

# Reward-field Guided Motion Planner for Navigation with Limited Sensing Range

Jan Bayer

Jan Faigl

**Abstract**—In this paper, we focus on improving planning efficiency for ground vehicles in navigation and exploration tasks where the environment is unknown or partially known, leading to frequent updates of the navigational goal as new sensory information is acquired. Asymptotically optimal motion planners like RRT\* or FMT\* can be used to plan the sequence of actions the robot can follow to achieve its current goal. Frequent replanning of the whole action sequence becomes computationally demanding when actions are not executed precisely because of limited information about the foreground terrain. The decoupled approach can decrease the computational burden with separated path planning and path following; however, it might lead to suboptimal solutions. Therefore, we propose a novel approach based on generating a reusable reward function that guides a fast sampling-based motion planner. The proposed method provides improved results in navigation scenarios compared to the former approaches, and it led to about 7% faster autonomous exploration than the decoupled approach. The present results support the suitability of the proposed method in navigation tasks with continuously updated navigation goals.

## I. INTRODUCTION

Efficient autonomous navigation of ground vehicles is essential in various application scenarios. The herein-studied approach is motivated by scenarios where only limited information about the foreground terrain is available, such as exploration missions or navigation, avoiding uncontrolled elements like pedestrians or other vehicles. In these scenarios, the whole vehicle’s trajectory cannot be precomputed in advance or finalized in a significant period before the end of the mission since the navigational goal or collision-free path is continuously changing. Thus, the robot’s trajectory needs to be adjusted each time the goal is changed, or the current trajectory becomes unfeasible.

Two main categories of trajectory planning approaches can be found in the literature. The first category contains motion planning, which directly plans a sequence of actions leading to the updated navigational goal [1]. Direct planning can be demanding, especially when the robot moves fast and there is high uncertainty in executing the action that requires frequent replanning. Therefore, the second approach is to decouple planning a sequence of directly executable actions into path planning and path following.

The authors are with the Faculty of Electrical Engineering, Czech Technical University, Technická 2, 166 27, Prague, Czechia. {bayerjal|faigljl}@fel.cvut.cz

The work has been supported by the Czech Science Foundation (GAČR) under research project No. 22-05762S and by the European Union under the project ROBOPROX - Robotics and advanced industrial production (reg. No. CZ.02.01.01/00/22.008/0004590).

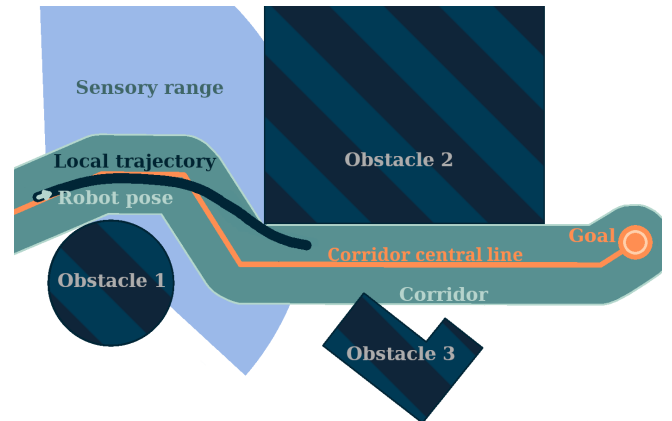


Fig. 1. The proposed reward-field guided motion planner uses local trajectory to predict the effects of the planned actions. The reward function generated during the first step of the algorithm covers the expected (and currently known) navigation corridor within the vehicle sensory range. Then, the reward function is used in motion planning to select the sequence of actions (local trajectory) directly steering the robot’s motion.

Decoupling is used in exploration [2], where a plan to the continuously updated goal location is planned first using a low-fidelity robot model. Then, a path-following algorithm is used with a high-fidelity robot model to generate a sequence of actions that steer the robot to follow the planned path as precisely as possible. Since the demanding high-fidelity model is employed only within the course of the planned path, the decoupled approach is less demanding than direct planning of the sequence.

In this paper, we argue that a decoupled approach with path planning and path following might lead to suboptimal performance, not to mention the necessity to tune the tradeoff between the precision and speed of the path following. Besides, we further studied the hypothesis that even the optimal sequence of actions for each navigational goal might not lead to the best possible performance when the navigational goals are changed during the course toward the final vehicle destination. We propose a novel motion planner to validate the hypotheses by empirical evaluation in navigation tasks with a real robotic vehicle.

We propose a novel planner that operates in two steps to test the raised hypotheses. First, a reward function covering the corridor between the current robot location and the given goal is generated. Then, the reward function guides the growth of a random-sampled tree of actions toward the goal, generating a local trajectory for the robot. The principle of the method is summarized in Fig. 1.

The method advantage is that the reward function and motion plan can be updated separately. It can save computa-

tional resources and allow frequent updates, especially when the action execution is not precise, and the local trajectory needs to be recomputed; the previously generated corridor can be reused. Besides, a relatively short planning horizon can be used in the motion planning step. Since the trajectory is recomputed in every planning iteration, it is unnecessary to reach the final goal by the local trajectory, which decreases the computational burden of the motion planning step.

The real performance of the proposed solution is compared with existing approaches, and the empirical results support the method's advantages. We consider the following main highlights of the proposed method.

- A novel randomized sampling-based motion planning method that allows fast recomputation of the robot trajectory and thus supports corrective actions when execution of the plan is not precise.
- The proposed method avoids decoupling path planning and path following to improve the efficiency of the mission execution.
- It balances greedy behavior when the given navigation goal is far and close-to-optimal behavior when the robot approaches the given goal. The results show that balancing is beneficial for autonomous exploration, where navigation waypoints are often changing as obstacles are continuously revealed during the mission.

The rest of the paper is organized as follows. The motion planners and path-following approaches utilized in relevant navigation scenarios and exploration missions are overviewed in Section II. The studied problem is specified in Section III. Because the planning requirements come from the exploration framework, it is summarized in Section IV. The proposed motion planner is described in Section V. Evaluation results from simulated and real-world experimental deployments are reported in Section VI. The concluding remarks are summarized in Section VII.

## II. RELATED WORK

The studied motion planning problem is motivated by autonomous robot navigation, where navigational goals are transformed to control actions applied to reach the goal location. The most straightforward approach is to use motion planning, which directly plans the robot's motion in its action space. Lattice-based planning methods use explicitly discretized representation [3], such as state lattice [4] to generate feasible smooth paths, including lane changes and merging between vehicles [5], such as [4], to generate feasible smooth paths for a rover. Randomized sampling-based methods such as RRT and RRT\* use geometrical and vehicle dynamics models to directly generate a sequence of actions by applying possible actions and simulating the vehicle motion to grow a roadmap to reach the goal location [6].

Although improved methods have been proposed to address the high computational requirements of the sampling-based methods, such as FMT\* [7] and BIT\* [8], the methods are suitable for systems where the planned sequence of actions can be reliably executed [8]. It is because the trajectory needs to be replanned whenever an action is not

executed with sufficient precision or its effect differs from the expectations. It might happen for ground robots, specifically for tracked and skid-steer platforms, where the effects of the executed actions strongly depend on the surface on which the vehicle moves [9]. Hence, frequent sequence replanning is needed, which can be very demanding for long sequences or complex vehicle models.

Contrary to the methods mentioned above, decoupling methods split the problem into path planning with a simplistic robot model and path following (tracking) that allows fast replanning of long sequences of actions or quick updates of the actions, which is beneficial for fast-moving vehicles. Decoupled collision-free planning and path following are reported to be employed in autonomous driving systems [10], [11]. Such locally planned paths can be tracked by control techniques designed for moderate driving conditions with kinematic models, such as the Single-Track (ST) model (bicycle model) [12]. However, these kinematic models assume that wheels do not slip.

The Pure Pursuit controller [13] is one of the earliest approaches built on the simple kinematic model that has been deployed in various applications, including the DARPA Grand Challenge [14] and the DARPA Urban Challenge [15]. Improved controller performance is reported for methods that use the front or rear wheel position for generating the control feedback, specifically when the curvature of the reference tracked path is non-zero [16]. Nevertheless, the Stanley lateral controller with front wheel feedback is used by the DARPA Grand Challenge winning team [17], and its deployments have also been reported recently in [18].

The decoupled approaches are also used by the teams within DARPA Subterranean Challenge (SubT) [19], where the navigation is performed in unknown environments as a part of the exploration mission, thus, with frequent replanning. The path following methods used in SubT include the Model Predictive Control (MPC) [20] frameworks, where the optimal sequence of control actions is planned for a relatively short time horizon. In MPC-based control, the next best sequence of actions is planned from which the first action is applied; then, the sequence is recomputed again to obtain the next action. MPC-based following of local plans has been successfully deployed by the CoSTAR team [2] for legged robot navigation in uneven terrains. Path following with a look-ahead point at the fixed distance from the robot has been used by the teams MARBLE [21] and CTU-CRAS-NORLAB [22]. In that approach, velocities applied to the robot control are generated, so the heading is steered to the look-ahead point.

The decoupled method is also reported to be used in autonomous racing, where the raceline can be determined in advance of the race [23] using optimization techniques to minimize time [24], energy [25], or maximize velocity [26]. Local planning or tracking of the raceline for a horizon with a fixed length is then used for navigation during the racing. Graph-based local planning with precomputed action primitives to avoid obstacles is used in [27], and the usage of RRT\* is reported in [28].

Although the decoupled approach is prevalent in the reported deployments, including autonomous driving systems, we argue it is possible to couple path planning and path following to opt for increasing the effectiveness of navigation while keeping the advantages of the decoupled method. Based on the literature review, we selected representative methods from both approaches for comparison with the proposed method. Direct methods are represented by the **RRT\*** motion planner selected to compare the performance with the asymptotically optimal sampling-based method that directly determines the sequence of actions. The method denoted as the **Decoupled approach** in the reported results is represented by an A\* path planner used for path generation using a simplified robot model combined with a look-ahead follower. In particular, we selected the method of steering the robot to the look-ahead point used in [21], and [22].

### III. PROBLEM SPECIFICATION

The addressed navigation task is to drive a robot from its current pose  $\mathbf{s}_t$  to the given goal location  $\mathbf{q}$  characterized by the position in a plane  $(x, y) \in \mathbb{R}^2$ . The robot configuration (pose)  $\mathbf{s}_t = \{\mathbf{x}_t, \theta_t\}$  at the time instant  $t$  is characterized by the robot pose consisting of the robot position  $\mathbf{x}_t = (x, y)$  in a plane  $\mathbf{x}_t \in \mathbb{R}^2$ , and its orientation  $\theta_t \in \mathbb{S}$ . The robot is controlled by a sequence of actions  $\mathbf{u}_i$  from the robot action space  $\mathcal{A}$ , where each  $i$ -th action is a pair of the magnitude of forward  $v_{fw}$  and angular  $v_{ang}$  velocities. The planning problem is to determine a sequence of actions to reach the goal  $\mathbf{q}$  from  $\mathbf{s}_t$  in the shortest time possible.

It is assumed that for motion planning, a traversability map capturing obstacles is available for every time instance  $t$ , such that for a given robot configuration  $\mathbf{s}_t$ , it returns a binary value reflecting if the robot can reach the location corresponding to  $\mathbf{s}_t$ .

### IV. AUTONOMOUS EXPLORATION

The primary motivation deployment scenarios of the studied motion planning problem are exploration missions, where the navigational goal changes based on the mission's progress and where other algorithms may use computational resources saved by navigation to improve the overall performance. We consider the autonomous exploration scenario described in [29], where the robot starts with no information about the environment, then the robot continuously identifies new exploration waypoints, towards which the robot is navigated to collect information about the environment by its sensors. In [29], the exploration mission ends after the robot explores the whole environment.

We provide a summary of the exploration framework [30] used in Section VI, for which we aim to improve its navigation performance by the proposed motion planner. The architecture of the framework is depicted in Fig. 2. The 3D LiDAR captures scans at 10 Hz, which are used for the SLAM-based localization [31], leading to a pose update at 10 Hz. Aligned 3D scans are used to build the 3D map. The map is then used for traversability assessment based on the terrain slope and maximum step height [30]. Since the range

of the LiDAR is 35 m, the untraversable areas (obstacles) and new exploration waypoints are mostly identified from a distance above 5 m. For the robot, it would take at least 5 s to reach places at such distance; thus, traversability is updated at 0.5 Hz to save onboard computational resources. The rate of updating the next navigational waypoint depends on how long it takes the robot to reach its current waypoint, which corresponds to 0.05–1 Hz for the testing environment reported in Section VI.

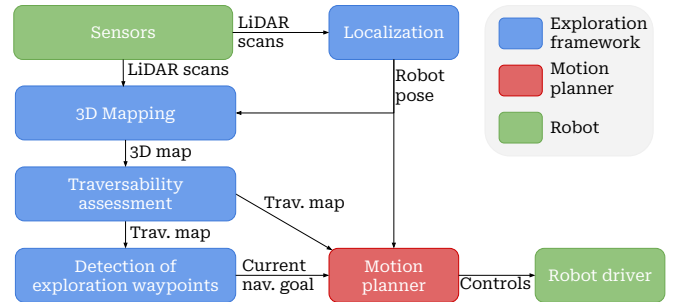


Fig. 2. The architecture of the exploration framework into which the motion planner is integrated.

The employed motion planner takes the following inputs with the listed expected update rates.

- The robot's recent pose is updated at 10 Hz.
- The traversability map (trav. map) is updated at 0.5 Hz.
- The current navigational goal is updated at 0.05–1 Hz.

The motion planner output is a sequence of control commands (actions) that are applied to steer the robot until a new sequence is generated or the final goal is reached.

Regarding the exploration mission, the key performance indicator is the time needed to explore a specific area. Thus, the proposed motion planner's impact on exploration performance is measured as the time needed to explore a given area compared to the reference (baseline) motion planners. Similarly, the time needed to finish the mission is used to compare the methods in the rest of the navigation scenarios presented in Section VI.

### V. PROPOSED MOTION PLANNER

The proposed motion planner is composed of two parts. The purpose of the first part is to compute a reward function used in the second part to guide a randomized sampling-based trajectory space search. Both parts can run separately, which saves computational resources by recomputing only the trajectory or reward function when needed. The planner structure is depicted in Fig. 3.

#### A. Reward Function Generation

The reward function  $r(\mathbf{x}, \mathbf{P})$  is the sum of functions  $r_1$  and  $r_2$ .  $r_1$  rewards the location  $\mathbf{x} \in \mathbb{R}^2$  reached by the motion planner based on how far it can get within the corridor.  $r_2$  rewards the reached location based on how close it is to the current navigational goal.  $\mathbf{P}$  represents a central corridor line in which the trajectory toward  $\mathbf{q}$  should be found. It is a sequence of 2D positions  $\mathbf{P} = \{\mathbf{p}_i; i = 1, 2, \dots, n; \mathbf{p}_n = \mathbf{q}\}$

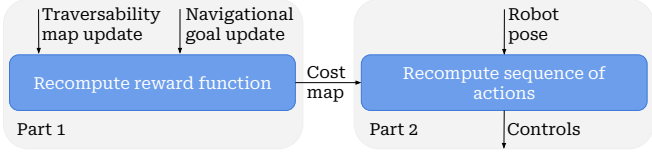


Fig. 3. Two parts of the proposed motion planner with the input triggers to recompute the reward function and sequence of actions. The robot pose is updated more frequently than the map and goal (see Section IV), and thus, the reward function is recomputed at a lower rate than the motion planning.

that can be computed in a grid map of the environment. The reward is calculated as

$$r(\mathbf{x}, \mathbf{P}) = r_1(\mathbf{x}, \mathbf{P}) + r_2(\mathbf{x}, \mathbf{P}), \quad (1)$$

where

$$r_1(\mathbf{x}, \mathbf{P}) = \begin{cases} \max\{i | d(\mathbf{p}_i) < c_{rad}\} & \text{if } \exists \mathbf{p}_i \text{ such that} \\ & d(\mathbf{p}_i) < c_{rad} \\ 0 & \text{otherwise} \end{cases}$$

$$d(\mathbf{p}_i) = \|\mathbf{x} - \mathbf{p}_i\|,$$

$$r_2(\mathbf{x}, \mathbf{P}) = \begin{cases} \frac{c_{rad} - \|\mathbf{x} - \mathbf{p}_n\|}{p_{dist}} & \text{if } \|\mathbf{x} - \mathbf{p}_n\| < c_{rad} \\ 0 & \text{otherwise} \end{cases}, \quad (2)$$

$c_{rad}$  is the corridor radius where the reward is precomputed, and  $p_{dist}$  is an average distance of consecutive positions  $\|\mathbf{p}_i - \mathbf{p}_{i+1}\|$ , obtained during corridor central line determination. The distance  $\|\mathbf{x} - \mathbf{p}_j\|$ ,  $\mathbf{p}_j \in \mathbf{P}$  is Euclidean distance of the robot position  $\mathbf{x}$  of  $\mathbf{s} = \{\mathbf{x}, \theta\}$  and a position on the central line  $\mathbf{P}$ .

A fast planner, such as A\*, generates the corridor central line using a simplistic kinematic robot model. Its computational requirements are much lower than trajectory planning in the robot action space with complex robot kinematics and dynamics constraints. The central line of the corridor is planned as a conservative collision-free path using grid-based planning for a point robot and grown obstacles with maximization of the path distance from obstacles, which can define the corridor; hence, the name corridor central line.

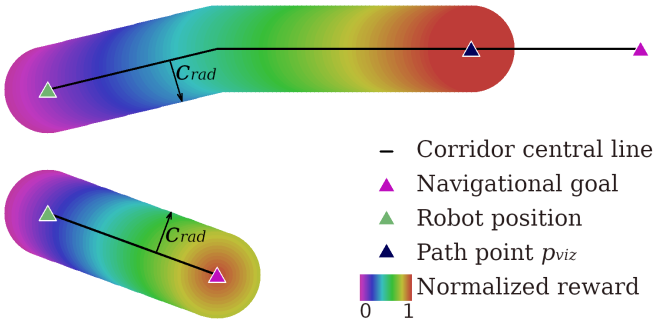


Fig. 4. Visualization of the reward function  $r(\mathbf{x}, \mathbf{P})$  for two different goal locations. The upper case visualizes the sampled reward function for the navigational goal outside the sensory range. In that case,  $\mathbf{p}_{viz}$  denotes the last point of the sensory range at a simplified feasible path (a part of the corridor central line) to the goal location. In the lower example, the navigational goal is within the sensory range of the robot. The value of  $r(\mathbf{x}, \mathbf{P})$  is normalized independently for each shown case.

Although the obstacles can define the corridor itself, in the present work, the corridor is further restricted by overlaying disks centered at the grid-based positions of  $\mathbf{P}$  and the radius  $c_{rad}$  to guide the search during the motion planning. The reward function is computed only for the part of the corridor defined by the current robot position and the farthest point of the central line within the robot's sensory range. The farthest point is denoted  $\mathbf{p}_{viz}$ , but only if the farthest point in the range is not the goal  $\mathbf{q}$ . It is because the central line and the reward function are recomputed whenever the robot reveals new obstacles (the traversability map is updated). Examples of the reward function are visualized in Fig. 4.

### B. Randomized Sampling-based Motion Planning

The motion planner is employed to find a sequence of  $m$  feasible actions (controls)  $\{\mathbf{u}_i; i = 0, 1, \dots, m; \mathbf{u}_i \in \mathcal{A}\}$  in the robot action space  $\mathcal{A}$  that maximizes the reward  $r(\mathbf{x}, \mathbf{P})$  along the sequence

$$\{\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_m\} = \arg \max_{\{\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_m\}} r(\mathcal{M}(\mathbf{s}_t, \mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_m), \mathbf{P}). \quad (3)$$

The robot model  $\mathcal{M}$  predicts the position reached by the

---

#### Algorithm 1: Compute the Sequence of Actions.

---

**Input:**  $\mathbf{P}$  – Precomputed central line,  $\mathcal{M}$  – Robot model,  $\mathbf{s}_t = \{\mathbf{x}_t, \theta_t\}$  – Initial robot pose,  $r(\mathbf{x}, \mathbf{P})$  – Sampled reward function.

**Parameters:**  $c_{rad}$  – Half of the corridor width;  $f_r$  – Output update frequency;

**Output:**  $\{\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_m\}$  – Action sequence.

---

```

1  $g_0 \leftarrow \text{create\_init\_vertex}(\mathbf{s}_t)$ 
2  $G \leftarrow \text{init\_graph\_by\_first\_vertex}(g_0)$ 
3 while until_elapsed( $1/f_r$ ) do
4    $\mathbf{s} = \{\mathbf{x}, \theta\} \leftarrow \text{sample\_within\_corridor}(\mathbf{P}, c_{rad})$ 
5    $g \leftarrow \text{get\_closest\_vert}(G, \mathbf{s})$  // Considering  $\theta$ .
6    $\mathbf{u}' \leftarrow \text{generate\_action}(g, \mathcal{M})$ 
7    $\mathbf{s}_{end} = \{\mathbf{x}_{end}, \theta_{end}\} \leftarrow$ 
8      $\leftarrow \text{simulate\_action}(\mathcal{M}, g.state, \mathbf{u}')$ 
9   if is_collision_free( $\mathbf{s}_{end}$ ) then
10     $r_s \leftarrow r(\mathbf{x}_{end}, \mathbf{P})$ 
11     $g_{new} \leftarrow \text{create\_new\_vertex}(\mathbf{s}_{end}, \mathbf{u}', r_s)$ 
12     $g_{close} \leftarrow \text{get\_closest\_vert}(G, g_{new})$ 
13    if considered_equal( $g_{close}.state, \mathbf{s}_{end}$ ) then
14      if  $g_{close}.elapsed > g.elapsed + 1/f_r$ 
15        then
16           $G \leftarrow \text{replace\_verts}(G, g_{close}, g_{new})$ 
17    else
18       $G \leftarrow \text{merge\_with\_vert}(G, g_{new})$ 
19 return backtrack_action_sequence( $g'$ )

```

---

robot when it applies a given sequence of actions  $\mathbf{u}_i$  while starting at the configuration (pose)  $\mathbf{s}_t$ . The proposed solution

of (3) is based on randomized sampling-based motion planning [32]. The proposed algorithm builds a search graph  $G$  using random sampling, summarized in Algorithm 1. Each vertex of  $G$  has the following associated data fields.

- *parent* – A predecessor of the vertex within the graph.
- *elapsed* – Time required to execute the sequence, including the preceding vertices.
- *state* – Predicted robot state  $s_j$  at the vertex.
- *action* – Control action associated with the vertex.

The reward (1) is precomputed and available when Algorithm 1 is running, which operates as follows.

The graph  $G$  is initialized by the current robot state  $s_t$ , and the algorithm proceeds with the graph (tree) expansion. At each iteration, a random sample within the corridor is sampled with the preference according to the reward function. Then, the closest vertex  $g \in G$  to the randomly generated sample is found, also considering the orientation of the robot  $\theta$ . Next, the control input  $\mathbf{u}' \in \mathbf{A}$  is generated toward the random sample. The robot model  $\mathcal{M}$  is used to predict the next robot state  $s_{end}$  when applying  $\mathbf{u}'$  from the state corresponding to  $g$ . If the corresponding motion of the robot from  $g$  to  $s_{end}$  by applying  $\mathbf{u}'$  is collision-free, a new vertex is merged with  $G$  (Lines 12–17 of Algorithm 1).

In our case, the algorithm is terminated after a given timeout defined according to the robot pose update rate detailed in Section IV,  $f_r = 10$  Hz. The resulting sequence is obtained by backtracking the vertex  $g \in G$  associated with the most rewarding vertex.

## VI. RESULTS

The proposed motion planner has been empirically evaluated in real-time simulations and real-world deployments with the skid-steer robot Husky by Clearpath Robotics, steered by actions composed of forward and angular velocity. Its performance has been compared within the navigation scenario with the RRT\* motion planner and the decoupled approach used in [21], [22]. Due to the computational requirements of the RRT\*, only the decoupled approach is deployed in the exploration scenario. The parameterization of the robot and all the methods are identical for the simulation and real-world scenarios, and they are summarized in Table I. The presented results are structured to simulation navigation scenarios reported in Section VI-A, real-world navigation scenarios (Section VI-B), real-world exploration scenarios (Section VI-C), and implementation comments in Section VI-D.

### A. Simulation Racetrack Navigation Scenarios

Four navigation scenarios S1–S4, with increasing difficulty, have been created, as shown in Fig. 5. In each scenario, the robot’s task is to finish the racetrack as quickly as possible. The provided navigational goal to the methods is always at the central line of the racetrack. The robot is simulated using a kinematic unicycle model, where applied forward and angular velocities are affected by 5% simulation random error drawn from a normal distribution. Ten trials have been performed for each scenario and each method.

TABLE I  
ROBOT AND MOTION PLANNERS PARAMETERS

Parameter	Value
Update frequency of robot controls $f_r$ [Hz]	10.0
Robot forward velocity range [ $\text{m s}^{-1}$ ]	(−1.0, 1.0)
Robot angular velocity range [ $^\circ \text{s}^{-1}$ ]	(−40.0, 40.0)
<i>RRT* motion planner</i>	
Maximum goal distance on racetrack $d_{sense}$ [m]	6.0
<i>Decoupled approach – Look-ahead follower</i>	
Next waypoint minimal distance [m]	0.8
Forward velocity proportional multiplier [-]	3.0
Angular velocity proportional multiplier [-]	1.1
Only turning angular displacement [ $^\circ$ ]	35
<i>Proposed method</i>	
Sensory range $d_{sense}$ [m]	6.0
Corridor radius $c_{rad}$ [m]	1.0
Average distance $p_{dist}$ [cm]	15.0

Average values with the standard deviations of the time to complete the racetrack are listed in Table II.

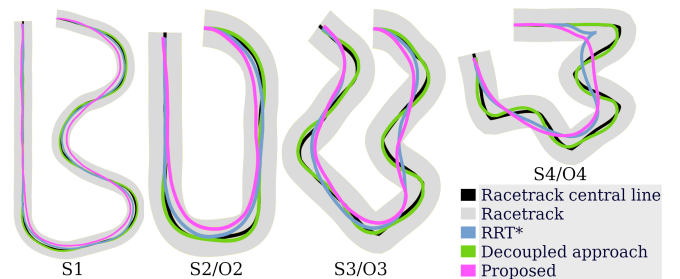


Fig. 5. Racetrack navigation scenarios used in the simulated (S1–S4) and real-world (O2–O4) deployments. The width of the racetrack (shown in gray) is 2 m in all cases. The shown curves denote the paths performed by the robot in the simulations. The space outside the racetrack is untraversable. The RRT\* and the proposed motion planners exploit the width of the racetracks as indicated by the shortcuts. In S4, the RRT\* switched the direction of the robot motion to follow the reference path fast and still be within the racetrack.

TABLE II  
TIME TO COMPLETE SIMULATION RACETRACK NAVIGATION SCENARIOS

Motion planning method	Time to finish the racetrack [s]			
	S1	S2	S3	S4
RRT*	74.7±0.6	28.5±0.4	30.7±0.4	26.8±1.1
Decoupled approach	71.4±0.1	27.3±0.1	34.7±0.1	32.7±0.1
Proposed method	<b>69.0±0.3</b>	<b>25.9±0.4</b>	<b>29.2±0.5</b>	<b>24.9±1.1</b>

Results in Table II show that the proposed method yields the fastest motion in all scenarios. The gap between the proposed method and reference methods increases with the increasing difficulty of the scenarios. The RRT\* and proposed planner exploit the corridor width to follow the path fast yet still within the corridor. The decoupled method maintains relatively high precision of path following, which leads to worse performance in S3 and S4. Although the same path planner is used in the decoupled approach and for generating the proposed method’s corridor central line, the proposed planner exploits the whole corridor, which results in better performance than the decoupled approach. In S1 and S2, the decoupled approach utilizes the maximum speed of the vehicle during the whole scenario and steers only

the heading, which leads to slightly better performance than RRT\*. On the other hand, the decoupled approach performed worse than the RRT\* in more complex scenarios, which is the effect of the decoupled method’s tradeoff between the path following precision and time to reach the path endpoint. The proposed method utilized the maximum speed of the vehicle during most of the S1 and S2 scenarios. Although it slowed down several times, it performed better in S1 and S2 than the decoupled approach because of the shortcuts.

### B. Real-world Racetrack Navigation Scenarios

The real skid-steer robot Husky has been used for experimental evaluation in one indoor scenario I1, depicted in Fig. 6, and three outdoor scenarios O2–O4 that follow the simulation tracks, see Fig. 5. Each method has been deployed twice in each experimental scenario. Table III shows the time



Fig. 6. Environments used for the skid-steer robot deployments. The indoor scenario I1 with a marked center of the racetrack is depicted on the left. The outdoor environment (right) was combined with racetracks O2–O4 that implement the simulation scenarios S2–S4 shown in Fig. 5.

required by each method to reach the end of the racetrack.

TABLE III

TIME TO COMPLETE REAL-WORLD RACETRACK NAVIGATION SCENARIOS

Motion planning method	Time to finish the racetrack [s]			
	I1	O2	O3	O4
RRT*	19.7±0.2	35.7±0.3	39.4±0.4	32.8±0.1
Decoupled approach	20.3±0.4	36.7±0.1	49.3±0.8	47.4±0.1
Proposed method	<b>17.9±0.1</b>	<b>33.5±0.1</b>	<b>37.7±0.8</b>	<b>28.7±0.5</b>

From Table III, it can be seen that in all outdoor scenarios O2–O4, the worst performance is provided by the decoupled method and the best by the proposed method. In the indoor scenario, the difference between the performance of RRT\* and the decoupled approach is insignificant, but the proposed method was still the fastest. Regarding the results for simulation scenarios S2–S4 and the corresponding outdoor scenarios O2–O4, which use identical racetracks, the time to finish the track is significantly longer in real-world deployment. It is primarily caused by the more slippery surface in the outdoor area, partially covered by gravel.

The precision of action execution on the gravel significantly affects the performance of the decoupled approach. The correcting actions of the decoupled approach were executed later than for the RRT\* and proposed planners, which led to worse performance. It is even noticeable for O2 corresponding to S2, where the decoupled approach performed worse than the RRT\*. The set tradeoff between the speed and precision of the path following causes the

“delayed” corrective actions of the decoupled approach in outdoor scenarios. Suppose the tradeoff is set towards faster corrections of the heading and position. In that case, it will lead to a bit slower speed and shorter trajectory, which would perform better in outdoor scenarios. However, such settings would cause worse performance in indoor scenario I1 and real-world exploration.

### C. Real-world Exploration Scenario

Since the original motivation for the proposed planner was raised from the exploration, we further deploy the motion planners in exploring the indoor environment with the framework described in Section IV. The navigational goals are frequently updated, so only the decoupled and proposed methods are examined. The exploration area is shown in Fig. 7, and the paths taken by the robot during the exploration are depicted in Fig. 8. The resulting exploration times and traveled distances are summarized as average values in Table IV.



Fig. 7. Snapshots from the indoor exploration environment, which is 30 m × 35 m in size.

TABLE IV

MOTION PLANNERS PERFORMANCE IN EXPLORATION SCENARIO

Motion planning method	Average exploration time [s]	Average trajectory length [m]
Decoupled approach	160.4	108.6
Proposed method	149.7	106.8

From the resulting paths depicted in Fig. 8, we can see that the proposed method provides smoother paths than the decoupled approach. The advantage of the proposed method is that planning with a few seconds long horizon rarely turns the robot with zero turn radius. When the robot is close to a given goal, using the proposed method, the robot plans its motion, targeting the exact goal location. However, if a goal is far from the robot’s location, the proposed method tends to move the robot greedily toward the end of the corridor, not targeting the exact goal location, which is the effect of the proposed reward function. The robot benefits from such behavior when the goals are often changing since the robot is not sacrificing fast actions in order to reach each goal precisely. The proposed method exhibits noticeably faster exploration and slightly shorter traveled distance than the decoupled approach, as indicated in Table IV.

The parameterization of the decoupled approach can be considered its weak point. A proper setting of the path follower with respect to tradeoff the path following precision with the robot speed toward the path end is tricky to tune, especially in exploration, where it is not known when the

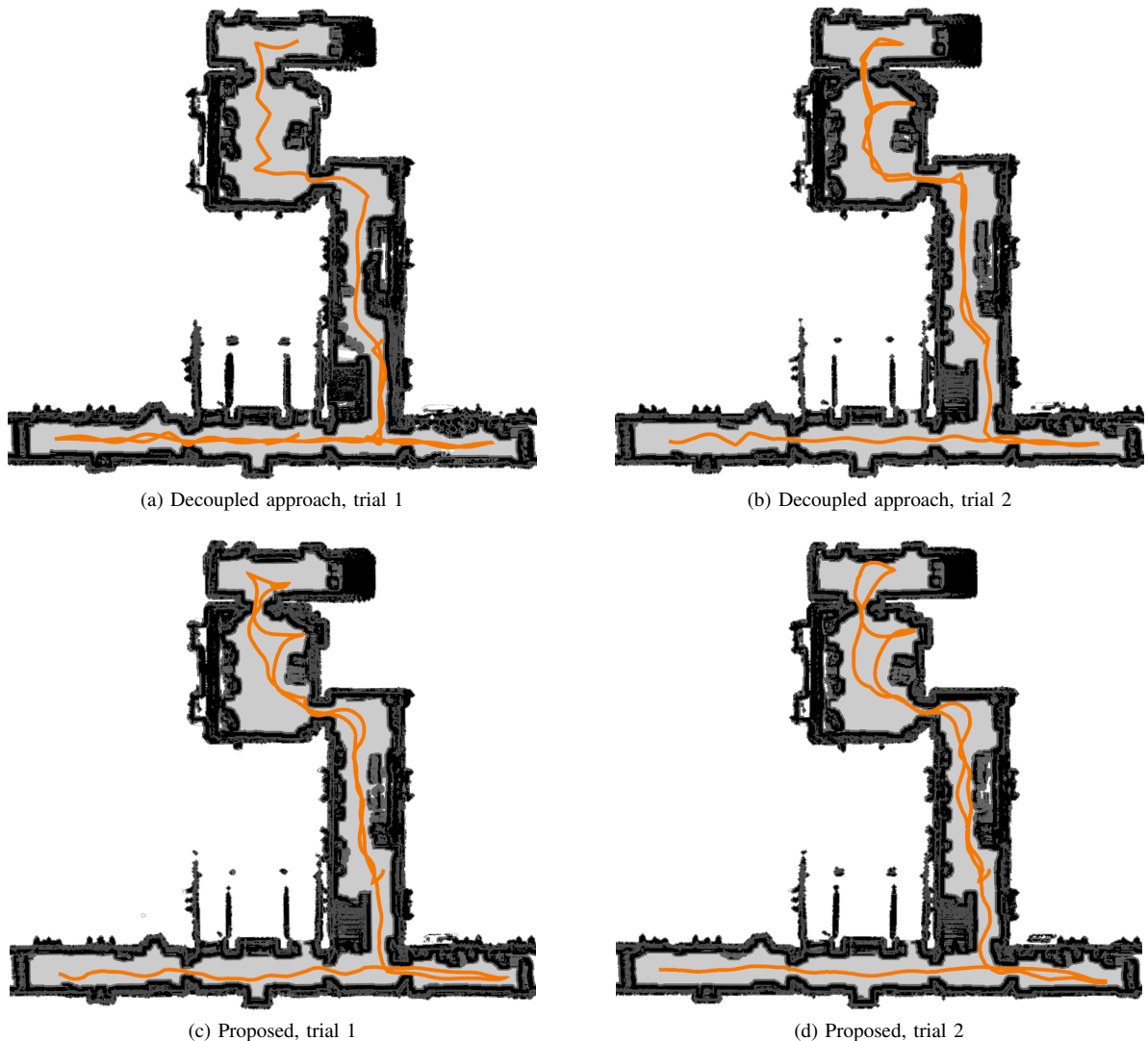


Fig. 8. The robot traveled paths in two trials for each tested method in the exploration scenario. The paths are marked by orange, and dark colors display the identified untraversable areas.

navigational goal changes or how close the robot would have to be to the obstacles to uncover the whole environment. The proposed method avoids such a tradeoff setting by detecting possible collisions during the motion planning part.

#### D. Implementation Notes

The proposed method is targeted to be applied within a complex and computationally demanding system for autonomous mobile robot operation. Therefore, it is implemented as a single thread, saving the resources of multi-core systems for localization, mapping, and high-level autonomy. The deployments on the real system have been done on the Intel i7-8565U CPU.

The collision detection in the motion planning part of the proposed method has been implemented by indicating obstacles as negative reward values. Thus, dynamic obstacles in the environment would decrease the benefits of reusing the precomputed reward function. Besides, we use relatively sparse discretization of the forward velocity in the action space, which yields a relatively high probability of selecting

the maximum robot velocity when possible. Note that even though the proposed method is shown for generating forward and angular velocity to steer the robot, the underlying principles are also applicable to a model with different state vectors and control inputs.

#### VII. CONCLUSION

The proposed motion planner is motivated by increasing the performance of decoupled approaches in scenarios with changing navigational goals by coupling the trajectory planning with a more explicit consideration of the space around the path being followed. The main novelty of the proposed approach is the utilization of the reward function in motion planning. The proposed method couples local motion planning and path following into a single method while still allowing the recomputation of the reward function and motion planning separately, which supports motion planning at high rates. The reported results support the feasibility of the proposed method, which outperforms direct motion planning based on the RRT\* and the decoupled approach.

The proposed method avoids tuning the tradeoff between the performance and precision of path following needed in decoupled approaches. Based on the experimental deployment, the proposed method is viable and provides a feasible solution exploiting the motion capabilities of the platform. Furthermore, it improves performance in autonomous exploration, which was the main original motivation of the presented work.

#### ACKNOWLEDGEMENTS

We would like to thank Anastázia Rišková for her technical and development support.

#### REFERENCES

- [1] C.-b. Moon and W. Chung, “Kinodynamic planner dual-tree rrt (dt-rrt) for two-wheeled mobile robots using the rapidly exploring random tree,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 2, pp. 1080–1090, 2014.
- [2] A. Agha, K. O. B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund, M. F. Ginting, K. Ebadi, M. Anderson, T. Pailevanian, E. Terry, M. Wolf, A. Tagliabue, T. S. Vaquero, M. Paliéri, S. Tepsuporn, Y. Chang, A. Kalantari, F. Chavez, B. Lopez, N. Funabiki, G. Miles, T. Touma, A. Buscicchio, J. Tordesillas, N. Alatur, J. Nash, W. Walsh, S. Jung, H. Lee, C. Kanellakis, J. Mayo, S. Harper, M. Kaufmann, A. Dixit, G. J. Correa, C. Lee, J. Gao, G. Merewether, J. Maldonado-Contreras, G. Salhotra, M. S. D. Silva, B. Ramtoula, S. Fakoorian, A. Hatteland, T. Kim, T. Bartlett, A. Stephens, L. Kim, C. Bergh, E. Heiden, T. Lew, A. Cauligi, T. Heywood, A. Kramer, H. A. Leopold, H. Melikyan, H. C. Choi, S. Daftry, O. Toupet, I. Wee, A. Thakur, M. Feras, G. Beltrame, G. Nikolakopoulos, D. Shim, L. Carlone, and J. Burdick, “NeBula: Team CoSTAR’s robotic autonomy solution that won phase ii of darpa subterranean challenge,” *Field Robotics*, vol. 2, pp. 1432–1506, 2022.
- [3] A. Botros, “Lattice-based motion planning with optimal motion primitives,” Ph.D. dissertation, University of Waterloo, 2021.
- [4] M. Pivtoraiko, R. A. Knepper, and A. Kelly, “Differentially constrained mobile robot motion planning in state lattices,” *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.
- [5] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, “Motion planning for autonomous driving with a conformal spatiotemporal lattice,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 4889–4895.
- [6] D. González, J. Pérez, V. Milanés, and F. Nashashibi, “A review of motion planning techniques for automated vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.
- [7] L. Janson, E. Schmerling, A. Clark, and M. Pavone, “Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions,” *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.
- [8] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Batch informed trees (bit\*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3067–3074.
- [9] V. Rajagopalan, Ç. Meriçli, and A. Kelly, “Slip-aware model predictive optimal control for path following,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4585–4590.
- [10] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz, et al., “Self-driving cars: A survey,” *Expert Systems with Applications*, vol. 165, p. 113816, 2021.
- [11] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, “A survey of autonomous driving: Common practices and emerging technologies,” *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.
- [12] A. D. Luca, G. Oriolo, and C. Samson, *Feedback control of a nonholonomic car-like robot*. Springer, 1998, pp. 171–253.
- [13] R. S. Wallace, A. Stentz, C. E. Thorpe, H. P. Moravec, W. Whittaker, and T. Kanade, “First results in robot road-following,” in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 1985, pp. 1089–1095.
- [14] M. Buehler, K. Iagnemma, and S. Singh, Eds., *The 2005 DARPA grand challenge: the great robot race*. Springer, 2007, vol. 36.
- [15] —, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer, 2009, vol. 56.
- [16] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.
- [17] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, “Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing,” in *American Control Conference*, 2007, pp. 2296–2301.
- [18] A. AbdElmoniem, A. Osama, M. Abdelaziz, and S. A. Maged, “A path-tracking algorithm using predictive stanley lateral controller,” *International Journal of Advanced Robotic Systems*, vol. 17, no. 6, 2020.
- [19] V. Orekhov, A. Maio, R. Daniel, and T. Chung, “Inspiring field robotics advances through the design of the darpa subterranean challenge,” *Field Robotics*, vol. 3, p. 560–604, 2022.
- [20] C. E. Garcia, D. M. Prett, and M. Morari, “Model predictive control: Theory and practice—a survey,” *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.
- [21] M. Ohradzansky, E. Rush, D. Riley, A. Mills, S. Ahmad, S. McGuire, H. Biggie, K. Harlow, M. Miles, E. Frew, C. Heckman, and J. Humbert, “Multi-agent autonomy: Advancements and challenges in subterranean exploration,” *Field Robotics*, pp. 1068–1104, 2022.
- [22] T. Rouček, M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalanský, T. Azayev, D. Heřt, M. Petrlička, T. Báča, V. Spurný, V. Krátký, P. Petráček, D. Baril, M. Vaidis, V. Kubelka, F. Pomerleau, J. Faigl, K. Zimmermann, M. Saska, T. Svoboda, and T. Krajník, “System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA subterranean challenge,” *Field Robotics*, vol. 2, pp. 1779–1818, 2022.
- [23] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, “Autonomous vehicles on the edge: A survey on autonomous vehicle racing,” *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022.
- [24] J. L. Vázquez, M. Brühlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, “Optimization-based hierarchical motion planning for autonomous racing,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2397–2403.
- [25] T. Herrmann, F. Christ, J. Betz, and M. Lienkamp, “Energy management strategy for an autonomous electric racecar using optimal control,” in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 720–725.
- [26] T. Herrmann, A. Wischnewski, L. Hermansdorfer, J. Betz, and M. Lienkamp, “Real-time adaptive velocity optimization for autonomous electric cars at the limits of handling,” *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 4, pp. 665–677, 2020.
- [27] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, “Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios,” in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3149–3154.
- [28] J. J. hwan, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma, “Optimal motion planning with the half-car dynamical model for autonomous high-speed driving,” in *American Control Conference*, 2013, pp. 188–193.
- [29] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA’97. Towards New Computational Principles for Robotics and Automation*, 1997, pp. 146–151.
- [30] J. Bayer, P. Čížek, and J. Faigl, “Autonomous multi-robot exploration with ground vehicles in darpa subterranean challenge finals,” *Field Robotics*, vol. 3, pp. 266–300, 2023.
- [31] S.-P. Deschênes, D. Baril, V. Kubelka, P. Giguère, and F. Pomerleau, “Lidar scan registration robust to extreme motions,” in *Conference on Robots and Vision (CRV)*, 2021, pp. 17–24.
- [32] I. Noreen, A. Khan, and Z. Habib, “Optimal path planning using rrt\* based approaches: a survey and future directions,” *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016.