Autonomous Hierarchy Creation for Path Planning of Mobile Robots in Large Environments

Jelena Gregorić, Marija Seder and Ivan Petrović

All authors are from University of Zagreb, Faculty of Electrical Engineering and
Computing, Laboratory for Autonomous Systems and Mobile Robotics (LAMOR),
Zagreb, Croatia {jelena.gregoric, marija.seder, ivan.petrovic}@fer.hr

Abstract. The main task of mobile robots in large environments such as factories, warehouses, and open spaces is to transport goods and people. Planning the path in large environments using classical graph-based search is computationally too intensive. 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 hierarchy 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 in 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 our University premises resulting in five abstract levels.

1 Introduction

Mobile robots have increasingly been used in industry and households. Their main task is to move through environment avoiding obstacles and living beings. Until now, mobile robots have mostly performed their tasks in small environments that can be easily described using classical representations of space such as occupancy grid maps [12] and topological graphs [2, 11]. It is obvious that the workspaces of mobile robots expand and encompass large, complex outdoor and indoor environments. Spaces like multi-room floors, multi-floor buildings, or a group of buildings are difficult to represent using classical approaches. Also, path planning in a large graph becomes very inefficient. To solve the mentioned problem of describing large environments, it is necessary to use a different approach and organize the space into simpler graphs to reduce the complexity and increase the efficiency of path planning. A suitable choice is the hierarchical decomposition of the environment based on hierarchical graphs (H-graphs) [3, 4, 8].

The H-graph is a combination of classical metric and topological maps at different levels of abstraction (see Fig. 1). The H-graph uses a skeleton of precalculated paths (partial paths) between the key points called the bridge nodes.

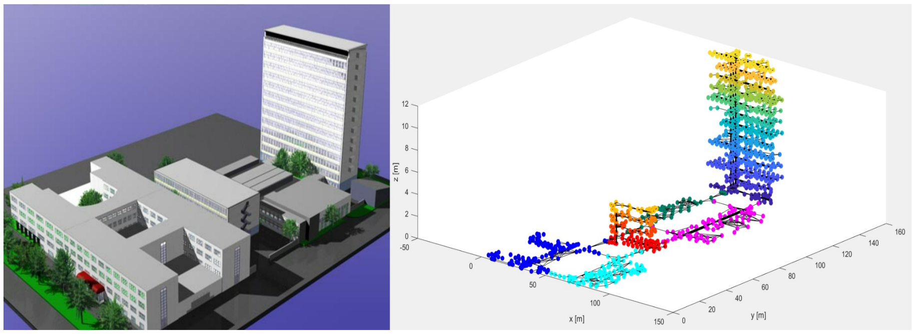


Fig. 1: Representation of FER buildings in reality and as an H-graph.

This approach to graph creation represents large environments simultaneously at different resolutions, i.e., at different abstract levels. A lot of research on creating the hierarchical graph exist in the literature [1,3–5,8,9,13]. The creation of an H-graph is usually done manually, which is a time-consuming and complex process. 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 [8] 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.

This paper proposes autonomous hierarchy creation with extension to more than three levels of abstraction. The proposed algorithm receives the floor plans as input data and creates an H-graph as output. We follow the approach of room extraction algorithm presented in [8] 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. We apply the E* algorithm [10] 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 [6] and provides a path that is very close to the optimal solution [7]. 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 ° [7, 9]. The process of graph creation is done offline and without any human intervention. The generated H-graph is a good basis for online and inter-building path planning. We showed the efficiency of the H-graph creation of a multi-building environment at the Faculty of Electrical Engineering and Computing (FER). 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 or more levels; ii) generation of nodes in H-graph that represent the elevators and the passages between the buildings; and iii) autonomous creation of an H-Graph with natural close-to-optimal paths and autonomous identification of bridge nodes.

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 FER example concluded by Section V.

2 Modeling of the Large Environment Using H-Graph

Hierarchical decomposition is necessary to achieve feasible path planning in large environments. In the following, the definition of the H-graph is given [9], together with the classification of nodes and optimality conditions for graph creation.

2.1 H-Graph Definition

The H-Graph is a sequence of hierarchical levels $L = L_0, L_1, ..., L_D$, where D is the depth of the hierarchy. 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 environment. Each level $L_i (0 \leqq i \leqq D)$ 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 egde $a(n_J, n_K, w_H) \in A$ 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 $x, y, z \in \mathbb{N}$. A weight $w_I \in W$ is a positive real number $(w_I \in \mathbb{R})$.

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 which can be selected from a robot path planner. Bridge nodes, noted by $B \subset N$, were divided in two classes in [3]: vertical bridge nodes are the vertical connections, i.e., elevators, that connect two floors (submap nodes) in a building, and horizontal bridge nodes connect two submap nodes as the horizontal connections, i.e., doorways. The concept of vertical bridge nodes does not imply a connection between hierarchical levels but a connection of the environment parts within the same level and allow the robot movement in the vertical direction through environment.

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:

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

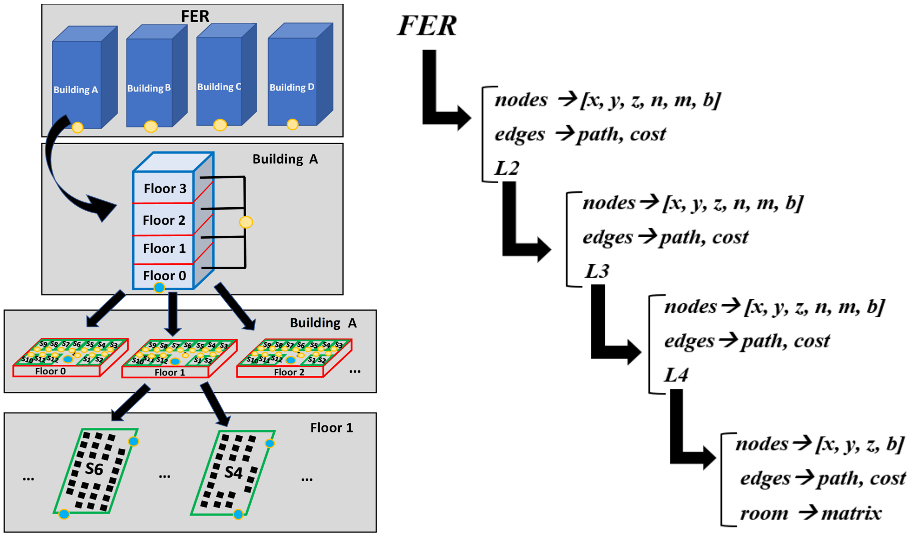
where $|T|$ denotes the number of nodes on the partial path T. The cost of the partial path $c(T)$ is defined as the sum of the weights of the edges on the partial path. Partial paths are optimal-length paths calculated offline. Partial paths are introduced to avoid time-consuming path corrections at deeper levels of abstraction in the case of path replanning. 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 Optimality

To ensure optimal global path and reduce the computational complexity of path replanning in real-time, we follow [9] 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 duplicated bridge nodes to maintain consistency through hierarchy.

The indoor environment is geometrically divided into rooms and halls. 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. To ensure path optimality, bridge nodes must be placed in a way that the union of the optimal partial paths between the bridge nodes gives the optimal global path. Also, the positions of bridge nodes play an important role in solving the problem of robots passing through narrow passages. The safest path in narrow passages is the one in the middle of the passage, because in this way the robot has the minimized risk of colliding with an obstacle. Therefore, we placed the bridge nodes exactly in the middle of the passage, at the door that separates the rooms. In this way, path optimality is achieved, and safety is increased.

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, and 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, which 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 level above and regular bridge nodes are at the borders of two submap nodes of that level. 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, 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



(a) (b)

Fig. 2: (a) An illustration of the hierarchical division of FER; (b) structure as output data from the autonomous hierarchy creation algorithm.

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.

3 Autonomous H-Graph Creation

In this work, the autonomous hierarchy creation is based on the principles of consistency and optimality, as described in the previous section. An H-graph is created only once and offline. The H-graph is organized into a data structure which is shown in Fig. 2(a) and Fig. 2(b). In this structure is a list of *nodes* at each level. The nodes are described by the corresponding (x, y, z) coordinates. The variables n and m indicate which nodes are children at the level below. The variable b is the index of each node at some level. At each level is a structure called *edges* that describes all partial paths of that level. The structure *edges* consists of the matrix *path* that stores the path described in the coordinate space and the real number *cost* that stores the price of the path. At each level, there is also a substructure that leads to a lower level. At the lowest level, we have a set of room matrices that contains a room occupancy grid maps. The occupancy grid map is continuously updated as the robot moves in the environment according to the detected obstacles by the robot sensors.

3.1 Creating Occupancy Grid Map from the Floor Plan

The input data given to the algorithm for the autonomous hierarchy creation are floor plans of all floors in *.png* format and the corresponding parameter files in *.yaml* format. The floor plan images could be obtained from CAD program or automatically via SLAM, and must not contain details such as door markings, room descriptions, elevations, etc. Each *yaml* file contains characteristic parameters that describe each floor plan: the name of image which is paired with *yaml* file, map resolution and origin in meters. Additional parameters specify the creation of the occupancy grid map and the hierarchy: the radius of the described circle around the robot's footprint noted as the radius of the robot in the following, the size of the cell, and the maximal width of the narrow passage. The algorithm read 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. 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. 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 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.

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 [8] and extend the algorithm to autonomously find locations of elevators. 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 the narrow passage that can be detected is equal to two widths of the safety cost mask. Each group of cells which are defined as narrow passage is assigned a central node with coordinates at the center of the narrow passage.

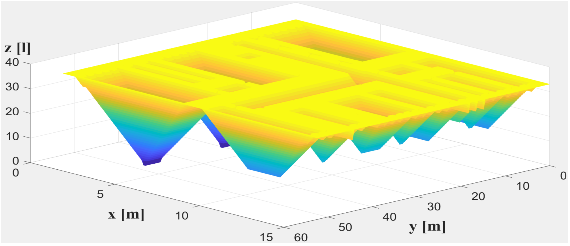


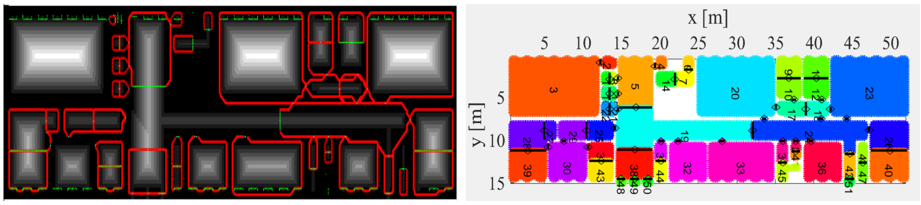
Fig. 3: Occupancy grid map with a safety cost mask represented as cell occupancy function in 3D.

Bridge nodes must be placed in the middle of narrow passages and at the boundary between two rooms. The position of the bridge nodes is not necessarily determined at the door location, but it is certainly in the middle of a narrow passage. 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. If such closed loop exists, the examined coordinates are declared as the 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, 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 [13]. Furthermore, the higher the parameter for the maximal width of the narrow passage is, i.e., the larger safety cost mask, the more found rooms will exist. The procedure of determining the bridge nodes is described in more details in [8].

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

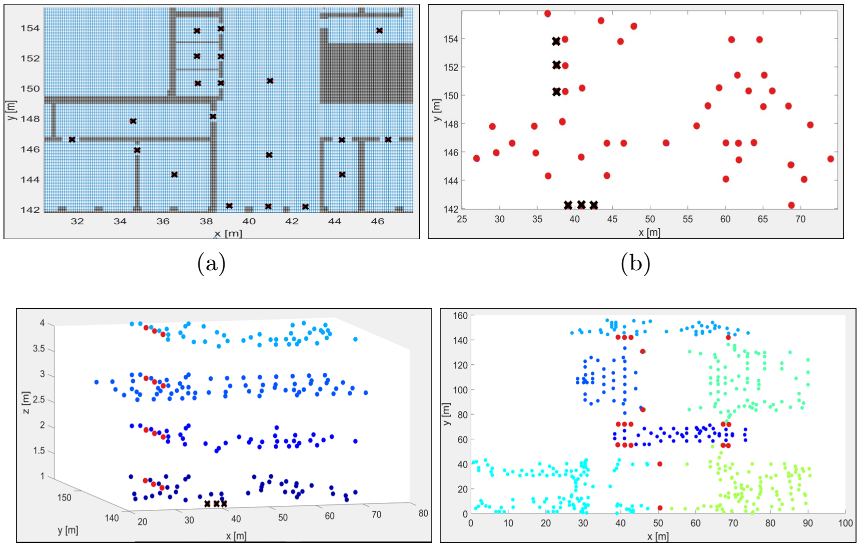
The described H-graph of FER has five hierarchical levels ($D = 4$). The most abstract level L_0 contains only one node. At level L_D is a graph created from the occupancy grid map. The occupancy grid map at level L_4 is shown in Fig.



(a)

(b)

Fig. 4: (a) Loops across the occupied cells 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).



(a) (b)

(c) (d)

Fig. 5: Nodes of: (a) level L_4; (b) level L_3; (c) level L_2; (d) level L_1. For clarity all bridge nodes are drawn at each level, but only the red ones are actually nodes of the graph at that level. The black x sign indicates marginal bridge nodes.

5(a) along with the marginal bridge nodes at that level (black x). Bridge nodes are located at each passage between two rooms.

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 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 bridge nodes at the level below.

The marginal bridge nodes at level L_3 are located at places of elevators and at the ground floor intersections of two adjacent buildings. 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. Fig. 5(b) shows the nodes of level L_3 composed of regular (red dots) and marginal (black x) bridge nodes.

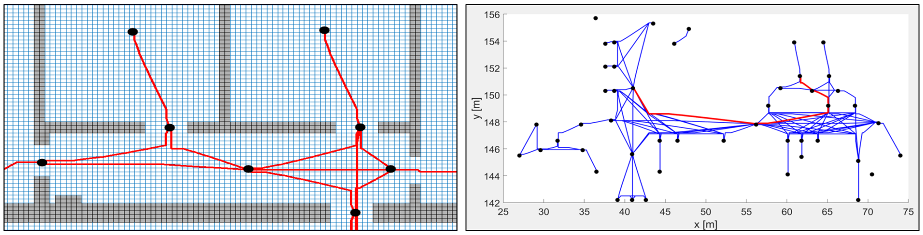
At the next level L_2, the graph consists of marginal bridge nodes from level L_3. Thus, the marginal bridge nodes from level L_3 with the corresponding

(x, y, z) coordinates, are duplicated in level L_2, and used to create a graph on that level. 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(c) represent all bridge nodes at level L_2, while the marginal bridge nodes (three black x) 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. Fig. 5(d) 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 color representing each building. We can notice that the bridge nodes of level L_1 are at connections of buildings.

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. In this work, we use partial paths that connect two marginal bridge nodes within a common submap node. 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 last level which contains bridge nodes.



(a)

(b)

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

Partial paths between the marginal bridge nodes at level L_4 are determined by an E* algorithm. The E* algorithm search the graph from the goal node and finds all possible paths in the graph. For this purpose, we single out an occupancy grid map 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 E* 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 abstract graph consists of nodes and edges which build a simple one-level graph. The nodes are marginal bridge nodes from level L_4, and the edges are partial paths determined at level L_4. At all higher levels, the partial 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 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.

4 Simulation Results

In this section we analyse the proposed algorithm according to the reduction of nodes comparing to the plain graph search, and partial paths quality compared to the ones obtained by the A* graph search. FER has four buildings marked with letters: A, B, C, and D (see Fig. 2). The floor plan of each floor in each building is individually conducted through the described algorithm. Each floor plan is the basis for creating all abstract levels using the previously described parameters: we selected a Pioneer-like robot with the radius 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 room 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. The floor plans of buildings B and D were divided into two parts due to their large dimensions and therefore considered as four buildings.

4.1 H-graph Analysis

Results of the proposed algorithm for autonomous H-graph creation is given in Table 1 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 $B_{\mathrm{Nmarginal}}$. 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. 1. 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.

Table 1: Review of nodes at each level.

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

4.2 H-graph and Plain Graph Paths

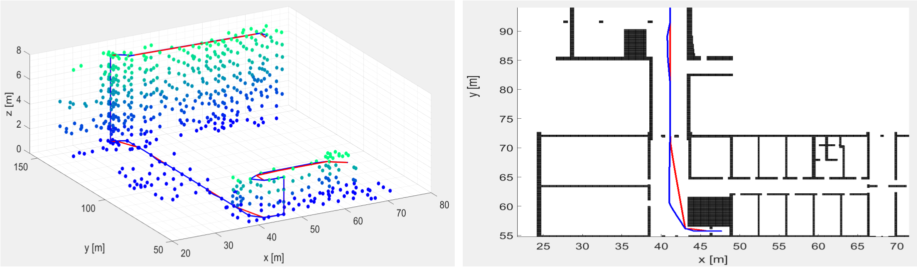
We tested the proposed hierarchy in the path planning example where the start is on the third floor of building A and the goal is on the seventh floor of building C. First, the rooms where the start and goal are located are searched by the E* algorithm to find the path to the bridge nodes (doors). The rest of the path is found by searching the marginal bridge nodes at each level. On the other hand, path planning within the plain graph searches all nodes on each floor through which the path passes. The results of the comparison are shown in Table 2 and the path comparison is shown in Fig. 7. The results show that about 50 times fewer nodes need to be searched using the H-graph compared to the plain graph. Searching fewer nodes significantly saves computational time. Path planning within an H-graph results in a 2 % longer path compared to a plain graph because somewhere bridge nodes are not on the doors but in the middle of the passage to form rooms with fewer end nodes. This proves that position of the bridge nodes is indeed at optimal place because they do not significantly affect the length of the path.

Table 2: Comparison of path planning in H-graph and plain graph.

Graph	N	l [m]
H-graph	5 497	189.4575
Plain graph	289 199	185.3772

4.3 Comparison of Partial Paths by E* and A*

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 path changes direction $n\alpha$, the sum of the all direction change angles $\sum \alpha$ and the number of points in which the change of direction is greater than or equal to $45\,^\circ$ $n\alpha_{\geqq 45\,^\circ}$. Algorithms E* and A* are executed at 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 the



(a) (b)

Fig. 7: Path found within H-graph (blue) and within plain graph (red): (a) set in the context of the H-graph; (b) zoomed in order to detect deviations.

Table 3: Comparison of E* and A* algorithm in map of room.

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

initial planning are shown in Table 3 and the path comparison is shown in Fig. 8. The E* algorithm outperforms the A* algorithm in path length and direction changes. On average, for the found paths from each node in room shown in Fig. 8 to the door, A* has about 3 % higher path cost and the path has significant directional changes at several path points.

5 Conclusion and Future Work

We presented the autonomous hierarchy creation for path planning of mobile robots in large environments. We demonstrate the necessary conditions of op-

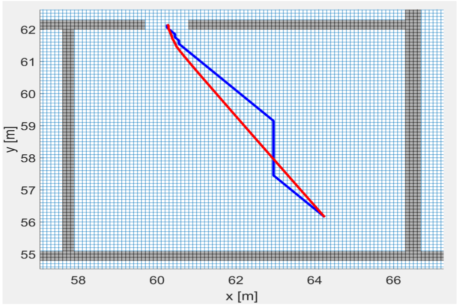


Fig. 8: Path found in the occupancy grid map with E* algorithm (red) and A* algorithm (blue).

timality during the creation of H-graph. The test of the algorithm on FER buildings demonstrated successful construction of the graph. We have shown that the path planning algorithm in the H-graph explored 50 times fewer nodes than in the classical non-hierarchical graph. Furthermore, testing shows that our algorithm supports modular graph upgrades when it is needed. Future work will focus on fast path replanning within created H-graph.

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. 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.
2. C. Paul Bonnington and Charles H.C. Little, ”The Foundations of Topological Graph Theory,” Springer-Verlag, New York, 1995.
3. D. Cagigas, ”Hierarchical D* algorithm with materialization of costs for robot path planning,” Robotics and Autonomous Systems, vol. 52, pp. 190-208, 2005.
4. H. Ryu, ”Hierarchical Path-Planning for Mobile Robots Using a Skeletonization-Informed Rapidly Exploring Random Tree*,” Applied Sciences 10, no. 21, 2020.
5. 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.
6. 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.
7. M. Čikeš, M. Đakulović and I. Petrović, ”The path planning algorithms for a mobile robot based on the occupancy grid map of the environment — A comparative study,” 2011 XXIII International Symposium on Information, Communication and Automation Technologies, pp. 1-8, 2011.
8. 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.
9. 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.
10. R. Philippsen, ”A Light Formulation of the E* Interpolated Path Replanner,” Autonomous Systems Lab, Ecole Polytechnique Federale de Lausanne, 2006.
11. 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.
12. S. Thrun, W. Burgard and D. Fox, ”Probabilistic robotics,” Cambridge, Massachusetts: MIT Press, 2005.
13. 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.