Enabling robots to autonomously search dynamic cluttered post-disaster environments

Rado, K.; Baglioni, M.; Jamshidnejad, A.

DOI
10.1038/s41598-025-18573-y

Publication date
2025

Document Version
Final published version

Published in
Scientific Reports

scientific reports

OPEN

Enabling robots to autonomously search dynamic cluttered post-disaster environments

Karlo Rado, Mirko Baglioni✉ & Anahita Jamshidnejad

Robots will bring Search and Rescue (SaR) in disaster response to another level, in case they can autonomously take over dangerous SaR tasks from humans. A main challenge for autonomous SaR robots is to safely navigate in cluttered environments with uncertainties, while avoiding static and moving obstacles. We propose an integrated control framework for SaR robots in dynamic, uncertain environments, including a computationally efficient heuristic motion planning system that provides a nominal (assuming there are no uncertainties) collision-free trajectory for SaR robots and a robust motion tracking system that steers the robot to track this reference trajectory, taking into account the impact of uncertainties. The control architecture guarantees a balanced trade-off among various SaR objectives, while handling the hard constraints, including safety. The results of various computer-based simulations, presented in this paper, showed significant out-performance (of up to 42.3%) of the proposed integrated control architecture compared to two commonly used state-of-the-art methods (Rapidly-exploring Random Tree and Artificial Potential Function) in reaching targets (e.g., trapped victims in SaR) safely, collision-free, and in the shortest possible time.

Keywords Autonomous motion-planning, Dynamic obstacle avoidance, Robust model predictive control, Search and rescue robotics

Autonomous Search and Rescue (SaR) robots are emerging in post-disaster response, in order to reduce the exposure of human SaR staff to life-threatening risks. Structural damages and collapsed buildings reshape an environment post disaster. Thus, reliable and effective control methods for autonomous, safe navigation in dynamic and (partially) unknown environments are crucial for SaR robots[1–6]. Another main challenge for autonomous navigation of SaR robots is avoiding the obstacles that move according to generally nonlinear trajectories, without (significantly) compromising the mission criteria. State-of-the-art solutions either rely on assuming static obstacles only, or provide ad-hoc solutions that cannot be generalized or provide guarantees on the performance of SaR missions via robots[7–9].

To tackle these challenges, we propose a novel control architecture for autonomous SaR robots that should reach known target positions in dynamic cluttered post-disaster environments: We first divide the mission planning of SaR robots into two stages, motion planning and optimal motion tracking. We extend the greedy heuristic path planning method[10] that has been introduced to generate a nominal obstacle-free path towards the targets of the robot, in order to account for dynamic obstacles that may appear on the way of the robot. Note that by nominal, we are referring to a trajectory that is determined assuming there are no uncertainties or uncontrollable disturbances. We then integrate this path planning method with an optimal motion tracking system that closely follows the nominal trajectory estimated by the heuristic motion planning system while guaranteeing robustness to bounded uncontrollable disturbances.

The optimal motion tracking system is based on a robust version of MPC, called Tube-based Model Predictive Control (TMPC). TMPC has proven very effective in steering robots for post-disaster SaR[11], due to its unique capability in providing robustness to bounded uncertainties, balanced trade-offs among competing objectives of SaR (e.g., maximizing the area coverage and minimizing the mission time[4]), systematic handling of the states and inputs constraints, and on-the-go stability guarantees[12].

Main contributions and road map of the paper

This paper introduces a novel planning and control framework for autonomous navigation in dynamic, cluttered environments with the following main contributions:

Control and Operations Department, Delft University of Technology, 2629 HS Delft, The Netherlands. ✉email: m.baglioni@tudelft.nl

- **Unified planning and control:** We integrate a greedy heuristic path planning system with a robust Tube-based Model Predictive Control (TMPC) system within a single-layer architecture. This design synergizes the responsiveness of heuristic reasoning with the constraint-handling guarantees of Model Predictive Control (MPC)-based systems, leading to enhanced computational efficiency and safety compared to traditional bi-level planning-control schemes.
- **Dynamic obstacle avoidance via obstacle belts:** We extend the path planning system to handle moving obstacles by predicting their trajectories and aggregating them into time-indexed constraint regions, called *obstacle belts*, to enable anticipatory collision avoidance.
- **Uncertainty-aware TMPC reformulation:** We reformulate TMPC by replacing its nominal controller with the heuristic planning mechanism, while retaining the ancillary controller for robust trajectory tracking. Time-varying constraints and dynamic tightening account for both external disturbances and perception uncertainty.
- **Stateless, perception-driven control:** The framework operates without memory of past states, by relying solely on real-time perception. While this condition can be relaxed in the presence of advanced sensing and computational resources, this non-restrictive design reduces reliance on such infrastructure and enables deployment in partially observable environments typical of SaR missions.

Extensive simulations demonstrate the superior performance of the proposed method compared to state-of-the-art planners, particularly Horizon-based Lazy Rapidly-exploring Random Tree (HL-RRT*), which embodies heuristic reasoning similar to our planning system, and COLREGS Artificial Potential Function (APF), which emphasizes systematic obstacle avoidance and target convergence, as does the TMPC component of our proposed framework. The proposed approach outperforms both, especially in scenarios involving uncertainty and dynamic constraints.

In the rest of this paper we provide a background discussion, the problem statement and assumptions, our proposed methods, the results of the case study with discussions, conclusions, and topics for future work. Moreover, Table 1 gives the mathematical notation that is frequently used in the paper.

Background discussion

Planning the disaster response via robots depends on the disaster's environment. Three operational environments are identified for SaR[13], urban (involving constrained environments[14], e.g., inside a collapsed building), wilderness (involving open-ended environments[15], e.g., a forest), and air-sea (involving waters[16], e.g., a sea where vessels accidents or water landing has occurred). Ground robots[17], flying robots[18], and underwater robots[19] may be used in SaR, while for various indoor constrained environments ground robots are preferred[20]. Our focus is on urban SaR via ground robots. We address the problem of autonomous, effective mission planning and safe navigation of these robots. This yields to a generally nonlinear constrained optimization-based problem with multiple competing objectives, e.g., reducing the mission time while increasing the area coverage.

For SaR robots, it is common to use heuristic methods, especially those based on shortest path planners and artificial intelligence, e.g., reaction-based swarming[21], fuzzy logic control[22], and ant colony optimization[23]. The main motivation for using these methods is their computational efficiency, making them suitable for on-board deployment in SaR robotics. The main shortcomings of heuristic approaches, however, are their ad-hoc case-specific nature and the lack of performance guarantees. With advances in computational power and robotics technology[24,25], incorporating guaranteed mathematical approaches or novel integrated versions of them that provide balanced trade-offs between a high performance and computational efficiency has been emerging[4,10,26–29].

Model Predictive Control (MPC) is a mathematics-based, systematic control approach that determines a sequence of control inputs by optimizing an objective function within a given prediction window, satisfying the state and input constraints. MPC is implemented in a rolling horizon fashion, i.e., after determining the control input sequence, only the first one is injected into the controlled system and the MPC problem is solved for the shifted prediction window at the next time step. MPC has proven very effective for addressing constrained

Notation	Explanation	Notation	Explanation
κ	Discrete time step	$x^{\text{static,obs}}(o)$	The *x* position of static obstacle *o*
c	Control sampling time	$y^{\text{static,obs}}(o)$	The *y* position of static obstacle *o*
N^{p}	Prediction horizon of MPC	$x_{\kappa}^{\text{dyn,obs}}(o)$	The *x* position of dynamic obstacle *o* at time step κ
x_{κ}^{rob}	The *x* position of the robot at time step κ	$y_{\kappa}^{\text{dyn,obs}}(o)$	The *y* position of dynamic obstacle *o* at time step κ
y_{κ}^{rob}	The *y* position of the robot at time step κ	$v_{x,\kappa}^{\text{dyn,obs}}(o)$	Horizontal velocity of dynamic obstacle *o* at time step κ
$\theta_{\kappa}^{\text{rob}}$	Heading angle of the robot at time step κ	$v_{y,\kappa}^{\text{dyn,obs}}(o)$	Vertical velocity of dynamic obstacle *o* at time step κ
v_{κ}^{rob}	Linear velocity of the robot at time step κ	ρ^{obs}	Radius of the obstacle or of the smallest circular area encountering it
$\omega_{\kappa}^{\text{rob}}$	Angular velocity of the robot at time step κ	$\boldsymbol{x}_{\kappa}^{\text{ref}}$	Vector of robot's reference states for time step κ
ρ^{rob}	Radius of the robot (assuming a circular shape)	$\boldsymbol{u}_{\kappa}^{\text{ref}}$	Vector of robot's reference control inputs for time step κ

Table 1. Frequently used mathematical notations.

optimization-based problems. MPC has often been used in static SaR environments for tracking a reference trajectory that is assumed to be provided by another path planning approach[20,27,30,31].

In connection to path planning for robots, exploration and coverage of the SaR environment are both crucial[32,33]. Various approaches for area coverage have been proposed based on random search[34] or artificial intelligence (especially deep learning, reinforcement learning, fuzzy logic control[22,35,36]). In this regard, MPC has been used for SaR robots to determine control inputs that maximize a reward for visiting new parts of the environment[28,37,38]. Another novel application of MPC in multi-objective SaR via robots is through a bi-level architecture[4], where a supervisory MPC level enhances the area coverage by re-distributing SaR robots when they locally decide to visit the same or neighboring parts of the environment. This division of local and supervisory decision making provides a balanced trade-off between optimizing the global objectives of the SaR mission and meeting the computational requirements.

Robust versions of MPC[39], in particular robust Tube-based Model Predictive Control (TMPC)[40], deal with bounded uncertainties, e.g., bounded disturbances and perception errors that often occur for SaR robots. In TMPC, first the nominal version of the MPC problem is solved where no uncertainties are considered and the constraints have been tightened compared to the original problem[41]. Constraint tightening implies adjusting the bounds on the states and/or control inputs (e.g., by shrinking their admissible sets) to ensure that the controlled system always operates within its safe operational limits[12], although the operational conditions may be affected by larger disturbances than those considered in the decision making procedure. During the online implementation of TMPC, an ancillary control input minimizes the error between the nominal and actual states[42]. While the actual state trajectory may deviate from the nominal one, it always remains within a safe bounded region, called the *tube*, where the constraints are guaranteed to always be satisfied. The diameter of the cross section of the tube is determined according to the maximum difference between the nominal and actual states, considering the worst-case uncertainty scenario.

Navigating SaR robots is often target-driven[43], i.e., it involves steering the robot via reference points towards a given target. In this context, learning-based methods, such as reinforcement learning, are widely adopted[44,45]. Recent hybrid approaches combine high-level learning-based decision policies with classical planners to achieve more adaptive navigation. For example, RLoPlanner[46] integrates reinforcement learning with low-level motion planning, while hierarchical navigation algorithms, e.g.,[47], employ model-free strategies to guide flying robots in structured environments.

However, many of these methods focus on static or slowly changing environments and lack explicit mechanisms for handling time-varying constraints introduced by moving obstacles. They often require extensive training data and do not provide formal safety guarantees.

In contrast, this paper proposes a novel control architecture specifically tailored for dynamic SaR scenarios. Our framework integrates a heuristic steering approach with robust, constraint-aware MPC. This allows for real-time adaptation to moving obstacles and bounded disturbances[48], without relying on learned policies or hierarchical coordination, and results in a lightweight, yet reliable, architecture that maintains safety and computational efficiency under high environmental uncertainty.

Problem statement and assumptions

We consider the control problem of a SaR ground robot that should autonomously navigate a dynamic, cluttered, and (partially) unknown environment to reach a known target point. The control problem is formulated within a 2D continuous-space framework in discrete time, with time step variable κ. In the mathematical derivations, we consider a differential drive[49] ground robot with a circular shape of radius ρ^{rob}, but the proposed methods are adoptable for different types and shapes of robots and post-disaster environments. For the sake of simplicity of the mathematical formulations, the static and dynamic obstacles all have a circular shape with a fixed radius ρ^{obs}. This is equivalent to considering the smallest circular area that encounters an obstacle an area for the robot to avoid. The robot has a camera that provides visual information within a circular perception field centered around the robot and perceives the shape, position, and velocity of the obstacles. In addition to static obstacles (e.g., rubble or stones) there are moving obstacles (e.g., humans or falling debris) in the environment. The control system should determine per control time step the linear and angular velocities that steer the robot towards its next desired states.

We consider the following assumptions:

A1 Per time step, the robot has perfect information of its own states and scans and gathers information about the part of its environment that falls within the perception field of the robot's camera. The robot keeps no memory of the past.

A2 Environmental unmodeled disturbances (e.g., unevenness or non-smoothness of terrain) affect the states of the robot. In other words, the position, given for the center of the robot, and the velocity may diverge from their nominal values due to terrain-induced disturbances. Such effects are treated as bounded external disturbances.

A3 The perception of the robot about the position of the static obstacles is perfect, but for dynamic obstacles this perception is prone to a bounded error (which may be due to, e.g., the motion blur or the obstacle moving faster than the camera's update rate).

Bi-level control architecture for autonomous safe searching of dynamic cluttered areas

The control architecture that is proposed for steering a robot in dynamic cluttered SaR environments is illustrated in Fig. 1: In order to improve the computational efficiency and the responsiveness of the steering system of the robot, both highly crucial for SaR missions[50,51], we opt to separate the steering system into a motion planning system and an optimal motion tracking system, which must follow the trajectory that is generated by the motion

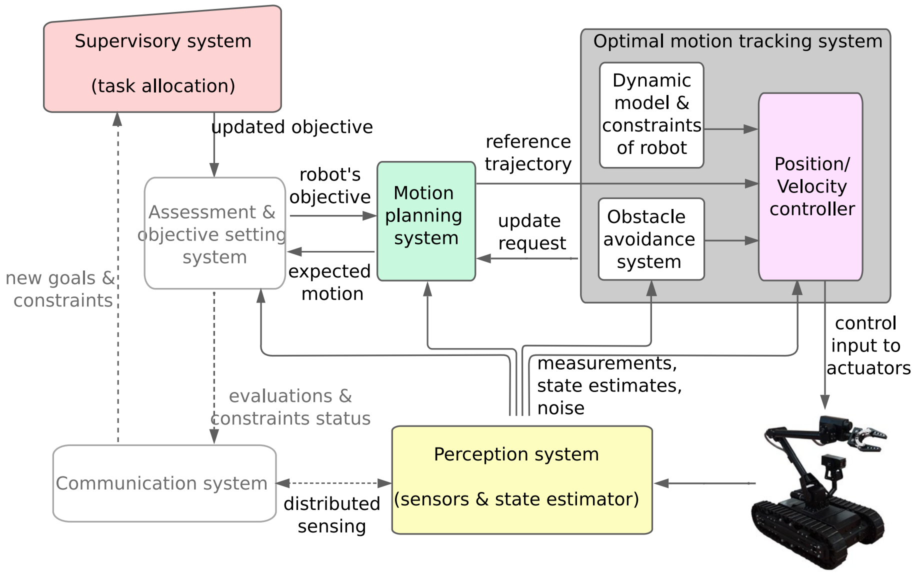

Supervisory system
(task allocation)
updated objective
Optimal motion tracking system
Dynamic model & constraints of robot
reference trajectory
robot's objective
Motion planning system
Position/ Velocity controller
Assessment & objective setting system
update request
Obstacle avoidance system
expected motion
new goals & constraints
control input to actuators
measurements, state estimates, noise
evaluations & constraints status
Perception system
(sensors & state estimator)
Communication system
distributed sensing

Fig. 1. General control architecture proposed for autonomous steering of robots in cluttered dynamic SaR environments.

planning system as closely as possible. The output of the motion planning system (shown via a green box in Fig. 1) is injected, as the reference trajectory, into the optimal motion tracking system (shown via a gray box in Fig. 1), which will request the motion planning system to re-plan the trajectory whenever needed.

Motion planning system: Heuristic planning

Due to its ease of implementation and computational efficiency, the obstacle-avoiding shortest path approach introduced by Jamshidnejad and Frazzoli[10] was chosen as the basis for the motion planning system. There are, however, two main challenges regarding the adoption of this approach for a dynamic, cluttered SaR environment that should be addressed: First, the approach only includes static obstacles and does not incorporate dynamic ones. Second, since this approach relies on local knowledge of the robot from its environment, its feasibility may be at risk.

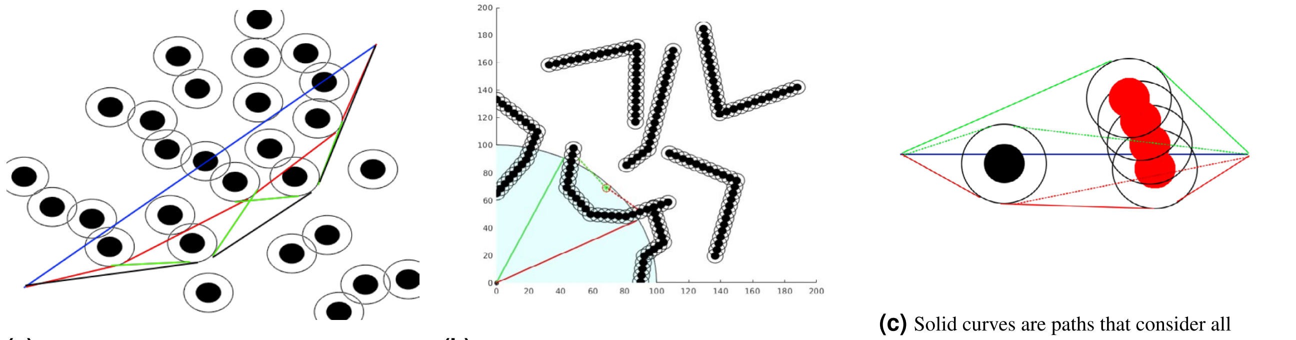
Modified obstacle-avoiding shortest path approach for dynamic environments

In the original algorithm[10], a greedy heuristic approach has been adopted to avoid obstacles that are composed of any arbitrary union of circular shapes, by generating the path across the tangential arcs to these circular shapes (see Fig. 2a). This way various arbitrarily-shaped obstacles can be handled by dividing them into smaller pieces and by approximating each piece by the smallest circle that encounters the piece. Due to its limited perception field, the robot determines a temporary intermediate target, which it intends to approach, per control time step. The motion planning system evaluates the shortest paths at both sides of the straight path that, regardless of the obstacles, connects the current position of the robot and the position of its current temporary target. In case the candidate paths are of the same length, one is chosen randomly. Equidistant points on the shortest path are injected as reference points into the local motion tracking system of the robot. In case the resulting shortest path is non-smooth or self-intersecting (see the green curves in Fig. 2a), a tangent line is drawn across the outer obstacles to smoothen the path (see the black curve in Fig. 2a).

We made the following modifications to the heuristic path planning approach, in order to make it suited for practical cases where the robot should avoid moving obstacles as well and does not store detailed past information for efficiency of the on-board computations and the memory and energy consumption (see Assumption A1): Due to the limited awareness of the robot about its environment (limited to the perception field of its camera), it may face situations where the temporary target that connects its current position and the final target is unreachable according to the robot's environmental awareness (Fig. 2b). We address this by replacing the original candidate paths with traversable paths within the detection zone of the robot that have an endpoint as close as possible to the unreachable temporary target.

Moreover, in order to incorporate the dynamic obstacles into the heuristic motion planning approach, the following steps are taken, assuming perfect knowledge about the dynamic equations of the obstacle:

Step 1 The obstacle avoiding path planning approach is implemented to generate traversable, safe paths for the robot considering only the obstacles perceived as static.
Step 2 Using the predicted dynamics of the dynamic obstacles within a given prediction window, all relevant dynamic obstacles are modeled, each as a set of static obstacles located at these predicted positions.
Step 3 The reference points of the robot for all the time steps within this prediction window are specified on the shortest traversable and safe path obtained via Step 1, with all positions of the dynamic obstacle for the time steps within the prediction window included in the picture of the robot as static obstacles.



(a) Path smoothening occurs in case of self-intersection: The plot shows the connecting line in blue, the first and final iterations in red and green, and the smoothened path in black.

(b) Current candidate paths (solid+dashed red and green), with a target that is unreachable according to current perception, are replaced by paths with reachable targets (solid green and red).

(c) Solid curves are paths that consider all positions of the dynamic obstacle (red circle) within the prediction window. Dashed curves are original paths assuming the obstacle remains static (black circle).

Fig. 2. Main idea behind the heuristic path planning approach[10]. Black hollow circles surrounding static (black) and dynamic (red) obstacles show forbidden areas for the robot. In plot 2b the robot is at the origin and its current perception field is the light blue area. In plot 2c the robot is at the left-hand side end of the black piece of line and its current target is at the right-hand side end of it.

Step 4 In case, by comparing the position of the robot and the static obstacles that represent the position of the dynamic obstacle any risks of collision are detected, the illustrated static obstacles for the time steps with a risk of collision will be united to form an obstacle belt. A new collision-free path is then generated.
Step 5 Repeat Step 1–Step 4 until the shortest traversable and safe path is determined.

Figure 2c illustrates how the shortest traversable and safe path is determined, by first considering only the static obstacle (shown via the black circle) and then by modifying the resulting path in order to avoid collisions with the static obstacles that represent the dynamic obstacle for all the time steps within the given prediction window. We assume that the robot moves across this path with linear and angular velocities that ensure a trade-off between performance and safety. This, for instance, is obtained by selecting the higher threshold between half of the maximum velocity of the robot and its midpoint velocity.

Remark 1 Obstacle representation and generalization: To enable safe navigation in environments with arbitrarily-shaped or moving obstacles, we adopt the concept of a *forbidden belt*, originally introduced in[10]. This construct allows multiple circular (primitive) obstacle regions, each representing predicted positions of dynamic objects at discrete time steps, to be unified into a deformable constraint region. The resulting belt approximates the occupancy of arbitrarily shaped or time-evolving obstacles over a prediction horizon. This abstraction supports the generalization of our method to real-world SaR environments, where obstacle boundaries are often irregular and unknown a priori. Crucially, the method does not require explicit modeling of obstacle geometry, but instead incorporates their aggregated occupancy.

Remark 2 Compatibility with memory-augmented settings: While the no-memory assumption enables a lightweight and reactive implementation, the proposed framework remains fully compatible with memory-augmented modules (e.g., SLAM or local map tracking), which will potentially further improve robustness in partially observable environments.

Next, we expand the discussions for situations where the dynamics of the moving obstacles is not perfectly known by the robot, thus the robot's predictions about the future positions of the obstacle are prone to errors.

Including dynamic obstacles when perception errors may exist in the motion planning system
In practice, especially in cluttered SaR environments that are prone to various uncontrolled stimuli, the dynamics of the obstacles cannot be perfectly captured via mathematical models. Additionally, the perceived position of dynamic obstacles based on the images of the camera is prone to errors (Assumption A3). Therefore, the path planning approach must be made robust to these uncertainties. This is guaranteed if the robot safely navigates in the environment even when maximum uncertainties are realized, i.e., when the forbidden areas (hollow circles around each dynamic obstacle in Fig. 2) are expanded considering the maximum modeling or perception error. Moreover, the cumulative errors enhance the uncertainty of the predictions. This effect is incorporated by increasing the upper value of the errors according to the proximity of the prediction to the current time. In fact, a larger prediction window allows to incorporate and assess longer-term impacts of current control inputs, which increases the chances of a safe and optimal mission and decreases the risk of recursive infeasibility. However, in addition to heavier online computations, prediction in larger windows leads to larger cumulative errors and thus risks to safety and degrading the performance. This further motivates the introduction of a bi-level control architecture that generates a reference path that approximately provides safety and desirable performance, and that delegates the obstacle avoiding task to a reference tracking control system (see Fig. 1).

Whenever the local reference tracking control problem becomes infeasible or the performance criteria falls under desirable thresholds, the motion planning system updates the reference path, based on the updated information. Meanwhile, the optimal motion tracking system keeps the motions of the robot safe and crash-free.

Optimal motion tracking system: Robust Tube-based Model Predictive Control (TMPC)

Based on assumptions A1 and A3, two sources of uncertainties affect the performance of the robot: The unmodeled disturbances that deviate the states of the robot from the planned states and the errors in perceiving the position of dynamic obstacles. Therefore, we use robust TMPC in motion tracking in order to optimally follow the reference trajectory that is determined by the motion planning system, while systematically incorporating all the constraints into the decision making procedure. TMPC uses dynamic mathematical models to predict the states of the robot and of the dynamic obstacles in a given prediction window, and uses these predictions to optimize the control inputs. The state vector of the robot, which encapsulates all the necessary past information for time step κ to predict the future states, is given by $\boldsymbol{x}_\kappa^{\text{rob}} = [x_\kappa^{\text{rob}}, y_\kappa^{\text{rob}}, \theta_\kappa^{\text{rob}}]^\top$, which includes, respectively, the 2D position of the center of the robot and the robot's orientation with respect to the horizontal axis. The control input vector that steers the motion of the robot for time step κ is given by $\boldsymbol{u}_\kappa^{\text{rob}} = [v_\kappa^{\text{rob}}, \omega_\kappa^{\text{rob}}]^\top$, which includes, respectively, the linear and the angular velocities of the robot. The obstacles are identified by their position vector $\boldsymbol{r}^{\text{static,obs}}(o) = [x^{\text{static,obs}}(o), y^{\text{static,obs}}(o)]^\top$ for static obstacle o and by their state vector $\boldsymbol{x}_\kappa^{\text{dyn,obs}} = [x_\kappa^{\text{dyn,obs}}(o), y_\kappa^{\text{dyn,obs}}(o), v_{x,\kappa}^{\text{dyn,obs}}(o), v_{y,\kappa}^{\text{dyn,obs}}(o)]^\top$ per time step κ for dynamic obstacle o, including the 2D position $\boldsymbol{r}_\kappa^{\text{dyn,obs}}(o)$ and the velocity elements of the obstacle, respectively (note that due to the circular shape of obstacles, instead of their orientation or angular velocity, the robot perceives the components of the linear velocity). The state evolves due to the accelerations $a_{x,\kappa}^{\text{obs}}(o)$ and $a_{y,\kappa}^{\text{obs}}(o)$ by the driving forces of the obstacle. Next, we explain how the motion tracking system predicts the states of the robot and of the dynamic obstacles within a given prediction window.

Dynamic prediction models for the motion of the robot and of the dynamic obstacles

The kinematics equations for translational motion of the centroid of a differential drive mobile SaR robot, as well as the orientation of the robot, both used by the motion tracking TMPC system are given by:

$$x_\kappa^{\text{rob}} = x_{\kappa-1}^{\text{rob}} + c\Big(v_\kappa^{\text{rob}} \cos\left(\theta_{\kappa-1}^{\text{rob}}\right) - c\omega_\kappa^{\text{rob}} v_\kappa^{\text{rob}} \sin\left(\theta_{\kappa-1}^{\text{rob}}\right)\Big) \tag{1a}$$

$$y_\kappa^{\text{rob}} = y_{\kappa-1}^{\text{rob}} + c\Big(v_\kappa^{\text{rob}} \sin\left(\theta_{\kappa-1}^{\text{rob}}\right) + c\omega_\kappa^{\text{rob}} v_\kappa^{\text{rob}} \cos\left(\theta_{\kappa-1}^{\text{rob}}\right)\Big) \tag{1b}$$

$$\theta_\kappa^{\text{rob}} = \theta_{\kappa-1}^{\text{rob}} + c\omega_\kappa^{\text{rob}} \tag{1c}$$

The state update equations (1a)-(1c) have been discretized in time using sampling time c, i.e., the states are updated every c time units, while during this interval their most recently updated values are used.

In the proposed architecture, both the motion planning system (in Step 3) and the optimal motion tracking system (as a prediction model embedded in MPC) need to model the dynamics of the moving obstacles. In case any knowledge exists or is deducible via filters[52] about the pattern of movement of the obstacles, the corresponding kinematics equations may be obtained.

Otherwise, since the proposed framework does not rely on prior knowledge of precise obstacle trajectories, it should predict obstacle motions based on rational assumptions, e.g., obstacles follow motion patterns similar to the behavior of the robot itself.

Since based on assumption A2, robots deploy perception pipelines that estimate obstacle motion under bounded uncertainty, predicted uncertainty envelopes can be incorporated into the obstacle belt representation to preserve safety guarantees, even in the presence of unpredictable or partially observed obstacle motions[29].

Regardless of the prediction method, the TMPC system, as is detailed in the next section, incorporates time-varying constraint tightening, which accounts for uncertainty in obstacle motion. This ensures that safety constraints remain satisfied, even under bounded deviations. The robot continuously monitors the motion of objects within its perception field, and if significant changes are detected, the framework triggers real-time re-planning, supported by the rolling-horizon strategy of TMPC. This combination of conservative planning and adaptive response enables safe navigation in environments with limited or noisy motion information.

Formulating the TMPC problem of the motion tracking system incorporating the impact of uncertainties

The objective function of the MPC problem for motion tracking per time step κ is composed of two terms: The first term includes the offset of the state vector trajectory of the robot from the reference trajectory $\left\{\boldsymbol{x}_{\kappa+1}^{\text{ref}}, \ldots, \boldsymbol{x}_{\kappa+N^{\text{P}}}^{\text{ref}}\right\}$ within the prediction window $\mathbb{P}_\kappa = \{\kappa+1, \ldots, \kappa+N^{\text{P}}\}$ that is generated by the heuristic motion planning system. The second term of the objective function represents the kinetic energy of the robot and incorporates the impact of the velocity vector of the robot to improve the energy efficiency for its motion and to assist with obstacle avoidance. This term serves dual purposes: Reducing energy consumption and moderating velocity near dynamic obstacles, which enhances safety in cluttered or uncertain environments. Note that replacing this term with a time-minimization one encourages maximum velocity, which may compromise energy efficiency and increase the risk of unsafe behavior near obstacles. Alternatively, adding a time-related term alongside the existing objectives introduces a competing priority into the optimization problem that potentially undermines the real-time tractability and responsiveness required in safety-critical scenarios.

These terms are weighed using positive parameters $w_1 < 1$ and w_2, where the impact of the predictions that correspond to farther times in the future is discounted (due to being prone to larger estimation errors) by multiplying them by w_1^k. In other words, when k is a larger time step within the prediction horizon, the weight of the corresponding term is smaller, because $w_1 < 1$. The TMPC problem for time step κ, with $\boldsymbol{x}_\kappa^{\rm rob}$ and $\boldsymbol{u}_{\kappa-1}^{\rm rob}$ given, is formulated within the prediction window $\mathbb{P}_\kappa$ via the following minimization problem:

$$\min_{\tilde{\boldsymbol{u}}_\kappa^{\rm rob}(N^{\rm c},N^{\rm p}),\tilde{\boldsymbol{x}}_\kappa^{\rm rob}(N^{\rm p})} \left(\sum_{k=\kappa+1}^{\kappa+N^{\rm p}} w_1^{k/(\kappa+1)} \left\| \boldsymbol{x}_k^{\rm rob} - \boldsymbol{x}_k^{\rm ref} \right\| + w_2 \sum_{k=\kappa}^{\kappa+N^{\rm p}-1} \left({\boldsymbol{u}_k^{\rm rob}}^\top \cdot \boldsymbol{u}_k^{\rm rob} \right) \right) \tag{2}$$

subject to the following constraints, within the given prediction window $\mathbb{P}_\kappa$:

$$(1) \text{ holds for updating the states of the robot} \tag{3a}$$

$$(1) \text{ or a dynamic equation deduced from a Kalman filter holds for moving obstacles} \tag{3b}$$

$$\boldsymbol{x}^{\min} \le \boldsymbol{x}_k^{\rm rob} \le \boldsymbol{x}^{\max} \tag{3c}$$

$$\boldsymbol{u}^{\min} \le \boldsymbol{u}_{k-1}^{\rm rob} \le \boldsymbol{u}^{\max} \tag{3d}$$

$$\left\| \boldsymbol{u}_{k-1}^{\rm rob} - \boldsymbol{u}_{k-2}^{\rm rob} \right\| \le u^{\rm smooth} \tag{3e}$$

$$\left\| \boldsymbol{r}_k^{\rm rob} - \boldsymbol{r}_\kappa^{\rm rob} \right\| \le \rho_\kappa^{\rm plan} - \rho^{\rm safe} \tag{3f}$$

$$\left\| \boldsymbol{r}_k^{\rm rob} - \boldsymbol{r}^{\rm static,obs}(o_1) \right\| \ge \rho^{\rm safe} + w_k^{\rm rob}, \qquad \text{for } o_1 \in \mathcal{I}_k^{\rm static,obs} \tag{3g}$$

$$\left\| \boldsymbol{r}_k^{\rm rob} - \boldsymbol{r}_{k-1}^{\rm dyn,obs}(o_2) \right\| \ge \rho^{\rm safe} + w_k^{\rm rob} + w_{k-1}^{\rm dyn,obs}, \qquad \text{for } o_2 \in \mathcal{I}_k^{\rm dyn,obs} \text{ and for } k > \kappa+1 \tag{3h}$$

$$\left\| \boldsymbol{r}_k^{\rm rob} - \boldsymbol{r}_k^{\rm dyn,obs}(o_2) \right\| \ge \rho^{\rm safe} + w_k^{\rm rob} + w_k^{\rm dyn,obs}, \quad \text{for } o_2 \in \mathcal{I}_k^{\rm dyn,obs} \tag{3i}$$

$$\left\| \boldsymbol{r}_k^{\rm rob} - \boldsymbol{r}_{k+1}^{\rm dyn,obs}(o_2) \right\| \ge \rho^{\rm safe} + w_k^{\rm rob} + w_{k+1}^{\rm dyn,obs}, \qquad \text{for } o_2 \in \mathcal{I}_k^{\rm dyn,obs} \text{ and for } k < \kappa + N^{\rm p} \tag{3j}$$

$$\boldsymbol{u}_{k-1}^{\rm rob} = \boldsymbol{u}_{k-1}^{\rm ref} + K_{k-1} \left(\boldsymbol{x}_{k-1}^{\rm rob} - \boldsymbol{x}_{k-1}^{\rm ref} \right) \tag{3k}$$

where $\tilde{\boldsymbol{u}}_\kappa^{\rm rob}(N^{\rm c}, N^{\rm p}) = \left\{ \boldsymbol{x}_{\kappa+1}^{\rm rob}, \ldots, \boldsymbol{x}_{\kappa+N^{\rm p}}^{\rm rob} \right\}$ and $\tilde{\boldsymbol{x}}_\kappa^{\rm rob}(N^{\rm p}) = \left\{ \boldsymbol{u}_\kappa^{\rm rob}, \ldots, \boldsymbol{u}_{\kappa+N^{\rm c}-1}^{\rm rob}, \ldots, \boldsymbol{u}_{\kappa+N^{\rm p}-1}^{\rm rob} \right\}$, with $\tilde{\boldsymbol{u}}_{\kappa+N^{\rm c}-1}^{\rm rob} = \ldots = \boldsymbol{u}_{\kappa+N^{\rm p}-1}^{\rm rob}$. Constraints (3c) and (3d) impose lower and upper limits on, respectively, the states and control inputs of the robot, where the symbol $\le$ for vectors is executed element-wise on those vectors. Constraint (3e) limits the rate of the changes in the consecutive control inputs that are injected into the actuators of the robot, and guarantees smooth mechanical movements or dynamic variations for the actuators of the robot. Constraint (3f) ensures that the position of the center of the robot for the entire prediction window remains within a bounded zone with safe borders. This zone should embed the trajectory that is planned via the heuristic motion planning system of the robot as much as safety considerations allow for it. In other words, this constraint keeps the position of the center of the robot within a circle that is centered around the current position $\boldsymbol{r}_\kappa^{\rm rob}$ of the robot with a radius $\rho_\kappa^{\rm plan} - \rho^{\rm safe}$, where $\rho_\kappa^{\rm plan}$ is the largest distance of $\boldsymbol{r}_\kappa^{\rm rob}$ from the planned path and $\rho^{\rm safe} \ge \rho^{\rm rob} + \rho^{\rm obs}$ is a parameter that is tuned based on how conservatively the safety guarantees are defined. Subtracting $\rho^{\rm safe}$ from the planned radius ensures that when the center of the robot is positioned on the borders of the safe zone, the robot will not crash into potential obstacles outside the zone. Figure 3 illustrates the essence of including constraint (3f). The variables $\mathcal{I}_\kappa^{\rm static,obs}$ and $\mathcal{I}_\kappa^{\rm dyn,obs}$ in (3g)–(3j) are, respectively, the sets of all the static and dynamic obstacles that fall within the perception field of the robot at time step $k \in \mathbb{P}_\kappa$. Constraint (3g) keeps the robot away from the detected static obstacles for all time steps $k \in \mathbb{P}_\kappa$ taking into account the impact of the disturbances on the position of the robot, by including the upper bound $w_k^{\rm rob}$ for the disturbances. In other words, terrain-induced disturbances, as described in Assumption A2, are modeled as bounded external disturbances within the motion tracking system and are compensated for through the TMPC formulation via the corresponding constraint. Constraints (3h)–(3j) prevent the robot from crashing into the dynamic obstacles that have been detected within the perception field of the robot at time step κ, by incorporating the impact of both the external disturbances that affect the position of the robot, with the upper bound $w_k^{\rm rob}$ (for $k \in \mathbb{P}_\kappa$), and the error in the perception of the estimated states of the dynamic obstacles, with the upper bounds $w_{k-1}^{\rm dyn,obs}$, $w_k^{\rm dyn,obs}$, and $w_{k+1}^{\rm dyn,obs}$ for time steps $k-1$, k, and $k+1$, respectively, where $k \in \mathbb{P}_\kappa \backslash \{\kappa, \kappa + N^{\rm p}\}$. Note that since a discrete-time problem is solved, per time step $k \in \mathbb{P}_\kappa \backslash \{\kappa, \kappa + N^{\rm p}\}$ the collision avoidance of the robot and the obstacle is enforced by providing the safe distance between their centers for the current time step and its immediate previous and next time steps, in order to reduce the risk of infeasibility. The reason for excluding time steps κ and $\kappa + N^{\rm p}$ from these constraints is the following: Since the robot does not hold any memories of the previous time steps (Assumption A1), at the beginning of the prediction horizon, i.e., at time step $k = \kappa$, it does not have access to the information of the dynamic obstacle for time step $k - 1 = \kappa - 1$. Therefore, it cannot estimate (3h). Moreover, since the upper limit of the prediction horizon starting at time step $k = \kappa$ is time step $\kappa + N^{\rm p}$, thus at time step $k = \kappa + N^{\rm p}$ the robot does not have any information about the dynamic obstacle for time step $k + 1 = \kappa + N^{\rm p} + 1$ and hence, cannot estimate (3j). The upper bounds are time-varying to account for the errors accumulated within the prediction horizon, while preventing too much conservatism for TMPC[39].

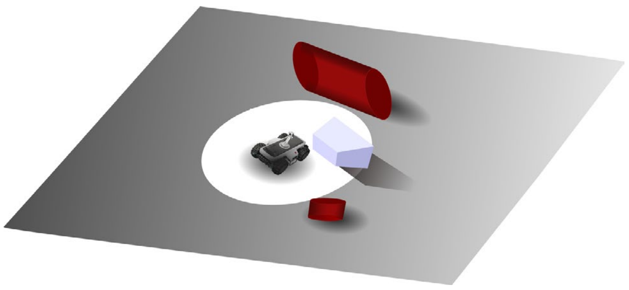


Fig. 3. The white area around the robot shows its perception field. Any (parts of the) obstacles that fall within this perception field are known to the robot, while the robot is unaware of those obstacles that fall outside this field. If no safety measures are considered, by positioning its center on the borders of the perception field, the body of the robot may crash into the obstacles that are positioned close-by to the borders of its perception field and outside of it.

TMPC estimates or deploys nominal trajectories for the state and control inputs, and adjusts the control inputs online in order to ensure that, despite the external disturbances and perception errors, the realized state trajectory of the controlled system remains within a safe, feasible region, called the tube. In our proposed architecture, the nominal state and control input trajectories for (2), subject to constraints (3a)–(3k), are instead those that have been injected into the motion tracking system via the motion planning system, i.e., $\boldsymbol{x}_k^{\text{ref}}$ and $\boldsymbol{u}_k^{\text{ref}}$. The control input adjusted online via TMPC is then determined via (3k), where the reference control input is adjusted using a control increment term that is determined based on a feedback from the system, i.e., the error between the realized and reference state trajectories. The gain K_k used for $k+1 \in \mathbb{P}_\kappa$ is determined per time step, as is common in TMPC literature, such that it stabilizes the error dynamics (which is usually done by linearizing (1) per time step k to obtain the dynamic and input matrices A_k and B_k, respectively, and by enforcing the condition $f^{\text{SR}}\left(A_k + B_k K_k\right) < 1$ where $f^{\text{SR}}(\cdot)$ is the spectral radius function, i.e., a function that determines the largest eigenvalue of the input matrix, in this case $A_k + B_k K_k$). In particular, (3h)–(3j) ensure that the realized states remain within a safe tube that is dynamically tuned via the time-varying bounds w_k^{rob} and $w_k^{\text{dyn,obs}}$, which we explain in the next section how to determine.

In case the tracking TMPC problem is deemed infeasible, it calls back to the heuristic motion panning system to ask updating the reference trajectories. The optimization problem of TMPC is in general nonlinear and non-convex and may be solved by state-of-the-art global algorithms, e.g., genetic algorithm[53] or pattern search[54], using multiple starting points.

Dynamic adjustment of the tube in TMPC due to disturbances and perception errors

We decouple the evolution of the states of the robot due to its dynamics and due to the approaching dynamic obstacles, and independently incorporate the influence of these sources of uncertainties on the tube of TMPC. The values of w_k^{rob} and $w_k^{\text{dyn,obs}}$, computed at time step κ and for $k \in \mathbb{P}_\kappa$, in the worst case are determined via the following geometric sequences:

$$w_k^{\text{rob}} = \bar{w}^{\text{rob}} \sum_{i=0}^{k-\kappa-1} \left(1 - \xi^{\text{rob}}\right)^i \tag{4}$$

$$w_k^{\text{dyn,obs}} = \bar{w}^{\text{dyn,obs}} \sum_{i=0}^{k-\kappa-1} \left(1 - \xi^{\text{dyn,obs}}\right)^i \tag{5}$$

where $\xi^{\text{rob}} \in [0,1]$ and $\xi^{\text{dyn,obs}} \in [0,1]$ are the damping values for the deviation of the robot states and for the perception error regarding the position of the dynamic obstacles, respectively, and $\bar{w}^{\text{rob}}$ and $\bar{w}^{\text{dyn,obs}}$ are the upper bounds for the external disturbances that impact the robot states and for the perception error of the robot, respectively. Based on these values, constraint tightening may be performed. Note that the damping vales should carefully be tuned to provide a balanced trade-off between increased robustness and reduced conservativeness for TMPC.

Case study

Simulations were run to compare the performance of the proposed control architecture (called HP+TMPC, referring to integrated heuristic motion planning and TMPC) with two state-of-the-art methods, Horizon-based Lazy Rapidly-exploring Random Tree (HL-RRT*)[55] and COLREGS Artificial Potential Function (APF)[56]. These two state-of-the-art methods were selected due to their complementary strengths: On the one hand, COLREGS APF represents reactive and safety-focused navigation and offers explicit obstacle avoidance and target convergence mechanisms via attraction–repulsion fields. On the other hand, HL-RRT* exemplifies computationally efficient, heuristic-based planning and responsiveness suited for partially known environments. These methods reflect two critical attributes that our framework is designed to unify: (1) robustness and safety in dynamic, uncertain environments; and (2) real-time feasibility with scalable planning capabilities. Thus, this evaluation can safely be considered sufficient and representative for demonstrating the effectiveness of the proposed method within the scope of this study.

For all the simulations, the CPU used was a 4 core 2.5 GHz Intel® Core™ i7-4710MQ with 8GB of RAM memory and an integrated GPU of Intel® HD Graphics 4600. The operating system was Ubuntu 18.04.6 LTS, a 64-bit OS, with open-source drivers, where applicable. The simulations were done on MATLAB R2020b, where the parameters used have been made publicly available in the 4TU.ResearchData repository[57].

HL-RRT* is a path planning approach based on Rapidly-exploring Random Tree (RRT*), which uses random sampling in its search space and builds up a tree with branches that connect the nearest points of the tree to each random sample, when this connection corresponds to a collision-free path. Once the target point is connected to the tree, the suitable path from the starting point to this target is selected. For enhanced computational efficiency, HL-RRT* uses a horizon-based strategy to guide the exploration, where the sampling is biased toward the points that are closer to the horizon and/or to the target. Moreover, HL-RRT* only checks the final candidate path for collision avoidance[58,59].

COLREGS APF refers to the deployment of APF for navigation, complying with the rules of COLREGS[56], i.e., international regulations for preventing collisions at sea. APF steers the heading and velocity of the robot based on the vector that combines all attraction and repulsion (e.g., due to obstacles on the way) forces between the robot and its target. The core aspect of COLREGS APF used in this case study is a horizon-based collision-avoidance strategy, which makes the comparison with an MPC-based method more relevant.

A square-shaped environment of size $14 \text{ m} \times 14 \text{ m}$ was simulated, considering **case 1** with 10 scenarios, each including 6 static and 5 dynamic obstacles, and **case 2** with 10 other scenarios, each including 8 static and 8 dynamic obstacles. In **case 2** in particular initial configurations and kinematics were designed for the obstacles such that a temporarily infeasible problem would appear for the robot. The scenarios were carefully designed to ensure that reaching the destination for the robot was always feasible in the long term. The random variables (e.g., the external disturbance affecting the position of the robot) were different among the scenarios for each case (for details see the data repository[57]). Note that while the error in perceiving the position of dynamic obstacles and the deviation of the states of the robot due to external disturbances were bounded and randomly generated, the same values were used for different control methods in order to make the comparisons fair.

The simulations were run following two setups: (1) A computation budget (0.15 s per decision making for MPC and HL-RRT*, while APF does not in practice need this time budget, as it solves the problem almost in real time) was considered, where the simulations were terminated as soon as the budget was exhausted. (2) The three approaches ran until either the target was reached by the robot or reaching the target was deemed infeasible. The comparisons of the performance among the three approaches were with regards to the rate of success of the three methods (i.e., whether or not the robot reaches the target without any collisions and without falling into a livelock, e.g., circling) and the length of the path taken by the robot to the target. In setup (2), the overall mission time (i.e., the time taken by the robot to reach its target) was also compared.

To simulate the motion of each dynamic obstacle o, the following nonlinear equations were considered:

$$x_{\kappa+1}^{\text{dyn,obs}}(o) = x_{\kappa}^{\text{dyn,obs}}(o) + \text{RK}\left(v_{x,\kappa}^{\text{dyn,obs}}(o), c\right), \quad v_{x,\kappa+1}^{\text{dyn,obs}}(o) = v_{x,\kappa}^{\text{dyn,obs}}(o) + \text{RK}\left(\alpha(x^{\text{att}}(o) - x_{\kappa}^{\text{dyn,obs}}(o)), c\right), \tag{6}$$

$$y_{\kappa+1}^{\text{dyn,obs}}(o) = y_{\kappa}^{\text{dyn,obs}}(o) + \text{RK}\left(v_{y,\kappa}^{\text{dyn,obs}}(o), c\right), \quad v_{y,\kappa+1}^{\text{dyn,obs}}(o) = v_{y,\kappa}^{\text{dyn,obs}}(o) + \text{RK}\left(\beta(y^{\text{att}}(o) - y_{\kappa}^{\text{dyn,obs}}(o)), c\right) \tag{7}$$

with $\text{RK}(\cdot, c)$ Runge-Kutta 3/8 operator that integrates the given variable across one sampling time c. The initial positions and the velocities of the obstacles were sampled based on a uniform distribution, and the motions were around fixed attraction points with coordinates $[x^{\text{att}}(o), y^{\text{att}}(o)]^\top$ per obstacle o. For the constant multipliers α and β we considered:

$$\alpha = 0.2\frac{1+4\eta}{v_x^{\text{rob,max}} - v_x^{\text{rob,min}} + \left\|x_0^{\text{dyn,obs}}(o) - x^{\text{att}}(o)\right\|}, \quad \beta = 0.2\frac{1+4\eta}{v_y^{\text{rob,max}} - v_y^{\text{rob,min}} + \left\|y_0^{\text{dyn,obs}}(o) - y^{\text{att}}(o)\right\|}$$

with $\eta \in [0, 1]$ a random number per obstacle that is sampled from a uniform distribution, and $v_x^{\text{rob,max}}$, $v_x^{\text{rob,min}}$, $v_y^{\text{rob,max}}$, $v_y^{\text{rob,min}}$ the maximum and minimum velocities of the robot in the x and y directions, respectively. These choices allow for relative velocities for the robot and the dynamic obstacles that result in obstruction of the path of the robot, thus evaluating the given approaches based on relevant, meaningful simulations. We assume that the robot uses a filter to estimate the motion of the dynamic obstacles, simply using linear approximation for the velocities. Such an approximation, considering the prediction horizon of $N^{\text{p}} = 5$ used in the case study, results in a bounded error of maximum 3.57%, which is acceptable for the simulations.

Simulation results

In this section the results of the simulations are presented. Due to the large number of experiments, we have presented only a few representative cases: Figs. 4, 5, Figs. 6, 7, and Figs. 8, 9 correspond to deploying, respectively, HP+TMPC, HLRRT*, and APF for **case 1** and **case 2**. The dash-dotted blue and solid green trajectories in the plots correspond to, respectively, setup 1 and setup 2. Static and dynamic obstacles are illustrated as solid black and red circles, respectively. In the plots on the right-hand side of these figures, the realized distance of the robot from the closest obstacle during the simulation, as well as the minimum safety radius (black dashed lines) are shown.

Tables 2 and 3 show the results for the path length and mission time of the robot for **case 1** and **case 2**, respectively. A dash symbol is used to indicate mission failure, i.e., the robot did not reach the target, due to either collisions or falling into livelocks.

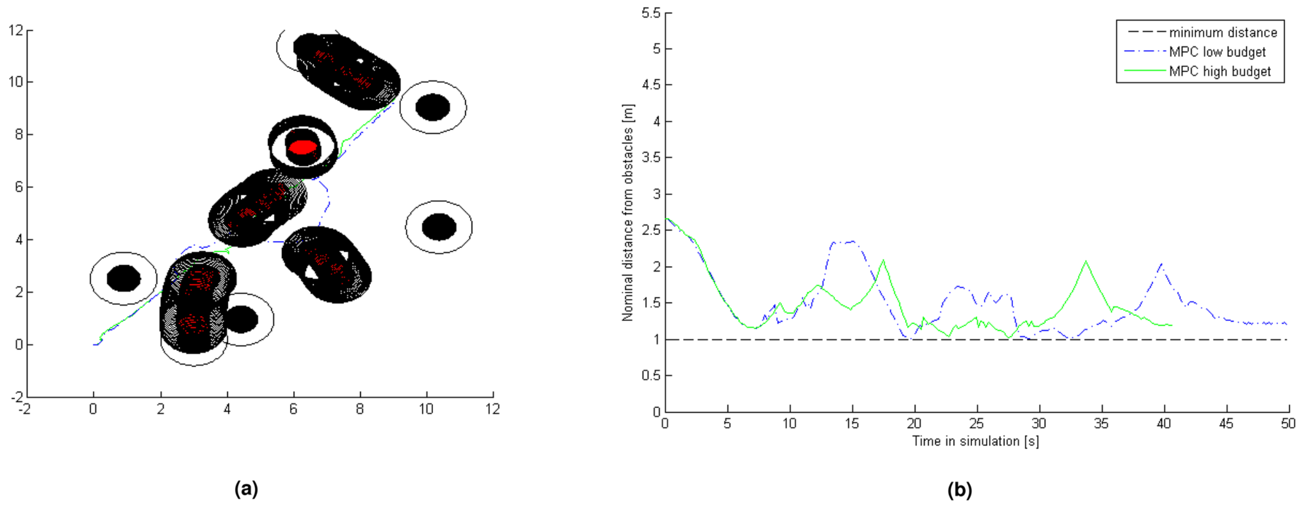


Fig. 4. (**a**) Sample robot paths for **case 1**, setups 1 and 2 (called low and high budget respectively) for the **proposed control architecture** and (**b**) the distance of the robot from the closest obstacles.

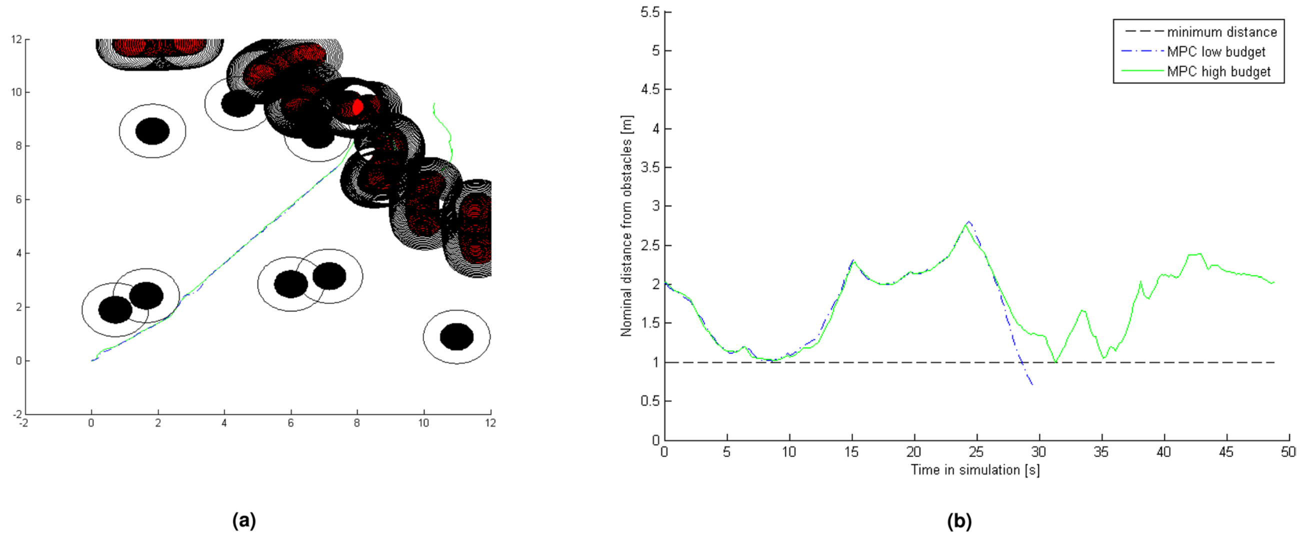


Fig. 5. (**a**) Sample robot paths for **case 2**, setups 1 and 2 (called low and high budget respectively) for the **proposed control architecture** and (**b**) the distance of the robot from the closest obstacles.

Discussion of the results

In this section, the results are discussed. In general, APF failed to perform satisfactorily in both **case 1** and **case 2**, showing only a single success from the 10 scenarios for both setup 1 and setup 2. Failures occurred because the robot got stuck in an 8-shaped path (see Fig. 8a and 9a). This turned out to be related to the tuning of APF, since after re-tuning it, the livelock disappeared and a trajectory towards the target was found (see Fig. 10a). This behavior was ultimately linked to the tuning of the hyper-parameters of APF, since after re-tuning them the livelock was eliminated and a feasible trajectory to the target was found (see Fig. 10a). This, however, highlights a fundamental limitation of APF and similar approaches, i.e., their high sensitivity to hyper-parameter settings. This sensitivity stems from the way APF combines attractive and repulsive potential fields, i.e., using fixed weights and influence radii, to compute motion commands. Any small changes in these parameters can significantly distort the resulting gradient field and lead to undesired behaviors, such as livelocks or oscillations, particularly in complex or dynamic environments. In practice, manual re-tuning of these parameters is often infeasible, especially in unstructured or time-critical scenarios. While online adaptation or learning-based tuning mechanisms will theoretically address this issue, such solutions introduce new practical and computational burdens that raise serious concerns about reliability and applicability in safety-critical domains, including SaR robotics.

From Table 2, HP+TMPC had a higher success rate than HL-RRT* for setup 1. All failures of HL-RRT* were due to crashing of the robot into obstacles, especially in a specific scenario designed to increase the risk

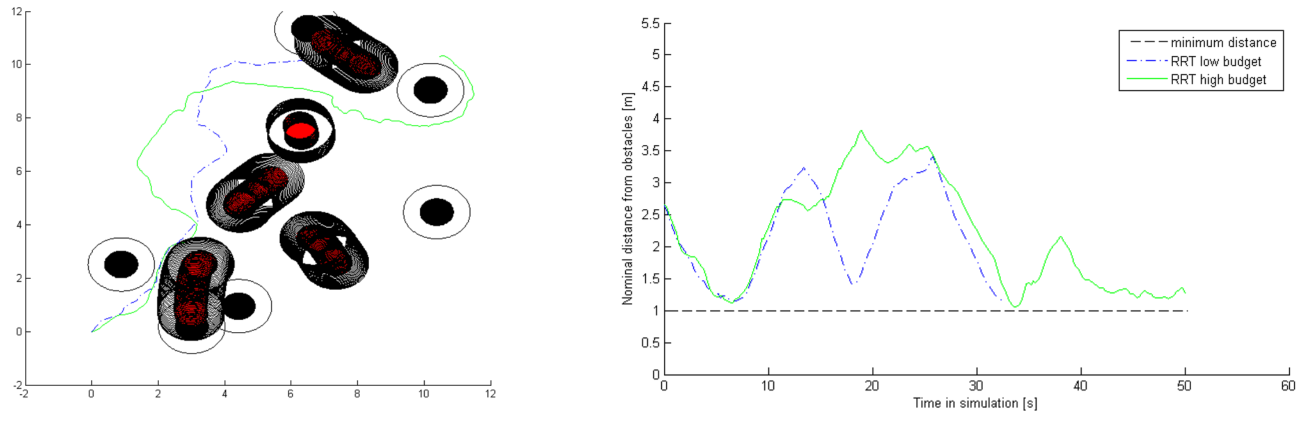


Fig. 6. (**a**) Sample robot paths for **case 1**, setups 1 and 2 (called low and high budget respectively) for **HL-RRT*** and (**b**) the distance of the robot from the closest obstacles.

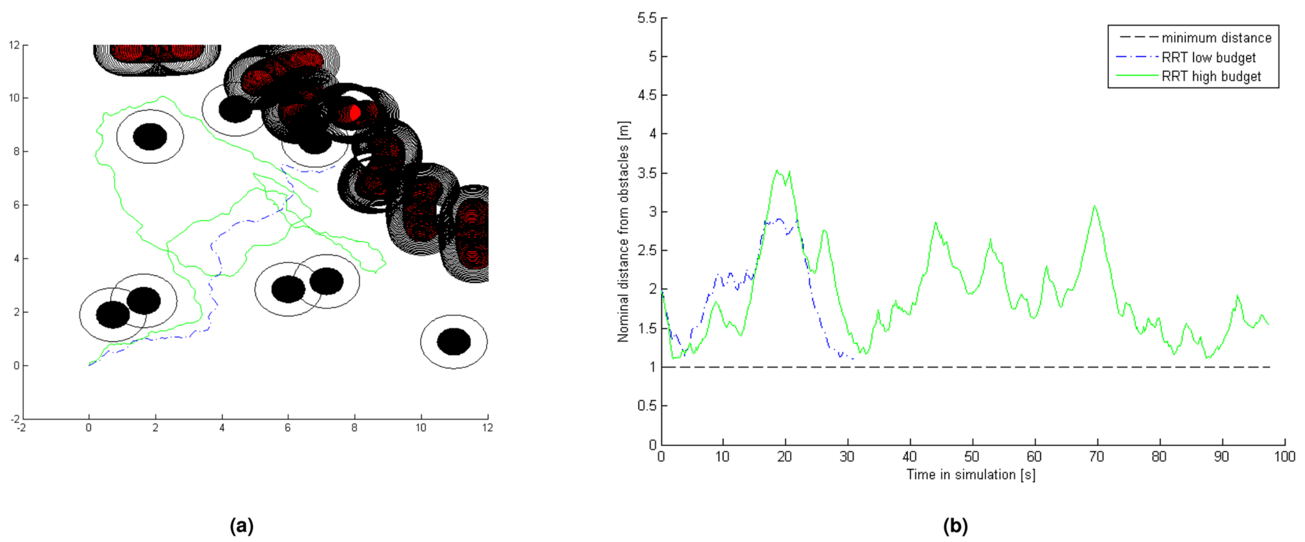


Fig. 7. (**a**) Sample robot paths for **case 2**, setups 1 and 2 (called low and high budget respectively) for **HL-RRT*** and (**b**) the distance of the robot from the closest obstacles.

of crashing (see Fig. 10b), where the robot was placed between two dynamic obstacles or one dynamic and one static obstacle. In this case, HL-RRT* did not find an alternative lower cost path in time that keeps the robot safe. These failures reflect key structural limitations of HL-RRT*. Specifically, this planner lacks predictive modeling of obstacle motion and therefore, unlike TMPC, cannot anticipate future constraint tightening. Its reliance on lazy collision checking and a fixed-time planning horizon further exacerbates the problem. Once the robot enters a narrow or transiently safe corridor, the planner may fail to re-plan in time, or may commit to paths that become infeasible during execution. In summary, unlike feedback-based methods, such as TMPC, HL-RRT* offers no mechanism for online correction or constraint adaptation. These make the planner vulnerable in scenarios where reactive safety is critical.

Instead, HP+TMPC avoided crashing into the obstacles in this challenging scenario, by either considering an obstacle belt (as a result of modeling the motion of dynamic obstacles within the prediction horizon) to avoid, or setting a reference trajectory outside the unsafe region.

For setup 2, however, HL-RRT* always handled this high-risk scenario, but with a high computational cost. In fact, using HL-RRT* the robot only moves across a path that leads to a lower cost, in this case a path that is closer to the target. Thus, unless a node was found closer to the target than the current position of the robot, it simply refused to move. Since in setup 2 there was always time to find such a node, HL-RRT* showed a 100% success rate.

Comparing the path lengths, HL-RRT* generally took a longer path than HP+TMPC, while the path of HL-RRT* was generally shorter in setup 2 compared to the same approach used in setup 1. In a few cases, HL-RRT* took a slightly shorter path than HP+TMPC, due to its random nature. See the path made by HL-RRT* in

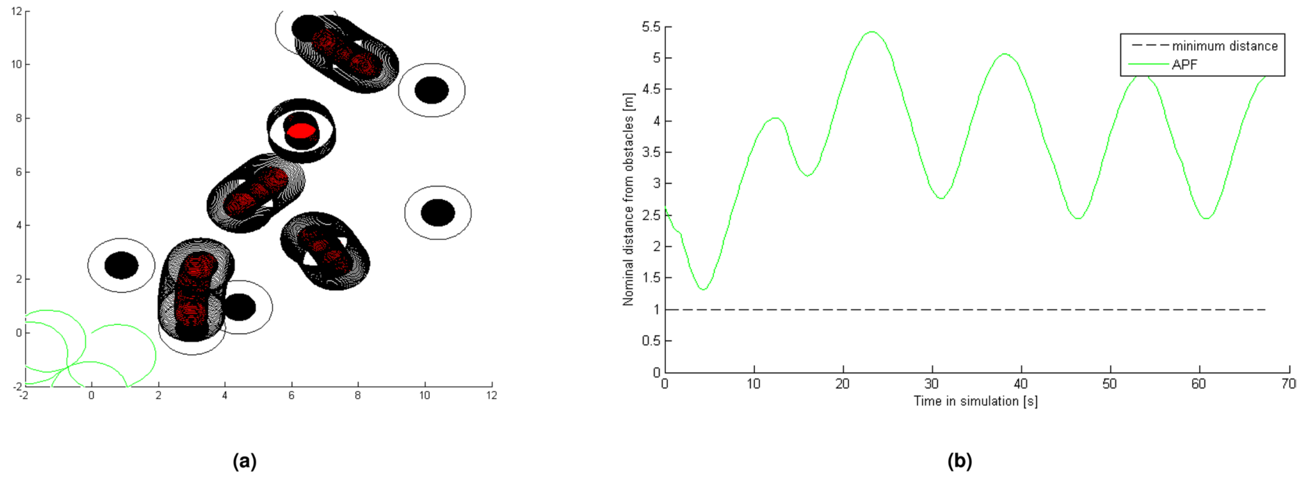


Fig. 8. (**a**) Sample robot paths for **case 1** for **APF** and (**b**) the distance of the robot from the closest obstacles.

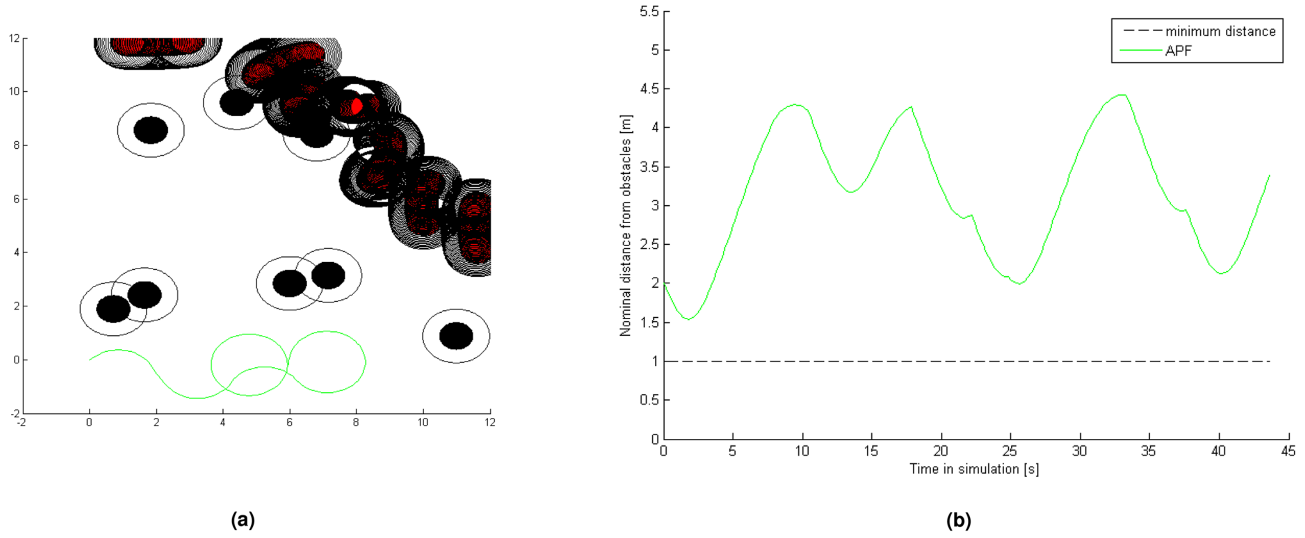


Fig. 9. (**a**) Sample robot paths for **case 2** for **APF** and (**b**) the distance of the robot from the closest obstacles.

Fig. 6a, where a wide berth is made to avoid the middle obstacles, partly due to the sampling and partly because of the local cost propagation in the tree, a limitation of RRT*. This is confirmed considering the tree that was used at the time of the berth (see Fig. 10c).

For HP+TMPC, setup 1 resulted in paths that were approximately 10% longer on average than for the same approach used in setup 2. This can be explained via Fig. 4a, where a small berth for setup 1 is observed that is avoided in setup 2 (see Figure 5a). This is linked to the limited knowledge about dynamic obstacles within a limited computational window. In fact, for setup 2 HP+TMPC found a lower cost path by slightly changing the course and speeding up, and it almost always found a local minimum, while the optimization in setup 1 sometimes stopped prematurely.

For the mission time, in setup 1 in various cases when HL-RRT* has found a path, this path has generally resulted in a smaller mission time than with HP+TMPC. In setup 2, in almost half of the cases HL-RRT* wins in achieving a smaller mission time, while in the other cases HP+TMPC wins, with, on average, 11% reduced time for the winning approach in both cases. Comparing, for instance, Figs. 4 and 6, it is clear that HP+TMPC has opted for the shortest path to the target, but since this requires moving closely to various obstacles, it may have compromised its speed for remaining crash-free, whereas by taking a longer path, HL-RRT* has avoided the obstacles significantly. This is mainly due to the formulation of the optimization problem for HP+TMPC (see (2)), where no explicit term for reducing the mission time has been considered in the objective function, but rather the controller is asked to minimize its distance from a reference trajectory that, according to the heuristic motion planning system, provides the shortest path to the target. This implies that a potential point of improvement for reducing the mission time of HP+TMPC is to include the mission time as an additional term in the objective function of TMPC.

	HP+TMPC		HL-RRT*		APF	
Scenario	Path	Time	Path	Time	Path	Time
1	18.8	49.8	-	-	-	-
2	20.1	55.7	20.0	44.5	-	-
3	22.3	50.5	-	-	-	-
4	17.9	53.3	19.3	40.2	-	-
5	16.5	41.1	19.8	44.2	-	-
6	21.0	55.2	21.9	52.1	-	-
7	15.2	32.3	15.0	30.9	15.4	30.8
8	22.4	49.7	-	-	-	-
9	20.0	59.5	24.9	22.9	-	-
10	17.8	39.3	19.2	39.2	-	-
Mean	19.2	48.6	20.0	44.2	15.4	30.8
Standard deviation	2.40	8.50	2.99	8.94	-	-

	HP+TMPC		HL-RRT*		APF	
Scenario	Path	Time	Path	Time	Path	Time
1	15.5	40.6	26.4	50.7	-	-
2	16.8	37.1	19.4	41.2	-	-
3	21.0	52.1	24.0	46.3	-	-
4	16.8	37.6	18.7	35.0	-	-
5	16.5	42.7	17.6	32.5	-	-
6	18.2	38.1	21.0	44.5	-	-
7	15.0	31.8	16.0	30.9	15.4	30.8
8	19.1	50.6	22.9	50.4	-	-
9	18.2	41.1	20.5	42.5	-	-
10	16.9	41.9	18.3	33.3	-	-
Mean	17.4	41.4	20.5	40.7	15.4	30.8
Standard deviation	1.78	6.13	3.18	7.74	-	-

Table 2. Results in terms of the path length (m) and the mission time (s) for **case 1**, setup 1 (left-hand side table) and setup 2 (right-hand side table).

Based on Table 3, for **case 2**, similarly to APF, HL-RRT* failed to show any success in reaching the target point. In particular in setup 2, none of the failures of HL-RRT* was due to colliding with any obstacles, but was mainly because, even with the larger computational budget, the algorithm failed to find any feasible paths. From Fig. 7a, HL-RRT* explores a variety of options, which are quickly dismissed, because of the lack of dynamic prediction for the moving obstacles. The main reason for the back-and-forth motions is that HL-RRT* followed a path for a while, which turned out to be infeasible later. The consistent failure of HL-RRT* in **case 2**, even under extended computational budget (setup 2), can be attributed again to fundamental limitations of this algorithm in dynamic environments. First, HL-RRT* lacks any mechanism for predicting the motion of dynamic obstacles, causing it to generate paths that are quickly invalidated during execution. Second, its horizon-based sampling strategy and cost bias tend to prune exploratory branches that may be necessary for avoiding moving obstacles, especially in constrained or deceptive regions. Third, the lazy collision checking of this algorithm delays the detection of invalid paths, leading to repeated re-planning cycles without structural adaptation. These factors combined to make the problem effectively infeasible for HL-RRT* in the more complex, dynamic scenarios of **case 2**.

The failures of HP+TMPC in **case 2** were primarily due to the inability of the optimization solver to find a feasible control solution within the allowed number of iterations. Specifically, the heuristic planner is unaware of future obstacle motion and may steer the robot into narrow regions (e.g., Fig. 10b), where obstacle trajectories eventually close in, forming a so-called 'crushing zone'. Once inside this zone, the TMPC controller receives an infeasible problem, as in fact no admissible sequence of control inputs within the velocity and safety constraints can lead to a collision-free trajectory. In these situations, the optimization solver often reaches the iteration limit without success, especially when the prediction horizon is too small to find a viable escape plan for the robot.

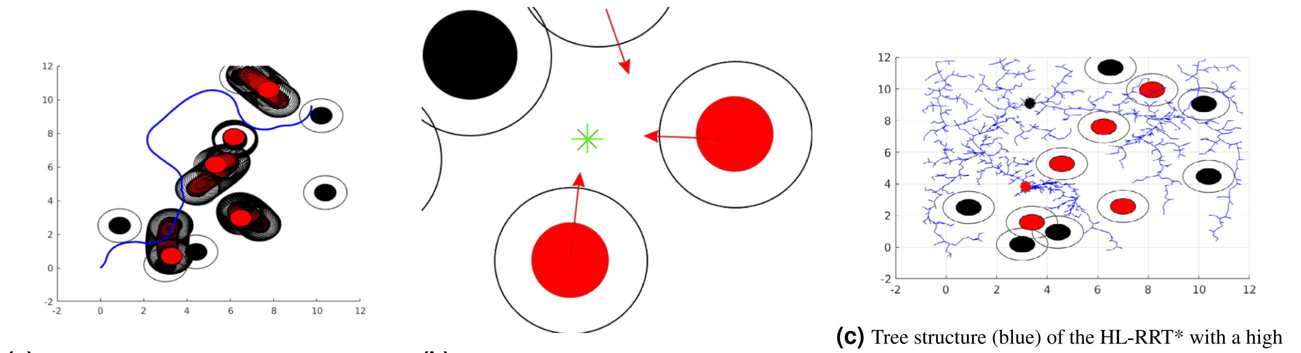
In setup 2, we observed additional cases where the controller generated overly aggressive inputs, which lead to a velocity constraint violation. This occurred when TMPC attempted to recover from a deteriorating situation introduced by the planner, and accordingly pushed the system close to the limits of feasibility. Without a sufficiently large prediction horizon or relaxed constraints, this led to constraint violations or solver failure.

These observations point to a fundamental limitation of the decoupled heuristic+MPC structure. The heuristic planner lacks awareness of dynamic feasibility, and the controller has limited authority to correct flawed plans, particularly under real-time constraints. Addressing this limitation requires tighter integration, larger prediction horizons, or more predictive planning mechanisms.

Scenario	HP+TMPC		HL-RRT*		APF	
	Path	Time	Path	Time	Path	Time
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	19.1	47.8	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-
6	20.9	51.0	-	-	-	-
7	-	-	-	-	-	-
8	-	-	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
Mean	20.0	49.4	-	-	-	-
Standard deviation	1.26	2.26	-	-	-	-

Scenario	HP+TMPC		HL-RRT*		APF	
	Path	Time	Path	Time	Path	Time
1	19.2	50.7	-	-	-	-
2	16.2	43.1	-	-	-	-
3	17.4	41.0	-	-	-	-
4	18.4	45.0	-	-	-	-
5	19.5	52.0	-	-	-	-
6	17.2	48.7	-	-	-	-
7	18.4	46.8	-	-	-	-
8	-	-	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
Mean	18.0	46.8	-	-	-	-
Standard deviation	1.16	4.01	-	-	-	-

Table 3. Results in terms of the path length (m) and the mission time (s) for **case 2**, setup 1 (left-hand side table) and setup 2 (right-hand side table).



(a) Trajectory towards the target (shown in blue) generated by APF following sub-optimal re-tuning of its hyper-parameters.

(b) Crushing motion scenario. Three dynamic obstacles (with arrows showing their movement) move towards the robot (star).

(c) Tree structure (blue) of the HL-RRT* with a high computational budget, at the time of the berth in Figure 7a. Collision checking is done from the current position (red star) to the end of the horizon (black star).

Fig. 10. Particular conditions in the simulations of the case study.

To further investigate these failure modes, we tested variations with increased prediction horizons and larger solver budgets. Both mitigated the failure cases, suggesting that windows with longer look-ahead planning improve safety, but this comes at the cost of increased computation time. This highlights the inherent trade-off between prediction depth (safety) and responsiveness (reactivity) that should be carefully balanced in real-time applications.

Under setup 1 with limited computational budget, occasional solver timeouts were observed due to the strict 0.15 s time limit. This particularly occurred in scenarios with dense obstacles or high dynamic constraints (e.g., Fig. 10b). Nevertheless, feasibility was preserved in the majority of cases because (i) the solver returned the best feasible iterate available at timeout, and (ii) tightened constraints within the optimization loop of TMPC inherently maintained safety margins.

For unlimited computation budget (setup 2), whenever the problems were structurally feasible, the optimization algorithm always converged, considering a threshold of 10^{-6} over an average objective function cost in range 20–200.

Overall, these results indicate that, under the given computational resources and simulation setup, TMPC achieved real-time feasibility. Given the consistent convergence of the optimization solver in the unlimited computation budget setting (i.e., setup 2) and its robustness under limited computation budget setting (i.e., setup 1), these findings suggest that, with hardware comparable to or exceeding our simulation platform, the proposed architecture is well-positioned for making the next step to real-world deployment. Naturally, transitioning from computer-based simulations to physical experiments will introduce new challenges (e.g., sensor noise, unmodeled dynamics, onboard resource constraints) that will require further validation and potential adaptations to ensure reliable real-time performance.

Conclusions and topics for future research

In this paper, we proposed a novel control architecture for motion planning and reference tracking of autonomous robots in dynamic cluttered environments, based on a modified version of a heuristic motion planning method[10] and robust Tube-based Model Predictive Control (TMPC). In a case study, we compared our proposed approach to two state-of-the-art methods and showed, especially for complex scenarios, to have similar or significantly higher success rates and shorter path lengths with the proposed control architecture, while producing collision-free trajectories despite multiple moving obstacles.

In the future, the proposed control architecture will be validated for different robot models, and for more variations in the motion of the obstacles, where proper filters should be merged into the control architecture to estimate the motion of these obstacles in real time. In addition, more scenarios will be simulated, including environments with varied shapes and obstacle configurations. In missions where time minimization is critical, such as emergency response, evacuation support, or medical delivery, the objective function of TMPC can be adapted to explicitly penalize time-to-go or to reward forward progress toward the target. This adaptation shifts the trade-off in favor of mission speed. Exploring such reformulations represents a promising direction for extending the applicability of the proposed framework to a broader range of real-world scenarios. This control architecture should further be extended to multi-robot systems, with potentially heterogeneous characteristics. Expanding the comparative simulations to include more recent methods, such as deep reinforcement learning, imitation learning, or hybrid planning approaches, will provide a broader benchmarking context. This constitutes a relevant future direction, especially for extending our framework to more complex or data-driven navigation scenarios. Accordingly, as part of future work, the comparative analyses should be extended by including more recent learning-based and hybrid planning methods to further benchmark the proposed framework against a broader range of navigation strategies. Finally, real-life experiments should be performed to validate the effectiveness of the control architecture beyond computer-based simulations.

Data availability

The data and parameters used in the simulations are publicly available in the 4TU.ResearchData repository: Baglioni, M. Tables with parameters values underlying the publication: Enabling robots to autonomously search dynamic cluttered post-disaster environments. https://doi.org/10.4121/aa7528da-0986-453c-b196-4277a2db4daa (2024).

Received: 13 January 2025; Accepted: 2 September 2025
Published online: 06 October 2025

References

1. Murphy, R. R. *Disaster Robotics* (MIT press, 2014).
2. Liu, Y. & Nejat, G. Robotic urban search and rescue: A survey from the control perspective. *J. Intell. Robotics Syst.* **72**, 147165. https://doi.org/10.1007/s10846-013-9822-x (2013).
3. Rajan, J., Shriwastav, S., Kashyap, A., Ratnoo, A. & Ghose, D. Disaster management using unmanned aerial vehicles. In *Unmanned Aerial Systems*, 129–155 (Elsevier, 2021).
4. de Koning, C. & Jamshidnejad, A. Hierarchical integration of model predictive and fuzzy logic control for combined coverage and target-oriented search-and-rescue via robots with imperfect sensors. *J. Intell. & Robotic Syst.* **107**, 40 (2023).
5. Mohanan, M. & Salgoankar, A. A survey of robotic motion planning in dynamic environments. *Robotics Auton. Syst.* **100**, 171–185. https://doi.org/10.1016/j.robot.2017.10.011 (2018).
6. Katrakazas, C., Quddus, M., Chen, W.-H. & Deka, L. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transp. Res. Part C: Emerg. Technol.* **60**, 416–442. https://doi.org/10.1016/j.trc.2015.09.011 (2015).
7. Pandey, A., Pandey, S. & Parhi, D. Mobile robot navigation and obstacle avoidance techniques: A review. *Int Rob Auto J* **2**, 00022 (2017).
8. Ohki, T., Nagatani, K. & Yoshida, K. Collision avoidance method for mobile robot considering motion and personal spaces of evacuees. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1819–1824 (IEEE, 2010).
9. Liu, Z., Li, M., Fu, D. & Zhang, S. Design of intelligent controller for obstacle avoidance and navigation of electric patrol mobile robot based on PLC. *Scientific Reports* **14**, 13476 (2024).
10. Jamshidnejad, A. & Frazzoli, E. Adaptive optimal receding-horizon robot navigation via short-term policy development. In *15th International Conference on Control, Automation, Robotics and Vision, ICARCV 2018, Singapore, November 18-21, 2018*, 21–28, https://doi.org/10.1109/ICARCV.2018.8581157 (IEEE, 2018).
11. Surma, F. & Jamshidnejad, A. State-dependent dynamic tube MPC: A novel tube MPC method with a fuzzy model of disturbances. *International Journal of Robust and Nonlinear Control* **35**, 1319–1354 (2025).

12. Hoy, M., Matveev, A. S. & Savkin, A. V. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey. *Robotica* **33**, 463–497 (2015).
13. Grogan, S., Pellerin, R. & Gamache, M. The use of unmanned aerial vehicles and drones in search and rescue operations–a survey. *Proc. PROLOG* (2018).
14. Nagasawa, R., Mas, E., Moya, L. & Koshimura, S. Model-based analysis of multi-UAV path planning for surveying postdisaster building damage. *Sci. Reports* **11**, 18588 (2021).
15. Hashimoto, A., Heintzman, L., Koester, R. & Abaid, N. An agent-based model reveals lost person behavior based on data from wilderness search and rescue. *Sci. Reports* **12**, 5873 (2022).
16. Serra, M. et al. Search and rescue at sea aided by hidden flow structures. *Nat. Commun.* **11**, 2525 (2020).
17. Kruijff, G.-J. M. et al. Rescue robots at earthquake-hit Mirandola, Italy: A field report. In *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 1–8 (IEEE, 2012).
18. Schedl, D. C., Kurmi, I. & Bimber, O. Search and rescue with airborne optical sectioning. *Nat. Mach. Intell.* **2**, 783–790 (2020).
19. Fattah, S. et al. R3Diver: Remote robotic rescue diver for rapid underwater search and rescue operation. In *2016 IEEE Region 10 Conference (TENCON)*, 3280–3283 (IEEE, 2016).
20. Colas, F., Mahesh, S., Pomerleau, F., Liu, M. & Siegwart, R. 3D path planning and execution for search and rescue ground robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 722–727 (IEEE, 2013).
21. Arnold, R., Jablonski, J., Abruzzo, B. & Mezzacappa, E. Heterogeneous UAV multi-role swarming behaviors for search and rescue. In *2020 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, 122–128 (IEEE, 2020).
22. San Juan, V., Santos, M. & Andújar, J. M. Intelligent UAV map generation and discrete path planning for search and rescue operations. *Complexity* **2018**, 6879419 (2018).
23. Loukas, G. & Timotheou, S. Connecting trapped civilians to a wireless ad hoc network of emergency response robots. In *2008 11th IEEE Singapore International Conference on Communication Systems*, 599–603 (IEEE, 2008).
24. Davids, A. Urban search and rescue robots: From tragedy to technology. *IEEE Intell. Syst.* **17**, 81–83 (2002).
25. Bogue, R. Disaster relief, and search and rescue robots: The way forward. *Ind. Robot: Int. J. Robotics Res. Appl.* **46**, 181–187 (2019).
26. Stecz, W. & Gromada, K. UAV mission planning with SAR application. *Sensors* **20**, 1080 (2020).
27. Berger, J. & Lo, N. An innovative multi-agent search-and-rescue path planning approach. *Comput. & Oper. Res.* **53**, 24–31 (2015).
28. de Alcantara Andrade, F. A. et al. Autonomous unmanned aerial vehicles in search and rescue missions using real-time cooperative model predictive control. *Sensors* **19**, 4067 (2019).
29. Baglioni, M. & Jamshidnejad, A. A novel MPC formulation for dynamic target tracking with increased area coverage for search-and-rescue robots. *J. Intell. & Robotic Syst* **110**, 140 (2024).
30. Hoy, M., Matveev, A. S. & Savkin, A. V. Collision free cooperative navigation of multiple wheeled robots in unknown cluttered environments. *Robotics Auton. Syst.* **60**, 1253–1266 (2012).
31. Farrokhsiar, M., Pavlik, G. & Najjaran, H. An integrated robust probing motion planning and control scheme: A tube-based MPC approach. *Robotics Auton. Syst.* **61**, 1379–1391 (2013).
32. Nattero, C., Recchiuto, C. T., Sgorbissa, A. & Wanderlingh, F. Coverage algorithms for search and rescue with UAV drones. In *Artificial Intelligence, Workshop of the XIII AI* IA Symposium on*, vol. 12 (2014).
33. Galceran, E. & Carreras, M. A survey on coverage path planning for robotics. *Robotics Auton. Syst.* **61**, 1258–1276 (2013).
34. Brooks, A., Kaupp, T. & Makarenko, A. Randomised MPC-based motion-planning for mobile robot obstacle avoidance. In *2009 IEEE International Conference on Robotics and Automation*, 3962–3967 (IEEE, 2009).
35. Paez, D., Romero, J. P., Noriega, B., Cardona, G. A. & Calderon, J. M. Distributed particle swarm optimization for multi-robot system in search and rescue operations. *IFAC-PapersOnLine* **54**, 1–6 (2021).
36. Niroui, F., Zhang, K., Kashino, Z. & Nejat, G. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics Autom. Lett.* **4**, 610–617 (2019).
37. Mohseni, F., Doustmohammadi, A. & Menhaj, M. B. Distributed model predictive coverage control for decoupled mobile robots. *Robotica* **35**, 922–941 (2017).
38. Ibrahim, M., Matschek, J., Morabito, B. & Findeisen, R. Hierarchical model predictive control for autonomous vehicle area coverage. *IFAC-PapersOnLine* **52**, 79–84 (2019).
39. Bemporad, A. & Morari, M. Robust model predictive control: A survey. In *Robustness in Identification and Control*, 207–226 (Springer, 1999).
40. Langson, W., Chryssochoos, I., Raković, S. V. & Mayne, D. Q. Robust model predictive control using tubes. *Automatica* **40**, 125–133 (2004).
41. Liu, Z. & Stursberg, O. Recursive feasibility and stability of MPC with time-varying and uncertain state constraints. In *2019 18th European Control Conference (ECC)*, 1766–1771 (IEEE, 2019).
42. Mayne, D. Q., Kerrigan, E. C., Van Wyk, E. & Falugi, P. Tube-based robust nonlinear model predictive control. *Int. J. Robust Nonlinear Control* **21**, 1341–1353 (2011).
43. Wang, H., Tan, A. H. & Nejat, G. NavFormer: A transformer architecture for robot target-driven navigation in unknown and dynamic environments. *IEEE Robotics Autom. Lett.* (2024).
44. Guo, Y., Song, A., Bao, J., Hongru, T. & Cui, J. A combination of terrain prediction and correction for search and rescue robot autonomous navigation. *Int. J. Adv. Robotic Syst.* **6**, 24 (2009).
45. Devo, A., Mezzetti, G., Costante, G., Fravolini, M. L. & Valigi, P. Towards generalization in target-driven visual navigation by using deep reinforcement learning. *IEEE Transactions on Robotics* **36**, 1546–1561 (2020).
46. Xue, Y. & Chen, W. RLoPlanner: Combining learning and motion planner for UAV safe navigation in cluttered unknown environments. *IEEE Transactions on Veh. Technol.* **73**, 4904–4917 (2023).
47. Chen, W. & Xue, Y. Hierarchical UAV autonomous navigation algorithm based on event-triggered deep reinforcement learning. *IEEE Transactions on Veh. Technol.* (2025).
48. Sani, M., Robu, B. & Hably, A. Pursuit-evasion games based on game-theoretic and model predictive control algorithms. In *2021 International Conference on Control, Automation and Diagnosis (ICCAD)*, 1–6 (IEEE, 2021).
49. Dhaouadi, R. & Hatab, A. A. Dynamic modelling of differential-drive mobile robots using Lagrange and Newton-Euler methodologies: A unified framework. *Adv. Robotics & Autom.* **2**, 1–7 (2013).
50. Scattolini, R. Architectures for distributed and hierarchical model predictive control - a review. *J. Process. Control.* **19**, 723–731 (2009).
51. Basescu, M. & Moore, J. Direct NMPC for post-stall motion planning with fixed-wing UAVs. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 9592–9598 (IEEE, 2020).
52. Esteves Henriques, B., Baglioni, M. & Jamshidnejad, A. Camera-based mapping in search-and-rescue via flying and ground robot teams. *Mach. Vis. Appl.* **35**, 117 (2024).
53. Mitchell, M. *An Introduction to Genetic Algorithms* (MIT press, 1998).
54. Torczon, V. On the convergence of pattern search algorithms. *SIAM J. on Optim.* **7**, 1–25 (1997).
55. Chen, Y., He, Z. & Li, S. Horizon-based lazy optimal RRT for fast, efficient replanning in dynamic environment. *Auton. Robots* **43**, 2271–2292 (2019).
56. Lyu, H. & Yin, Y. COLREGS-constrained real-time path planning for autonomous ships using modified artificial potential fields. *The J. Navig.* **72**, 588–608 (2019).

57. Baglioni, M. *Tables with parameters values underlying the publication: Enabling robots to autonomously search dynamic cluttered post-disaster environments.* https://doi.org/10.4121/aa7528da-0986-453c-b196-4277a2db4daa (2024).
58. Bruce, J. & Veloso, M. M. Real-time randomized path planning for robot navigation. In *Robot Soccer World Cup*, 288–295 (Springer, 2002).
59. Yang, M.-S., Lai, C.-Y. & Lin, C.-Y. A robust EM clustering algorithm for Gaussian mixture models. *Pattern Recognit.* **45**, 3950–3961 (2012).

Acknowledgements

This research has been supported jointly by the NWO Talent Program Veni project "Autonomous drones flocking for search-and-rescue" (18120), which has been financed by the Netherlands Organisation for Scientific Research (NWO) and by the TU Delft AI Labs & Talent programme.

Author contributions

K. Rado performed the conceptualization, designed and performed the simulations, contributed to methodology, analysis and validity assessment of the results, and wrote the first draft of the paper. M. Baglioni contributed to methodology, analysis and validity assessment of the results, supervision, and reviewing the final draft of the paper. A. Jamshidnejad contributed to conceptualization, methodology, analysis and validity assessment of the results, editing the final draft of the paper, project administration and supervision, and funding acquisition.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to M.B.

Reprints and permissions information is available at www.nature.com/reprints.