Autonomous hierarchy creation for computationally feasible near-optimal path planning in large environments

Jelena Gregorić[a,*], Marija Seder[a], Ivan Petrović[a]

[a]University of Zagreb, Faculty of Electrical Engineering and Computing,
Department of Control and Computer Engineering,
Laboratory for Autonomous Systems and Mobile Robotics (LAMOR),
Unska 3, Zagreb, HR-10000, Croatia

Abstract

Mobile robots have increasingly been used in warehouses, factories, open spaces, on roads, and urban areas to transport goods and people. Path planning in complex and large environments using classical graph-based search is computationally too intensive. One of the main challenges of mobile robotics is describing the large robot's workspaces with a reduction of graph complexity and improvement of path planning efficiency. A representation by a hierarchical graph (H-graph) facilitates graph creation and reduces the complexity of path planning. In this paper, we present an algorithm for autonomously generating a hierarchical graph of the environment from floor plans. The hierarchical abstraction depicts the environment in levels, from the most detailed to the most abstract representation of the environment, where pre-computed partial paths at the most detailed level are graph edges at a higher level. We use the E* algorithm to find partial paths in the most detailed abstraction level, and we propose the extraction of higher levels automatically from lower levels. We verified the proposed H-graph creation on three cases: our University premises as an example of a multi-building environment, a large warehouse with a lot of shelves, and an outdoor urban environment which is represented by a city map. All cases result in five abstract levels. The result shows that the path planning algorithm searches about 50 to 800 times fewer nodes when using the H-graph as the environment representation than when using a plain graph representing the entire environment as an occupancy grid map.

Keywords: H-graph, hierarchical abstraction, large environments, mobile robot, partial path, bridge node

1. Introduction

The main task of mobile robots is to move autonomously through the environment to transport goods and/or people while avoiding obstacles and living beings. The task of the path planning algorithm, which is one of the core components of the navigation system of any autonomous mobile robot, is to calculate a feasible, possibly optimal, path to a given target position and to re-plan the path if the previously planned path is blocked by obstacles. Most path planning algorithms compute a graph of possible paths to the target position using an environment representation based on occupancy grid maps [1, 2] or topological graphs [3, 4] and then use a graph search algorithm such as A* [5], D* [6], TWD* [7], E* [8], etc. to find an optimal path. These path planning algorithms are commonly used in small environments, but in large environments, they become computationally too complex due to the large dimension of the graph representing the environment and are therefore not suitable for online path planning.

The use of mobile robots is rapidly expanding to many different application areas where they are required to operate in large, complex indoor and/or outdoor environments. Some examples of application areas where mobile robots operate in large environments include e-commerce, logistics, and manufacturing industries with large warehouses and factory facilities [9, 10, 11, 40, 41, 42], hospitals and office buildings consisting of a group of interconnected multi-story building clusters [12, 13, 14], urban transportation systems with open spaces, roads, and parking lots [15, 16], etc. Those environments are difficult to represent with graphs using classical approaches. To reduce the complexity of graphs and improve the efficiency of path planning in these applications, it is necessary to use different approaches to environment representation that lead to simpler graphs. The new approach to environment representation must reduce the number of graph nodes to be searched during path planning but retain all the properties required to obtain the best possible path (such as the high resolution of the grid map, avoidance of obstacles, etc.). A suitable choice for representing large environments is a hierarchical graph (H-graph) based on a hierarchical decomposition of the environment [17, 18, 19].

The H-graph is a combination of classical metric and topological maps at different levels of abstraction. The H-graph uses a skeleton of pre-calculated paths (partial paths) between the key points called the bridge nodes. This approach to graph creation represents large environments simultaneously at different

*Corresponding author
Email addresses: jelena.gregoric@fer.hr (Jelena Gregorić), marija.seder@fer.hr (Marija Seder), ivan.petrovic@fer.hr (Ivan Petrović)

resolutions, i.e., at different abstract levels. Low level maps can be used to represent individual rooms in a large building or in a large complex of buildings, without representing the spatial relationships of rooms. For example, metric maps at the lower level can be used to navigate the mobile robot to the goal within a particular room, without considering the maps of other rooms. Higher level maps can be used to represent larger areas of a building, for example as a graph that connects rooms or corridors, without showing the details of rooms. These topological maps can be used to build abstract plans for moving from one room to another.

A lot of research on creating the hierarchical graph exist in the literature [17, 18, 19, 23, 24, 25, 28, 29, 30, 31, 32, 33, 34]. A hierarchical graph is usually created starting from the lowest level by environment segmentation. This ensures that consistency is maintained across the hierarchical levels of the H-graph [19]. Most studies have done environment segmentation using various metrics by reference to a generalized Voronoi graph [20, 21, 22, 23, 24, 25], and others have focused more on dividing a grid map [26]. The watershed segmentation algorithm is a region-based method which yields reliable results on grid map segmentation and has found its application in the creation of hierarchy [18]. Room segmentation via instance semantic segmentation technique (Mask-RCNN) [27] reduces the floorplan graph inference into the reconstruction of multiple polygonal loops, one for each room. It is also possible to use sensory data directly for the creation of a higher level map. A set of images of the robot's environment could be grouped based on both the presence of the automatically extracted visual landmarks and the constraints imposed by the geometry of the environment [28] with the goal to construct a graph. The creation of an H-graph which is done manually [17, 33] is a time-consuming and complex process. Futhermore, created hierarchical graphs contain only two [18, 28, 34] or three [23, 25, 30, 31, 32] abstract levels of hierarchy. In some cases, three levels of hierarchical abstraction are optimal for some environment, but sometimes an H-graph with so few levels is not enough for modeling an environment with multiple floors and buildings and does not allow path planning between different floors or buildings. In order to speed up the process of graph creation and path planning, it is necessary to aim for autonomous hierarchy creation. The algorithm for the autonomous creation of an H-graph presented in [19] maintains consistency across the hierarchical levels and keeps the model consistent even after changes in the environment. However, the algorithm creates only three levels of the hierarchy. Also, in the mentioned works, during the H-graph creation, high attention is given to the extraction of the higher level nodes, but the optimal algorithms for the pre-calculated partial paths finding have not been sufficiently studied which directly affects the quality of a global path in process of hierarchical path planning.

This paper proposes a method for autonomous hierarchy creation with five levels of abstraction, i.e., the creation of an H-graph for path planning between different floors and buildings. The proposed method creates a hierarchy with fewer levels if the conditions for creating all five levels of the hierarchy are not met. Also, the proposed method can be easily extended to create a hierarchy of more than five levels. The algorithm receives the floor plans as input data and creates an H-graph as output. We follow the approach of the room extraction algorithm presented in [19] and expand it to create higher levels automatically from lower levels. We also performed an autonomous selection of nodes representing elevators inside the building and the passages between the buildings. While references in the literature use the standard A* algorithm, D* algorithm and their modifications to find pre-calculated paths [17, 19, 30, 34], we apply the E* algorithm [8, 35] to find partial paths at the most detailed level represented by the occupancy grid map. Partial paths at higher levels are formed by merging partial paths from lower levels. The E* algorithm extends the standard D* algorithm by using interpolation based on the Level Set Method [36] to produce the approximation of the continuous optimal path costs at grid points and provides a path that is very close to the optimal solution [37]. The path is obtained by following the gradient descent of calculated path costs. Using E*, we avoid the resulting partial paths to be a zigzag line with sharp turns, angles of which are limited to an increment of 45° [36, 37]. The process of graph creation is done offline and without any human intervention. The generated H-graph is a good basis for online, inter-building, and outdoor path planning. The proposed algorithm is the extension of our preliminary work presented in a conference paper [38] with an extension to the warehouse and outdoor environment and more experiments. We showed the efficiency of the H-graph creation in three cases of environment. The first case is a multi-building environment that represents the buildings of the Faculty of Electrical Engineering and Computing (FER) of the University of Zagreb. The second case is a large warehouse with a lot of shelves. Finally, the third case is the urban environment which is represented with a city map. The algorithm creates a graph consisting of five abstract levels and can be easily extended to more if needed.

In this paper, we highlight three main contributions:

i. autonomous creation of an H-graph with five levels,
ii. generation of nodes in H-graph that represent the elevators and the passages between the buildings,
iii. autonomous creation of an H-graph with natural close-to-optimal pre-calculated partial paths and autonomous extraction of nodes on all abstract levels.

The rest of the paper is organized as follows. Section II describes the concept of the H-graph. Section III describes the autonomous hierarchy creation. Section IV shows the algorithm validation on three cases: FER as a multi-building environment, warehouse environment, and outdoor urban environment. Also, a comparison of H-graph and plain graph paths and a comparison with other methods is shown in this section. Finally, Section V points out some conclusions.

2. Representation of Large Environments with H-graph

H-Graph is a representation of the environment that enables computationally feasible path planning for mobile robots in large environments. This section introduces the concept of the H-graph [17, 30] and the conditions for ensuring path optimality when using the H-graph.

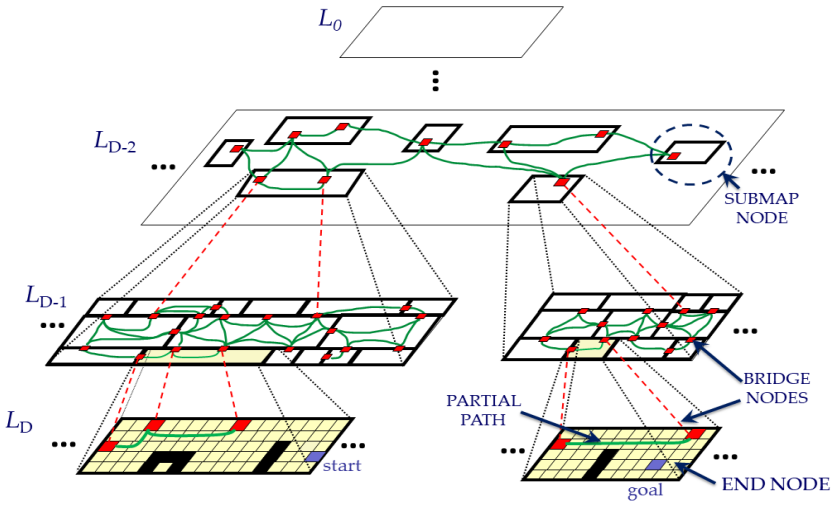


Figure 1: H-graph with marked levels, types of the nodes, and partial paths.

2.1. The Concept of the H-Graph

The H-graph is a sequence of hierarchical levels $L = L_0, L_1, ..., L_D$, where D is the depth of the hierarchy (see Fig. 1). L_0 is the 'root level', which represents the most abstract description of the environment, i.e., the highest level of abstraction. Analogously, the L_D level is the lowest level of hierarchical abstraction and the most detailed description of the environment. Each level L_i $(0 \leq i \leq D)$ is represented by graph $G_i = (N_i, A_i, C_i, W_i, T_i)$, where N_i is a set of nodes, A_i is a set of edges, C_i is a set of Cartesian coordinates for N_i, W_i is a set of weights for A_i and T_i is a set of partial paths. The union of graphs $G_0 \cup G_1 \cup ... \cup G_D$ represents the graph $G = (N, A, C, W, T)$, where $N = \cup_{i=1}^{D} N_i$, $A = \cup_{i=1}^{D} A_i$, $C = \cup_{i=1}^{D} C_i$, $W = \cup_{i=1}^{D} W_i$, $T = \cup_{i=1}^{D} T_i$. An edge $a(n_J, n_K, w_H) \in A$ is defined by three elements n_J, n_K, w_H, where $n_J, n_K \in N$, $n_J \neq n_K$ and $w_H \in W$. Edges are non-directed: a mobile robot can navigate between two nodes in both ways. Cartesian coordinates $c_I \in C$ are defined by (x, y, z), where $x, y, z \in \mathbb{R}$. A weight $w_I \in W$ is a positive real number $(w_I \in \mathbb{R})$. The H-graph represents an extension of the introduced graph G where the H-graph contains edges with weight equal to zero between different levels.

The nodes of the H-graph are classified into three groups: submap, end, and bridge nodes. Submap nodes, noted by $S \subset N$, represent a subset of nodes in a deeper level of abstraction. Submap nodes can be found at all levels of hierarchical abstraction except at the deepest level L_D. A node $s \in N_i$ is a *parent* submap node from the level L_i if it is composed of $K > 1$ *child* nodes from the level L_{i+1}. End nodes are start or goal points that can be selected from a robot path planner. All nodes at the most detailed level (grid map cells) are considered as end nodes. Bridge nodes, noted by $B \subset N$, were divided into two classes in [17]: *vertical bridge nodes* – the vertical connections, i.e., elevators, that connect two floors (submap nodes) in a building, and *horizontal bridge nodes* – the horizontal connections, i.e., doorways, that connect two submap nodes. The concept of vertical bridge nodes does not imply a connection between hierarchical levels of the H-graph but a connection of the environment parts within the same level and allows the robot movement in the vertical direction through the environment (for example vertical movement through the building between different floors). In the creation of the H-graph, the detection of nodes representing elevators and passages between buildings was ensured, but the concept of horizontal and vertical bridge nodes does not ensure connections between hierarchical levels. To connect the levels of the hierarchy, in the following section, we introduce new types of bridge nodes that are based on the detection of horizontal and vertical bridge nodes.

Each submap node has its set of partial paths defined between bridge nodes. The partial path $T(a, b, s)$ is defined by bridge nodes $a, b \in B$ and submap node $s \in S$ as follows:

$$\begin{aligned} T[1] &= a, \qquad T[|T|] = b, \\ T[i] &\in s, \qquad i = 1, ..., |T|, \\ a(T[j], T[j+1], w_{j,j+1}) &\in A, \qquad j = 1, ..., |T| - 1 \ , \end{aligned} \tag{1}$$

where $|T|$ denotes the number of nodes on the partial path T. Partial paths are least-cost paths calculated offline. The cost of the partial path $c(T)$ is defined as the sum of the weights of the edges on the partial path:

$$c(T) := \sum_{i=1}^{|T|-1} w_{i,i+1} \ . \tag{2}$$

From the set of all paths between nodes a and b, where all path elements are within the same submap node s, the partial path $T(a, b, s)$ has the smallest cost of the path $c(T)$. Partial paths are introduced to avoid time-consuming path corrections at deeper levels of abstraction in the case of path replanning and give the possibility of simple graph extension. On the other hand, partial paths and their costs require extra memory space. However, achieving a short online computational time is extremely important for path planning in large dynamic environments.

2.2. Conditions for Path Optimality

To ensure optimality of the global path and reduce the computational complexity of path replanning in real-time, we follow [30] and extend it for a higher number of hierarchical levels. First, we put bridge nodes at the narrow passages to ensure path optimality, and second, we duplicate bridge nodes to maintain consistency through the hierarchy.

The optimality of the global path in the environment is observed by the path cost condition. The cost of the global path $c(P)$ is defined as the sum of the edge weights on this path:

$$c(P) := \sum_{i=1}^{|P|-1} w_{P[i],P[i+1]} \ . \tag{3}$$

The $\Pi(S, G)$ is the set of all paths between two points $S, G \in N$ in the graph $G(N, A, C, W, T)$. The optimal path cost of the path from S to G is defined as follows:

$$g^*(S, G) := \min_P c(P) \ , \tag{4}$$

where $P \in \Pi(S, G)$, and $g^*(S, G) = \infty$ if $\Pi(S, G) = 0$. Optimal path $P^*(S, G)$ is the one with the optimal path cost $g^*(S, G)$:

$$P(S, G) \in \{P \in \Pi(S, G) \mid g(P) = g^*(S, G)\} \ . \tag{5}$$

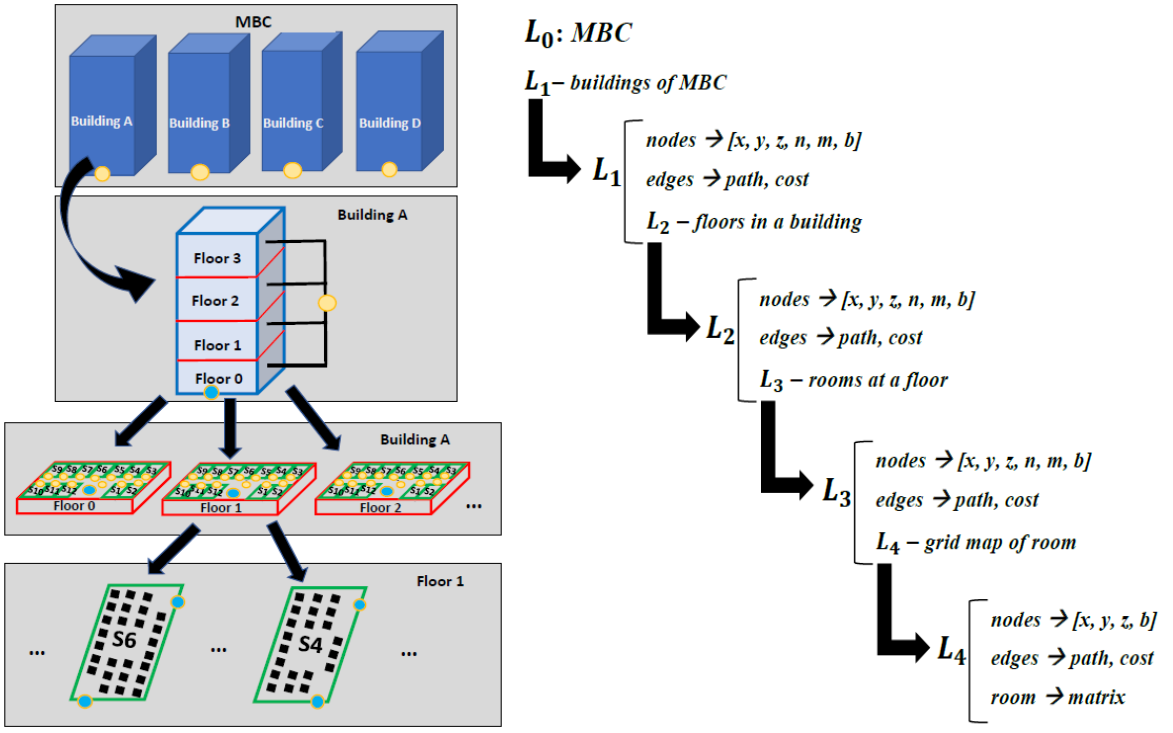
Partial paths in the H-graph are created from the most detailed level represented by the occupancy grid map with the safety cost mask. For this purpose, we apply the E* algorithm which uses interpolation based on the Level Set Method [36] to produce the approximation of the continuous optimal path costs at grid points and provides a path that is very close to the optimal solution [37]. The path is computed as a gradient descent over the estimated cost. Partial paths at higher levels are formed by merging partial paths from lower levels. Also, the global path through the environment is formed by merging partial paths from the H-graph. To ensure the optimality of the partial paths at higher levels of abstraction and the global path, bridge nodes must be placed such that the union of the optimal partial paths between the bridge nodes results in the optimal path. This is ensured by placing bridge nodes in narrow passages, in places where the value of the safety cost mask is the lowest. This position of bridge nodes also helps solve the problem of robots traveling through narrow passages. The safest path in narrow passages is the one in the middle of the passage because in this way the robot minimizes the risk of colliding with an obstacle. Therefore, we placed the bridge nodes exactly in the middle of the passages, at the doors that separate the rooms. In this way, the optimality of the path is achieved, and safety is increased. The free parts of the large environment are geometrically divided into "rooms" and "halls" (these are not necessarily physical rooms and halls but virtual ones). At level L_D, bridge nodes are located on the border between the two rooms (submap nodes) and belong to both rooms, i.e., they have two parent submap nodes.

We organize the higher levels of the hierarchy to allow optimal horizontal and vertical hierarchical path planning. To that goal, we defined a new division of bridge nodes: *regular* and *marginal*. Marginal bridge nodes represent connections to a higher level of the hierarchy, while regular bridge nodes represent connections between two submap nodes at the same level. We duplicate the marginal bridge nodes to the higher level. A duplicated bridge node $m^i \in N_i$ is defined as a newly created node at level L_i, whose coordinates are equal to the coordinates of the marginal bridge node $m^{i+1} \in N_{i+1}$ at level L_{i+1}. This two nodes are connected with an edge $a(m^i, m^{i+1}, w)$, where $w = 0$. At level L_i, these duplicated bridge nodes are again divided into regular and marginal bridge nodes, where marginal bridge nodes are at the borders of two submap nodes of the level above and regular bridge nodes are at the borders of two submap nodes of that level. The determination of marginal bridge nodes is based on the detection of vertical and horizontal nodes, i.e., on the detection of passages between rooms, elevators, and passages between buildings. At level L_D, all bridge nodes are marginal and represent a connection with the level L_{D-1}. End nodes appear only at the lowest level as the grid map cells, bridge nodes can be found at all levels except the root level L_0, and submap nodes are at all levels except the lowest level L_D. For each submap node, partial paths between all possible pairs of marginal bridge nodes are determined. Once partial paths are determined at one level, they form the basis for calculating new partial paths at a higher level.

In addition to the above conditions for optimality, it must be considered that consistency must be maintained through the hierarchical levels of the model, and the model must remain consistent after changes in the environment. This is achieved by creating abstract levels from the most detailed level where an occupancy grid map is used.

3. Autonomous H-graph Creation

This section presents the method for autonomous hierarchy creation based on the concept of the H-graph and the principles of consistency and optimality introduced in the previous section. The H-graph is created offline. The method of H-graph creation is summarized in Algorithm 1 and described in the following. The H-graph is organized into a data structure as the one shown in Fig. 2, which represents the hierarchical decomposition of a multi-building cluster (MBC). This data structure contains a list of *nodes* at each level. The nodes are described by the corresponding (x, y, z) coordinates. The variables n and m are the indexes of the nodes at the border of which the marginal bridge node with the coordinates (x, y, z) was created. The variable b is the index of each node at some level. Each level contains a structure called *edges* that describes all partial paths of that level. The structure *edges* consists of the list of nodes called *path* that stores the paths described in the coordinate space and the real number *cost* that stores the costs of the paths. At each level, there is also a substructure that leads to a lower level. At the lowest level, there is a set of room matrices that contain a room occupancy grid map. The occupancy grid map is continuously updated as the robot moves across the environment according to the detected obstacles by the robot sensors. The presented data structure in Fig.2(b) logically follows the shown decomposition of the environment in Fig. 2(a).



(a) (b)

Figure 2: (a) An illustration of the hierarchical division of a multi-building cluster (MBC); (b) structure as output data from the autonomous hierarchy creation algorithm. Marginal and regular bridge nodes are presented with blue and yellow circles, respectively, while end nodes are presented with black squares that demonstrate the grid map of an individual room.

3.1. Occupancy Grid Map Creation from the Floor Plan

The input data given to the algorithm for the autonomous hierarchy creation are floor plans of all parts of the environment in *png* format and the corresponding parameter files in *yaml* format. The floor plan images could be obtained from a CAD program, Quantum Geographic Information System (QGIS), or automatically via SLAM, and must not contain details such as door markings, room descriptions, elevations, street names, etc. Each *yaml* file contains characteristic parameters that describe each floor plan: the name of the image which is paired with *yaml* file, map resolution, and origin in meters. The map resolution indicates the dimension of the pixels on the loaded image in meters, and the image origin indicates the actual shift of the environment part represented with the image with respect to the user-defined coordinate system for a specific environment, also in meters. Parameter image origin is defined as the 3-D pose and rotation of the lower-left pixel at the image, as $[x, y, z, yaw]$, with yaw as counterclockwise rotation in degrees if the environment represented by a picture should be rotated during the grid map generation to be properly oriented in the H-graph. Additional parameters specify the creation of the occupancy grid map and the hierarchy: the radius of the described circle around the robot's footprint is noted as the radius of the robot in the following, the size of the cell, and the maximal width of the narrow passage. The parameter size of the cell defines the size of one cell in the grid map, i.e., the desired grid map resolution. The algorithm reads the list of *yaml* files. Loading the floor plan and its parameters is followed by the creation of the grid map with the cell occupancy function. The grid map is composed of cells that store the occupancy values which indicate unoccupied and occupied parts of the environment. Obstacles are extended by the radius of the robot so that the robot could be considered as a point in space. The safety cost mask defines the values of the cell occupancy function depending on the proximity of the obstacle. Each cell in the grid map within the safety cost mask around an occupied cell receives a corresponding safety cost value, which depends on its distance from the occupied cell. The cell occupancy function is represented as a function of two variables and denoted as $z(x, y)$. The lowest value of the occupancy function is in cells outside the safety cost mask and

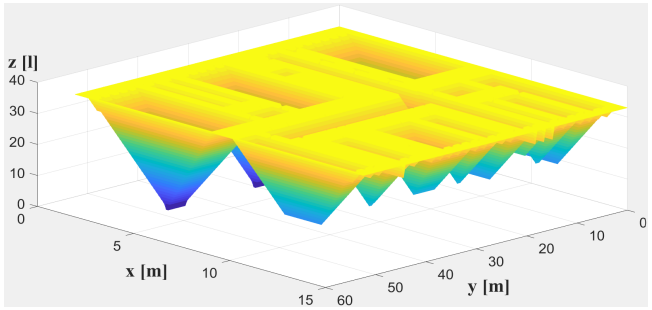


Figure 3: Occupancy grid maps with a safety cost mask represented as cell occupancy function in 3D.

Algorithm 1 H-Graph Creation

Input: List of *yaml* files, robot radius r, cell size a, cost mask cm, max width of the narrow passage $envBN$

for num = 1 . . . $length(yaml_List)$ **do**
  Load parameters from *yaml* file
  Create cost map
  Find narrow passages, bridge nodes and rooms
  for 1 . . . $number_of_rooms$ **do**
    $roomBN \leftarrow$ Select room bridge nodes
    $search_grid \leftarrow$ Select grid of room
    for i = 1 . . . $number_of_roomBN$ **do**
      for j = $(i+1)$ . . . $number_of_roomBN$ **do**
        $start \leftarrow roomBN(i)$
        $goal \leftarrow roomBN(j)$
        $partial_paths \leftarrow$ Create_partial_paths ($search_grid$, $start$, $goal$, E* search)
      end for
    end for
    $nodes \leftarrow roomBN$
    $edges \leftarrow partial_paths$
    $L_4 \leftarrow (nodes, edges, search_grid)$
  end for
  $BN \leftarrow$ Select all bridge nodes
  $search_graph \leftarrow L_4$
  for i = 1 . . . $number_of_BN$ **do**
    for j = $(i+1)$ . . . $number_of_BN$ **do**
      $start \leftarrow BN(i)$
      $goal \leftarrow BN(j)$
      $partial_paths \leftarrow$ Create_partial_paths ($search_graph$, $start$, $goal$, A* search)
    end for
  end for
  $nodes \leftarrow BN$
  $edges \leftarrow partial_paths$
  $L_3 \leftarrow (nodes, edges, L_4)$
  $elevator_nodes \leftarrow$ Find elevator nodes
  $buildings_nodes \leftarrow$ Find nodes between buildings
  $search_graph \leftarrow L_3$
  for i = 1 . . . $number_of_elevator_nodes$ **do**
    for j = $(i+1)$ . . . $number_of_elevator_nodes$ **do**
      $start \leftarrow elevator_nodes(i)$
      $goal \leftarrow elevator_nodes(j)$
      $partial_paths \leftarrow$ Create_partial_paths ($search_graph$, $start$, $goal$, A* search)
    end for
  end for
  $nodes \leftarrow elevator_nodes$
  $edges \leftarrow partial_paths$
  $L_2 \leftarrow (nodes, edges, L_3)$
  $search_graph \leftarrow L_2$
  for i = 1 . . . $number_of_buildings_nodes$ **do**
    for j = $(i+1)$ . . . $number_of_buildings_nodes$ **do**
      $start \leftarrow buildings_nodes(i)$
      $goal \leftarrow buildings_nodes(j)$
      $partial_paths \leftarrow$ Create_partial_paths ($search_graph$, $start$, $goal$, A* search)
    end for
  end for
  $nodes \leftarrow buildings_nodes$
  $edges \leftarrow partial_paths$
  $L_1 \leftarrow (nodes, edges, L_2)$
end for

is equal to 1, while the highest value is in occupied cells, as shown in Fig. 3. The safety cost mask is equal to half of the maximal width of the narrow passage. Finally, the occupancy grid map with the safety cost mask is included as the deepest level L_D of the H-graph.

3.2. Finding Narrow Passages, Bridge Nodes and Rooms

The method for determining narrow passages, bridge nodes, and rooms is performed on the previously created occupancy grid map with the safety cost mask. We adapt the room extraction algorithm from [19] and extend the algorithm to autonomously find locations of elevators, doorways, and crossroads. The algorithm performs the following operations in three steps on the occupancy grid map with the safety cost mask: (i) isolates narrow passages, (ii) determines the positions of bridge nodes based on narrow passages, and (iii) determines groups (rooms) of free cells surrounded by obstacles. A narrow passage is defined as a set of cells where the local minimum of the function z is located and the value of the function is greater than 1. Therefore, the maximal width of a narrow passage that can be detected is equal to two widths of the safety cost mask. Each group of cells, which are defined as a narrow passage, is assigned as a central node with coordinates at the center of the narrow passage.

Bridge nodes must be placed in the middle of narrow passages and at the boundary between two rooms. The position of bridge nodes is not necessarily determined at the door location, but it is certainly in the middle of a narrow passage. Possible positions of bridge nodes are located in the previously determined central nodes of narrow passages, but to declare a node as a bridge node, it must be on the border between two submap nodes. Cells belonging to a room are separated from other rooms by walls and narrow passages. The task is to find which narrow passages separate the rooms at doors and not only between obstacles in the room. For this purpose, we find a closed loop that passes through the boundaries of the room. In [19] closed loops are found by searching the occupied part of the environment. From each central coordinate of the narrow passages, the two nearest cells that indicate the occupied part of the environment are determined along the axis where a local minimum in the cell of the observed central coordinate is present.

Furthermore, one cell is declared as the start point and another as the end point. A search graph is created of the occupancy grid map cells with occupancy value of the occupied part of the environment, and of cells that form a junction which closes all other narrow passages. The shortest path from the start point to the end point in the created graph is then found using the FD* search algorithm. If the path exists, then it forms a closed loop that surrounds a room, and the observed central coordinate of the narrow passage is declared as a bridge node. Otherwise, the narrow passage lies inside the room and not at the border of the two rooms. In the case when a room is filled with obstacles that form a closed area, that area will be declared as a room (submap node), although in reality, it is not. But this case benefits hierarchical planning since it has been shown that the size of the submap node, i.e., the number of cells within one room, is closely related to the optimality and computational efficiency of the path planning algorithm [31]. Furthermore, the higher the parameter for the maximal width of the narrow passage is, i.e., the larger the safety cost mask, the more rooms will exist. The procedure of determining the bridge nodes is described in more details in [19].

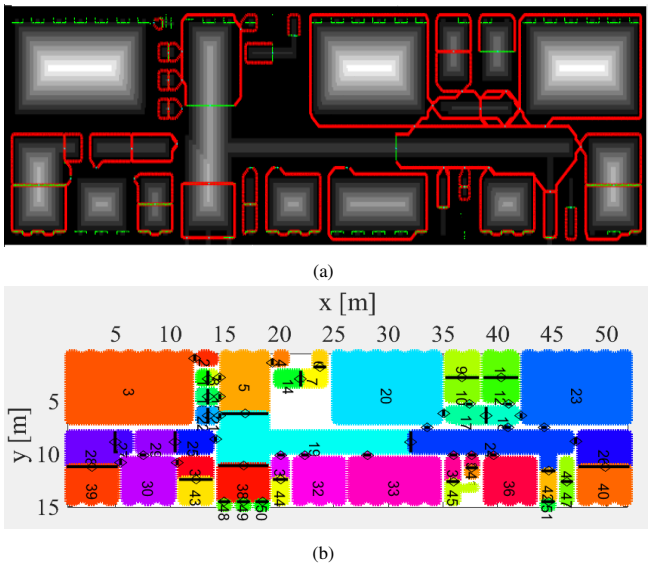


Figure 4: (a) Loops across the occupied part of the environment that surround the rooms (red) and narrow passages (green); (b) map of the floor with division to rooms: narrow passage (black line), bridge nodes (diamonds), and the rooms with their assigned numbers and colors (filled color).

The described method for finding bridge nodes is also the method for finding rooms. Rooms are rounded by the loop. Fig. 4(a) shows the found loop across the boundaries of the rooms and the narrow passages. Nodes inside the room represent one submap node at the higher level of the hierarchy. A room consists of a set of cells with the same associated number. Bridge nodes are assigned to the two rooms at whose boundaries they are located. Fig. 4(b) shows the narrow passage (black line), bridge nodes (diamonds), and the rooms with their assigned numbers and colors (filled color).

3.3. Creation of Higher Level Nodes

A graph at some level i consists of a set of nodes N_i and a set of edges A_i. Nodes from the set N_i are connected by edges from the set A_i. To create a graph at the higher level (level $i-1$), nodes that meet the conditions for being declared as marginal bridge nodes are selected from the set of nodes N_i. The selected marginal bridge nodes form a set of nodes N_{i-1} at level $i-1$. In order to connect nodes with edges at level $i-1$, a set of partial paths T_i at level i is found. These partial paths connect marginal bridge nodes of level i, and form edges between nodes at level $i-1$. At the most detailed level L_D, partial paths are found by the E* algorithm, and at all higher levels, partial paths are found by a simple A* graph search algorithm within the graph G_i. The union of edges between two marginal bridge nodes at level i forms a partial path and it becomes an edge of the graph at level $i-1$. The union is performed by simply merging edges that form the path from one marginal node to another.

An example on which the algorithm for autonomous hierarchy creation is described is the environment of FER buildings. The H-graph of FER has five hierarchical levels ($D = 4$). The most abstract level L_0 contains a single node. At level L_D is a graph created from the occupancy grid map. The occupancy grid map at level L_4 is shown in Fig. 5(d) along with the marginal bridge nodes at that level (black x sign). Bridge nodes are located at each passage between two rooms - submap nodes. At this level, the set of nodes N_4 consists of all grid map cells and the set of Cartesian coordinates C_4 contains the Cartesian coordinates of all grid map cells. Two nodes $i, j \in N_4$ from the graph G_4 are neighbors if $\| c_i - c_j \|_\infty = a_{cell}$, where a_{cell} is the cell size, c_i and c_j are corresponding Cartesian coordinates of cell centers, and $\| \cdot \|_\infty$ is the infinity norm. The set of edges A_4 consists of all grid map edges whose weights form the set W_4. The set of edge weights $W_4 = \{w_{i,j} \mid i, j \in N_4, i \text{ and } j \text{ are neighbors}\}$ is defined as the cost of transition between neighbors as

$$w_{i,j} := \| c_i - c_j \| \cdot \max\{z(c_i), z(c_j)\}, \tag{6}$$

where $z(c_i)$ and $z(c_j)$ are occupancy values of cells $i, j \in N_4$ within the safety cost mask depends on its distance from the obstacle. Finally, a set of partial paths T_4 consists of all paths determined between all marginal bridge nodes.

The marginal bridge nodes (all bridge nodes) from level L_4 are duplicated to the level above and they appear as nodes of the graph at level L_3. In that way, the bridge nodes connect levels L_3 and L_4. Bridge nodes in level L_3 are shown abstractly and they have inherited the coordinates from the level below. The bridge nodes from the most detailed level are stored in the structure with their corresponding (x, y, z) coordinates and used to create a graph on level L_3. The set of nodes N_3 consists of all marginal bridge nodes from the level below and the set of Cartesian coordinates C_4 contains the Cartesian coordinates of duplicated marginal bridge nodes. The nodes at this level are the parent nodes for nodes of both rooms between which they are located. The connections between nodes in the graph at level L_3 are partial paths determined between all marginal bridge nodes at the level below and form a set A_4 whose weights (path cost) are stored in the set W_3. A set of partial paths on

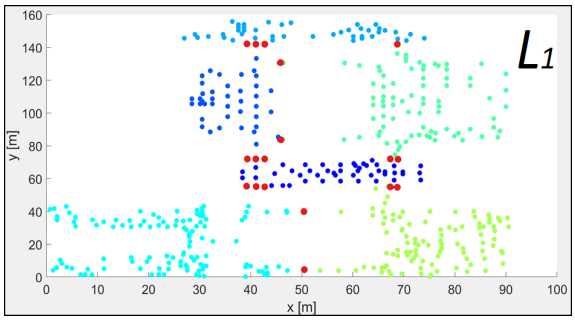
this level T_3 consists of all paths determined between marginal bridge nodes of this level. The marginal bridge nodes at level L_3 are located at places of elevators, at the ground floor intersections of two adjacent buildings or at the crossroads. The bridge nodes at the elevator locations play the role of vertical bridge nodes. Their determination is done automatically under three conditions. First, comparing the positions of the nodes on level L_3, second, the size of the rooms to which these nodes belong is taken into account, and third, the number of room doors is considered. For the H-graph of FER, the nodes of level L_3 are shown in Fig. 5(c), composed of regular (red dots) and marginal (black x sign) bridge nodes.

At the next level L_2, the graph consists of marginal bridge nodes from level L_3. Thus, the marginal bridge nodes from level L_3 with the corresponding (x, y, z) coordinates, are duplicated at level L_2, and used to create a graph on that level which contains a set of nodes N_2, a set of corresponding Cartesian coordinates C_2, set of edges A_2 (determined partial paths at the level below) with corresponding costs stored in the set W_2, and the set of partial paths T_2 again determined between marginal bridge nodes of this level. For the following example, the marginal bridge nodes at level L_2 are determined as the bridge nodes at the transition points from one building to another and they are located on the ground floor where the two buildings meet. The nodes colored red in Fig. 5(b) represent all bridge nodes at level L_2, while the marginal bridge nodes (three black x sign) are duplicated to a higher level where they form a new, more abstract graph. Blue dots are shown only for a reference to have a view of other floors in a building, and they represent bridge nodes from lower level L_3.

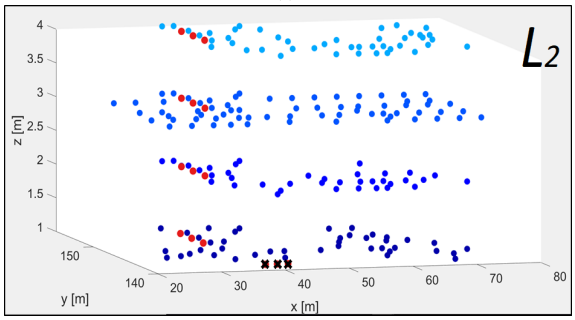
At level L_1, the graph is composed of marginal bridge nodes from level L_2 and form sets N_1 and C_1. Fig. 5(a) shows the nodes of level L_1 colored in red, while all other dots are the bridge nodes from level L_3 colored in different colors representing each building. We can notice that the bridge nodes of level L_1 are at connections of buildings and there are no more marginal bridge nodes but only regular bridge nodes at level L_1. At the level L_1, a set of edges A_1 consists of partial paths from the level L_2 and a set W_1 consists of corresponding weights. A set of partial paths T_1 is empty because there are no marginal bridge nodes at this level.

3.4. Finding Partial Paths

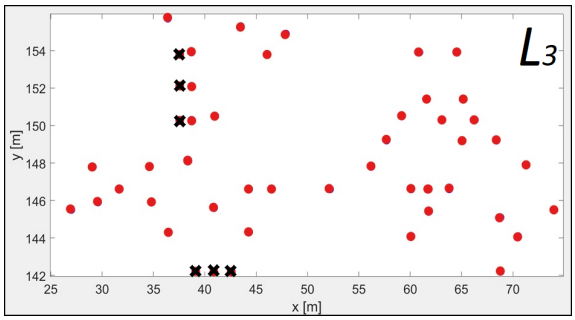
To complete the abstract graphs at higher levels, it is necessary to connect selected nodes with edges. Connecting nodes is also performed from the most detailed level (L_4). At the most detailed level, partial paths are determined within each room (submap) node between all possible pairs of marginal bridge nodes that belong to that room, shown in Fig. 6(a). Once the partial paths are determined at the most detailed level, they are the basis for computing the new partial paths at the higher level. The process of determining partial paths ends when the paths are determined in the highest level which contains marginal bridge nodes. Partial paths between the marginal bridge nodes at the most detailed level (L_4) are determined by an E* algorithm. The E* algorithm searches the graph from the goal node and finds all possible paths in the graph. For this purpose, we



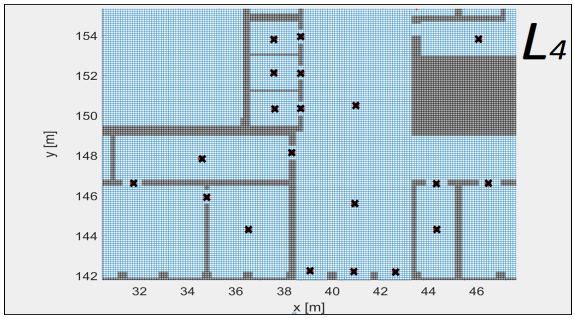
(a)



(b)



(c)



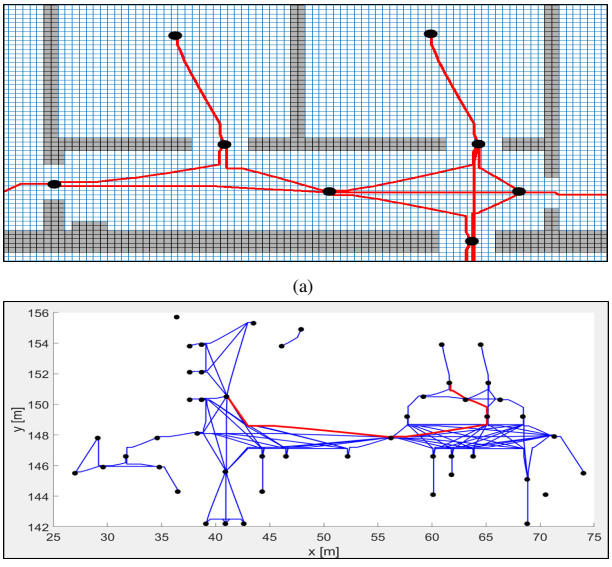
(d)

Figure 5: (a) The nodes of level L_1 are shown (red). Nodes in other colors are drawn for clarity to show the locations of different buildings. (b) The nodes of level L_2 (red) and marginal bridge nodes of level L_2 (black x sign) are shown. Blue nodes are drawn for clarity to represent the locations of different floors. (c) The nodes of level L_3 are shown, where the black x sign indicates marginal bridge nodes of that level. (d) A part of the grid map is shown. Grid map cells are nodes of level L_4 and the black x sign indicates marginal bridge nodes of level L_4.

single out an occupancy grid map with the safety cost mask of each room. If the selected room has more than one marginal bridge node, an algorithm performs a search of the entire room in order to find the paths between all marginal bridge nodes

within that room, storing the costs of the found partial paths. If a room has N bridge nodes, there must be found $\frac{N(N-1)}{2}$ partial paths within the room and the E* algorithm must be called $(N-1)$ times because one search connects the goal node with all other bridge nodes in the same room.

At the level above, L_3, the description of the environment becomes abstract and the environment is represented as a simple graph consisting of nodes and edges. The nodes of the graph are marginal bridge nodes from level L_4, and the edges of the graph are partial paths determined at level L_4. At all higher levels, after determining the marginal bridge nodes, the paths between two marginal bridge nodes are simply determined by the union of the corresponding partial paths from the level below, as shown in Fig. 6(b). In other words, a simple graph search using the A* algorithm finds the partial paths between all marginal bridge nodes by concatenating the partial paths from the level below. Partial paths from all levels and their costs are stored in memory.



(a)

(b)

Figure 6: Partial paths: (a) determined at the most detailed level within each room submap node between all possible pairs of marginal bridge nodes; (b) at level L_3 determined as the union of the partial paths from the level below.

4. Test Results

In this section, we analyze the proposed algorithm according to the reduction of nodes compared to the plain graph search, and partial paths quality compared to the ones obtained by the A* graph search. Also, we compare our approach for autonomous H-graph creation with the approaches based on a generalized Voronoi graph and watershed algorithm.

4.1. H-graph Creation and Analysis

The analysis and validation of the H-graph creation results were made for three cases: a multi-building environment which represents the buildings of FER, a large warehouse with a lot of shelves, and an outdoor urban environment which is represented by a city map. The algorithm creates a graph consisting of five abstract levels for all cases and can be easily extended to more if needed. This includes an extension of the algorithm where conditions must be set to determine which nodes at the next higher level should be duplicated. These nodes at a higher level are connected by edges that are partial paths from a lower level of the hierarchy. The number of nodes at each level is shown for all three cases in Tables 1, 2, and 3, respectively. Results of the proposed algorithm for autonomous H-graph creation are given in Tables in terms of the number of submap or end nodes K, the number of bridge nodes B_{N} and the number of marginal bridge nodes $B_{\mathrm{Nmarginal}}$.

A. Multi-building environment

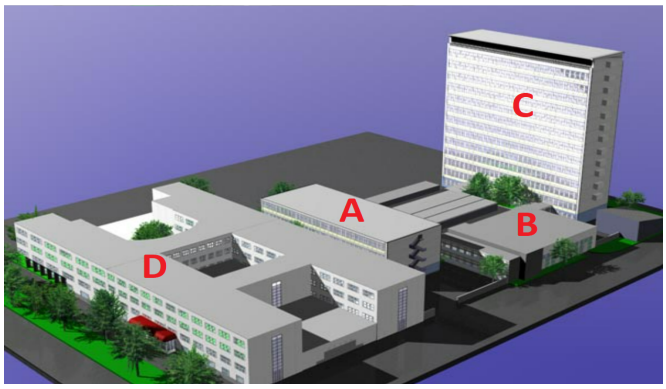


Figure 7: Four buildings of FER environment marked with letters: A, B, C, D.

FER has four buildings marked with letters: A, B, C, and D (see Fig. 7). The ground floor of all four buildings has dimensions 95 × 160 m and the whole multi-building environment lies on 43151 m^2 when the dimensions of all floors are taken into account. The floor plans of each floor in each building are obtained from the CAD program and individually conducted through the described algorithm for room segmentation and higher level nodes extraction (see Fig. 8). Each floor plan is the basis for creating all abstract levels using the previously described parameters: we selected a differential drive robot with a radius of 0.25 m, we selected the size of the cell 0.1 m to ensure passable doorways in the grid representation after enlargement of obstacles for the radius of the robot, and we selected the maximal width of the narrow passage as 6 m according to the average width of the real rooms to create more submap nodes in the hierarchy. The lowest level contains the occupancy grid map of all rooms, and the highest level contains only one node called FER. We already described our algorithm in Sec. 3 for this environment and therefore Fig. 5 shows the nodes of each level, and Fig. 6 shows the connections of nodes by partial paths. The floor plans of buildings B and D were divided into two parts due to their large dimensions and therefore considered as four buildings instead of two buildings. This case could be also solved by introducing an extra level into the overall organization of the graph. Modeling a hierarchical graph in this way only confirms the fact that the hierarchical abstraction of the

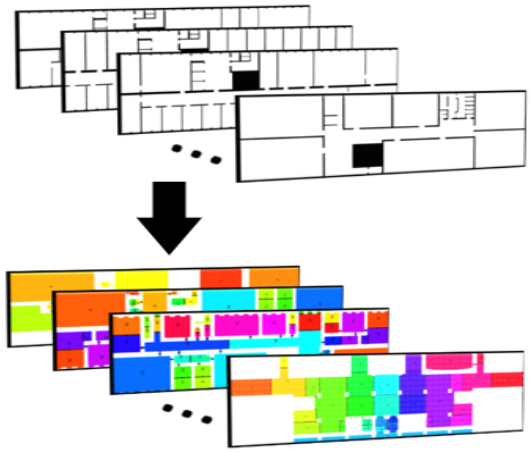


Figure 8: Floor plans of FER as input in algorithm and submap nodes after rooms segmentation.

environment is flexible and adaptable to different environments with many different features.

Results of the proposed algorithm for autonomous H-graph creation for a multi-building environment are given in Table 1. Expectingly, the number of nodes decreases at higher levels of the hierarchy. At the most detailed level, all bridge nodes are also marginal bridge nodes. Also, at each higher level, bridge nodes are marginal bridge nodes from a lower level. The created H-graph is shown in Fig. 9. The graph is completely connected by bridge nodes and the corresponding partial paths, so it is possible to plan the robot's path through the whole FER.

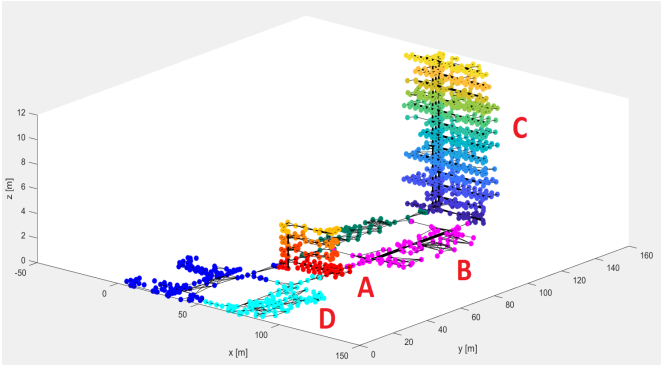


Figure 9: Representation of FER buildings as an H-graph.

Table 1: Review of nodes at each level for muti-building environment.

Level	*K*	B_{N}	$B_{\mathrm{N}marginal}$
L_0	1 environment	0	0
L_1	6 buildings	18	0
L_2	27 floors	92	18
L_3	1 924 rooms	1882	92
L_4	1 992 344 nodes	1882	1882

B. *Warehouse environment*

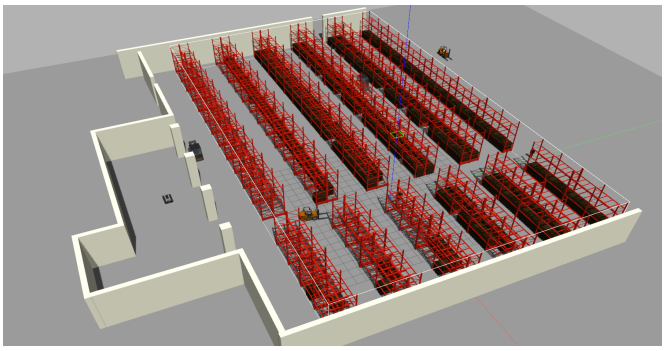


Figure 10: A warehouse environment with the shelves and dynamic obstacles

The warehouse environment shown in Fig. 10 has the dimensions 52 × 60 m and contains a lot of shelves. There are also dynamic obstacles in Fig.10 but only static obstacles were taken into the process of creating the H-graph. The map of the warehouse (see Fig. 11) is built using SLAM and gmapping [39]. Due to the large dimensions of the warehouse, the map of the floor plan is first divided into six equal parts as shown in Fig. 11 where red lines indicate the division of the warehouse into smaller parts. If the number of divided parts of the floor plans would be different, this will only affect the number of submap nodes at the levels L_1 and L_2, and the output of the algorithm would not change because all parts are merged in the same graph at the end. Each part of the map is individually conducted through the previously described algorithm as a picture in *png* format together with a corresponding text file in *yaml* format. From the floor plans, an occupancy grid map with the safety cost mask is generated using the previously described parameters: we selected a differential drive robot with a radius of 0.25 m, we selected the size of the cell 0.05 m to ensure passable doorways in the grid representation after enlargement of obstacles for the radius of the robot, and we selected the maximal width of the narrow passage as 8 m according to the average width of the corridor between the warehouse shelves to create more submap nodes in the H-graph. The environment is further segmented into rooms and narrow passages, while the bridge nodes are selected taking into account the assigned parameters of the algorithm (see Fig. 12).

In Fig. 12 is shown that this resolution of the grid map contributes to the detection of all narrow passages between the shelves. The grid map resolution is inversely proportional to cell size. If the narrowest passage in the environment has a width d, and the robot radius r is smaller than $\frac{d}{2}$ (otherwise there is no solution for passage detection), then the following condition must be satisfied to detect narrow passages: $a_{cell}\lfloor 2\frac{r}{a_{cell}} \rfloor < d$, where a_{cell} is the cell size.

By extracting bridge nodes and connecting them with partial paths, graphs of higher levels of abstraction were created (see Fig. 13). In Fig. 13, it can be seen that the number of nodes and partial paths decreases at each higher level. The lowest level of abstraction contains occupancy grid maps of 253 separated rooms, and the highest level contains only one node called

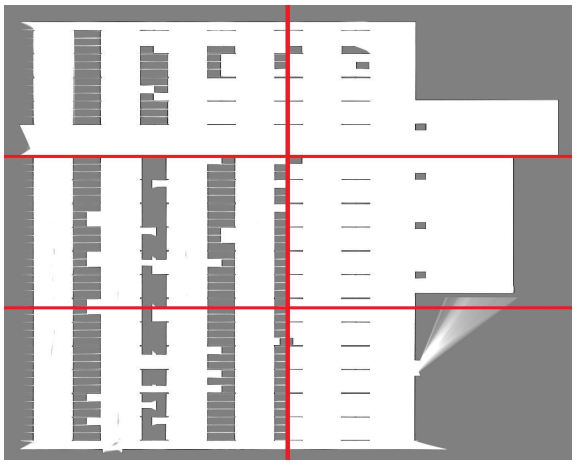


Figure 11: The map of the warehouse which is built using SLAM and gmapping. Red lines indicate the division of the map into six equal parts.

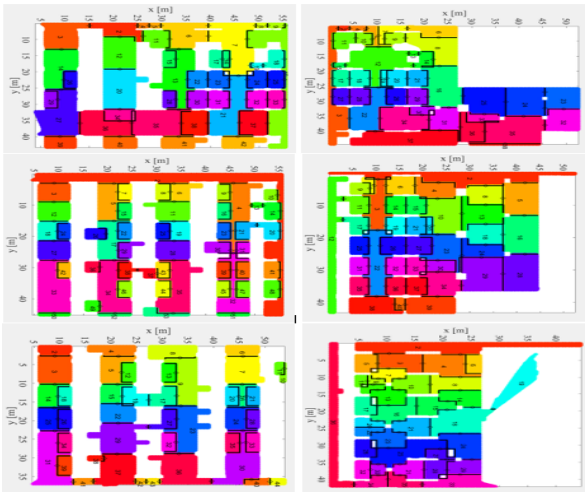


Figure 12: Segmented rooms as submap nodes obtained after room segmentation of the warehouse maps: narrow passage (black line), bridge nodes (diamonds), and the rooms with their assigned numbers and colors (filled color).

WAREHOUSE. The number of nodes in the rooms later significantly affects the execution time of the path planning and replanning algorithm. On average, one room contains 2961 nodes where nodes are the cells of the grid map. The maximum number of nodes within one selected room is 17358 nodes. The number of nodes in the six largest rooms are: 17358, 10213, 9470, 9203, 9130, and 9015. The number of nodes in all other rooms is less than 9000 nodes.

Table 2: Review of nodes at each level for a warehouse environment.

Level	*K*	B_{N}	$B_{\mathrm{N}marginal}$
L_0	1 warehouse	0	0
L_1	6 parts of warehouse	36	0
L_2	6 floors	72	36
L_3	253 rooms	300	72
L_4	748 900 nodes	300	300

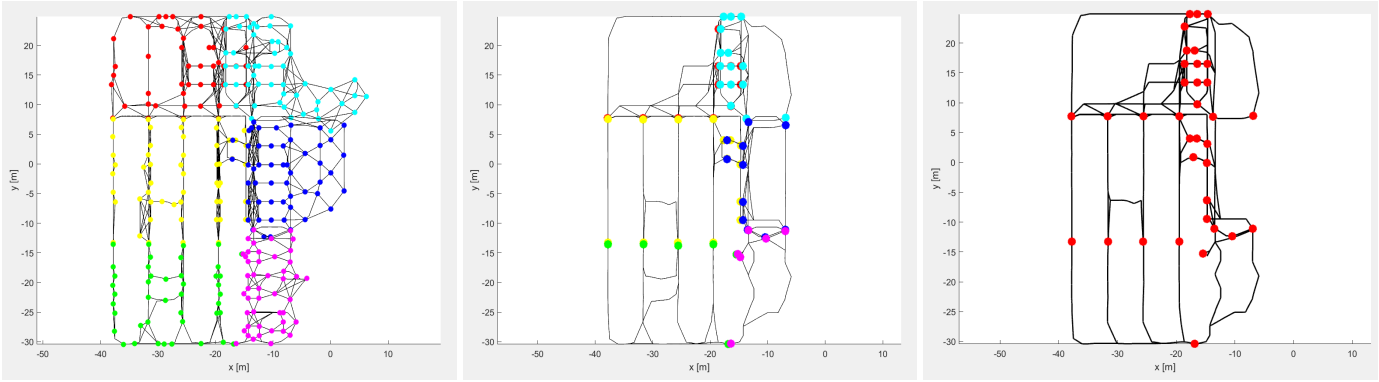
Results of the autonomous H-graph creation for the warehouse environment are given in Table 2. In this case, the number of nodes also decreases at higher levels of the hierarchy. As in the previous case, at the most detailed level, all bridge nodes are also marginal bridge nodes and at each higher level, bridge nodes are marginal bridge nodes from a lower level. In Fig. 13 it can be seen that the graph is completely connected by bridge nodes and the corresponding partial paths, which allows faster path planning of the mobile robots to perform their tasks through the warehouse. Due to the high resolution and detected narrow passages, the robot has greater freedom of movement through the workspace.

C. *Urban environment*



Figure 14: The streets of Zagreb city as an outdoor urban environment.

The urban environment shown in Fig. 14 is represented by a black and white street map of one part of Zagreb city. The map is obtained using Quantum Geographic Information System (QGIS) and has dimensions 2400 × 1400 m. Streets, parking, and places where vehicles are free to drive are shown in white color, and everything else is colored black on the map. Due to the large dimensions of the city map, the map is first divided into twelve equal smaller parts, similar to what was done in the case of the warehouse environment. The number of divided parts could be different, but this would only affect the number of submap nodes at the levels L_1 and L_2. The output of the algorithm would not change because all parts are merged in the same graph at the end. Each part of the map is individually conducted through the previously described algorithm as a picture in *png* format together with a corresponding text file in *yaml* format. From these parts of the map, an occupancy grid map with the safety cost mask is generated using the previously described parameters. For the urban environment: we selected a differential drive robot with a radius of 0.25 m, we selected the size of the cell 1 m, and we selected the maximal width of the narrow passage as 10 m according to the average width of the street. The segmentation algorithm further divides the streets of the city into virtual rooms and the positions of narrow passages and bridge nodes are selected taking into account the assigned parameters of the algorithm (see Fig. 15). With the goal of determining narrow passages on the previously created occupancy grid map with the safety cost mask, the algorithm searches for the minimum and maximum values of the occupancy function z using its first and second derivatives (as described in [19]). In urban environments, an entire group of



(a) (b) (c)

Figure 13: Higher levels nodes and partial paths of warehouse environment: (a) level L_3; (b) level L_2; (c) level L_1. For clarity, all nodes created from the same map parts are colored in the same color.

cells representing one street on the cost map may be detected as a narrow passage (based on the minimum and maximum values of the cell occupancy function z), but only the central coordinates of the detected group of cells become candidate for bridge nodes. Thus, if a whole street is recognized as a narrow passage, the bridge node is determined in the center of that street, since it is the central cell of the narrow passage. Fig. 15 shows the successful segmentation of streets into smaller parts, which could greatly reduce the search time of path planning in the urban environment. By extracting bridge nodes and connecting them with partial paths, graphs of higher levels of abstraction were created (see Fig. 16). Due to the large dimensions of the environment, level L_3 has many nodes. In Fig. 16, it can be observed that the number of nodes and partial paths decreases at each higher level.

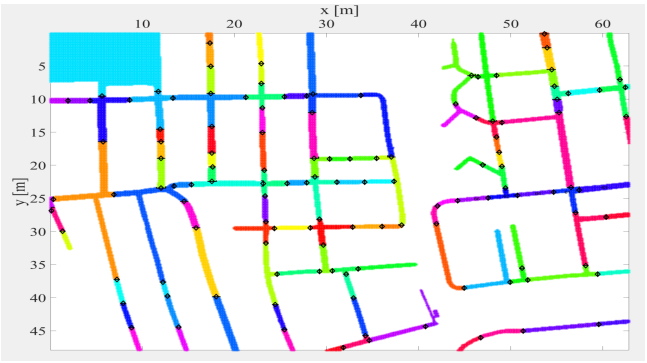


Figure 15: Urban environment after rooms segmentation and narrow passages and bridge nodes determination. Virtual rooms are submap nodes filled with different colors, narrow passages (black lines), and bridge nodes (diamonds).

Analysis of the created H-graph for the urban environment is given in Table 3. Table 3 shows that the number of nodes also decreases at higher levels of the hierarchy. The lowest level of abstraction contains occupancy grid maps of 1144 separated rooms, and the highest level contains only one node called CITY. On average, one room (submap node) contains 440 nodes where nodes are cells of the grid map. Also, at the most detailed level, all bridge nodes are also marginal bridge nodes and at

Table 3: Review of nodes at each level for the urban environment.

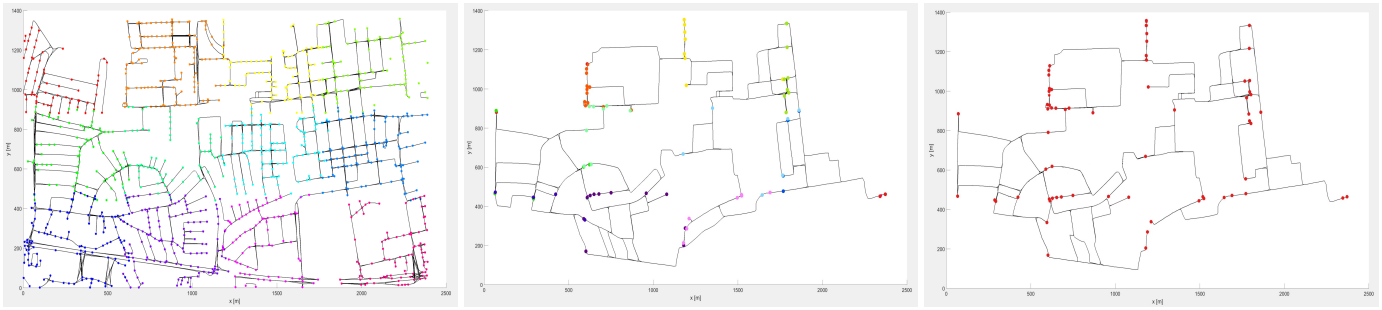
Level	*K*	B_{N}	$B_{\mathrm{N}marginal}$
L_0	1 city	0	0
L_1	12 parts of city	70	0
L_2	12 floors	142	70
L_3	1 144 rooms	1 123	142
L_4	503 211 nodes	1 123	1 123

each higher level, bridge nodes are marginal bridge nodes from a lower level. In Fig. 16 it can be seen that the graph is completely connected by bridge nodes and the corresponding partial paths, which allows better path planning of the autonomous vehicles to move and perform tasks through the urban environment. Compared to a multi-building environment and warehouse, the H-graph of the urban environment has fewer nodes than the H-graph of a multi-building environment because the city is considered just as a ground floor and has more nodes than the H-graph of the warehouse environment because the city has larger dimensions.

4.2. H-graph and Plain Graph Paths

We tested the proposed hierarchy on the path planning examples in all three cases of environment. Each environment is represented in two ways: with a plain graph as an occupancy grid map with the safety cost mask and with a created H-graph. For each environment example, four paths were found within the H-graph and within the plain graph. A plain graph as an occupancy grid map with the safety cost mask is generated using the same parameters that are used during the segmentation of the hierarchical most detailed level. Plain graph grid map resolution and resolution of the most detailed level in H-graph are equal in each case of path planning, but the resolution is different for each environment, as it was described previously.

The results of the comparison are shown in Table 4 where N is the number of searched nodes, c is the cost of the path, l is the length of the path, and t_{exec} is the execution time of the graph search algorithm. The optimal path in the occupancy grid map with a safety cost mask is the one with the lowest path cost.

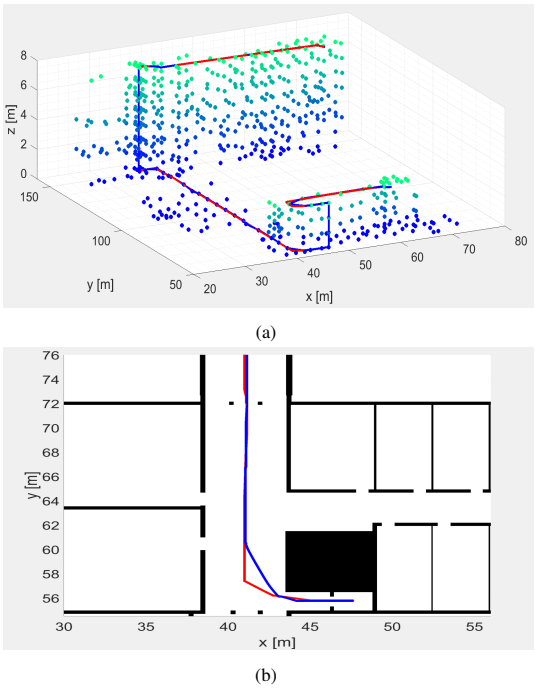


(a) (b) (c)

Figure 16: Higher levels nodes and partial paths of the urban environment: (a) level L_3; (b) level L_2; (c) level L_1. For clarity, all nodes created from the same map parts are colored in the same color.

The safety cost mask of the occupancy grid map moves the path away from obstacles and thus increases the length of the path. A shorter length path has a higher cost because it passes across more expensive cells in the occupancy grid map. The cost of the path is expressed in the number of weighted cells. Start and goal points for each path are given with $(x, y, floor)$, where (x, y) are the Cartesian coordinates of the start/goal point in the environment in relation to the user-defined coordinate system, and the value $floor$ indicates the floor on which the start/goal point is located.

The first case is a multi-building environment with a resolution of 100 cells per square meter. To create a plain graph as an occupancy grid map with the safety cost mask, we selected a differential drive robot with a radius of 0.25 m and the maximal width of the narrow passage as 6 m. The start and the goal points of the paths are selected on different floors and buildings. For the path planning within the H-graph, the rooms where the start and goal are located are searched by the E* algorithm to find the path to the bridge nodes (doors). The rest of the path is found by searching the marginal bridge nodes at each level of the hierarchy. On the other hand, path planning within the plain graph searches all nodes on each floor where the path passes also by the E* algorithm. The path comparison for the first planned path is shown in Fig. 17. The results in Table 4 show that a significantly lower number of nodes need to be searched using the H-graph compared to the plain graph (depending on the size of the segmented rooms). Searching fewer nodes significantly saves computational time. Path planning within a plain graph leads to a longer path compared to the H-graph and it is found that the paths are approximately equal in length. On the other hand, the plain graph search leads to a cheaper path cost. Furthermore, these results prove that the position of the bridge nodes is indeed at the optimal place in the environment because the path obtained by the H-graph search still has an approximately equal length with respect to the given value of the safety cost mask (path cost). The path obtained by searching the plain graph and the path obtained by searching the H-graph meet at the points where the bridge nodes are placed. Thus, the minimum cost of the path, i.e. the criterion of optimality, is directly achieved for a path obtained by searching the plain graph, and it is indirectly achieved by the position of the bridge nodes when searching the H-graph. The criterion of optimality was taken into account when determining the position of the bridge nodes in the H-graph, and it was inherited in all higher levels of the hierarchy. Moreover, the positions of the bridge nodes ensure the safety of the path.



(a)

(b)

Figure 17: Path found within the H-graph of FER (blue) and within the plain graph (red): (a) set in the context of the H-graph; (b) zoomed in order to detect deviations.

The second case is a warehouse environment with a resolution of 400 cells per square meter. To create a plain graph as an occupancy grid map with the safety cost mask, we selected a differential drive robot with a radius of 0.25 m and the maximal width of the narrow passage as 8 m. Path planning for both graph cases (plain graph and H-graph) is realized in the same way as for the multi-building environment by searching all nodes in the plain graph and nodes at all levels in the H-graph. For this case of the environment, the plain graph search also leads to a longer path compared to the H-graph search and obtained paths are approximately equal in length while the

Table 4: Comparison of path planning in H-graph and plain graph.

Environment	*Path*	*Start* [m]	*Goal* [m]	*Graph*	*N* [1]	*c* [1]	*l* [m]	t_{exec} [ms]
Multi-building	Path 1	(61.2, 56.6, 3)	(71.4, 142.3, 7)	H-graph	5 497	15 974	189.46	51
				Plain graph	302 171	15 531	191.87	2720
	Path 2	(33.3, 86.6, 0)	(35.5, 142.7, 10)	H-graph	2 019	13 925	95.55	20
				Plain graph	171 297	13 634	96.08	1542
	Path 3	(61.1, 153.2, 0)	(63.3, 64.5, 1)	H-graph	1 640	21 916	155.13	15
				Plain graph	253 141	21 675	157.62	2278
	Path 4	(54.1, 68.6, 0)	(63.3, 65.6, 1)	H-graph	1 925	15 169	64.21	19
				Plain graph	131 497	14 934	64.42	1183
Warehouse	Path 1	(-36.5, -28.0, 0)	(5.0, 10.0, 0)	H-graph	6 015	9 552	79.81	55
				Plain graph	611 284	9 436	80.42	5501
	Path 2	(-29.0, -30.0, 0)	(-31.5, 24.5, 0)	H-graph	3 707	7 749	67.64	35
				Plain graph	611 284	7 658	68.42	5501
	Path 3	(-19.0, -21.5, 0)	(-17.1, 13.7, 0)	H-graph	3 418	3 989	37.11	32
				Plain graph	611 284	3 979	37.81	5501
	Path 4	(-2.1, -4.7, 0)	(-14.5, -3.2, 0)	H-graph	5 469	1 261	12.81	50
				Plain graph	611 284	1 225	12.91	5501
Urban	Path 1	(328, 995, 0)	(1070, 175, 0)	H-graph	1 022	15 006	1933.27	11
				Plain graph	454 265	14 919	1935.10	4089
	Path 2	(850, 1271, 0)	(1200, 148, 0)	H-graph	557	9 193	1364.15	6
				Plain graph	454 265	9 053	1365.91	4089
	Path 3	(47, 403, 0)	(1063, 688, 0)	H-graph	1 870	9 968	1229.15	16
				Plain graph	454 265	9 633	1245.02	4089
	Path 4	(265, 380, 0)	(746, 795, 0)	H-graph	658	6 403	780.54	7
				Plain graph	454 265	6 112	782.00	4089

plain graph search leads to a cheaper path cost. In this case, the advantage of path planning within the H-graph is more pronounced because the resolution of the environment is higher, so for path planning within a plain graph, it is necessary to search four times more nodes than it would be needed in the case of two times lower resolution which results in four times shorter search time.

The third case is an urban environment with a resolution of 1 cell per square meter. To create a plain graph as an occupancy grid map with the safety cost mask, we selected a differential drive robot with a radius of 0.25 m and the maximal width of the narrow passage as 10 m. Results of paths obtained by searching the plain graph and H-graph show a significant reduction in the number of searched nodes when using the H-graph. The results in Table 4 show that, the length of the path obtained by searching the plain graph is longer than that obtained by searching the H-graph and it was found that the paths are also approximately equal in length. The path cost is lower when searching a plain graph. In this case, the resolution of the graph is lower compared to the previous two cases, but the dimensions of the urban environment are large, so if we look at the plain graph, the number of searched nodes is still large. Also, the size of the segmented rooms (submap nodes) contributes to the number of searched nodes in the H-graph. By searching two rooms at the most detailed level of the H-graph, the paths from the start to the bridge node of the first room and from the goal to the bridge node of the second room are found. If the segmented rooms in the H-graph are smaller, a smaller number of nodes should be searched to find these two parts of the path which results in a significantly shorter search time. On average, one room (submap node) contains 2961 nodes in the warehouse environment and 440 nodes in the urban environment. It can be concluded that the size of the submap nodes greatly affects the number of searched nodes in path planning within the H-graph.

4.3. Comparison of Partial Paths by E and A**

Pre-calculated partial paths that connect marginal bridge nodes at a given level are near-optimal paths that form the skeleton of the H-graph. The search for the global path of the robot is based on the search for the H-graph and the union of the partial paths. Therefore, it is important to pay attention to the algorithm used to create the partial paths. We compared the partial paths creation by the E* algorithm to the standard A* algorithm according to the following path characteristics: the length of the path l, the number of points in which the path changes direction $n\alpha$, the sum of all direction change angles $\sum \alpha$ and the number of points $n\alpha_{\geq 45^\circ}$ in which the change of direction is greater than or equal to 45 °. Algorithms E* and A* are executed at the occupancy grid map of one room. The start is chosen within the room, and the goal is set at the door of the room. The numerical results for one example of initial planning are shown in Table 5 and the path comparison is shown in Fig. 18. The E* algorithm outperforms the A* algorithm in path length and direction changes gives an approximation of the shortest (Euclidean) distance path with turning angles not limited to 45 ° and the path has minor direction changes at the same point. Unlike A*, E* does not impose restrictions on turning angles, allowing for more flexible path planning. On average, for all possible found paths from each node in the room shown in Fig. 18 to the door (number of paths is 4774), A* has about 3 % higher path length with the standard deviation 2.39 % and the path has significant directional changes at several path points.

Table 5: Comparison of E* and A* algorithm in a grid map of the room.

Algorithm	*l* [m]	$n\alpha$	$\sum \alpha$ [°]	$n\alpha_{\geq 45^\circ}$
E*	7.2504	4	93.74	1
A*	7.6569	7	315.00	7

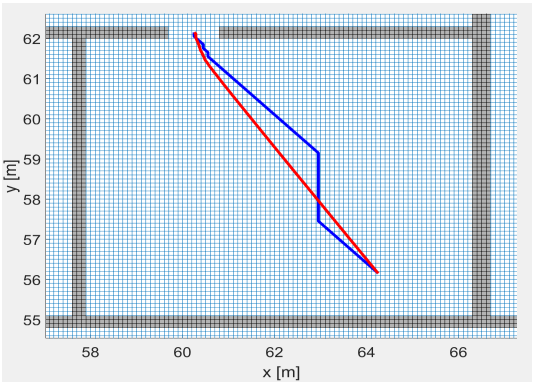


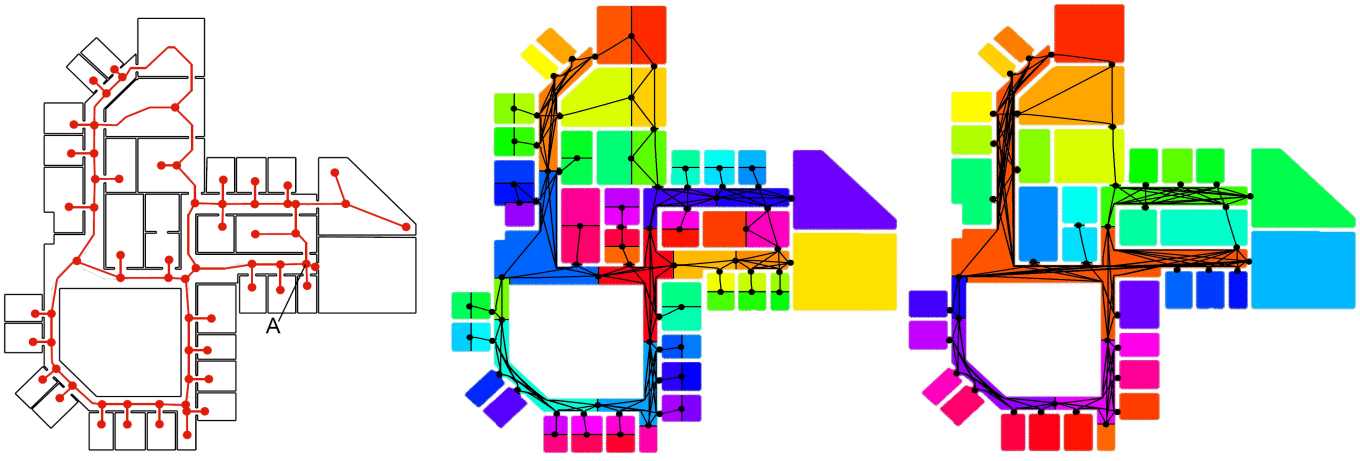
Figure 18: Path found in the occupancy grid map with E* algorithm (red) and A* algorithm (blue).

4.4. Comparison with Voronoi-based Hierarchy Creation

Many studies for the hierarchical abstraction of the environment are based on the environment segmentation using various metrics by reference to a generalized Voronoi graph [20, 21, 22, 23, 24, 25]. We compared our approach for autonomous H-graph creation with the approach given in [24]. This approach creates a Hierarchical Voronoi-based Route Graph using a Voronoi diagram to divide the environment into regions based on the distance to a set of landmarks. A generalized Voronoi graph (GVG) has the vertices placed at the positions of the corresponding meet points of the created Voronoi diagram of a 2D environment. We compare the approaches on two examples of environments on which the algorithm in [24] was tested. For clarity of comparison, we also create the H-graph at two levels of abstraction.

In the first case, the approaches are compared on the ground floor plan in the MZH building of the University of Bremen shown in Fig. 19. To create the H-graph from the floor plan, an occupancy grid map with the safety cost mask is generated using the parameters: radius of robot equals to 0.25 m, the size of the cell 0.05 m, and we varied the maximal width of the narrow passage with values: 3 m and 1 m to create H-graph with less and more submap nodes by grid map segmentation. The hierarchical Voronoi-based Route Graph shown in Fig. 19(a) contains 67 nodes at the higher level that are not strictly placed in the middle of narrow passages between rooms of the environment. The disadvantage of this approach is that the higher level nodes are located either in the middle of the room or in the middle of the corridor, and not at the narrow passages between the rooms, so the robot path from one room to another would have to pass through the middle of the room every time. Also, edges between nodes at the higher level pass exactly across the middle of rooms, corridors, and narrow passages which often does not ensure the shortest possible path. On the other hand, the H-graph that is created using our approach at the higher level contains 74 nodes for the value of the maximal narrow passage width equals to 4 m (shown in Fig. 19(b)) and 40 nodes at the higher level for the value of the maximal narrow passage width equals to 1 m (shown in Fig. 19(c)). Nodes of the higher level are placed in the middle of detected narrow passages - in transition points between detected rooms and in the middle of long corridors. It can be seen that the number of higher level nodes, detected narrow passages and submap nodes depends on the set value of the maximal width of the narrow passage. Depending on this parameter, our approach can generate bridge nodes in the middle of rooms and corridors as in the case of the Hierarchical Voronoi-based Route Graph. This parameter can be adjusted depending on the desired size of the detected rooms, which is important during the path planning search. Edges between nodes at the higher level are the optimal paths between nodes that are found using the E* algorithm at the lower level. Therefore, path safety is ensured by finding higher level edges on the occupancy grid map with a safety cost mask, and not necessarily by passing through the middle of the environment.

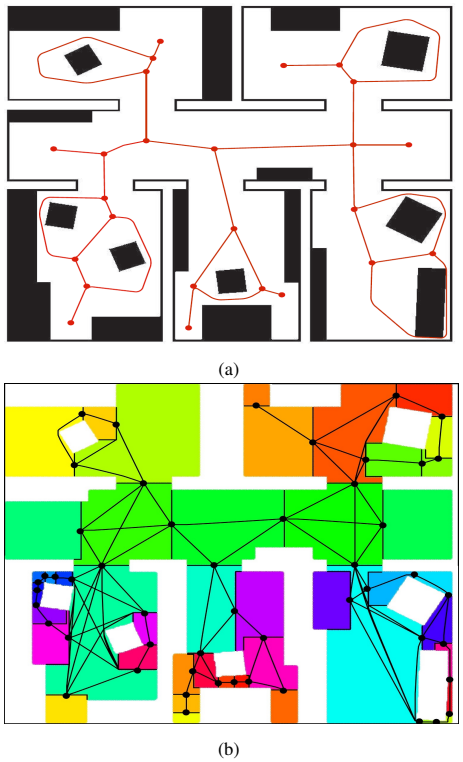
In the second case, the approaches were compared on the enclosed environment filled with static obstacles shown in Fig.



(a) (b) (c)

Figure 19: Comparison of hierarchical graphs created on the ground floor plan in the MZH building of the University of Bremen: (a) Hierarchical Voronoi-based Route Graph from [24]; (b) H-graph created using our approach with the value of the maximal narrow passage width 4 m; (c) H-graph created using our approach with the value of the maximal narrow passage width 1 m.

20. To create the H-graph from the floor plan, an occupancy grid map with the safety cost mask is generated using the parameters: we selected a robot with a radius of 0.25 m, we selected the size of the cell 0.05 m to ensure passable doorways and in the grid representation after enlargement of obstacles for the radius of the robot, and we select the maximal width of the narrow passage 3 m. Hierarchical Voronoi-based Route Graph shown in Fig. 20(a) contains 25 nodes at the higher level and H-graph created using our approach shown in Fig. 20(b) contains 51 nodes. It is found that our approach extracts more nodes at a higher level and finds narrow passages even between static obstacles in the environment while the Hierarchical Voronoi-based Route Graph does not extract nodes in narrow passages between static obstacles which affects the bad characteristics of the higher level edges. In our approach, edges between nodes at a higher level still retain the property of near-optimal paths.



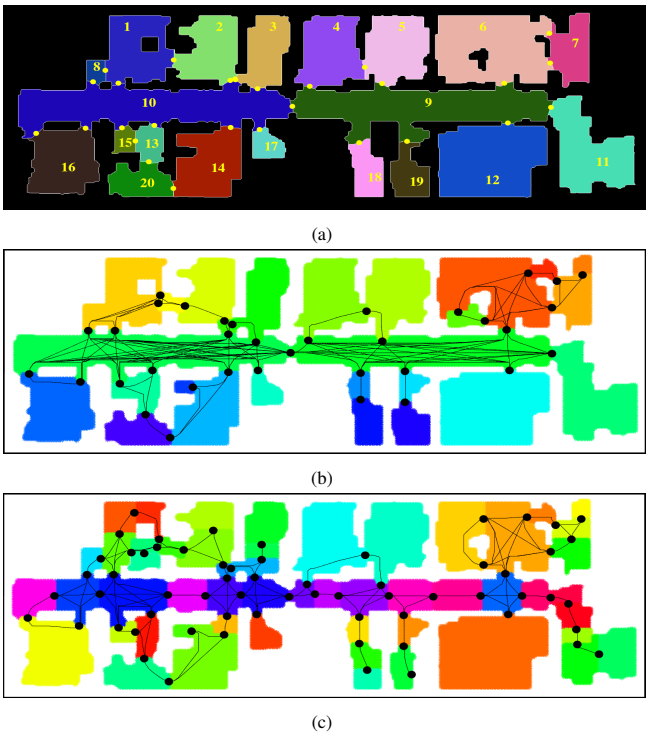
(a)

(b)

Figure 20: Comparison of hierarchical graphs created for the enclosed environment filled with static obstacles: (a) Hierarchical Voronoi-based Route Graph from [24]; (b) H-graph created using our approach

4.5. Comparison with Watershed-based Hierarchy Creation

A recent method [18] for creating a two-layer hierarchical graph uses a combination of the watershed segmentation algorithm and the Skeletonization-Informed Rapidly Exploring Random Tree* (SIRRT*) algorithm. In this method, a navigable graph of the environment is created consisting of local grid map segments, the adjacency matrix, and the junction nodes between all adjacent segments. The grid map image is first segmented using a watershed segmentation algorithm. Then, junction nodes are defined that lie on edge grid cells separating adjacent segments. The authors use the SIRRT* to compute intra-regional paths between the junction nodes of each grid map segment and construct the skeleton of the H-graph. We compare our approach and the approach proposed in [18] using the environment example given in [18], which consists of the ground floor plan shown in Fig. 21. To illustrate the comparison, we also construct the H-graph at two levels of abstraction. To create the H-graph from the floor plan, we generate an occupancy grid map with the safety cost mask, using the following parameters. We chose a differential drive robot with a radius of 0.2 m, we set the size of the cell to 0.1 m, and we varied the maximum width of the narrow passage with values of 2 m and 4 m to create an H-graph with fewer and more submap nodes by segmenting the grid. The H-graph created by the method proposed in [18] is shown in Fig. 21(a). The environment was divided into 20 local segments, and 28 junction nodes were registered at the higher abstraction level, while our approach extracted 35 nodes (Fig. 21(b)) and 66 nodes (Fig. 21(c)) for the maximum passage width values of 2 m and 4 m, respectively.



(a)

(b)

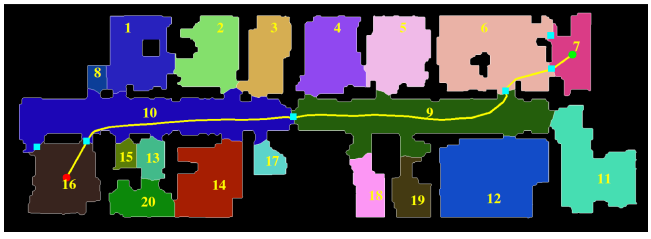
(c)

Figure 21: Comparison of hierarchical graphs created on the ground floor plan from [18]: (a) Hierarchical graph created using approach from [18]; (b) H-graph created using our approach with the value of the maximal narrow passage width 2 m; (c) H-graph created using our approach with the value of the maximal narrow passage width 4 m.

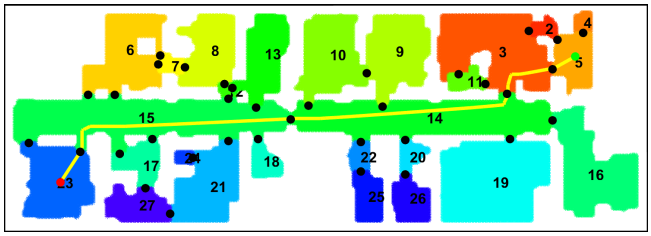
A comparison of these methods for creating H-graphs shows similarities in segmenting the environment into smaller parts. This is most obvious in the position of the nodes at a higher level of the hierarchy, where the nodes are located exactly at the transition points between two segmented rooms, as can be seen in Fig. 21(a) and Fig. 21(b). However, the major advantage of our approach is the adaptability of the size of the segmented rooms and the number of nodes at higher level using the maximum width of the narrow passage parameter shown in

Fig. 21(c). More segmented rooms result in fewer nodes in each room and searching smaller segmented rooms speeds up the path planning algorithm. Moreover, our approach generates an H-graph on more than two levels, which is not the case with the approach proposed in [18].

The main difference between these two approaches is the creation of partial paths between bridge or junction nodes. In [18], the authors use the SIRRT* algorithm to compute the intra-regional paths between the junction nodes of each grid map segment, while we use the E* algorithm. The quality of partial paths is most noticeable when searching for a global path through the environment in the H-graph. Fig. 22 shows a comparison of a global path through the environment, where (a) shows the path obtained by searching the H-graph from [18], and (b) shows the path obtained by searching the H-graph created by our method with the narrow passage 2 m value. To find a path similar to the one in [18], we chose the start at the point (6.5, 5.7) m and the goal at the point (58.6, 18.5) m, taking into account that the exact coordinates of the start and goal are not given in [18]. The cost of the path obtained from our H-graph is 2631, which is not comparable to the cost of the path from [18] since the cost maps are not obtained in the same way. In our H-graph, 5817 nodes were searched with a search time of 53 ms, which is an improvement over the shortest graph search time in [18], which is 0.28 s for fifty optimization iterations.



(a)



(b)

Figure 22: Comparison of global paths obtained by searching H-graphs: (a) created using approach from [18]; (b) created using our approach.

5. Conclusion and Future Work

We presented the autonomous hierarchy creation for path planning of mobile robots in large environments. In the presentation of our work, we create an H-graph with five hierarchy levels, but the proposed method creates a hierarchy with fewer levels if the conditions for creating all five levels of the hierarchy are not met (elevators or passages between buildings are not found). Also, the algorithm can be easily extended to create a hierarchy of more than five levels, which would allow the connection of even two different H-graphs. We demonstrate the necessary conditions of optimality during the creation of the H-graph. The tests and analysis of the algorithm on three cases demonstrated the successful creation of the H-graph with three different resolutions of the occupancy grid maps. Created H-graph contains natural close-to-optimal partial paths between all nodes at the same level which was demonstrated by the comparison of A* and E* algorithms. We have shown that the path planning algorithm in the H-graph explored 50 to 800 times fewer nodes than in the classical non-hierarchical graph which significantly reduces graph search time. Also, the paths found by hierarchical and non-hierarchical planning have approximately equal lengths. Furthermore, testing shows that our algorithm supports modular graph upgrades when it is needed. We compared our approach for autonomous H-graph creation with Voronoi-based and Watershed-based hierarchy creation and showed more advantages of the optimal placement of bridge nodes. Future work will focus on fast path planning and replanning problems in large, multi-building, and multi-robot environments using created H-graph. This problem is especially challenging since finding the best and collision-free paths is affected by many dynamically changing environments due to the movement of the robots and obstacles in the environment.

Acknowledgement

This work has been supported by the European Regional Development Fund under the grant KK.01.2.1.02.0119 – Research and development of an advanced unit for autonomous control of mobile vehicles in logistics (A-UNIT).

References

[1] S. Thrun, W. Burgard and D. Fox, "Probabilistic robotics," Cambridge, Massachusetts: MIT Press, 2005.

[2] H. Moravec and A. E. Elfes, "High resolution maps from wide angle sonar," Proceedings. 1985 IEEE International Conference on Robotics and Automation. Silver Spring, MO: IEEE Computer Society Press, pp. 116–121, 1984. DOI:10.1109/ROBOT.1985.1087316. S2CID 41852334

[3] C. Paul Bonnington and Charles H.C. Little, "The Foundations of Topological Graph Theory," Springer-Verlag, New York, 1995.

[4] S. Sam Ge, Q. Zhang, A. Thomas Abraham, B. Rebsamen, "Simultaneous Path Planning and Topological Mapping (SP2ATM) for environment exploration and goal oriented navigation," Robotics and Autonomous Systems, vol. 59, 2011. DOI: 10.1016/j.robot.2010.12.003

[5] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths." IEEE Transactions on Systems Science and Cybernetics, vol. 4, pp. 100–107, 1968. DOI: 10.1109/TSSC.1968.300136

[6] A. Stentz, "Optimal and efficient path planning for partially-known environments," Proceedings. 1994 IEEE International Conference on Robotics and Automation, vol. 4, pp. 3310–3317, 1994. DOI: 10.1109/ROBOT.1994.351061

[7] M. Đakulović and I. Petrović, "Two-way D* algorithm for path planning and replanning." Robotics and Autonomous Systems, vol. 59, pp. 329–342, 2011. DOI: 10.1016/j.robot.2011.02.007

[8] R. Philippsen and R. Siegwart, "An interpolated dynamic navigation function," Proceedings of the 2005 IEEE International Conference on Robotics and Automation, vol. 4, pp. 3782–3789, 2005. DOI: 10.1109/ROBOT.2005.1570697

[9] R. Bogue, "Growth in e-commerce boosts innovation in the warehouse robot market," Industrial Robot: An International Journal, vol. 43, no. 6, pp. 583-587, Oct. 2016. DOI:10.1108/IR-07-2016-0194

[10] N. Vimal Kumar, C. Selva Kumar, "Development of collision free path planning algorithm for warehouse mobile robot," Procedia Computer Science, vol. 133, pp. 456-463, 2018. DOI:10.1016/j.procs.2018.07.056

[11] A. Bolu, Ö. Korçak, "Path Planning for Multiple Mobile Robots in Smart Warehouse," 2019 7th International Conference on Control, Mechatronics and Automation (ICCMA), pp. 144-150, 2019. DOI: 10.1109/ICCMA46720.2019.8988635

[12] M. Takahashi, T. Suzuki, H. Shitamoto, T. Moriguchi and K. Yoshida, "Developing a mobile robot for transport applications in the hospital domain," Robotics and Autonomous Systems, vol. 58(7), pp. 889–899. 2010. DOI:10.1016/j.robot.2010.03.010

[13] A. G. Ozkil, Z. Fan, S. Dawids, H. Aanes, J. K. Kristensen and K. H. Christensen, "Service robots for hospitals: A case study of transportation tasks in a hospital," 2009 IEEE International Conference on Automation and Logistics, pp. 289-294, 2009. DOI: 10.1109/ICAL.2009.5262912.

[14] R. Stricker, S. Müller, E. Einhorn, C. Schröter, M. Volkhardt, K. Debes, H.-M. Gross, "Interactive mobile robots guiding visitors in a university building," 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, pp. 695-700, 2012. DOI: 10.1109/ROMAN.2012.6343832.

[15] P. Shruthi, R. Resmi, "Path Planning For Autonomous Car," 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), pp. 1387-1390, 2019. DOI: 10.1109/ICICICT46008.2019.8993295

[16] J. Chen, R. Zhang, W. Han, W. Jiang, J. Hu, X. Lu, X. Liu and P. Zhao, "Path Planning for Autonomous Vehicle Based on a Two-Layered Planning Model in Complex Environment," Journal of Advanced Transportation 2020, vol. 2020, pp. 1-14, 2020. DOI: 10.1155/2020/6649867

[17] D. Cagigas and J. Abascal, "Hierarchical Path Search with Partial Materialization of Costs for a Smart Wheelchair," Journal of Intelligent and Robotic Systems volume, vol. 39, pp. 409-431, 2004. DOI: 10.1023/B:JINT.0000026090.00222.40

[18] H. Ryu, "Hierarchical Path-Planning for Mobile Robots Using a Skeletonization-Informed Rapidly Exploring Random Tree*," Applied Sciences 10, vol. 10, no. 21, 2020. DOI: 10.3390/app10217846

[19] M. Seder, K. Juric-Kavelj and I. Petrović, "Automatic creation of hierarchical maps for indoor environments," in Proceedings of the Fifth International Conference on Computational Intelligence, Robotics and Autonomous Systems, pp. 19-24, 2008.

[20] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," Artificial Intelligence, vol. 99, pp 21–71, 1998. DOI: 10.1016/S0004-3702(97)00078-7

[21] W. Burgard, M. Moors, C. Stachniss, F. E. Schneider, "Coordinated multi-robot exploration," in IEEE Transactions on Robotics, vol. 21, no. 3, pp. 376-386, June 2005. DOI: 10.1109/TRO.2004.839232

[22] P. Beeson, N.K. Jong, B. Kuipers, "Towards autonomous topological place detection using the extended voronoi graph," Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 4373-4379, April 2005. DOI: 10.1109/ROBOT.2005.1570793

[23] B. Park, J. Choi and W. K. Chung,"An efficient mobile robot path planning using hierarchical roadmap representation in indoor environment," 2012 IEEE International Conference on Robotics and Automation, pp. 180-186, 2012. DOI: 10.1109/ICRA.2012.6225368.

[24] J. O. Wallgrün, "Autonomous Construction of Hierarchical Voronoi-Based Route Graph Representations," In Proceedings of the 4th international conference on Spatial Cognition: Reasoning, Action, Interaction (SC'04), Springer-Verlag, Berlin, Heidelberg, pp. 413–433, 2004. DOI: 10.1007/978-3-540-32255-9-23

[25] Z. He, H. Sun, J. Hou, Y. Ha S. Schwertfeger, "Hierarchical topometric representation of 3D robotic maps," Autonomous Robots, vol. 45, pp. 755–771, 2021. DOI: 10.1007/s10514-021-09991-8

[26] R. Bormann, F. Jordan, W. Li, J. Hampp, M. Hägele," Room segmentation: Survey, implementation, and analysis," 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1019-1026, 2006. DOI:10.1109/ICRA.2016.7487234

[27] J. Chen, C. Liu, J. Wu and Y. Furukawa, "Floor-SP: Inverse CAD for Floorplans by Sequential Room-Wise Shortest Path," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 2661-2670, 2019.

[28] Z. Zivkovic, B. Bakker, B. Krose, "Hierarchical Map Building Using Visual Landmarks and Geometric Constraints," 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. DOI:10.1109/IROS.2005.1544951

[29] C. Galindo, J. A. Fernandez-Madrigal and J. Gonzalez, "Improving efficiency in mobile robot task planning through world abstraction," in IEEE Transactions on Robotics, vol. 20, no. 4, pp. 677-690, Aug. 2004. DOI: 10.1109/TRO.2004.829480

[30] M. Seder, P. Mostarac and I. Petrović, "Hierarchical path planning of mobile robots in complex indoor environments," Transactions of the Institute of Measurement and Control, vol. 33 (3–4), pp. 332–358, 2011. DOI:10.1177/0142331208100107

[31] YW. Huang, N. Jing and E. Rundensteiner, "A Hierarchical Path View Model for Path Finding in Intelligent Transportation Systems," GeoInformatica 1, pp. 125–159, 1997. DOI:10.1023/A:1009784527790

[32] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernández-Madrigal, J. González, "Multi-hierarchical semantic maps for mobile robotics," 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2278-2283, 2005. DOI: 10.1109/IROS.2005.1545511

[33] K.-F. Richter, S. Winter, S. Santosa, "Hierarchical Representations of Indoor Spaces," Environment and Planning B: Planning and Design, vol. 38(6), pp. 1052–1070, 2011. DOI: 10.1068/b37057

[34] M. Mekni, B. Moulin, "Hierarchical Path Planning for Multi-agent Systems Situated in Informed Virtual Geographic Environments," 2010 Second International Conference on Information, Process, and Knowledge Management, pp. 48-55, 2010. DOI: 10.1109/eKNOW.2010.11

[35] R. Philippsen, "A Light Formulation of the E* Interpolated Path Replanner," Autonomous Systems Lab, Ecole Polytechnique Federale de Lausanne, 2006. DOI: 10.3929/ethz-a-010113621

[36] J. Sethian, "A fast marching level set method for monotonically advancing fronts," Proceedings of the National Academy of Sciences, vol. 93, pp. 1591-1595, 1996. DOI: 10.1073/pnas.93.4.1591

[37] M. Čikeš, M. Đakulović and I. Petrović, "The path planning algorithms for a mobile robot based on the grid occupancy map of the environment — A comparative study," 2011 XXIII International Symposium on Information, Communication and Automation Technologies, pp. 1-8, 2011.

[38] J. Gregorić, M. Seder and I. Petrović, "Autonomous Hierarchy Creation for Path Planning of Mobile Robots in Large Environments," 17th International Conference on Intelligent Autonomous Systems (IAS-17), pp. 798-810, 2022.

[39] G. Grisetti, C. Stachniss, W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," IEEE Transactions on Robotics 2007, vol. 23, pp. 34–46, 2007. DOI: 10.1109/TRO.2006.889486

[40] M. Benko Loknar, G. Klančar, S. Blažič, "Minimum-Time Trajectory Generation for Wheeled Mobile Systems Using Bézier Curves with Constraints on Velocity, Acceleration and Jerk," Sensors 2023, vol. 23, pp. 1982, 2023. DOI: 10.3390/s23041982

[41] Yong-tao Liu, Rui-zhi Sun, Tian-yi Zhang, Xiang-nan Zhang, Li Li, Guoqing Shi, "Warehouse-Oriented Optimal Path Planning for Autonomous Mobile Fire-Fighting Robots", Security and Communication Networks, vol. 2020, 2020. DOI: 10.1155/2020/6371814

[42] S. Ishihara, M. Kanai, R. Narikawa, T. Ohtsuka, "A Proposal of Path Planning for Robots in Warehouses by Model Predictive Control without Using Global Paths," IFAC-PapersOnLine, vol. 55, pp. 573-578, 2022. DOI: 10.1016/j.ifacol.2022.11.244.