

Reconstruction and Prediction of the Layout of Indoor Environments from Two-Dimensional Metric Maps

Matteo Luperto^a, Francesco Amigoni^b

^a*Dipartimento di Informatica, Università degli Studi di Milano*

^b*Dipartimento di Elettronica, Informatica e Bioingegneria, Politecnico di Milano*

Abstract

Metric maps, like occupancy grids, are one of the most common ways to represent indoor environments in autonomous mobile robotics. Although they are effective for navigation and localization, metric maps contain little knowledge about the structure of the buildings they represent. In this paper, we propose a method that identifies the structure of indoor environments from 2D metric maps by retrieving their *layout*, namely an abstract geometrical representation that models walls as line segments and rooms as polygons. The method works by finding regularities within a building, abstracting from the possibly noisy information of the metric map, and uses such knowledge to reconstruct the layout of the observed part and to predict a possible layout of the partially observed portion of the building. Thus, differently of other methods from the state of the art, our method can be applied both to fully observed environments and, most significantly, to partially observed ones. Experimental results show that our approach performs effectively and robustly on different types of input metric maps and that the predicted layout is increasingly more accurate when the input metric map is increasingly more complete. The layout returned by our method can be exploited in several tasks, such as semantic mapping, place categorization, path planning, human-robot communication, and task allocation.

Email addresses: `matteo.luperto@unimi.it` (Matteo Luperto),
`francesco.amigoni@polimi.it` (Francesco Amigoni)



Figure 1: An example of a layout returned by our method. Figure 1a shows a metric map of a simulated large-scale building, where some rooms have been only partially observed by the robot. Figure 1b displays the retrieved layout we obtained from that map, where shapes of fully-observed rooms are shown with solid colors and predicted shapes of partially-observed rooms are shown with dashed-color patterns (bottom-left and top-right).

1. Introduction

Understanding the environments in which they operate is an important capability for autonomous mobile robots. Buildings are strongly structured environments that are often organized in regular patterns. The identification of the building structure is useful for several tasks, such as semantic mapping, place categorization, path planning, human-robot communication, and task allocation [1]. For instance, knowledge about the building structure can be useful to efficiently spread the robots to incrementally build a map of an initially unknown environment, in the context of coordinated multirobot exploration [2].

Metric maps, like grid maps [3], which are the usual environment representation employed in autonomous mobile robotics, do not explicitly contain any knowledge about the building structure, but only represent the space occupation for navigation and localization purposes. In addition, metric maps may contain several inaccuracies and missing data, since portions of the environment may not be observed by the robot, as in case of occlusions or during exploration.

The *layout* of a building is a geometrical representation of its walls and

rooms that provides a “clean” and stable knowledge about the structure of the building, disregarding information about furniture and regularizing noisy and missing data, which often affect metric maps. Each room is represented either
20 by a polygon (in 2D) or by a box model or a set of planes (in 3D). Walls are accordingly represented as line segments or planes. *Layout reconstruction* is the task of retrieving the layout from a metric representation of a building and can be performed at room or at floor level, starting from 2D grid maps [4] or from 3D point clouds [5, 6]. 2D metric maps are easily available in most robotic
25 applications, thanks to cheap and reliable 2D sensors, like laser range scanners, and to widely used and efficient 2D mapping algorithms [3]. Moreover, indoor robots usually rely on 2D metric maps as they are more robust than 3D point clouds for several tasks [7] and can deal better both with large-scale buildings and with some types of maps, like maps of partially observed environments incrementally built during exploration. Such partial maps are particularly critical,
30 as they are often the only type of knowledge available online to a robot during exploration. However, the knowledge about the structure of buildings that could be extracted from 2D metric maps is more limited than that obtainable from 3D point clouds, where walls, ceilings, and doors are usually easier to identify. Therefore layout reconstruction is usually performed starting from 3D point
35 clouds and this reduces the applicability of layout reconstruction methods. The use of 2D metric maps could potentially smooth such limitation, but it requires to adapt layout reconstruction techniques to the more challenging input of 2D metric maps.

40 In this paper, we address the above issue by presenting a method that retrieves the layout of indoor environments starting from their 2D metric maps, in order to extend the applicability of layout reconstruction methods. Specifically, our method reconstructs the layout of the parts of the environment that have been fully observed by the robot and, most significantly, *predicts* a possible
45 layout for the parts of the environments that have been only partially observed by the robot. An example of a layout returned by our method starting from a partially observed metric map is shown in Figure 1.

Our method does not need some of the requirements needed by most methods that deal with 3D point clouds for layout reconstruction, like perfect alignment
 50 between scans and precise knowledge of their poses [6]. These requirements are hard to satisfy for metric maps built by mobile robots using 2D laser range scanners and SLAM algorithms [3]. These maps usually present several inaccuracies that our method can address, such as partial, missing, unaligned, and noisy data.

55 In a nutshell, our proposed method identifies the *representative lines* along which the walls of a building are aligned and uses these lines to segment the area in smaller parts, called *faces*, which are finally clustered in rooms. The representative lines allow to find regularities between different parts of the same building; for instance, two rooms placed at the opposite sides of the building
 60 can have aligned walls or a long corridor can connect rooms with the same shape and sharing the same wall. These regularities, extracted from the known parts of the environments, are exploited to predict a possible layout for partially observed rooms. Our approach is experimentally validated in an extensive range of settings, showing that it performs well with different metric maps provided
 65 as input and with metric maps representing environments at different degrees of completeness.

The availability of a layout, like that returned by our method, is useful in several robot applications. For instance, a vacuum-cleaner or a service robot can benefit from planning efficient paths in an abstract representation of the
 70 environment and adjusting them during actual motion. Moreover, the computing time of our method (few seconds) allows its use in the online process of exploration of unknown indoor environments [8, 9], in which the robot decision-making about where to move next is performed at a low frequency, as the robot has to physically move around the environment. We explicitly remark that
 75 applying the proposed method to specific settings is beyond the scope of this paper.

This paper is organized as follows. The next section surveys the related work. Section 3 presents the proposed method, which operates in two sequential steps:

reconstruction of the layout of the fully observed parts of the environment (Section 3.1) and prediction of a possible layout of the partially observed parts of the environment (Section 3.2). Section 4 discusses the results of the experimental activities performed to evaluate the proposed method. Finally, Section 5 concludes the paper.

Preliminary versions of the methods of Sections 3.1 and 3.2 appear in [10] and [11], respectively. This paper originally presents a more coherent view of the two methods and introduces some enhancements in the algorithms (for clustering line segments and for determining the candidate faces) that slightly improve the identification of walls and rooms in partially observed environments. Moreover, we re-implemented the methods and reduced computing times by performing code-level optimizations and by parallelizing and embedding our system into a ROS-based framework. This allowed the online use of our approach in an exploring mobile robot (Section 4.5). Finally, this paper presents several further experimental results, including those on maps built by real robots, that show the ability of the proposed method to reconstruct and predict the layout of buildings.

2. Related Work

Automatic analysis of representations of floors of buildings in order to extract structural information is an interdisciplinary topic addressed in different research fields, such as robotics, architecture, computer vision, and image analysis. While an exhaustive survey is out of the scope of this paper, here we discuss a significant sample of methods, focusing on those developed for mobile robots.

Room segmentation methods typically start from 2D metric maps obtained from laser range scanners and separate them into parts, each one corresponding to a different room. Authors of [1] present a survey of room segmentation techniques for mobile robots. They identify four main families of approaches, namely Voronoi-based partitioning [12], graph partitioning [13], feature-based segmentation [14, 15, 16], and morphological segmentation [17]. Representative

methods of the four families are compared in [1] with no method clearly outperforming the others, although Voronoi-based segmentation techniques appear to
110 be the most accurate.

A system for room segmentation that shares some similarities with our layout reconstruction approach can be found in [18]. Similarly to our method, Canny edge transform and Hough line transform are used to extract lines from a grid map. The main difference wrt our method is that, while we extract a relatively
115 small set of representative lines that are used for identifying walls, in [18] a larger set of lines are used for obtaining a scalable grid representation of the environment, similar to an octomap and called *A-grid*, which is eventually used to perform segmentation. Moreover, differently from our method, that of [18] does not perform layout reconstruction but room segmentation.

120 In general, our approach differs from room segmentation methods, which typically partition the metric maps in rooms, because it extracts from the metric map a more abstract representation in which rooms are modeled using geometrical primitives. In our layout representation, a room is a polygon, while, in room segmentation, a room is a set of cells of the grid map. One could
125 say that layout reconstruction provides a more abstract representation of the structure of a building that does not have to be metric exact. Although it is possible to recover the polygon representing a room from the set of cells associated with that room [6] (i.e., as a post-processing of room segmentation), this is largely based on information local to the rooms, hardly capturing global regular
130 features, like alignment of rooms along a corridor, which our approach is able to consider. Most of the methods that perform layout reconstruction require aligned 3D point clouds and usually exploit the knowledge of the poses from where the scans are taken. Our method is inspired by such approaches, but it is adapted to be used on 2D maps obtained by a mobile robot. More precisely,
135 we extend and adapt the method presented in [6], as explained extensively in Section 3.

Authors of [5] segment and reconstruct rooms of an indoor environment from a 3D point cloud perfectly aligned to a coordinate system. Points are projected

on the 2D plan, in order to find walls as sets of points whose projections are close
140 to each other. The entire floor is segmented in areas using these projections of
walls. Areas that are adjacent but not separated by a “peak-gap-peak” pattern
in the projected points distribution are merged in the final segmentation.

In [19], wall lines are retrieved from an analysis of the distribution of points
of a 3D point cloud projected on a 2D plane along the z axis. The structure
145 is then decomposed into its horizontal and vertical parts, which are used to
extract a volumetric model through an energy minimization problem solved by
a Graph-Cut method.

Two methods that are based on aligned 3D point clouds are presented in [20]
and [21]. In [20], 3D point cloud scans (and their poses) are used to recover the
150 3D model of a building. Walls are detected by projecting on a plane the wall
surfaces perceived in different rooms. Similarly to our approach, walls are used
for dividing the space. In particular, wall lines are used to reconstruct a planar
graph that is later segmented into different rooms by an energy minimization
approach. A similar, but improved, method is presented in [21]. It projects
155 a 3D point cloud on a 2D plane to retrieve walls. The 2D map is segmented
into rooms by identifying doors/openings along walls and by selecting a set of
viewpoints from the Voronoi graph built on the 2D projection. Finally, rooms
are found by solving an energy minimization problem on a cell planar graph
obtained from the lines of the wall segments. Viewpoints are used to initialize
160 the graph potentials following the intuition that points that can be seen from
the same viewpoint are more likely to be part of the same room.

Similarly to our work, [4] proposes a method that segments a 2D metric
map of an indoor environment built by a robot while reconstructing its layout.
It uses a framework based on Markov Logic Networks and data-driven Markov
165 Chain Monte Carlo (MCMC) sampling. Using MCMC, the system samples
many possible semantic worlds (layouts of the environment) and selects the one
that best fits the sensor data. Differently from our work, which identifies the
building structure by analyzing the entire metric map, each transition from a
state to another state in the MCMC is based on local edit operations on the

170 layout of a single room, ignoring other parts of the building.

All the above methods are able to reconstruct the layout of observed parts of the environments. The goal of obtaining knowledge on unobserved parts of environments has recently been addressed using heterogeneous approaches. In [22], the metric map perceived by the robot is augmented with knowledge of previously observed environments. The parts of the metric map that correspond to portions of the environment that have not yet been observed are completed by superimposing matching maps from a database of previously observed environments. The resulting map is used for predicting loop closures. In a similar way, the method of [23] first predicts the perimeter of the unobserved space and then uses a library of map structures to predict the inner unknown parts of a map while it is built by a team of robots. While the above methods rely on the presence of libraries of environments observed in the past, our approach can be applied also when such data are not available.

A method that shares some similarities with our approach is proposed in [24], which predicts the structure of an unexplored region of an environment to improve SLAM performance. The method starts from a frontier between known and unknown regions and tries to reconstruct its neighbourhood by identifying similar structures in the known map. If a match is found, the matching portion of the map is superimposed to the frontier, thus providing an estimate of the structure of its neighbourhood. There are also mechanisms that avoid the use of inaccurate predicted knowledge that could compromise the robot behaviour. The prediction of [24] considers the similarity between different parts of the same environment (a prediction is made if a part of the map which can constitute a good match to complete a frontier is found), while in our approach we consider more abstract features like the fact that rooms aligned along the same corridor share the same wall.

Some methods have been recently proposed to predict the presence of specific elements in the unknown parts of environments. For instance, the method of [25] trains, from a set of images representing floor plans of buildings, a convolutional neural network that is able to predict, given the partial map of an environment,

the location of an emergency exit. In this context, only one structural feature of the environment is extracted explicitly (the position of the emergency exit). The system proposed in [26] predicts the existence of a room (and its label) in the unexplored space, but it does not provide any information on the geometry of the room. Similarly, the approach of [27] predicts the topology and the labels of unvisited rooms by matching the known part of the environment (represented as a labeled graph) to a database of environments. Also [28] predicts portions of environments represented as graphs using a method based on graph kernels. Finally, an interesting integration between metric maps and natural language is proposed in [29] in order to infer the existence (in the maps) of spatial elements that are mentioned in some sentences.

3. Our Method for Retrieving the Layout from Metric Maps

The method we propose in this paper starts from a 2D metric map, like a grid map, identifies the walls in it, and uses them for finding rooms and for reconstructing and predicting the 2D layout of the environment. More precisely, given a metric map M of an indoor environment, our goal is to retrieve its layout \mathcal{L} :

$$\mathcal{L} = \{r_1, \dots, r_n\},$$

where r_i is the layout of a room, namely the polygon representing the room. Our method is designed to work both with fully observed rooms, whose layout is reconstructed, and with partially observed rooms, for which our method is able to predict a possible layout for the entire rooms. Hence, the layout \mathcal{L} could be partitioned in the layout of the set of fully observed rooms \mathcal{L}_C and in the layout of the set of partially observed rooms \mathcal{L}_P . In case of a complete metric map M , we have that $\mathcal{L}_P = \{\}$.

The method is composed of a number of steps executed sequentially. At first, the method reconstructs the layout \mathcal{L}_C of fully observed rooms. This part is described in Section 3.1. After that, our method predicts a possible layout \mathcal{L}_P for the partially observed rooms, leveraging the information from \mathcal{L}_C and

following the intuition that the structural features identified in \mathcal{L}_C are also
 225 common to partially observed rooms, as the fact that a wall could be shared
 by multiple rooms. This second step is described in Section 3.2. As discussed
 in the introduction, preliminary versions of the methods of Sections 3.1 and 3.2
 appear in [10] and [11], respectively.

3.1. Reconstructing the layout of fully observed rooms

230 In this section we describe our approach for reconstructing the layout \mathcal{L}_C of
 fully observed rooms with the help of a running example (Figure 2). We thus
 assume to start from a fully known metric map M . Some of the steps of our
 method (Algorithm 1) are inspired by the approach of [6]. More precisely, we
 use the method developed in [6] for dividing the metric map into smaller parts
 235 (which are used to identify rooms) using a set of lines. However, as already
 discussed, [6] reconstructs the 3D layout of a building from point clouds precisely
 aligned and registered in the same coordinate system, knowing the number and
 the poses of scans. Differently from [6], our method is intended to be used
 on 2D metric maps obtained from data acquired by laser range scanners and
 240 processed by SLAM algorithms, which can present misalignments and artefacts.
 Moreover, our method can be used effectively on different types of metric maps
 obtained from different sources, such as incomplete metric maps, blueprints,
 and evacuation maps, as described in Section 4.

The starting point is a metric map M representing an indoor environment,
 245 typically a floor of a building. Without loss of generality, we assume that M is
 a 2D grid map, like the one in Figure 2a, namely a two-dimensional matrix of
 cells (pixels), each one representing the probability that the corresponding area
 is occupied by an obstacle.

The first operation on M is the detection of significative edges using the
 250 Canny edge detection algorithm [30], which partitions the cells of M into free
 cells and obstacle cells. The binary metric map resulting from the application
 of the Canny edge detection algorithm is called M' and an example is shown in
 Figure 2b.

```

Input: a grid map  $M$ 
Output: the reconstructed layout  $\mathcal{L}_C$ 

/* Compute features from the map */
 $M' \leftarrow \text{CannyEdgeDetection}(M)$ 
 $S \leftarrow \text{pHoughLineTransform}(M')$ 

/* Obtain contour of the map */
 $M'' \leftarrow \text{thresholdMap}(M, q)$ 
 $\text{Inner} \leftarrow \text{computeMapContour}(M'')$ 

/* Create clusters of collinear segments */
 $\mathcal{C} \leftarrow \text{meanShiftClustering}(S)$ 
 $\mathcal{W} \leftarrow \text{spatialClustering}(\mathcal{C}, \text{wall})$ 

/* Find faces from representative lines */
 $\text{lines} \leftarrow \text{getsRepresentativeLines}(\mathcal{W})$ 
 $F \leftarrow \text{findFaces}(\text{lines})$ 

/* Compute spatial affinity between faces */
 $L \leftarrow \text{computeAffinityMatrix}(F)$ 
 $D \leftarrow \text{diag}(\sum_{j=1}^n L_{i,j})$ 
 $A \leftarrow D^{-1}L$ 

/* Remove external faces outside border */
for  $f \in F$  do
    if  $\text{area}(f \cap \text{Inner}) < \delta$  then
         $F \leftarrow F \setminus f$ 
    end
end

/* Remove partially observed faces */
for  $f \in F$  do
    if  $\text{containFrontier}(f)$  then
         $F \leftarrow F \setminus f$ 
    end
end

/* Cluster together faces into rooms */
 $\mathcal{L}_C \leftarrow \text{DBSCAN}(F, A, \epsilon, \text{minPoints})$ 
return  $\mathcal{L}_C$ 

```

Algorithm 1: Our method for layout reconstruction of fully observed rooms.

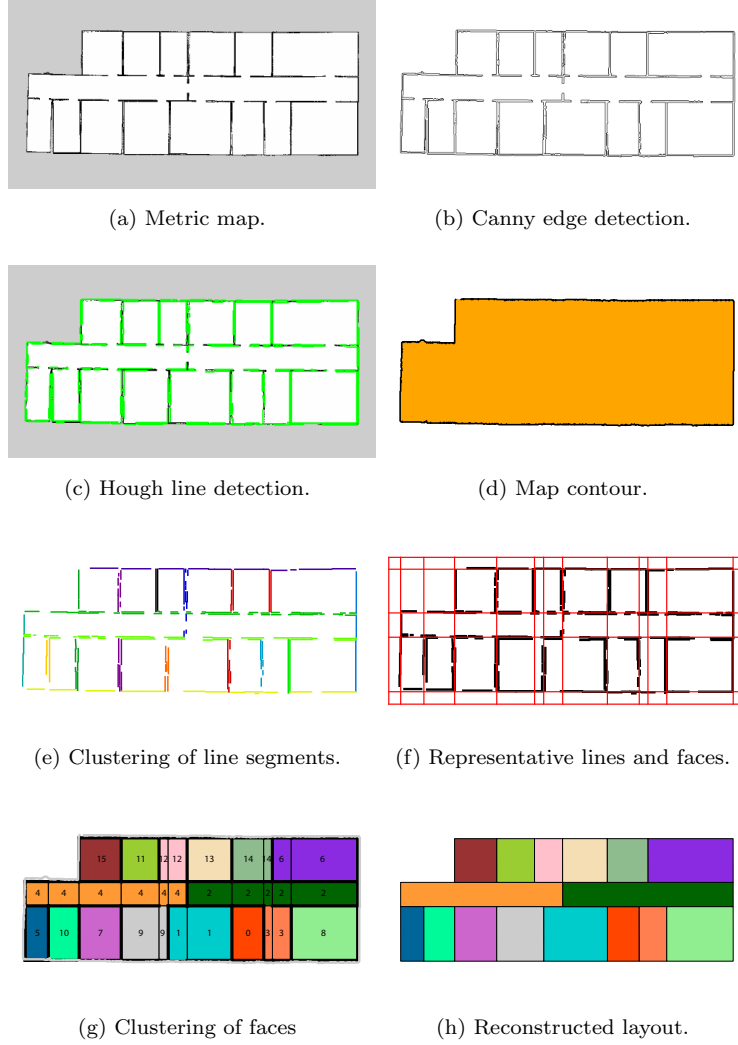


Figure 2: An example run of our method to reconstruct the layout from a complete metric map.

The set of edges (obstacle cells) is then processed by a probabilistic Hough
 255 line transform algorithm [31] to detect the line segments S that approximate
 the edges in M' . Figure 2c shows such line segments in green. Note that one of
 the parameters of the method, $minLinPoints$, represents the minimum number
 of points that are required for having a line. Then, the contour of the map is

obtained, using the contour detection algorithm of [32], after the application
of a threshold q that divides cells of the original map M into free or occupied.
260 Figure 2d shows in yellow the area inside the map border.

The line segments S are then clustered together according to the angular
coefficients of their supporting lines using the mean shift clustering algorithm
[33]. At the end of the angular clustering, we obtain a set of clusters $\mathcal{C} =$
265 $\{C_1, C_2, \dots\}$ such that each $C_j \subseteq S$ and $C_1 \cup C_2 \cup \dots = S$. Each cluster C_j
represents the set of line segments with similar angular coefficient α_j , namely
with similar direction, independent of their spatial proximity.

Next, the line segments belonging to the same angular cluster C_j are further
clustered in sets of collinear walls by grouping those line segments that are close
270 to each other. Specifically, line segments of C_j are partitioned in sets $W_{j,k}$. This
spatial clustering is performed using DBSCAN [34] with the smallest distance
between any two points of two line segments s and s' as metric. (We set the
two DBSCAN parameters to $\epsilon = 0.75$ and $minPoints = 2$.) Each line segment
in C_j is thus assigned to a cluster $W_{j,k}$, which represents a wall.

275 Collinear walls are further joined if the line segments composing them are
close enough. The distance between two walls is computed as follows. Given a
wall $W_{j,k}$, one of its line segments $s_{j,k}$ is selected as that with the median middle
point (along a direction perpendicular to α_j). Now, given two walls $W_{j,k}$ and
 $W_{j,k'}$, call l and l' the parallel lines passing through the middle points of $s_{j,k}$
280 and $s_{j,k'}$, both with angular coefficient α_j . If the distance between l and l' is less
than a threshold *wall* (intuitively, closer than the width of a doorway), then the
two corresponding walls $W_{j,k}$ and $W_{j,k'}$ are merged together. At the end of this
step, the set of clusters $\mathcal{C} = \{C_1, C_2, \dots\}$ is thus further partitioned into a set of
clusters $\mathcal{W} = \{W_{1,1}, W_{1,2}, \dots, W_{2,1}, W_{2,2}, \dots\}$, such that $C_1 = W_{1,1} \cup W_{1,2} \cup \dots$
285 and $C_2 = W_{2,1} \cup W_{2,2} \cup \dots$, and so on. Figure 2e shows the results of the
spatial clustering where the line segments belonging to the same cluster in \mathcal{W}
are displayed with the same color.

At this point, for each cluster $W_{j,k}$, a *representative line* $l_{j,k}$ that represents
all the line segments in $W_{j,k}$ is determined. This representative line is computed

290 as the line with angular coefficient α_j associated to C_j and that passes through
 the median of the set of middle points of the line segments in $W_{j,k}$. Each
 representative line, in red in Figure 2f, indicates the direction of a wall within
 the building. The intersections between all lines induce a tessellation of the
 metric map that divides the area of the map M' into different polygonal areas,
 295 called *faces*, as shown in Figure 2f. We call F the set of faces. Note that, as
 visible in Figure 2f, our method uses a relatively small number of representative
 lines for segmenting the environment, as line segments found in the metric map
 are hierarchically clustered together. This allows us to consider a small number
 of primitives in the environment (line segments and faces) in order to capture
 300 regularities in the metric map. For example, in our method, collinear walls which
 are slightly distorted and tilted due to map inaccuracies are not represented by
 different representative lines close to each other, as in [18, 21], but are considered
 as a single representative line. In this way, our method provides a more abstract
 representation of the environment that is used for layout reconstruction. In
 305 order to handle partially observed rooms (see next section) whose frontiers are
 not bounded by any representative line, we pad the map M with an additional
 set of representative lines that form a bounding box of M slightly larger than
 the actual size of the map contour.

Rooms are determined by grouping faces together. Adjacent faces that are
 separated by an edge corresponding to a wall should belong to different rooms,
 while adjacent faces that are separated by an edge not corresponding to any wall
 should be grouped together in the same room. More precisely, for each pair of
 faces f and f' that share a common edge $e_{f,f'}$ (belonging to the representative
 line $l_{j,k}$ of a spatial cluster $W_{j,k}$), we compute a weight $w(e_{f,f'})$ as follows (as
 in [6]):

$$w(e_{f,f'}) = \frac{cov(e_{f,f'})}{len(e_{f,f'})},$$

where $len(e_{f,f'})$ is the length of $e_{f,f'}$ and $cov(e_{f,f'})$ is the length of the projec-
 tions, on $e_{f,f'}$, of the line segments in $W_{j,k}$. The larger the weight $w(e_{f,f'})$,
 the stronger the hypothesis that there is a wall (obstacle) along $e_{f,f'}$ (namely,

between faces f and f'). If an edge is completely covered by projections of line segments in $W_{j,k}$, then its weight is 1. Following the definition of [6], weighted edges are used to compute an affinity measure L between all pairs of faces. L is similar to a Laplacian, and its entries $L_{f,f'}$ are defined as:

$$L_{f,f'} = \begin{cases} e^{-w(e_{f,f'})/\sigma} & \text{if } f \neq f' \text{ and } f \text{ and } f' \text{ are adjacent} \\ 1 & \text{if } f = f' \\ 0 & \text{otherwise} \end{cases},$$

where σ is a regularization factor. From the matrix L , a local affinity matrix A is defined as $A = D^{-1}L$, with $D = \text{diag}(\sum_{j=1}^n L_{i,j})$, where n is the number of faces in F and i is the row of the matrix. Each element $A_{f,f'}$ indicates an affinity value considering the local connectivity between faces f and f' . The matrix A is used as input for DBSCAN, which clusters faces. DBSCAN groups together those faces that are close to each other in a dense portion of the feature space represented by matrix A . (In our experiments, we set the two DBSCAN parameters to $\epsilon = 0.85$ and $\text{minPoints} = 1$.) Note that DBSCAN detects rooms without any additional information, while the method of [6] relies on the availability of scan poses. In this sense, our approach can flexibly adapt to different input sources for metric maps, as shown in the next section.

Before applying DBSCAN, some faces are discarded. Specifically, discarded faces are those called *external* and such that the area of their intersection with the inner area of M (obtained from the contour) is smaller than a threshold δ .

Then, DBSCAN is applied to internal faces to obtain a set of clusters $R = \{F_1, F_2, \dots\}$ of F . Each cluster F_i corresponds to a room r_i , which is represented as a polygon obtained by merging together all the faces belonging to F_i . Figure 2g shows the results of DBSCAN for our example. The number i in each face indicates its cluster F_i . Different clusters have also different colors.

The final reconstructed layout $\mathcal{L}_C = \{r_1, r_2, \dots\}$ is finally displayed in Figure 2h. Note that it is a “clean” and abstract representation of the original grid map of Figure 2a, which nevertheless retains the main structural features.

A side effect of our proposed method is that it could happen that gaps

```

Input:  $M, \mathcal{L}_C, F, lines$ 
Output: the predicted layout  $\mathcal{L}_P$ 

/* Get partial rooms */
 $P \leftarrow FindPartialRoomsFromFrontiers(M, F, \mathcal{L}_C)$ 
/* Get candidate faces */
for  $r_i \in P$  do
    /* Find initial layout composed of already observed faces */
     $\hat{F}'_i \leftarrow getObservedFacesInRoom(r_i, M, P, F)$ 
     $r'_i \leftarrow \hat{F}'_i$ 
    /* Evaluate best layout among all possible sets of faces */
    for  $\hat{F} \in \mathcal{F}$  do
        if  $\Phi(\hat{F} \cup F'_i) \geq \Phi(r'_i)$  then
             $r'_i \leftarrow \hat{F} \cup F'_i$ 
        end
    end
    /* Update map and set of faces after new predicted room */
     $\hat{r}_i^* \leftarrow r'_i$ 
     $\mathcal{L}_P \leftarrow \mathcal{L}_P \cup \hat{r}_i^*$ 
     $\mathcal{F} \leftarrow \mathcal{F} \setminus (\hat{F} \cup F'_i)$ 
end
return  $\mathcal{L}_P$ 

```

Algorithm 2: Our method for predicting the possible layout of partially observed rooms.

inside buildings are wrongly identified as rooms. However, such rooms are disconnected from the rest of the environment and can be easily filtered out. A post-processing refinement could be used to remove unconnected rooms when connectivity between rooms (e.g., doorways) is known, like when the input metric map comes from a blueprint.

3.2. Predicting the layout of partially observed rooms

In this section, we illustrate our method that predicts a possible layout of partial rooms in \mathcal{L}_P , namely of those rooms that have not been fully observed by the robot sensors (Algorithm 2).

The input metric map M , in this case, is partial, and does not represent the entire environment. In a partial metric map, *frontiers* can be identified as the boundaries between known and unknown space. For example, in a partial

grid map in which cells can be either known or unknown and known cells can
 345 be either free or obstacles, a frontier cell is a free cell that is adjacent to an
 unknown cell, while a frontier is a chain of adjacent frontier cells.

To account for the fact that the initial grid map M is partial, before faces
 are clustered in rooms (last step of Algorithm 1), *partial faces* are recognized
 as faces that contain a frontier and are excluded from the process that brings
 350 to identify $\{F_1, F_2, \dots\}$. After clustering, if a room r_i has faces in F_i that
 are adjacent to partial faces (with the common edge not being a wall), it is
 considered a *partial room*. Also partial faces that are not adjacent to any F_i are
 clustered to form additional partial rooms. The set of partial rooms is called P .

The main idea of our method is that the structure of a partial room shares
 355 some features with the structure of the part of the environment that is known.
 For example, the external walls of a building are shared by several rooms and
 the rooms along a corridor have a similar shape. Our method considers the
 partial rooms of P in sequence (a random sequence is fine since each partial
 room is processed independently) and, for each partial room, it completes its
 360 layout by predicting the unknown part of the room still to be observed by the
 robot. In practice, given a partial room $r_i \in P$, some (unobserved) faces are
 added to the (observed) faces in F_i in order to maximize an objective function
 that represents how well the new layout \hat{r}_i consistently matches with that of the
 known part of the environment.

365 In detail, given a partial room $r_i \in P$ with fully observed faces F_i (obtained
 as described in the previous section), our method first adds to F_i the partial
 faces that contain a frontier and that are adjacent to at least one face in F_i
 (and with the common edge not corresponding to a wall), obtaining a set of
 faces $F'_i \supseteq F_i$.

370 Then, F'_i is further enriched with a set of faces \hat{F}_i^* , the predicted layout \hat{r}_i^*
 of partial room r_i is obtained from merging faces in $F'_i \cup \hat{F}_i^*$, and \hat{r}_i^* is added to
 \mathcal{L}_P . \hat{F}_i^* is selected from a family \mathcal{F} of candidate sets of faces (also including the
 empty set). A candidate set of faces $\hat{F} \in \mathcal{F}$ satisfies the following conditions:

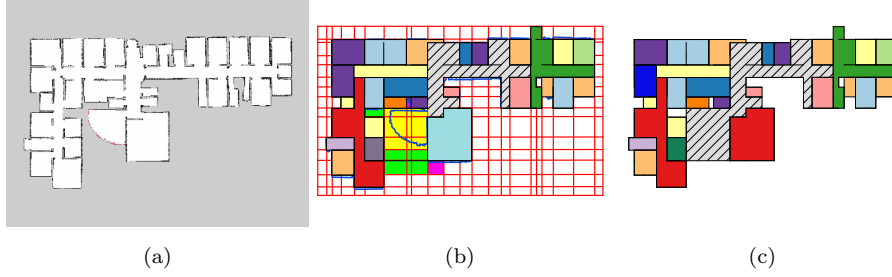


Figure 3: Figure 3b shows an example of prediction of the layout of partial rooms: faces in the two selected $F'_i \cup \hat{F}_i^*$ are in yellow, faces belonging to other candidate sets \hat{F} are in green, faces excluded from candidate sets \hat{F} due to conditions (1)-(3) are in fuchsia. Figure 3a shows the map M and Figure 3c the reconstructed layout.

- (1) \hat{F} contains faces adjacent to faces of F'_i or adjacent to faces which are in turn adjacent to faces in F'_i (namely, at most “two hops” away from faces in F'_i);
- (2) \hat{F} contains faces not adjacent to the external contour of the map;
- (3) \hat{F} contains faces not “behind” any of the walls that have already been observed for the room r_i .

Condition (1) limits the number of sets in \mathcal{F} ; conditions (2) and (3) keep the prediction of the layout of a room consistent with what has been observed. \hat{F}_i^* is chosen from \mathcal{F} to maximize an objective function:

$$\hat{F}_i^* = \arg \max_{\hat{F} \in \mathcal{F}} \Phi(F'_i \cup \hat{F}),$$

where $\Phi()$ is a weighted sum of three components:

$$\Phi(F'_i \cup \hat{F}) = k_{\Upsilon} \cdot \Upsilon(F'_i \cup \hat{F}) - k_{\Psi} \cdot \Psi(F'_i \cup \hat{F}) - k_{\Omega} \cdot \Omega(F'_i \cup \hat{F}).$$

Note that we evaluate a candidate \hat{F} by considering the set of faces $F'_i \cup \hat{F}$ and the corresponding layout \hat{r}_i . See Figure 3 for an example.

The first term of the objective function, $\Upsilon()$, embeds the intuition that the predicted layout \hat{r}_i should be consistent with that of known rooms. The

quantitative evaluation of this consistency is done by resorting to the representative lines. For example, if $l_{j,k}$ represents a wall along a long corridor, it will be common to all the rooms that are connected to that corridor. Thus, the same wall will delimit the partial rooms connected to the same corridor. Given a representative line $l_{j,k}$, we consider in turn all the line segments s of the associated cluster $W_{j,k}$, we project them on $l_{j,k}$, and we compute a weight $\bar{w}(l_{j,k}, s) = cov(s, l_{j,k}) / len(l_{j,k})$, where $cov(s, l_{j,k})$ is the length of the projection of s on $l_{j,k}$, and $len(l_{j,k})$ is the length of the segment of $l_{j,k}$ that is inside the contour of the map M . We then call \hat{L} the set of representative lines that would be the boundaries of room \hat{r}_i if \hat{F} is selected to complete the room, finally computing:

$$\Upsilon(F'_i \cup \hat{F}) = \sum_{l_{j,k} \in \hat{L}} \sum_{s \in W_{j,k}} \bar{w}(l_{j,k}, s).$$

The second term of the objective function, $\Psi()$, is designed in order to prefer simple (predicted) layouts over complex ones. This is done by computing the convex hull of the layout \hat{r}_i in case \hat{F} is added to F'_i . Calling $area()$ the operator that computes the area of a polygon and $hull()$ the operator that computes the convex hull of a polygon:

$$\Psi(F'_i \cup \hat{F}) = \frac{area(hull(\hat{r}_i)) - area(\hat{r}_i)}{area(\hat{r}_i)}.$$

The third term of the objective function, $\Omega()$, prefers the layouts that are delimited by a small number of walls. This is done by counting the number $nw_{\hat{r}_i}$ of walls of \hat{r}_i , namely the number of external edges of the faces in $F'_i \cup \hat{F}$ that lie on different representative lines, thus having $\Omega(F'_i \cup \hat{F}) = nw_{\hat{r}_i}$.

Figure 4 shows an example that illustrates the proposed method for the prediction of the layout of three partial rooms. The predicted layout of the partial rooms is dashed in Figure 4c. In our experiments, we set $k_\Upsilon = 1$, $k_\Psi = 5$, and $k_\Omega = 0.1$ (these values have been selected after several preliminary trials).

A particular situation is encountered when a partial room is at the border

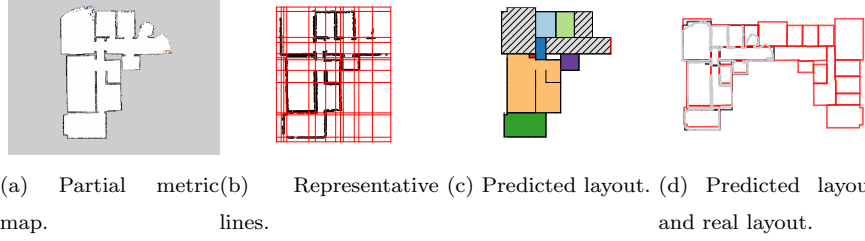


Figure 4: An example run of our approach starting from a partial grid map with predicted layout of partially observed rooms (dashed) and an open frontiers (red). Figure 4d shows the map perceived by the robot (gray), the layout predicted by our method (black), and the actual layout of the building (red).

of the map M . In this case, one or more sides of the room are not bounded by
 410 any representative line derived from the known map and the layout of the room
 cannot be predicted. When this happens, we label the room as containing an
open frontier and we highlight the corresponding edges in red, as in Figure 4.

Before starting the prediction of the layout of partially observed rooms,
 a pre-processing step is applied to \mathcal{L}_C in order to reduce under- and over-
 415 segmentation of rooms. These two issues are caused when two (or more) rooms
 are wrongly considered as a single entity (under-segmentation) or when a large
 room is incorrectly split in several smaller rooms (over-segmentation). (See the
 next section for details.) If the portion of a room observed by the robot is very
 small, as when the robot passes in front of a doorway without going inside the
 420 room, under-segmentation is very likely. A Voronoi graph obtained from M is
 used to spit (likely under-segmented) rooms divided by a doorway, following the
 approach presented in [1]. Over-segmentation is instead addressed by using the
 approach proposed as post-processing in [6] that merges two adjacent rooms
 for which the following function Q_{split} returns a value larger than a threshold
 425 $\tau = 0.2$:

$$Q_{split}(\mathcal{B}) = \frac{\sum_{i=1}^n w(e_i) * len(e_i)}{\sum_{i=1}^n len(e_i)},$$

where $\mathcal{B} = \{e_1, \dots, e_n\}$ is the set of edges common to those two rooms and
 $w(e_i) = cov(e_i)/len(e_i)$ as defined in Section 3.1.

4. Experimental Results

In this section, we evaluate our method for reconstructing and predicting
 430 the layout of buildings from 2D metric maps.

A reconstructed (or predicted) layout is expected to present two main characteristics: (1) all the rooms of the real building (and only them) should be in the layout; (2) the shape of each room in the layout should match that of its real counterpart. Evaluation is performed both visually and quantitatively,
 435 comparing the reconstructed or predicted layout \mathcal{L} and the ground truth layout Gt . Following the approach of [1], we introduce two matching functions between rooms in \mathcal{L} and in Gt (after \mathcal{L} and Gt have been aligned), namely *forward coverage* FC and *backward coverage* BC :

$$FC : r \in \mathcal{L} \mapsto r' \in Gt \quad BC : r' \in Gt \mapsto r \in \mathcal{L}.$$

For each room $r \in \mathcal{L}$, FC finds the room $r' \in Gt$ that maximally overlaps r ;
 440 conversely, for each room $r' \in Gt$, BC finds the room $r \in \mathcal{L}$ with the maximum overlap with r' . Recalling that $area()$ is a function that computes a polygon area, the amount of overlap between a room $r \in \mathcal{L}$ and a room $r' \in Gt$ is defined as $area(r \cap r')$.

Matching functions FC and BC are used for computing two measures of
 445 accuracy called *forward accuracy* A_{FC} and *backward accuracy* A_{BC} :

$$A_{FC} = \frac{\sum_{r \in \mathcal{L}} area(r \cap FC(r))}{\sum_{r \in \mathcal{L}} area(r)} \quad A_{BC} = \frac{\sum_{r' \in Gt} area(BC(r') \cap r')}{\sum_{r' \in Gt} area(r')}.$$

A_{FC} represents how well the reconstructed layout \mathcal{L} is described by the ground truth layout Gt , while A_{BC} represents how well the ground truth layout Gt is described by the reconstructed layout \mathcal{L} . The numbers of rooms in \mathcal{L} and Gt can be different, due to over- or under-segmentation. Over-segmentation results
 450 in high A_{FC} and low A_{BC} , while under-segmentation results in high A_{BC} and low A_{FC} .

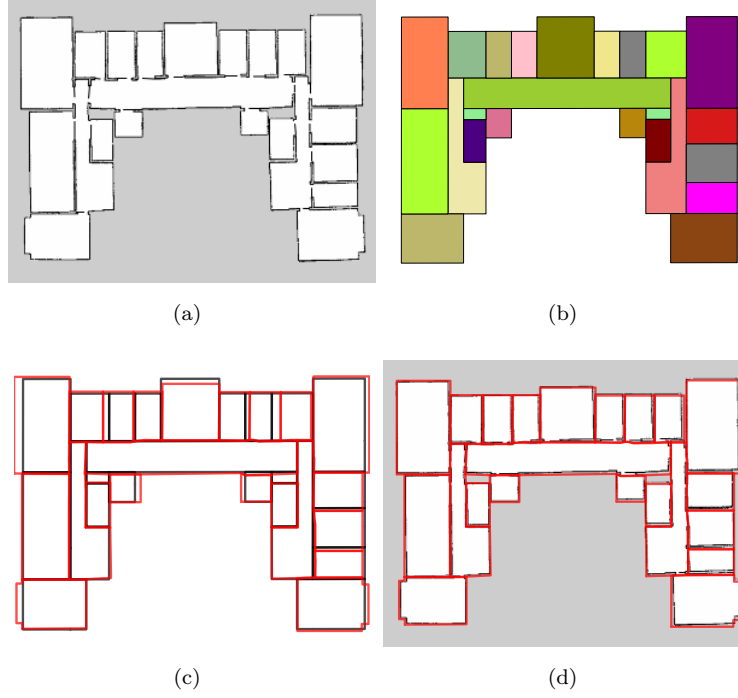


Figure 5: An example of a layout reconstruction. (a) Grid map. (b) Reconstructed layout. (c) Reconstructed layout (black) and ground truth (red). (d) Grid map and ground truth (red).

We evaluate separately the performance of reconstructing the layout \mathcal{L}_C of fully observed rooms and of predicting the layout \mathcal{L}_P of partially observed rooms. Recall that our method reconstructs \mathcal{L}_C and uses it to predict the layout
455 \mathcal{L}_P ; hence, the quality of \mathcal{L}_C affects that of \mathcal{L}_P .

Our method could be used online for supporting the execution of other tasks of autonomous mobile robots (Section 4.5). The computation of \mathcal{L} takes few seconds for small metric maps and less than 10s even for large metric maps (to compute both \mathcal{L}_C and \mathcal{L}_P). The method is run in a dedicated ROS node
460 so that an updated version of \mathcal{L} is always available for potential use without introducing delays.

4.1. Evaluation of layout reconstruction of fully observed environments

We consider 20 2D grid maps obtained running the ROS implementation¹ of the GMapping algorithm [35] on data collected by a robot equipped with a laser range scanner and moving autonomously in 20 large-scale buildings simulated in Stage². Buildings range from 1000 m² to 3500 m² in size and represent real-world school buildings. The same 20 input maps are considered for evaluating our method both for reconstructing the layout of fully observed rooms (in this section) and for predicting the layout of partially observed environments (Section 4.4). In the former case, we consider complete maps at the end of data collection runs, while, in the latter case, we evaluate our method using partial maps at different stages of data collection runs. More details are provided in Section 4.4.

It is important to point out that the evaluation is performed by comparing the layouts reconstructed from the grid maps built by the robot against the *actual* layouts of the simulated buildings fed to Stage. This allows us to evaluate if our layout reconstruction approach is able to cope with noise and errors introduced in the mapping process as a result of noisy readings from the sensor, localization errors, or errors due to inaccurate odometry readings. In our simulations, we assume that the robot has a translational odometry error of up to 0.05 m/m and a rotational odometry error of up to 2°/rad, which provide a reasonable approximation of the odometry accuracy of real wheeled robots. In this sense, we consider simulated but realistic metric maps obtained by robots.

We note that a comparison of our approach against similar methods such as [6, 20, 21] requires radically different input data (e.g., aligned 3D point scans vs. 2D grid maps) and is not reported here.

Average layout reconstruction accuracies over the 20 grid maps are the following: $A_{BC} = 87.8\% \pm 4.5\%$ and $A_{FC} = 87.6\% \pm 5.7\%$. On the set of 20 environments mapped by the simulated robot, our method is able to recon-

¹<http://wiki.ros.org/gmapping>

²<http://wiki.ros.org/stage>

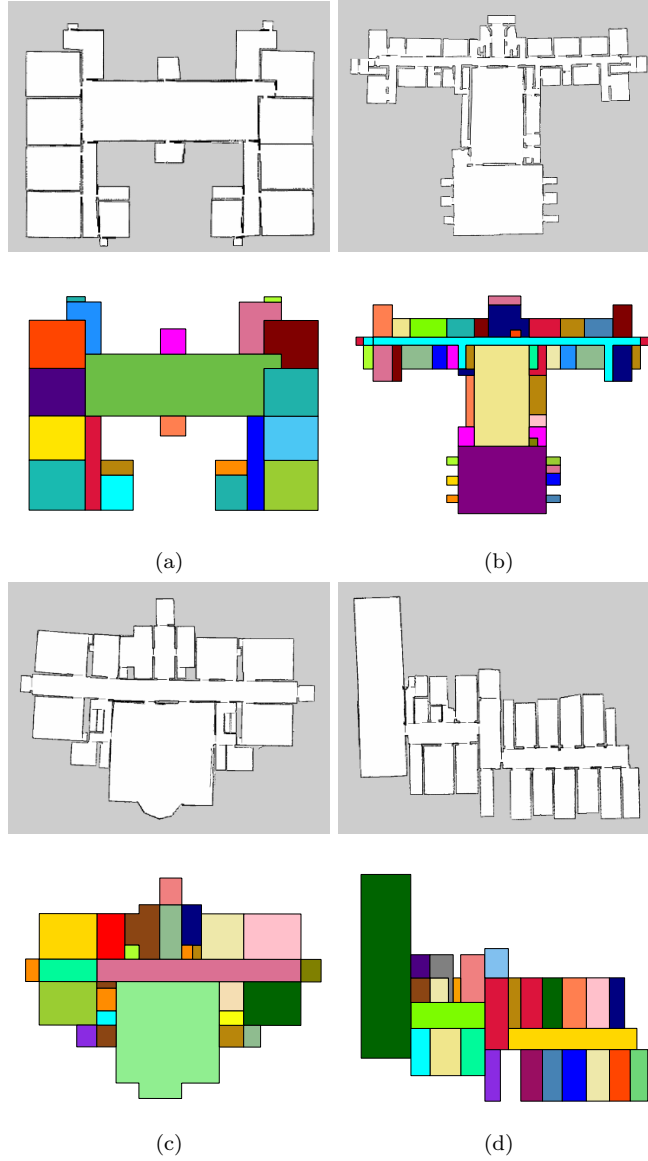


Figure 6: Examples of layout reconstruction.

490 struct successfully and with good accuracy the layout of the original buildings.
 For example, Figure 5 presents a grid map obtained by the robot (Figure 5a)
 and the layout reconstructed using our method (Figure 5b). For this partic-
 ular example, we have $A_{FC} = 90.1\%$ and $A_{BC} = 93.3\%$. In Figure 5c, the

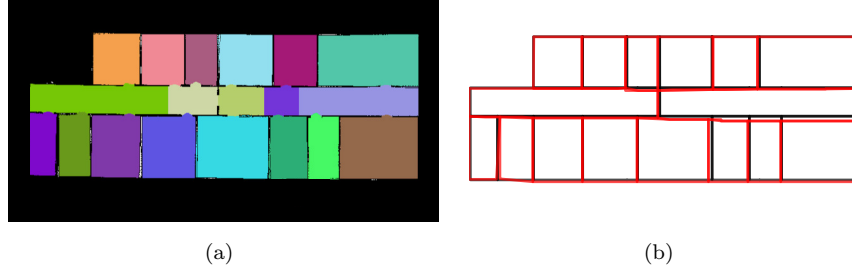


Figure 7: Segmentation by a Voronoi-based approach from [1] (a) and comparison between our reconstructed layout (black) and the ground truth layout (red) (b) for the metric map of Figure 2a.

reconstructed layout (in black) and the ground truth layout (in red) are super-
 495 imposed. Although the layout is generally accurate, there are small differences
 in the geometry of rooms, due to approximations introduced by our method,
 which result in slight performance degradation. In Figure 5d the grid map and
 the ground truth building layout (in red) are superimposed. While the grid map
 provides a good representation of the environment, some inaccuracies, such as
 500 irregular gaps between walls, are present. Our method tries to filter out those
 inaccuracies when reconstructing the layout.

Figure 6 shows four other examples of reconstructed layouts from grid maps.
 In all the four cases, our method is able to correctly reconstruct the layout of
 the environment, coping well with alignment and rotation errors and with gaps
 505 between rooms in the metric maps (e.g., see Figure 6c). Few inaccuracies are
 introduced, like long corridors split into smaller units, thus producing over-
 segmentation. Another error is made sometimes when there is a small gap
 within the building (e.g., due to a large wall or a pillar), which is misclassified
 as an internal face and subsequently added to an adjacent room or considered
 510 as a small independent room. However, in general, our method can successfully
 retrieve the layouts of large-scale buildings, as in Figure 6b, where the rooms
 connected to the top corridor (light blue) are the classrooms of a high school.

Since walls are represented by straight lines, round walls are approximated

no furniture		furniture	
recall	90.0% \pm 4.1%	recall	89.5% \pm 6.2%
precision	90.1% \pm 6.3%	precision	94.0% \pm 2.2%

Table 1: Layout reconstruction results on datasets from [1].

by polylines (see Figure 6c). Maps that present (relatively) small misalignments
515 are adjusted by our method, as shown in Figure 6c and 6d. A trade-off exists
between the accuracy of alignment that can be reached by our method and its
ability to approximate round walls with polylines. This trade-off can be set
by modifying the parameters of the line segment clustering step (Section 3.1),
like the *wall* threshold, currently set to the width of a doorway, during which
520 collinear walls are clustered together and used to identify representative lines.
In all the examples presented in this paper we prefer strong alignment over good
approximation of round walls.

4.2. Evaluation on publicly available datasets

Here, we present the results of the evaluation of our method for reconstruct-
ing the layout of fully observed buildings on the two datasets (each composed
of 20 metric maps) used in [1] for comparing different methods for room seg-
mentation. The datasets contain fully observed metric maps with and without
furniture, respectively, and, in [1], are evaluated employing precision and recall
metrics, which are defined similarly to A_{FC} and A_{BC} :

$$Precision = \frac{\sum_{r \in \mathcal{L}} area(r \cap FC(r)) / area(r)}{|\mathcal{L}|}$$

$$Recall = \frac{\sum_{r' \in Gt} area(BC(r') \cap r') / area(r')}{|Gt|}.$$

When the dataset with furniture is used, our method filters out some of the clut-
525 ter introduced by furniture by automatically clustering and removing isolated
obstacle cells using DBSCAN directly on the original metric map M (this is a
very simple way to handle furniture, which can be definitely improved). Results,
reported in Table 1, are good and confirm those obtained with our dataset.

A comparison with the results obtained by room segmentation methods of [1] is unfair, because those methods directly partition the cells of the metric maps, while our method uses a more abstract representation based on representative lines and faces, thus introducing some approximations. An example of the approximations introduced by our layout reconstruction with respect to room segmentation is displayed in Figure 7. Figure 7a shows the Voronoi-based segmentation of the metric map of Figure 2a, as reported in [1]. It can be observed that Voronoi-based methods tend to produce over-segmentation. In Figure 7b, we compare our reconstructed layout (in black) with the real structure of the building (in red). Despite being able to correctly capture the building layout, for this map we have $A_{FC} = 93.6$ and $A_{BC} = 94.6$, due to some visible approximations. That said, comparing Table 1 with Table II of [1], we can say that our method produces results comparable (within 1 sigma) with those of the methods surveyed in [1], further confirming that the reconstructed layout actually captures the structure of the buildings. Moreover, the availability of the layout enables a number of possible tasks, as suggested in the following sections.

Our method does not assume a Manhattan world and can be used in non-Manhattan environments, such as those with diagonal walls. Datasets from [1] feature some non-Manhattan buildings, as the two reported in Figure 8. In Figure 8a-8b, it can be seen how our method approximates round walls with a set of diagonal line segments, as already pointed out in the previous section. An example of reconstruction of the layout of a non-Manhattan building with several diagonal walls is shown in Figure 8c-8d, where the reconstructed layout of the building, despite some inaccuracies due to under-segmentation of the main central corridor (in blue), is basically correct.

4.3. Evaluation on blueprints and evacuation maps

Our method can be used also for reconstructing the layout of a building from its blueprint or evacuation map. Blueprints and evacuation maps are usually images in which the walls of buildings are represented together with symbols and words that explain the meaning and the locations of some features (e.g., the

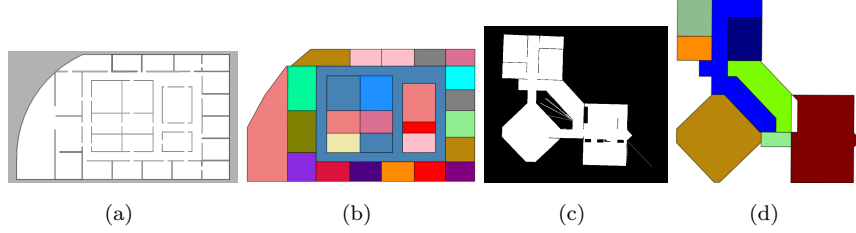


Figure 8: Reconstructed layout from metric maps of non-Manhattan buildings from [1].

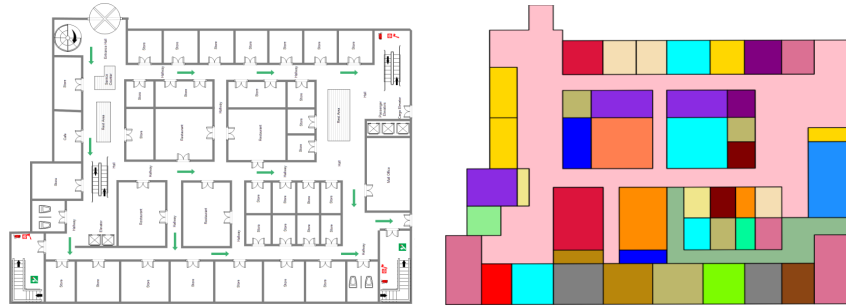


Figure 9: Reconstructed layout from a blueprint of a highly structured building.

functions of the rooms and the presence of fire extinguishers). As blueprints and
 560 evacuation maps represent entire floors of buildings, we apply here our method
 to reconstruct the layout starting from them.

We pre-process these input maps as follows. Words (like the name of the
 building or the functions of rooms) are recognized and filtered out using a stan-
 dard OCR method (Tesseract). Doors and other symbols (like fire extinguish-
 565 ers) are easily retrieved and eliminated, since they are represented with standard
 symbols, by using template-matching algorithms. An example of a layout re-
 constructed from a blueprint is shown in Figure 9. Other examples are reported
 in our repository³. Quantitative results of layout reconstruction starting from
 blueprints (and evacuation maps) are similar to those obtained with datasets
 570 from [1] and are not reported.

³<https://github.com/goldleaf3i/predicted2Dlayout>

The reconstructed layout of a building obtained from a blueprint can be a useful source of knowledge, especially when the environment in which robots operate is initially unknown, as in a situation where a team of robots performs search and rescue operations and an evacuation map of the environment can be
575 seen on a wall. One of the possible uses of such knowledge is localization, as done in [36], where a robot can localize itself using a floor plan instead of a metric map. A reconstructed layout may introduce some approximations compared to the real shape of the environment, as explained previously. However, [37, 38] show how partially inaccurate hand-drawn sketch maps can be effectively used
580 for robot localization. In principle, using similar approaches, a reconstructed layout of a building can be used for robot localization even if it is slightly inaccurate.

4.4. Evaluation of layout prediction of partially observed environments

In this section, we present some of the experimental activities we performed
585 to evaluate our approach for predicting possible layouts of partially observed environments, by trying to complete partial maps acquired during their exploration.

As before, the predicted layout of a partial room is evaluated according to how much it matches the actual layout of the room. Evaluation is performed
590 both visually and quantitatively, comparing the reconstructed layout \mathcal{L} and the ground truth layout Gt . For a quantitative evaluation, we adopt a metric different of forward accuracy and backward accuracy used so far, because predicting the layout of partially observed rooms often results in an increase of under-segmented rooms, since partial rooms where only a small part has been
595 observed are usually under-segmented. Hence, after aligning \mathcal{L} and Gt , given the predicted layout \hat{r}^* of a room in \mathcal{L} , we look for the room r in Gt that maximizes the overlap with \hat{r}^* using the forward coverage function and we compute their *Intersection over Union* (IoU) as:

$$\text{IoU}(\hat{r}^*, r) = \frac{\text{area}(\hat{r}^* \cap r)}{\text{area}(\hat{r}^* \cup r)}.$$

Intuitively, we consider the area $r \setminus \hat{r}^*$ as a false negative, the area $\hat{r}^* \setminus r$ as a
600 false positive, and the area of $\hat{r}^* \cap r$ as a true positive.

Also in the case of partially observed environments, we compare the layouts
reconstructed by our method and the actual layouts of the simulated buildings
fed to Stage simulator. This allows us to evaluate if our layout prediction
approach is able to cope with noise and errors introduced in the mapping process
605 as a result of the imprecise movements of the simulated robot.

We consider the same 20 large-scale school buildings that we used in Sec-
tion 4.1. For each simulated environment (in Stage), we incrementally map it
(with GMapping) following a closest frontier exploration strategy [39]. Since
the information we use for prediction comes from the parts of the environment
610 already observed, we expect the quality of our estimate to increase with the
amount of information in the grid map M . Consequently, partial grid maps ob-
tained at different stages of exploration are considered. We discard maps from
the early stages of exploration (those that covers less than 20% of the area of
the environment) because they do not contain enough information for prediction
615 (almost all the frontiers are open frontiers).

A particular case, that happens at final stages of exploration when maps are
almost complete, is that of a partial room for which a very small portion, as a
corner behind a door, has not been observed. The layout of such room can thus
be trivially reconstructed by considering only faces in F'_i (see Section 3.2). Those
620 rooms cannot be expanded anymore since they are usually surrounded by other
fully observed rooms and the unobserved portion of their area is usually less
than 1 m^2 . We do not consider such rooms in our evaluation because their layout
could be retrieved very reliably and their inclusion could cause an overestimate
of the overall prediction accuracy returned by the IoU. Hence, we evaluate the
625 IoU only for rooms for which an actual prediction of the layout is made, namely
for those partial rooms where $\hat{F}_i^* \neq \{\}$. Overall, we consider for evaluation 482
partial rooms P obtained in 196 grid maps M .

The results are shown in Table 2. Overall, considering maps obtained at
different stages of exploration altogether, we obtain an average IoU of 0.70,

$expl$	$\#M$	$ P $	IoU
$\geq 20\%$	196	482	0.70 (0.18)
$\geq 25\%$	173	412	0.72 (0.17)
$\geq 50\%$	100	203	0.75 (0.16)
$\geq 75\%$	25	40	0.82 (0.11)
$\geq 90\%$	18	23	0.86 (0.06)

Table 2: Accuracy (IoU) of our predicted layout obtained incrementally exploring 20 large-scale buildings. $expl$ is the percentage of the area explored by the robot, $\#M$ is the number of maps that cover at least $expl$ area, $|P|$ is the number of partial rooms in such maps. We indicate the average (standard deviation) IoU of our layout prediction method on partially observed rooms.

630 over all the partial rooms for which a layout has been predicted. However, it can be seen how the performance in predicting the layout of partial rooms increases during exploration. We obtain a performance of 0.75 after half of the environment has been observed and of 0.86 when the 90% of the environment has been observed. It is also interesting to point out that the standard deviation
635 of the IoU decreases when the explored area increases.

As discussed in Section 2, alternative methods require a library of environments, consider fully or almost fully observed rooms, or do not predict the geometrical structure of the unknown parts of the environment. So, comparing them with our approach is not immediate nor fair.

640 Figure 10 shows four examples of layout prediction starting from grid maps with partial rooms. The examples are relative to four different environments at different stages of exploration in order to assess the proposed method in different settings.

The first column of Figure 10 shows the case of a very large building of
645 which several rooms have been already observed, and where the map M covers only the 30% of the total area. Despite the fact that our method cannot infer correctly the T-shaped structure of the building from the partial grid map, we

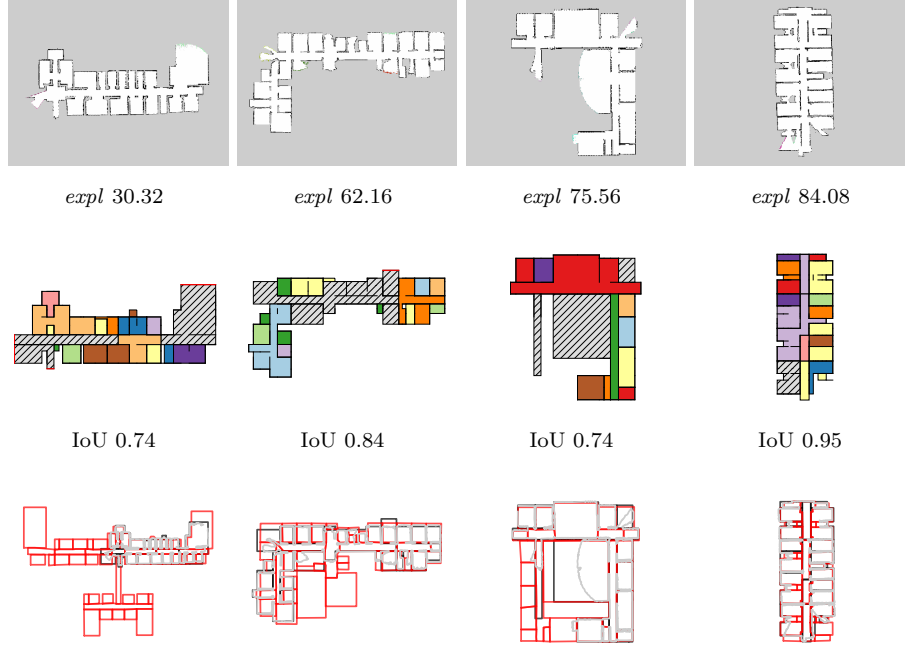


Figure 10: Four reconstructed layouts for different environments. First row: partial grid maps (*expl* is the percentage of explored area wrt the total area of the environment). Second row: reconstructed layouts (with average IoU over the partial rooms). Third row: partial grid maps (gray), reconstructed layouts (black), and ground truth (red).

are able to correctly identify the open frontiers (in red) that lead to other parts of the building. The average IoU is 0.74.

650 The second column of Figure 10 shows how our method can infer the shape of several rooms in a halfway-known building. It can be seen that the predicted shape of partial rooms is consistent wrt their real aspect, despite the fact that not all the walls have been fully observed, as it can be seen by looking at the rooms that are on the corridor (in orange) of the top-right of the map, even if
655 the reconstruction of such corridor is under-segmented.

The third column of Figure 10 shows how our method can correctly estimate the presence of a long corridor in the left part of the building, similar to the long corridor that has been observed in the right part of the building that has

already been explored.

660 The fourth column of Figure 10 presents an almost fully observed environment, where the layout of the few partial rooms in the bottom-left part of the building is correctly predicted using the structural information obtained from the other rooms. This example shows that our method performs well with partial grid maps of complex buildings and copes with gaps in the maps (small
665 frontiers left behind corners) and (relatively small) inaccuracies, like the fact that the metric map is slightly distorted.

We now discuss layout prediction for two environments at different stages of exploration. These two examples show what type of knowledge can be inferred by using our proposed approach during an exploration run. We evaluate the
670 results obtained when predicting the layout of partially observed rooms using the IoU metric and the results obtained when reconstructing the layout from the complete map at the end of exploration using the forward accuracy and backward accuracy metrics of Section 4.1.

The predictions at different exploration stages are independent of each other;
675 namely, the prediction performed from grid map M_t at time step t is not used to compute the prediction from M_{t+1} . While exploration progresses, our method is able to exploit more knowledge about the structure of the environment. This results in an increased accuracy (IoU) of the predicted layout of partially observed rooms. However, the increase of the IoU cannot be strictly monotonic,
680 because the partial rooms whose layout is predicted change at each stage.

Figure 11 presents a first example of the application of our approach to an environment at different stages of exploration. Note that, as expected, the accuracy of the predicted layout improves with the completeness of the grid map and, consequently, with the number of structural features discovered in
685 the environment. The predicted layout of partial rooms is accurate from the second stage shown in the figure, namely after both the upper and lower walls that constitute the border of the image are observed. The correctness of the reconstructed layout of fully observed rooms improves as exploration progresses. During the early stages of exploration, walls inside the environment are correctly

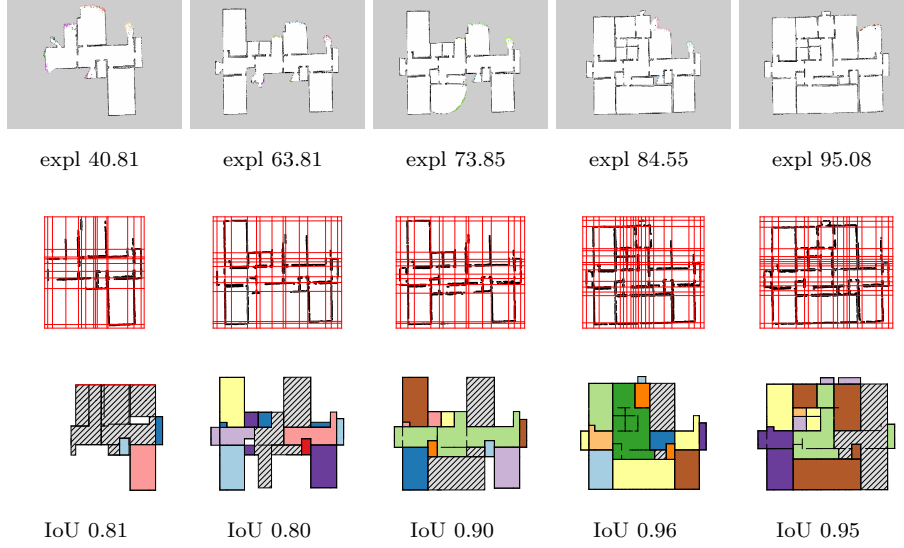


Figure 11: First example of application of our approach to increasingly complete grid maps of an environment.

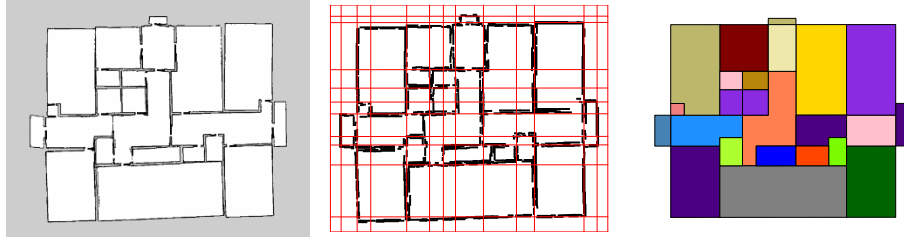


Figure 12: Reconstructed layout of the complete map of Figure 11.

reconstructed. However, the segmentation of the space into rooms is often incorrect, resulting in under-segmentation. This is particularly evident in the third and fourth stages shown in Figure 11, when exploration is between 70% and 85%. In both cases, the central corridor (in green) is considered as a part of a larger room. However, when exploration is complete, the full layout \mathcal{L} is correctly retrieved with $A_{FC} = 86.36$ and $A_{BC} = 88.74$, as shown in Figure 12.

Figure 13 shows the second example. Our method identifies correctly open frontiers that lead to other portions of the environment at early stages of explo-

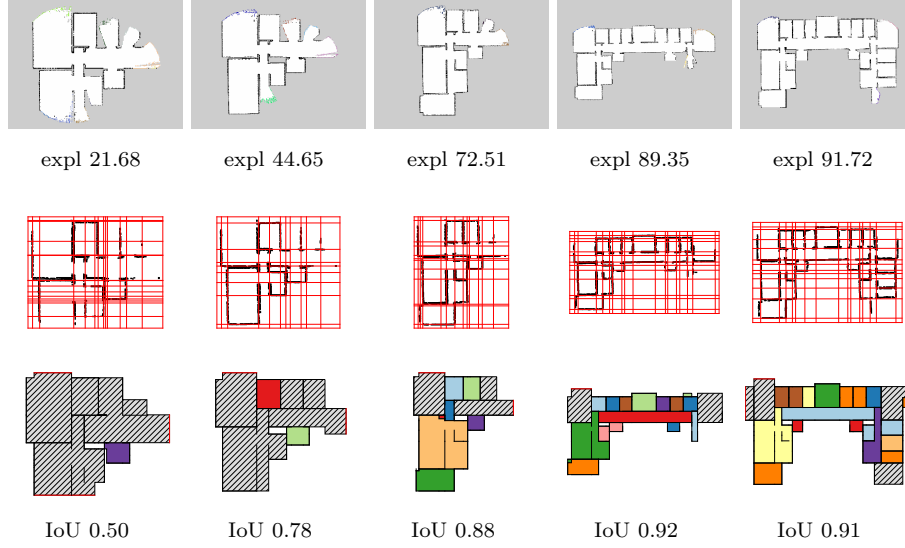


Figure 13: Second example of application of our approach to increasingly complete grid maps of an environment.

ration (e.g., the central corridor in the first three stages shown in the figure). After the largest part of the environment has been observed (fourth and fifth stages), our method is able to provide an accurate prediction of the layout. In particular, in the last stage, our method is able to predict the layout of the last room at the bottom of the right corridor by using the information obtained on the left corridor of the building. While exploration progresses, the accuracy of the layout reconstruction of fully observed rooms improves as well and, at the end of exploration, we obtain $A_{FC} = 89.6$ and $A_{BC} = 86.5$ (Figure 14).

Finally, in Figure 15 it can be seen that our method is able to correctly identify walls even when a small portion of the environment has been explored. More precisely, a good prediction can be obtained when the bounding walls of the environment or one of the corridors are observed. Using such information, our method is able to provide meaningful knowledge about the structure of the environment. Open frontiers, in red, are detected at the end of the two corridors that led to the rest of the building.

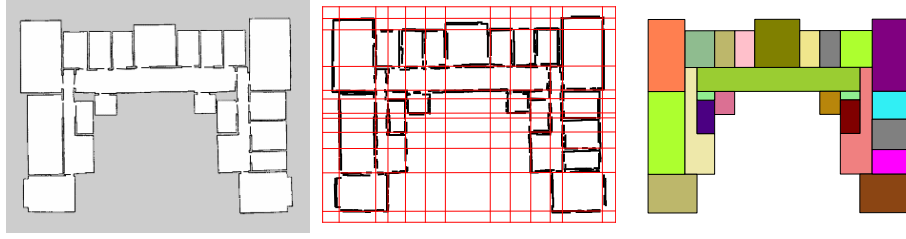


Figure 14: Reconstructed layout of the complete map of Figure 13.

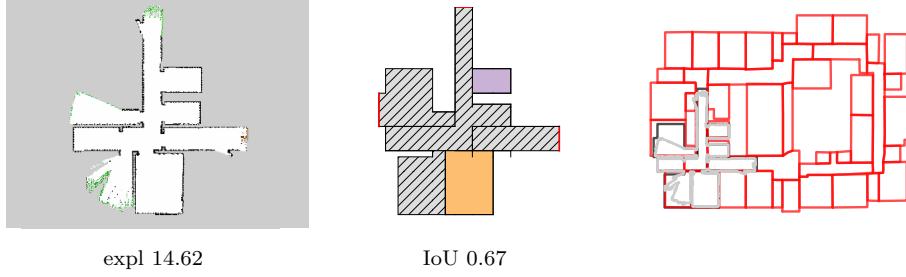


Figure 15: Retrieved layout at an early stage of exploration.

4.5. Computing time and application to an online exploratino process

Our proposed method retrieves a layout \mathcal{L} (both from complete and from
 715 incomplete maps) in less than 10s on a commercial laptop, meaning that an
 updated version of \mathcal{L} is available to a robot every 10s, at most. This computing
 time is compatible with the online use of our method in several applications,
 where a mobile robot can use \mathcal{L} for decision making.

In this section, we provide an example of such an application, namely the
 720 online use of our method for the exploration of initially unknown environments
 for map building. During this process, a robot iteratively selects the next best
 location to visit in order to quickly cover the entirety of an environment, ac-
 cording to an exploration strategy. A fully detailed account of what is reported
 here is available in [8].

725 The exploration process works with the robot that iteratively performs the
 following steps:

- (i) it extracts from the map M a list of frontiers and, for each frontier, it selects a candidate location p as the middle cell of the frontier;
- (ii) it evaluates the candidate locations $\{p\}$ using an evaluation function $f()$;
- 730 (iii) it selects the next best location p^* as $p^* = \arg \max_p f(p)$;
- (iv) it plans and follows a path towards p^* , integrating the perceptions acquired along the way into the map M ; and
- (v) once it reaches p^* , it restarts from (i).

In this process, we use the layout \mathcal{L} in order to inform the evaluation of candidate
735 locations during step (ii). In particular, \mathcal{L} is used to estimate the amount of new area that is visible from a candidate location p , which is one of the components of $f()$. (Please see [8] for full details.) What is important here is that step (ii) happens at scales of dozens of seconds (considering the time spent by the robot in traveling to a new location a large environment). If no updated \mathcal{L} is
740 available when executing step (ii), the robot waits before proceeding to step (iii). We empirically observed that this never happened in our experiments, indicating that obtaining an updated \mathcal{L} is not a bottleneck for the robot during exploration. In what follows, we present some results in order to evaluate the online use of our method in this exploration context.

745 We implement our exploration framework in ROS, using the ROS navigation stack. Explorations are performed in 10 large-scale buildings (from 1000 m² to 3500 m²) simulated in Stage (the same environments of Section 4.4, as that of Figure 13), using a simulated robot equipped with a laser range scanner with a field of view of 180° and a range of 6 m and traveling with a speed of 0.8 m/s.
750 Our method is embedded into a dedicated ROS node. The ROS node receives the latest metric map M from the SLAM algorithm (in this case the ROS implementation of GMapping) and a list of frontiers computed from M by an external node. After computing \mathcal{L} , the node publishes it on a dedicated topic. Note that outsourcing the computation of the frontiers (which is one of the differences between our implementation and that of [10] and [11]) allows the introduction of
755

parallelization within our method, as the process of identifying the frontiers inside a grid map can be time consuming [40]. While the computation of frontiers is not a core part of our method, the availability of a list of frontiers is required to compute \mathcal{L} (specifically, to predict \mathcal{L}_P , as discussed in Section 3.2). Frontier
760 computation amounts on average to about 40% of total computing time, with a maximum around 60%.

In each of the environments, we perform 10 runs, for a total of 100 runs, in which the robot fully explores the environments. The total area of the environments and the time required to fully explore them (averaged over 10 runs) are
765 reported in Table 3. The complete exploration system, including our module for retrieving \mathcal{L} , the SLAM module, and the ROS navigation stack (as well as the robot simulation), is executed on a commercial laptop, without any dedicated hardware. As said, in none of the runs, the robot has to wait for the computation of \mathcal{L} to end in step (ii). Thus, we have empirically shown that our method
770 can be used for supporting the online decision-making of an exploring robot.

In general, the time required to compute \mathcal{L} is proportional to the number of rooms in each map. However, as the rooms are considered sequentially, computing the layout of a room is independent from computing the layout of other rooms. The mere prediction of the layout of a single room (i.e., the body
775 of the for cycle in Algorithm 2) for the environments considered in Section 4.4 takes on average 0.028 s. Note that although we could parallelize the retrieval of the layout for different rooms, we do not exploit this opportunity in our experiments.

4.6. Evaluation with real-world maps

In order to assess the ability of our proposed method to reconstruct and predict the layout starting from metric maps acquired by real robots, which usually feature the presence of furniture, clutter, and non-structural elements, we present two examples of layout reconstruction and prediction performed starting from such metric maps. In these examples, the layout of the environment
785 is retrieved without applying significant changes to the proposed method used

Environment	Area	Time
#1	1420 m ²	1715 s
#2	1000 m ²	1636 s
#3	3410 m ²	4187 s
#4	1270 m ²	2482 s
#5	1740 m ²	2477 s
#6	2820 m ²	2842 s
#7	3180 m ²	4265 s
#8	2150 m ²	3537 s
#9	1950 m ²	4275 s
#10	1800 m ²	3508 s

Table 3: Area and average time required by a robot to perform a full exploration of environments where our method is used to retrieve the layout \mathcal{L} that informs the selection of the next location to reach.

for the more “regular” metric maps obtained from simulations and described in the previous sections. The only change, as we discuss in more detail in the following section, regards the values of two parameters.

Metric maps obtained in cluttered environments present a large number of
790 frontiers (of different sizes) that are due to unexplored parts of the environment that are behind some obstacles. For this reason, as described in Section 3.2, our method considers all the rooms in which the frontiers lie as partially observed and thus their layout is predicted. Our method, consequently, tries to remove clutter and to fill gaps to obtain a clean and abstract representation of each
795 room, for example approximating as rectangles the rooms with small columns close to the walls. Despite this fact, and the fact that metric maps obtained in real-world environments are significantly different from those obtained in simulation, our method is able to successfully reconstruct and predict their layout.

800 Figure 16 presents the first example, relative to a house environment from

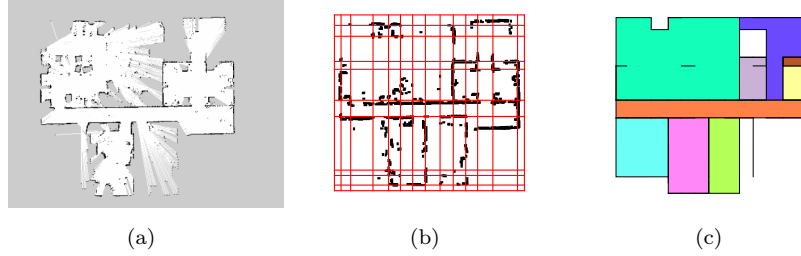


Figure 16: Retrieved layout of a cluttered map obtained by a real robot (from [41]).

[41]. The map of Figure 16a presents clutter in all of the rooms, with only the main corridor in the middle being completely observed and mapped by the robot. The big room in the upper-left part of the map has been almost fully observed by the robot, but presents several unseen parts due to the clutter. However, the representative lines obtained from the observed portion of the environment (Figure 16b) provide a segmentation of the map into a few meaningful faces. The reconstructed layout, in Figure 16c, provides a reasonable guess of the structure of the environment, as shown by the fact that the room in the upper-left part of the map is correctly predicted to be a big almost-rectangular room and that the layout of the three rooms connected to the corridor at the bottom of the map (in light-blue, pink, and green, respectively) are successfully predicted. Note that all of the rooms of this example, except the central corridor, are classified as partial rooms due to the presence of frontiers. However, for better clarity, we adopt here a color scheme different of the one used for the other examples of Section 4.4, where partially observed rooms are denoted with a dashed gray pattern.

In the second example, we applied our method to one of the publicly available maps of [42] (Figure 17). Also in this case, the map of Figure 17a presents clutter in most of the rooms. Moreover, some rooms connected to the upper part of the corridor are only partially observed, as the robot has not entered in them while mapping. In Figure 17b, it can be seen how our method is able to provide a valid segmentation of the environment, which is then used in Figure 17c to

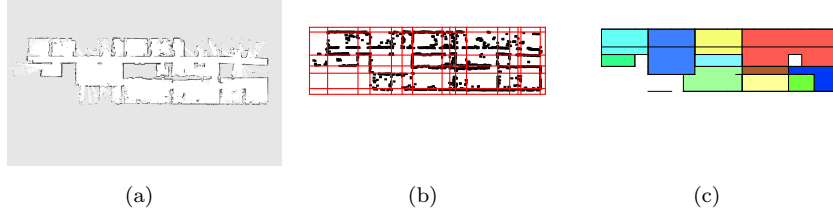


Figure 17: Retrieved layout of a cluttered map obtained by a real robot (from [42]).

reconstruct and predict the layout of the environment in a meaningful way. In this example, due to the presence of several partially observed rooms, our method tends to under-segment the rooms, in particular those connected to the upper corridor.

This fact suggests that, despite segmentation of the environment using few representative lines could effectively predict the layout of the environment, the clustering method used for grouping faces into rooms could be adapted in order to cope better with cluttered settings, as in the case of maps obtained with real robots. For example, it would be interesting to execute the proposed method online by filtering out clutter using a method similar to that of [43].

4.7. Discussion

The examples presented above show that our proposed method is able to reconstruct and predict the layout of environments from metric maps in several different settings. At the core of the method there is the fact that it segments the environment using relatively *few* representative lines which successfully capture the underlying structure of buildings and their regularities. Indeed, these representative lines are used to retrieve the direction of the main walls, ignoring clutter and small details that can be found in metric maps (see, for a different approach, [18]).

For this reason, our method can be easily adapted without changes to different settings, provided that representative lines are determined according to the

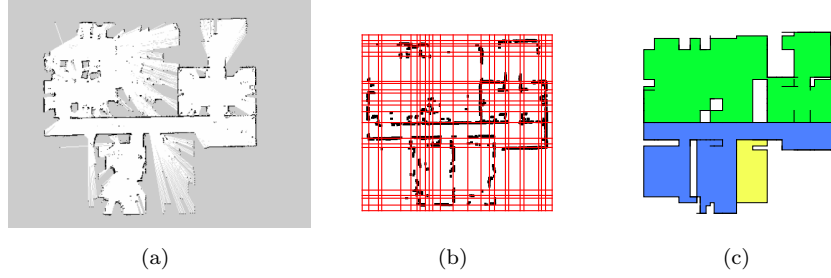


Figure 18: Reconstructed layout of the map of Figure 16, where more representative lines are used.

main walls of the environment. From our experience, the only modifications re-
845 quired in order to adapt our method to different input metric maps are relative
to the identification of the line segments S , using the probabilistic Hough line
transform (Figure 2c), and to the clustering of collinear line segments in sets
 $W_{j,k}$, which are later used for computing the representative lines and the faces.
In particular, we observed that the outcome of our method is mainly influenced
850 by two parameters $\theta = (\text{minLinPoints}, \text{wall})$, namely the minimum number of
points that could be considered as a line and the threshold for merging collinear
line segments. In general, we set $\theta = (7, 10)$, but we change them for input
metric maps obtained by real robots. Specifically, we use $\theta = (40, 20)$ for the
map of Figure 16 and $\theta = (60, 12)$ for the map of Figure 17. Consider, for
855 example, Figure 18, which shows the layout reconstruction of the same environ-
ment of Figure 16 when using $\theta = (30, 10)$, which results in a larger number
of representative lines detected within the map and, consequently, in a larger
number of faces. Comparing Figure 16b to Figure 18b and Figure 16c to Figure
18c, a larger number of representative lines results into a less abstract and less
860 “clean” representation of the environment. The rooms identified in Figure 18c
represent not only the main structure of the rooms, but also small rooms due to
clutter. With more suitable values $\theta = (40, 20)$ our proposed method, as shown
in Figure 16c, produces a cleaner and more abstract representation of rooms’
shapes, representing each room with a polygon that is retrieved from the main

walls and better capturing the underlying structure of the entire building. As a consequence, although several parameters are used in the steps composing our method, only the two parameters θ should be set to identify, within the input metric map M , only those line segments S that lie along the main walls of the building.

In all the examples discussed above, experimental evaluation has been performed considering as metric maps M grid maps whose cells can assume (possibly after thresholding the maps built by GMapping) only three values: free, obstacle, and unknown. This allows to apply our method to input metric maps coming from different sources, like evacuation maps, blueprints, and partial maps built by real robots (see Sections 4.3 and 4.6), as we consider as candidate cells for wall identification only those cells labeled as obstacle. However, more informative map representations based on fine-grained probability distributions on the occupancy of cells can be adopted. These maps, as those based on the Dempster-Shafer theory [44] and the NDT maps [45], represent the degree of uncertainty about the occupancy of each portion of the environment and have been proven to be particularly useful when considering long-term and dynamic settings, in which distinguishing between stable and static map elements (e.g., walls, which typically correspond to highly certain occupancy) and dynamic and transient ones (e.g., moving obstacles, which typically correspond to uncertain occupancy) is particularly important. The integration of such knowledge into our method could be beneficial and information about the occupancy probability of cells might be useful in order to retrieve good candidates for the representative lines of the walls in the environment.

Our method is based on 2D metric maps, typically grid maps obtained with 2D laser range scanners, that, as we discussed previously, are simpler and less rich representations than the 3D point clouds used for example in [5, 21]. 3D maps provide more knowledge about the environment (e.g., they allow to retrieve the walls of the environment from the detection of the ceilings or from the identification of the doors as gaps on planar surfaces) at the expenses of requiring more complex sensors (e.g., Velodynes or RGBD cameras) and more complex

SLAM processing, implying larger computing efforts. Our approach, requiring only 2D metric maps, can be applied to input maps obtained by cheap and simple robots, like domestic robots (e.g., vacuum cleaning and socially assistive robots) and research platforms (e.g, TurtleBots). The layout returned by our method can be used for several tasks, starting from identifying rooms within the map and regularizing inaccurate portions of the maps. For instance, a map can have some “gaps” due to the fact that the robot has observed and mapped only portions of some rooms while passing (without entering) in front of their open doors. This lack of knowledge constitutes a limitation to the robot’s operations, because it cannot plan to perform any task in the unmapped areas. Our method could be used to smooth such a limitation by predicting portions of the map that are not fully observed at mapping time. An example of a possible online use of our method is provided in Section 4.5 and detailed in [8]. Moreover, an adaptation of our method to predict the layout of completely unexplored rooms that are behind closed doors is presented in [46]. Finally, in [47] our method is embedded in a framework for detecting robust features in indoor maps in order to evaluate map quality.

While the current performance of our method allows its use in some online applications of ground robots, fast-moving robots (e.g., UAVs) may require the availability of a fresh \mathcal{L} at higher frequencies. In this case, our method can be further improved as follows. Some steps, like the computation of line segments from walls and the computation of frontiers, that extract rather stable features about the environment, can be performed at low frequency or on-demand upon major changes in the metric map M . Other steps, like the prediction of the layout of rooms, can be performed as a separate process at higher frequency and only for rooms for which the metric map M has significantly changed. Such high-frequency updates can be obtained up to 5 Hz, according to results reported at the end of Section 4.5. However, we leave such improvements as future work.

925 5. Conclusions

In this paper, we have presented a method for retrieving the layout of an indoor environment given its 2D metric map, like the one that could be obtained from laser range scanners onboard of autonomous mobile robots. Our method reconstructs the layout of those rooms which have been completely observed by the robot and predicts a possible layout of rooms that have been only partially
930 observed, exploiting structural features extracted from the observed parts of the environment.

Experimental results show that the proposed method performs well and is able to cope with complete and partial metric maps of large-scale buildings,
935 not only when the input maps are grid maps, but also with blueprints and evacuation maps. Experiments performed using maps at different degrees of completion show that our method is able to predict a possible layout, even when only relatively small portions of the environment have been covered by the robot.

Future work, in addition to what has been mentioned along the paper, includes the combination of our method with room segmentation approaches to investigate if better performance can be obtained by looking both at the global structural features and at the local features of the buildings. Our method could be extended to input maps that can be transformed in 2D maps composed of
945 line segments (e.g., 3D point clouds). The use of sketch maps [37, 38] as input is also a promising further development. Improvements of the proposed method include, for example, enlarging the family \mathcal{F} of candidate sets of faces, by using a heuristic to efficiently search it, and by exploiting regularities identified in other environments in the utility function $\Phi()$. It would be interesting
950 to integrate our method with prior knowledge of the floor plans and assess its performance in partial maps with increasingly larger levels of noise and clutter. Moreover, the proposed method could be improved by detecting and explicitly exploiting symmetries between different parts of the building, for example using the methods from [28, 48]. Finally, we look for other applications of the method

955 proposed in this paper, beyond those relative to exploration [8, 9].

References

- [1] R. Bormann, F. Jordan, W. Li, J. Hampp, M. Hägele, Room segmentation: Survey, implementation, and analysis, in: Proc. ICRA, 2016, pp. 1019–1026.
- 960 [2] A. Quattrini Li, R. Cipolleschi, M. Giusto, F. Amigoni, A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings, *Auton Robot* 40 (4) (2016) 581–597.
- [3] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, The MIT Press, 2005.
- 965 [4] Z. Liu, G. von Wichert, A generalizable knowledge framework for semantic indoor mapping based on Markov logic networks and data driven MCMC, *Future Gener Comp Sy* 36 (2014) 42–56.
- [5] I. Armeni, O. Sener, A. Zamir, H. Jiang, I. Brilakis, M. Fischer, S. Savarese, 3D semantic parsing of large-scale indoor spaces, in: Proc. CVPR, 2016, pp. 1534–1543.
- 970 [6] C. Mura, O. Mattausch, A. Villanueva, E. Gobbetti, R. Pajarola, Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts, *Comput Graph* 44 (2014) 20–32.
- [7] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, T. Krajník, Artificial intelligence for long-term robot autonomy: A survey, *IEEE Robot Autom Lett* 3 (4) (2018) 4023–4030.
- 975 [8] M. Luperto, L. Fochetta, F. Amigoni, Exploration of indoor environments through predicting the layout of partially observed rooms, in: Proc. AA-MAS, 2021.

- 980 [9] M. Luperto, M. Antonazzi, F. Amigoni, N. A. Borghese, Robot exploration of indoor environments using incomplete and inaccurate prior knowledge, *Robot Auton Syst* 133 (2020) 103622.
- [10] M. Luperto, F. Amigoni, Extracting structure of buildings using layout reconstruction, in: *Proc. IAS-15*, 2018, pp. 652–667.
- 985 [11] M. Luperto, F. Amigoni, Predicting the layout of partially observed rooms from grid maps, in: *Proc. ICRA*, 2019, pp. 6898–6904.
- [12] S. Thrun, Learning metric-topological maps for indoor mobile robot navigation, *Artif Intell* 99 (1) (1998) 21–71.
- [13] E. Brunskill, T. Kollar, N. Roy, Topological mapping using spectral clustering and classification, in: *Proc. IROS*, 2007, pp. 3491–3496.
- 990 [14] O. Mozo, Semantic labeling of places with mobile robots, Vol. 61 of *Springer Tracts in Advanced Robotics*, Springer, 2010.
- [15] S. Friedman, H. Pasula, D. Fox, Voronoi random fields: Extracting the topological structure of indoor environments via place labeling, in: *Proc. IJCAI*, 2007, pp. 2109–2114.
- 995 [16] K. Sjo, Semantic map segmentation using function-based energy maximization, in: *Proc. ICRA*, 2012, pp. 4066–4073.
- [17] P. Buschka, A. Saffiotti, A virtual sensor for room detection, in: *Proc. IROS*, 2002, pp. 637–642.
- 1000 [18] R. Capobianco, G. Gemignani, D. Bloisi, D. Nardi, L. Iocchi, Automatic extraction of structural representations of environments, in: *Proc. IAS-13*, 2014, pp. 721–733.
- [19] S. Oesau, F. Lafarge, P. Alliez, Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut, *ISPRS J Photogramm* 90 (2014) 68–82.
- 1005

- [20] S. Ochmann, R. Vock, R. Wessel, R. Klein, Automatic reconstruction of parametric building models from indoor point clouds, *Comput Graph* 54 (2016) 94–103.
- [21] R. Ambrus, S. Claici, A. Wendt, Automatic room segmentation from un-
 1010 structured 3-D data of indoor environments, *IEEE RA-L* 2 (2) (2017) 749–756.
- [22] D. Perea Ström, I. Bogoslavskyi, C. Stachniss, Robust exploration and homing for autonomous robots, *Robot Auton Syst* 90 (2017) 125 – 135.
- [23] A. Smith, G. Hollinger, Distributed inference-based multi-robot explo-
 1015 ration, *Auton Robot* 42 (8) (2018) 1651–1668.
- [24] J. Chang, G. Lee, Y.-H. Lu, C. Hu, P-SLAM: Simultaneous localization and mapping with environmental-structure prediction, *IEEE T Robot* 23 (2) (2007) 281–293.
- [25] J. Caley, N. Lawrance, G. Hollinger, Deep learning of structured environ-
 1020 ments for robot search, in: *Proc. IROS*, 2016, pp. 3987–3992.
- [26] A. Pronobis, P. Jensfelt, Large-scale semantic mapping and reasoning with heterogeneous modalities, in: *Proc. ICRA*, 2012, pp. 3515–3522.
- [27] A. Aydemir, P. Jensfelt, J. Folkesson, What can we learn from 38,000 rooms? Reasoning about unexplored space in indoor environments, in:
 1025 *Proc. IROS*, 2012, pp. 4675–4682.
- [28] M. Luperto, F. Amigoni, Predicting the global structure of indoor environments: A constructive machine learning approach, *Auton Robot* 43 (4) (2019) 813–835.
- [29] S. Hemachandra, M. Walter, S. Tellex, S. Teller, Learning spatial-semantic
 1030 representations from natural language descriptions and scene classifications, in: *Proc. ICRA*, 2014, pp. 2623–2630.

- [30] J. Canny, A computational approach to edge detection, *IEEE T Pattern Anal* 8 (6) (1986) 679–698.
- [31] N. Kiryati, Y. Eldar, A. M. Bruckstein, A probabilistic Hough transform,
1035 *Pattern Recogn* 24 (4) (1991) 303–316.
- [32] S. Suzuki, K. Abe, Topological structural analysis of digitized binary images by border following, *Comput Vision Graph* 30 (1) (1985) 32–46.
- [33] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, *IEEE T Pattern Anal* 24 (5) (2002) 603–619.
- 1040 [34] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: *Proc. KDD*, 1996, pp. 226–231.
- [35] G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with Rao-Blackwellized particle filters, *IEEE T Robot* 23 (2007) 34–46.
- 1045 [36] W. Winterhalter, F. Fleckenstein, B. Steder, L. Spinello, W. Burgard, Accurate indoor localization for rgb-d smartphones and tablets given 2D floor plans, in: *Proc. IROS*, 2015, pp. 3138–3143.
- [37] B. Behzadian, P. Agarwal, W. Burgard, G. D. Tipaldi, Monte Carlo localization in hand-drawn maps, in: *Proc. IROS*, 2015, pp. 4291–4296.
- 1050 [38] F. Boniardi, B. Behzadian, W. Burgard, G. D. Tipaldi, Robot navigation in hand-drawn sketched maps, in: *Proc. ECMR*, 2015, pp. 1–6.
- [39] B. Yamauchi, A frontier-based approach for autonomous exploration, in: *Proc. CIRA*, 1997, pp. 146–151.
- [40] M. Keidar, G. Kaminka, Efficient frontier detection for robot exploration,
1055 *Int J Robot Res* 33 (2) (2014) 215–236.
- [41] J. Ruiz-Sarmiento, C. Galindo, J. González-Jiménez, Robot@Home, a robotic dataset for semantic mapping of home environments, *Int J Robot Res* 36 (2) (2017) 131–141.

- [42] A. Howard, N. Roy, The robotics data set repository (Radish) (2003).
 1060 URL <http://radish.sourceforge.net/>
- [43] R. Ambrus, N. Bore, J. Folkesson, P. Jensfelt, Meta-rooms: Building and maintaining long term spatial models in a dynamic world, in: Proc. ICRA, 2014, pp. 1854–1861.
- [44] D. Pagac, E. Nebot, H. Durrant-Whyte, An evidential approach to map-
 1065 building for autonomous vehicles, IEEE T Robotic Autom 14 (4) (1998) 623–629.
- [45] E. Einhorn, H.-M. Gross, Generic NDT mapping in dynamic environments and its application for lifelong SLAM, Robot Auton Syst 69 (2015) 28–39.
- [46] M. Luperto, F. Amadelli, F. Amigoni, Completing robot maps by predicting
 1070 the layout of rooms behind closed doors, in: Proc. ECMR, 2021.
- [47] T. Kucner, M. Luperto, S. Lowry, M. Magnusson, A. Lilienthal, Robust frequency-based structure extraction, in: Proc. ICRA, 2021.
- [48] M. Luperto, A. Riva, F. Amigoni, Semantic classification by reasoning on the whole structure of buildings using statistical relational learning techniques, in: Proc. ICRA, 2017, pp. 2562–2568.
 1075