

ARTICLE

Modeling Time-Aware Mobile Robot Navigation by Learning Subjective Time Maps (STM)

Adrián Bañuls-Arias, Cipriano Galindo, Ana Cruz-Martín, Manuel Castellano-Quero, Juan M. Gandarias, Juan-Antonio Fernández-Madrigal and Vicente Arévalo-Espejo*

UnCORE Team of the Research Institute for Mechatronics Engineering and Cyber-Physical Systems (IMECH.UMA), Systems Engineering and Automation Department, University of Málaga, Málaga, Spain

*Corresponding Author: Vicente Arévalo-Espejo. Email: varevalo@uma.es

Received: 21 May 2026; Accepted: 17 June 2026; Published: 27 July 2026

ABSTRACT: The basic operation of a mobile robot is navigating to some target, avoiding collisions and possibly minimizing other criteria. A diversity of methods have been developed since the past century, and the research is still active, but there is one aspect that is often neglected: the duration of the steps in which computational devices divide the navigation process. Usually, it is set heuristically to a small, constant value for sampling observations frequently enough to ensure safety; however, each robot and environment has particularities that can make such a fixed timestep sub-optimal under some criteria. This paper explores the possibility of learning the best time for each step in a particular robot-environment interaction, and shows that those subjective, variable timesteps constitute an important aspect of both motion safety and efficiency. The proposed approach has an initial learning stage in a general, minimalistic environment designed for providing the main situations the robot will face; through deep RL, a time-aware reactive navigation policy is found that optimizes collision avoidance, total navigation time, and accurateness in reaching the target, yielding (*motion, timestep-duration*) commands. Due to the generalization capabilities of DRL, that policy induces Subjective Time Maps (STMs) for time-aware navigation in other scenarios. Thus, STMs become time-aware subjective models of the robot-world interaction that improve that particular robot navigation and also allow it to leverage time effort, e.g., longer timesteps can be used for energy savings or for other operations. Furthermore, the integration of uncertainties in both localization and decision-making is addressed. To achieve this, a time-aware motion policy is first learned while ignoring localization uncertainty, in order to improve learning convergence and reduce computational cost. The policy is then integrated with the resulting STMs through a particle filter (PF), either by using the time-aware motion command provided by the STM on the representative particle or by associating each particle (robot pose hypothesis) and its importance (weight) with it. The goal is to enable a full treatment of uncertainty with reduced complexity. Diverse experiments and comparisons with STMs show the utility of this spatio-temporal representation of knowledge for safe navigation, and the pros and cons of the two methods of uncertainty integration.

KEYWORDS: Mobile robots; deep learning; navigation; mapping; localization; time-aware operation

1 Introduction

Mobile robots have three basic abilities: localization, mapping, and navigation, all of them strongly intertwined. Localization and mapping were solved for practical scenarios through probabilistic filters in the early 2000s [1], while navigation had made substantial advances earlier, since the mid-1990s [2]. Today, there is a mature toolbox of methods for all of these problems, in particular for navigation [3,4], though

research is still very active, e.g., in safe navigation, particularly within crowded environments [5] or critical applications [6], automatic learning [7], and dynamic or unstructured environments [8].

Navigation can be defined as the task of reaching a target while avoiding collisions with obstacles and also optimizing some criteria, typically total navigation time, energy consumption, or gear wear. There are two fundamental ways of addressing that: *deliberatively*, i.e., planning the route, path, or trajectory from the current robot pose to the target considering a model (map) of the environment/obstacles, or *reactively*, i.e., responding to the situation perceived at each moment. The former can address the optimization of the navigation criteria from a global perspective and therefore obtain better solutions that avoid local minima, while the latter is suitable for handling unpredictable situations. In practice, a mixture of both is implemented [4], frequently called *global* and *local* navigation, respectively, or deliberation directly integrated, in some limited fashion, into a reactive process [9,10].

The quality of the solutions found by any of these methods depends on the models used for the robot motion, the environment, and the interactions between both, being those models either explicitly formalized, implicit in the robot-environment interaction, or automatically learned. However, in all cases there is an aspect that is often neglected, more evident in the reactive paradigm since it directly copes with the discrete time nature of computer motion controllers: computing devices can only issue sequences of motion commands that are not continuous in time. Usually, a fixed, often heuristically designed timestep duration is encoded into the navigator that allows the controller to keep important sensory information for the task, but choosing such a timestep is not straightforward nor free from limitations, since sensors must provide data frequently enough and the computer must have enough time for processing them and inferring the current motion command before the next one; on the other hand, there are situations where the robot is safe enough with the latest command and could use the remaining time for other computations or to improve the calculations needed in later navigation steps (or even just remaining idle for saving energy). Thus, there are situations where the robot may afford longer timestep durations and others where they should be tighter, and they depend not only on the sensory information but also on the way that particular robot interacts physically with the environment, rather than on a predefined temporal abstraction or hierarchical action structure.

In this paper, we claim that by studying the effects of timestep duration in the navigation task, and more concretely by allowing variable durations that are adapted to both the robot and the perceived situation, an efficient and robust navigation task can be carried out, especially when the robot has limited resources (sensors, computation), where conventional navigation methods can produce poorer results.

Since explicitly modeling the situations that a specific mobile robot can find in a certain class of environments (e.g., indoor structured) is in general unfeasible, we propose that the robot learns automatically time-aware motion commands subject to its own capabilities, i.e., actions extended with information about how long they should be maintained for correct navigation. This can be achieved through Deep Reinforcement Learning (DRL), which can find near-optimal policies to solve the task. Unlike hierarchical or temporally abstract RL approaches, the learning process of our proposal does not decompose behaviors into higher-level actions, but directly optimizes reactive motion commands jointly with their execution duration. Notice that the process of learning these time-aware motion commands can also embed other optimality criteria pertaining to the global navigation perspective, e.g., minimizing the total navigation time or maximizing metrical accuracy when reaching the target.

Moreover, DRL has generalization capabilities that allow us to cope with both the uncertainty in the observations of the robot and its physical interactions with the environment, and with different scenarios not previously used when learning. For that reason, we design special training environments where the robot can gather a diversity of experiences that it will face later on in real operational scenarios.

Once such time-aware, subjective policy is learned, it will encode metrical information, concretely what time-motion command the robot should execute if the target is at a certain relative position with respect to the robot and the sensors provide some particular readings. Due to the metrical nature of this information, the policy induces a Subjective Time Map (STM) for each given environment where the robot is to navigate. While learning variable-duration actions has been explored in other contexts, here the learned temporal decisions are externalized into spatial representations, enabling navigation maps to jointly encode motion and time information. These spatio-temporal maps can be used both for visualization and for path or trajectory planning in those environments as well. In short, the learned policy becomes an efficient encoding of multiple time-aware metrical maps that are proper of the sensorimotor apparatus of that particular robot. Recently, different paradigms for the inclusion of temporal information into robot maps have appeared (e.g., [11]), but they are focused on extracting long-term cyclical patterns in the observed dynamics of the world rather than in adapting a particular robot to the interactions occurring during individual timesteps.

Since our interest is in improving navigation in systems where it is difficult to perform it optimally, we focus this study on robots with limited sensory data. Also, we pursue obtaining policies as efficiently as possible; thus, we do not include the uncertainty about robot localization in the training stage, but use the simulated true position. As it is only logical, the missing uncertainty is vital for correct navigation in a real robot; therefore, we integrate it *a posteriori* with STMs. For that purpose, we use a probabilistic localization estimator, concretely a Particle Filter (PF). We first merge a representative pose of the particle cloud with the time-aware motion command corresponding to that pose, implicitly summarizing the cloud uncertainty into that representative; as an alternative, we also transfer the cloud uncertainty particle by particle to time-aware motion commands decided by the STM: the particle cloud, which originally only represents probabilistic localization hypotheses, becomes then a cloud of time-aware decision hypotheses, from which a representative navigation command can be drawn. Keeping our goal of studying robots with limited resources, where time-aware navigation can be more useful, we employ a modest number of particles in both cases.

In that context, we have evaluated the three basic components of the approach: first of all, we have designed a general indoor, structured training scenario where DRL can find a near-optimal time-aware policy. We have tested different parameters defining that scenario and assessed which ones produce the best navigation policy.

Secondly, from that policy we have deduced STMs for other scenarios, and visualized their content through suitable simplifications in order to examine whether these maps capture interesting information about the robot-environment interactions for time-aware safe navigation.

Finally, we have evaluated the integration of localization uncertainty into STMs by comparing both alternatives commented before. In our experiments, we show when each alternative produces better results than the other in a multi-metric comparison.

The novelty of this work does not lie solely in learning variable action durations, but in structuring this learned temporal knowledge into STMs and exploiting it for uncertainty-aware reactive navigation. In summary, our contributions are:

- The consideration and evaluation of the temporal aspects of navigation for mobile robots in a reactive controller.
- A novel kind of spatio-temporal map for reactive navigation containing time-aware, motion commands specifically aimed at a particular robot and kind of environment.

- A procedure for learning the content of such maps in a general way, indirectly through a single DRL policy that encodes all the required information for any STM to be built and that is adapted to the particular temporal robot-environment interactions.
- A rigorous integration of the localization uncertainty into the STMs through two different methods (PF_1 and PF_2).
- A qualitative and quantitative analysis of all the components of this approach, including comparisons with a fixed-timestep reactive navigation, ablation studies, and real experiments, and therefore a thorough assessment of the possibilities of dealing with subjective temporal aspects of reactive navigation, particularly for robots with limited sensory and computational resources.

The document is structured as follows: [Section 2](#) provides a comparison of our research with relevant approaches in the state of the art. [Section 3](#) details the proposed architecture, covering policy training, generation of Subjective Time Maps (STMs), and their integration with the localization filter. [Section 4](#) presents the experimental methodology and analyzes the obtained results. Finally, [Section 5](#) highlights the main conclusions and outlines future lines of work.

2 Related Work

Mobile robot navigation is a fundamental topic in robotics that has been extensively studied for decades. Modern approaches have been applied to various domains, including planetary exploration [12], human-centered environments [13], and dynamic urban settings [14], among others. In this section, we focus on recent works, which cope with quite specific issues such as machine learning and the treatment of uncertainty and temporal aspects.

2.1 Probabilistic Methods for Navigation

Probabilistic methods have been widely adopted to address navigation in environments where uncertainty is present, particularly because sensors, actuators, and the very robot parameters used in the models are inherently noisy. Petti and Fraichard [15] proposed a Partial Motion Planner (PMP) to plan under hard real-time constraints within a dynamic environment, combined with the Inevitable Collision States (ICS) framework [16] to address safety. Later, a stochastic mobility-based planning approach was introduced [17], extending the Rapidly-exploring Random Tree (RRT) algorithm [18] by explicitly considering robot mobility and parameter uncertainty based on the Stochastic Response Surface Method (SRS) [19], a general mathematical framework for uncertainty propagation. RRT was also extended in [20] to account for uncertainties, thereby building the concept of Safe-RRT to ensure collision avoidance and find a safe path despite existing uncertainties. Castillo-Lopez et al. [21] introduced a real-time chance-constrained planner for dynamic obstacle avoidance, offering probabilistic guarantees while maintaining computational traceability.

These considerations extend beyond planning to human-centered approaches and localization (localization itself is essential for many navigation tasks). Regarding the former, Kollmitz et al. [22] incorporated social cost maps for time-dependent planning, enhancing human-robot coexistence. More recently, a heat-map-based planner has been proposed to ensure safe navigation around pedestrians [23].

Navigation-focused probabilistic filters for robot localization are also present in the literature. Hafez et al. [24] investigated the integrity of particle-filter-based localization and introduced monitoring mechanisms to quantify safety under sensor uncertainty. They proposed a novel fault detector and derived a performance measure for the particle filter, known as integrity risk.

All the works mentioned here cope with uncertainty with the aim of enhancing safety in either localization, navigation, or motion planning tasks. However, they do not transfer the uncertainty estimation

provided by localization filters explicitly into a learned navigation policy in order to improve its robustness, which we propose in this work. Of particular interest in that matter is [25], which reports a theoretically based method for that integration. Like ours, their work adapts to the particularities of the robot, but they do not deal with the temporal aspects of the task, an essential matter, as discussed in this paper, to improve even more navigation performance (that also affects other robot operations because the robot can know how long it can be computing other potentially important processes, or improving the very navigation task, during the idle time between motion steps). Moreover, since we do not only integrate localization uncertainty into the decision-making process, but also deduce maps from the navigation temporal information, our solution can also be useful for path and trajectory planning.

2.2 Risk-Aware Navigation

A diversity of approaches have specifically studied the possible risks of navigation with mobile robots aiming at improving navigation safety. As a key example, Nguyen et al. [26] introduced a control-based navigation strategy that allows for socially acceptable robot behavior in environments with human presence. To improve navigation safety, many works even quantify risk by developing some form of related metrics that are then integrated into motion planners. Candela and Wettergreen [27], for instance, proposed a risk-aware approach integrating scientific objectives with safety constraints, where the risk was estimated using a probabilistic model that predicts slippage based on geometric and semantic information. Xiao et al. [28] proposed a robot motion risk reasoning framework based on propositional logic and probability theory, and a path planner to minimize that risk. More recently, Stone et al. [29] advanced this paradigm with a risk-bounded probabilistic roadmap (PRM)-based offline motion planner for arbitrarily shaped holonomic robots, where the risk is lower-bounded by a user-defined value.

There also exist approaches grounded on the use of maps that encode some kind of knowledge related to navigation risk [30], often human-related information (e.g., motion, activity, and their prediction) or measures capturing the geometry of the scene. For instance, several works leverage Maps of Dynamics (MoDs) [31], i.e., maps representing human motion patterns, in order to plan safe paths by making them aware of human motion and adapting them to it [32]. There also exist works that leverage knowledge on the geometry of the scene; as an example, Castellano-Quero et al. [33] introduce the use of alignability maps, i.e., maps that aim at identifying feature-sparse regions in the environment, for generating paths minimizing the risk of localization accuracy loss.

A number of contributions also aim at improving safety in motion planning by explicitly considering time. Hu et al. [34], for instance, introduce a spatiotemporal variable-step-size A* algorithm for increasing safety and efficiency in trajectory planning. Zhang and Swevers [35], as well as Taylor et al. [36], demonstrate, on the other hand, that the use of time as a variable in optimal control problems for motion planning produces solutions that better adapt to the constraints of the particular context (e.g., velocity, efficiency, collision avoidance, etc.). This idea of explicitly optimizing time has also been leveraged for human-aware navigation [37].

In contrast to the mentioned works, our approach is not strictly related to planning (planning can benefit from it, though we leave that for future work), but to reactive navigation, where coping with time is not as straightforward as optimizing total navigation time and is only feasible on a step-by-step basis. In particular, knowledge about the influence of time in reactive navigation is more naturally learned in an automated way, entangled with learning the very navigation procedure and optimizing for subjective robot behaviors, in a way that is better fitted to a particular robot morphology and its motion capabilities. All this time-aware knowledge about the idiosyncratic ways of interaction between that robot and the environment can be further embedded into a subjective metrical map from which the robot can deduce the period of time during which

it will be safe and does not need to execute further motion commands or gather new observations. Therefore, our approach enables a richer treatment of risks, adding the temporal dimension.

2.3 Learning-Based Navigation

Learning-based methods can be applied to autonomous navigation systems in many different ways. Several review papers present diverse approaches, challenges, and opportunities for that [38–40]. Recent contributions have also investigated intelligent and hybrid learning-based navigation frameworks aimed at improving trajectory optimization and control stability [41,42], further evidencing the growing interest in integrating learning and optimization techniques into mobile robot navigation. This subsection focuses on those that use Reinforcement Learning (RL), which are most closely related to our approach.

Deep reinforcement learning (DRL) has emerged as a powerful tool for navigation in high-dimensional and partially observable environments. The work in [8] provides a comprehensive and updated review of DRL-based navigation strategies in dynamic settings, highlighting their adaptability and scalability. Specific contributions include DRL-based navigation in constrained pedestrian environments [43], mapless short-range navigation under time-critical constraints [44], and adaptive replanning strategies [45]. Zhang et al. [46] exploited DRL to handle autonomous navigation in unknown terrains. In addition, inverse reinforcement learning has also been explored for human-aware navigation [47], enabling socially compliant behaviors in shared spaces.

Despite this significant progress, safe mobile robot navigation is still not considered fully solved today. Existing methods often trade off between formal safety guarantees, computational efficiency, and adaptability to dynamic and uncertain environments. Model-based (i.e., risk-aware and probabilistic) planners provide theoretical safety bounds but often struggle with scalability and generalization. Conversely, learning-based approaches exhibit strong adaptability but lack formal guarantees, raising concerns in safety-critical domains. Bridging these paradigms—by integrating formal safety mechanisms with adaptive learning strategies under uncertainty—remains an open challenge. The work in [48] provides an exhaustive and recent review of these methods, introducing a unified framework that reveals the common modular structure underlying many previously distinct approaches. This unified perspective helps clarify the relationships between existing techniques and points toward future advances.

This category shares important aspects with the perspective of probabilistic navigation, commented on before; therefore, the observations made there are also applicable here. In this work, we address the challenge of improving navigation from the perspective of learning-based methods but considering the temporal aspects of the task as an essential component. Learning them, and leveraging that knowledge for navigation enhancement, represents a novel contribution in the scope of the existing learning-based paradigms.

2.4 Time-Aware Reinforcement Learning

The temporal dimension—specifically the duration of the decision steps—remains a critical factor often simplified in DRL standard formulations. Traditional RL relies on the Markov Decision Process (MDP) framework [49], which usually assumes a fixed, synchronous discretization of time. This rigid temporal assumption is often suboptimal for mobile robotics, where the environment complexity varies dynamically in complex ways: some situations require high-frequency control for safety, while others allow for longer durations to minimize computational load or optimize trajectory fluency.

A notable contribution to handling non-uniform time in RL is the work of [50], who propose the Inhomogeneous Deep Q-Network (IDQN). By embedding agent-environment interactions into a continuous-time space using point processes, their approach accounts for irregular temporal intervals.

However, IDQN primarily focuses on adapting the value function to given time irregularities rather than empowering the agent to actively decide the optimal duration of each navigation step to balance optimality and efficiency.

More closely related to our proposal, recent works have incorporated action duration as an explicit component of the action space, allowing the agent to output commands for both motion and duration [51–53]. While these approaches successfully demonstrate the feasibility of variable timesteps in general control tasks, they generally lack an analysis of their impact on learning performance under real-world constraints. Crucially, they do not explore the spatial consistency of these temporal decisions—introduced here as Subjective Time Maps (STMs)—nor do they address how these variable-duration policies interact with probabilistic belief states, such as the localization uncertainty handled by Particle Filters.

Finally, regarding the impact of time settings on performance, Mahmood et al. [54] examined how action cycle times affect learning stability. However, their analysis treats the timestep as a hyperparameter to be tuned or analyzed in isolation, rather than a dynamic variable to be learned and adapted in real-time. In contrast, our work seeks to learn a policy that actively modulates the timestep to manage the trade-off between motion safety, navigation optimality, and uncertainty, providing a unified framework for time-aware navigation.

3 Proposed Architecture and Components

The proposed approach is split into three core components, described in detail in this section: (i) DRL for learning the navigation policy; (ii) deduction of Subjective Time-Aware Maps (STMs) that encode and represent the information derived from the learned policy once projected into particular metrical spaces; and (iii) efficient integration of the uncertainty of these maps into the one of a Particle Filter (PF) for a complete and consistent handling of the existing information.

3.1 Time-Aware DRL for Reactive Navigation

This is the first stage of our approach. We first overview DRL, then describe the task to be learned, and finally explain how it has been learned in order for the resulting STM to contain all the knowledge needed by that robot to navigate in a diversity of environments.

3.1.1 DRL Formalization

The underlying formalism of most DRL methods is a discrete-time continuous Markov Decision Process (MDP) [49], defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. In this framework, $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces, respectively; $\mathcal{P}$ denotes the environment transition dynamics $p(s_{t+1} | s_t, a_t)$; $\mathcal{R}$ is the reward function $r_t = R(s_t, a_t)$; and $\gamma \in (0, 1]$ is the discount factor that determines the importance of future rewards. The agent goal is to learn a stochastic policy $\pi(a_t | s_t)$ that maximizes the expected discounted return:

$$G = \mathbb{E} \left[\sum_{t=0}^H \gamma^t r_t \right] \quad (1)$$

where H represents the time horizon of the episode.

To ensure robust performance in robotic control, which typically requires navigating continuous action spaces, we employ the Soft Actor-Critic (SAC) algorithm [55,56]. SAC has emerged as one of the most efficient and stable choices for tasks requiring continuous control, showing superior sample efficiency and asymptotic performance compared to its predecessors [57–59].

The SAC underlying optimization objective incorporates an entropy term $\mathcal{H}$ to promote exploration and prevent premature convergence to sub-optimal policies:

$$J(\pi) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t))) \right] \quad (2)$$

where α is an optimizable temperature parameter that controls the trade-off between task reward and entropy. SAC implements an actor-critic architecture, where the actor models a Gaussian policy $\pi_\phi(a_t | s_t)$ and the critic leverages two independent Q-functions, Q_{θ_1} and Q_{θ_2} , to mitigate overestimation bias. These critics are trained by minimizing the Bellman residual using transitions (s_t, a_t, r_t, s_{t+1}) sampled from a replay buffer $\mathcal{D}$.

Notice that policies learned by SAC are optimized to maximize the expected value while maintaining sufficient stochasticity, ensuring the robot learns effective navigation behaviors that remain robust under environmental uncertainties.

3.1.2 Time-Aware Navigation Task

In this work the SAC algorithm is applied to a bespoke navigation task specifically designed to optimize three criteria: safety (obstacle avoidance), temporal efficiency (minimizing travel time), and spatial accuracy (in reaching the goal). To evaluate multiple levels of accuracy and the impact of action duration, we introduce a bullseye-shaped target (see Fig. 1) composed of three concentric rings (i_1, i_2, i_3), where each ring offers a different reward.

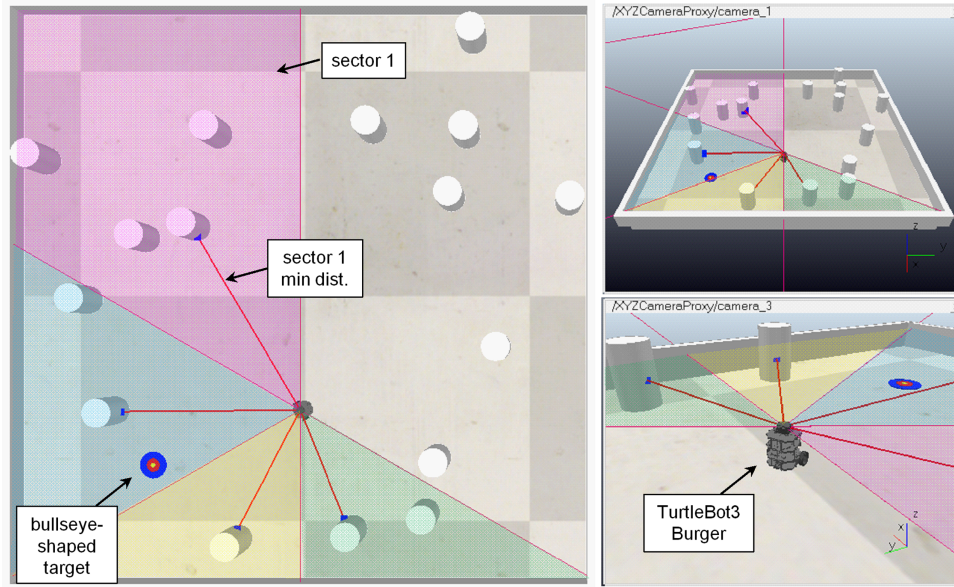


Figure 1: Training scenario designed in CoppeliaSim [60] showing the robot, the 3-ringed target (blue, red and yellow) and a bunch of cylindrical obstacles. The partition of the LiDAR Field of View (FoV), and the minimum distance measurements for each of four sectors, are also illustrated. Different colors are used to visually differentiate each sector of the LiDAR FOV.

The observation (state) space $\mathcal{S}$ for the task is designed to provide the necessary information. It comprises the distance and relative angle to the target, along with obstacle perception. The target distance is bounded between 0 and 15 m, covering the maximum theoretical distance across all experimental scenarios,

while the relative angle is defined within $[-\pi, \pi]$. Regarding obstacle detection, the LiDAR FoV is restricted to 240° centered on the robot heading. This range is partitioned into four equal sectors, where only the minimum distance measurement per sector is integrated into S , with each value bounded within $[0.16, 8]$ m (see Fig. 1). This sparse configuration is intentionally designed to evaluate the robustness of the proposed approach under limited sensory data.

The action space $\mathcal{A}$ is defined by a robot angular velocity $\omega \in [-1, 1]$ rad/s and a timestep duration $\tau \in [0.25, 2]$ s. It should be noted that the angular velocity has been constrained relative to the robot maximum physical capability ($\omega_{\text{orig}} \in [-2.84, 2.84]$ rad/s) to mitigate overspinning, thereby enhancing training convergence and overall stability. The linear velocity is fixed at 0.15 m/s, a value empirically chosen that is sufficient for the navigational requirements of our study. Notice that allowing for a variable linear speed would improve navigation performance, but also make less clear the effects of variable timesteps in our study, since linear speed and timestep are closely related magnitudes; that would also increase the dimensionality of STMs. We leave variable linear speeds as future work.

The temporal dimension of this decision-making process relies on the duration of the timestep, τ , that is also incorporated as part of the action and therefore must be learned as well. Its range has been selected to provide reasonable values for controlling navigation, covering a spectrum broad enough to allow the agent to adapt to diverse environmental densities: from high-occupancy areas requiring frequent trajectory adjustments to clear zones where a lower sampling frequency (higher τ) is permissible without compromising safety.

The learning process is guided by a reward function that is also adapted to the time-aware learning process: a sparse formulation independent of the total number of steps per episode has been adopted to ensure coherence in learning in spite of different temporal resolutions. Episode termination is triggered by four distinct conditions: reaching a target ring; colliding with an obstacle—defined as a measurement below 0.17 m—; exceeding the maximum permitted episode duration ($T_{\text{max}} = 120$ s); or surpassing a predefined distance threshold from the target ($D_{\text{max}} = 15$ m). To turn the criterion of accuracy in reaching the target into a measurable value, the episode concludes immediately upon contact with the first ring encountered; consequently, the agent cannot continue advancing toward inner rings to get higher rewards. Each ring is characterized by its radius (d_i) and an associated reward magnitude (r_i): the outer ring (i_1 , blue) is defined by $d_1 = 0.12$ m and $r_1 = 0.25$; the middle ring (i_2 , red) by $d_2 = 0.06$ m and $r_2 = 0.5$; and the inner ring (i_3 , yellow) by $d_3 = 0.02$ m and $r_3 = 1.0$.

The positive reward at the end of the episode is formulated as follows:

$$R = r_i - \frac{r_i}{2} \cdot \min\left(\frac{t}{T_{\text{max}}}, 1\right) \quad (3)$$

where T_{max} imposes a linear penalty on prolonged navigation by discounting the reward as a function of time t . The penalty scales such that if the agent reaches the target at $t = T_{\text{max}}$, the effective reward would be half of the maximum possible reward r_i for that stopping ring. This setup encourages both time-efficient trajectories and accurately reaching the target at the last step.

By adding penalties for undesirable outcomes, the reward function is formally defined as:

$$r(s_t, a_t) = \begin{cases} -1, & \text{if a collision is detected} \\ -0.8, & \text{if } t > T_{\text{max}} \text{ or } d_t > D_{\text{max}} \\ R, & \text{if } d_t < d_i \text{ for some } i \in \{1, 2, 3\} \end{cases} \quad (4)$$

The proposed task is implemented and validated using the TurtleBot3 Burger [61]. This platform is widely adopted in research due to its seamless compatibility with the Robot Operating System 2 (ROS2). As a differential drive mobile robot, it is equipped with two independently actuated wheels mounted on a coaxial axis, complemented by a passive caster wheel to ensure mechanical stability during motion. For environmental perception, the robot features a 360° LiDAR sensor—providing the data for the aforementioned sector-based observation space—with an angular resolution of 1°, which is ideal for performing Simultaneous Localization and Mapping (SLAM) and navigation.

3.1.3 Learning

The training and subsequent exploitation phases of the DRL policy have been conducted within a simulation test-bed based on CoppeliaSim [60]. The simulator physics engine has been configured with a dynamics step of 0.005 s, while the simulation step—governing the execution frequency of internal scripts, and, in our case, of the received motion commands—is set to 0.05 s. CoppeliaSim offers straightforward communication with Python scripts, a key advantage for our setup as we are using a decoupled architecture (facilitated by a custom library publicly available at [62]): on one side, the agent—comprising the robot, target, and obstacles within the CoppeliaSim scene—and on the other side, the DRL policy, which is also implemented via Python scripts using a custom Gymnasium-based environment [63]. Furthermore, the inclusion of the simROS2 package enables standardized communication through the Robot Operating System (ROS2), providing a seamless bridge for the eventual deployment of the learned policy onto physical robotic platforms.

The DRL policy has been implemented and trained with the Stable Baselines3 framework, an extensively adopted open-source library that provides reliable implementations of DRL algorithms [64]. We have used TensorBoard, a visualization tool from the TensorFlow ecosystem [65], for facilitating a thorough evaluation of training.

We have trained the robot in two variations of a carefully designed scenario that presents diverse navigational challenges. It is a synthetic, minimalist environment bounded by four walls where obstacles are simple columns, depicted in Fig. 1.

To promote generalization and diminish the training-to-exploitation-scenarios gap, the spatial configurations of both obstacles and the target, along with the robot initial orientation, are randomized at the beginning of each episode. Notice that, though randomization is considered beneficial in DRL, it does not automatically improve learning since it is not straightforward to control the variance it produces and to prevent over-conservative policies (e.g., [66]). We limit randomization by setting a safety distance between the robot and scene elements in order to avoid physically implausible starting places. Conversely, no lower bound is imposed on the initial robot-to-target distance, as episodes where the robot starts in close proximity to the goal facilitate early reward discovery. This strategy aligns with the principles of Curriculum Learning [67] and may benefit training especially during its early stages. To ensure a rigorous evaluation of the results in Section 4.1, episodes initialized within this immediate proximity are excluded from the performance analysis.

We have conducted several training experiments in these scenarios. Each one has been configured for a fixed duration of one million steps, a threshold empirically verified to guarantee learning convergence in total expected discounted reward for all configurations. The workload of learning has been distributed across several computers:

- IMECH: A high-performance workstation equipped with 4 NVIDIA RTX A6000 48 GB GPUs and an Intel Xeon Gold 5317 processor featuring 48 threads. It is provided by the Institute of Mechatronic Engineering & Cyber-Physical Systems [68] of the University of Málaga.
- Garthim1: A desktop computer with an NVIDIA GeForce GT 710 GPU and an Intel(R) Core(TM) i9-9900K CPU, offering 16 threads.

In any instance, both the DRL algorithm and the agent environment have been deployed on the same machine using our decoupled architecture. Furthermore, to ensure rigorous and fair comparisons across all experiments, the SAC hyperparameters (Table 1) have been maintained at their default values as specified in the Stable Baselines3 implementation [64].

Table 1: SAC default hyperparameters used in the experiments.

Hyperparameter	Value
Learning Rate	3×10^{-4}
Buffer Size	1,000,000
Learning Starts	100
Batch Size	256
Discount Factor (γ)	0.99
Soft Update Coefficient (τ)	0.005

The resulting policies are discussed in detail in Section 4.1.

3.2 Subjective Time-Aware Maps

Once the DRL policy has been trained for the particular robot and general kind of environment, and its robustness validated (Section 4.1), the next stage consists in deducing STMs for particular operational scenarios. In this section we describe these spatio-temporal maps.

An STM provides, given any pose of the robot in the metrical map of a given scenario and any target location, the motion command to execute and its execution time in order to navigate from that pose to the target, pursuing minimal total navigation time, maximum accuracy when reaching the target and maximum collision avoidance (all of that in a probabilistic sense, i.e., as an expectation). These maps are a deployment of the DRL policy described in the previous section into a particular metrical space; thus, once learned, a different STM is induced for any given environment.

STMs are difficult to visualize with all their information at once given the infinity of possible poses and targets within the environment, in addition to their two-fold content (timestep durations $\times$ motion commands). In order to understand their essential information, we explain in the following how simplified maps can be pictured from a subset of robot poses and headings relative to a given target.

In short, we fix a target to reach, and then define a grid of possible robot positions on the environment (farther than a minimum safety distance from obstacles). For each pair (robot, target) taken from the grid, we sample random variations of both the robot orientation and the target position; in particular we define an orientation variation window of $[-10^\circ, 10^\circ]$, sampled at $-10^\circ, -5^\circ, 0^\circ, 5^\circ$ and 10° , relative to the robot heading that faces directly the target, and, regarding the target position, we sample five locations from a Gaussian with $\sigma = 0.1$ m that is centered in the target. With all these samples, we consult the DRL policy and average the resulting outputs, selecting one out of angular velocity (ω) or timestep duration (τ). The resulting output picture is smoothed by interpolating between robot positions using the Inverse Distance Weighting (IDW) method [69].

Fig. 2 depicts the three different scenarios used in the rest of the paper for evaluating our proposal, with examples of robot starting positions and targets. Figs. 3, A1 and A2 show the visualizations of the timestep duration component in the STMs corresponding to those real scenarios. The resulting surfaces have been mapped to a color scheme where darker tones correspond to shorter timesteps, indicating either the presence of nearby obstacles or confined spaces, and lighter colors represent longer timesteps, therefore defining clearer or safer areas.

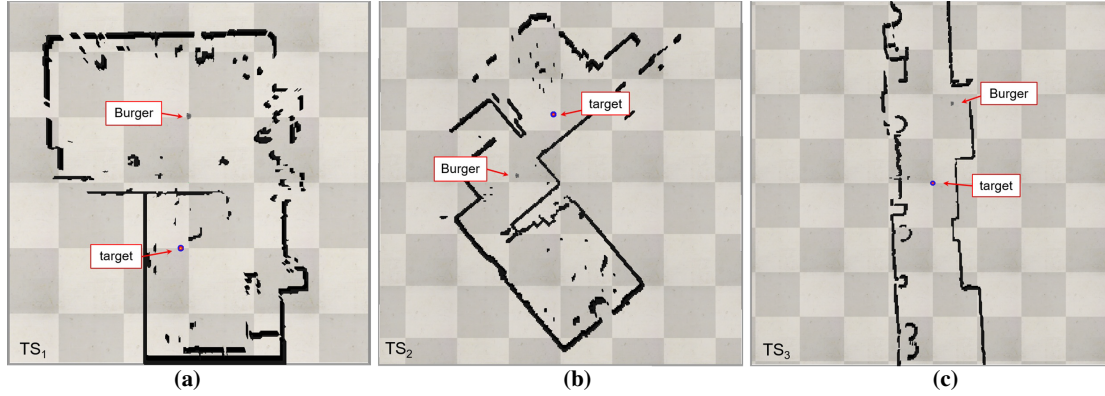


Figure 2: Cental view of testing scenes TS_1 (a), TS_2 (b) and TS_3 (c), with examples of target-robot locations.

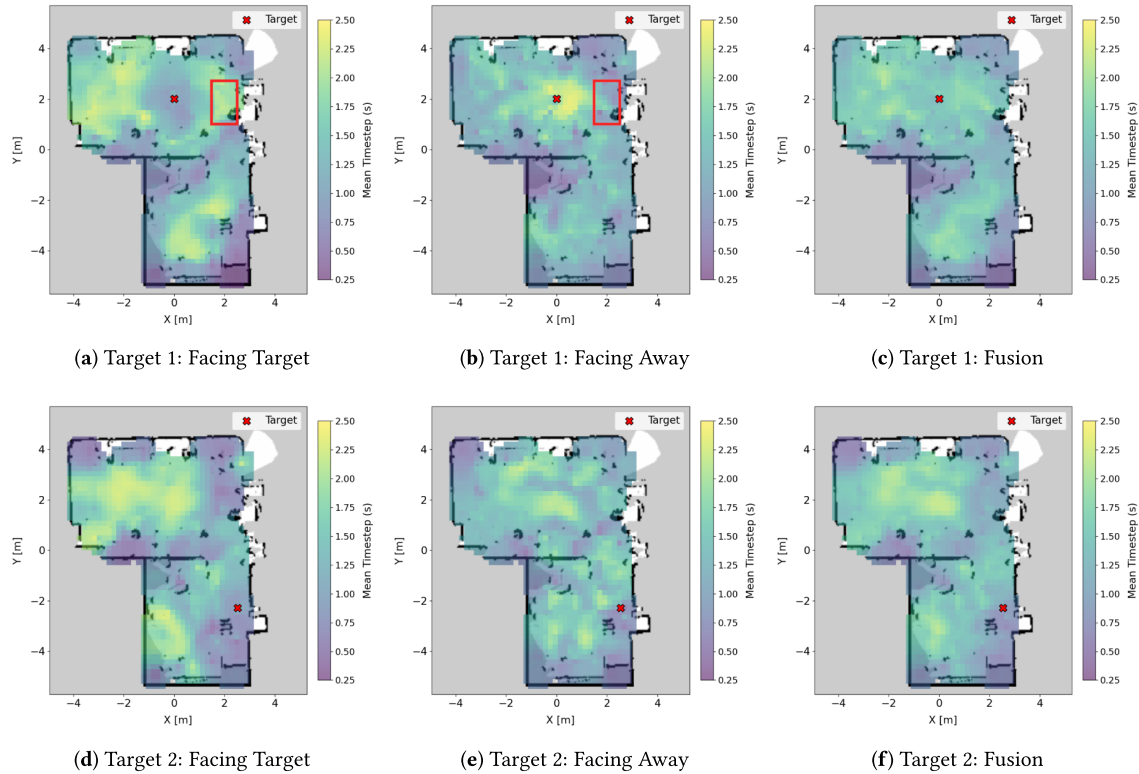


Figure 3: STM visualization (average timestep) for TS_1 across Target 1 (top) and Target 2 (bottom). Columns: facing target (a,d), facing away (b,e), and fused map (c,f).

For visualizing even more information, this process has been repeated with the opposite orientation of the robot: facing away from the target. Both maps are merged a single picture per target in the right-most figures.

As shown in the target-facing configurations (subfigures *a* and *d* in Figs. 3, A1 and A2), brighter regions (longer timesteps) align with unobstructed areas. However, cases such as Fig. 3a demonstrate that longer timestep values can appear despite the proximity of obstacles—in the region marked with a red rectangle—. This behavior emerges because the robot is oriented towards the target, effectively classifying those obstacles as low-priority risks for the current trajectory. In sharp contrast, the same region in Fig. 3b exhibits significantly shorter timesteps when the robot is directly facing the obstacles. Such orientation-dependent risk assessment underscores a fundamental advantage of STMs over traditional safety maps, which often ignore robot-target orientation and other subjective traits of the robot motion.

Conversely, the facing-away configurations in Figs. 3, A1 and A2b,e exhibit darker tones overall, as the agent must perform a reorientation maneuver—a hazardous task in confined spaces where detected obstacles can change rapidly during pivoting.

Regarding the fused maps of Figs. 3, A1 and A2c,f, they exhibit a more diffuse distribution of timesteps compared to single-orientation maps. This is expected, as the robot orientation relative to the target is a critical factor: reversing the orientation fundamentally alters the policy output.

To numerically validate these STMs, specific regions of interest have been selected for histogram analysis (see Fig. 4). As illustrated in Fig. 5, Region #1 corresponds to an unobstructed area; consequently, its values are clustered around the 1.75–2.5 s range of timesteps, indicating a lower decision-making frequency suitable for safer zones. In contrast, Region #2 represents a high-risk area, exhibiting shorter timesteps primarily in the 0.25–0.6 s and 0.6–1.25 s ranges. Finally, Region #3 encompasses the area surrounding the target. Given a fixed linear velocity $v = 0.15$ m/s and the target ring radii ($r_1 = 0.12$, $r_3 = 0.02$ m), the distance to be covered to reach the innermost ring is $\Delta d = 0.10$ m. According to a uniform linear motion, the theoretical minimum timestep required to cross this zone is ~ 0.66 s. The histogram in Fig. 5c is consistent with this theoretical calculation, showing a significant increase of occurrences of intervals near this value.

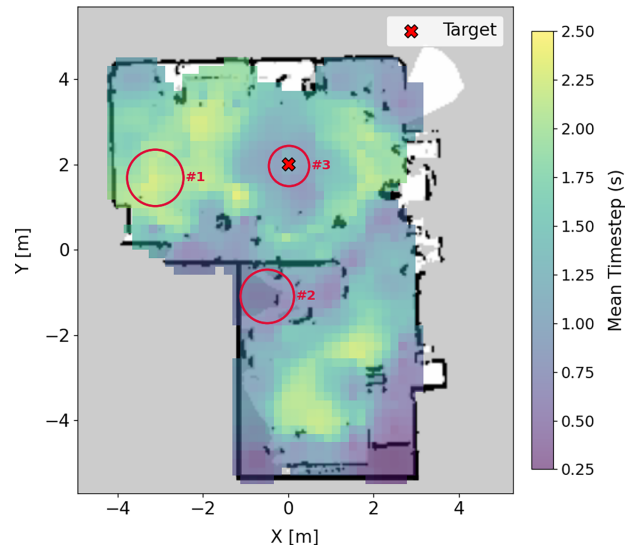


Figure 4: STM visualization (average timestep) for TS_1 across Target 1, obtained with robot facing target. Circled regions denote the specific areas detailed in Fig. 5.

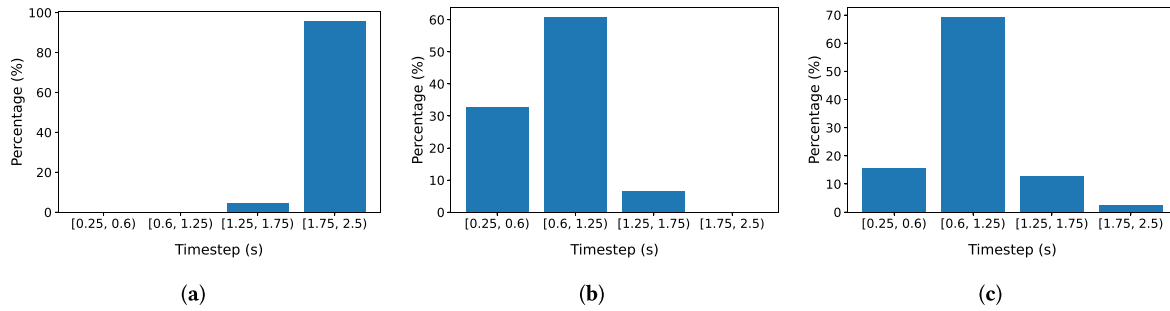


Figure 5: Timestep histograms extracted from the STM regions defined in Fig. 4: (a) Region #1 (unobstructed zone), (b) Region #2 (high-risk area), and (c) Region #3 (target vicinity).

Having established the distinct behaviors across spatial contexts, it is also interesting to analyze how sensitive these maps are to the orientation sampling bounds that we have used earlier. When implementing STMs as discrete grids, the choice of these sampling parameters directly influences the operational map content, as sampling a wider range of orientations progressively smooths out the resulting control commands—by averaging the policy actions. This mechanism for mitigating sharp variations in the control loop promotes a more stable and fluid navigation behavior during execution.

Visually, this command-smoothing effect alters the topography of the STMs. To assess this, we have systematically evaluated a wider range of orientation bounds, from $[-5^\circ, 5^\circ]$ up to $[-90^\circ, 90^\circ]$. Fig. 6 illustrates the spatial structure generated with three representative configurations: $[-5^\circ, 5^\circ]$, $[-50^\circ, 50^\circ]$, and $[-90^\circ, 90^\circ]$. As the sampling interval widens, the delineated boundaries within the map gradually diffuse. Importantly, this diffusion reflects the varying sensitivity of the previously analyzed regions. In the unobstructed zone, the diffusion is minimal since the orientation has a negligible impact on the policy-consistent preference for longer timesteps; conversely, in the high-risk area and the target vicinity, the robot heading is highly determinant for obstacle avoidance and target approaching, respectively. In those critical zones, averaging over wider orientation bounds mixes divergent policy outputs, manifesting visually as a blurred and less localized timestep distribution.

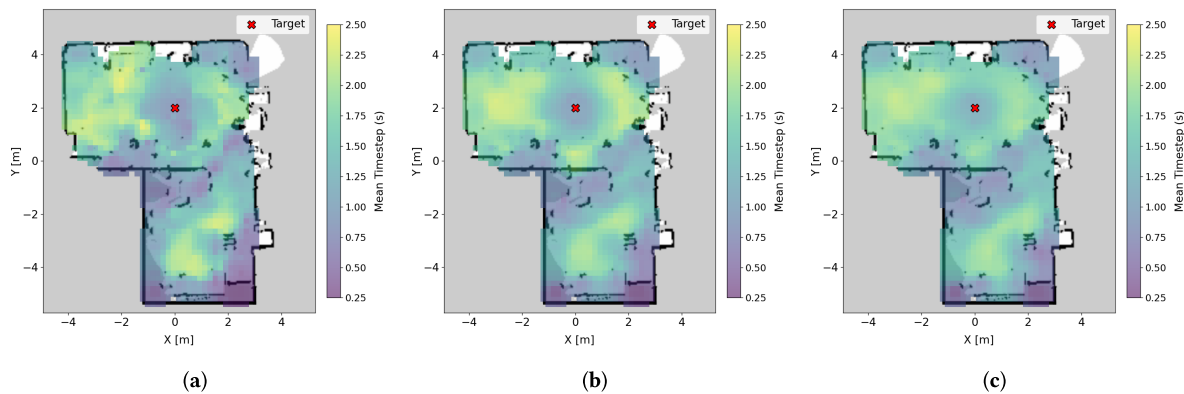


Figure 6: Comparison of STM spatial structures for TS_1 generated under increasingly wider orientation sampling intervals: (a) $[-5^\circ, 5^\circ]$, (b) $[-50^\circ, 50^\circ]$, and (c) $[-90^\circ, 90^\circ]$.

The procedure we have described so far to visualize STMs can also be used to observe motion outputs, specifically angular velocity (see Fig. 7). This yields a spatial representation of the agent's steering tendencies: darker zones denote areas where the policy selects high angular velocities to avoid imminent obstacles,

whereas lighter regions indicate areas where the robot prioritizes straight trajectories due to sufficient clearance. Notably, the vicinity of the target also exhibits lighter tones, reflecting a tendency to reduce angular velocity for precise alignment during the final steps of the trajectory.

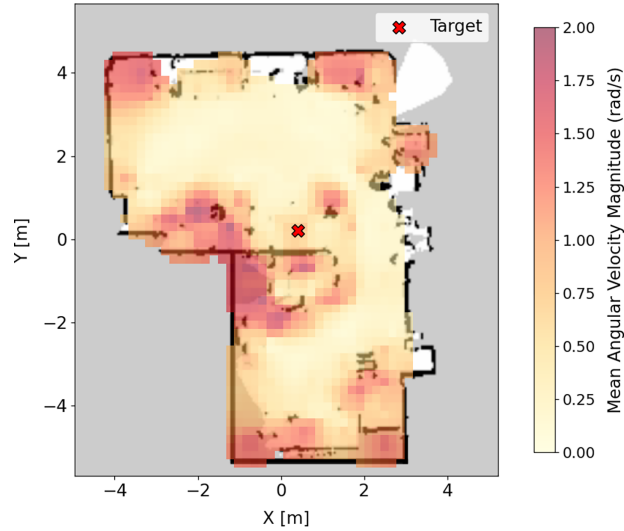


Figure 7: Spatial visualization of the angular velocity magnitude $|\omega|$ (rad/s) content of the STM of TS_1 . The map assumes the robot is oriented towards the target (marked with a red 'X').

All in all, our visualization approach provides a valuable view of the STM and of the time-aware decision-making process, covering both the timing and the direction of the robot actions. The ability of the agent to modulate its decision frequency serves as a crucial tool for identifying risky zones without requiring hazardous physical exploration; furthermore, it delineates safe regions where the agent may reallocate computational resources to other processes (being free from intensive movement and sensing tasks), such as policy refinement through exploitation feedback, which is aligned with the principles of Continual Learning [70].

3.3 Integration of the Localization Uncertainty

The final phase of our approach, to be run in real-time when the robot is operating, involves integrating the localization uncertainty of the robot, which has a very relevant role in navigation, with the one learned by the decision-making method and included in the STM. As motivated in the introduction, for an optimal and efficient learning process, the DRL policy has been learned using the true robot pose, i.e., with no localization uncertainty at all. The inherent uncertainty of stochastic policies primarily reflects the variance resulting from the learning process and reward optimization, rather than originating from the input space; conversely, deterministic policies lack any measure of uncertainty in their decision-making, always executing the same action for a specific state. We need to incorporate localization uncertainty in order to bridge this gap and ensure that the agent actions are informed by a complete measure of confidence.

A probabilistic localization estimator provides a mathematically rigorous representation of the belief state, and therefore of the uncertainty in the robot pose. More concretely, we use a Sequential Monte Carlo method, commonly known as a Particle Filter (PF) [1,71]. This method has been chosen due to its ability to handle non-linear motion and observation models, which are inherent in mobile robot navigation, and also to address different kinds of localization problems (“awakening”, “tracking” and “kidnapping”). In addition,

its integration with the decision-making process is computationally simpler than that of other parametric methods such as the Extended Kalman Filter (EKF).

PFs approximate the posterior probability distribution p of the robot pose using a set of weighted random particles; if x_t is the random variable representing the current pose, conditioned on the history of observations $z_{1:t}$ and control actions $u_{1:t}$, that posterior is approximately:

$$p(x_t | z_{1:t}, u_{1:t}) \approx \sum_{i=1}^N w_t^{(i)} \delta(x_t - x_t^{(i)}) \quad (5)$$

where each $x_t^{(i)}$ represents the localization hypothesis proposed by the i -th particle at timestep t , $w_t^{(i)}$ its associated importance weight with respect to other hypotheses, and $\delta(\cdot)$ is the Dirac delta function. Our PF implementation is a simple iterative cycle with four phases:

- **Initialization:** A set of N particles is randomly generated. In our implementation, $N = 100$ for a robot with modest computing power onboard. They are initially distributed within a 0.1×0.1 m area around the initial pose, with their orientation perturbed by up to 0.01 rad (this is a “tracking” problem, where there is some knowledge about the initial pose; see Fig. 8a). To account for motion uncertainty, we introduce zero-mean Gaussian noise in the motion model during the prediction step, with $\sigma_{X,Y} = 0.001$ m and $\sigma_\theta = 0.005$ rad.
- **Prediction:** Each particle is propagated according to the robot kinematic model during τ secs, as it can be seen in the example in Fig. 8b.
- **Update:** A simulated observation is generated from each particle, using the map of the environment. These measurements are compared against the actual sensor data z_t to update weights: $w_t^{(i)} \propto w_{t-1}^{(i)} \cdot p(z_t | x_t^{(i)})$.
- **Resampling:** Particles with weights below a predefined threshold are removed, while those with bigger weights are replicated to keep the initial N (see Fig. 8c). This algorithm diminishes the depletion and impoverishment problems of vanilla PF with less computational burden than a pure Sampling Importance Resampling (SIR), due to the reduced generation of random numbers ([1]).

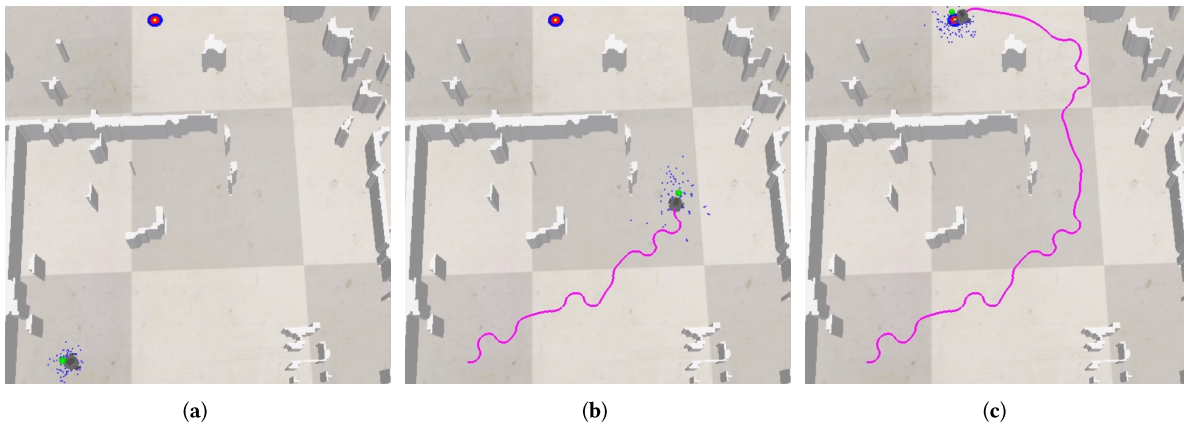


Figure 8: Instances of a pose estimation process using a PF (under manual navigation). At each iteration, the estimated robot pose is marked with a green ball, while the particles are propagated as blue points. The sequence illustrates the temporal progression of the filter: (a) initial particle distribution, (b) intermediate state reflecting some uncertainty of the particle cloud, and (c) final state with the particles accurately tracking the robot.

The decisions we have made for limiting the computational cost of the PF, needed for a practical implementation on a low-power mobile platform, can increase the robot uncertainty about its location. This has both pros and cons for our approach: on the one hand, increased uncertainty also increases the probability of bad localization, but on the other, it allows the robot to escape local minima that are natural in reactive navigators. In the experiments discussed in this paper, we have obtained reasonable results with this setting.

In this work we propose to integrate the localization uncertainty into the STMs through two methods:

- PF_1 : A point estimated representative of the particle cloud is gathered (its expectation); the resulting pose incorporates the uncertainty in localization in a condensed form, i.e., the approximate first moment of the posterior. Then, the STM is queried just once for the motion command that corresponds to that expected pose.
- PF_2 : The full posterior uncertainty is transferred to the decision-making navigator: a query is issued to the STM for every particle i , yielding N distinct time-aware motion commands ($v^{(i)}, \omega^{(i)}, \tau^{(i)}$). These are then combined using the particles' weights $w^{(i)}$ to produce a single command. This is a sort of particle filtering of time-aware motion commands instead of poses.

Notice that PF_1 consists essentially in calculating a function (motion command) of an expectation (representative pose from the cloud), while PF_2 does the opposite: it calculates the expectation of several values of that function when evaluated at different poses (particles); therefore, PF_1 relies directly on the estimation of the robot pose while PF_2 will be less influenced by errors in that estimate. Basic probability theory indicates that both approaches provide different results except when the function is linear, but our entire function is not (it is modelled non-linearly by the neural networks underlying DRL).

We can expect that, when the particle cloud has little variance and it is well centered around the true robot pose, both methods will behave approximately the same; only when the cloud has large variance and/or has a relevant bias with respect to the true pose they will be different, and in that case the question is whether the work of PF_2 , that translates uncertainty to motion commands without relying directly on an estimate of the robot pose, serves to improve the navigation performance metrics obtained with the direct estimation of the robot pose used by PF_1 .

3.4 Implementation Details

The proposed architecture is structured around simulation, probabilistic localization, robot and reinforcement learning software. The interaction between these components, depicted in Fig. 9, forms a closed-loop control system:

- Robot agent interface (RAI): This component manages the robot motion via CoppeliaSim (in simulation) or interfacing with the corresponding ROS2 topics (in physical deployment). Regardless of the operation mode, it is used for providing the robot pose relative to the target and also for predicting the observation of each particle in the update of the localization filter.
- Map server (STM): A Python-based server that hosts the maps derived from the trained policy. It processes state observations and returns time-aware control actions (v, ω, τ).
- ROS2 Particle Filter (PF) node: It executes the prediction and update steps of the particle cloud based on sensor data and motion commands obtained from the STM.
- Central manager node (CM): It coordinates the data flow between nodes, initializes the experimental parameters, and issues timed movement commands to the agent depending on the specific configured experiment (see Section 4.2).

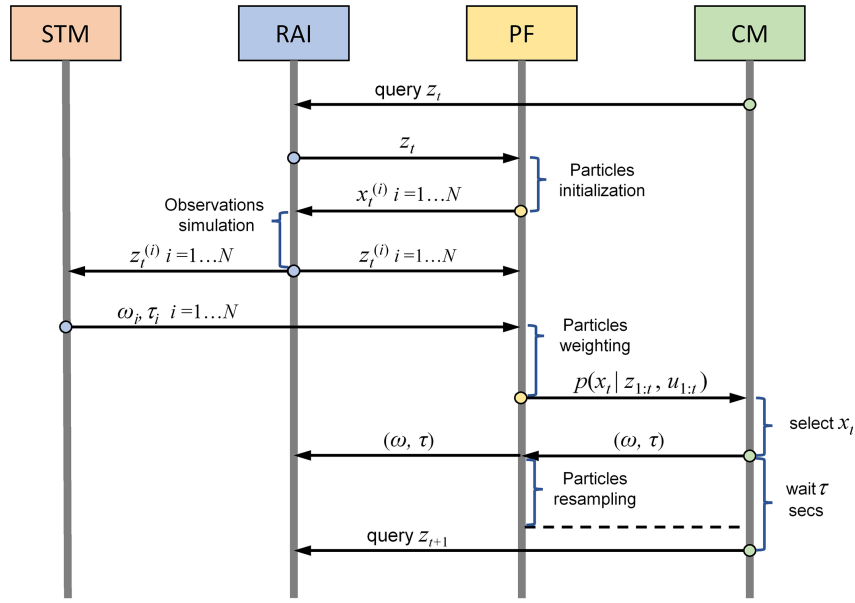


Figure 9: Software communication architecture of the proposed approach.

The operational sequence is as follows:

1. **Observation Retrieval:** The CM requests the current localization from the RAI and, depending on the execution mode, observations from the simulation engine or the physical sensors.
2. **Particle initialization:** The RAI forwards observations to the PF node, which initializes 100 particles around the estimated pose. These hypothetical poses are sent back to the RAI to synchronize the belief state within the simulation environment.
3. **Observation querying:** For each hypothetical pose, its simulated observation is sent simultaneously to the STM server and to the PF. The STM queries the corresponding angular velocity and timestep for each particle perspective.
4. **Particles weighting:** Those observations are received by the PF node, which can update the weight of the particles and send the posterior cloud information to the manager.
5. **Execution:** The manager transmits the resulting movement command (ω, τ) to the RAI for execution. After a delay equivalent to the selected timestep τ , it requests a new observation from the agent, closing the loop and continuing the trajectory until the goal is reached.

Besides, careful management of time must be implemented as well, particularly considering that a motion command duration will not be obeyed exactly—the system must do “extra” processing in order to obtain sensory data, update the localization filter, and decide the command. That “extra” time extends the effective duration of the previous motion command, since the robot does not stop while running the extra processes. Note that those temporal extensions do not affect the quality of learning, since the DRL learns to correlate motion command durations with the “extra” times found on the computer it used (in addition, no localization filter is active during learning, which also tends to produce shorter “extra” times). However, that learned mapping between commanded durations and effective durations is only valid on the training computer; when the STMs are used later on in the operational computing platform, the “extra” times that extend motion commands will be different, which may affect the optimality of the policy.

In order to cope with this implementation issue, preserving as much as possible the optimality of the learned policy, we (i) reduce to the minimum the time employed by the “extra” processing in the operational

platform, and (ii) adjust the commanded durations yielded by the STM, aiming at replicating the ones of the training environment. Point (i) is achieved by minimizing the logging of data during the experiments (which would involve additional delays not needed for operation) and relying on a likelihood field [72] in the update stage of the localization filter, a computational method largely used in robotics and artificial intelligence to efficiently calculate the probability of a specific sensor measurement given a robot pose and a known map; point (ii) is implemented by drawing random “extra” times from a probability distribution of the “extra” times recorded during learning, and then adjusting the commanded duration according to that artificial value and the actual “extra” times measured in the robot, to replicate the distribution of the former.

As an analysis of the possibilities of scaling our approach to robots with more sensory data, we have also analyzed the effects on computation time of that circumstance, taking into account the main sources of computational cost explained in the previous paragraph. In particular, we have measured the times employed in computing the core of the PF_2 method, i.e., requesting motion commands for 100 particles (pose hypotheses), under several densities of the LiDAR sensor. Fig. 10 shows the result of these experiments. An approximately linear complexity can be deduced from those results, which, through extrapolation, indicates that our methods could support up to 512 beams adequately processed in around 300 ms (when we also consider the computation time due to the particle filter and the rest of the processes, which are analyzed in Section 4.2.1).

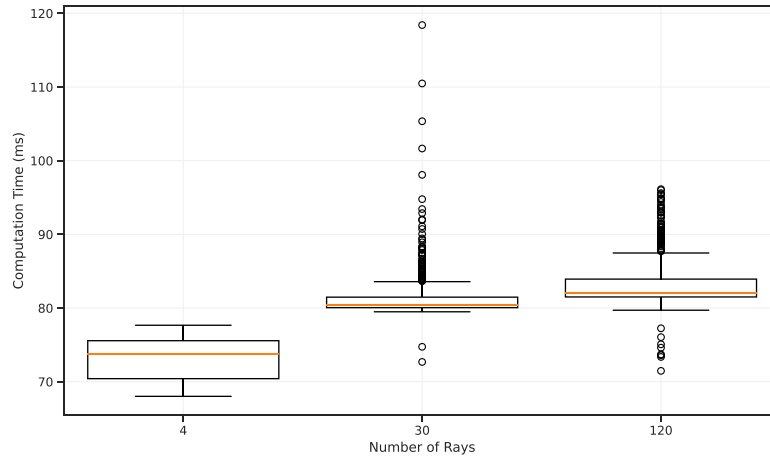


Figure 10: Computation times of DRL policies trained with different numbers of sensor beams.

4 Evaluation and Discussion

In this section, we evaluate the proposal by defining several metrics. Section 4.1 details the comparison of different DRL policies used for deducing the STMs in order to identify the best-performing and most general one.

The evaluation of the PF method applied to safe robot navigation with those STMs under different configurations and scenarios is discussed in Section 4.2.

4.1 Evaluation of the DRL Policy

The time-aware navigation DRL policy used for deducing STMs must be as general as possible to be robust in any situation the robot may find when navigating different scenarios. For finding the best policy that achieves such generalization, we vary three fundamental design parameters: the scale of the training scenario (Constrained vs. Large), the obstacles distribution (Fixed vs. Randomized, in order to find a suitable

level of randomization for learning), and the angular velocity range (Limited vs. Not Limited). Constrained and Large scenarios are denoted TR_1 and TR_2 , respectively, in the following:

- TR_1 : A limited $1.2 \times 2.4 \text{ m}^2$ space featuring 2 columns with a diameter of 0.12 m.
- TR_2 : A $5 \times 5 \text{ m}^2$ square area with 10 columns with a diameter of 0.25 m.

Eight different learned policies result from the combinations of these three variables. Their initial performance is summarized in Table 2; they have been exploited in both TR_1 and TR_2 , and we have averaged their reward over 500 episodes. It is important to emphasize that, during these evaluation trials, obstacle positions are always randomized, ensuring that the agent encounters as many distinct situations as possible.

Table 2: Experimental results for 8 training configurations. The table shows average rewards across TR_1 and TR_2 considering the training environment size (C/L), obstacle variability (R/F) and angular range (L/N). The bold value highlights the highest average reward achieved in the TR_2 scene.

Training Scene	Obstacles	Angular Vel. Range	Exp. ID	Reward TR_1	Reward TR_2
Constrained	Randomized	Limited	C-R-L	0.7333	-0.5165
		Not Limited	C-R-N	0.7419	-0.5239
	Fixed	Limited	C-F-L	0.6360	-0.4910
		Not Limited	C-F-N	0.7221	-0.3498
Large	Randomized	Limited	L-R-L	0.4835	0.7390
		Not Limited	L-R-N	0.5190	0.6825
	Fixed	Limited	L-F-L	0.6946	0.8328
		Not Limited	L-F-N	0.5982	0.7402

The quantitative analysis in Table 2 reveals three decisive trends regarding the robot performance. First, the influence of the environment scale is critical for generalization: policies trained in constrained scenarios (C-series) achieve high rewards in their native environment (TR_1) but fail catastrophically when transferred to the larger validation scene (TR_2), consistently yielding negative rewards (e.g., $C - R - N$ drops to -0.5239). In contrast, those trained in large environments (L-series) successfully adapt to smaller ones, demonstrating that environmental scale during training is a prerequisite for robust navigation; the role of the durations of actions in this matter seems to be relevant, since smaller scenarios lead to learning mostly shorter motion times.

Regarding action space optimization, the results confirm that limiting the angular velocity is beneficial. In the large-scale experiments, the *Limited* configurations consistently outperform their *Not Limited* counterparts (e.g., in the fixed obstacle setup, the reward increases from 0.7402 ($L - F - N$) to 0.8328 ($L - F - L$)). This indicates that removing extreme angular commands stabilizes the exploration process, allowing the agent to converge on smoother trajectories. However, such extremes seem still to be useful in the constrained environment TR_1 , where it may be necessary to apply higher angular velocities more often in order to avoid obstacles faster. Notice, again, that this effect is also influenced by the need of shorter motion times in the small scenario.

Finally, contrary to the common assumption that higher stochasticity (i.e., randomization) always yields better robustness, and as we have already commented in Section 3.1.3, the stability of obstacles has played a significant role in maximizing the final reward. The highest overall performance has been achieved by the $L - F - L$ policy (0.8328), surpassing the randomized version $L - R - L$ (0.7390). This suggests that a fixed

configuration of obstacles keeps the variance of the search space adequately bounded and thus facilitates a more solid consolidation of learning for the navigation task (recall that the robot starting and ending poses are randomized).

Regarding the learning dynamics, Fig. 11 illustrates the cumulative reward convergence for the trained policies. It reveals a distinct performance gap between policies trained in constrained environments (C-series, bottom curves) and those in large scenarios (L-series, top curves), validating the hypothesis that spatial scale is the dominant factor for potential reward maximization in our time-aware context, as the robot is severely restricted in constrained scenarios.

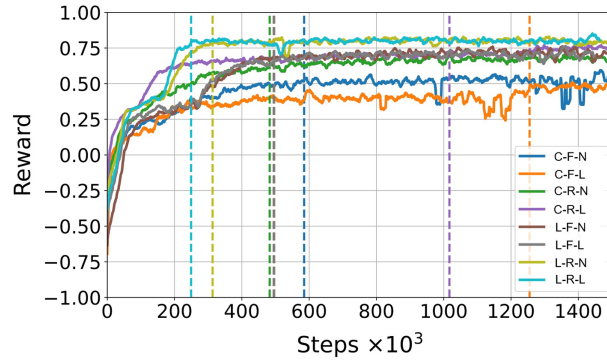


Figure 11: Training reward convergence vs. training steps $\times 10^3$ for the evaluated DRL policies.

Among the top-performing models, $L-R-L$ and $L-R-N$ policies (light blue and yellow lines in Fig. 11) exhibit the fastest convergence, stabilizing around step 250×10^3 and 300×10^3 , respectively. However, the $L-F-L$ and $L-F-N$ policies show a more gradual but steady learning curve, stabilizing later (approx. step 500×10^3). This smoother consolidation correlates with the superior generalization results observed in the validation phase (as shown in Table 2).

Based on all these findings, the $L-F-L$ scenario configuration is selected as the best for training, and consequently it is the one integrated with the proposed navigation framework.

4.2 Evaluation of STMs + PFs for Navigation

Three real scenarios have been studied during the evaluation of STMs and their integration with localization uncertainty, gathered from maps automatically built by a real robot on different spaces of our laboratories and offices, already illustrated in Fig. 2:

- Test scene 1 (TS_1): Two labs divided by a straight wall. Multiple objects are located near the boundaries, while the central space is mainly unobstructed. However, a relevant number of very small obstacles exist around the space (mainly chair and table legs), therefore making localization more difficult.
- Test scene 2 (TS_2): Main floor of a residence, comprising three rooms connected through a short corridor. This map is generally clear, with the exception of some minor obstacles.
- Test scene 3 (TS_3): Tight and elongated corridor which presents some curved but large enough obstacles.

In each scenario, we define 4 different pairs of target and robot starting positions (denoted $Exp.1 \dots Exp.4$ in the figures), and store the results of navigation for each pair. Notice that target-robot position #4 in scenario TS_1 is deliberately hard for our approach: it involves a long path and therefore presents global navigation aspects that a local navigator may not deal with well; it also resides in the scenario that includes

close-to-path obstacles that are much smaller than the ones used in training. We expect that the results in that case show a small percentage of successes.

Some representative trajectories followed by the robot during these experiments are illustrated in Fig. 12 for the first scenario. The rest are in Appendix B so as not to make the document too cumbersome to follow.

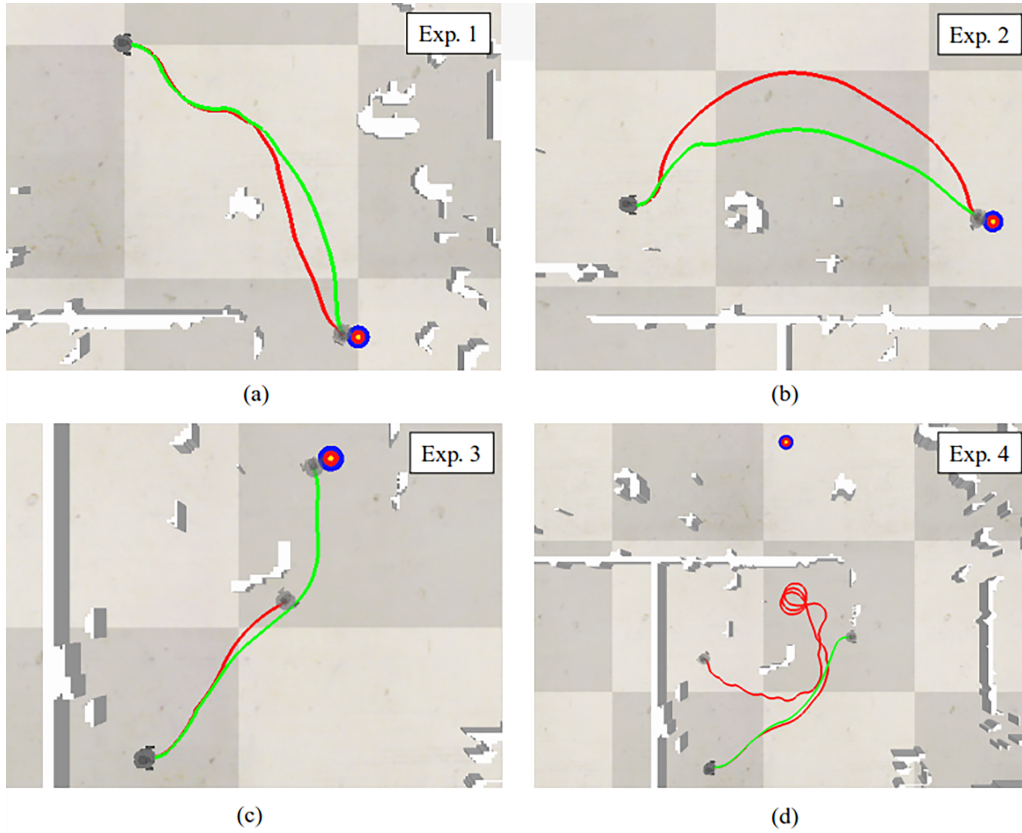


Figure 12: Resulting navigation paths for PF_1 (red) and PF_2 (green) across TS_1 with 4 combinations of target-robot positions: (a) Exp. #1, (b) Exp. #2, (c) Exp. #3, and (d) Exp. #4. Notably, Exp. #3 shows the robot colliding with a large obstacle while using PF_1 due to a bad localization. Exp. #4 shows the difficulties in finding suitable local navigation solutions, as explained in the text: the selected experiment for the figure shows the robot stuck in a local minimum with PF_1 and colliding with a very small obstacle with PF_2 .

To quantify the results obtained in these experiments and thus get a rigorous evaluation of the performance of our approach, the following metrics have been calculated:

- *Success rate*: The percentage of episodes where the robot reached the target without collision within a 100-step limit. Episodes exceeding this limit are considered failures.
- *Total episode time*: Navigation time elapsed from the start of the episode until the target is reached.
- *Idle time percentage*: Proportion of time during which the agent is waiting for motion commands to end, in relation to the total (wall-clock) time employed in the episode. This indicates the potential for time savings either used for reducing computer energy consumption or execution of other tasks.
- *Travelled distance*: An indicator of trajectory optimality. Shorter paths imply higher efficiency, reducing mechanical wear on motors and wheels; however, the robot could achieve this by taking more risks.
- *Minimum distance to obstacles*: A safety metric used to evaluate the agent ability to maintain a secure clearance during navigation.

- *Rotational energy*: An estimate of the energy employed in changing angular speed, calculated at each step k as $|I \cdot (\omega_k^2 - \omega_{k-1}^2)/2|$, where I is the rotational inertia of the robot (considered a cylinder) and w_* the commanded angular speeds. This reflects both the smoothness of the paths and a part of the mechanical energy used for navigation. The other energy component, the one used for maintaining the linear speed (displacing the robot), is not included since it is proportional to the linear velocity (fixed), and therefore to the total episode time, already recorded. The energy consumed for computing is implied in the idle time.
- *Maximum localization variance*: Maximum variance of the particle cloud used for localization in each episode. This serves to detect navigations where the localization has been less reliable and also to detect possible differences between PF_1 and PF_2 according to the explanations given at the beginning of this subsection.

4.2.1 PF_1 and PF_2 Comparisons

For comparing both PF_1 and PF_2 , we have averaged the metrics explained before over 30 iterations for each robot-target combination, and gathered the results in [Tables 3 and 4](#). Bold values indicate the best value in each case, though in many cases those values are quite close to each other for their differences to be statistically significant.

Table 3: Comparative quantitative results for PF_1 and PF_2 . Success requires reaching the target in under 100 steps with a minimum obstacle clearance of ≥ 0.17 m. Metrics are averaged over successful episodes only. Bold values indicate the best result for each evaluated metric.

Map	Exp.	Success Rate (%)		Time (s)		Distance (m)	
		PF_1	PF_2	PF_1	PF_2	PF_1	PF_2
TS_1	1	96.7	100.0	22.5	23.7	3.38	3.55
	2	100.0	100.0	29.0	28.4	4.36	4.26
	3	0.0	60.0	–	20.6	–	3.09
	4	10.0	20.0	43.2	42.8	6.49	6.42
TS_2	1	100.0	100.0	34.8	35.8	5.22	5.36
	2	100.0	100.0	18.7	18.7	2.80	2.80
	3	100.0	100.0	18.5	18.5	2.77	2.78
	4	90.0	100.0	20.5	21.7	3.08	3.25
TS_3	1	100.0	100.0	30.4	30.8	4.55	4.62
	2	100.0	100.0	21.5	21.6	3.22	3.25
	3	100.0	100.0	34.6	35.4	5.19	5.32
	4	100.0	100.0	36.0	36.9	5.39	5.53

Table 4: Continuation of [Table 3](#). As in the previous table, bold values denote the approach with the best performance for each experiment.

Map	Exp.	Idle Time (%)		Min. Dist. (m)		Rotational Energy (10^{-4} J)		Max. Variance	
		PF_1	PF_2	PF_1	PF_2	PF_1	PF_2	PF_1	PF_2
TS_1	1	76.2	74.9	0.27	0.35	217.45	211.78	134.01	132.13
	2	83.5	79.4	0.44	0.44	212.15	210.61	119.29	120.56

(Continued)

Table 4 (continued)

Map	Exp.	Idle Time (%)		Min. Dist. (m)		Rotational Energy (10^{-4} J)		Max. Variance	
		PF_1	PF_2	PF_1	PF_2	PF_1	PF_2	PF_1	PF_2
	3	–	78.7	–	0.18	–	214.73	–	125.08
	4	78.2	74.1	0.19	0.18	237.88	221.98	122.58	134.90
	1	78.1	75.4	0.38	0.39	223.80	211.59	134.87	128.34
	2	79.5	76.0	0.25	0.31	215.59	206.80	100.76	104.92
TS_2	3	76.1	74.9	0.42	0.42	219.60	214.96	92.67	90.74
	4	63.5	60.6	0.24	0.28	232.01	217.41	100.97	101.29
TS_3	1	85.8	80.5	0.48	0.47	215.08	213.38	144.44	143.46
	2	73.9	70.9	0.32	0.31	213.64	210.16	153.04	166.16
	3	83.7	80.3	0.41	0.40	215.37	213.65	158.53	174.33
	4	85.3	81.3	0.53	0.53	203.28	203.19	157.07	160.43

As it can be seen, our expectations on the experiment #4 of scenario TS_1 are met: both methods have a very low probability of success for the reasons already explained: a relevant number of small obstacles that hinder localization and a long path to traverse that makes navigation more global than local (Fig. 12d illustrates such problems). Nevertheless, working with one motion command per particle and then averaging into a representative command, as PF_2 does, instead of working on a direct representative pose of the robot and issuing just one command for it (PF_1), becomes more robust and reliable.

Maybe surprisingly, experiment #3 in the same scenario, which does not have the long path problem, produces even worse results for PF_1 . This does not occur in experiments #1 and #2 of the same scenario, which also suffer from similar relevant localization errors (Fig. 13 shows the localization errors of PF_1 across all experiments and scenarios, where it is clear that TS_1 induces significant issues in this regard). The failures in exp. #3 are due to the fact that the robot has to pass closer to obstacles than in the rest of the experiments of the same scenario, and then the localization error is enough to produce collisions. Notice that, in spite of that, PF_2 obtains much better results than PF_1 in exp. #3, though not entirely optimal (60% of successes), due to its much less dependence on the robot pose estimation.

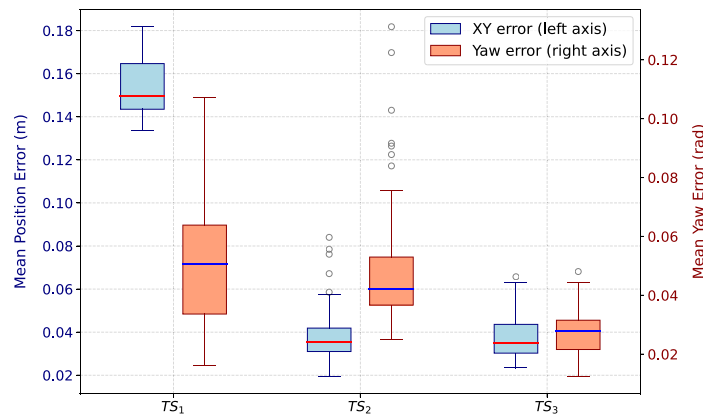


Figure 13: Statistical distribution of cumulative tracking errors across scenarios TS_1 , TS_2 and TS_3 (PF_1 configuration). The left axis displays the Euclidean position error (X , Y), while the right axis shows the yaw error.

The rest of the experiments assess the adequacy of the training environment, which is very general and minimalistic, to a diversity of real situations: in all of them both methods achieve 100% of success or very close to that. PF_1 tends to produce very slightly shorter trajectories (no statistical significance), as seen in the *Time* and *Distance* columns of Table 3.

It is also logical that PF_2 needs more computation time, since it requests many motion commands instead of just one, and therefore its idle times are shorter, as seen in Table 4, although not by much. This implies that we could take advantage of the motion smoothing provided by this method in order to assure success at a quite modest cost regarding efficiency. We have also recorded the computation time needed for PF_1 and PF_2 in all experiments (Fig. 14). These times set a lower bound for the idle times we have available. In the case that our methods would use a fixed timestep heuristically (or intuitively) configured, it should be longer than those values (around 150 or 200 ms, respectively). The resulting navigation will lose all the advantages of our approach though, since there will be no possible way of reacting to different safety situations from a temporal perspective, and thus no good fitting to the particular needs of the task at all moments. In general, with longer idle times (arbitrarily fixed), the risk of collision will increase since the robot will be blind for longer times, while with shorter ones the energy consumption and the possibility of running other processes will be worse.

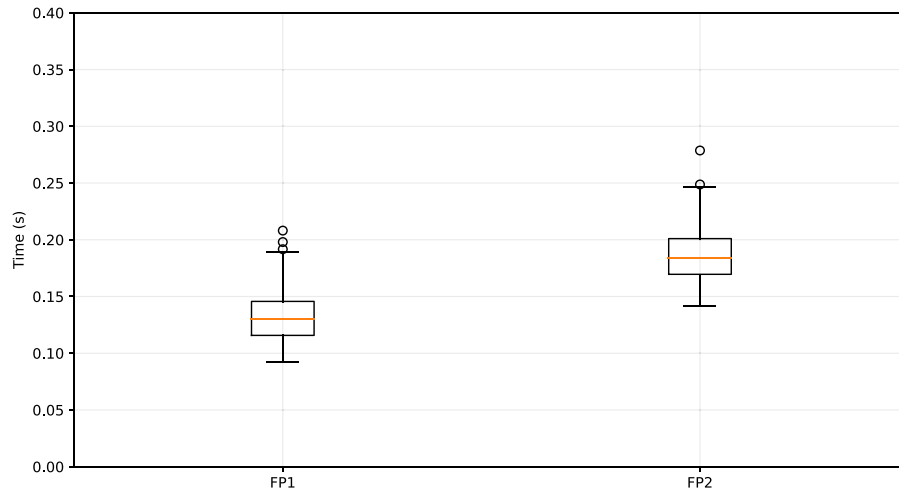


Figure 14: Computation time devoted by PF_1 and PF_2 to processing data before executing the motion command received from the STMs.

As for safety, the recorded minimum distance to obstacles along the trajectories shown in Table 4 is also very similar between both methods, with some scenarios giving a slight advantage to one or another (not statistically significant). PF_1 and PF_2 keep the robot, in general, about 20–50 cm away from collisions, from the center of the robot; considering that its radius is ~ 8 cm, the robot keeps itself far enough away from the minimum distance for deciding a collision (9 cm if considering a radius of 8 cm) in all cases.

Rotational energy is both a measure of energy consumption to turn the robot and of the smoothness of the resulting trajectory (less angular speed changes lead to smoother paths). We see in Table 4, a consistently better behavior in this sense by PF_2 , due to the smoothing of motion commands we have already commented, in the vast majority of scenarios.

Finally, PF_2 and PF_1 show some statistically significant differences in the variance of their particle clouds, mainly due to the different trajectories they produce and consequently the different parts of the environment where they have to localize the robot. Which one has more precise localization varies along the table as we set one or another experiment.

4.2.2 Comparison with Exact Localization and Other Methods

We have also analyzed the performance of our proposed time-aware methods when compared to not depending on probabilistic localization and also to other common local navigators. For those purposes, we have focused on scenario TS_3 and the combination of target-robot position #1 in order to work in a highly successful setting, where localization errors or too long paths do not enter as additional variables. The results are condensed in Table 5.

Table 5: Comparison of PF_1 and PF_2 with an exactly localized robot. Bold values indicate the best result for each evaluated metric.

Map	Exp.	Time (s)			Distance (m)			Min. Dist. (m)		
		PF_1	PF_2	GT	PF_1	PF_2	GT	PF_1	PF_2	GT
TS_3	1	30.4	30.8	30.4	4.55	4.62	4.57	0.48	0.47	0.48

The main results are that the exactly localized robot exhibits a behavior similar to the one of PF_1 , which again is due to the fact that PF_1 uses a more direct and accurate estimate of the pose while PF_2 relies on an indirect smoothing of motion commands; nevertheless, the three approaches provide similar results, with no particular statistical significance. This latter result confirms the good performance of both proposed methods.

We have also compared PF_1 and PF_2 to a common and very popular navigation algorithm, Virtual Force Fields (VFFs), also called Artificial Potential Fields, that could also be used as global planners [73]. In this work, we have used an implementation of VFFs for local navigation, using a fixed timestep of 100 ms, something very reasonable for this kind of robot and its onboard computational power. The obtained results are in Table 6. VFFs follow slightly shorter trajectories (no statistical significance) with a relevantly higher rotational energy, therefore producing motions with more changes in orientation. Notice that VFFs have some parameters that can be adjusted for improving safety and that the timestep can be set to another value to vary the idle time as well, which are the reasons why we limit the study to navigation time and energy consumption.

Table 6: Comparison of PF_1 and PF_2 with a VFFs local navigator. Bold values indicate the best result for each evaluated metric.

Map	Exp.	Time (s)			Rotational Energy (10^{-4} J)		
		PF_1	PF_2	VFF	PF_1	PF_2	VFF
TS_3	1	30.4	30.8	30.3	215.08	213.38	271.85

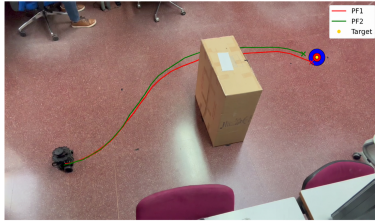
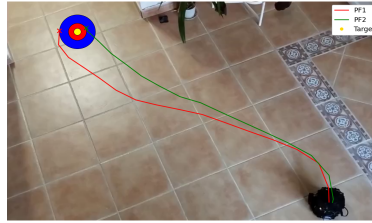
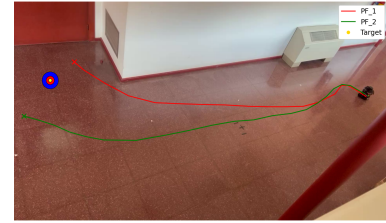
4.2.3 Real Experiments

Finally, we have also used STMs for experiments conducted in real scenarios with the same robot (Turtlebot Burger), both using PF_1 and PF_2 methods.

Table 7 summarizes the metrics measured for the real experiments in TS_3 . Notice how close are these results to the patterns obtained in the simulated experiments. In these real ones both methods differ somehow in the trajectory length and duration, as Fig. 15 shows. In that figure we have collected snapshots of these experiments and the resulting trajectories. PF_2 is shown to be superior to PF_1 here as well, reaching the target more directly.

Table 7: Performance of PF_1 and PF_2 in real scenarios. Bold values indicate the best result for each evaluated metric.

Map	Exp.	Min. Dist. (m)				Rotational Energy (10^{-4} J)				Max. Variance			
		PF_1		PF_2		PF_1		PF_2		PF_1		PF_2	
		Sim	Real	Sim	Real	Sim	Real	Sim	Real	Sim	Real	Sim	Real
TS_3	1	0.48	0.47	0.47	0.44	215.08	215.21	213.38	213.86	144.44	146.50	143.46	135.45

(a) Scenario TS_1 (b) Scenario TS_2 (c) Scenario TS_3 **Figure 15:** Real-world experiments comparing the PF_1 (red) and PF_2 (green) methods across the evaluated scenarios: TS_1 , TS_2 , and TS_3 .

5 Conclusions

This work has presented a novel subjective, time-aware navigation framework that integrates Deep Reinforcement Learning (DRL) with probabilistic localization in order to enhance navigation safety and efficiency of a particular robot in structured environments. The core contribution of this study is the introduction of a new kind of spatio-temporal metric map, so-called Subjective Time Maps (STMs), a representational tool that characterizes the environment not merely as a geometric space, but as a temporal cost surface dependent on the relative pose between the robot and the navigation target.

The proposed experimental setup, featuring a bespoke multi-ring target configuration and a sparse reward function, successfully enables a SAC agent to learn a policy that balances spatial accuracy with temporal efficiency. The analysis of the generated STMs confirms that the learned policy exhibits a subjective perception of risk: shorter timestep values (indicating caution) are dynamically assigned to surrounding obstacles only when the robot is facing them, whereas the same obstacles are treated as low-priority constraints when the robot is oriented towards the goal. This directional sensitivity validates the hypothesis that effective navigation requires assessing risk relative to the agent's immediate trajectory rather than relying solely on objective maps.

Regarding the integration of the localization uncertainty with the one included in the decision-making navigator, our results validate that while DRL policies can generalize against noise, they fundamentally lack an explicit mechanism to quantify state uncertainty. By bridging this gap with a mathematically rigorous belief representation method (particle filters), we ensure that navigation is informed by a reliable measure of confidence rather than relying solely on learned heuristics. This approach can be implemented in two basic ways: with a first-moment pose estimator (PF_1) or by integrating multiple motion commands into the particle filter (PF_2). Both methods provide similar results in general, though PF_2 shows special robustness in hard situations where localization estimates are not good enough, can increase the probability of solving global navigation problems, and produces smoother trajectories (smaller orientation changes).

We have also compared our methods to a common Virtual Force Field local navigator and to clairvoyant localization; the comparison with the former shows that our approaches generate smoother trajectories,

while the latter leads to the conclusion that they produce very good results even with a modest number of particles in particle filter localization. We have finally tested the methods in different real scenarios, and show how the results replicate the same performance patterns as in the simulated environments. All of these analyses have been based on several performance metrics for a multi-criteria evaluation of the proposal.

Crucially, the explicit management of the timestep τ implies a significant reduction in the computational load compared to fixed-frequency planners. By adapting the decision frequency to the environmental complexity, our framework minimizes the processing requirements, making it particularly suitable for resource-constrained systems. This efficiency not only translates into energy savings but also frees up computational capacity for parallel tasks.

Future lines of research will focus on a more in-depth analysis of the real-time aspects we have found in the implementation of our methods. Also, their adaptation to global navigation, i.e., path planning, is of relevance: there are several ways to extract complete paths from STMs under the reported optimality criteria, such as RRTs that use the information in the map for selecting motion commands, relaxation of mathematical paths over the safety “valleys” in the STM, or even finding suitable abstractions that enable its integration with symbolic planners.

Additionally, the computational idle time generated by the proposed approach opens new avenues for Continuous Learning, allowing the agent to refine its policy online and improve performance continuously without disrupting the navigation task.

Finally, the inclusion of variable linear speeds in the DRL policies and therefore in the STMs is of great interest, since having variable linear velocity should improve the navigation capabilities of the robot.

Acknowledgement: The authors thank the technical support of IMECH.UMA through PPRO-IUI-2023-02.

Funding Statement: This work was supported by the TYRELL research project (<https://babel.isa.uma.es/research/projects/tyrell/>) (PID2023-147392NB-I00), funded by MICIUAEI10.13039501100011033 and by the EU Regional Development Funds (EDRF).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization: Juan-Antonio Fernández-Madrigal, Vicente Arévalo-Espejo, Adrián Bañuls-Arias; methodology: Adrián Bañuls-Arias, Juan-Antonio Fernández-Madrigal; software: Adrián Bañuls-Arias, Cipriano Galindo, Vicente Arévalo-Espejo, Ana Cruz-Martín; validation: Adrián Bañuls-Arias, Cipriano Galindo, Ana Cruz-Martín; investigation: Manuel Castellano-Quero, Juan M. Gandarias; resources: Juan-Antonio Fernández-Madrigal, Juan M. Gandarias; writing—original draft preparation: Adrián Bañuls-Arias, Vicente Arévalo-Espejo, Manuel Castellano-Quero, Juan M. Gandarias; writing—review and editing: Adrián Bañuls-Arias, Juan-Antonio Fernández-Madrigal, Vicente Arévalo-Espejo, Manuel Castellano-Quero, Juan M. Gandarias, Ana Cruz-Martín; supervision: Juan-Antonio Fernández-Madrigal; project administration: Juan-Antonio Fernández-Madrigal, Juan M. Gandarias, Ana Cruz-Martín; funding acquisition: Juan-Antonio Fernández-Madrigal, Juan M. Gandarias. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The source code for the experiments described in the paper is publicly available at https://github.com/uncore-team/subjective_time_maps.git.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A Supplementary SMTs for TS_2 and TS_3 Scenes

In this appendix we collect the STMs generated for TS_2 and TS_3 scenes, including configurations where the robot faces the target as well as the opposite direction, as detailed in [Section 3.2](#).

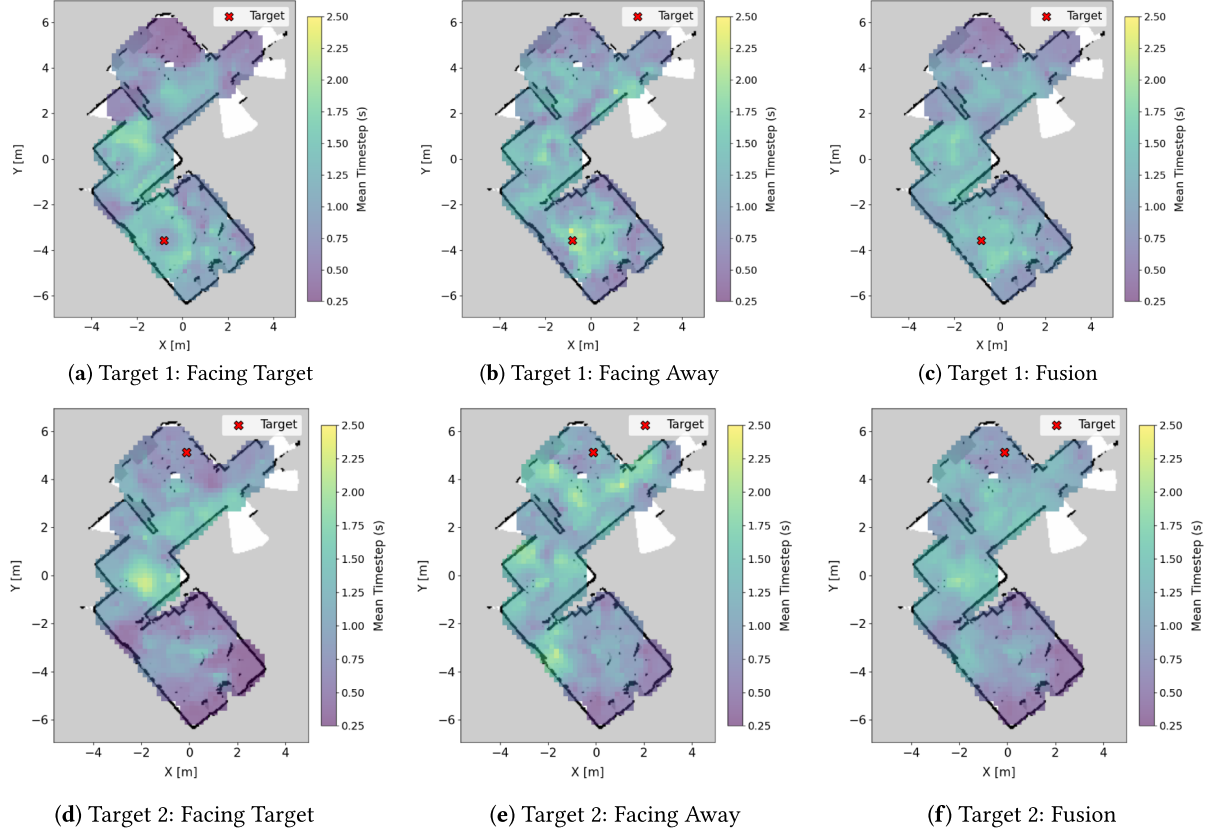


Figure A1: STM visualization (average timestep) for TS_2 across Target 1 (top) and Target 2 (bottom). Columns: facing target (a,d), facing away (b,e), and fused map (c,f).

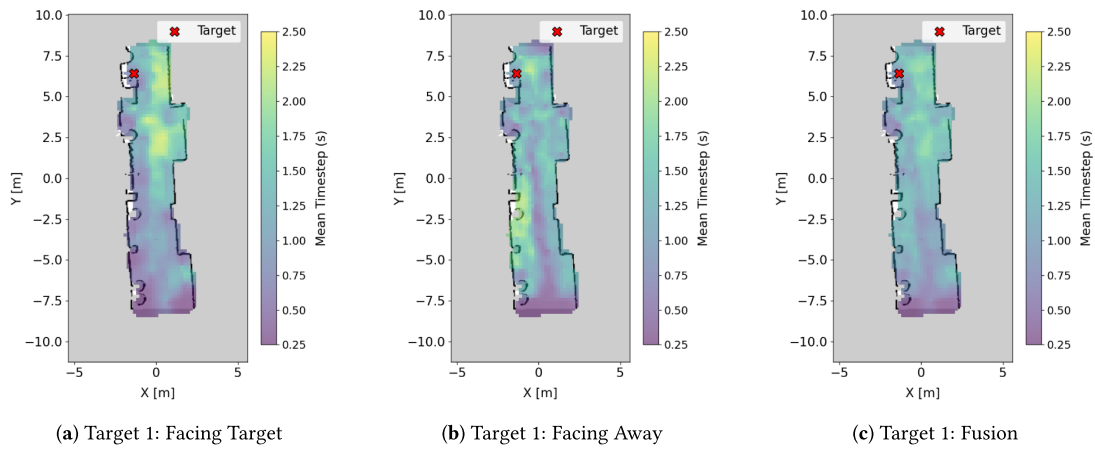


Figure A2: (Continued)

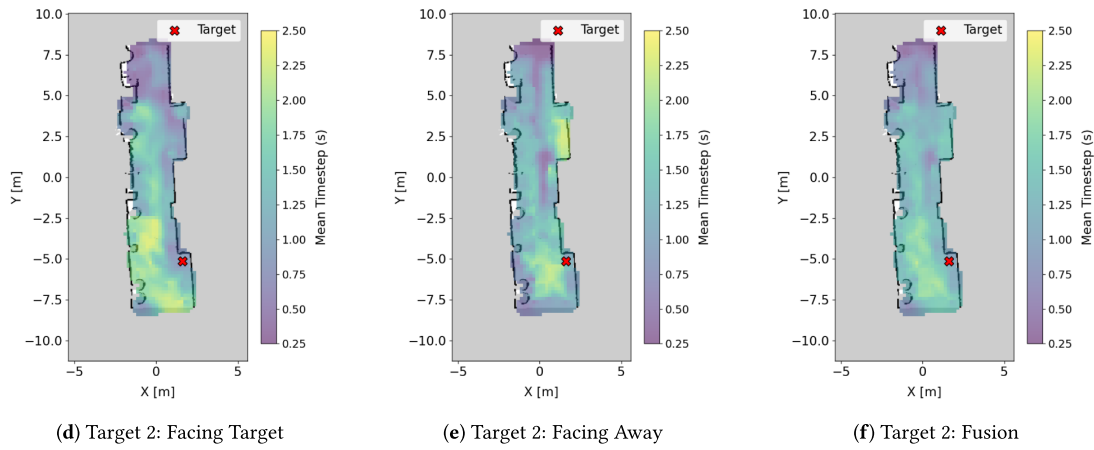


Figure A2: STM visualization (average timestep) for TS_3 across Target 1 (top) and Target 2 (bottom). Columns: facing target (a,d), facing away (b,e), and fused map (c,f).

Appendix B Trajectory Comparisons for TS_1 and TS_3 Maps

This appendix presents a comparative visualization of the robot trajectories executed in the TS_2 and TS_3 scenes for four different selected targets in each map. For each configuration, both the PF_1 and PF_2 methods have been evaluated, as analyzed in [Section 4.2](#).

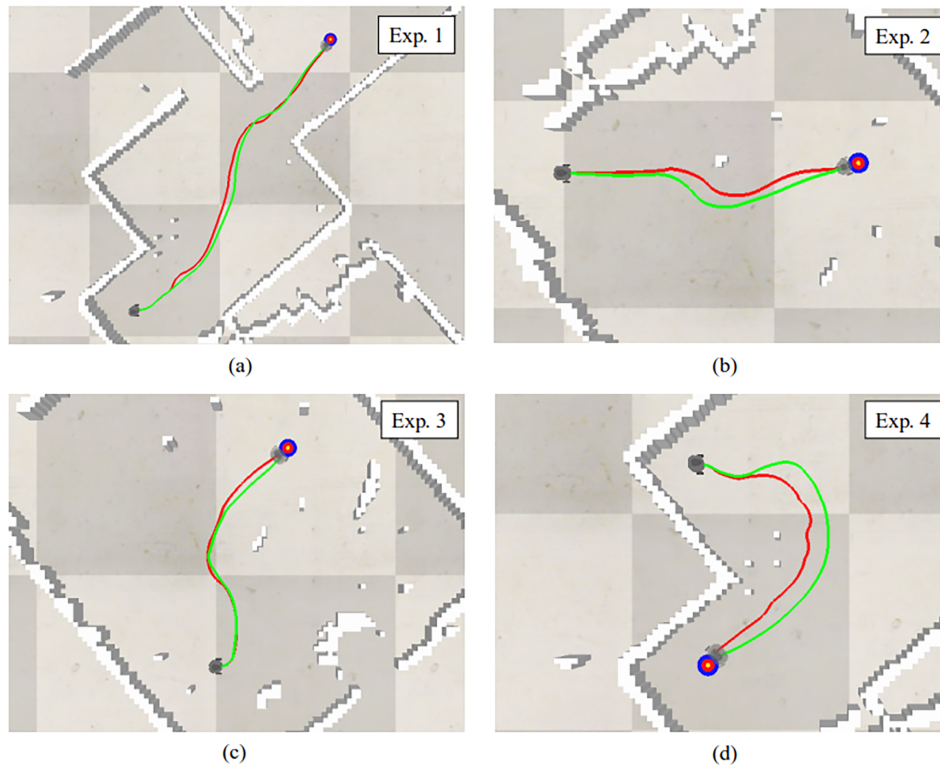


Figure A3: Resulting navigation paths for PF_1 (red) and PF_2 (green) across TS_2 , for (a) Exp. #1, (b) Exp. #2, (c) Exp. #3, and (d) Exp. #4. In this scenario the smoother trajectories produced by PF_2 are more evident (lower rotational energy).

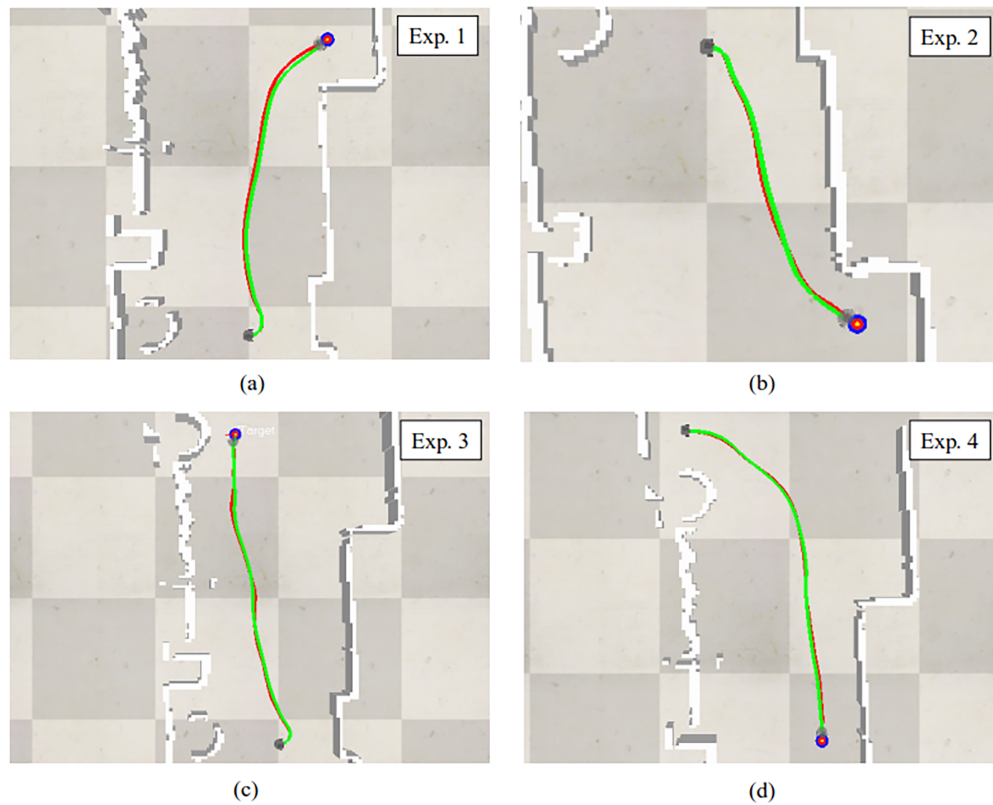


Figure A4: Resulting navigation paths for PF_1 (red) and PF_2 (green) across TS_3 , for (a) Exp. #1, (b) Exp. #2, (c) Exp. #3, and (d) Exp. #4. In this scenario the STMs are more linear along the robot path and the localization estimates are accurate (little bias), therefore both methods behave very similarly.

References

1. Fernández-Madrigal JA, Blanco-Claraco JL. Simultaneous localization and mapping for mobile robots: introduction and methods. Hershey, PA, USA: IGI Global; 2012.
2. Siciliano B, Khatib O. Springer handbook of robotics. 2nd. Berlin/Heidelberg, Germany: Springer; 2016.
3. Waga A, Benhlila S, Bekri A, Abdouni J, Saber FZ. A survey on autonomous navigation for mobile robots: from traditional techniques to deep learning and large language models. J King Saud Univ Comput Inf Sci. 2025;37(7):198. doi:10.1007/s44443-025-00216-x.
4. Macenski S, Moore T, Lu DV, Merzlyakov A, Ferguson M. From the desks of ROS maintainers: a survey of modern & capable mobile robotics algorithms: progress, challenges, and research frontiers. Robot Auton Syst. 2023;169:104493. doi:10.1016/j.robot.2023.104493.
5. Singamaneni PT, Bachiller-Burgos P, Manso LJ, Garrell A, Sanfeliu A, Spalanzani A, et al. A survey on socially aware robot navigation: taxonomy and future challenges. Int J Robot Res. 2024;43(10):1533–72. doi:10.1177/02783649241230562.
6. Hwang S, Jang I, Kim D, Kim HJ. Safe motion planning and control for mobile robots: a survey. Int J Control Autom Syst. 2024;22(10):2955–69. doi:10.1007/s12555-024-0784-5.
7. Zhu K, Zhang T. Deep reinforcement learning based mobile robot navigation: a review. Tsinghua Sci Technol. 2021;26(5):674–91. doi:10.26599/tst.2021.9010012.
8. Zhu Y, Wan Hasan WZ, Harun Ramli HR, Norsahperi NMH, Mohd Kassim MS, Yao Y. Deep reinforcement learning of mobile robot navigation in dynamic environment: a review. Sensors. 2025;25(11):3394. doi:10.3390/s25113394.

9. Xu T. Recent advances in rapidly-exploring random tree: a review. *Heliyon*. 2024;10(11):e32451. doi:10.1016/j.heliyon.2024.e32451.
10. Alexander A, Venkatesan K, Mounsef J, Ramanujam K. A comprehensive survey of path planning algorithms for autonomous systems and mobile robots: traditional and modern approaches. *IEEE Access*. 2025;13(4):176287–326. doi:10.1109/access.2025.3618837.
11. Vintr T, Blaha J, Ulrich J, Rouček T, Broughton G, Duckett T, et al. Estimating the dominant frequencies of real-valued cyclic processes in natural environments for long-term operation of autonomous robots. *Robot Auton Syst*. 2026;198(1):105299. doi:10.1016/j.robot.2025.105299.
12. Tunstel E, Seraji H, Howard A. Soft computing approach to safe navigation of autonomous planetary rovers. In: *intelligent control systems using soft computing methodologies*. Boca Raton, FL, USA: CRC Press; 2001. p. 213–42.
13. Lam CP, Chou CT, Chiang KH, Fu LC. Human-centered robot navigation—towards a harmoniously human-robot coexisting environment. *IEEE Trans Robot*. 2010;27(1):99–112. doi:10.1109/tro.2010.2076851.
14. Macek K, Vasquez D, Fraichard T, Siegwart R. Safe vehicle navigation in dynamic urban scenarios. In: *Proceedings of the 2008 11th International IEEE Conference on Intelligent Transportation Systems*; 2008 Oct 12–15; Beijing, China. New York, NY, USA: IEEE; 2008. p. 482–9.
15. Petti S, Fraichard T. Safe motion planning in dynamic environments. In: *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*; 2005 Aug 2–6; Edmonton, AB, Canada. New York, NY, USA: IEEE; 2005. p. 2210–5.
16. Fraichard T, Asama H. Inevitable collision states—a step towards safer robots? *Adv Robot*. 2004;18(10):1001–24. doi:10.1163/1568553042674662.
17. Kewlani G, Ishigami G, Iagnemma K. Stochastic mobility-based path planning in uncertain environments. In: *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*; 2009 Oct 10–15; Louis, MO, USA. New York, NY, USA: IEEE; 2009. p. 1183–9.
18. Cheng P, Shen Z, LaValle SM. RRT-based trajectory design for autonomous automobiles and spacecraft. *Arch Control Sci*. 2001;3(4):167–94.
19. Isukapalli SS. Uncertainty analysis of transport-transformation models. Piscataway, NJ, USA: Rutgers The State University of New Jersey, School of Graduate Studies; 1999.
20. Pepy R, Lambert A. Safe path planning in an uncertain-configuration space using RRT. In: *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*; 2006 Oct 9–15; Beijing, China. New York, NY, USA: IEEE; 2006. p. 5376–81.
21. Castillo-Lopez M, Ludvig P, Sajadi-Alamdari SA, Sanchez-Lopez JL, Olivares-Mendez MA, Voos H. A real-time approach for chance-constrained motion planning with dynamic obstacles. *IEEE Robot Autom Lett*. 2020;5(2):3620–5. doi:10.1109/lra.2020.2975759.
22. Kollmitz M, Hsiao K, Gaa J, Burgard W. Time dependent planning on a layered social cost map for human-aware robot navigation. In: *Proceedings of the 2015 European Conference on Mobile Robots (ECMR)*; 2015 Sep 2–4; Lincoln, UK. New York, NY, USA: IEEE; 2015. p. 1–6.
23. Ravankar A, Ravankar AA, Hoshino Y, Watanabe M, Kobayashi Y. Safe mobile robot navigation in human-centered environments using a heat map-based path planner. *Artif Life Robot*. 2020;25(2):264–72. doi:10.1007/s10015-020-00591-w.
24. Hafez OA, Joerger M, Spenko M. How safe is particle filtering-based localization for mobile robots? An integrity monitoring approach. *IEEE Trans Robot*. 2024;40:3372–87. doi:10.1109/tro.2024.3420798.
25. Bockrath K, Ernst L, Nadeem R, Pedraza B, Dera D. Trustworthy navigation with variational policy in deep reinforcement learning. *Front Robot AI*. 2025;12:1652050. doi:10.3389/frobt.2025.1652050.
26. Nguyen TP, Nguyen H, Thinh Ngo HQ. Planning the emergency collision avoidance strategy based on personal zones for safe human-machine interaction in smart cyber-physical system. *Complexity*. 2022;2022(1):2992379. doi:10.1155/2022/2992379.
27. Candela A, Wettergreen D. An approach to science and risk-aware planetary rover exploration. *IEEE Robot Autom Lett*. 2022;7(4):9691–8. doi:10.1109/lra.2022.3191949.

28. Xiao X, Dufek J, Murphy RR. Robot risk-awareness by formal risk reasoning and planning. *IEEE Robot Autom Lett.* 2020;5(2):2856–63. doi:10.1109/lra.2020.2974434.
29. Stone RF, Wang J, Sha Z. Risk-bounded and probabilistic roadmap-based motion planner for arbitrarily shaped robots with uncertainty. *J Comput Inf Sci Eng.* 2025;25(8):081002. doi:10.1115/1.4068407.
30. Stracca E, Rudenko A, Palmieri L, Salaris P, Castri L, Mazzi N, et al. DARKO-Nav: hierarchical risk and context-aware robot navigation in complex intralogistic environments. In: Huber M, Verl A, Kraus W, editors. *European robotics forum 2025*. Cham, Switzerland: Springer Nature; 2025. p. 155–61.
31. Kucner TP, Lilienthal AJ, Magnusson M, Palmieri L, Swaminathan CS. Probabilistic mapping of spatial motion patterns for mobile robots. Cham, Switzerland: Springer Nature; 2020.
32. Molina S, Mannucci A, Magnusson M, Adolfsson D, Andreasson H, Hamad M, et al. The ILIAD safety stack: human-aware infrastructure-free navigation of industrial mobile robots. *IEEE Robot Autom Mag.* 2024;31(3):48–59. doi:10.1109/mra.2023.3296983.
33. Castellano-Quero M, Kucner TP, Magnusson M. Alignability maps for the prediction and mitigation of localization error. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop on Closing the Loop on Localization*; 2023 Oct 1–5; Detroit, MI, USA.
34. Hu H, Wen X, Hu J, Chen H, Xia C, Zhang H. Safe trajectory planning for incremental robots based on a spatiotemporal variable-step-size A* algorithm. *Sensors.* 2024;24(11):3639. doi:10.3390/s24113639.
35. Zhang S, Swevers J. Time-optimal point-to-point motion planning: a two-stage approach. *IFAC-PapersOnLine.* 2024;58(18):139–45. doi:10.1016/j.ifacol.2024.09.022.
36. Taylor S, Linton C, Biggs J, Holderbaum W. Time parametrized motion planning. *Mathematics.* 2024;12(21):3404. doi:10.3390/math12213404.
37. Faroni M, Beschi M, Pedrocchi N. Safety-aware time-optimal motion planning with uncertain human state estimation. *IEEE Robot Autom Lett.* 2022;7(4):12219–26. doi:10.1109/lra.2022.3211493.
38. Bagnell JA, Bradley D, Silver D, Sofman B, Stentz A. Learning for autonomous navigation. *IEEE Robot Autom Mag.* 2010;17(2):74–84. doi:10.1109/mra.2010.936946.
39. Tang Y, Zhao C, Wang J, Zhang C, Sun Q, Zheng WX, et al. Perception and navigation in autonomous systems in the era of learning: a survey. *IEEE Trans Neural Netw Learn Syst.* 2022;34(12):9604–24. doi:10.1109/tnnls.2022.3167688.
40. Nahavandi S, Alizadehsani R, Nahavandi D, Mohamed S, Mohajer N, Rokouzzaman M, et al. A comprehensive review on autonomous navigation. *ACM Comput Surv.* 2025;57(9):1–67. doi:10.1145/3727642.
41. Al-Kamil SJ, Szabolcsi R. Optimizing path planning in mobile robot systems using motion capture technology. *Results Eng.* 2024;22(12):102043. doi:10.1016/j.rineng.2024.102043.
42. Al-Kamil SJ, Laufer E, Szabolcsi R. Deep-adaptive fuzzy predictive navigation framework for stable and intelligent mobile robot control. *Results Control Optim.* 2026;22(11):100669. doi:10.1016/j.rico.2026.100669.
43. Pérez-D'Arpino C, Liu C, Goebel P, Martín-Martín R, Savarese S. Robot navigation in constrained pedestrian environments using reinforcement learning. In: *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA)*; 2021 May 30–Jun 5; Xi'an, China. New York, NY, USA: IEEE; 2021. p. 1140–6.
44. Li B, Huang Z, Chen TW, Dai T, Zang Y, Xie W, et al. MSN: mapless short-range navigation based on time critical deep reinforcement learning. *IEEE Trans Intell Transp Syst.* 2022;24(8):8628–37. doi:10.1109/tits.2022.3192480.
45. Honda K, Yonetani R, Nishimura M, Kozuno T. When to replan? an adaptive replanning strategy for autonomous navigation using deep reinforcement learning. In: *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA)*; 2024 May 13–17; Yokohama, Japan. New York, NY, USA: IEEE; 2024. p. 6650–6.
46. Zhang K, Niroui F, Ficocelli M, Nejat G. Robot navigation of environments with unknown rough terrain using deep reinforcement learning. In: *Proceedings of the 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*; 2018 Aug 6–8; Philadelphia, PA, USA. New York, NY, USA: IEEE; 2018. p. 1–7.
47. Sun S, Zhao X, Li Q, Tan M. Inverse reinforcement learning-based time-dependent A* planner for human-aware robot navigation with local vision. *Adv Robot.* 2020;34(13):888–901. doi:10.1080/01691864.2020.1753569.
48. Hsu KC, Hu H, Fisac JF. The safety filter: a unified view of safety-critical control in autonomous systems. *Annu Rev Control Robot Auton Syst.* 2023;7(1):47–72. doi:10.1146/annurev-control-071723-102940.

49. Puterman ML. Markov decision processes: discrete stochastic dynamic programming. 1st ed. Hoboken, NJ, USA: John Wiley & Sons; 2014.
50. Chen X, Wang J. Inhomogeneous deep Q-network for time sensitive applications. *Artif Intell.* 2022;312(7540):103757. doi:10.1016/j.artint.2022.103757.
51. Wang D, Beltrame G. Variable time step reinforcement learning for robotic Applications. arXiv: 2407.00290. 2024. doi:10.48550/arXiv.2407.00290.
52. Karimi A, Jin J, Luo J, Mahmood AR, Jagersand M, Tosatto S. Dynamic decision frequency with continuous options. arXiv:2212.04407. 2023. doi:10.48550/arXiv.2212.04407.
53. Zhou H, Huang A, Azzadenesheli K, Childers D, Lipton Z. Timing as an action: learning when to observe and act. In: Dasgupta S, Mandt S, Li Y, editors. *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*. Vol. 238. New York, NY, USA: PMLR; 2024. p. 3979–87.
54. Mahmood AR, Korenkevych D, Komer B, Bergstra J. Setting up a reinforcement learning task with a real-world robot. In: *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; 2018 Oct 1–5; Madrid, Spain. p. 4635–40. doi:10.1109/IROS.2018.8593894.
55. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv:1801.01290. 2018. doi:10.48550/arXiv.1801.01290.
56. Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, Tan J, et al. Soft actor-critic algorithms and applications. arXiv: 1812.05905. 2019. doi:10.48550/arXiv.1812.05905.
57. Kuo PH, Huang CT, Chang CW, Feng PH, Lin YS. Design and implementation of a soft Actor-Critic controller for a robotic arm. *Eng Appl Artif Intell.* 2025;151:110589. doi:10.1016/j.engappai.2025.110589.
58. Zhang K, Xu Y, Zhu J. A SAC-based deep reinforcement learning approach for autonomous underwater vehicle combat. In: *Proceedings of the 2023 5th International Conference on Robotics, Intelligent Control and Artificial Intelligence (RICAI)*; 2023 Dec 1–3; Hangzhou, China. p. 95–9.
59. de Jesus JC, Kich VA, Kolling AH, Grando RB, de Souza Leite Cuadros MA, Gamarra DFT. Soft actor-critic for navigation of mobile robots. *J Intell Robot Syst.* 2021;102(2):31. doi:10.1007/s10846-021-01367-5.
60. Robotics C. Coppeliasim. 2024 [cited 2025 May 8]. Available from: <https://www.coppeliarobotics.com/>.
61. ROBOTIS. TurtleBot3 e-Manual. 2023 [cited 2026 Jan 16]. Available from: <https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/>.
62. UnCORE. RL spin decoupler. GitHub. 2026 [cited 2026 May 10]. Available from: https://github.com/uncore-team/rl_spin_decoupler.
63. Towers M, Kwiatkowski A, Terry J, Balis JU, Cola GD, Deleu T, et al. Gymnasium: a standard interface for reinforcement learning environments. arXiv:2407.17032. 2024. doi:10.48550/arXiv.2407.17032.
64. Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N. Stable-baselines3: reliable reinforcement learning implementations. *J Mach Learn Res.* 2021;22:1–8.
65. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467. 2016. doi:10.48550/arXiv.1603.04467.
66. Slaoui RB, Clements WR, Foerster JN, Toth S. Robust domain randomization for reinforcement learning. 2020 [cited 2026 May 12]. Available from: <https://openreview.net/forum?id=HlxSOTVtvH>.
67. Bengio Y, Louradour J, Collobert R, Weston J. Curriculum learning. In: *Proceedings of the International Conference on Machine Learning*; 2009 Jun 14–18; Montreal, QC, Canada. p. 41–8. doi:10.1145/1553374.1553380.
68. IMECHUMA. University research institute in mechatronics engineering and cyber-physical systems, University of Málaga, Spain. 2025 [cited 2025 Jul 14]. Available from: <https://www.imech-uma.es/>.
69. Shepard D. A two-dimensional interpolation function for irregularly-spaced data. In: *Proceedings of the 1968 23rd ACM National Conference*. New York, NY, USA: ACM; 1968. p. 517–24.
70. Lesort T, Lomonaco V, Stoian A, Maltoni D, Filliat D, Díaz-Rodríguez N. Continual learning for robotics: definition, framework, learning strategies, opportunities and challenges. arXiv:1907.00182. 2019. doi:10.48550/arXiv.1907.00182.
71. Thrun S, Burgard W, Fox D. Probabilistic robotics. Cambridge, MA, USA: MIT Press; 2005.

72. Burgard W, Fox D, Thrun S. Markov localization for mobile robots in dynamic environments. *J Artif Intell Res.* 1999;11:391–427. doi:10.1613/jair.616.
73. Liu L, Wang X, Yang X, Liu H, Li J, Wang P. Path planning techniques for mobile robots: review and prospect. *Expert Syst Appl.* 2023;227(4):120254. doi:10.1016/j.eswa.2023.120254.