Automatica 173 (2025) 112105

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Hybrid and oriented harmonic potentials for safe task execution in unknown environment☆

Shuaikang Wang, Meng Guo *

Department of Mechanics and Engineering Science, College of Engineering, Peking University, Beijing 100871, China

ARTICLE INFO

Article history:
Received 12 June 2023
Received in revised form 21 October 2024
Accepted 5 November 2024
Available online xxxx

ABSTRACT

Harmonic potentials provide globally convergent potential fields that are provably free of local minima. Due to its analytical format, it is particularly suitable for generating safe and reliable robot navigation policies. However, for complex environments that consist of a large number of overlapping non-sphere obstacles, the computation of associated transformation functions can be tedious. This becomes more apparent when: (i) the workspace is initially unknown and the underlying potential fields are updated constantly as the robot explores it; (ii) the high-level mission consists of sequential navigation tasks among numerous regions, requiring the robot to switch between different potentials. Thus, this work proposes an efficient and automated scheme to construct harmonic potentials incrementally online as guided by the task automaton. A novel two-layer harmonic tree (HT) structure is introduced that facilitates the hybrid combination of oriented search algorithms for task planning and harmonic-based navigation controllers for non-holonomic robots. Both layers are adapted efficiently and jointly during online execution to reflect the actual feasibility and cost of navigation within the updated workspace. Global safety and convergence are ensured both for the high-level task plan and the low-level robot trajectory. Known issues such as oscillation or long-detours for purely potential-based methods and sharp-turns or high computation complexity for purely search-based methods are prevented. Extensive numerical simulation and hardware experiments are conducted against several strong baselines.

1. Introduction

Autonomous robots can replace humans to operate and accomplish complex missions in hazardous environments. However, it is a demanding engineering task to ensure both the safety and efficiency during execution, especially when the environment is only partially known. First, the control strategy that drives the robot from an initial state to the goal state while staying within the allowed workspace (see e.g., Karaman & Frazzoli, 2011; Khatib, 1999; Koditschek, 1987; LaValle, 2006), should be reactive to the newly-discovered obstacles online. Second, the planning method that decomposes and schedules sub-tasks (see e.g., Fainekos, Girard, Kress-Gazit, & Pappas, 2009; Ghallab, Nau, & Traverso, 2004) should be adaptive to the actual feasibility and cost of sub-tasks given the updated environment. Existing work often ignores the close dependency of these two modules and treats them separately, which can lead to inefficient or even unsafe executions, as also motivated in Garrett et al. (2021) and Kim, Shimanuki, Kaelbling, and Lozano-Pérez (2022). How to construct a fully integrated task and motion planning scheme with provable safety and efficiency guarantee within unknown environments still remains challenging, see Loizou and Rimon (2022) and Rousseas, Bechlioulis, and Kyriakopoulos (2022b).

1.1. Related work

As the most relevant to this work, the method of artificial potential fields from Khatib (1986), Panagou (2014), Rousseas, Bechlioulis, and Kyriakopoulos (2022a) and Warren (1989) introduces an intuitive yet powerful framework for tackling the safety and convergence property during navigation. The main idea is to introduce attractive potentials to the goal state and repulsive potentials from obstacles and the workspace boundary. However, naive design of these potentials would introduce undesired local minima, where the combined forces are zero and thus prevents further progress. Navigation functions (NF) pioneered by Koditschek (1987) provably guarantee that such minima are saddle points and more importantly of measure zero. Although the underlying static workspace could be as general as *forest of*

☆ This work was supported by the National Natural Science Foundation of China (NSFC) under grants 62203017, T2121002, U2241214; and by the Fundamental Research Funds for the central universities, China. The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Dimitra Panagou under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: wangs@pku.edu.cn (S. Wang), meng.guo@pku.edu.cn (M. Guo).

https://doi.org/10.1016/j.automatica.2024.112105

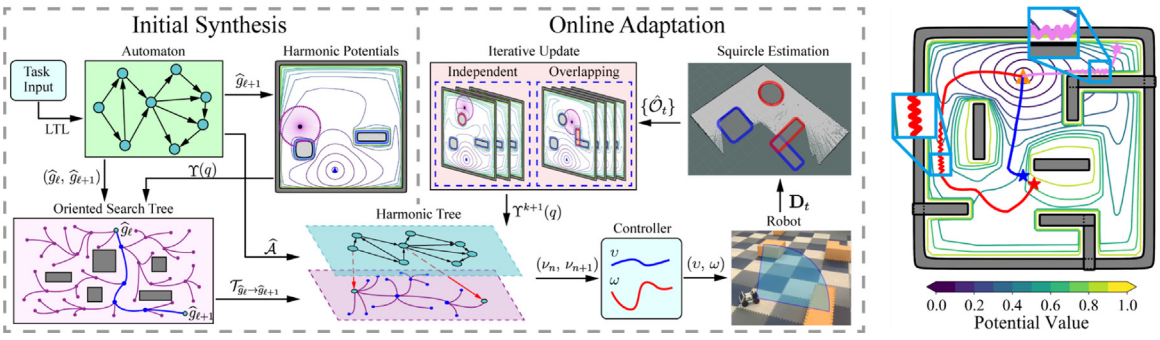


Fig. 1. Left: Illustration of the proposed framework, which consists of the initial synthesis, the online adaption of the task plan and harmonic potentials, and the squircle estimation. **Right**: Oscillations and long detours might occur via classic navigation functions as shown in the red, violet and blue trajectories. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

stars, some key design parameters require fine-tuning for the safety and convergence properties to hold. The work in Fan, Liu, Zhang, and Xu (2022) employs the conformal transformations to map the multiply-connected workspaces to a sphere world without any tuning parameter, which however requires a numerical solution of continuous integrals. The work in Dahlin and Karayiannidis (2023a), Huber, Billard, and Slotine (2019) and Huber, Slotine, and Billard (2024) utilizes reactive potentials to guide dynamic systems around obstacles. Model predictive control has been adopted in Dahlin and Karayiannidis (2023b), Faulwasser and Findeisen (2015), Sánchez, D'Jorge, Raffo, González, and Ferramosca (2021) and Yu, Li, Chen, and Allgöwer (2015) to track the derived potentials. Moreover, harmonic potentials proposed in Kim and Khosla (1992), Loizou (2011, 2017) and Vlantis, Vrohidis, Bechlioulis, and Kyriakopoulos (2018) alleviate such limitations by introducing a novel transformation scheme from obstacle-cluttered environments to point worlds, while retaining these properties. Furthermore, recent work in Rousseas, Bechlioulis, and Kyriakopoulos (2021) and Rousseas et al. (2022a, 2022b) resolves the need for a diffeomorphic mapping onto sphere disks, by adopting a wider set of basis functions for workspace boundaries. Nevertheless, such methods require solving numerous complex parametric optimizations, instead of an analytic solution. Lastly, despite of their global convergence guarantee, there are several notable limitations as illustrated in Fig. 1: (i) oscillations or jitters may appear especially when the trajectory slides along the boundary of obstacles or crosses narrow passages; (ii) drastically different trajectories may develop within the same potential field when the initial pose is changed slightly; (iii) the resulting trajectory is far from the optimal one in terms of trajectory length or control efforts; (iv) the final orientation at the goal pose cannot be controlled freely.

Moreover, sampling-based search methods, such as RRT* in Karaman and Frazzoli (2011), PRM in Hsu, Latombe, and Kurniawati (2006), FMT* in Janson, Schmerling, Clark, and Pavone (2015), have become the dominant paradigm to tackle high-dimensional motion planning problems, especially for systems under geometric and dynamic constraints. However, one potential limiting factor is the high computational complexity due to the collision checking process between sampled states and the excessive sampling to reach convergence. Since artificial potential fields are analytical with theoretical guarantee, it makes sense to combine these two paradigms: vector fields are used in Ko, Kim, and Park (2013) to bias the branching of search trees, thus improving the efficiency of sampling and reducing the number of iterations; similar ideas are adopted in Tahir, Qureshi, Ayaz, and Nawaz (2018) and Qureshi and Ayaz (2016) as the potential-based RRT*, by designing directional samples as induced by the underlying potential fields. However, these methods mostly focus on static environments for simple navigation tasks, where the planning is performed offline and neither the potential fields nor the search structure are adapted during execution.

When a robot is deployed in a partially-unknown environment, an online approach is required such that the underlying trajectory adapts to real-time measurements of the actual workspace such as new obstacles. For instances, a fully automated tuning mechanism for navigation functions is presented in Filippidis and Kyriakopoulos (2011), while the notion of dynamic windows is proposed in Ogren and Leonard (2005) to handle dynamic environment. Moreover, the harmonic potentials-based methods are developed further in Rousseas et al. (2022b) for unknown environments, where the weights over harmonic basis are optimized online. A similar formulation is adopted in Loizou and Rimon (2022) where the parameters in the harmonic potentials are adjusted online to ensure safety and global convergence. Furthermore, a semantic perceptual feedback method is introduced in Vasilopoulos, Pavlakos, Schmeckpeper, Daniilidis, and Koditschek (2022) to recognize the size of the obstacles from a pre-trained dataset. Lastly, the scenario of time-varying targets is analyzed in Li and Tanner (2018) by designing an attractive potential that evolves with time. Similarly, dynamic environments are considered in Dahlin and Karayiannidis (2023a) and Huber, Slotine, and Billard (2022) by allowing time-varying and reactive potentials. On the other hand, search-based methods are also extended to unknown environments where various real-time revision techniques are proposed in Otte and Frazzoli (2016) and Shen, Wilson, Harvey, and Gupta (2021). However, less work can be found where the search tree and the underlying potentials should be updated simultaneously and dependently.

Last but not least, the desired task for the robot could be more complex than the point-to-point navigation. Linear Temporal Logics (LTL) in Baier and Katoen (2008) provide a formal language to describe complex high-level tasks, such as sequential visit, surveillance and response. Many recent papers can be found that combine robot motion planning with model-checking-based task planning, e.g., a single robot under LTL tasks (Fainekos et al., 2009; Guo, Andersson, & Dimarogonas, 2018; Lindemann, Matni, & Pappas, 2021), a multi-robot system under a global task (Guo & Dimarogonas, 2015; Leahy et al., 2021; Luo, Kantaros, & Zavlanos, 2021), However, many aforementioned work assumes an existing low-level navigation controller, or considers a simple and known environment with circular and non-overlapping obstacles. The synergy of complex temporal tasks and harmonic potential fields within unknown environments has not been investigated.

1.2. Our method

This work proposes an automated planning framework termed that utilize harmonic potentials for navigation and oriented search trees for planning, as illustrated in Fig. 1. The design

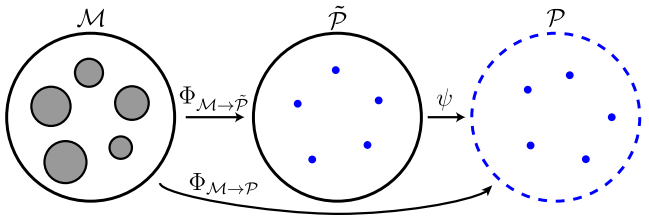


Fig. 2. Illustration of the diffeomorphic transformation from sphere word $\mathcal{M}$ to bounded point world $\tilde{\mathcal{P}}$ and to unbounded point world $\mathcal{P}$.

and construction of the search tree is specially tailored for the task automaton and co-designed with the underlying navigation controllers based on harmonic potentials. Intermediate waypoints are introduced between task regions to improve task efficiency and smoothness of the robot trajectory. Additionally, a novel orientation-aware harmonic potential is proposed for nonholonomic robots, based on which a nonlinear tracking controller is utilized to ensure safety. Furthermore, during online execution, as the robot explores the environment gradually, an efficient adaptation scheme is proposed to update the estimated obstacles, the search tree and the harmonic potentials simultaneously, where intermediate variables are saved and re-used. For validation, extensive simulations and hardware experiments are conducted for nontrivial tasks.

Main contribution of this work lies in the hybrid framework that combines two powerful methods in control and planning, for non-holonomic robots to accomplish complex tasks in unknown environments. Specifically, it includes: (i) the two-layer and automaton-guided Harmonic trees that unify task planing and motion control; (ii) a new "purging" method for forests of overlapping squircles, which is tailored for the online case where obstacles are added gradually; (iii) an integrated method to update the estimated obstacles, the Harmonic potentials and the search trees simultaneously and recursively online. It has been shown via both theoretical analyses and numerical studies that it avoids the common problem of oscillation or long-detours for purely potential-based methods, and sharp-turns or high computation complexity for purely search-based methods.

1.3. Note for practitioners

To apply the proposed method to practical systems, the following steps are recommended given the framework in Fig. 1: (i) the workspace model should be constructed w.r.t. the specified task including the regions of interest and their properties, as modeled in Section 2.2; (ii) the abstraction method and relative distance for the vertices within the harmonic tree should be chosen according to the characteristics of the workspace, such as size and typical structure, as described in Section 4.1.1; (iii) the nonlinear tracking controller in Section 4.1.3 is tuned for the specific hardware platform such that it can track the gradient of the oriented harmonic potentials in Section 4.1.2 with a desired accuracy; (iv) the segmentation and clustering of the online data points for the obstacle estimation should be adjusted according the range and resolution of the Lidar sensor, as mentioned in Section 4.1.2; (v) the update rate of the estimated obstacles and the condition for replanning should be tuned according to the estimated density of obstacles, as discussed in Section 4.2.

2. Preliminaries

2.1. Diffeomorphic transformation and harmonic potentials

A 2D *sphere world* $\mathcal{M}$ is defined as a compact and connected subset of $\mathbb{R}^2$, which has an outer boundary $O_0 = \{q \in \mathbb{R}^2 : \|q - q_0\|^2 - \rho_0^2 \le 0\}$ centered at q_0 with radius ρ_0, and inner boundaries of M disjoint sphere obstacles $O_i = \{q \in \mathbb{R}^2 : \|q - q_i\|^2 - \rho_i^2 \le 0\}$ centered at q_i with radius ρ_i, for $i = 1, \dots, M$. There is a goal point denoted by $q_G \in \mathcal{M}$. By using a diffeomorphic transformation proposed in Loizou (2017), this sphere world can be mapped to an unbounded *point world*, as shown in Fig. 2. It is denoted by $\mathcal{P} = \mathbb{R}^2 \backslash \{P_1, \dots, P_M\}$, which consists of M point obstacles $P_i \in \mathbb{R}^2$. Specifically, the diffeomorphic transformation $\Phi_{\mathcal{M}\to\mathcal{P}}(q)$ from sphere world to point world is constructed as follows:

$$\begin{aligned} \Phi_{\mathcal{M}\to\mathcal{P}}(q) &\triangleq \psi \circ \Phi_{\mathcal{M}\to\tilde{\mathcal{P}}}(q), \\ \Phi_{\mathcal{M}\to\tilde{\mathcal{P}}}(q) &\triangleq \mathrm{id}(q) + \sum_{i=1}^{M} \big(1 - s_\delta(q,\ O_i)\big)(q_i - q), \\ \psi(\tilde{q}) &\triangleq \frac{\rho_0}{\rho_0 - \|\tilde{q} - q_0\|}(\tilde{q} - q_0) + q_0, \end{aligned} \tag{1}$$

where $\Phi_{\mathcal{M}\to\tilde{\mathcal{P}}}(q)$ transforms the sphere world $\mathcal{M}$ to a bounded point world $\tilde{\mathcal{P}} = O_0 \backslash \{\tilde{P}_1, \dots, \tilde{P}_M\}$, of which $\tilde{P}_i$ is the inner point-shape obstacle with $\tilde{P}_i = \Phi_{\mathcal{M}\to\tilde{\mathcal{P}}}(q_i)$, for $i = 1, \dots, M$; the summed element $s_\delta(q,\ O_i)$ is the contraction-like transformation for obstacle O_i, which is composed by $\eta_\delta(x) \circ \sigma(x) \circ b_i(x)$ as the switch function, smoothing function and distance function, respectively. The exact definitions can be found in Loizou (2017) and the supplementary files. To obtain an infinite harmonic domain, it is essential to map the bounded point world into the unbounded point world via the diffeomorphic transformation $\psi(\tilde{q})$. Given the point world $\mathcal{P}$, its associated *harmonic potential* function, is introduced in Loizou and Rimon (2021, 2022) and defined as follows.

Definition 1. The harmonic potential function in a point world, denoted by $\phi_{\mathcal{P}} : \mathcal{P} \to \mathbb{R}^+$, is defined as:

$$\phi_{\mathcal{P}}(x) \triangleq \phi(x,\ P_G) - \frac{1}{K}\sum_{i=1}^{M} \phi(x, P_i), \tag{2}$$

where $\phi(x,\ q) = \ln\big(\|x - q\|^2\big)$ is the primitive harmonic function for $x,\ q \in \mathbb{R}^2$; $\phi(x,\ P_G)$ is the potential for the transformed goal $P_G = \Phi_{\mathcal{M}\to\mathcal{P}}(q_G)$, whereas $\phi(x,\ P_i)$ for the obstacle P_i, where $i = 1, \dots, M$; $K \ge 1$ is a tuning parameter. ■

Lastly, the logistic function is used to transform the unbounded range of $\phi_{\mathcal{P}}$ to a finite interval $[0,\ \mu]$ for $\mu \ge 1$.

2.2. Linear temporal logic and büchi automaton

The basic ingredients of Linear Temporal Logic (LTL) formulas are a set of atomic propositions AP, and several Boolean or temporal operators. Atomic propositions are Boolean variables that can be either true or false. The syntax of LTL is defined as: $\varphi \triangleq \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$, where $\top \triangleq \texttt{True}$, $p \in AP$, $\bigcirc$ (*next*), U (*until*) and $\bot \triangleq \neg\top$. The derivations of other operators, such as $\Box$ (*always*), $\Diamond$ (*eventually*), $\Rightarrow$ (*implication*) are omitted here for brevity. A complete description of the semantics and syntax of LTL can be found in Baier and Katoen (2008). Moreover, there exists a Nondeterministic Büchi Automaton (NBA) for formula φ as follows:

Definition 2. A NBA $\mathcal{A} \triangleq (S,\ \Sigma,\ \delta,\ (S_0,\ S_F))$ is a 4-tuple, where S are the states; $\Sigma = AP$; $\delta : S \times \Sigma \to 2^S$ are transition relations; $S_0, S_F \subseteq S$ are initial and accepting states. ■

An infinite word w over the alphabet 2^{AP} is defined as an infinite sequence $W = \sigma_1\sigma_2 \cdots, \sigma_i \in 2^{AP}$. The language of φ is defined as the set of words that satisfy φ, namely, $\mathcal{L} = Words(\varphi) = \{W \mid W \models \varphi\}$ and $\models$ is the satisfaction relation.

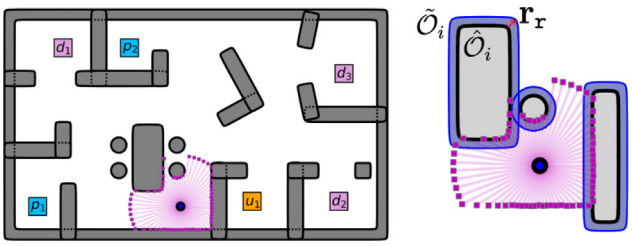


Fig. 3. Left: Robot in the workspace with overlapping squircles and several regions of interest. **Right:** Estimated (in black) and inflated (in blue) obstacles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

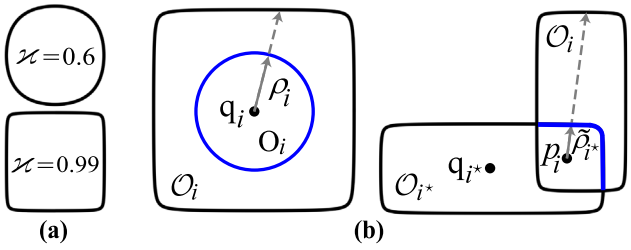


Fig. 4. (a) Squircles with the parameter $\varkappa = 0.6$ and $\varkappa = 0.99$; (b) Ray scaling process. The boundary of the star-shaped obstacle is mapped onto the boundary of a sphere (**Left**). The boundary of the child obstacle is mapped onto a segment of the boundary of the parent obstacle (**Right**).

Additionally, the resulting *run* of w within $\mathcal{A}$ is an infinite sequence $\rho = s_0 s_1 s_2 \cdots$ such that $s_0 \in S_0$, and $s_i \in S$, $s_{i+1} \in \delta(s_i, \sigma_i)$ hold for all index $i \geq 0$. A run is called *accepting* if it holds that $\inf(\rho) \cap S_F \neq \emptyset$, where $\inf(\rho)$ is the set of states that appear in ρ infinitely often. In general, an accepting run has the prefix–suffix structure from an initial state to an accepting state that is contained in a cyclic path. Typically, the size of $\mathcal{A}$ is double exponential to the length of formula φ.

3. Problem description

Consider a mobile robot that occupies a circular area with radius $r_r > 0$ and follows the unicycle dynamics:

$$\dot{x} = v\cos(\theta),\ \dot{y} = v\sin(\theta),\ \dot{\theta} = \omega, \tag{3}$$

where $q = (x, y) \in \mathcal{W}$ is the robot position and $\theta \in [-\pi, \pi]$ as its orientation; (v, ω) are its linear and angular velocities as control inputs. The workspace $\mathcal{W}_0 \subset \mathbb{R}^2$ is compact and connected, with M potentially overlapping internal obstacles $\mathcal{O}_i \subset \mathcal{W}_0$, $\forall i \in \{1, \ldots, M\}$. Considering the robot size, the outer workspace $\mathcal{W}_0$ and inner obstacle $\mathcal{O}_i$ are inflated by a margin r_r, denoted by $\tilde{\mathcal{W}}_0$ and $\tilde{\mathcal{O}}_i$. Therefore, the feasible workspace is given by $\mathcal{W} \triangleq \tilde{\mathcal{W}}_0 \setminus \bigcup_{i=1}^{M} \tilde{\mathcal{O}}_i$.

Each obstacle $\mathcal{O}_i$ belongs to a type of obstacle called squircle, which are particularly useful for representing walls and corners, see Li and Tanner (2018). As shown in Fig. 4, a squircle interpolates smoothly between a circle and a square, while avoiding non-differentiable corners. In particular, a unit squircle centered at the origin in $\mathbb{R}^2$ is given by:

$$\beta_{\mathrm{sc}}(q) \triangleq \frac{q^2 + \sqrt{q^4 - 4\varkappa^2\left[(q^\intercal e_1)(q^\intercal e_2)\right]^2}}{2} - 1, \tag{4}$$

where $\varkappa \in (0,\ 1)$ is the curvature; e_1, e_2 are unit basis in $\mathbb{R}^2$. Non-unit and rotated squircles with general centers can be derived via scaling, translation and rotation.

Assumption 1. The workspace boundary and inner obstacles all follow the model of squircles in (4). ■

Initially at $t = 0$, the workspace is only *partially* known to the robot, i.e., the outer boundary and some inner obstacles. Starting from any valid initial state $(q_0,\ \theta_0)$, the robot can navigate within the workspace and observe more obstacles, via a range-limited sensor modeled as follows:

$$\mathcal{S}(q) \triangleq \left\{\hat{q} \in \mathcal{W} \mid \left(\hat{q} \in \mathcal{D}_{r_s}(q)\right) \wedge \left(\mathcal{L}(q,\ \hat{q}) \subset \mathcal{W}\right)\right\}, \tag{5}$$

where $\mathcal{S}(q)$ is the set of points $\hat{q}$ observed by the robot at position $q \in \mathcal{W}$; $\mathcal{D}_{r_s}(q)$ is a disk centered at q with radius r_s and $\mathcal{L}(q,\ \hat{q})$ is the line connecting q and $\hat{q}$. As shown in Fig. 3, it returns the 2D point cloud from the robot to any blocking surface within the sensing range. This model mimics a 360° Lidar scanner as also used in Rousseas et al. (2022b). Given the measurements, an observed obstacle can be estimated accordingly.

Assumption 2. An obstacle $\mathcal{O}_i \in \mathcal{W}_0$ can be estimated accurately once $S(q)$ in (5) intersects with its occupied area. ■

Note that the implication and relaxation of Assumption 2 are discussed in Sections 4.1.2 and 4.2.1 in the sequel. Lastly, there is a set of non-overlapping regions of interest $g_n \subset \mathcal{W}$, $n = 1, \ldots, N$ within the freespace. With slight abuse of notation, the associated atomic propositions are also denoted by $G = \{g_n\}$, standing for "the robot is within region g_n, i.e., $q \in g_n$". The desired task is specified as a LTL formula φ over G, i.e., $\varphi = LTL(G)$ by the syntax described in Section 2.2. Given the robot trajectory $\mathbf{q}$, its trace is given by the sequence of regions over time, i.e., $\omega(\mathbf{q}) = g_{\ell_1} g_{\ell_2} \cdots$, where $g_{\ell_k} \in G$ and $q(t_k) \in g_{\ell_k}$, for some time instants $0 \leq t_k \leq t_{k+1}$ and $k \in \mathbb{Z}$.

Thus, the objective is to design an online control and planning strategy for system (3) such that starting from an initial pose $(q_0,\ \theta_0)$, the trace of the resulting trajectory $\omega(\mathbf{q})$ fulfills the given task φ, while avoiding collision with all obstacles.

4. Proposed solution

As illustrated in Fig. 1, the proposed solution is a hybrid control framework as two-layer harmonic trees (HT) that combines harmonic potentials for navigation and oriented search trees for planning. Initially, the automaton-guided search trees and the orientation-aware harmonic potentials are constructed in a dependent manner given the partially-known workspace. Then, as the robot explores more obstacles, an online adaptation scheme is proposed to revise the search tree and update the harmonic potentials recursively and simultaneously.

4.1. Initial synthesis

4.1.1. Two-layer and automaton-guided harmonic trees

As described in Section 2.2, the NBA associated with φ is given by $\mathcal{A}_\varphi = (S,\ \Sigma,\ \delta,\ (S_0,\ S_F))$, which captures all potential traces that satisfy the task. To begin with, the initial *navigation map* is constructed as a weighted and fully-connected graph $\mathcal{G} \triangleq (\widehat{G},\ E,\ d,\ (g_0, \theta_0))$, where: (i) $\widehat{G} = G \times \Theta$ is the set of regions of interest plus a set of orientations $\Theta \subset [0, 2\pi)$; (ii) $E \subset \widehat{G} \times \widehat{G}$ is the set of transitions; (iii) $d : E \to \mathbb{R}^+$ is the cost function, which is initialized as $d((g, \theta),\ (g', \theta')) \triangleq \|g - g'\|_2 + w|\theta - \theta'|$, $\forall (g, \theta),\ (g', \theta') \in \widehat{G}$ and parameter $w > 0$; and (g_0, θ_0) is the initial pose. Note that E is initialized as fully-connected since the actual feasibility and cost can only be determined after the associated controllers are constructed.

Given $\mathcal{G}$ and $\mathcal{A}_\varphi$, the standard model-checking procedure is followed to find the task plan. Namely, their synchronized product is built as $\widehat{\mathcal{A}} \triangleq \mathcal{G} \times \mathcal{A}_\varphi = (\widehat{S},\ \widehat{\delta},\ \widehat{d},\ (\widehat{S}_0, \widehat{S}_F))$, where $\widehat{S} =$

$\widehat{G} \times S$; $\widehat{S}_0, \widehat{S}_F \subset \widehat{S}$ are the sets of initial and accepting states; $\widehat{\delta} \subset \widehat{S} \times \widehat{S}$ that $(\langle g, s\rangle, \langle g', s'\rangle) \in \widehat{\delta}$ if $(g, g') \in E$ and $s' \in \delta(s, \{g\})$; $\widehat{d}(\langle g, s\rangle, \langle g', s'\rangle) = d(g, g')$, $\forall(\langle g, s\rangle, \langle g', s'\rangle) \in \widehat{\delta}$. Note that the product $\widehat{\mathcal{A}}$ is still a Büchi automaton, of which the accepting run satisfies the prefix–suffix structure. Namely, consider the following run of $\widehat{\mathcal{A}}$: $\widehat{\mathbf{S}} = \widehat{s}_1\widehat{s}_2\cdots\widehat{s}_L(\widehat{s}_{L+1}\widehat{s}_{L+2}\cdots\widehat{s}_{L+H})^\omega$, which an infinite sequence of $\widehat{S}$ with $\widehat{s}_1\cdots\widehat{s}_L$ being the prefix and $\widehat{s}_{L+1}\cdots\widehat{s}_{L+H}$ being the suffix repeated infinitely often; $\widehat{s}_1 \in \widehat{S}_0$ and $\widehat{s}_{L+1} \in \widehat{S}_F$; and $L, H \geq 1$. A nested Dijkstra algorithm is used to find the best pair of initial and accepting states $(\widehat{s}_0^\star, \widehat{s}_F^\star)$. Thus, the optimal plan given the initial environment is obtained by projecting $\widehat{\mathbf{S}}$ onto $\widehat{G}$, i.e.,

$$\widehat{\mathbf{g}} = \widehat{g}_1\widehat{g}_2\cdots\widehat{g}_L(\widehat{g}_{L+1}\widehat{g}_{L+2}\cdots\widehat{g}_{L+H})^\omega, \tag{6}$$

where $\widehat{g}_\ell = \widehat{s}_\ell|_{\widehat{G}}$ are the regions. More algorithmic details can be found in Guo and Dimarogonas (2015).

To avoid oscillation or jitters and long detours as mentioned in Section 1.1, the structure of oriented harmonic trees (HT) is proposed. More specifically, the oriented HT associated with $(\widehat{g}_\ell, \widehat{g}_{\ell+1})$ is a tree structure defined by a 4-tuple:

$$\mathcal{T}_{\widehat{g}_\ell \to \widehat{g}_{\ell+1}} \triangleq \big(V,\ B,\ \gamma,\ (v_0,\ v_G)\big), \tag{7}$$

where $V \subset \mathcal{W} \times (-\pi, \pi]$ is the set of vertices; $B \subset V \times V$ is the set of edges; $\gamma : B \to \mathbb{R}_{\geq 0}$ returns the edge cost to be estimated; $v_0 = \widehat{g}_\ell$ and $v_G = \widehat{g}_{\ell+1}$ are the initial and target poses. The goal is to find a sequence of vertices in $\mathcal{T}$ as the path from v_0 to v_G. Initially, $V = \{v_0, v_G\}$ and $B = \emptyset$. Then, the set of vertices can be generated in various ways, e.g., the visibility graph from Huang and Chung (2004) or sampling-based methods from LaValle (2006). It is worth mentioning that these vertices should be augmented by *orientations* if not already. Afterwards, any vertex is connected to all vertices within its *free* vicinity with an estimated cost. Thus, given the weighted and directed tree $\mathcal{T}_{\widehat{g}_\ell \to \widehat{g}_{\ell+1}}$, the shortest path from v_0 to v_G is determined by A* from LaValle (2006), denoted by:

$$\mathbf{P}_{\widehat{g}_\ell \to \widehat{g}_{\ell+1}} \triangleq v_0 v_1 \cdots v_{N-1} v_G, \tag{8}$$

where $v_n \in V$ and $(v_n,\ v_{n+1}) \in B, \forall n \in [0,\ N-1]$. In other words, each vertex along the path serves as the intermediate waypoints to navigate from region $\widehat{g}_\ell$ to $\widehat{g}_{\ell+1}$.

4.1.2. Orientation-aware harmonic potentials

As explained in (5), a 2D point cloud is returned that consists of points on any obstacle surface within the sensing range. More specifically, denoted by $\mathbf{D}_t = \{d_j\}$ the set of 2D points that are already transformed from the local coordinate to global coordinate where $d_j \in \partial\mathcal{W}$. To begin with, these data points are divided into K clusters representing K separate obstacles, i.e., $\mathbf{D}_t = \{\mathbf{D}_{t,k}\}$, by e.g., checking the relative distance and change of curvature between consecutive points. The exact thresholds would depend on the specification of the Lidar scanner. Then, each cluster is fitted to the model of squircles in (4) to estimate the curvature, translation, scaling and rotation, via general nonlinear optimization solvers, e.g., Gavin (2019). As shown in Fig. 5, the estimation accuracy relies heavily on the actual parameters of the model in (4), the noise level and the distribution of the measurements. More specifically, the estimation is rather accurate when the curvature $\varkappa$ is close to 0 such as circles; however becomes inaccurate or uncertain when $\varkappa$ is close to 1 such as rectangles and when the data points are few or noisy. In the later case, either a pre-stored database of common obstacles and their shapes can be retrieved based on object recognition as adopted in Vasilopoulos et al. (2022), or an optimistic strategy that chooses the one with the minimum area among the candidates. For the rest of this section, it is assumed that the estimation is accurate by Assumption 2.

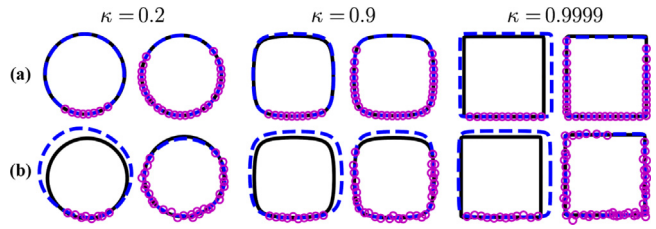


Fig. 5. Estimated squircles (in blue dashed lines) under different curvatures κ and different distributions of (a) accurate or (b) noisy measurements. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Relaxation of this assumption via online adjustment as more measurements are gathered is discussed in Section 4.2.1.

Consequently, denote by $\{\hat{\mathcal{O}}_t\}$ the collection of obstacles detected and fitted at time $t \geq 0$. An example is shown in Fig. 3, where an initial workspace model is constructed given the sensory data at $t = 0$. Given the set of squircle obstacles $\{\hat{\mathcal{O}}_0\}$, the initial navigation function $\varphi_{\text{NF}}(q)$ is constructed by *three* major diffeomorphic transformations: (i) $\Phi_{\mathcal{F}\to\mathcal{S}}$ transforms the forest of stars into a star world via "purging"; (ii) $\Phi_{\mathcal{S}\to\mathcal{M}}$ transforms the star world into its model sphere world; and (iii) $\Phi_{\mathcal{M}\to\mathcal{P}}$ transforms the sphere world into a point world. The complete navigation function for the original workspace is given by:

$$\varphi_{\text{NF}}(q) = \sigma \circ \phi_{\mathcal{P}} \circ \Phi_{\mathcal{M}\to\mathcal{P}} \circ \Phi_{\mathcal{S}\to\mathcal{M}} \circ \Phi_{\mathcal{F}\to\mathcal{S}}(q), \tag{9}$$

where $\sigma \circ \phi_{\mathcal{P}}$ is defined in (2); and the transformation $\Phi_{\mathcal{M}\to\mathcal{P}}$ is given in (1). The remaining part of this section describes the two essential and nontrivial transformations $\Phi_{\mathcal{S}\to\mathcal{M}}$ and $\Phi_{\mathcal{F}\to\mathcal{S}}$.

Star-to-Sphere Transformation. The star world $\mathcal{S}$ has an outer boundary of squircle workspace $\mathcal{O}_0 = \{q \in \mathbb{R}^2 | \beta_0(q) \leq 0\}$ and M inner squircle obstacles $\mathcal{O}_i = \{q \in \mathbb{R}^2 | \beta_i(q) \leq 0\}$, where $\beta_i(q)$ is the obstacle function defined in (4), for $i = 0, 1, \ldots, M$. The star-to-sphere transformation is constructed by the ray scaling process from Rimon (1990), as shown in Fig. 4. To begin with, define the following scaling factor:

$$v_0(q) \triangleq \rho_0 \frac{1 - \beta_0(q)}{\|q - \mathbf{c}_0\|}, \quad v_i(q) \triangleq \rho_i \frac{1 + \beta_i(q)}{\|q - \mathbf{c}_i\|}, \tag{10}$$

where $\mathbf{c}_i$ is the geometric center of the associated squircle and ρ_i is the radius of the transformed sphere. Moreover, the translated scaling map T_i for each obstacle $\mathcal{O}_i$ is defined by:

$$T_i(q) \triangleq v_i(q)(q - \mathbf{c}_i) + \mathbf{c}_i, \tag{11}$$

for $i = 0, 1, \ldots, M$. Consequently, the transformation from star world $\mathcal{S}$ to its model sphere world $\mathcal{M}$ is given by:

$$\Phi_{\mathcal{S}\to\mathcal{M}} \triangleq \Big(1 - \sum_{i=0}^{M} \sigma_i(q)\Big)\,\text{id}(q) + \sum_{i=0}^{M} \sigma_i(q)\, T_i(q), \tag{12}$$

where $\sigma_i(q) \triangleq \frac{\gamma_G(q)\overline{\beta}_i(q)}{\gamma_G(q)\overline{\beta}_i(q) + \lambda\beta_i(q)}$ is the analytic switch for the workspace boundary and obstacles; $\text{id}(q)$ is the identity function; $\lambda > 0$ is a parameter; $\gamma_G(q) \triangleq \|q - q_G\|^2$ is the distance-to-goal; and $\overline{\beta}_i(q) \triangleq \prod_{j=0, j\neq i}^{M} \beta_j(q)$ is the omitted product.

Leaf-Purging Transformation. As described in Li and Tanner (2018), Loizou and Rimon (2022) and Rimon (1990), besides disjointed star worlds, it is essential to consider the workspace formed by unions of *overlapping* stars, called the forest of stars. It consists of several disjointed clusters of obstacles as trees of stars, which in turn is a finite union of overlapping star obstacles whose adjacency graph is a tree. Since overlapping stars cannot be transformed directly as a whole obstacle, each tree has to be transformed via successively *purging* its leaves. Specifically, a forest of stars is described by $\mathcal{F} = \mathcal{W}_0 \setminus \bigcup_{n=1}^{N} \mathcal{T}_n$, where $\mathcal{T}_n$

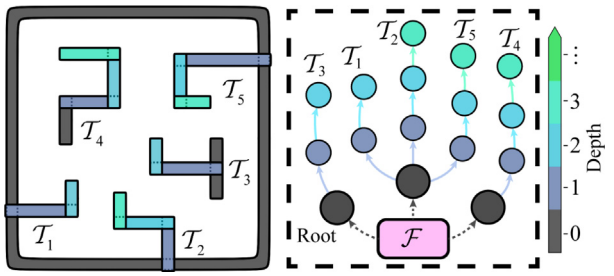


Fig. 6. A forest world $\mathcal{F}$ with overlapping squircles (**Left**), which consists of 5 trees of stars $\mathcal{T}_i$ with different depths from the root to the leaves (**Right**).

is the nth tree of stars among the N trees. Each tree of stars has a unique root, and its obstacles are arranged in a parent–child relationship. An example is shown in Fig. 6, where the trees have different depths according to the level of leaves in the tree. Without loss of generality, denote by d_n the depth of the tree $\mathcal{T}_n$ and $\mathcal{L}$ the set of indices for all leaf obstacles in all trees, and by $\mathcal{I}$ the set of indices associated with all obstacles within $\mathcal{F}$. Furthermore, denote by $\mathcal{O}_i \triangleq \{q \in \mathbb{R}^2 : \beta_i(q) \leq 0\}$ the leaf obstacles, where $\beta_i(q)$ is the obstacle function defined in (4). The *parent* of obstacle $\mathcal{O}_i$ is denoted by $\mathcal{O}_{i^\star}$, of which the centers are chosen to be the same and denoted by $p_i \in \mathcal{O}_i \bigcap \mathcal{O}_{i^\star}$.

Then, the diffeomorphic transformation from the forest of stars $\mathcal{F}$ to the star world S is constructed via the successive purging transformations for each tree as follows:

$$\Phi_{\mathcal{F}\to\mathcal{S}}(q) \triangleq \Phi_N \circ \cdots \Phi_2 \circ \Phi_1(q), \tag{13}$$

where $\Phi_n(q)$ is the purging transformation for the nth tree of stars, where $n = 1, \ldots, N$. It has the following format:

$$\Phi_n(q) \triangleq f_{n,1} \circ f_{n,2} \cdots \circ f_{n,d_n}(q), \tag{14}$$

where $f_{n,i}(q)$ is the purging transformation for the ith leaf obstacle in the nth tree for $i = 1, \ldots, d_n$, which is constructed by the ray-scaling process to purge the leaf obstacle into its parent as shown in Fig. 4. Due to the specific representation of the squircle, the length of rays from the common center p_i to the boundary of the parent obstacle $\mathcal{O}_{i^\star}$, is calculated by:

$$\tilde{\rho}_{i^\star}(\hat{q}) \triangleq \frac{1}{\|\tilde{\mathbf{A}}_{i^\star}^{-1}\hat{q}\|}\, \rho_{sc}\left(\frac{\tilde{\mathbf{A}}_{i^\star}^{-1}\hat{q}}{\|\tilde{\mathbf{A}}_{i^\star}^{-1}\hat{q}\|}\right), \tag{15}$$

where $\hat{q} = \frac{q-p_i}{\|q-p_i\|}$ is the normalized vector of $q - p_i$; $\rho_{sc}(\cdot)$ is the length of the rays for unit squircle drived from (4); and $\tilde{\mathbf{A}}_{i^\star} \in \mathbb{R}^{2\times 2}$ is a piecewise scaling matrix tailored for the parent squircle which is diagonal with the following entries:

$$\tilde{a}_{i^\star}^{\ell}(\hat{q}) \triangleq a_{i^\star}^{\ell} + \mathbf{sgn}(\hat{q}^\intercal e_\ell)[\mathbf{c}_{i^\star} - p_i]^\intercal e_\ell, \tag{16}$$

for $\ell = 1, 2$, where $a_{i^\star}^{\ell}$ are the entries of the diagonal scaling matrix $\mathbf{A}_{i^\star}$ of the parent squircle; $\mathbf{c}_{i^\star}$ is the geometric center of the parent obstacle; and e_ℓ are two base vectors for $\ell = 1, 2$.

Lemma 1. *The length of rays $\tilde{\rho}_{i^\star}(q)$ is smooth in $\mathcal{W}_0 \backslash \mathcal{O}_i \bigcup \mathcal{O}_{i^\star}$.*

Proof. The derivative of $\tilde{\rho}_{i^\star}(q)$ is given by the chain rule:

$$\nabla\tilde{\rho}_{i^\star}(q) = \frac{\|q-p_i\|^2\mathbf{I} - (q-p_i)(q-p_i)^\intercal}{\|q-p_i\|^3}\nabla\tilde{\rho}_{i^\star}(\hat{q}),$$

where the derivative of $\tilde{\rho}_{i^\star}(\hat{q})$ boils down to computing the derivatives of $\rho_{sc}(q)$ and $Z(\hat{q}) = \tilde{\mathbf{A}}_{i^\star}^{-1}\hat{q}$ in (15). The smoothness of $\rho_{sc}(q)$ and its derivative can be obtained directly from (4). Since $\nabla a_{i^\star}^{\ell}(\hat{q}) = \nabla(a_{i^\star}^{\ell} + \mathbf{sign}(\hat{q}^\intercal e_\ell)[q_{i^\star} - p_i]^\intercal e_\ell) = 0$, for $\ell = 1, 2$, the derivative of $Z(\hat{q})$ is calculated and simplified as:

$$\nabla Z(\hat{q}) = \tilde{\mathbf{A}}_{i^\star}^{-1}\mathbf{I} - \tilde{\mathbf{A}}_{i^\star}^{-1}[\nabla\tilde{\mathbf{A}}_{i^\star}]\tilde{\mathbf{A}}_{i^\star}^{-1}\hat{q} = \tilde{\mathbf{A}}_{i^\star}^{-1}.$$

Besides, as $\|q - p_i\| > 0$ holds for every point $q \in \mathcal{W}_0 \backslash \mathcal{O}_i \bigcup \mathcal{O}_{i^\star}$, the derivative of $\tilde{\rho}_{i^\star}(q)$ exists and is well-defined. On the other hand, $\nabla\tilde{\rho}_{i^\star}(q)$ is continuous since its compositions are all continuous. Therefore, $\tilde{\rho}_{i^\star}(q)$ is at least C^1 smooth. Additionally, higher-order smoothness can be proven by the same procedure, which is omitted here. □

Given the length of rays $\tilde{\rho}_{i^\star}(q)$, the star deforming factor for each leaf obstacle $\mathcal{O}_i$ is calculated by:

$$v_i(q) = \tilde{\rho}_{i^\star}(q)\,\frac{1 + \beta_i(q)\tilde{\kappa}_i(q)}{\|q - p_i\|}, \tag{17}$$

where $\tilde{\kappa}_i(q)$ is defined by: $\tilde{\kappa}_i(q) \triangleq \beta_{i^\star}(q) + (\beta_i(q) - 2E_i) + \sqrt{\beta_{i^\star}^2(q) + ((\beta_i(q) - 2E_i))^2}$. The parameter $E_i > 0$ is a geometric constant satisfying that $\mathcal{O}_i(2E_i) \bigcap \mathcal{O}_j(2E_j) = \emptyset$ and $\gamma_G^{-1}([0, 2E_G]) \bigcap \mathcal{O}_i(2E_i) = \emptyset$ with $E_G > 0$ being a geometric constant and $\mathcal{O}_i(x) \triangleq \{q \in \mathbb{R}^2 | \beta_i(q) \leq x\}$ with $x > 0$, for $i, j \in \mathcal{I}$ with $i \neq j$ and $i, j \neq j^\star$, where $\mathcal{O}_{i^\star}$ and $\mathcal{O}_{j^\star}$ are the parent obstacles of $\mathcal{O}_i$ and $\mathcal{O}_j$ in the tree, respectively. Furthermore, the translated scaling map T_i for each obstacle $\mathcal{O}_i$ with the common center p_i is defined by:

$$T_i(q) \triangleq v_i(q)(q - p_i) + p_i. \tag{18}$$

Therefore, the purging transformation $f_{n,i}(q)$ for the ith leaf obstacle $\mathcal{O}_i$ in the nth tree is defined by:

$$f_{n,i}(q) \triangleq \big(1 - \sigma_i(q, \xi_i)\big)\,\mathrm{id}(q) + \sigma_i(q, \xi_i)\,T_i(q), \tag{19}$$

where $\sigma_i(q, \xi_i) \triangleq \frac{\gamma_G(q)\overline{\beta}_i(q)}{\gamma_G(q)\overline{\beta}_i(q) + \xi_i\beta_i(q)}$ is the analytic switch; $\xi_i > 0$ is a positive parameter; $\gamma_G(q)$ is the same as in (12); and $\overline{\beta}_i(q)$ is the omitted product defined by:

$$\overline{\beta}_i(q) \triangleq \Big(\prod_{j\in\mathcal{I}\backslash\{i,i^\star\}} \beta_j(q)\Big)\Big(\prod_{j\in\mathcal{L}\backslash\{i\}} \beta_j(q)\Big)\tilde{\beta}_i(q),$$

where $\tilde{\beta}_i(q)$ is defined by: $\tilde{\beta}_i(q) \triangleq \beta_{i^\star}(q) + (2E_i - \beta_i(q)) + \sqrt{\beta_{i^\star}^2(q) + (2E_i - \beta_i(q))^2}$. The positive geometric constant E_i satisfies the same condition as previously described.

Remark 1. The original purging method proposed in Rimon (1990) deals with overlapping obstacles via Boolean combinations and applies only to planar and parabolic obstacles. The work in Li and Tanner (2018) proposes a simpler method for computing the ray scaling transformations using varying ray lengths. However, these approaches purge all the leaves *at once*, which is not applicable to the dynamic scenarios where the obstacles are added to the workspace one by one. Thus, a novel method is proposed in (19) that purges the leafs *one by one*. ■

Lemma 2. *Given a workspace $\mathcal{W}$ containing N tree-of-stars with different depth $d_n \geq 0$, for $n = 1, \ldots, N$, the complete harmonic potential function $\varphi_{\mathrm{NF}}(q)$ in (9) is a valid navigation function for the workspace $\mathcal{W}$, if the parameters $K > N$ and $\mu \geq 1$ hold in (2); $\lambda > \Lambda$ for a positive number $\Lambda > 0$; and $\xi_i > \Xi_i$ for some positive numbers $\Xi_i > 0$, for $i = 0, \ldots, d_n$.*

Proof. Similar statements have been proven in Filippidis and Kyriakopoulos (2011), Loizou and Rimon (2022) and Rousseas et al. (2021), thus detailed proofs are omitted here and refer the readers to e.g., Theorem 1 of Loizou and Rimon (2022), Theorem 2 in Li and Tanner (2018) and Proposition 5 of Loizou (2017). Briefly speaking, it is shown that the transformation $\Phi_{\mathcal{S}\to\mathcal{M}}\,\Phi_{\mathcal{F}\to\mathcal{S}}$ are diffeomorphic as long as $\lambda > \Lambda$ and $\xi_i > \Xi_i$ for some sufficient large numbers Λ and Ξ_i, and the final potential function $\varphi_{\mathrm{NF}}(q)$

has a unique global minimum at the goal q_G and more importantly, a set of isolated saddle points of measure zero if $K > N$ and $\mu \geq 1$ hold. Since the purging process proposed in (19) is novel, this proof mainly shows that $f_{n,i}(q)$ still satisfies the diffeomorphic conditions: (i) the Jacobian of $f_{n,i}(q)$ is nonsingular; (ii) $f_{n,i}(q)$ is a bijection on the boundary. To begin with, since $\tilde{\rho}_{i^*}(q)$ is proven to be smooth in Lemma 1, the Jacobian of $f_{n,i}$ exists and is given by:

$$J_{f_{n,i}} = \sigma_i(q-p_i)\nabla v_i^{\mathsf{T}} + (v_i-1)(q-p_i)\nabla\sigma_i^{\mathsf{T}} + (1-\sigma_i)\mathbf{I} + \sigma_i v_i \mathbf{I}.$$

Similar to Lemmas 7 and 8 in Li and Tanner (2018), it can be shown that there exists a positive constant Ξ_i, such that if $\xi_i > \Xi_i$, $J_{f_{n,i}}$ is nonsingular within the set near $\mathcal{O}_i$, denoted by $\mathcal{O}_i(\epsilon_i) = \{q \in \mathbb{R}^2 | \beta_i \leq \epsilon_i\}$, and the set away from $\mathcal{O}_i$, denoted by $\mathcal{A}(\epsilon_i) = \{q \in \mathbb{R}^2 | \beta_i \geq \epsilon_i\}$, for any $\epsilon_i > 0$. In addition, it remains to show the injectivity and surjectivity of $f_{n,i}(q)$. On the boundary of $\mathcal{O}_i$, it holds that $f_{n,i}(q)|_{q\in\partial\mathcal{O}_i} = \frac{\tilde{\rho}_{i^*}(q)}{\|q-p_i\|}(q-p_i)+p_i$. Assume that there exist two points q, $q' \in \partial\mathcal{O}_i$ such that $f_{n,i}(q) = f_{n,i}(q')$, it can be derived that $q-p_i$ and $q'-p_i$ are on the same ray, yielding to the contradiction that q, q' are the same point. Thus, $f_{n,i}(q)$ is injective on $\partial\mathcal{O}_i$. On the other hand, since it has been shown that the Jacobian of $f_{n,i}(q)$ is nonsingular if $\xi_i > \Xi_i$, $f_{n,i}(q)$ is a local homeomorphism according to the Inverse Function Theorem. Since a local homeomorphism from a compact space into a connected one is surjective, it follows that $f_{n,i}(q)$ is a bijection on the boundary. □

The derived potential fields in (9) can be used to drive holonomic robots from any initial point to the given goal point q_G, e.g., by following the negated gradient in Filippidis and Kyriakopoulos (2011), Loizou and Rimon (2022) and Rousseas et al. (2021, 2022b). However, the final orientation of the robot at the destination *cannot* be controlled, yielding it impractical for the oriented path $\mathbf{P}_{\widehat{g}_\ell \to \widehat{g}_{\ell+1}}$ in (8). Thus, an additional two-step rotation is proposed for the potential fields above.

Two-step Rotation to Harmonic Potentials: As shown in Fig. 7, the main idea is to design a two-step rotation transformation to the harmonic potentials, such that a "dipole-like" field is formed near the goal point with the desired orientation. More specifically, the transformation is given by:

$$\Gamma(q) \triangleq \mathbf{R}\big(\theta_2(q)\big)\,\mathbf{R}\big(\theta_1(q)\big), \tag{20}$$

where $\mathbf{R}(\theta)$ is the standard rotation matrix, i.e., $\mathbf{R}(\theta) = [\cos\theta, -\sin\theta; \sin\theta, \cos\theta]$ for $\theta \in [0, -\pi)$;

$\theta_1(q)$ is the angle to be rotated such that the direction of the transformed potential is aligned with the direction from q_G to q near the goal, i.e.,

$$\theta_1(q) \triangleq s_d(q)\delta_\theta(q) + \mathbf{sgn}\big(\delta_\theta(q)\big)[1-s_d(q)]\delta_c, \tag{21}$$

where $\delta_\theta(q) \triangleq \theta_{q-q_G} - \theta_{\nabla\varphi_{\mathrm{NF}}(q)}$ is the relative angle between $q-q_G$ and $\nabla\varphi_{\mathrm{NF}}(q)$; $\delta_c \triangleq 2\pi$; $s_d(q) \triangleq \exp(\tau - \frac{\tau\mu^2}{(\mu-\varphi_{\mathrm{NF}}(q))^2})$ is a smooth switch function that is "on" near the destination and "off" near the obstacle, with $\tau \in (0,1)$ being a design parameter and μ being the maximum value of $\varphi_{\mathrm{NF}}(q)$. Moreover, the variable $\theta_2(q)$ is the angle to be rotated, such that the transitional potential fields mimic a point-dipole field, i.e.,

$$\theta_2(\check{q}) \triangleq s_d(\check{q})\delta'_\theta(\check{q}) + \mathbf{sgn}\big(\delta'_\theta(\check{q})\big)[1-s_d(\check{q})]\delta'_c + \delta'_c, \tag{22}$$

where $\check{q}$ is the transformed coordinate of q in the new coordinate system, where the goal point q_G is the origin and the desired orientation θ_G is the positive direction of the x-axis, i.e., $\check{q} \triangleq \mathbf{R}^{-1}(\theta_G)(q-q_G)$, where $\mathbf{R}(\theta_G)$ is the rotation matrix associated with the goal angle θ_G; $\delta'_\theta(\check{q}) \triangleq \theta_{\check{q}}$ is the angle of the vector $\check{q}$; $\delta'_c \triangleq \pi$; $s_d(\cdot)$ is the same as in (21).

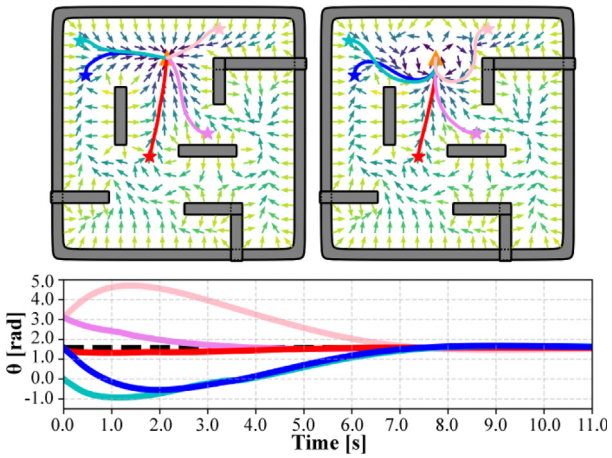


Fig. 7. Robot trajectories from different initial poses (filled stars) to the same goal pose (orange triangle), under the original potential fields $-\nabla\varphi_{\mathrm{NF}}(q)$ (**Top Left**) and the *oriented* potential fields $\Upsilon(q)$ (**Top Right**) via the proposed control law in (24). **Bottom**: The robot orientation θ converges to the desired value (dashed line), along these trajectories. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Definition 3. The *oriented* harmonic potential fields are defined as the fields obtained by transforming the original potential fields through a two-step rotation in (20), denoted by:

$$\Upsilon(q) \triangleq -\Gamma(q)\nabla\varphi_{\mathrm{NF}}(q), \tag{23}$$

where $\nabla\varphi_{\mathrm{NF}}(q)$ is the gradient of the original potentials. ■

Given the oriented harmonic potential fields, the integral curves of $\Upsilon(q)$ correspond to the trajectories $\bar{\xi}(t)$ of the autonomous system described by ordinary differential equations: $\frac{d}{dt}\bar{\xi}(t) = \Upsilon(\bar{\xi}(t)) \in \mathbb{R}^2$, with initial value $\bar{\xi}(t_0) = q_0$. The convergence property of the integral curves is proven as follows.

Lemma 3. *All integral curves of $\Upsilon(q)$ in (23) converge to q_G with the desired orientation θ_G asymptotically, with no collision with any obstacles, as long as the conditions in Lemma 2 hold.*

Proof. To begin with, it is proven in Lemma 2 that $\varphi_{\mathrm{NF}}(q)$ is a valid navigation function, it holds that $\varphi_{\mathrm{NF}}(q) = \mu$ on the boundary of the workspace $\partial\mathcal{W}$. Thus, the switch functions $s_d(q) = 1$ hold both in (21) and (22), and $\theta_1(q) = \theta_2(q) = \pi$ holds. In this case, the rotation matrices $\mathbf{R}(\theta_1) = \mathbf{R}(\theta_2)$ are both identity matrices, thus $\Gamma(q) = \mathbf{I}$ is an identical transformation and the transformed field coincides with the original potential field, i.e., $-\nabla\varphi_{\mathrm{NF}}(q)$. Furthermore, since the negated gradient points away from the boundary $\partial\mathcal{W}$, the transformed field inherits the property of collision avoidance.

Secondly, the transformation $\Gamma(q)$ is nonsingular as both $\mathbf{R}(\theta_1)$ and $\mathbf{R}(\theta_2)$ are non-singular. Thus, all critical points of $\varphi_{\mathrm{NF}}(q)$, including the global minimum q_G and saddle points, retain their properties after the transformation. In other words, the saddle points of $\varphi_{\mathrm{NF}}(q)$ remain to be non-degenerate with an attraction region of measure zero. Since any integral curve in a closed compact set $\overline{\mathcal{W}}$ is bounded, there must be a limit set inside $\overline{\mathcal{W}}$ to which the integral curves converge. Lemma 4 of Valbuena and Tanner (2012) states that no such limit cycles exist if there are no additional attractors other than q_G. Therefore, the goal point q_G is the only attractive component of the limit set within $\overline{\mathcal{W}}$, i.e., the integral curves of $\Upsilon(q)$ converge to q_G.

Lastly, it remains to be shown that the asymptotic convergence to q_G is achieved with the desired orientation θ_G. Clearly,

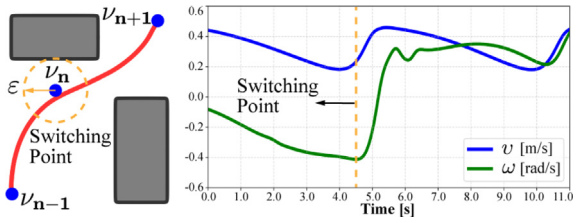


Fig. 8. Illustration of the smooth transition strategy in (25) when the agent switches from one intermediate waypoint to another. The switch function η_ϵ is activated once the current waypoint is reached with a margin ϵ. The resulting trajectory and control inputs are all smooth.

when q approaches q_G, the switch function $s_d(q,\tau)$ approaches 1, while $\theta_1(q), \theta_2(q)$ approach $\delta_\theta(q), \delta'_\theta(q)+\pi$, respectively. For brevity, let $\theta_1(q)=\delta_\theta(q)$ and $\theta_2(q)=\delta'_\theta(q)+\pi$. Therefore, the transformed potential field $\Upsilon(q)$ is simplified as:

$$\Upsilon(q)=\begin{bmatrix}\|\nabla\varphi_{\text{NF}}(q)\|\cos\big(\delta'_\theta(q)+\theta_{q-q_G}\big)\\ \|\nabla\varphi_{\text{NF}}(q)\|\sin\big(\delta'_\theta(q)+\theta_{q-q_G}\big)\end{bmatrix},$$

where $\delta'_\theta(q)\in(-\pi,\pi]$ is defined in (22). Then, the inner product of $\Upsilon(q)$ and q_G-q is given by: $\langle\Upsilon(q),\ q_G-q\rangle=-\|\nabla\varphi_{\text{NF}}(q)\|\|q_G-q\|\cos\big(\delta'_\theta(q)\big)$. It should be noted that the integral curves can only converge to the q_G in the direction of steepest descent, i.e., from the region where $\delta'_\theta(q)$ approaches π. Since $\delta'_\theta(q)=\theta_{\check{q}}$ and $\check{q}=\mathbf{R}^{-1}(\theta_G)(q-q_G)$, $\delta'_\theta(q)$ approaches π implies that θ_{q_G-q} approaches θ_G. Thus, the integral curves converge to q_G along the desired orientation θ_G. □

4.1.3. Smooth harmonic-tracking controller

Given the rotated potentials in (23), a nonlinear feedback controller can be used for non-holonomic robots in (3) to track its rotated and negated gradient. More specifically, the linear and angular velocity is set as follows:

$$\upsilon(q,\ q_G)=k_\upsilon\ \tanh\left(\|q-q_G\|\right); \tag{24a}$$

$$\omega(\theta,\ \theta_G)=-k_\omega\left(\theta-\theta_\Upsilon\right)+\upsilon\left[\nabla\theta_\Upsilon\right]^\intercal J_\Upsilon\,\Theta, \tag{24b}$$

where q, θ are the robot pose and orientation; k_υ, $k_\omega>0$ are controller gains; θ_Υ is the direction of vector $\Upsilon\triangleq[\Upsilon_x,\ \Upsilon_y]^\intercal$; $\nabla\theta_\Upsilon=[\frac{\partial\theta_\Upsilon}{\partial\Upsilon_x},\ \frac{\partial\theta_\Upsilon}{\partial\Upsilon_y}]^\intercal$; J_Υ being the 2×2 Jacobian matrix of Υ, whose entries are $[J_\Upsilon]_{ij}=\frac{\partial\Upsilon_i}{\partial q_j}$, for $i,j\in\{1,2\}$; and $\Theta\triangleq[\cos\theta,\ \sin\theta]^\intercal$. The main idea is to decompose the control objective into two sub-objectives: (i) the robot orientation is regulated sufficiently fast to θ_Υ via (24b); (ii) the robot velocity along this direction is adjusted to reach q_G via (24a). The control gains k_υ, k_ω should be chosen according to the desired safety margin and minimum distance among the obstacles.

Proposition 1. *Under the control law (24), the robot in (3) converges to the goal q_G with orientation θ_G asymptotically, from almost all initial poses in the workspace $\mathcal{W}$, as long as the conditions in Lemma 2 hold.*

Proof (*Sketch*)**.** As discussed, the robot system (3) under control law (24) can be decomposed into two subsystems via singular perturbation based on Khalil (2002). The fast subsystem is equivalent to $\omega=-k_\omega(\theta-\theta_\Upsilon)+\dot\theta_\Upsilon$,

which ensures a global exponential convergence of θ to the rotated fields θ_Υ. Secondly, the slow subsystem is given by $\dot q=\upsilon\frac{\Upsilon}{\|\Upsilon\|}$. The robot trajectory corresponds one integral curve of the vector field $\Upsilon(q)$. As already proven in Lemma 2 that the integral curves of the vector fields $\Upsilon(q)$ converge to q_G with orientation θ_G, the slow subsystem converges to the goal pose with the desired orientation. □

More importantly, since the robot needs to switch along a sequence of waypoints in its path $\mathbf{P}_{\widehat{g}_\ell\to\widehat{g}_{\ell+1}}$ from (8), a smooth transition between the intermediate waypoints is crucial to ensure both smooth trajectories and control inputs. As shown in Fig. 8, the switching strategy from $\nu_n=(q_n,\theta_n)$ to $\nu_{n+1}=(q_{n+1},\theta_{n+1})$ in $\mathbf{P}_{\widehat{g}_\ell\to\widehat{g}_{\ell+1}}$ is given by:

$$u=u(q,\ q_n)+\eta_\varepsilon\left(\gamma_n(q)\right)\big(u(q,\ q_{n+1})-u(q,\ q_n)\big), \tag{25}$$

where u stands for inputs υ and ω; $\gamma_n(q)\triangleq\|q-q_n\|$; and $\eta_\varepsilon(x)\triangleq\frac{1}{2}\big(1+\tanh(k_s(\varepsilon-\gamma_n(q)))\big)$ is the smooth switch function, with $k_s>0$ being a positive gain, and $\varepsilon>0$ the switching margin around q_n, which can be tuned according to how dense the obstacles are located to ensure safety during transition. Namely, a smaller ε should be chosen when obstacles are close to the switching point. Last but not least, the control cost under the proposed control law in (24) to drive the robot from waypoint ν_n to ν_{n+1} is estimated by:

$$\gamma(\nu_n,\ \nu_{n+1})\triangleq d(q_n,\ q_{n+1})+\mathbf{w}\,Q_\Upsilon^\intercal(\theta_n,\theta_{n+1}), \tag{26}$$

where the first part $d(q_n,\ q_{n+1})=\|q_n-q_{n+1}\|$ measures the straight-line distance; the second part approximates the rotation cost: (i) $\mathbf{w}\triangleq[w_1,\ w_2]$ are the weighting parameters with $w_1,\ w_2>0$; (ii) $Q_\Upsilon(\theta_n,\theta_{n+1})\triangleq[|\theta_{\Upsilon(q_n)}-\theta_n|,\ |\theta_{q_{n+1}-q_n}-\theta_{n+1}|]$, where the first item estimates the rotation cost between robot orientation and the rotated gradient at the starting pose; the second item estimates the steering cost between the vector $q_{n+1}-q_n$ and goal orientation. Consequently, it can be used to compute the edge cost of the HT $\mathcal{T}_{\widehat{g}_\ell\to\widehat{g}_{\ell+1}}$ in (7), yielding a more accurate estimate of the navigation cost from $\widehat{g}_\ell$ to $\widehat{g}_{\ell+1}$. In turn, the actual cost within the navigation map $\mathcal{G}$ is given by:

$d(\widehat{g},\ \widehat{g}')=\sum_{n=0}^{N-1}\gamma(\nu_n,\ \nu_{n+1})$, where $\mathbf{P}_{\widehat{g}\to\widehat{g}'}=\nu_0\nu_1\cdots\nu_N$ is the shortest path from $\widehat{g}$ to $\widehat{g}'$ in the associated HT $\mathcal{T}_{\widehat{g}\to\widehat{g}'}$.

Remark 2. The control cost in (26) is estimated through a weighted combination of the translation cost and the steering cost. As shown in Fig. 9, this estimation approaches the actual measured cost when the distance between consecutive waypoints are selected properly. ■

4.1.4. Hybrid execution of the initial plan

The initial plan $\widehat{\mathbf{g}}=\widehat{g}_1\widehat{g}_2\cdots\widehat{g}_L(\widehat{g}_{L+1}\widehat{g}_{L+2}\cdots\widehat{g}_{L+H})^\omega$ in (6) can be executed via the following hybrid strategy: (i) starting from $\ell=1$, determine the next goal region $\widehat{g}_{\ell+1}$; (ii) construct the HT as $\mathcal{T}_{\widehat{g}_\ell\to\widehat{g}_{\ell+1}}$ and determine the shortest path $\mathbf{P}_{\widehat{g}_\ell\to\widehat{g}_{\ell+1}}=\nu_0\cdots\nu_{N-1}\nu_G$ in (8); (iii) starting from the first vertex $n=0$, once the robot enters the small vicinity of vertex ν_n, it switches to the next mode of traversing the subsequent intermediate edge $(\nu_n,\ \nu_{n+1})$. The associated controller (24) is activated with the goal pose ν_{n+1} and potential $\Upsilon_{\nu_n\to\nu_{n+1}}(q)$. The controller remains active until the robot enters the small vicinity of ν_{n+1}. This execution repeats until the vertex ν_G, i.e., $\widehat{g}_{\ell+1}$, is reached, after which the index ℓ is incremented by 1 and returns to step (i). This procedure could be repeated until the last state of plan suffix $\hat{s}_{L+H}$ is reached, if the environment remains static. However, since the environment is only partially known, more obstacles would be detected during execution and added to the environment model. which might invalidate not only the current navigation map but also the underlying harmonic potentials.

Remark 3. The design of intermediate waypoints between subtasks can alleviate certain limitations of the classic navigation functions from Koditschek (1987) and Rimon (1990) for robotic navigation. As shown in Fig. 10, when the robotic system is geometrically and dynamically constrained, it can *not* follow the negated gradient perfectly. Consequently, oscillations appear

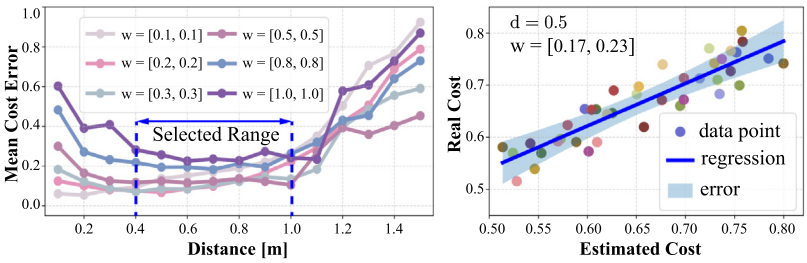


Fig. 9. **Left:** The difference between the estimated cost by (26) and the measured cost under different weights $\mathbf{w}$, for different choice of the distance $d(q_n, q_{n+1})$. **Right:** The difference actual with 40 samples, when $\mathbf{w} = [0.17,\ 0.23]^\intercal$ and $d = 0.5$.

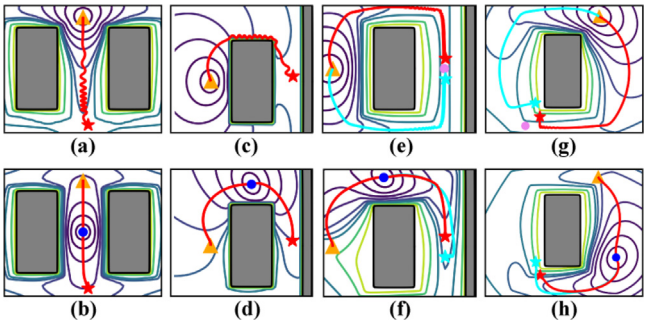


Fig. 10. Illustration of how intermediate waypoints can reduce oscillations and long detours, under different initial poses (solid stars) and goal poses (orange triangles). (i) Oscillations appear as the robot traverses narrow passage in **(a)** or near the obstacle boundary in **(c)**. Intermediate waypoints (blue dots) are introduced in **(b)** and **(d)**, yielding smooth trajectories. (ii) Divergent trajectories appear for two close-by initial poses (red and green stars) due to their proximity to the saddle point (violet dot) in **(e)** and **(g)**. Intermediate waypoints (blue dots) are introduced in **(f)** and **(h)**, yielding nearly identical trajectories. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

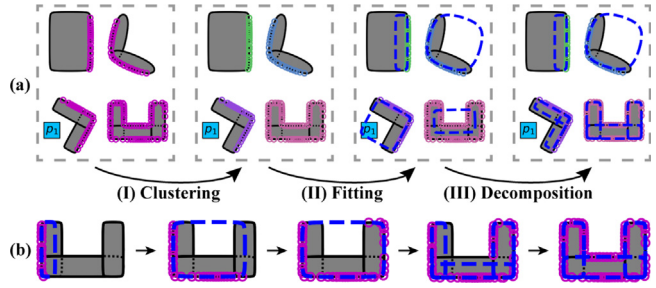


Fig. 11. **(a)** Estimation of the obstacles (in blue dashed lines) in Section 4.2.1, which includes clustering, fitting and decomposition; **(b)** Online adjustment as more measurements (pink circles) are gathered. The cluster is ultimately decomposed into three segments, which are fitted to three squircles when the error exceeds the desired threshold for fitting a single squircle. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

along narrow passages and close to boundary of obstacles, where the underlying gradient changes directions abruptly. As also emphasized in Loizou and Rimon (2022) and Rimon (1990), the phenomenon of "vanishing" valleys and the resulting oscillations often occur when the parameter λ is set too large. Moreover, when two initial poses are located close to saddle points, the resulting trajectories can be diverging in opposite directions. To overcome these issues, the introduced intermediate waypoints can decompose the long path into shorter segments of which the associated potentials are often easier to track. ■

4.2. Online adaptation

As the robot detects more obstacles during execution, the estimated obstacles, the HTs and the underlying harmonic potentials are all updated as follows.

4.2.1. Adaptive obstacle estimation

As discussed in Section 4.1.2 and shown in Fig. 5, the estimation of squircle obstacles can be inaccurate and uncertain when the measurements are noisy and partial. Instead of simply relying on Assumption 2, the obstacle estimation can be adapted online as more measurements are gathered, as shown in Fig. 11. In particular, this process involves three steps that are executed at a desired frequency: (i) clustering the measurements to identify potential obstacle regions; (ii) fitting the geometric models to these clusters for improved accuracy; and (iii) decomposing the fitted models to refine the obstacle representation.

More specifically, **Clustering**: Given the set of measurements $\mathbf{D}_t$ at the current time step t, the data points are clustered into K clusters $\mathbf{D}_t = \{\mathbf{D}_{t,1}, \ldots, \mathbf{D}_{t,K}\}$ by their relative distances and curvature. **Fitting**: The data points in each cluster $\mathbf{D}_{t,k}$ are then fitted to a squircle $\hat{\mathcal{O}}_{t,k}$ using nonlinear least squares methods (Gavin, 2019), $\forall k = 1, \ldots, K$. If the fitting error E_k falls below a predefined threshold, the estimated model $\hat{\mathcal{O}}_{t,k}$ is accepted, by which either a new squircle is added to the forest world or the geometric parameters of an existing squircle is updated. Since the fitting process is nonlinear, multiple estimated models with comparable fitting errors may exist. In such cases, the model that is most *conservative* is selected. For instance, if only one side of a rectangular squircle is detected, a minimum width or height is assumed for the missing side, as shown in Fig. 11. **Decomposition**: If the fitting error E_k is larger than the threshold, it indicates that this cluster $\mathbf{D}_{t,k}$ might not correspond to a single squircle and should be decomposed into multiple segments $\mathbf{D}_{t,k} = \{\mathbf{S}_{t,1}, \ldots, \mathbf{S}_{t,L}\}$, each of which is then fitted to a squircle model $\hat{\mathcal{O}}_{t,\ell}$, $\forall \ell = 1, \ldots, L$. Namely, the curvature of each point $p_j \in \mathbf{D}_{t,k}$ is captured locally based on its neighboring points. Then, the cluster $\mathbf{D}_{t,k}$ is divided into several segments based on the changes in curvatures between consecutive points. Moreover, if the estimated squircle overlaps with the robot or the regions of interest, the cluster should also be further decomposed to avoid these regions. This process is repeated until all segments are fitted to squircles with the accepted accuracy.

More technical details can be found in the supplementary files in Wang (2023). Some illustrations are shown in Fig. 11, with more numerical examples provided in Section 5. Via this adaptive scheme, both the obstacle parameters and the structure of the forest world can be updated online.

4.2.2. Incremental update of harmonic potentials

The classic methods as proposed in Li and Tanner (2018), Loizou and Rimon (2022) and Rimon (1990) construct the underlying navigation function at once by taking into account *all*

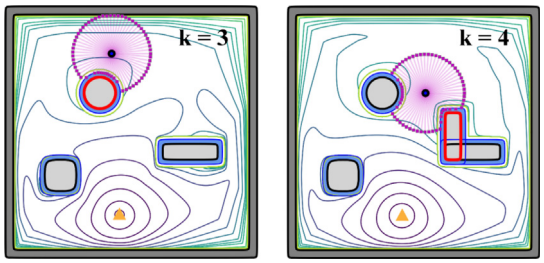


Fig. 12. Iterative update of the harmonic potentials as new obstacles (red lines) are added: an *independent* obstacle for $k = 3$ (**Left**); and an *overlapping* obstacle for $k = 4$ (**Right**).

obstacles in the workspace. However, this means that the whole process has to be repeated from scratch each time an additional obstacle is added. A more intuitive approach would be to incrementally update different parts of the computation process by storing and re-using some of the intermediate results. Consider the following *two* cases as shown in Fig. 12: (i) independent stars are added to existing world of stars; and (ii) overlapping stars are added to existing forests of stars.

Independent Stars: To begin with, consider the simple case that a set of non-overlapping star obstacles, denoted by $\mathcal{O}_k$, $k = 1, \ldots, M$, is added to an empty workspace $\mathcal{W} = \mathcal{O}_0$ in sequence. As discussed in Section 4.1.2, the star-to-sphere transformation is constructed via the translated scaling map in (11).

Definition 4. The online omitted product $\overline{\beta}_i^k(q)$ of star-to-sphere transformation for obstacle $\mathcal{O}_i$ after adding the kth obstacle, for $i = 0, 1, \ldots, k$ and for $k = 1, \ldots, M$, is a real-valued function defined for the star world $\mathcal{S}$ as: $\overline{\beta}_i^k(q) \triangleq \prod_{j=0, j\neq i}^{k} \beta_j(q)$, where $\beta_j(q)$ is the obstacle function defined in (4). ■

Definition 5. The online analytic switch $s_i^k(q)$ of star-to-sphere transformation for obstacle $\mathcal{O}_i$, after adding the kth obstacle, for $i = 0, 1, \ldots, k$ and for $k = 1, \ldots, M$, is a real-valued function defined for a star world $\mathcal{S}$ as:

$$s_i^k(q) \triangleq \frac{\gamma_G(q)\overline{\beta}_i^k(q)}{\lambda_k \beta_i(q) + \gamma_G(q)\overline{\beta}_i^k(q)},$$

where $\gamma_G(q)$ is the distance-to-goal function; λ_k is a positive parameter associated with k; $\overline{\beta}_i^k(q)$ is the online omitted product; and $\beta_i(q)$ is the obstacle function defined in (4). ■

Then, the transformation from the star world to the sphere world after adding the kth star is restated as:

$$\Phi_{\mathcal{S}\to\mathcal{M}}^k(q) \triangleq \mathrm{id}(q) + \sum_{i=0}^{k} s_i^k(q)\left(v_i(q) - 1\right)(q - q_i), \tag{27}$$

where $s_i^k(q)$ is the online analytic switch associated with both i and k; $v_i(q)$ is the scaling factor defined in (10); and q_i is the geometric center of the obstacle $\mathcal{O}_i$, for $i = 0, 1, \ldots, k$. Moreover, the following intermediate variables are defined:

$$\mathbf{F}_i^k(q) \triangleq s_i^k(q)\,\mathbf{r}_i(q), \tag{28}$$

where $\mathbf{r}_i \triangleq \left(v_i(q) - 1\right)(q - q_i)$. Consequently, the transformation in (27) is adjusted to: $\Phi_{\mathcal{S}\to\mathcal{M}}^k(q) = \mathrm{id}(q) + \sum_{i=0}^{k} \mathbf{F}_i^k(q)$. Note that for $k = 0$, the function $\mathbf{F}_0^0(q)$ for the empty workspace is computed using the initial transformation via (12). Afterwards, each time a new $(k+1)$-th obstacle is added, $\mathbf{F}_i^k(q)$ should be updated to $\mathbf{F}_i^{k+1}(q)$, $\forall i = 0, 1, \ldots, k$, and $\mathbf{F}_{k+1}^{k+1}(q)$ for the added obstacle should also be constructed. To compute these variables efficiently, a recursive method is proposed here. For the ease of notation, define the multi-variate auxiliary function useful for the recursive computation as follows:

$$\Psi_{\mathrm{a}}(\mathbf{F}, \mathbf{r}, \tilde{\mathbf{r}}, \alpha) \triangleq \frac{\|\mathbf{F}\|}{\alpha\|\mathbf{r}\| + (1-\alpha)\|\mathbf{F}\|}\tilde{\mathbf{r}}, \tag{29}$$

where $\mathbf{F}, \mathbf{r}, \tilde{\mathbf{r}}$ are vectors; $\alpha \in (0, 1)$ is a scalar variable; and $\alpha\|\mathbf{r}\| + (1-\alpha)\|\mathbf{F}\| \neq 0$. Then, the function $\mathbf{F}_0^{k+1}(q)$ associated with the workspace is updated from $\mathbf{F}_0^k(q)$ as follows:

$$\mathbf{F}_0^{k+1}(q) = \Psi_{\mathrm{a}}\left(\mathbf{F}_0^k(q),\ \mathbf{r}_0(q),\ \mathbf{r}_0(q),\ \alpha^k(q)\right), \tag{30}$$

where $\mathbf{r}_0(q)$ is defined in (28) for the workspace; the variable $\alpha^k(q) \triangleq \lambda_{k+1}/\left(\lambda_k\,\beta_{k+1}(q)\right)$; λ_k and λ_{k+1} are positive parameters defined in 5, for $k = 0, 1, \ldots, M$.

Lemma 4. *Each time an independent obstacle $\mathcal{O}_{k+1}$ is added to the workspace, the following holds:*

(i) the online omitted product for $\mathcal{O}_0$ is given by $\overline{\beta}_0^{k+1}(q) = \overline{\beta}_0^k(q)\,\beta_{k+1}(q)$;

(ii) the online analytic switch for $\mathcal{O}_0$ is given by $s_0^{k+1}(q) = \frac{s_0^k(q)}{\alpha^k(q)(1-s_0^k(q))+s_0^k(q)}$, where $\alpha^k(q) = \lambda_{k+1}/\left(\lambda_k\,\beta_{k+1}(q)\right)$.

Proof. (Omitted) Available in the supplementary files. □

Proposition 2. *Each time an independent obstacle $\mathcal{O}_{k+1}$ is added to the workspace, the recursive calculation of $\mathbf{F}_0^{k+1}(q)$ in (30) for $\mathcal{O}_0$ is valid for iteration $k = 0, 1, \ldots, M-1$.*

Proof. (Sketch) The proof is done by induction. Namely, the initial value of $\mathbf{F}_0^k(q)$ is given by $\mathbf{F}_0^0(q) = s_0^0(q)\,\mathbf{r}_0(q)$ with $s_0^0(q) = \frac{\gamma_G(q)}{\lambda_0\beta_0(q)+\gamma_G(q)}$, which satisfies (28). Now assume that $\mathbf{F}_0^\ell(q) = s_0^\ell(q)\,\mathbf{r}_0(q)$ and $s_0^\ell(q) = \frac{\gamma_G(q)\overline{\beta}_0^\ell(q)}{\lambda_\ell\beta_0(q)+\gamma_G(q)\overline{\beta}_0^\ell(q)}$ hold for an integer $\ell \geq 0$. Then, $\mathbf{F}_0^{\ell+1}(q)$ is derived via (30) as:

$$\mathbf{F}_0^{\ell+1}(q) = \frac{\|s_0^\ell(q)\,\mathbf{r}_0(q)\|\,\mathbf{r}_0(q)}{\alpha^\ell(q)\,\|\mathbf{r}_0(q)\| + \left(1-\alpha^\ell(q)\right)\|s_0^\ell(q)\,\mathbf{r}_0(q)\|}.$$

Given (4), it holds that $\beta_0(q) \geq 0$ and $\overline{\beta}_0^\ell(q) \geq 0$, which implies that $s_0^\ell(q) \in [0, 1]$. Therefore, $\mathbf{F}_0^{\ell+1}(q)$ is simplified as $\mathbf{F}_0^{\ell+1}(q) = \frac{s_0^\ell(q)\,\mathbf{r}_0(q)}{\alpha^\ell(q)+(1-\alpha^\ell(q))s_0^\ell(q)}$ Further, Lemma 4 implies that $s_0^\ell(q) = \frac{\alpha^\ell(q)\,s_0^{\ell+1}(q)}{(\alpha^\ell(q)-1)s_0^{\ell+1}(q)+1}$. Combined with $\mathbf{F}_0^{\ell+1}(q)$, it yields that $\mathbf{F}_0^{\ell+1}(q) = s_0^{\ell+1}(q)\,\mathbf{r}_0(q)$, which is consistent with (28), thus concluding the proof. □

Furthermore, the function $\mathbf{F}_{i+1}^{k+1}(q)$ for each inner obstacle $i = 0, 1, \ldots, k$ is calculated from $\mathbf{F}_0^{k+1}(q)$ iteratively by:

$$\mathbf{F}_{i+1}^{k+1}(q) = \Psi_{\mathrm{a}}\left(\mathbf{F}_i^{k+1}(q),\ \mathbf{r}_i(q),\ \mathbf{r}_{i+1}(q),\ \alpha_i(q)\right), \tag{31}$$

where $\mathbf{r}_i(q)$ and $\mathbf{r}_{i+1}(q)$ are defined in (28) for the ith and $(i+1)$-th obstacles; $\alpha_i(q) = (\beta_{i+1}(q))^2/(\beta_i(q))^2$.

Lemma 5. *Each time an independent obstacle $\mathcal{O}_{k+1}$ is added to the workspace, the following holds:*

(i) the online omitted product for $\mathcal{O}_{i+1}$ is given by $\overline{\beta}_{i+1}^{k+1}(q) = \overline{\beta}_i^{k+1}(q)\,\frac{\beta_i(q)}{\beta_{i+1}(q)}$;

(ii) the online analytic switch for $\mathcal{O}_{i+1}$ is given by $s_{i+1}^{k+1}(q) = \frac{s_i^{k+1}(q)}{\alpha_i(q)(1-s_i^{k+1}(q))+s_i^{k+1}(q)}$, where $\alpha_i(q) = (\beta_{i+1}(q))^2/(\beta_i(q))^2$.

Proof (*Omitted*)**.** Available in the supplementary files. □

Proposition 3. *Each time an independent obstacle $\mathcal{O}_{k+1}$ is added to the workspace, the recursive calculation of $\mathbf{F}_{i+1}^{k+1}(q)$ in (30) for $\mathcal{O}_{i+1}$ is valid for iteration $i = 0, 1, \ldots, k$.*

Proof (*Sketch*)**.** Similar to Proposition 2, the proof is done by induction. The initial value of $\mathbf{F}_i^{k+1}(q)$ is given by $\mathbf{F}_0^{k+1}(q) = s_0^{k+1}(q)\,\mathbf{r}_0(q)$, which fulfills (28). Now assume that $\mathbf{F}_\ell^{k+1}(q) = s_\ell^{k+1}(q)\,\mathbf{r}_\ell(q)$ with $s_\ell^{k+1}(q) = \frac{\gamma_G(q)\overline{\beta}_\ell^{k+1}(q)}{\gamma_G(q)\overline{\beta}_\ell^{k+1}(q)+\lambda_{k+1}\beta_\ell(q)}$ holds for an integer $\ell \geq 0$. Then, $\mathbf{F}_{\ell+1}^{k+1}(q)$ is derived via (30) as:

$$\mathbf{F}_{\ell+1}^{k+1}(q) = \frac{\|s_\ell^{k+1}(q)\,\mathbf{r}_\ell(q)\|\,\mathbf{r}_{\ell+1}(q)}{\alpha_\ell(q)\,\|\mathbf{r}_\ell(q)\| + \big(1-\alpha_\ell(q)\big)\,\|s_\ell^{k+1}(q)\,\mathbf{r}_\ell(q)\|}.$$

Since $s_\ell^{k+1}(q) \in [0, 1]$ holds, it can be simplified to $\mathbf{F}_{\ell+1}^{k+1}(q) = \frac{s_\ell^{k+1}(q)\mathbf{r}_{\ell+1}(q)}{\alpha_\ell(q)+(1-\alpha_\ell(q))s_\ell^{k+1}(q)}$. Furthermore, Lemma 5 implies that $s_\ell^{k+1}(q) = \frac{s_{\ell+1}^{k+1}(q)}{\alpha_\ell(q)(1-s_{\ell+1}^{k+1}(q))+s_{\ell+1}^{k+1}(q)}$. Combined with $\mathbf{F}_{\ell+1}^{k+1}(q)$, it yields that $\mathbf{F}_{\ell+1}^{k+1}(q) = s_{\ell+1}^{k+1}(q)\,\mathbf{r}_{\ell+1}(q)$, which is consistent with (28), thus concluding the proof. □

Remark 4. Note that $\mathbf{F}_0^{k+1}(q)$ in (30) is iterated over the index k with initial value $\mathbf{F}_0^0(q)$ while $\mathbf{F}_{i+1}^{k+1}(q)$ in (31) is iterated over the index i with initial value $\mathbf{F}_0^{k+1}(q)$. The reason is that as the $(k+1)$-th obstacle is added to the existing star world, $\mathbf{F}_0^k$ for the outer boundary should be updated to $\mathbf{F}_0^{k+1}$ first, and then the transformation $\mathbf{F}_{i+1}^{k+1}$ for each inner obstacle $i = 0, 1, \ldots, k$ is constructed incrementally based on $\mathbf{F}_0^{k+1}$. ■

Overlapping Stars: Secondly, it is also possible that the new obstacle is overlapping with an existing obstacle. More precisely, assume that a sequence of star obstacles, denoted by $\{\mathcal{O}_k, k = 1, \ldots, N\}$, is added to a star world and overlapping with one of the existing obstacles as a leaf. As discussed in Section 4.1.2 for the static scenario, the purging transformation (14) for the leaf obstacles is also constructed by ray scaling process(18). To be specific, the purging transformation for the kth obstacle in one of the tree is restated as:

$$f_k(q) \triangleq \mathrm{id}(q) + \sigma_k(q)\,\big(v_k(q)-1\big)\,(q-p_k), \tag{32}$$

where $\sigma_k(q)$ is the online analytic switch associated with k; $v_k(q)$ is the scaling factor defined in (18); and p_k is the common center between the obstacle $\mathcal{O}_k$ and its parent $\mathcal{O}_k^\star$.

Definition 6. The online omitted product $\overline{\beta}_k(q)$ of leaf-purging transformation for obstacle $\mathcal{O}_k$, after adding the kth obstacle for $k = 1, \ldots, N$, is defined on the forest world $\mathcal{F}$ as:

$$\overline{\beta}_k(q) \triangleq \Big(\prod_{j=0, j\neq k^\star}^{k-1} \beta_j(q)\Big)\Big(\prod_{j\in\mathcal{L}\setminus\{k\}} \beta_j(q)\Big)\tilde{\beta}_k(q), \tag{33}$$

where $\tilde{\beta}_k(q)$ is defined in (19); $\beta_j(q)$ is the obstacle function defined in (4); and $\mathcal{L}$ is the set of indices for all leaves in $\mathcal{F}$. ■

Definition 7. The online analytic switch $\sigma_k(q)$ of leaf-purging transformation for obstacle $\mathcal{O}_k$, after adding the kth obstacle for $k = 1, \ldots, N$, is defined on the forest world $\mathcal{F}$ as:

$$\sigma_k(q) \triangleq \frac{\gamma_G(q)\overline{\beta}_k(q)}{\xi_k\beta_k(q) + \gamma_G(q)\overline{\beta}_k(q)}, \tag{34}$$

where $\gamma_G(q)$ is the distance-to-goal function; ξ_k is a positive parameter associated with k; $\overline{\beta}_k(q)$ is the online omitted product; and $\beta_k(q)$ is the obstacle function. ■

Similar to (28), the intermediate variables are defined as:

$$\mathbf{H}_k(q) \triangleq \sigma_k(q)\,\mathbf{r}_k(q), \tag{35}$$

where $\mathbf{r}_k(q) \triangleq \big(v_k(q)-1\big)(q-p_k)$. Consequently, the transformation from forest world to star world after adding the kth star obstacle is given by:

$$\Phi_{\mathcal{F}\to\mathcal{S}}^k(q) \triangleq \mathbf{H}_1 \circ \mathbf{H} \circ \cdots \circ \mathbf{H}_k(q). \tag{36}$$

Note that for $k = 1$, the new obstacle forms a new tree of stars and the transformation function $\mathbf{H}_1(q)$ is computed via (19). Afterwards, each time a new $(k+1)$-th obstacle is added, the transformation (36) should be updated as follows:

$$\Phi_{\mathcal{F}\to\mathcal{S}}^{k+1}(q) = \Phi_{\mathcal{F}\to\mathcal{S}}^k \circ \mathbf{H}_{k+1}(q), \tag{37}$$

where $\mathbf{H}_{k+1}(q)$ is calculated from $\mathbf{H}_k(q)$ iteratively by:

$$\mathbf{H}_{k+1}(q) = \Psi_{\mathrm{a}}\left(\mathbf{H}_k(q),\ \mathbf{r}_k(q),\ \mathbf{r}_{k+1}(q),\ \alpha_k(q)\right), \tag{38}$$

where $\Psi_{\mathrm{a}}(\cdot)$ is defined in (29); $\mathbf{r}_{k+1}(q)$ and $\mathbf{r}_k(q)$ are defined in (35); and $\alpha_k(q)$ is another intermediate variable, given by:

$$\alpha_k(q) = \zeta_k\,\frac{\beta_{k+1}(q)\,\tilde{\beta}_k(q)\,\beta_{(k+1)^\star}(q)}{(\beta_k(q))^3\,\tilde{\beta}_{k+1}(q)\,\beta_{k^\star}(q)},$$

where $\zeta_k = \xi_{k+1}/\xi_k$ with ξ_k and ξ_{k+1} being the design parameters from (14); $\tilde{\beta}_{k+1}(q)$ and $\tilde{\beta}_k(q)$ are defined as in (19).

Lemma 6. *Each time an obstacle $\mathcal{O}_{k+1}$ is added to the workspace and overlapping with an existing obstacle, the following holds:*

(i) the online omitted product for $\mathcal{O}_{k+1}$ is given by $\overline{\beta}_{k+1}(q) = \overline{\beta}_k(q)\,\frac{(\beta_k(q))^2\,\tilde{\beta}_{k+1}(q)\,\beta_{k^\star}(q)}{\tilde{\beta}_k(q)\,\beta_{(k+1)^\star}(q)}$;

(ii) the online analytic switch for $\mathcal{O}_{k+1}$ is given by: $\sigma_{k+1}(q) = \frac{\sigma_k(q)}{\alpha_k(q)(1-\sigma_k(q))+\sigma_k(q)}$, where $\alpha_k(q) = \frac{\xi_{k+1}\,\beta_{k+1}(q)\,\tilde{\beta}_k(q)\,\beta_{(k+1)^\star}(q)}{\xi_k\,(\beta_k(q))^3\,\tilde{\beta}_{k+1}(q)\,\beta_{k^\star}(q)}$.

Proof (*Omitted*)**.** Available in the supplementary files. □

Proposition 4. *Each time an obstacle $\mathcal{O}_{k+1}$ is added to the workspace and overlapping with an existing obstacle, the recursive calculation of $H_{k+1}(q)$ in (38) is valid, for each new leaf obstacle $k = 0, 1, \ldots, N-1$.*

Proof. Similar to Proposition 3, the proof is done by induction. The initial value of $\mathbf{H}_k(q)$ is given by $\mathbf{H}_1(q) = \sigma_1(q)\,\mathbf{r}_1(q)$, which satisfies (35). Assume that $\mathbf{H}_\ell(q) = \sigma_\ell(q)\,\mathbf{r}_\ell(q)$ with $\sigma_\ell(q) = \frac{\gamma_G(q)\overline{\beta}_\ell(q)}{\gamma_G(q)\overline{\beta}_\ell(q)+\xi_k\beta_\ell(q)}$ holds for an integer $\ell \geq 0$. Then, $\mathbf{H}_{\ell+1}(q)$ is computed via (38) as:

$$\mathbf{H}_{\ell+1}(q) = \frac{\|\sigma_\ell(q)\,\mathbf{r}_\ell(q)\|\,\mathbf{r}_{\ell+1}(q)}{\alpha_\ell(q)\,\|\mathbf{r}_\ell(q)\| + (1-\alpha_\ell(q))\,\|\sigma_\ell(q)\,\mathbf{r}_\ell(q)\|}.$$

Since $\sigma_\ell(q) \in [0, 1]$, it holds that $\mathbf{H}_{\ell+1}(q) = \frac{\sigma_\ell(q)\mathbf{r}_{\ell+1}(q)}{\alpha_\ell(q)+(1-\alpha_\ell(q))\sigma_\ell(q)}$. When combined with $\sigma_\ell(q) = \frac{\sigma_{\ell+1}(q)}{\alpha_\ell(q)(1-\sigma_{\ell+1}(q))+\sigma_{\ell+1}(q)}$ from Lemma 6, $\mathbf{H}_{\ell+1}(q) = \sigma_{\ell+1}(q)\,\mathbf{r}_{\ell+1}(q)$ holds, which is consistent with (35), thus concluding the proof. □

Remark 5. Note that the *same* auxiliary function $\Psi(\cdot)$ in (29) is used for the recursive computation in (30), (31) and (38). ■

Remark 6. The parameters λ_k in Definition 5 for $k = 0, 1, \ldots, M$, and ξ_k in Definition 7 for $k = 1, 2, \ldots, N$ should satisfy the conditions in Lemma 2, i.e., $\lambda_k > \Lambda_k$ for some positive numbers $\Lambda_k > 0$; and $\xi_k > \Xi_k$ for some positive numbers $\Xi_k > 0$. ■

Combination of both cases: Overall, the above two cases can be summarized into a more general formula. Namely, the

recursive transformation from forest of stars to sphere world after adding the kth obstacle $\mathcal{O}_k$ is given by:

$$\Phi^k_{\mathcal{F}\to\mathcal{M}}(q) \triangleq \Phi^k_{\mathcal{S}\to\mathcal{M}} \circ \Phi^k_{\mathcal{F}\to\mathcal{S}}(q), \tag{39}$$

where if the new obstacle $\mathcal{O}_{k+1}$ is added to the workspace as an independent star, $\Phi^k_{\mathcal{S}\to\mathcal{M}}$ is updated to $\Phi^{k+1}_{\mathcal{S}\to\mathcal{M}}$ by (27); on the other hand, if the new obstacle $\mathcal{O}_{k+1}$ is overlapping with any existing obstacle, $\Phi^k_{\mathcal{F}\to\mathcal{S}}$ is updated to $\Phi^{k+1}_{\mathcal{F}\to\mathcal{S}}$ by (37). Moreover, each time an independent star $\mathcal{O}_{M+1}$ is added to the workspace, the *complete* harmonic potential function defined in point world should be updated by the recursive rule:

$$\phi^{k+1}_{\mathcal{P}}(x) = \frac{K}{K+1}\,\phi^k_{\mathcal{P}}(x) + \frac{1}{K+1}\big(\phi(x,\ P_d) + \phi(x, P_{M+1})\big),$$

where $\phi^k_{\mathcal{P}}(x)$ and $\phi^{k+1}_{\mathcal{P}}(x)$ are the potential function in the kth and $(k+1)$-th step; $\phi(x, P_{M+1})$ is the harmonic term for the new point obstacle. Given the updated transformation $\Phi^{k+1}_{\mathcal{F}\to\mathcal{M}}$ and the underlying harmonic potentials in point world $\phi^{k+1}_{\mathcal{P}}$, the new navigation function is adapted to $\varphi^{k+1}_{\mathrm{NF}}$ accordingly.

4.2.3. Local revision of harmonic trees

As new obstacles are detected, parts of some Oriented Harmonic Trees $\{\mathcal{T}_{\widehat{g}_\ell\to\widehat{g}_{\ell+1}}\}$ are revised in *three* main steps: (I) *Trimming of Harmonic trees*. For each HT $\mathcal{T}_{\widehat{g}_\ell\to\widehat{g}_{\ell+1}}$, find the vertices that are within the newly-added obstacle, i.e., $v \in \mathcal{O}_k$, and remove them from the set of vertices V, along with the attached edges; (II) *Potential-biased regeneration*. To re-connect the oriented HT to the goal vertex, new vertices and edges are generated.

Depending on the particular method of discretization, new vertices could be generated with the bias, e.g., more towards the area with large change in its potential value; (III) *Path revision*. Once new vertices and edges are added, the associated edge costs should be re-evaluated. Suppose that at each time of update H edges have been traversed in total, of which their actual costs are given by $\boldsymbol{\gamma}^\star = (\gamma_1^\star, \ldots, \gamma_H^\star)$. Their estimated costs are $\bar{\gamma} = (\gamma_1, \ldots, \gamma_H)$ with $\bar{\mathbf{d}}$ and $\bar{\mathbf{Q}}$ being the distance and rotation cost from (26). Then, the weighting parameters can be updated by: $\mathbf{w}^\star = (\bar{\mathbf{Q}}^\intercal\bar{\mathbf{Q}})^{-1}\bar{\mathbf{Q}}^\intercal\,(\boldsymbol{\gamma}^\star - \bar{\mathbf{d}})$, which is then used to update all edge costs, even for other HTs. Afterwards, the revised path is found via the same search algorithm to the goal vertex.

4.2.4. Asymptotic adaptation of task plan

On the highest level, since the HTs are updated, the associated feasibility and costs in the navigation map $\mathcal{G}$ should be updated accordingly, which in turn leads to a different task plan $\hat{\mathbf{g}}$. More specifically, the navigation map $\mathcal{G}$ is updated in *two* steps: (i) the feasibility of navigation from $\widehat{g}$ to $\widehat{g}'$ in $\mathcal{G}$ is re-evaluated based on whether the path $\mathbf{P}_{\widehat{g}\to\widehat{g}'}$ exists within the HT $\mathcal{T}_{\widehat{g}\to\widehat{g}'}$. If not, the transition $(g,\ g')$ is removed from the edge set E. This is often caused by newly-discovered obstacles blocking some passages; (ii) the costs of transitions within E are re-computed given the updated path within the new HTs. Consequently, the navigation map $\mathcal{G}$ is up-to-date with the latest environment model. Thus, the task plan $\widehat{\mathbf{g}}^\star$ is re-synthesized within the updated product $\widehat{\mathcal{A}}$ by searching for a path from the current product state to an accepting state. In other words, as the environment is gradually explored, the suffix of the task plan often would converge to optimal one asymptotically.

4.3. Complexity analyses

The computational complexity to construct the product $\widehat{\mathcal{A}}$ is $\mathcal{O}(|G|^2|\mathcal{A}_\varphi|^2)$, where $|G|$ is the number of goal regions and $|\mathcal{A}_\varphi|$ is the number of states within $\mathcal{A}_\varphi$. The complexity to construct a HT $\mathcal{T}$ is $\mathcal{O}(|\mathcal{T}|)$, where $|\mathcal{T}|$ is the number of intermediate waypoints in the tree structure. For a forest world with $|\mathcal{I}|$ total stars and $|\mathcal{F}|$ trees of stars with maximum depth d, the complexity to compute the initial Harmonic potential Υ is $\mathcal{O}(|\mathcal{F}|^2 + d|\mathcal{I}|)$. Thus, the complexity to compute the complete $\mathcal{G}$ is $\mathcal{O}(|G|^2|\mathcal{A}_\varphi|^2 + |\mathcal{T}|(|\mathcal{F}|^2 + d|\mathcal{I}|))$. During online adaptation, each time an additional obstacle is added, the recursive update of Υ has complexity $\mathcal{O}(|\mathcal{F}| + |\mathcal{I}|)$. The complexity of local revision of $\mathcal{G}$ and $\mathcal{T}$ are $\mathcal{O}(|\mathcal{G}'|(|\mathcal{F}| + |\mathcal{I}|))$ and $\mathcal{O}(|\mathcal{T}'|)$, respectively, where $|\mathcal{G}'|$ is the number of revised vertices, $|\mathcal{T}'|$ is the number of generated intermediate waypoints.

5. Simulation and experiments

To further validate the effectiveness of our proposed method, extensive numerical simulations and hardware experiments are conducted, against several state-of-the-art approaches. The proposed method is implemented in Python3 and tested on an Intel Core i7-1280P CPU. More descriptions, accompanied videos, and source code can be found in the website (Wang, 2023).

5.1. Description of robot and task

Consider a unicycle robot with the dynamics in (3) that operates within a workspace measuring 7 m × 4 m, with an initial model depicted in Fig. 3. The robot occupies a circular area of radius $r_{\mathrm{r}} = 0.1m$, and is equipped with a Lidar sensor capable of detecting objects up to a maximum range of $1.0m$. The control gains as specified in (24) are set to $k_\upsilon = 1.0$ and $k_\omega = 0.8$, such that the tracking accuracy is below 0.1 m. Both the robot control and the Lidar measurements are updated at 10 Hz. The design parameter for the smooth switch function in (21) and (22) is set to 0.5. The parameter K in the harmonic potentials defined in (2) is set to 2 initially. Each time an independent obstacle is added to the workspace, K is increased by 1. In addition, the parameter μ is set to 1, while the geometric parameters E_i and E_G are both set to 0.02. The parameters λ_k, ξ_k in Definition 5 and 7 are set to 5.0×10^2 and 1.0×10^5, i.e., sufficiently large to accommodate the complex environment. The weight parameter $\mathbf{w}$ utilized in the control cost estimation in (26) is initially given as $[0.1,\ 0.1]^\intercal$. The set of vertices V in (7) is generated by the visibility graph from Huang and Chung (2004) with cost estimation based on (26) and a safety buffer of 0.15 m. Fig. 3 illustrates the complete environment, which mimics a complex office setting with numerous overlapping obstacles. Initially, the robot *only knows* the workspace boundary, the task regions and none of the obstacles.

Regarding the robot task, the workspace consists of a set of regions of interest, denoted by $p_1, p_2, d_1, d_2, d_3, u_1$. The high-level task requires the robot to transport objects from either the storage room p_1 or p_2 to the destinations d_1, d_2, and d_3. Moreover, during runtime, a contingent task might be triggered by an external event and requested in region u_1. In such cases, the robot must prioritize this task for execution. This can be expressed as the formula $\varphi = \big(\ \Diamond\big((p_1 \vee p_2) \wedge (\Diamond d_1)\big) \wedge \big(\ \Diamond\big((p_1 \vee p_2) \wedge (\Diamond d_2)\big) \wedge \big(\ \Diamond\big((p_1 \vee p_2) \wedge (\Diamond d_3)\big) \wedge (\circ \to u_5)$. It is worth noting that there exist numerous high-level plans to fulfill the specified tasks, and their actual costs rely heavily on the complete workspace, which can only be explored online.

5.2. Results

As visualized in Fig. 13, the robot starts from the initial position (3.2 m, 0.4 m) with the orientation π. The task automaton $\mathcal{A}_\varphi$ is constructed with 29 states and 474 edges. The FTS associated with the regions of interest is initialized as a fully-connected graph that each edge is built as a HT consisting of 4 intermediate waypoints, with an average computation time of $0.08s$. Then, the product automaton $\widehat{\mathcal{A}}$ is constructed with 192 states and 1435 edges, of which the initial task plan is $P_0 = p_1 d_1 d_3 d_2$.

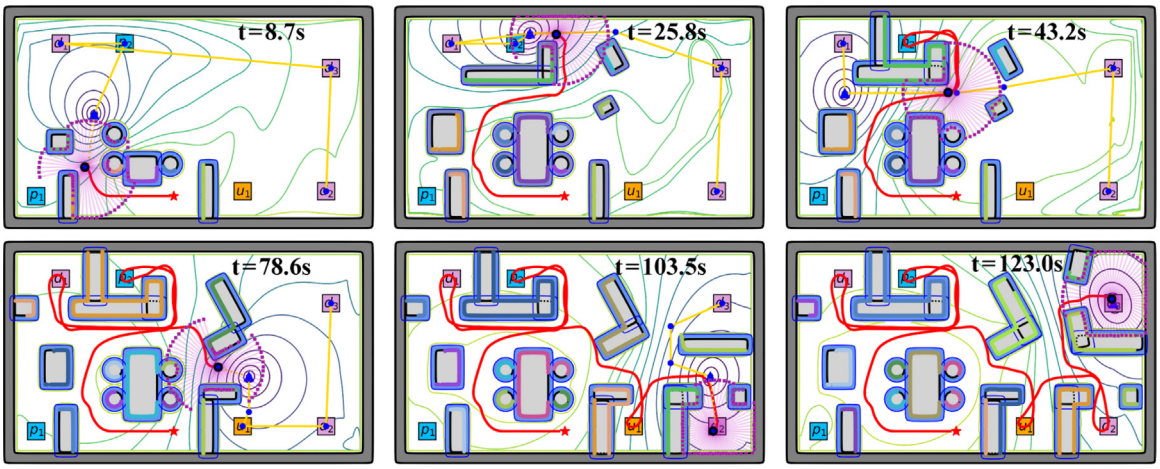


Fig. 13. Snapshots of simulation when the high-level plan and the underlying harmonic potentials are updated, as the robot explores gradually the workspace. The robot trajectory is depicted in red; the local goal of φ_{NF} are highlighted in blue triangle; and the intermediate waypoints in $\mathcal{T}$ are marked in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Guided by this high-level plan, the robot navigates first towards task p_1 along the intermediate waypoints. Between any pair of the waypoint (v_n, v_{n+1}), the initial harmonic potential φ_{NF} is constructed by (9) with an average computation time of 10.46 ms. Then, the oriented harmonic fields Υ are constructed with almost no additional time, given the desired orientation θ_{n+1}. The switch function in (25) is utilized with $k_s = 2.0$ and $\epsilon = 0.1$. Furthermore, the underlying HT and high-level plan are updated at discrete instants during execution, e.g., at $t = 8.7s$ and $43.2s$ when new obstacles are detected and estimated, blocking the way to the next task region. In particular, at $t = 8.7s$, the estimated costs for the plans $p_1d_1d_3d_2$ and $p_2d_1d_3d_2$ are 22.91 and 16.07, respectively. Thus, the robot moves toward p_2 instead of p_1 due to the higher cost associated with rotation. The mean computation time for the new navigation functions at $t = 8.7$ s and 43.2 s is 14.06 ms and 15.82 ms, respectively. The weight $\mathbf{w}$ in the control cost is updated to $[0.67, 0.34]^\intercal$ given the traversed edges in HT. Afterwards, the navigation map $\mathcal{G}$ is updated in 0.16 s given the new visibility graph. At $t = 78.6$ s, the urgent task in region u_1 is triggered, thus the new subtask is added and the associated product automaton is reconstructed to get the new task plan $u_1d_2d_3$, as shown in Fig. 13. Finally, the whole task is accomplished in 123 s and the resulting trajectory has a total distance 27.29 m. It is worth pointing out that the final estimated forest world is *not the same* as the actual workspace in Fig. 3, e.g., the L-shape obstacle to the middle-left, and the squircle to the bottom-right are not fully explored.

5.3. Comparisons

To further demonstrate the effectiveness of our proposed Harmonic Tree (HT) structure, a quantitative comparison is conducted against **five** baselines: (i) the oriented harmonic potential fields (OHPF) proposed in this work with the high-level task adaptation, but omitting the second-layer search trees; (ii) the Navigation Function (NF) from Loizou (2017), Rimon (1990) without the high-level task adaptation; (iii) the harmonic maps (HM) in Vlantis et al. (2018) via the open-sourced implementation; (iv) the Non-holonomic RRT* in LaValle (2006) and Park and Kuipers (2015) without the high-level task adaptation; and (v) the conformal navigation transformation (CNT) from Fan et al. (2022), which has to be modified considerably for the complex workspace here. As summarized in Table 1, the metrics to compare are computation time for each plan synthesis and adaptation, the total cost of final trajectory as the distance traveled, and whether oscillations appear during execution.

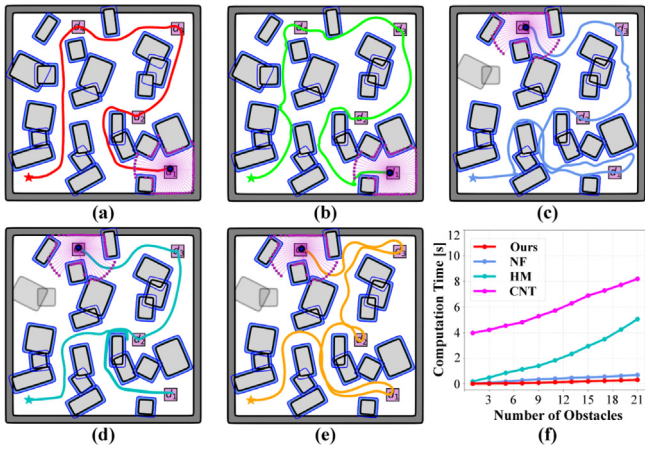


Fig. 14. (**a**-**e**) Final trajectories via the proposed method (in red), OHPF without the search trees (in green), NF without plan adaptation (in blue), HM without plan adaptation (in cyan), and RRT* without plan adaptation (in orange). The obstacles marked in blue are detected online, while those in light gray are undetected; (**f**) Comparison of computation time between our incremental method (in red) and the baselines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Comparison with baselines.

Method	Planning Time [s]	Execution Time [s]	Travel Distance [m]	Adaptation	Oscillation
HT(ours)	4.28	**47.64**	**15.32**	**Yes**	**No**
OHPF	3.19	65.80	18.75	**Yes**	Yes
NF	**0.0**	85.23	22.16	No	Yes
HM	13.09	54.29	18.85	No	**No**
RRT*	97.68	69.81	20.50	No	**No**
CNT	47.26	77.27	19.64	No	**No**

For a more detailed comparison, an obstacle cluttered workspace of size 5.0 m × 5.0 m with 20 overlapping obstacles is considered as shown in Fig. 14. It is worth noting that the associated forest world has a maximum depth of 4, which is quite difficult for model-based methods such as NF, OHPF and ours due to the purging process. The robot starts from (0.4 m, 0.4 m) and has a sensing range of 1.0 m. The task requires the robot to surveil the regions of interest d_1, d_2, d_3, d_4 in an arbitrary order. Other parameters are same to Section 5.1. The final trajectories and numerical results are shown in Fig. 14 and Table 1. The

proposed HT exhibits lowest cost for task completion with a fast planning and adaptation, with no collision or oscillations during execution. As for NF, a blind execution of the initial plan leads to a more costly trajectory with numerous oscillations since the gradients near the obstacle cannot be tracked perfectly. Besides, the nominal NF cannot control the final orientation as the robot approaches the task areas, resulting in high steering cost when task regions switch. In contrast, OHPF optimizes the orientations at each task region and adapts the high-level plan online as the proposed HT, leading to a much smaller execution time and overall cost. However, it is worth noting that without the guidance of search trees, oscillations can still occur close to obstacles for OHPF. Further, RRT* takes significantly more time to synthesize the initial plan and adapt the new plan, due to high computationally complexity of collision detection between samples. Namely, it takes around 98 s to compute the complete plan for RRT*, compared with merely 4.3 s by the proposed HT. The HM method has a travel distance of 18.85 m with a total planning time 13.09 s, which includes 20 times of replanning and each replanning takes 0.65 s. Each obstacle is represented by average 80 boundary points, which is the minimum number that can ensure safety in our tests. Similarly, the CNT method takes significantly more planning time and execution time (close to 26 s and 77 s, with 20 times of replanning).

Lastly, the computational efficiency of the proposed iterative approach is compared with nominal NF, HM, and CNT, as summarized in Fig. 14. As the number of obstacles increases, the computation time of our method remains relatively low due to its analytical computation, incremental update and the hierarchical structure. In contrast, the nominal NF method requires nearly twice the time due to the frequent re-calculation of the potentials from scratch. Furthermore, the computation time for CNT increases drastically with the number of boundary points (each obstacle has 150 boundary points), e.g., each replanning takes more than 8.0 s for CNT with 20 obstacles.

5.4. Hardware experiments

The proposed method is deployed to a differential-driven robot of radius $r_r = 0.2m$. As shown in Fig. 15, the workspace is constructed with dimensions of 5.2 m × 5.2 m, which has 4 independent and 4 overlapping obstacles. The controller gains in (24) are set to $k_\upsilon = 0.1$ and $k_\omega = 0.2$, ensuring that the tracking error remains below 0.2 m. The robot state is estimated using SLAM technology, while the communication between the robot and the workstation is achieved by ROS with a frequency of 10 Hz. The point clouds are collected by a forward-facing 180° Lidar within a radius of 8.0 m around the robot. Other parameters are similar to the simulation. The task is designed to transport objects from p_1 to d_1, d_2, and d_3. The robot starts from its initial position at $(0.0\ m, 0.0\ m)$. The initial task plan is derived within $0.09s$ as $p_1d_1d_2d_3$. The robot adjusts its task plan and harmonic potentials as 6 obstacles are detected online. As the robot detects more obstacles and familiar with the workspace, the task plan is updated as $p_1d_2d_3d_1$ and finally converges to $p_1d_2d_3d_1$. The whole task is accomplished in 338 s, of which the complete video can be found in Wang (2023) The resulting trajectory and snapshots are shown in Fig. 15 with a total distance 21.74 m, which is safe and smooth despite of the localization and control uncertainties. These results are consistent with the simulations.

6. Conclusion

This work proposes an automated framework for task and motion planning, employing harmonic potentials for navigation and oriented search trees for planning. The design and construction of the search tree is specifically customized for the task automaton and co-designed with the underlying navigation controllers based on harmonic potentials. Efficient and secure task execution is ensured for partially-known workspace. As described earlier, although an online approach is proposed to adapt the forest world and obstacle estimation during execution, it lacks a systematic analysis for more general workspaces, which is part of our ongoing work. Moreover, the extension to 3D navigation remains our future work.

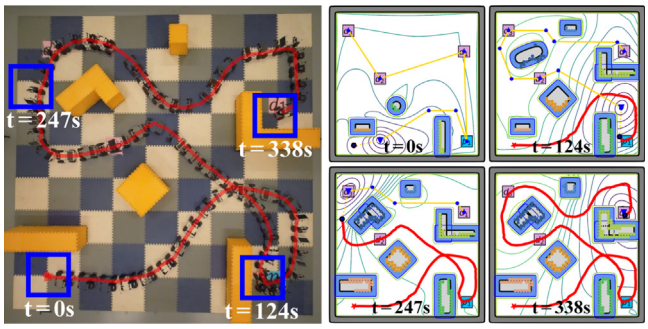


Fig. 15. Left: Recorded execution results; **Right:** Snapshots of robot trajectories and the potential fields, where obstacles in blue are detected online. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

References

Baier, C., & Katoen, J.-P. (2008). *Principles of model checking*. MIT Press.

Dahlin, A., & Karayiannidis, Y. (2023a). Creating star worlds: Reshaping the robot workspace for online motion planning. *IEEE Transactions on Robotics*, *39*, 3655–3670.

Dahlin, A., & Karayiannidis, Y. (2023b). Obstacle avoidance in dynamic environments via tunnel-following mpc with adaptive guiding vector fields. In *IEEE conference on decision and control* (pp. 5784–5789).

Fainekos, G. E., Girard, A., Kress-Gazit, H., & Pappas, G. J. (2009). Temporal logic motion planning for dynamic robots. *Automatica*, *45*, 343–352.

Fan, L., Liu, J., Zhang, W., & Xu, P. (2022). Robot navigation in complex workspaces using conformal navigation transformations. *IEEE Robotics and Automation Letters*, *8*, 192–199.

Faulwasser, T., & Findeisen, R. (2015). Nonlinear model predictive control for constrained output path following. *IEEE Transactions on Automatic Control*, *61*, 1026–1039.

Filippidis, I., & Kyriakopoulos, K. J. (2011). Adjustable navigation functions for unknown sphere worlds. In *IEEE conference on decision and control and European control conference* (pp. 4276–4281).

Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., & Lozano-Pérez, T. (2021). Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, *4*, 265–293.

Gavin, H. P. (2019). *The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems*. Department of civil and environmental engineering, Duke University, 19.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated planning: theory and practice*. Elsevier.

Guo, M., Andersson, S., & Dimarogonas, D. V. (2018). Human-in-the-loop mixed-initiative control under temporal tasks. In *IEEE international conference on robotics and automation* (pp. 6395–6400).

Guo, M., & Dimarogonas, D. V. (2015). Multi-agent plan reconfiguration under local ltl specifications. *The International Journal of Robotics Research*, *34*, 218–235.

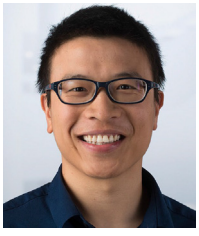
Hsu, D., Latombe, J.-C., & Kurniawati, H. (2006). On the probabilistic foundations of probabilistic roadmap planning. *The International Journal of Robotics Research*, *25*, 627–643.

Huang, H.-P., & Chung, S.-Y. (2004). Dynamic visibility graph for path planning. *3*, In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 2813–2818).

Huber, L., Billard, A., & Slotine, J.-J. (2019). Avoidance of convex and concave obstacles with convergence ensured through contraction. *IEEE Robotics and Automation Letters*, *4*, 1462–1469.

Huber, L., Slotine, J.-J., & Billard, A. (2022). Avoiding dense and dynamic obstacles in enclosed spaces: Application to moving in crowds. *IEEE Transactions on Robotics*, *38*, 3113–3132.
Huber, L., Slotine, J.-J., & Billard, A. (2024). Avoidance of concave obstacles through rotation of nonlinear dynamics. *IEEE Transactions on Robotics*, *40*, 1983–2002.
Janson, L., Schmerling, E., Clark, A., & Pavone, M. (2015). Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research*, *34*, 883–921.
Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, *30*, 846–894.
Khalil, H. (2002). *Nonlinear systems* (3rd ed.). Upper Saddle River, NJ: Prentice Hall.
Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles* (pp. 396–404). Springer.
Khatib, O. (1999). Mobile manipulation: The robotic assistant. *Robotics and Autonomous Systems*, *26*, 175–183.
Kim, J.-o., & Khosla, P. (1992). Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, *8*, 338–349.
Kim, B., Shimanuki, L., Kaelbling, L. P., & Lozano-Pérez, T. (2022). Representation, learning, and planning algorithms for geometric task and motion planning. *The International Journal of Robotics Research*, *41*, 210–231.
Ko, I., Kim, B., & Park, F. C. (2013). Vf-rrt: Introducing optimization into randomized motion planning. In *IEEE Asian control conference* (pp. 1–5).
Koditschek, D. (1987). Exact robot navigation by means of potential functions: Some topological considerations. In *IEEE international conference on robotics and automation (Vol. 4)* (pp. 1–6).
LaValle, S. M. (2006). *Planning algorithms*. Cambridge University Press.
Leahy, K., Serlin, Z., Vasile, C.-I., Schoer, A., Jones, A. M., Tron, R., & Belta, C. (2021). Scalable and robust algorithms for task-based coordination from high-level specifications (scratches). *IEEE Transactions on Robotics*, *38*, 2516–2535.
Li, C., & Tanner, H. G. (2018). Navigation functions with time-varying destination manifolds in star worlds. *IEEE Transactions on Robotics*, *35*, 35–48.
Lindemann, L., Matni, N., & Pappas, G. J. (2021). Stl robustness risk over discrete-time stochastic processes. In *IEEE conference on decision and control* (pp. 1329–1335).
Loizou, S. G. (2011). Closed form navigation functions based on harmonic potentials. In *IEEE conference on decision and control and European control conference* (pp. 6361–6366).
Loizou, S. G. (2017). The navigation transformation. *IEEE Transactions on Robotics*, *33*, 1516–1523.
Loizou, S. G., & Rimon, E. D. (2021). Correct-by-construction navigation functions with application to sensor based robot navigation. ArXiv preprint arXiv:2103.04445.
Loizou, S. G., & Rimon, E. D. (2022). Mobile robot navigation functions tuned by sensor readings in partially known environments. *IEEE Robotics and Automation Letters*, *7*, 3803–3810.
Luo, X., Kantaros, Y., & Zavlanos, M. M. (2021). An abstraction-free method for multirobot temporal logic optimal control synthesis. *IEEE Transactions on Robotics*, *37*, 1487–1507.
Ogren, P., & Leonard, N. E. (2005). A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics*, *21*, 188–195.
Otte, M., & Frazzoli, E. (2016). Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, *35*, 797–822.
Panagou, D. (2014). Motion planning and collision avoidance using navigation vector fields. In *IEEE international conference on robotics and automation* (pp. 2513–2518).
Park, J. J., & Kuipers, B. (2015). Feedback motion planning via non-holonomic rrt for mobile robots. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 4035–4040).
Qureshi, A. H., & Ayaz, Y. (2016). Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots*, *40*, 1079–1093.
Rimon, E. (1990). *Exact robot navigation using artificial potential functions* (Ph.D. thesis), Yale University.
Rousseas, P., Bechlioulis, C., & Kyriakopoulos, K. J. (2021). Harmonic-based optimal motion planning in constrained workspaces using reinforcement learning. *IEEE Robotics and Automation Letters*, *6*, 2005–2011.
Rousseas, P., Bechlioulis, C. P., & Kyriakopoulos, K. J. (2022a). Optimal motion planning in unknown workspaces using integral reinforcement learning. *IEEE Robotics and Automation Letters*, *7*, 6926–6933.
Rousseas, P., Bechlioulis, C. P., & Kyriakopoulos, K. J. (2022b). Trajectory planning in unknown 2d workspaces: A smooth, reactive, harmonics-based approach. *IEEE Robotics and Automation Letters*, *7*, 1992–1999.
Sánchez, I., D'Jorge, A., Raffo, G. V., González, A. H., & Ferramosca, A. (2021). Nonlinear model predictive path following controller with obstacle avoidance. *Journal of Intelligent and Robotic Systems*, *102*, 1–18.
Shen, Z., Wilson, J., Harvey, R., & Gupta, S. (2021). Smarrt: Self-repairing motion-reactive anytime rrt for dynamic environments. arXiv preprint arXiv:2109.05043.
Tahir, Z., Qureshi, A. H., Ayaz, Y., & Nawaz, R. (2018). Potentially guided bidirectionalized rrt* for fast optimal path planning in cluttered environments. *Robotics and Autonomous Systems*, *108*, 13–27.
Valbuena, L., & Tanner, H. G. (2012). Hybrid potential field based control of differential drive mobile robots. *Journal of Intelligent and Robotic Systems*, *68*, 307–322.
Vasilopoulos, V., Pavlakos, G., Schmeckpeper, K., Daniilidis, K., & Koditschek, D. E. (2022). Reactive navigation in partially familiar planar environments using semantic perceptual feedback. *The International Journal of Robotics Research*, *41*, 85–126.
Vlantis, P., Vrohidis, C., Bechlioulis, C. P., & Kyriakopoulos, K. J. (2018). Robot navigation in complex workspaces using harmonic maps. In *IEEE international conference on robotics and automation* (pp. 1726–1731).
Wang, S. (2023). Harmonic tree. https://shuaikang-wang.github.io/HarmonicTree/.
Warren, C. W. (1989). Global path planning using artificial potential fields. In *IEEE international conference on robotics and automation* (pp. 316–317).
Yu, S., Li, X., Chen, H., & Allgöwer, F. (2015). Nonlinear model predictive control for path following problems. *International Journal of Robust and Nonlinear Control*, *25*, 1168–1182.

Shuaikang Wang received the B.E. degree in Robotics Engineering in 2023 from Peking University, Beijing, China. He is currently working toward the M.S. degree in the Department of Mechanics and Engineering Science, College of Engineering, Peking University. His current research interests include planning and control for robotics. He was the finalist for the Best Multi-Robot Systems Paper Award at the IEEE International Conference on Robotics and Automation (ICRA) in 2024.



Meng Guo received the M.Sc. degree (2011) in System, Control, and Robotics and the Ph.D. degree (2016) in Electrical Engineering from KTH Royal Institute of Technology, Sweden.

He was a postdoctoral associate with the Department of Mechanical Engineering and Materials Science, Duke University, USA. During 2018–2021, he worked as a senior research scientist on Reinforcement Learning and Planning at the Bosch Center for Artificial Intelligence (BCAI), Germany.

Since 2022, he is an assistant professor at the Department of Mechanics and Engineering Science, College of Engineering, Peking University, China. His main research interests include task and motion planning for robotic systems.