partial topometric maps to identify their overlapping areas. After comparing various descriptor approaches we conclude that the spectral approach approach gives the best results. We also find that embedding the context of a room into the room's descriptor improves matching performance. So does hypothesis growing in the average case, but it may also cause map matching to fail completely. Based on the above we further conclude that the topological aspect of topometric maps can be used to increase map matching performance. Possible failure modes of this approach include similar rooms which have similar contexts and differences in segmentation between partial maps.

Our third subquestion asked how to fuse the partial topometric maps after matches have been identified. We find that a combination of RANSAC based global registration using FPFH features and gravity-aligned ICP local registration can succesfully identify the transformation between rooms. We also find that DBSCAN can be used to cluster these transformations based on their similarity and that a correct final transformation can be computed based on this clustering.

With the above conclusions we can answer our main research question: partial topometric maps can be extracted from partial point clouds by using visibility clustering and voxel convolution. Matches can be found using a combination of spectral descriptors, contextual embedding and hypothesis growing. By using these matches we can succesfully find the transformation between rooms, from which the final transformation can be found by clustering them using DBSCAN. Based on our conclusions we identify multiple avenues for future research, for which the focus should lie on robust topometric map extraction, improved room descriptors and non-rigid map fusion.

8 Future Works

In this section we discuss our recommendations for future developments of the three major components of this thesis: map extraction, map matching and map fusion. We make these recommendations based on the achieved results and our research during the creation of this thesis.

8.1 Map extraction

Room segmentation As mentioned before, the current room segmentation approach differs from how humans identify rooms as it does not take into account the perceived purpose of the room. Taking both visibility and purpose into account could lead to a segmentation that more closely matches one a human would perform, but more importantly, one that is more consistent between partial maps and more robust to incompleteness and error. Assuming that a computer can succesfully infer a room's purpose based on its voxelized representation, large rooms such as hallways that are now arbitrarily divided based on clustering could be merged into one whole. Inferring a room's purpose is a subjective task that would be very hard to solve using traditional techniques. To achieve this, we suggest training a deep learning model that is suitable for segmentation of voxel or point cloud representations on a manually labeled ground truth dataset.

Robust topological graph extraction One of the major bottlenecks of our approach is the extraction of the topological graph. If this step fails then map extraction fails and map matching becomes impossible. Thus, in the future it would be important to identify an approach to topological graph extraction that is robust to the failure modes described in section 6.1. This would include a way to interpolate the navigation graph to fill in any missing holes that cause it to become disconnected. The main challenge here is differentiating between voxels that should be present but are missing and voxels that should not be; making the wrong choice could lead to worse results than no interpolation at all. For example, a small gap in the floor can be either a piece of missing data or a gap between walls. Solving this challenge could drastically improve the robustness of our map extraction approach.

Hierarchical topological representation Our current approach to map extraction results in a topometric map with a 'flat' graph representing the environment's topology. In reality, indoor environments can be considered as complex multi-level hierarchies. For example, a building can be divided into storeys which contain rooms which contain areas. As such, the structure of an indoor environment can also be represented by a hierarchical graph. The extra information contained in such a graph could be applied to improve feature embedding performance. For example, two rooms are more similar if their storeys are also similar than if they are not. Various techniques have been proposed in the literature surrounding this subject but none so far are based on visibility

clustering. We hypothesize that by applying hierarchical clustering to visibility it is possible to extract a hierarchical structure of the environment. Whether this is true and what the characteristics of the resultant map are could be a valuable topic of research. A hierarchical topological representation could allow or require different methods for hypothesis growing and map fusion. For the former, work in the area of hierarchical graph matching could be applied. The latter is, to the knowledge of the author, still unresearched.

Volumetric representation Our current approach only represents the surface of the geometry of the environment. This is because 3D scanners only capture that part of the environment. In reality, indoor environments are enclosed volumes. A possible improvement to our approach would be to describe the environment's geometry volumetrically, where each occupied voxel represents a volume within the building that is not obstructed. This would require a method to extract the volume from the surface geometry. Various research into this topic exists but they fail when parts of the ceiling or floor are missing from the map, which is often the case. They also make assumptions such as constant storey height and only horizontal floors, which are often not the case in reality. Using a volumetric representation of the environment has a number of benefits. Navigation would no longer only be possible on the floor but throughout the entire volume. In reality most scanners use the floor to navigate but the advent of drones that operate indoors might change this. Another avenue of research that moving to a volumetric approach would require would concern efficiently storing and processing the exponentially larger amount of data used in doing so. While this subject has been considered in this research by using sparse voxel octrees, variations on this or other data structures might be more effective. For example, implementations of sparse voxel octrees for GPUs exist.

8.2 Map Matching

Deep learning descriptors Most current research points to deep learning as the current state of the art for place recognition. This does not conform to the results of our research. Future research could look into applying different model architectures and compare their performance. Alternatively, if it is concluded that the problem is not related to model architecture then an existing model could be trained on manually labelled indoor data.

Advanced spectral shape descriptors ShapeDNA is a relatively early and simple method for spectral feature embedding. More recent research has suggested alternatively approaches that are purported to have better performance for incomplete geometry. These approaches have steep hardware requirements which were not available for this thesis. Future research should include these approaches and look into their computational feasibility.

Graph features Currently our descriptors are based on aggegrating the geometric descriptors within a neighbourhood. A more complete descriptor could also include the node's graph properties, such as its degree or centrality. Future research should look into which graph features are best used for this and whether the topology is consistent enough between partial maps to make it useful.

Multiway matching In our approach, every node in one partial map is matched to at most one node in the other partial map. In practice the segmentation is not perfectly equal between partial maps. This leads to cases where one node should reasonably be matched with multiple nodes. This would greatly complicate both map matching and fusion but could lead to better results. Future research should investigate methods for multi-node map matching and its effect on map matching performance.

8.3 Map Fusion

Non-rigid registration Our approach assumes that the partial maps are only transformed rigidly. While this is a reasonable assumption to a dregree, it doesn't always hold for real-world data. Measurement error and error introduced during data processing can lead to deformations between partial maps that persist after rigid registration. These deformations could be corrected during map merging by finding a non-rigid registration between the partial maps, which deforms the source partial map in such a way that every matched room overlaps optimally. Future research should look into methods for estimating and applying non-rigid transformations and their effect on map fusion performance.

Global registration Our current approach uses FPFH features to match points during RANSAC global registration. This choice was made because an implementation was readily available and their computation is fast. Future research should try different approaches for local feature embedding, such as deep learning or spectral methods. A local feature with sufficient quality could make the local registration step completely unnecessary because enough correspondences are known between partial maps to register them directly.

9 Reflection

In this section I will give a reflection of my process in creating this thesis and the final results. I started working on this thesis in the summer of 2021. For the first few months the goal of the thesis wasn't entirely clear. I knew I wanted to research map merging but didn't know in which context I would do so. I also felt that I had trouble aligning my view of what I wanted to research with those of my supervisors. This might have been made worse by the fact that I had three supervisors with significantly different perspectives. My own frustration with my slow progress demotivated me, alhtough not to the point of not getting any work done. This problem continued until shortly before my P2. Although at that point my methodology was not nearly defined yet I knew what I wanted to research. Unfortunately, this did not leave me with enough time to polish my P1 report to my own satisfaction. I was especially dissatisfied with the maths part of the report because I had very little experience with this. Nevertheless, my P1 was well received by my supervisors, although they did express their concerns with the amount of work that needed to be done. I noted that the amount of work was large, but not impossible for me to complete.

Between my P2 and P3 I mostly worked on map extraction component. The good results achieved here motivated me to make a large amount of progress during this time. However, at this point my approach to map matching and map fusion were still unclear to me. In hindsight, I should have defined my entire methodology before implementing any parts. The problem with that is that I enjoy programming much more than writing or researching, and I was unable to resist the temptation to work on what I enjoy instead of what needed to be done. I completed my work on the map extraction component before my P3, which was well received.

After my P3 I needed to start working on the map matching and map fusion components. In contract to my work on map extraction, progress on map matching was very slow. Although my initial idea for my methodology wasn't far off from my final one, I had a lot of trouble getting a working implementation. Approaches that in my mind should work, in fact did not. As my P4 date approached I still had not finished the map matching component, let alone the map fusion component. Coincidentally, about a month before my P4 date I was told that I did not have the required ECTS to participate, and that I would have to wait until next semester to do my P4. This gave me some relief, as I now had the time to properly finish what I started. Between my initial P4 date and my actual one I mostly worked on getting map matching and map fusion working, which I did eventually. Unfortunately, this did not leave me with much time to finish this report. I worked hard to get everything written down, which I feel like I succeeded at, but there were still a lot of things that I would have liked to add. My P4 was again well received, with most comments being about mathematical notation, which I have fixed for my P5 document. Overall, I am satisfied with my results but not with my process. I took on too much work to finish in the allotted time and often left parts that I did not enjoy to the last moment. Nevertheless, I feel I have learned from the experience the

things that I wanted to, which for me was the ultimate goal of this project. I am also happy with the contribution of my supervisors, who had to spend a lot of time listening to me talk about my various ideas. Although their multitude of perspectives at some points led to frustration on my part, I feel that in the end it contributed positively to my process and results.

References

Amanatides, J., & Woo, A. (1987). A Fast Voxel Traversal Algorithm for Ray Tracing. , 6.

Ambrus, R., Claici, S., & Wendt, A. (2017, April). Automatic Room Segmentation From Unstructured 3-D Data of Indoor Environments. *IEEE Robotics and Automation Letters*, *2*(2), 749–756. Retrieved 2022-09-29, from `http://ieeexplore.ieee.org/document/7814251/` doi: 10.1109/LRA.2017.2651939

Andersone. (2019, August). Heterogeneous Map Merging: State of the Art. *Robotics*, *8*(3), 74. Retrieved 2021-07-16, from `https://www.mdpi.com/2218-6581/8/3/74` doi: 10.3390/robotics8030074

Aoki, Y., Goforth, H., Srivatsan, R. A., & Lucey, S. (2019). PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet. In (pp. 7163–7172).

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2016). NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. In (pp. 5297–5307).

Arandjelovic, R., & Zisserman, A. (2013). All About VLAD. In (pp. 1578–1585).

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., & Savarese, S. (2016, June). 3D Semantic Parsing of Large-Scale Indoor Spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1534–1543). Las Vegas, NV, USA: IEEE. Retrieved 2022-09-30, from `http://ieeexplore.ieee.org/document/7780539/` doi: 10.1109/CVPR.2016.170

Bonanni, T. M., Della Corte, B., & Grisetti, G. (2017, April). 3-D Map Merging on Pose Graphs. *IEEE Robotics and Automation Letters*, *2*(2), 1031–1038. Retrieved 2021-11-09, from `https://ieeexplore.ieee.org/document/7822998/` doi: 10.1109/LRA.2017.2655139

Bonnassie, A., Peyrin, F., & Attali, D. (2003, August). A new method for analyzing local shape in three-dimensional images based on medial axis transformation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *33*(4), 700–705. (Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)) doi: 10.1109/TSMCB.2003.814298

Bormann, R., Jordan, F., Li, W., Hampp, J., & Hagele, M. (2016, May). Room segmentation: Survey, implementation, and analysis. In *2016 IEEE International Conference on Robotics and Automation*

(ICRA) (pp. 1019–1026). Stockholm, Sweden: IEEE. Retrieved 2022-02-08, from https://ieeexplore.ieee.org/document/7487234/ doi: 10.1109/ICRA.2016.7487234

Bot, F. J., Nourian, P., & Verbree, E. (2019, June). A Graph-Matching Approach To Indoor Localization Using A Mobile Device And A Reference BIM. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-2/W13*, 761–767. Retrieved 2021-06-18, from https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W13/761/2019/ doi: 10.5194/isprs-archives-XLII-2-W13-761-2019

Bronstein, M. M., & Kokkinos, I. (2010, June). Scale-invariant heat kernel signatures for non-rigid shape recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 1704–1711). (ISSN: 1063-6919) doi: 10.1109/CVPR.2010.5539838

Chen, J., & Clarke, K. C. (2020, March). Indoor cartography. *Cartography and Geographic Information Science*, *47*(2), 95–109. Retrieved 2022-09-29, from https://doi.org/10.1080/15230406.2019.1619482 (Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/15230406.2019.1619482) doi: 10.1080/15230406.2019.1619482

Dubé, R., Dugas, D., Stumm, E., Nieto, J., Siegwart, R., & Cadena, C. (2017, May). SegMatch: Segment based place recognition in 3D point clouds. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 5266–5272). doi: 10.1109/ICRA.2017.7989618

Dudek, G., Jenkin, M., Milios, E., & Wilkes, D. (1998). Topological Exploration With Multiple Robots. , 6.

Elfes, A. (1990). Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception. *arXiv:1304.1098 [cs]*. Retrieved 2021-12-21, from http://arxiv.org/abs/1304.1098 (arXiv: 1304.1098)

Garcia-Fidalgo, E., & Ortiz, A. (2017, October). Hierarchical Place Recognition for Topological Mapping. *IEEE Transactions on Robotics*, *33*(5), 1061–1074. Retrieved 2022-09-29, from http://ieeexplore.ieee.org/document/7938750/ doi: 10.1109/TRO.2017.2704598

Gholami Shahbandi, S., & Magnusson, M. (2019, June). 2D map alignment with region decomposition. *Autonomous Robots*, *43*(5), 1117–1136. Retrieved 2021-11-09, from http://link.springer.com/10.1007/s10514-018-9785-7 doi: 10.1007/s10514-018-9785-7

Gorte, B., Zlatanova, S., & Fadli, F. (2019, May). NAVIGATION IN INDOOR VOXEL MODELS. *ISPRS Annals*

of the Photogrammetry, Remote Sensing and Spatial Information Sciences, *IV-2/W5*, 279–283. Retrieved 2021-12-06, from https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/IV-2-W5/279/2019/ doi: 10.5194/isprs-annals-IV-2-W5-279-2019

Han, X.-F., Jin, J. S., Xie, J., Wang, M.-J., & Jiang, W. (2018). A comprehensive review of 3D point cloud descriptors. , 44.

He, Z., Sun, H., Hou, J., Ha, Y., & Schwertfeger, S. (2021, June). Hierarchical topometric representation of 3D robotic maps. *Autonomous Robots*, *45*(5), 755–771. Retrieved 2021-11-09, from https://link.springer.com/10.1007/s10514-021-09991-8 doi: 10.1007/s10514-021-09991-8

Hermann, M., Pentek, T., & Otto, B. (2016, January). Design Principles for Industrie 4.0 Scenarios. In *2016 49th Hawaii International Conference on System Sciences (HICSS)* (pp. 3928–3937). (ISSN: 1530-1605) doi: 10.1109/HICSS.2016.488

Huang, W. H., & Beevers, K. R. (2005, August). Topological Map Merging. *The International Journal of Robotics Research*, *24*(8), 601–613. Retrieved 2021-11-29, from http://journals.sagepub.com/doi/10.1177/0278364905056348 doi: 10.1177/0278364905056348

Jonker, R., & Volgenant, A. (1987, December). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, *38*(4), 325–340. Retrieved 2022-09-30, from https://doi.org/10.1007/BF02278710 doi: 10.1007/BF02278710

Koguciuk, D. (2017, September). Parallel RANSAC for Point Cloud Registration. *Foundations of Computing and Decision Sciences*, *42*(3), 203–217. Retrieved 2022-09-05, from https://www.sciendo.com/article/10.1515/fcds-2017-0010 doi: 10.1515/fcds-2017-0010

Komorowski, J. (2021). MinkLoc3D: Point Cloud Based Large-Scale Place Recognition. In (pp. 1790–1799).

Kubelka, V., Vaidis, M., & Pomerleau, F. (2022, March). *Gravity-constrained point cloud registration.* arXiv. Retrieved 2022-08-29, from http://arxiv.org/abs/2203.13799 (arXiv:2203.13799 [cs])

Kuipers, B. (1978, April). Modeling spatial knowledge. *Cognitive Science*, *2*(2), 129–153. Retrieved 2022-01-10, from https://www.sciencedirect.com/science/article/pii/S0364021378800032 doi: 10.1016/S0364-0213(78)80003-2

Kuipers, B., & Byun, Y.-T. (1988). A Robust, Qualitative Method for Robot Spatial Learning.

Kuipers, B., & Byun, Y.-T. (1991, November). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, *8*(1), 47–63. Retrieved 2022-01-10, from https://www.sciencedirect.com/science/article/pii/092188909190014C doi: 10.1016/0921-8890(91)90014-C

Lajoie, P.-Y., Ramtoula, B., Wu, F., & Beltrame, G. (2022, March). Towards Collaborative Simultaneous Localization and Mapping: a Survey of the Current Research Landscape. *Field Robotics*, *2*(1), 971–1000. Retrieved 2022-09-30, from http://arxiv.org/abs/2108.08325 (arXiv:2108.08325 [cs]) doi: 10.55417/fr.2022032

Li, Y., & Olson, E. B. (2010, November). A General Purpose Feature Extractor for Light Detection and Ranging Data. *Sensors*, *10*(11), 10356–10375. Retrieved 2022-01-24, from https://www.mdpi.com/1424-8220/10/11/10356 (Number: 11 Publisher: Molecular Diversity Preservation International) doi: 10.3390/s101110356

Liu, L., Chambers, E. W., Letscher, D., & Ju, T. (2011, November). Extended grassfire transform on medial axes of 2D shapes. *Computer-Aided Design*, *43*(11), 1496–1505. Retrieved 2022-10-31, from https://www.sciencedirect.com/science/article/pii/S0010448511002338 doi: 10.1016/j.cad.2011.09.002

Liu, Z., Zhou, S., Suo, C., Yin, P., Chen, W., Wang, H., ... Liu, Y. (2019, October). LPD-Net: 3D Point Cloud Learning for Large-Scale Place Recognition and Environment Analysis. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 2831–2840). Seoul, Korea (South): IEEE. Retrieved 2022-09-29, from https://ieeexplore.ieee.org/document/9009029/ doi: 10.1109/ICCV.2019.00292

Ma, J. W., Czerniawski, T., & Leite, F. (2020, May). Semantic segmentation of point clouds of building interiors with deep learning: Augmenting training datasets with synthetic BIM-based point clouds. *Automation in Construction*, *113*, 103144. Retrieved 2022-09-29, from https://linkinghub.elsevier.com/retrieve/pii/S0926580519311884 doi: 10.1016/j.autcon.2020.103144

Mille, J., Leborgne, A., & Tougne, L. (2019, March). Euclidean Distance-Based Skeletons: A Few Notes on Average Outward Flux and Ridgeness. *Journal of Mathematical Imaging and Vision*, *61*(3), 310–330. Retrieved 2022-10-31, from http://link.springer.com/10.1007/s10851-018-0836-7 doi: 10.1007/s10851-018-0836-7

Mura, C., Mattausch, O., Jaspe Villanueva, A., Gobbetti, E., & Pajarola, R. (2014, November). Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics*, *44*, 20–32. Retrieved 2022-02-17, from

https://linkinghub.elsevier.com/retrieve/pii/S0097849314000661 doi: 10.1016/j.cag.2014.07.005

Mura, C., Mattausch, O., & Pajarola, R. (2016, October). Piecewise-planar Reconstruction of Multi-room Interiors with Arbitrary Wall Arrangements. *Computer Graphics Forum*, *35*(7), 179–188. Retrieved 2022-09-29, from https://onlinelibrary.wiley.com/doi/10.1111/cgf.13015 doi: 10.1111/cgf.13015

Ochmann, S., Vock, R., Wessel, R., & Klein, R. (2014). Towards the Extraction of Hierarchical Building Descriptions from 3D Indoor Scans. , 8.

Phan, A. V., Nguyen, M. L., Nguyen, Y. L. H., & Bui, L. T. (2018, December). DGCNN: A convolutional neural network over large-scale labeled graphs. *Neural Networks*, *108*, 533–543. Retrieved 2022-09-29, from https://www.sciencedirect.com/science/article/pii/S0893608018302636 doi: 10.1016/j.neunet.2018.09.001

Pintore, G., Mura, C., Ganovelli, F., Fuentes-Perez, L., Pajarola, R., & Gobbetti, E. (2020). State-of-the-art in Automatic 3D Reconstruction of Structured Indoor Environments. *Computer Graphics Forum*, *39*(2), 667–699. Retrieved 2022-02-16, from https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14021 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14021) doi: 10.1111/cgf.14021

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In (pp. 652–660).

Queralta, J. P., Taipalmaa, J., Pullinen, B. C., Sarker, V. K., Gia, T. N., Tenhunen, H., ... Westerlund, T. (2020, August). *Collaborative Multi-Robot Systems for Search and Rescue: Coordination and Perception.* arXiv. Retrieved 2022-09-29, from http://arxiv.org/abs/2008.12610 (arXiv:2008.12610 [cs])

Reuter, M., Wolter, F.-E., & Peinecke, N. (2006, April). Laplace–Beltrami spectra as 'Shape-DNA' of surfaces and solids. *Computer-Aided Design*, *38*(4), 342–366. Retrieved 2022-09-29, from https://www.sciencedirect.com/science/article/pii/S0010448505001867 doi: 10.1016/j.cad.2005.10.011

Rincon, J. L. S., & Carpin, S. (2019, August). Map Merging of Oriented Topological Semantic Maps. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)* (pp. 202–208). doi: 10.1109/MRS.2019.8901093

Rozemberczki, B., Allen, C., & Sarkar, R. (2021, March). *Multi-scale Attributed Node Embedding.* arXiv. Retrieved 2022-09-29, from http://arxiv.org/abs/1909.13021 (arXiv:1909.13021 [cs, stat])

Rusinkiewicz, S., & Levoy, M. (2001, May). Efficient variants of the ICP algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling* (pp. 145–152). doi: 10.1109/IM.2001.924423

Rusu, R. B., Blodow, N., & Beetz, M. (2009, May). Fast Point Feature Histograms (FPFH) for 3D registration. In *2009 IEEE International Conference on Robotics and Automation* (pp. 3212–3217). (ISSN: 1050-4729) doi: 10.1109/ROBOT.2009.5152473

Rusu, R. B., & Cousins, S. (2011, May). 3D is here: Point Cloud Library (PCL). In *2011 IEEE International Conference on Robotics and Automation* (pp. 1–4). (ISSN: 1050-4729) doi: 10.1109/ICRA.2011.5980567

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017, July). DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems*, *42*(3), 19:1–19:21. Retrieved 2022-09-30, from `https://doi.org/10.1145/3068335` doi: 10.1145/3068335

Schuster, M. J., Müller, M. G., Brunner, S. G., Lehner, H., Lehner, P., Sakagami, R., ... Wedler, A. (2020, October). The ARCHES Space-Analogue Demonstration Mission: Towards Heterogeneous Teams of Autonomous Robots for Collaborative Scientific Sampling in Planetary Exploration. *IEEE Robotics and Automation Letters*, *5*(4), 5315–5322. (Conference Name: IEEE Robotics and Automation Letters) doi: 10.1109/LRA.2020.3007468

Serafin, J., & Grisetti, G. (2015, September). NICP: Dense normal based point cloud registration. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 742–749). doi: 10.1109/IROS.2015.7353455

Shan, T., Englot, B., Duarte, F., Ratti, C., & Rus, D. (2021, May). Robust Place Recognition using an Imaging Lidar. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 5469–5475). (ISSN: 2577-087X) doi: 10.1109/ICRA48506.2021.9562105

Tang, S., Li, X., Zheng, X., Wu, B., Wang, W., & Zhang, Y. (2022, September). BIM generation from 3D point clouds by combining 3D deep learning and improved morphological approach. *Automation in Construction*, *141*, 104422. Retrieved 2022-09-29, from `https://linkinghub.elsevier.com/retrieve/pii/S0926580522002953` doi: 10.1016/j.autcon.2022.104422

Thrun, S. (1998, February). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, *99*(1), 21–71. Retrieved 2022-01-10, from `https://www.sciencedirect.com/science/article/pii/S0004370297000787` doi: 10.1016/S0004-3702(97)00078-7

Tomatis, N., Nourbakhsh, I., & Siegwart, R. (2003, July). Hybrid simultaneous localization and map building: a natural integration of topological and metric.

Robotics and Autonomous Systems, *44*(1), 3–14. Retrieved 2021-12-20, from https://linkinghub.elsevier.com/retrieve/pii/S092188900300006X doi: 10.1016/S0921-8890(03)00006-X

Uy, M. A., & Lee, G. H. (2018). PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition. In (pp. 4470–4479).

van Dongen. (2000, January). *A cluster algorithm for graphs.* CWI. Retrieved 2022-09-30, from https://ir.cwi.nl/pub/4463 (Issue: R 0010 Publication Title: Information Systems [INS])

Volodine, T. (2007). *Point Cloud Processing Using Linear Algebra and Graph Theory* (Unpublished doctoral dissertation).

Wang, C., Wang, J., Li, C., Ho, D., Cheng, J., Yan, T., ... Meng, M. Q.-H. (2019, January). Safe and Robust Mobile Robot Navigation in Uneven Indoor Environments. *Sensors*, *19*(13), 2993. Retrieved 2022-09-29, from https://www.mdpi.com/1424-8220/19/13/2993 (Number: 13 Publisher: Multidisciplinary Digital Publishing Institute) doi: 10.3390/s19132993

Wang, Y., & Solomon, J. M. (2019). Deep Closest Point: Learning Representations for Point Cloud Registration. In (pp. 3523–3532).

Yang, J., Cao, Z., & Zhang, Q. (2016, June). A fast and robust local descriptor for 3D point cloud registration. *Information Sciences*, *346-347*, 163–179. Retrieved 2022-01-18, from https://www.sciencedirect.com/science/article/pii/S0020025516300378 doi: 10.1016/j.ins.2016.01.095

Yu, S., Fu, C., Gostar, A. K., & Hu, M. (2020, December). A Review on Map-Merging Methods for Typical Map Types in Multiple-Ground-Robot SLAM Solutions. *Sensors*, *20*(23), 6988. Retrieved 2021-07-12, from https://www.mdpi.com/1424-8220/20/23/6988 doi: 10.3390/s20236988