Hybrid and Oriented Harmonic Potentials for Safe Task Execution in Unknown Environment

Shuaikang Wang and Meng Guo[1]

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

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); LaValle (2006); Koditschek (1987); Khatib (1999)), should be reactive to the newly-discovered obstacles online. Second, the planning method that decomposes and schedules sub-tasks (see e.g., Ghallab et al. (2004); Fainekos et al. (2009)) 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); Kim et al. (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 & Rimon (2022); Rousseas et al. (2022b).

[1]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. Contact: wangshuaikang, meng.guo@pku.edu.cn.

1.1. Related Work

As the most relevant to this work, the method of artificial potential fields from Khatib (1986); Warren (1989); Panagou (2014); Rousseas et al. (2022a) 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 stars*, some key design parameters require fine-tuning for the safety and convergence properties to hold. The work in Fan et al. (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. Moreover, harmonic potentials proposed in Kim & Khosla (1992); Loizou (2011, 2017); Vlantis et al. (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 et al. (2021, 2022b,a) resolves the need for a diffeomorphic mapping onto sphere disks, by adopting a wider set of basis functions for workspace boundaries. Nevertheless, such

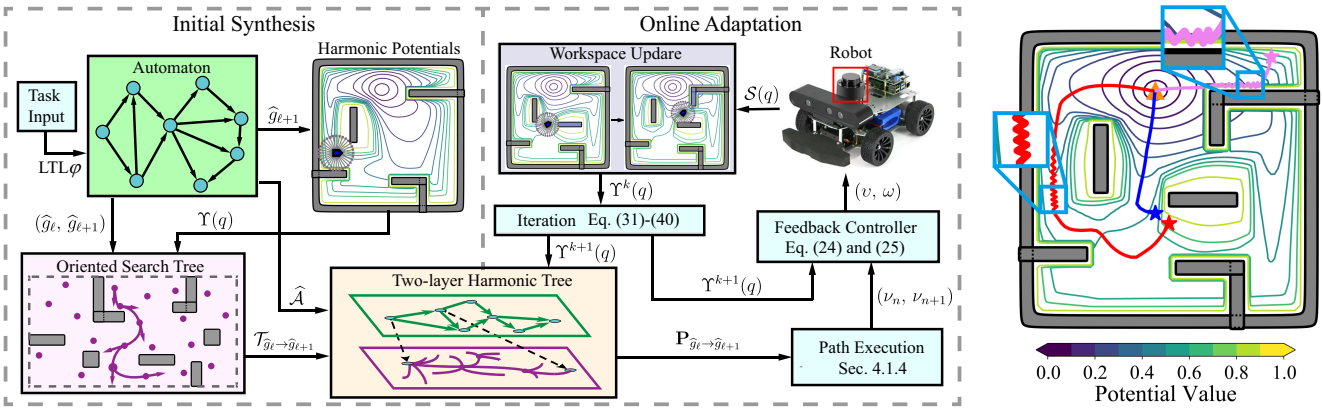

Figure 1: **Left**: Illustration of the proposed framework, which consists of the initial synthesis, the online adaption of the task plan and harmonic potentials, and the hybrid execution. **Right**: Oscillations and long detours might occur via classic navigation functions as shown in the red, violet and blue trajectories.

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 can not be controlled freely.

Moreover, sampling-based search methods, such as RRT* in Karaman & Frazzoli (2011), PRM in Hsu et al. (2006), FMT* in Janson et al. (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 and can ensure safety and convergence as described earlier, it make sense to combine these two paradigms. Some recent work can be found in this direction: vector fields are used in Ko et al. (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 Qureshi & Ayaz (2016); Tahir et al. (2018) as the potential-function-based RRT*, by designing directional samples as induced by the underlying potential fields. Higher efficiency and faster convergence are shown compared with the raw RRT* algorithm. 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 & Kyriakopoulos (2011), while the notion of dynamic windows is proposed in Ogren & 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 & 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 et al. (2022) to recognize the size of the obstacles from a pre-trained dataset. Lastly, the scenario of time-varying targets is analyzed in Li & Tanner (2018) by designing an attractive potential that evolves with time. On the other hand, search-based methods are also extended to unknown environments where various real-time revision techniques are proposed in Otte & Frazzoli (2016); Shen et al. (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 & 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 et al. (2018); Lindemann et al. (2021), a multi-robot system under a global task Guo & Dimarogonas (2015); Luo et al. (2021); Leahy et al. (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 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 search tree and harmonic potentials simultaneously, where the intermediate variables are saved and re-used to avoid total re-computation. 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) a nonlinear controller for non-holonomic robots based on the oriented harmonic potentials, for safe and smooth full-body navigation; (iv) an integrated method to update both 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.

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 $O_0 = \{q \in \mathbb{R}^2 : \|q-q_0\|^2-\rho_0^2 \leqslant 0\}$ centered at q_0 with radius ρ_0, and inner boundaries of M disjoint sphere obstacles $O_i = \{q \in \mathbb{R}^2 : \|q-q_i\|^2-\rho_i^2 \leqslant 0\}$ centered at q_i with radius ρ_i, for $i = 1, \cdots, M$. 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 $\mathcal{P} = \mathbb{R}^2 \backslash \{P_1, \cdots, P_M\}$, 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 $\tilde{\mathcal{P}} = O_0 \backslash \{\tilde{P}_1, \cdots, \tilde{P}_M\}$, 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 $i = 1, \cdots, M$; 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

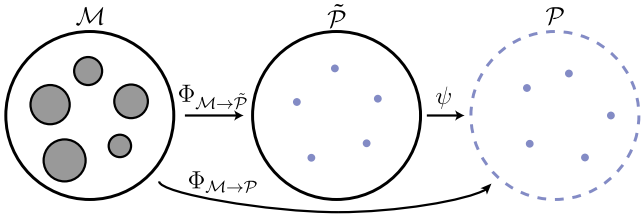

Figure 2: Illustration of diffeomorphic transformation from sphere word $\mathcal{M}$ to unbounded point world $\tilde{\mathcal{P}}$ and to unbounded point world $\mathcal{P}$.

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 & Rimon (2022, 2021) 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 $i = 1, \cdots, M$; $K \geqslant 1$ 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 $\mu \geqslant 1$.

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 & 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. 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 $i \geqslant 0$. A run is called *accepting* if it holds that $\inf(\rho) \cap S_F \neq \varnothing$, 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.

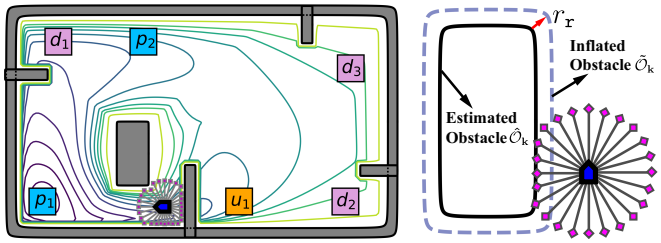

Figure 3: **Left:** Robot in the initially-known workspace with overlapping obstacles and several regions of interest. **Right:** Reconstruction of obstacle model.

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 internal obstacles $\mathcal{O}_i$ such that $\mathcal{O}_i \subset \mathcal{W}_0$ and $\mathcal{O}_i \bigcap \mathcal{O}_j = \varnothing$, $i \neq j$, $\forall i, j \in \{1, \cdots, M\}$. 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 $\mathcal{W} \triangleq \tilde{\mathcal{W}}_0 \backslash \bigcup_{i=1}^{M} \tilde{\mathcal{O}}_i$.

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, θ_0), the robot can navigate within the workspace and observe more obstacles, via a range-limited sensor modeled as follows:

$$\mathcal{S}(q) \triangleq \{\hat{q} \in \mathcal{W} \mid (\hat{q} \in \mathcal{D}_{r_s}(q)) \wedge (\mathcal{L}(q, \hat{q}) \subset \mathcal{W})\}, \tag{4}$$

where $\mathcal{S}(q)$ is the set of boundary 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 straight 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).

Lastly, there is a set of non-overlapping regions of interest $g_n \subset \mathcal{W}$, $n = 1, \cdots, N$. 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 Sec. 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 $0 \leqslant t_k \leqslant t_{k+1}$ 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, θ_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 recrusively and simultaneously.

4.1. Initial Synthesis

This section presents the algorithmic details of the initial planning and control strategy. First, the automaton-guided harmonic trees are constructed given the task automaton and the set of intermediate waypoints, within which the initial discrete plan is derived. Then, a novel method to construct orientation-aware harmonic potentials is introduced, based on which a nonlinear controller is designed for full-state navigation.

4.1.1. Two-layer and Automaton-guided Harmonic Trees

As described in Sec. 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 can only be determined after the associated controllers are constructed. Due to the same reason, the navigation cost is estimated by the Euclidean distance initially, and later updated online.

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

$$\widehat{\mathbf{g}} = \hat{g}_1 \hat{g}_2 \cdots \hat{g}_L (\hat{g}_{L+1} \hat{g}_{L+2} \cdots \hat{g}_{L+H})^\omega, \tag{5}$$

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

The obtained plan $\widehat{\mathbf{g}}$ can be executed by sequentially traversing the transitions $(\hat{g}_\ell, \hat{g}_{\ell+1})$, by designing appropriate potential fields as in Loizou & Rimon (2022); Rousseas et al. (2021) However, as mentioned in Sec. 1.1, when $\hat{g}_\ell, \hat{g}_{\ell+1}$ are far away within a partially-known workspace, the resulting trajectory may

suffer from oscillation or jitters and long detours. To tackle these issues, 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:

$$\mathcal{T}_{\widehat{g}_\ell \to \widehat{g}_{\ell+1}} \triangleq \big(V,\, B,\, \gamma,\, (\nu_0,\, \nu_G)\big), \tag{6}$$

where $V \subset \mathcal{W} \times (-\pi,\, \pi]$ is the set of vertices; $B \subset V \times V$ is the set of edges; $\gamma : B \to \mathbb{R}_{\geqslant 0}$ returns the edge cost to be estimated; $\nu_0 = \widehat{g}_\ell$ and $\nu_G = \widehat{g}_{\ell+1}$ are the initial and target poses. The goal is to find a sequence of vertices in $\mathcal{T}$ as the path from ν_0 to ν_G. Initially, $V = \{\nu_0,\, \nu_G\}$ and $B = \varnothing$. Then, the set of vertices can be generated in various ways, e.g., via direct discretization including uniform, triangulation and voronoi partitions, see e.g., LaValle (2006); or sampling-based methods such as PRM from Hsu et al. (2006) and RRT from Karaman & Frazzoli (2011). In other words, the oriented HT is not limited to one particular choice of abstraction method. 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, i.e., $(\nu,\, \nu') \in B$ if $\|\nu - \nu'\| < \varepsilon$ holds for a radius $\varepsilon > 0$. For each edge $(\nu,\, \nu') \in E$, the control cost $\gamma(\nu,\, \nu')$ for the robot (3) to navigate from ν to ν' is derived in the sequel. Thus, given the weighted and directed tree $\mathcal{T}_{\widehat{g}_\ell \to \widehat{g}_{\ell+1}}$, the shortest path from ν_0 to ν_G can be determined by any search algorithm from LaValle (2006), e.g., Dijkstra and A* (with $\|\nu - \nu_G\|$ being the "cost-to-go" heuristic). The resulting path is denoted by:

$$\mathbf{P}_{\widehat{g}_\ell \to \widehat{g}_{\ell+1}} \triangleq \nu_0 \nu_1 \cdots \nu_{N-1} \nu_G, \tag{7}$$

where $\nu_n \in V$ and $(\nu_n,\, \nu_{n+1}) \in B$, $\forall n \in [0,\, N-1]$. 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 (4), 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 slope between consecutive points. The exact thresholds would depend on the specification of the Lidar scanner. Then, each cluster is fitted to a particular 2D obstacle. This work primarily focuses on a special type of obstacle called squircle. Squircles are particularly useful for representing walls and corners, see Li & 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:

$$\beta_{\text{sc}}(q) \triangleq \frac{q^2 + \sqrt{q^4 - 4\varkappa^2 \left[(q^\intercal\, e_1)(q^\intercal\, e_2)\right]^2}}{2} - 1, \tag{8}$$

where $\varkappa \in (0,\, 1)$ is a positive parameter; $e_1 \triangleq [1, 0]^\intercal$ and $e_2 \triangleq [0, 1]^\intercal$ are two unit basis in $\mathbb{R}^2$. Non-unit squircles with general

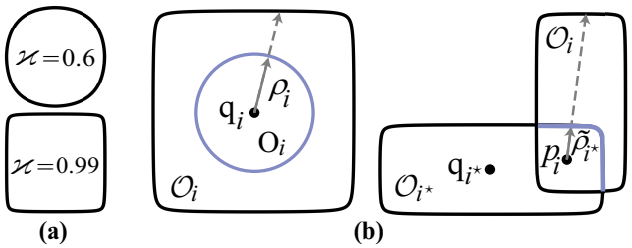

Figure 4: (a) Squircles with the parameter $\varkappa = 0.6$ and $\varkappa = 0.99$; (b) Ray scaling process. The boundary of the star-shaped obstacle is mapped onto the boundary of a sphere (**Left**). The boundary of the child obstacle is mapped onto a segment of the boundary of the parent obstacle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

centers can be derived via scaling $\mathbf{A}$ and translation $\mathbf{T}$, i.e., $\mathbf{A} \triangleq \text{diag}(\mathbf{s})$ and $\mathbf{T} \triangleq \mathbf{c}^\intercal$, where $\mathbf{s} \triangleq (a^1,\, a^2)$ represents the width and height of the squircle; $\mathbf{c} \triangleq (c^1,\, c^2)$ is the geometric center of the squircle. Given the clusters of the observed point clouds, the model of obstacles can be reconstructed based on the above model of squircle, as illustrated in Fig. 3.

Lemma 1. *Given a cluster of point cloud $\mathbf{D}_{t,k} = \{q_1, q_2, \cdots, q_N\}$ associated with the same squircle obstacle, if $N \geqslant 4$ and $q_i \neq q_j$, $i \neq j$, $\forall i, j \in \{1, \cdots, N\}$, then this obstacle can be uniquely reconstructed as $\hat{\mathcal{O}}_k \triangleq (\mathbf{s}_k,\, \mathbf{c}_k)$, i.e., the scaling $\mathbf{A}_k$ and translation $\mathbf{T}_k$ can be uniquely determined.*

Proof. Since the cluster of points lay on the surface of the squircle obstacle, each point $q_i \in \mathbf{D}_{t,k}$ satisfies: $\beta_{\text{sc}}(\hat{q}_i) = 0$ holds for $\hat{q}_i = \mathbf{A}_k^{-1}(q_i - \mathbf{c}_k)$. It can be further derived as: $[a_k^2(x_i - c_k^1)]^2 + [a_k^1(y_i - c_k^2)]^2 - (a_k^1\, a_k^2)^2 - \varkappa^2[(x_i - c_k^1)(y_i - c_k^2)]^2 = 0$, $\forall i = 1, \cdots, N$. The above N equations include four solving variables $a_k^1, a_k^2, c_k^1, c_k^2$. By the Gröbner bases, any four of these equations yield a unique solution, denoted by $(\hat{a}_k^1, \hat{a}_k^2, \hat{c}_k^1, \hat{c}_k^2) = (\hat{\mathbf{s}}_k,\, \hat{\mathbf{c}}_k)$. Then, the obstacle is reconstructed by (8). □

Remark 1. In practice, the point-cloud data might be noisy and not align with a squircle perfectly. Higher accuracy can be achieved by using more data points by the optimization method like Levenberg-Marquardt algorithm from Gavin (2019). ■

Consequently, denote by $\{\hat{\mathcal{O}}_t\}$ the collection of obstacles detected and fitted at time $t \geqslant 0$. 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 $\mathcal{O}_0 = \{q \in \mathbb{R}^2 | \beta_0(q) \leqslant 0\}$ and M inner squircle obstacles $\mathcal{O}_i = \{q \in \mathbb{R}^2 | \beta_i(q) \leqslant 0\}$,

where $\beta_i(q)$ is the obstacle function defined in (8), for $i = 0, 1, \cdots, M$. 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 $i = 0, 1, \cdots, M$. Consequently, the transformation from star world $\mathcal{S}$ to its model sphere world $\mathcal{M}$ is given by:

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

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; $\mathrm{id}(q)$ 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 Loizou & Rimon (2022); Li & Tanner (2018); 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 can not 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 $\mathcal{F} = \mathcal{W}_0 \backslash \bigcup_{n=1}^{N} \mathcal{T}_n$, where $\mathcal{T}_n$ is the n-th 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. 5, 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 $\mathcal{O}_i \triangleq \{q \in \mathbb{R}^2 : \beta_i(q) \leqslant 0\}$ the leaf obstacles, where $\beta_i(q)$ is the obstacle function defined in (8). 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 n-th tree of stars, where $n = 1, \cdots, N$. 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 i-th leaf obstacle in the n-th tree for $i = 1, \cdots, d_n$, 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 (8); 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 $\ell = 1, 2$.

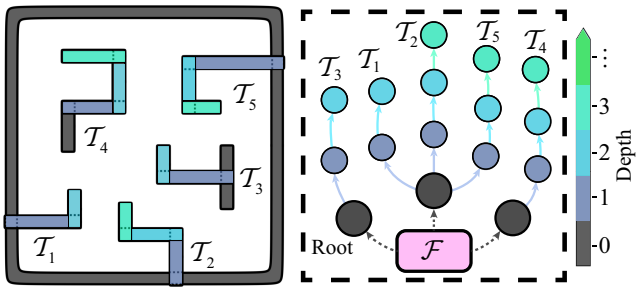

Figure 5: A forest world $\mathcal{F}$ with overlapping squircles (**Left**), which consists of 5 trees of stars $\mathcal{T}_i$ with different depths from the root to the leaves (**Right**).

Lemma 2. *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 (8). 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. □

Remark 2. The modified length of rays $\tilde{\rho}_{i^\star}(q)$ in (15) is essential since the common center p_i is not always aligned with the geometric center $q_{i^\star}$ of the parent obstacle. A similar technique is introduced in Li & Tanner (2018) via a virtual obstacle between the parent and son obstacles. In contrast, the piecewise scaling matrix proposed here in (15) is less conservative and computationally more efficient. ■

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: $\tilde{\kappa}_i(q) \triangleq \beta_{i^\star}(q) + (\beta_i(q) - 2E_i) + \sqrt{\beta^2_{i^\star}(q) + ((\beta_i(q) - 2E_i))^2}$. The parameter $E_i > 0$ is a geometric constant satisfying that $\mathcal{O}_i(2E_i) \bigcap \mathcal{O}_j(2E_j) = \varnothing$ and $\gamma^{-1}([0, 2E_d]) \bigcap \mathcal{O}_i(2E_i) = \varnothing$ with $E_d > 0$ being a geometric constant and $\mathcal{O}_i(2E_i) \triangleq \{q \in \mathbb{R}^2 | \beta_i(q) \leqslant 2E_i\}$ for $i,\, j \in \mathcal{I}$ with $i \neq j$ and $i, j \neq j^\star$. 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 i-th leaf obstacle $\mathcal{O}_i$ in the n-th 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:

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

where $\tilde{\beta}_i(q)$ with a geometric constant $E_i > 0$ is defined by: $\tilde{\beta}_i(q) \triangleq \beta_{i^\star}(q) + (2E_i - \beta_i(q)) + \sqrt{\beta^2_{i^\star}(q) + (2E_i - \beta_i(q))^2}$.

Remark 3. 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 & 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*. ■

Remark 4. Recently, a novel conformal transformation was introduced in Fan et al. (2022) to map the complex workspace to a sphere world without the decomposition of tree of stars. However, this approach requires a numerical solution of continuous integral equations, which is time-consuming and not applicable to unknown environment. Conversely, the proposed method offers an analytical solution, making it superior for navigation in unknown scenarios. ■

Lemma 3. *Given a workspace $\mathcal{W}$ containing N tree-of-stars with different depth $d_n \geqslant 0$, for $n = 1, \cdots N$, the complete harmonic potential function $\varphi_{\text{NF}}(q)$ in* (9) *is a valid navigation function for the workspace $\mathcal{W}$, if the parameters $K > N$ and $\mu \geqslant 1$ hold in* (2)*; $\lambda > \Lambda$ for a positive number $\Lambda > 0$; and $\xi_i > \Xi_i$ for some positive numbers $\Xi_i > 0$, for $i = 0, \cdots, d_n$.*

Proof. Similar statements have been proven in Rousseas et al. (2021); Filippidis & Kyriakopoulos (2011); Loizou & Rimon (2022), thus detailed proofs are omitted here and refer the readers to e.g., Theorem 1 of Loizou & Rimon (2022), Theorem 2 in Li & 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 $\varphi_{\text{NF}}(q)$ 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 $\mu \geqslant 1$ 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 $\tilde{\rho}_{i^\star}(q)$ is proven to be smooth in Lemma 2, the Jacobian of $f_{n,i}$ exists and is given by:

$$\begin{aligned} J_{f_{n,i}} =& \sigma_i(q - p_i)\nabla v_i^\intercal + (v_i - 1)(q - p_i)\nabla\sigma_i^\intercal \\ &+ (1 - \sigma_i)\mathbf{I} + \sigma_i v_i \mathbf{I}. \end{aligned}$$

Similar to Lemmas 7 and 8 in Li & 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 $\mathcal{O}_i(\epsilon_i) = \{q \in \mathbb{R}^2 | \beta_i \leqslant \epsilon_i\}$, and the set away from $\mathcal{O}_i$, denoted by $\mathcal{A}(\epsilon_i) = \{q \in \mathbb{R}^2 | \beta_i \geqslant \epsilon_i\}$, 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 $f_{n,i}(q)|_{q\in\partial\mathcal{O}_i} = \frac{\tilde{\rho}_{i^\star}(q)}{\|q-p_i\|}(q - p_i) + p_i$. Assume that there exists 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 Rousseas et al. (2021, 2022b); Filippidis & Kyriakopoulos (2011); Loizou & Rimon (2022). However, the final orientation of the robot at the destination *can not* be controlled, yielding it impractical for the oriented path $\mathbf{P}_{\hat{g}_\ell\to\hat{g}_{\ell+1}}$ in (7). Thus, an additional two-step rotation is proposed for the potential fields above.

Two-step Rotation to Harmonic Potentials: As shown in Fig. 6, 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, see e.g., LaValle (2006) associated with angle $\theta \in [0, -\pi)$, i.e.,

$$\mathbf{R}(\theta) \triangleq \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix};$$

the variable $\theta_1(q)$ is the angle to be rotated such that the direction of the transformed potential fields 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 $\delta_\theta(q) \triangleq \theta_{q-q_G} - \theta_{\nabla\varphi_{\text{NF}}(q)}$ is the relative angle between $q - q_G$ and $\nabla\varphi_{\text{NF}}(q)$; $\delta_c \triangleq 2\pi$; $s_d(q) \triangleq \exp\left(\tau - \frac{\tau\mu^2}{(\mu-\varphi_{\text{NF}}(q))^2}\right)$ 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 $\varphi_{\text{NF}}(q)$. 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).

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:

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

where $\nabla\varphi_{\text{NF}}(q)$ 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 4. *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 3 hold.*

Proof. To begin with, it is proven in Lemma 3 that $\varphi_{\text{NF}}(q)$ is a valid navigation function, it holds that $\varphi_{\text{NF}}(q) = \mu$ 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., $-\nabla\varphi_{\text{NF}}(q)$. 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 $\varphi_{\text{NF}}(q)$, including the global minimum q_G and saddle points, retain their properties after the transformation. In other words, the saddle points of $\varphi_{\text{NF}}(q)$ 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 & 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.

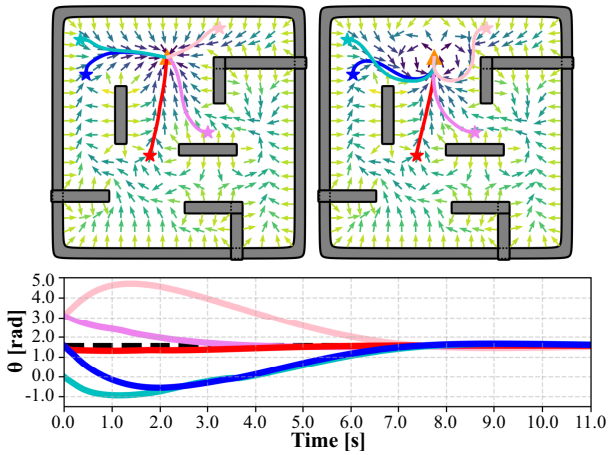

Figure 6: Robot trajectories from different initial poses (filled stars) to the same goal pose (orange triangle), under the original potential fields $-\nabla\varphi_{\text{NF}}(q)$ (**Top Left**) and the *oriented* potential fields $\Upsilon(q)$ (**Top Right**) via the proposed control law in (24). **Bottom**: The robot orientation θ converges to the desired value (dashed line), along these trajectories.

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

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

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: $\langle \Upsilon(q),\ q_G - q \rangle = -\|\nabla\varphi_{\text{NF}}(q)\|\|q_G - q\| \cos\left(\delta'_\theta(q)\right)$. 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 $\theta_{q_G - q}$ approaches θ_G. Thus, the integral curves converge to q_G along the desired orientation θ_G. □

Remark 5. Many existing navigation functions in Rousseas et al. (2021, 2022b); Filippidis & Kyriakopoulos (2011); Loizou & Rimon (2022, 2021) are globally convergent, but do not take into account the desired orientation at the goal point. Similar idea of rotating the gradients is adopted in Valbuena & Tanner (2012), but it requires that the goal orientation is aligned with the positive direction of x-axis, thus yielding a more restrictive approach than the two-step rotation in (20). ■

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:

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

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

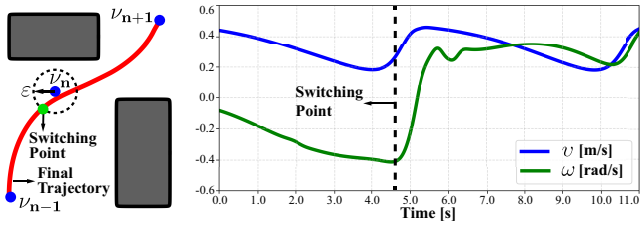

Figure 7: Illustration of the smooth transition strategy in (25) when the agent switches from one intermediate waypoint to another. The switch function η_ϵ is activated once the current waypoint is reached with a margin ϵ. The resulting trajectory and control inputs are all smooth.

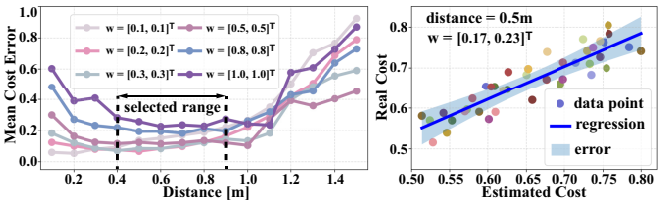

Figure 8: **Left:** The difference between the estimated cost by (26) and the measured cost under different weights $\mathbf{w}$, for different choice of the distance $d(q_n,\, q_{n+1})$. **Right:** The difference actual with 40 samples, when $\mathbf{w} = [0.17,\, 0.23]^\intercal$ and $d = 0.5$.

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).

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 3 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 3 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 $\mathbf{P}_{\hat{g}_\ell\to\hat{g}_{\ell+1}}$ from (7), a smooth transition between the intermediate waypoints is crucial to ensure both smooth trajectories and control inputs. As shown in Fig. 7, the switching strategy from $\nu_n = (q_n, \theta_n)$ to $\nu_{n+1} = (q_{n+1}, \theta_{n+1})$ in $\mathbf{P}_{\hat{g}_\ell\to\hat{g}_{\ell+1}}$ is given by:

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

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. 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:

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

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 $\mathcal{T}_{\hat{g}_\ell\to\hat{g}_{\ell+1}}$ in (6), yielding a more accurate estimate of the navigation cost from $\hat{g}_\ell$ to $\hat{g}_{\ell+1}$. In turn, the actual cost within the navigation map $\mathcal{G}$ is given by:

$$d(\hat{g},\, \hat{g}') = \sum_{n=0}^{N-1} \gamma(\nu_n,\, \nu_{n+1}), \quad (27)$$

where $\mathbf{P}_{\hat{g}\to\hat{g}'} = \nu_0\nu_1\cdots\nu_N$ is the shortest path from $\hat{g}$ to $\hat{g}'$ in the associated HT $\mathcal{T}_{\hat{g}\to\hat{g}'}$.

Remark 6. The control cost in (26) is estimated through a weighted combination of the translation cost and the steering cost. As shown in Fig. 8, 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 $\hat{\mathbf{g}} = \hat{g}_1\hat{g}_2\cdots\hat{g}_L(\hat{g}_{L+1}\hat{g}_{L+2}\cdots\hat{g}_{L+H})^\omega$ in (5) can be executed via the following hybrid strategy: (i) staring from $\ell = 1$, determine the next goal region $\hat{g}_{\ell+1}$; (ii) construct the HT as $\mathcal{T}_{\hat{g}_\ell\to\hat{g}_{\ell+1}}$ and determine the shortest path $\mathbf{P}_{\hat{g}_\ell\to\hat{g}_{\ell+1}} = \nu_0\cdots\nu_{N-1}\nu_G$ in (7); (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., $\hat{g}_{\ell+1}$, 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 7. The design of intermediate waypoints between subtasks can alleviate certain limitations of the classic navigation functions from Koditschek (1987); Rimon (1990) for robotic navigation. As shown in Fig. 9, when the robotic system is geometrically and dynamically constrained, it can *not* follow

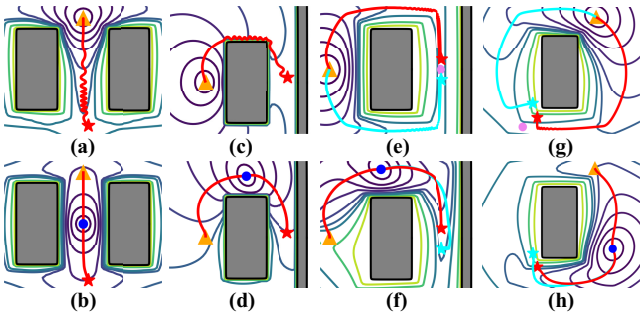

Figure 9: Illustration of how intermediate waypoints can reduce oscillations and long detours, under different initial poses (solid stars) and goal poses (orange triangles). (I) Oscillations appear as the robot traverses narrow passage in **(a)** or near the obstacle boundary in **(c)**. Intermediate 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.

the negated gradient perfectly. Consequently, oscillations appear along narrow passages and close to boundary of obstacles, where the underlying gradient changes directions abruptly. As also emphasized in Rimon (1990); Loizou & Rimon (2022), 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, both the HTs and the underlying harmonic potentials should be updated. Total re-computation of both parts whenever a new obstacle is detected would require significant computational resources, yielding it unsuitable for online real-time applications. Instead, recursive algorithms are proposed here to incrementally update the harmonic potentials and tree structures.

4.2.1. Incremental Update of Harmonic Potentials

The classic methods as proposed in Loizou & Rimon (2022); Li & Tanner (2018); Rimon (1990) construct the underlying navigation function at once by taking into account *all* 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. 10: (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 $\mathcal{O}_k$, $k = 1, \cdots, M$, is added to an empty workspace $\mathcal{W} = \mathcal{O}_0$ in sequence. As discussed in Sec. 4.1.2, the star-to-sphere transformation for each obstacle is constructed via the translated

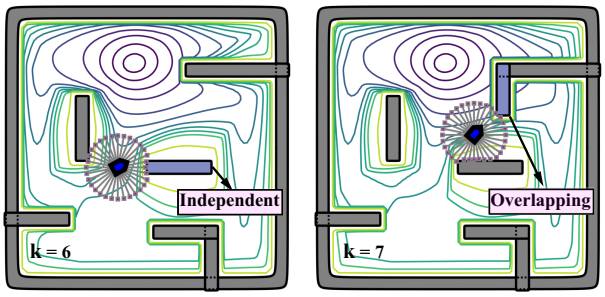

Figure 10: Iterative update of the harmonic potentials as new obstacles are detected and added into the workspace. An independent obstacle is added for $k = 6$ (**Left**), while an overlapping obstacle is added for $k = 7$ (**Right**).

scaling map in (11). Specifically, the transformation from star world to sphere world after adding the k-th star is restated as:

$$\Phi^k_{\mathcal{S}\to\mathcal{M}}(q) \triangleq \mathrm{id}(q) + \sum_{i=0}^{k} s^k_i(q)\,\big(v_i(q)-1\big)\,(q-q_i), \quad (28)$$

where $s^k_i(q)$ 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 $i = 0, 1, \cdots, k$.

Definition 4. The online omitted product $\overline{\beta}^k_i(q)$ of star-to-sphere transformation for obstacle $\mathcal{O}_i$ after adding the k-th obstacle, for $i = 0, 1, \cdots, k$ and for $k = 1, \cdots, M$, is a real-valued function defined for the star world $\mathcal{S}$ as: $\overline{\beta}^k_i(q) \triangleq \prod^k_{j=0, j\neq i} \beta_j(q)$, where $\beta_j(q)$ is the obstacle function defined in (8). ■

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

$$s^k_i(q) \triangleq \frac{\gamma_G(q)\overline{\beta}^k_i(q)}{\lambda_k\beta_i(q) + \gamma_G(q)\overline{\beta}^k_i(q)},$$

where $\gamma_G(q)$ is the distance-to-goal function; λ_k is a positive parameter associated with k; $\overline{\beta}^k_i(q)$ is the online omitted product; and $\beta_i(q)$ is the obstacle function defined in (8). ■

Then, the following intermediate variables are defined:

$$\mathbf{F}^k_i(q) \triangleq s^k_i(q)\,\mathbf{r}_i(q), \quad (29)$$

where $\mathbf{r}_i \triangleq \big(v_i(q)-1\big)(q-q_i)$. Consequently, the transformation in (28) is adjusted to: $\Phi^k_{\mathcal{S}\to\mathcal{M}}(q) = \mathrm{id}(q) + \sum^k_{i=0} \mathbf{F}^k_i(q)$. 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, $\mathbf{F}^k_i(q)$ should be updated to $\mathbf{F}^{k+1}_i(q)$, $\forall i = 0, 1, \cdots, k$, 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:

$$\Psi_{\mathrm{a}}(\mathbf{F}, \mathbf{r}, \tilde{\mathbf{r}}, \alpha) \triangleq \frac{\|\mathbf{F}\|}{\alpha\,\|\mathbf{r}\| + (1-\alpha)\,\|\mathbf{F}\|}\,\tilde{\mathbf{r}}, \quad (30)$$

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:

$$\mathbf{F}_0^{k+1}(q) = \Psi_{\mathsf{a}} \left(\mathbf{F}_0^k(q),\, \mathbf{r}_0(q),\, \mathbf{r}_0(q),\, \alpha^k(q) \right), \tag{31}$$

where $\mathbf{r}_0(q)$ is defined in (29) for the workspace; the variable $\alpha^k(q) \triangleq \lambda_{k+1} / \big(\lambda_k\, \beta_{k+1}(q)\big)$; λ_k and λ_{k+1} are positive parameters defined in (5), for $k = 0, 1, \cdots, M$.

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 $\mathcal{O}_0$ 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 $\alpha^k(q) = \lambda_{k+1} / \big(\lambda_k\, \beta_{k+1}(q)\big)$.

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 (31) for $\mathcal{O}_0$ is valid for iteration $k = 0, 1, \cdots, M-1$.*

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 (29). 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 $\ell \geqslant 0$. Then, $\mathbf{F}_0^{\ell+1}(q)$ is derived via (31) as:

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

Given (8), it holds that $\beta_0(q) \geqslant 0$ and $\overline{\beta}_0^\ell(q) \geqslant 0$, 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 5 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 (29), thus concluding the proof. □

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

$$\mathbf{F}_{i+1}^{k+1}(q) = \Psi_{\mathsf{a}} \left(\mathbf{F}_i^{k+1}(q),\, \mathbf{r}_i(q),\, \mathbf{r}_{i+1}(q),\, \alpha_i(q) \right), \tag{32}$$

where $\mathbf{r}_i(q)$ and $\mathbf{r}_{i+1}(q)$ are defined in (29) for the i-th and $(i+1)$-th obstacles; $\alpha_i(q) = (\beta_{i+1}(q))^2 / (\beta_i(q))^2$.

Lemma 6. *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 (31) for $\mathcal{O}_{i+1}$ is valid for iteration $i = 0, 1, \cdots, k$.*

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 (29). 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 $\ell \geqslant 0$. Then, $\mathbf{F}_{\ell+1}^{k+1}(q)$ is derived via (31) 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 6 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 (29), thus concluding the proof. □

Remark 8. Note that $\mathbf{F}_0^{k+1}(q)$ in (31) is iterated over the index k with initial value $\mathbf{F}_0^0(q)$ while $\mathbf{F}_{i+1}^{k+1}(q)$ in (32) 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 $i = 0, 1, \cdots, k$ 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 $\{\mathcal{O}_k, k = 1, \cdots, N\}$, is added to a star world and overlapping with one of the existing obstacles as a leaf. As discussed in Sec. 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 k-th obstacle in one of the tree is restated as:

$$f_k(q) \triangleq \mathrm{id}(q) + \sigma_k(q) \big(v_k(q) - 1\big)\, (q - p_k), \tag{33}$$

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 k-th obstacle for $k = 1, \cdots, N$, is defined on the forest world $\mathcal{F}$ as:

$$\overline{\beta}_k(q) \triangleq \Big(\prod_{j=0, j\neq k^\star}^{k-1} \beta_j(q) \Big) \Big(\prod_{j \in \mathcal{L} \setminus \{k\}} \beta_j(q) \Big) \tilde{\beta}_k(q), \tag{34}$$

where $\tilde{\beta}_k(q)$ is defined in (19); $\beta_j(q)$ is the obstacle function defined in (8); 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 k-th obstacle for $k = 1, \cdots, N$, is defined on the forest world $\mathcal{F}$ as:

$$\sigma_k(q) \triangleq \frac{\gamma_G(q) \overline{\beta}_k(q)}{\xi_k \beta_k(q) + \gamma_G(q) \overline{\beta}_k(q)}, \tag{35}$$

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 (29), the intermediate variables are defined as:

$$\mathbf{H}_k(q) \triangleq \sigma_k(q)\,\mathbf{r}_k(q), \tag{36}$$

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 k-th star obstacle is given by:

$$\Phi^k_{\mathcal{F}\to\mathcal{S}}(q) \triangleq \mathbf{H}_1 \circ \mathbf{H} \circ \cdots \circ \mathbf{H}_k(q). \tag{37}$$

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 (37) should be updated as follows:

$$\Phi^{k+1}_{\mathcal{F}\to\mathcal{S}}(q) = \Phi^k_{\mathcal{F}\to\mathcal{S}} \circ \mathbf{H}_{k+1}(q), \tag{38}$$

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

$$\mathbf{H}_{k+1}(q) = \Psi_{\mathsf{a}}\left(\mathbf{H}_k(q),\, \mathbf{r}_k(q),\, \mathbf{r}_{k+1}(q),\, \alpha_k(q)\right), \tag{39}$$

where $\Psi_{\mathsf{a}}(\cdot)$ is defined in (30); $\mathbf{r}_{k+1}(q)$ and $\mathbf{r}_k(q)$ are defined in (36); 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 7. *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* (39) *is valid, for each new leaf obstacle $k = 0, 1, \cdots, N-1$.*

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 (36). 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 $\ell \geqslant 0$. Then, $\mathbf{H}_{\ell+1}(q)$ is computed via (39) 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 7, $\mathbf{H}_{\ell+1}(q) = \sigma_{\ell+1}(q)\,\mathbf{r}_{\ell+1}(q)$ holds, which is consistent with (36), thus concluding the proof. □

Remark 9. Note that the *same* auxiliary function $\Psi(\cdot)$ in (30) is used for the recursive computation in (31), (32) and (39). ■

Remark 10. The parameters λ_k in Def. 5 for $k = 0, 1, \cdots, M$, and ξ_k in Def. 7 for $k = 1, 2, \cdots, N$ should satisfy the conditions in Lemma 3, 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 k-th obstacle $\mathcal{O}_k$ is given by:

$$\Phi^k_{\mathcal{F}\to\mathcal{M}}(q) \triangleq \Phi^k_{\mathcal{S}\to\mathcal{M}} \circ \Phi^k_{\mathcal{F}\to\mathcal{S}}(q), \tag{40}$$

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 (28); 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 (38). 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 k-th 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 $\varphi^{k+1}_{\text{NF}}$ accordingly.

4.2.2. Local Revision of Harmonic Trees

As new obstacles are detected, parts of some Oriented Harmonic Trees $\{\mathcal{T}_{\hat{g}_\ell\to\hat{g}_{\ell+1}}\}$ constructed in Sec. 4.1.1 might become invalid as the existing vertices might fall inside the new obstacle, and more importantly, the cost associated with certain edges should be recomputed as the underlying harmonic potentials are updated in Sec. 4.2.1. Instead of total reconstruction, this section describes how the Harmonic Trees and the resulting paths are revised incrementally as new obstacles are detected.

The revision procedure consists of *three* main steps: (I) *Trimming of Harmonic trees*. For each HT $\mathcal{T}_{\hat{g}_\ell\to\hat{g}_{\ell+1}}$, find the vertices that are within the newly-added obstacle, i.e., $\nu \in \mathcal{O}_k$, and remove them from the set of vertices V, along with the attached edges. As a result, the remaining tree might become disconnected and thus a path from its current vertex the goal might no longer exist; (II) *Potential-biased regeneration*. To re-connect the oriented HT to the goal vertex, new vertices and edges are generated. A process similar to the initial construction in Sec. 4.1.1 could be followed. However, depending on how the workspace has changed, it could be more efficient to focus the growth towards the obstacle areas that has been modified. Depending on the particular method of discretization, new vertices could be generated with the bias: $b(q) \propto |\zeta^{k+1}(q) - \zeta^k(q)|$, where $b(q)$ is the probability of generating a new vertex at q; $\zeta^k(q) \triangleq \exp(1 - \frac{1}{\varphi^k_{\text{NF}}(q)})$ and similarly for $\zeta^{k+1}(q)$; the right-hand side measures the relative change of the potential after adding obstacle $\mathcal{O}_{k+1}$. In other words, the harmonic tree are

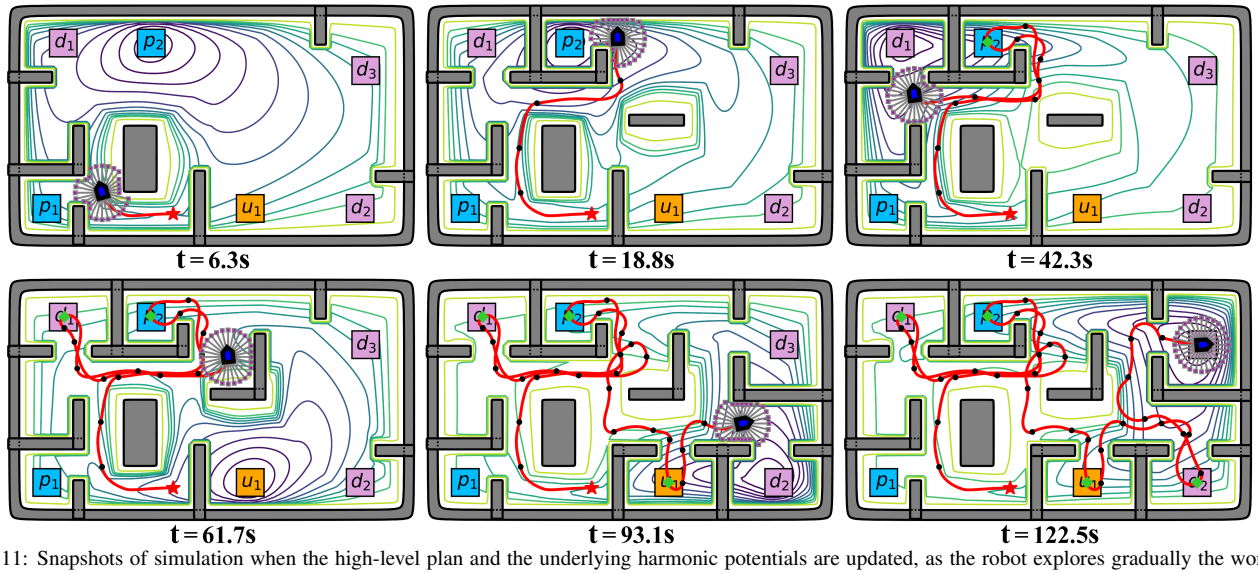

Figure 11: Snapshots of simulation when the high-level plan and the underlying harmonic potentials are updated, as the robot explores gradually the workspace. The robot trajectory is depicted in red; the task nodes in $\mathcal{G}$ are highlighted in green; and the intermediate waypoints in $\mathcal{T}$ are marked in black.

re-grown 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 by (27). Since numerous edges have been traversed during execution, their actual costs have been measured and saved. Consequently, the estimation model in (26) can be updated to approximate the transition cost more accurately. More specifically, suppose that at each time of update H edges have been traversed in total, of which their actual costs are given by $\boldsymbol{\gamma}^\star = (\gamma_1^\star, \cdots, \gamma_H^\star)$. Their estimated costs are $\bar{\gamma} = (\gamma_1, \cdots, \gamma_H)$ with $\bar{\mathbf{d}}$ and $\bar{\mathbf{Q}}$ being the distance and rotation cost respectively 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.

4.2.3. 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 $\hat{g}$ to $\hat{g}'$ in $\mathcal{G}$ is re-evaluated based on whether the path $\mathbf{P}_{\hat{g}\to\hat{g}'}$ exists within the HT $\mathcal{T}_{\hat{g}\to\hat{g}'}$. 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 by (27) 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 $\hat{\mathbf{g}}^\star$ is re-synthesized within the updated product $\hat{\mathcal{A}}$ 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. Summary and Complexity Analyses

To summarize, during the initial synthesis, the navigation map $\mathcal{G}$ is constructed hierarchically by building first the HTs $\{\mathcal{T}_{\hat{g}\to\hat{g}'}\}$ associated with each transition of $\mathcal{G}$, and then the oriented Harmonic potentials $\{\Upsilon_{\nu\to\nu'}\}$ associated with each edge of $\mathcal{T}_{\hat{g}\to\hat{g}'}$. Given $\mathcal{G}$, the initial plan $\hat{\mathbf{g}}$ is derived after computing the product automaton $\hat{\mathcal{A}}$. Afterwards, the plan $\hat{\mathbf{g}}$ is executed also hierarchically by tracing first the sequence of transitions and then the sequence of edges associated with each transition. Meanwhile, as the robot explores more obstacles, the harmonic potentials $\{\Upsilon\}$ are updated recursively, based on which the HTs $\{\mathcal{T}_{\hat{g}\to\hat{g}'}\}$ and navigation map $\mathcal{G}$ are revised locally. Consequently, the task plan $\hat{\mathbf{g}}$ is adapted given the current robot pose and the updated product automaton $\hat{\mathcal{A}}$. This procedure continues indefinitely until termination.

The computational complexity to construct the product $\hat{\mathcal{A}}$ 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 and experiment videos can be found in Wang (2023).

Table 1: Comparison With State-of-the-art Methods

Method	Planning Time [s]	Execution Time [s]	Travel Distance [m]	Adaptation	Oscillation
HT(ours)	3.29	55.83	15.54	**Yes**	**No**
NF	—	83.30	20.11	No	Yes
OHPF	**1.36**	74.74	18.03	**Yes**	Yes
RRT*	83.74	**45.63**	19.81	No	—
HM	12.28	49.15	**14.32**	No	**No**
CNT	26.94	87.31	18.89	No	**No**

5.1. Description of Robot and Task

Consider a unicycle robot with the dynamics in (3) that operates within a workspace measuring $7m \times 4m$, with an initial model depicted in Fig. 3. The robot occupies a rectangular area of size $0.2m \times 0.3m$, and is equipped with a Lidar sensor capable of detecting objects up to a maximum range of $\mathcal{L}_{\max} = 0.5m$. The control gains for the robot's movement, as specified in (24), are set to $k_\upsilon = 1.0$ and $k_\omega = 0.8$. Both the robot control and the Lidar measurements are updated at 10Hz. 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_d are both set to 0.02. The parameters λ_k, ξ_k in Def. 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 (6) is generated by RRT* with cost estimation based on (26) and an expansion radius of $0.5m$. Fig. 11 illustrates the complete environment, which mimics a complex office setting with numerous overlapping obstacles. Initially, the robot has the prior knowledge of the boundaries of the workspace, the task regions, and some inner obstacles as shown in Fig. 3.

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 $\varphi = \big(\Diamond\big((p_1 \vee p_2) \wedge (\Diamond d_1)\big) \wedge \big(\Diamond\big((p_1 \vee p_2) \wedge (\Diamond d_2)\big) \wedge \big(\Diamond\big((p_1 \vee p_2) \wedge (\Diamond d_3)\big) \wedge (\circ \rightarrow u_5)$. 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

At time $t = 0$, the robot starts from the initial position $(2.8m, 0.4m)$ and 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 an average of 9 intermediate waypoints, with an average computation time of $0.103s$. The weight of each edge is estimated using the cost function in Sec. 4.1.1. Then, the product automaton $\widehat{\mathcal{A}}$ is constructed with 192 states and 1435 edges, of which the initial task plan is $P_0 = p_1\, d_1\, d_3\, d_2$. Guided by this high-level plan, the robot navigates first towards task p_1 along the intermediate waypoints. Between any pair of the waypoint $(\nu_n,\, \nu_{n+1})$, the initial navigation function $\varphi_{\text{NF}}(\text{q})$ is constructed using the associated transformations in (9). The mean computation time for $\varphi_{\text{NF}}(q)$ at every point within the workspace is $9.17ms$ and the corresponding potentials are visualized in Fig. 3 and 11. Given this navigation function, the oriented harmonic fields $\Upsilon(q)$ is constructed, with the associated orientation θ_{n+1}, then the nonlinear controller in (24) is utilized to drive the robot towards the current goal as illustrated in Fig. 11. As the robot executes its current plan and explores the workspace gradually, new obstacles are discovered at discrete time instants, as shown in Fig. 11. For example, at $t = 6.3s$ and $18.8s$, new obstacles are detected and block the way; the underlying navigation functions are updated incrementally by the approach described in Sec. 4.2.1. In detail, at $t = 6.3s$, the estimated costs for the plans $p_1d_1d_2d_3$ and $p_2d_1d_2d_3$ are 17.1 and 15.9, 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 are $13.94ms$ and $16.63ms$, respectively. The weight $\mathbf{w}$ in the control cost is updated by the strategy in Sec. 4.2.3 with the traversed edges in HT. Afterwards, the navigation map $\mathcal{G}$ is updated accordingly to satisfy the feasibility and cost minimization. The new task plan is $p_2d_1d_3d_2$. Given this new plan, the robot continues to navigate along the associated potential fields. At $t = 61.7s$, 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. 11. Finally, the whole task is accomplished in $122.5s$ and the resulting trajectory has a total distance $30.86m$.

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 Navigation Function (NF) from Rimon (1990); Loizou (2017) without the high-level task adaptation; (ii) the oriented harmonic potential fields (OHPF) proposed in this work with the high-level task adaptation, but omitting the second-layer search trees; (iii) the Non-holonomic RRT* in LaValle (2006); Park & Kuipers (2015) without the high-level task adaptation; (iv) the harmonic maps (HM) in Vlantis et al. (2018) via the open-sourced implementation; 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 travelled, and whether oscillations appear during execution.

For easier comparison, a smaller workspace of size $4m \times 4m$ with 11 overlapping obstacles is considered as shown in

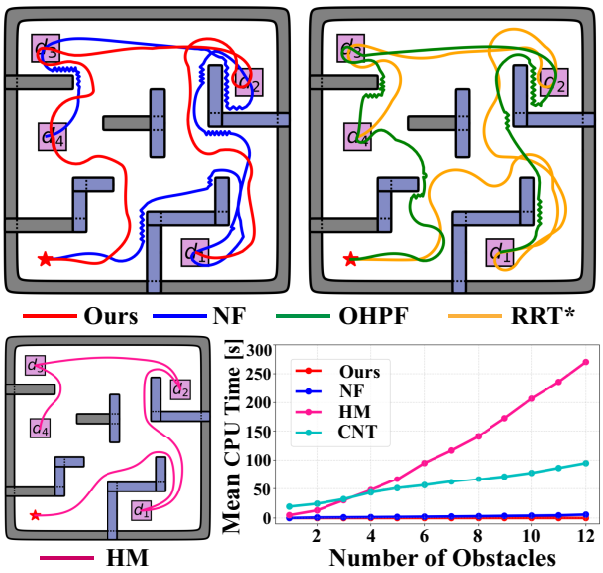

Figure 12: **Top and Bottom Left:** Comparison of the final trajectories via the proposed method (in red), NF without plan adaptation (in blue), OHPF without the search trees (in green), RRT* without plan adaptation (in orange), and HM without plan adaptation (in pink). The obstacles in grey are initially known and in blue are detected online. **Bottom Right:** Comparison of computation time between our incremental method (in red), and the baselines.

Fig. 12. The robot starts from $(0.5m, 0.4m)$ with a prior knowledge of 3 obstacles. 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 Sec. 5.1. The final trajectories and numerical results are shown in Fig. 12 and Table 1. The proposed HT exhibits lower 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 obstalce can not be tracked perfectly. Besides, the nominal NF can not 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 $83s$ to compute the complete plan for RRT*, compared with merely $3.3s$ by the proposed HT. The HM method takes the shortest distance of $14.32m$ to finish the task with a longer planning time $12s$ due to the off-line construction of the navigation map with 2048 boundary points. Similarly, the CNT method takes significantly more planning time and execution time (close to $26s$ and $87s$).

Lastly, the computational efficiency of the proposed iterative approach is compared with nominal NF, HM, and CNT, as summarized in Fig. 12. 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 HM and CNT increases drastically with the number of boundary points (each obstacle has 2000 boundary points), e.g., it takes more than $300s$ for HM when there are more than 12 obstacles.

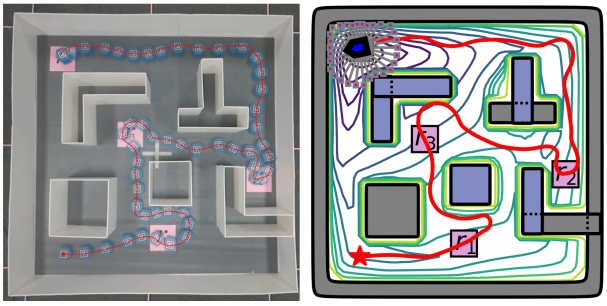

Figure 13: **Left:** Snapshots of hardware experiments. **Right:** Final trajectory with the final potential fields, where obstacles in blue are detected online.

5.4. Hardware Experiments

The proposed method is deployed to a differential-driven robot of radius $r_r = 0.1m$ As shown in Fig. 13, the workspace is constructed by whiteboards of size $2m \times 2m$, which has 2 independent and 6 overlapping obstacles. The controller gains in (24) are set to $k_v = 0.15$ and $k_\omega = 0.3$. The robot state is monitored by an overhead camera and the communication between the robot and the workstation is achieved by ROS with a frequency of 10Hz. The point clouds are collected within a radius of 0.3m around the robot. Other parameters are similar to the simulation study. The surveillance task is similar to the simulated case for regions r_1, r_2, r_3, r_4. The robot starts from $(0.25m, 0.2m)$ with orientation 0. The initial task plan is derived within $0.48s$ as $r_1\, r_2\, r_3\, r_4$. During the online execution, the robot adjusts its high-level plan and underlying harmonic potentials as the environment changes. As the robot detects more obstacles and familiar with the workspace, the task plan is updated as $r_1 r_3 r_4 r_2$ and finally converges to $r_1 r_3 r_2 r_4$. The whole task is accomplished in $169s$. The resulting trajectory is shown in Fig. 13 with a total distance $5.83m$, which is safe and smooth despite of the localization and control uncertainties. These results are consistent with the simulations above.

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. Future work involves the consideration of dynamic environments, multi-agent scenarios and 3D navigation.

References

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

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.

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). IEEE.

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 (ICRA)* (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.

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, 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.

Kim, J.-o., & Khosla, P. (1992). Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, *8*, 338–349.

Ko, I., Kim, B., & Park, F. C. (2013). Vf-rrt: Introducing optimization into randomized motion planning. In *IEEE Asian Control Conference (ASCC)* (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* (pp. 1–6). volume 4.

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 (CDC)* (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 (ICRA)* (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 (IROS)* (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.

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 & 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1726–1731). IEEE.

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 (ICRA)* (pp. 316–317).