Model Predictive Control of Wheeled Robots Based on Hierarchical Planning

Jelena Rogina, Marija Seder, and Ivan Petrović

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

Abstract. This paper presents a motion planning method for wheeled mobile robots in large environments based on the integration of model predictive control (MPC) and hierarchical path planning. While the Hierarchical E* (HiE*) algorithm solves the path planning problem in large environments by employing a multi-level hierarchy abstraction to reduce graph complexity and enable time-efficient path planning, MPC ensures safe robot motion by generating feasible control sequences. The MPC objective function is based on the cost-to-goal of bridge nodes on the path, ensuring passage through narrow corridors and eliminating the need for exhaustive free-space searches to avoid local minima. The algorithms are verified by simulation with a differential drive mobile robot equipped with a laser range finder in a large warehouse environment. The simulation results demonstrate real-time navigation, significantly enhancing path planning and motion control efficiency in large-scale environment compared to a non-hierarchical motion planning algorithm.

Keywords: motion planning · hierarchical path planning · H-graph · Model Predictive Control · navigation function

1 Introduction

Motion planning is a core task of a wheeled mobile robot to ensure safe and efficient navigation from a starting point to a goal position while avoiding obstacles. It consists of global path planning and local path following processes. While path planning methods [1], [2] focus on finding a feasible and often optimal path from the start point to the goal point, ensuring path efficiency according to many different performance criteria, path following involves controlling the robot's motion to follow a pre-computed global path by adjusting translational and rotational velocities and ensuring obstacle avoidance [3]- [5].

Since the performance of conventional motion planning methods strongly depends on the properties of the map (graph), such as size, number of nodes, resolution, etc., they become inefficient in large environments. Hierarchical path planning methods [11]- [15] address this problem by introducing hierarchical graphs (H-graphs) [11], [13], [16], [17], which divide the large environment into a multi-level structure and combine higher level topological maps representing

larger regions of the environment (rooms, corridors, etc.) with lower level classical metric maps representing smaller regions of the environment with full details about static and dynamic obstacles. This hierarchical structure reduces computational complexity by first planning a path through key waypoints at higher levels and then refining the path by adding smaller parts of the path that lie within the start and goal regions. While hierarchical path planning effectively addresses the problem of computationally efficient path planning in large environments, the challenge of path tracking or ensuring accurate real-time convergence to the goal in large environments remains unexplored. A very common approach in this research area is model predictive control (MPC), which solves the numerical optimization problem over a receding time horizon by searching the space of linear and angular velocities to predict the system outputs based on the current states, the mathematical model, and considering the kinematic and dynamic constraints. Some works propose different MPC typologies for path tracking that combine multiple criteria to reduce the lateral and longitudinal deviation of the robot from the given path [6], while others introduce penalty functions to avoid dead ends and detours [10]. However, these approaches are very complex as the local path tracking problem is split into a combination of several components, which are often very complex for real-time optimization and require the separate setting of many different parameters. In contrast, MPC-based point (goal) stabilization guarantees the convergence of the robot to the goal point by using the cost to the goal in the objective function computed by grid-based search algorithms such as D* or E* [7], [8], [9]. However, none of the mentioned approaches is suitable for use in large environments such as warehouses and multi-building environments, as the computation of the navigation function is expensive in cases where local-minima-free navigation in the entire environment needs to be ensured.

In this paper, we present a novel approach for MPC based on hierarchical path planning to solve robot motion planning problems in large environments. We use the H-graph structure from our previous work [16] with natural close-to-optimal subpaths between bridge nodes and guaranteed global path optimality conditions. Our first contribution is the Hierarchical E* (HiE*) algorithm, which finds near-optimal paths by merging partial paths. HiE* is based on the E* algorithm [18], which is classified as an *Any-angle* algorithm [2] and uses interpolation to approximate continuous optimal path costs, leading to near-optimal solutions as we have shown in [16]. Placing the bridge nodes in narrow passages also ensures that the union of the optimal partial paths between the bridge nodes yields the optimal total path. The second contribution builds MPC based on the hierarchy to generate the admissible robot trajectories that ensure safe and efficient robot motion. We introduce a navigation function based on convergence to bridge nodes on the optimal path considering the cost-to-goal ratio of the bridge nodes, ensuring passage through narrow corridors without requiring exhaustive free-space search and connectivity to avoid local minima. In this way, time-efficient path planning in large environments and selection of appropriate motion commands was achieved. We have shown the efficiency of

our approach in large-scale warehouse simulation on different resolutions of grid cells and compared it with a non-hierarchical motion planning approach based on the E* algorithm and the MPC proposed in [8]. The proposed hierarchical path planning algorithm finds the near-optimal path through H-graph search, which ensures the safety and the passage of the path through the center of the narrow passages with significantly reduced graph search time. Moreover, the proposed MPC navigation algorithm generates robot trajectories that reach the goal in about the same time as the non-hierarchical approach, but with significantly lower computational cost. Moreover, our method is particularly effective in high-resolution environments, which is crucial for navigating through narrow passages.

The rest of this paper is organized as follows. In Section 2, a path planning algorithm based on an H-graph is presented. Model predictive control with a new navigation function that includes a hierarchy is presented in Section 3. The evaluation of the results is presented in Section 4, followed by the conclusion in Section 5.

2 Hierarchical E* Path Planning Algorithm

The creation of an H-graph is covered in our preliminary work in [16], where an algorithm for the autonomous creation of H-graph from floor plans is presented together with a discussion about conditions for global path optimality. The H-graph is described as a sequence of hierarchical levels $L = L_0, L_1, ..., L_D$, where L_D is the lowest level of abstraction and represents the most detailed description of the environment. Each level L_i $(0 \leq i \leq D)$ is represented by graph $G_i = (N_i, A_i, C_i, W_i, T_i)$, with sets of nodes, edges, Cartesian coordinates of nodes, weights of edges and partial paths. The complete H-graph G is the union of graphs from all levels where connections between hierarchical levels are achieved through bridge nodes.

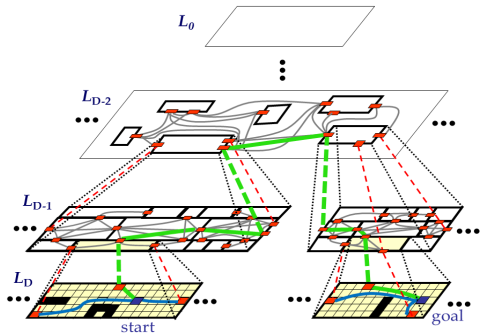


Fig. 1: Bottom-up path planning process.

Algorithm 1 Start room search (S)

Input: start point S
1: $\mathcal{O} \leftarrow S$, $h(S) = 0$
2: $S_room, S_floor, S_building \leftarrow$ **where-is-point**(S)
3: $roomBN \leftarrow$ Select bridge nodes of start room
4: $grid \leftarrow$ Select search grid of start room
5: **for** $i = 1 \ldots number_of_roomBN$ **do**
6: $n \leftarrow roomBN[i]$
7: $\mathcal{T}(S, n) \leftarrow$ Find partial path ($grid$, S, n, E* search)
8: $\mathcal{O} \leftarrow \mathcal{O} \cup n$
9: $h(n) \leftarrow c(\mathcal{T}(S, n))$
10: $b_S(n) \leftarrow S$
11: $path_b_S(n) \leftarrow \mathcal{T}(S, n)$
12: **end for**
13: **while** $\mathcal{O} \neq 0$ **do**
14: $\mathcal{O} \leftarrow \mathcal{O} \backslash m$
15: $\mathcal{C} \leftarrow \mathcal{C} \cup m$
16: $k \leftarrow parent_submap_node(m)$
17: **if** $\exists n \in k \mid \exists \overline{\mathcal{T}}(n, m) \in \mathcal{A}$ **then**
18: **if** $n \notin \mathcal{O}$ or $h(n) > h(m) + c(\mathcal{T}(n, m))$ **then**
19: $\mathcal{O} \leftarrow \mathcal{O} \cup n$
20: $h(n) \leftarrow h(m) + c(\mathcal{T}(n, m))$
21: $b_S(n) \leftarrow m$
22: $path_b_S(n) \leftarrow \mathcal{T}(n, m)$
23: **end if**
24: **end if**
25: **end while**

The Hierarchical E* algorithm (HiE*) uses a bottom-up process (see Fig. 1) to search the H-graph through abstract levels to obtain the cost-optimal path solution in lower computational time. We used D* (or A*) search kernel [19]. The optimality of the global path in the environment is observed by the global path cost condition, which is defined as the sum of the edge weights on the path. The H-graph search starts at the deepest level L_D from the start and goal, separately by searching the "rooms" where the start and goal are located (Algorithm 1 is listed for the start room search only, while the goal room search is explained in the text). The start and goal nodes are added in the open list $\mathcal{O}$ with the value of cost functions $h(S) = 0$ and $g(G) = 0$, respectively (Alg. 1, line 1). Searched and expanded nodes are removed from the open list and added in the closed list $\mathcal{C}$ together with the pointers to the previous node $b_S(\cdot)$ or $b_G(\cdot)$ and the partial path connecting them $path_b_S(\cdot)$ or $path_b_G(\cdot)$. At the deepest level L_D, all end nodes in the start room are searched by the E* algorithm (Alg. 1, line 7), where h is stored as the accumulated cost of the path from the start. Analogously, the end nodes in the goal room are searched by the E* algorithm, where g is stored as the accumulated cost of the path from the goal. The search at level L_D continues until all bridge nodes in the start (goal) submap node are reached and added to the open list (Alg. 1, lines 3-12). When all bridge nodes from the deepest level are reached and added to the open list, hierarchical expansion to higher levels begins.

At a higher level of the hierarchy, a new parent submap node is determined and all associated bridge nodes are expanded from the duplicated bridge nodes from the level below according to the cost function from the start (or from the

Algorithm 2 HiE* Path Planning Algorithm

Input: start point S, goal point G, graph of the environment H_graph

```
 1: k_S ← parent_submap_node(S)
 2: k_G ← parent_submap_node(G)
 3: if k_S = k_G then
 4:     grid ← Select search grid of start room
 5:     total_path ← Find path (grid, S, G, E* search)
 6: else
 7:     Start room search(S)
 8:     Goal room search(G)
 9:     f_min ← ∞
10:     for i = 1 ... length(C) do
11:         n ← C[i]
12:         f(n) ← g(n) + h(n)
13:         if f(n) ≤ f_min then
14:             n* ← n
15:             f_min ← f(n*)
16:             index_min ← i
17:         end if
18:     end for
19:     n ← C[index_min]
20:     while n ≠ S do
21:         m ← backpointer(n)
22:         start_path = start_path ∪ T(n, m)
23:         n ← m
24:     end while
25:     while n ≠ G do
26:         m ← backpointer(n)
27:         goal_path = goal_path ∪ T(n, m)
28:         n ← m
29:     end while
30:     Reverse(start_path)
31:     total_path ← start_path ∪ goal_path
32: end if
33: return total_path
```

goal) (Alg. 1, lines 13-25). Two bridge nodes m and n at level L_i are connected by a partial path $\mathcal{T}(m, n)$ determined at the level below during the H-graph creation, where $c(\mathcal{T}(m, n))$ is the cost of the path between these two nodes that is saved in the H-graph structure. Then, the value of the cost function from the start to the node m is determined as $h(m) = h(n) + c(\mathcal{T}(n, m))$ (Alg. 1, line 20). Analogously, the cost from the goal $g(m)$ is determined for node expanding from the goal. Newly expanded nodes are added to the open list, together with pointers to the previous node and the partial path that connects them. The search from the start (goal) is repeated in higher levels of the hierarchy as long as there are bridge nodes in the open list, i.e., until the open list is empty, which happens at level L_0.

The initial HiE* path planning is shown by Algorithm 2. If start S and goal G have the same parent submap node, the path planning is realized by using a simple E* search within a common submap node (Alg. 2, lines 1-5). Suppose the parent submap node of the start and the goal is different. In that case, the H-graph from the start and the goal is searched (Alg. 2, lines 7-8) to determine the bridge node $n*$ with the lowest total path cost, which is defined as $f(n) = h(n) + g(n)$ (Alg. 2, lines 12-17) using cost from the start $h(n)$ and

cost from the goal $g(n)$ values calculated in Alg. 1. The path from the node $n*$ to the start S is determined by following the backpointers to the start and then reversed. The path from the node $n*$ to the goal G is determined by following the backpointers to the goal and then merged with the path from the start to obtain the complete path from start to goal.

3 Model Predictive Control for Path Following

To ensure convergent navigation of a wheeled robot in large environments, in this section, we present the concept of the proposed MPC based on hierarchical path planning.

3.1 Model Predictive Control Concept

We describe the concept of MPC [7], [8] that is used to develop a closed-loop navigation algorithm with guaranteed convergence in finite time for a non-holonomic robot with velocity and acceleration constraints. The MPC is based on solving an optimization problem at each control step, predicting the future behavior of a system based on a system model, and computing an optimal control sequence [20]. The kinematic model of a non-holonomic robot in the discrete-time form is as follows:

$$\begin{aligned} x(t+1) &= x(t) + v(t)T_s \cos(\theta(t) + \tfrac{\omega(t)T_s}{2}), \\ y(t+1) &= y(t) + v(t)T_s \sin(\theta(t) + \tfrac{\omega(t)T_s}{2}), \\ \theta(t+1) &= \theta(t) + \omega(t)T_s, \end{aligned} \tag{1}$$

where v and ω are linear and angular velocities, respectively, that are constant within each sampling time interval T_s while $t \in \mathbb{N}_0$, (x, y) are the Cartesian coordinates of the robot position, and θ is the orientation of the robot with respect to the positive x axis. The robot state is defined by $s(t) = (x(t), y(t), \theta(t)) \in \mathcal{S} \subset \mathbb{R}^3$, and the control input is defined by $u(t) = (v(t), \omega(t)) \in \mathcal{U} \subset \mathbb{R}^2$, where $\mathcal{S}$ is the state constraint set containing collision-free robot states, and $\mathcal{U}$ is the velocity constraint set. At time t, the control input $u(t)$ is subject to the kinematic and dynamic constraints of the form:

$$\begin{aligned} &u(t) \in \mathcal{U} = [0, v_{max}] \times [-\omega_{max}, \omega_{max}], \\ &\tfrac{u(t)-u(t-1)}{T_S} \in \mathcal{A} = [-\dot{v}_{max}, \dot{v}_{max}] \times [-\dot{\omega}_{max}, \dot{\omega}_{max}], \end{aligned} \tag{2}$$

where $\mathcal{A}$ is the acceleration constraint set, while $(\cdot)_{max}$ denotes the limits on velocities and accelerations. Note that backward motion is forbidden because the sensor field of view is in the forward direction. For the robot state $s := s(t) = (x(t), y(t), \theta(t))$ at the current time t, control actions are found by optimizing the objective function on the fixed horizon H. The MPC problem is defined as:

$$\begin{aligned} P_H(s): \quad & J^*(s) := \min_{\{u_k\}_0^{H-1}} J(\{s_k\}_1^H, \{u_k\}_0^{H-1}) \\ & \text{s.t.} \\ & \{u_k\}_0^{H-1}, \{s_k\}_1^H \in \mathcal{F}, \\ & N(s_H) \leq N(s_k), \ \text{ for } k = 1, ..., H, \end{aligned} \tag{3}$$

where $\{s_k\}_1^H$ is the evolution of the robot state over a horizon H (as a function of the control sequence $\{u_k\}_0^{H-1}$), according to the model (1), starting with $s_0 = s(t)$ and J is the objective function given by:

$$J(\{s_k\}_1^H, \{u_k\}_0^{H-1}) := \sum_{k=1}^{H} N(s_k) + \rho \sum_{k=0}^{H-1} ||u_k||_1, \tag{4}$$

where $\rho > 0$ is a small positive scalar value, and $N(s_k)$ is the navigation function measuring the cost-to-goal over states (see Sec. 3.2), with the constraint on the robot states so that the cost of the navigation function of the last state in the horizon interval does not increase (important for the convergence of the closed-loop system, see Sec. 3.3) [7], [8]. To ensure computational efficiency in real-time motion control, we include the reduced set of feasible control sequences $\mathcal{F} \subset \mathbb{R}^{2H}$ that are computed to ensure tractable online optimization as it is proposed in [7] instead of using an expensive optimization algorithm to search control actions in the velocity space $\mathcal{U}$. This method utilizes the Dynamic Window Approach (DWA) to search a space of discrete set of uniformly distributed translational and rotational velocities around the current robot velocity, taking into account the kinematic and dynamic constraints of the robot. This approach generates discrete circular arc-shaped collision-free trajectories with constant velocities for the $H_0 < H$ time intervals, whose velocity is zero at the end of the prediction horizon, ensuring that constraints are satisfied across the prediction horizon, with stopping conditions enabling finite-time convergence. The optimal trajectory is selected by minimizing an objective function, and the first control action is applied iteratively, maintaining computational efficiency and safety.

Any control sequence that attains the minimum in (3)–(4) is called an optimal control sequence and is denoted by $\{u_k^*\}_0^{H-1}$. Without loss of generality, we can assume that $\{u_k^*\}_0^{H-1}$ is unique. Then, the control applied to the system at time t is the first element of this sequence, $u(t) = u_0^*$. The same procedure (3)–(4) is repeated at time $t + 1$ for the new state $s(t + 1)$.

3.2 Navigation Function

The navigation function N is derived from the H-graph search algorithm, which finds the near-optimal path to the goal [16] where the optimality of the global path is observed by the path cost condition. The global path consists of partial paths between bridge nodes, ensuring both optimality and safe passage through narrow corridors. To avoid exhaustive free-space search in large environments and no local minima navigation simultaneously, we introduce a navigation function based on convergence to bridge nodes on the optimal path while considering their cost-to-goal. The lowest level of H-graph, L_D, in each room represents the environment as an occupancy grid map with a safety cost mask, dividing it into square cells of size a_{cell} and enlarging obstacles for the value of the safety cost mask as it was described in [16]. The mobile robot is considered as a point in the environment, while the real obstacles are enlarged for the integer number of

cells $\lceil r_{robot}/a_{cell} \rceil$, where r_{robot} is the robot radius. Given the robot's position, its corresponding H-graph subnode is always identifiable.

A piecewise continuous navigation function $N : \mathcal{S} \rightarrow [0, \infty\rangle$ is calculated for all states encompassed by the horizon H during robot navigation relative to the referent bridge node on the path B_{Nref} and its cost-to-goal value g. Let $p = (x, y, \theta)$ be a state with position (x, y) and orientation θ in the environment encompassed by the horizon H during the online navigation. Navigation function N for the state p is calculated as follows:

$$N(p) = g(B_{Nref}) + ||p - B_{Nref}||, \tag{5}$$

where $|| \cdot ||$ is the Euclidean distance. The referent bridge node B_{Nref} is a temporary sub-goal on the path, which changes as the robot follows a global path and navigates to the goal. The set of bridge nodes on the global path is defined as $B_{Npath} := \{B_{N1}, B_{N2}, ..., B_{Nm}\}$, where B_{Nm} is the global goal. The B_{Nref} at some time t is defined as:

$$B_{Nref}(t) := B_{Ni} \begin{cases} i := 1, & \text{if } t = 0, \\ i := i + 1, & \text{if } ||s(t) - B_{Ni}|| \leq R_{cl}(B_{Ni}), i < m, \end{cases} \tag{6}$$

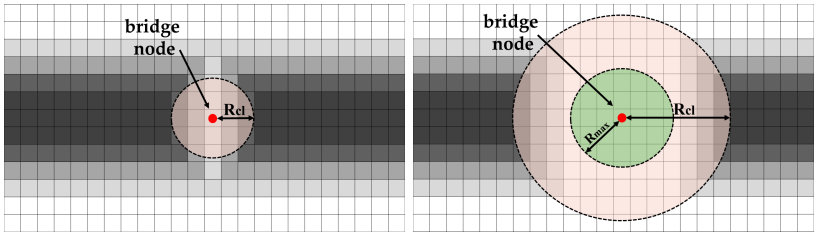
where $s(t)$ is the current robot state at the current time t and $R_{cl}(B_{Ni})$ is a clearance radius that determines a safe circular area around the bridge node B_{Ni}. Given that bridge nodes are located in narrow passages, a safe circular area around the bridge node ensures safe entry and exit from the narrow passage without colliding with an obstacle and a safe possible deviation from the path depending on the width of the narrow passage. A clearance radius R_{cl} around each bridge node indicates the distance from the bridge node to the closest obstacle (see Fig. 2) and affects the change of the reference bridge node. A clearance radius R_{cl} around some bridge node B_{Ni} is determined as follows:

$$R_{cl}(B_{Ni}) = \min(\min_{\forall o \in \mathcal{X}} ||B_{Ni} - o||, R_{max}), \tag{7}$$

where $\mathcal{X}$ is the set of occupied cells with occupancy values ∞ and R_{max} is the limit of maximum clearance radius (see Fig. 2 (b)) to ensure the precise convergence to each bridge node on the path, even in wider narrow passages.

3.3 Convergence of the Closed-Loop System

We omit the formal proof of the convergence to the goal of the closed-loop system for brevity. Instead, we give a brief idea of the proof as follows. The optimal value of the objective function J^*, employed as a Lyapunov function, needs to decrease with time during the navigation to the goal. According to the constraint in (3), which requires the decreasing value of the navigation function of the robot states at the end of horizon interval, we can derive $J^*(t) - J^*(t+1) \geq \rho ||u_0^*||_1$, which means that the Lyapunov function decreases with time if the optimal control u_0^* is not zero. If the optimal control at some time is $u_0^* = 0$, we need to switch to the



(a) (b)

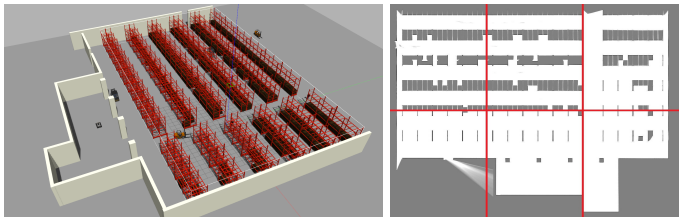
Fig. 2: A safe circular area around B_{Nref} determined (a) with the radius clearance R_{cl}; (b) with the the maximum clearance radius R_{max} for wider passages.

separate control procedure that will move the robot to the state with the lower value of the navigation function than the current state. This can be achieved by rotating in the direction of the next sub-goal (B_{Nref}) and then translating towards that sub-goal, and then switching back to the original control procedure (3)–(4). It needs to be noted that the sub-goal may not be visible from the current state due to the local obstacle configuration blocking its straight-line view. In that case, a new sub-goal needs to be selected from the partial path between bridge nodes that is visible from the current state. In practice, the switching of control procedures occurs only initially (before the robot started to move), in cases in which the robot is oriented in the opposite direction from the sub-goal.

4 Test Results

The proposed hierarchical path planning algorithm and the hierarchy-based MPC were tested in the simulation of a large-scale warehouse-like environment with dimensions of $52 \times 60\,\mathrm{m}^2$ (see Fig. 3) using a Pioneer-like differential drive mobile robot equipped with a laser range finder. The simulations were performed under the Robot Operating System (ROS) with two different map resolutions to evaluate computational efficiency, path optimality, and trajectory performances. The results were compared with a non-hierarchical motion planning approach that uses the E* algorithm [18] and MPC algorithm which uses navigation function derived from the E* grid-based search algorithm [8], in this work called grid-based-NF MPC. Hierarchical path planning was performed by searching the H-graph of the environment consisting of five abstract levels created in our previous work [16], while for non-hierarchical path planning, the entire environment represented by the occupancy grid map was searched. The algorithms were tested for different hierarchical levels where common submap node of the start and goal point are located. We also compared algorithms when the path passes through a narrow passage.

Both algorithms use an occupancy grid map with the safety cost mask to enlarge obstacles for the robot radius which is selected to be 0.25 m and to push the path 3.5 m from obstacles when possible. We set the maximum limits on the translation and rotation velocities to values $v_{max} = 2\,\mathrm{m/s}$, $\omega_{max} = 50\,^\circ/\mathrm{s}$,



(a) (b)

Fig. 3: Warehouse-like simulation environment shown: (a) in a 3D Gazebo world representation; (b) as the map which is built using SLAM and gmapping. Red lines indicate division of the map into six equal parts in H-graph on level L_1.

Table 1: Comparison of algorithms for different map resolutions.

Level of common submap node	Start/ Goal [m]	Algorithm	N [1]	c [1]	l_{path} [m]	t_{init} [ms]	l_{traj} [m]	t_{goal} [s]	t_{mpc} [ms]
L_0	Start (7.7, 14.7) Goal (-32.5, -4.4)	E* (cell=0.1 m)	269 861	11 354	58.32	891	55.21	30.5	4
		HiE* (cell=0.1 m)	3 168	12 082	56.93	13	55.83	32.6	3
		E* (cell=0.05 m)	1 128 568	43 145	57.42	3 329	55.89	30.7	9
		HiE* (cell=0.05 m)	6 409	44 524	56.17	26	55.26	31.4	2
L_0 narrow passage case	Start (-15.37, -3.49) Goal (-11.88, -9.50)	E* (cell=0.1 m)	269 861	3 076	13.21	802	12.42	12.1	10
		HiE* (cell=0.1 m)	4 136	3 361	12.87	18	12.69	10.3	2
		E* (cell=0.05 m)	1 128 568	11 918	13.27	3 525	12.49	12.0	6
		HiE* (cell=0.05 m)	8 657	12 841	11.92	41	12.83	11.4	2
L_3	Start (61.2, 56.6) Goal (71.4, 142.3)	E* (cell=0.1 m)	269 861	75	2.93	870	3.06	5.0	3
		HiE* (cell=0.1 m)	2 350	75	2.92	10	2.94	3.7	1
		E* (cell=0.05 m)	1 128 568	184	2.90	3 584	3.14	5.1	3
		HiE* (cell=0.05 m)	18 510	185	2.97	62	2.95	4.4	2

$\dot{v}_{max} = 1\,\mathrm{m/s^2}$, $\dot{\omega}_{max} = 200\,^\circ/\mathrm{s^2}$. The length of horizon for both algorithms is $H = 30$ with the duration of the time step $\triangle t = T_s = 0.1\,s$. For the hierarchy-based MPC algorithm, a safe circular area around the bridge node is determined with clearance radius R_{cl} calculated for each bridge node, while the value of maximum clearance radius is set to $R_{max} = 1\,\mathrm{m}$.

Table 1. shows the results of the comparison of algorithms for the grid map with the size of the cells 0.1 m and 0.05 m. To evaluate the efficiency of our approach we used relevant data for algorithm comparison in terms of the number of searched nodes N, the cost of the path c, the length of the path l_{path}, the initial path planning time t_{init} [1], the trajectory length l_{traj}, the execution time t_{goal} which represents driving time to the goal and the maximum time t_{mpc} required to calculate the navigation function in the trajectory points. The optimal path on the occupancy grid map with a safety cost mask is the one that minimizes the total path cost expressed in the number of weighted cells. The start and goal points are determined with the Cartesian coordinates (x, y) concerning the user-defined coordinate system. The results show for all cases that the HiE* algorithm significantly reduces the number of explored nodes compared to the E* algorithm, resulting in improved computational efficiency and significantly lower

[1] Inital planning time was measured on 11th Gen Intel Core i7-1165G7 CPU at 2.80GHz processor.

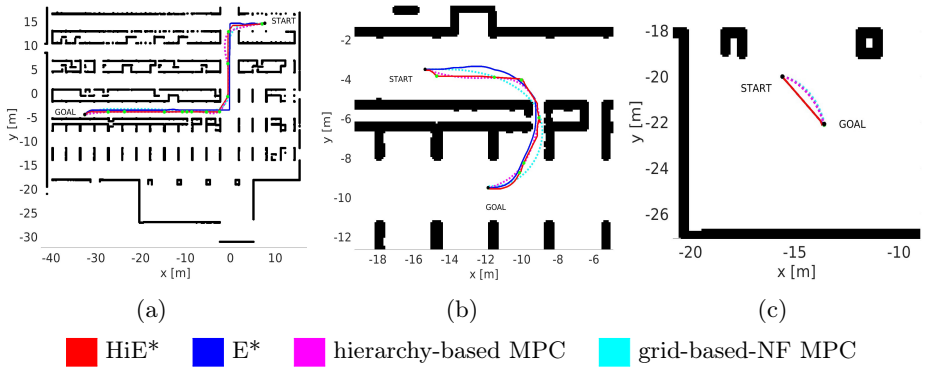


Fig. 4: A warehouse environment maps (cell size 0.1 m) with the paths found by the E* algorithm and HiE* algorithm and trajectories obtained by the grid-based-NF MPC and hierarchy-based MPC when the common submap node of start and goal is (a) at level L_0; (b) at level L_0 with passage through a narrow corridor; (c) at level L_3. Green dots indicate bridge nodes on the path.

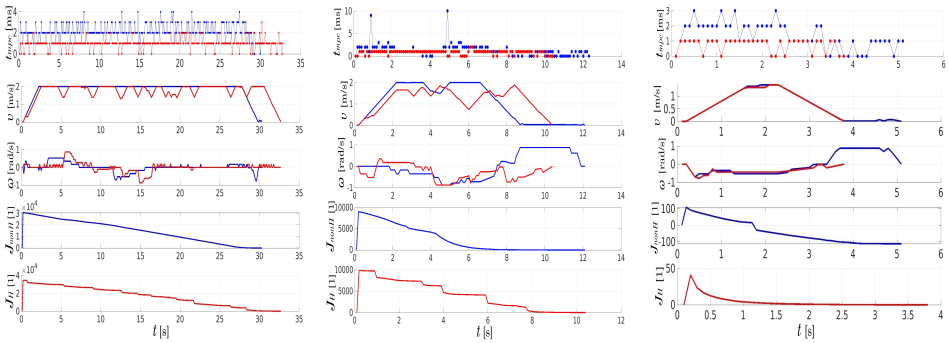


Fig. 5: Velocity profile, objective function and t_{mpc} for trajectories obtained by the grid-based-NF (blue) and hierarchy-based (red) MPC for different levels.

initial path planning time while maintaining near-optimal path cost and approximately equal path length. Also, it can be seen that for both grid map resolutions, both MPC navigation algorithms generate similar trajectory lengths in approximately equal time to reach the goal. From time t_{mpc}, it can be seen that the hierarchy-based MPC ensures smooth and safe robot motion while minimizing computational overhead by eliminating the computation of a cost interpolation to calculate the navigation function. This saves time for using other algorithms, such as advanced obstacle detection, while keeping the real-time navigation.

For visual comparison, Fig. 4 shows the planned paths and trajectories for all three cases with a grid cell size of 0.1 m. The hierarchy-based MPC ensures that a precise trajectory runs over the coordinates of the bridge nodes (green dots), ensuring safe passage through narrow passages and navigation without local

minima. In Fig. 4 (b), the simulation results confirm that the proposed hierarchy-based MPC navigation effectively guides the robot through narrow passages by tracking bridge nodes as sub-goals on the path. Fig. 4 (c) shows the case where the start and goal are in the same room, so that the H-graph search simplifies to an E* search at the level L_4, resulting in an overlap of the planned paths obtained by the E* search and the HiE* search as well as the trajectories traveled by the robot, which shows that the hierarchy-based navigation behaves similarly to the grid-based navigation in this case. In Fig. 4, the hierarchy-based MPC leads to a local overshoot of the trajectory due to convergence to a particular sub-goal on the path without considering the position of the next bridge node, which could be improved if a local search for the optimal bridge node were included in the MPC criterion, but this would further slow down the navigation algorithm. Figure 5 shows linear and angular velocity profiles and objective function values over time t_{goal}. The grid-based-NF MPC produces smoother velocities than the hierarchy-based MPC, which slows down the robot before sub-goals (bridge nodes) on the path to ensure safe passage through corridors. To overcome this problem, smoothing the velocities on straight parts of the path could be considered in future work. The objective functions J_{nonH} for the grid-based-NF MPC and J_H for the hierarchy-based MPC decrease with time, confirming the successful reach of the goal without searching the entire large environment.

5 Conclusion

The presented HiE* path planning algorithm significantly reduces the number of searched nodes and the computational complexity, thus enabling real-time path planning in large environments. The hierarchy-based MPC ensures smooth convergence to the goal by considering the cost-to-goal of bridge nodes and local trajectories, thus avoiding expensive optimization. The objective function evaluates local control sequences based on the distance to the bridge node and its cost-to-goal. Compared to a non-hierarchical approach, HiE* effectively overcomes the challenges of large-scale path planning by reducing the number of searched nodes and the initial path planning time, while the proposed hierarchy-based MPC ensures smooth and safe navigation through the environment without the need for exhaustive free-space search and connectivity to avoid local minima. This method is well-suited for applications such as autonomous warehouse navigation, with future work focusing on dynamic and multi-robot environments.

Acknowledgement

This research has been supported by the European Regional Development Fund under the grant PK.1.1.10.0007 (DATACROSS).

References

1. K. Karur, N. Sharma, C. Dharmatti, J.E. Siegel, "A Survey of Path Planning Algorithms for Mobile Robots," Vehicles, 2021; 3(3):448-468.

2. J. R. Sánchez-Ibáñez , C. J. Pérez-del-Pulgar, A. García-Cerezo, "Path Planning for Autonomous Mobile Robots: A Review," Sensors, 21(23):7898, 2021.
3. L. Pacheco, N. Luo, "Testing PID and MPC Performance for Mobile Robot Local Path-Following," International Journal of Advanced Robotic Systems, 2015;12(11).
4. Nguyen Hung et al., "A Review of Path Following Control Strategies for Autonomous Robotic Vehicles: Theory, Simulations, and Experiments," Journal of Field Robotics, vol. 40, pp. 747-779, 2023.
5. R. Oftadeh et al., "Universal Path-Following of Wheeled Mobile Robots: A Closed-Form Bounded Velocity Solution," Sensors, 21(22):7642, 2021.
6. P. Stano, U. Montanaro, D. Tavernini, M. Tufo, G. Fiengo, L. Novella, A. Sorniotti, "Model predictive path tracking control for automated road vehicles: A review, Annual Reviews in Control," vol. 55, pp. 194-236, 2023.
7. M. Seder, M. Baotić and I. Petrović, "Receding Horizon Control for Convergent Navigation of a Differential Drive Mobile Robot," in IEEE Transactions on Control Systems Technology, vol. 25, no. 2, pp. 653-660, 2017.
8. M. Seder, G. Klančar, "Convergent wheeled robot navigation based on an interpolated potential function and gradient," Robotics and Autonomous Systems, Vol. 177, ISSN 0921-8890, 2024.
9. P. Ogren, N. E. Leonard, "A convergent dynamic window approach to obstacle avoidance," IEEE Transactions on Robotics 21 (2), pp. 188–195, 2005.
10. 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, issue 37, pp. 573-578, 2022.
11. D. Cagigas and J. Abascal, "A Hierarchical Extension of the D* Algorithm," Journal of Intelligent and Robotic Systems, vol. 42, pp. 393–413, 2005.
12. H. Ryu, "Hierarchical Path-Planning for Mobile Robots Using a Skeletonization-Informed Rapidly Exploring Random Tree*," Applied Sciences 10, vol. 10, 2020.
13. 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.
14. Z. Zivkovic, B. Bakker and B. Krose, "Hierarchical map building and planning based on graph partitioning," Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA 2006.), pp. 803-809, Orlando, USA, 2006.
15. Z. Yijun, X. Jiadong and L. Chen, "A Fast Bi-Directional A* Algorithm Based on Quad-Tree Decomposition and Hierarchical Map," in IEEE Access, vol. 9, pp. 102877-102885, 2021.
16. J. Gregorić, M. Seder, I. Petrović, "Autonomous hierarchy creation for computationally feasible near-optimal path planning in large environments," Robotics and Autonomous Systems, vol. 172, 2024.
17. 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.
18. 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.
19. 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.
20. G. C. Goodwin, M. M. Seron, J. A. De Doná, "Constrained control and estimation: an optimisation approach," Springer Verlag, 2004.