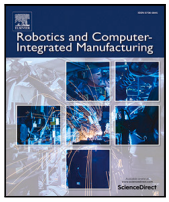
Robotics and Computer–Integrated Manufacturing 82 (2023) 102522

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

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



Full length article

A hierarchical long short term safety framework for efficient robot manipulation under uncertainty

Suqin He [a,1], Weiye Zhao [b,1], Chuxiong Hu [a,*], Yu Zhu [a], Changliu Liu [b,*]

[a] State Key Laboratory of Tribology & Beijing Key Laboratory of Precision/Ultra-Precision Manufacture Equipments and Control, Department of Mechanical Engineering, Tsinghua University, Beijing, 100084, China
[b] Robotics Institute, Carnegie Mellon University, Pittsburgh, 15213, PA, USA

ARTICLE INFO

Keywords:
Robot safety
Safe control
Hierarchical control
Motion planning

ABSTRACT

Safe and efficient robot manipulation in uncertain clustered environments has been recognized to be a key element of future intelligent industrial robots. Unlike traditional robots that work in structured and deterministic environments, intelligent industrial robots need to operate in dynamically changing and stochastic environments with limited computation resources. This paper proposed a hierarchical long short term safety system (HLSTS), where the upper layer contains a long term planner for global reference trajectory generation and the lower layer contains a short term planner for real-time emergent safety maneuvers. Additionally, a hierarchical coordinator is proposed to enable smooth coordination of the two layers by compensating the communication delay through trajectory modification. The theoretical results verify that the long term planner can always find a feasible trajectory (feasibility guarantee); and the short term planner can guarantee safety in the probabilistic sense. The proposed architecture is validated in industrial settings in both simulations and real robot experiments, where the robot is interacting with randomly moving obstacles while performing a goal reaching task. Experimental results demonstrate that the proposed HLSTS framework not only guarantees safety but also improves task efficiency.

1. Introduction

A key requirement for future robots is that they should be able to operate safely and efficiently in uncertain clustered environments [1]. The applications range from robot manipulators that collaborate with humans in factories, to autonomous vehicles that interact with various road participants. It is desired to ensure that the robot system can achieve its task efficiently while providing safety guarantee. The safety requirement considered in this paper is a hard constraint on the system's state space, e.g., a collision avoidance constraint. It is challenging to design a safe and efficient robotic system that satisfies hard constraint while maximizing task efficiency, for the following reasons.

Firstly, there are non-trivial environmental uncertainties: (1) the measurements of other entities in the environment (other agents or obstacles) can be noisy which may lead to large prediction errors on the entities' states; (2) the robot may experience large tracking errors due to its own model uncertainty and actuation noises. In order to be provably safe, the robot needs to meet the safety constraint in the worst case scenario, which may result in conservative and inefficient behaviors.

Secondly, the robot system can have low task efficiency when the planning and control do not look far into the future, which may result in unsmooth movements, or get stuck at livelock/deadlock. It has been shown by Grover et al. [2] that robots that utilize the reactive safe control methods, such as Control Barrier Function (CBF) [3], are prone to deadlock, where robots get stuck in some locations before completing their tasks. It is desired to equip the robot with capabilities to foresee the potential deadlocks/livelocks and avoid these inefficient motions.

Thirdly, the computational capacity of the robot systems is usually limited, which may delay the robot's response to generate new reference trajectories in emergency situations. To ensure real-time safety, the robot should be safe guarded by a safety module that can act defensively and respond to emergencies in real-time.

There are methods that are able to address a subset of these challenges. To tackle the first challenge under the dynamic uncertainty for the ego robot, FasTrack can plan a trajectory that is robustly safe against the worst-case deviations from the planned trajectory [4].

* Corresponding authors.
E-mail addresses: hesq16@mails.tsinghua.edu.cn (S. He), weiyezha@andrew.cmu.edu (W. Zhao), cxhu@tsinghua.edu.cn (C. Hu), zhuyu@tsinghua.edu.cn (Y. Zhu), cliu6@andrew.cmu.edu (C. Liu).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.rcim.2022.102522
Received 12 May 2022; Received in revised form 11 October 2022; Accepted 28 December 2022
Available online 16 January 2023

List of Symbols

$\boldsymbol{\xi}$	The closest point on the boundary of the infeasible set with respect to robot state
$\boldsymbol{s}$	Robot state trajectory
$\boldsymbol{s}_r$	Reference robot state trajectory
$\boldsymbol{s}_{\mathcal{O}}$	Obstacle trajectory
$\boldsymbol{u}$	Robot control command
$\boldsymbol{x}$	Robot configuration
$\boldsymbol{x}_0$	Robot start pose at first time step
$\boldsymbol{x}_r$	Robot reference pose from reference trajectory
$\boldsymbol{x}_{goal}$	Robot goal pose at final time step
δ	General uncertainties on robot and obstacle
δ_u, δ_x, δ_ξ	Robot control, state, and obstacle state uncertainties
Γ_u	Admissible control set
Γ_x	System state reachable set
D	Distance metric between robot and obstacle
$\mathcal{F}$	Convex feasible set
$\mathcal{J}$	Objective function
$\mathcal{C}$	Convex hull
ϕ	Safety index
D	Signed distance function in configuration space
d_m	Distance margin
M	Total trajectory horizon
N	Robot configuration space dimension
T	Time duration of the robot motion
t	Time
t_0	Initial time
t_s	Sampling time
U_S	Set of safe control
X	System state space
X_S	Safe set

There are planning strategies that can tackle the first challenge under the environmental uncertainties, e.g., fail-safe planning [5], and non-conservatively defensive strategy [6].

To tackle the second challenge, two categories of methods are adopted to improve the task efficiency, including reactive short term control methods and long term planning methods. Grover et al. [2,7] proposed a short term deadlock-resolution strategy to resolve deadlock of multiple mobile robot system, which ensures robots complete theirs tasks while avoiding collision. However, the proposed algorithm can neither ensure task completion nor guarantee deadlock resolution in clustered dynamic uncertain environments. On the other hand, long term planning methods [8–10] are able to generate the trajectory with long time horizon to improve the task efficiency while ensuring the trajectory is collision free with environmental obstacles. However, these methods tend to be computationally expensive due to long horizon sampling or nonlinear and nonconvex safety constraints (costs).

To address the third challenge, reactive safe control methods are proposed [3,11–13]. However, those traditional methods do not explicitly consider the improvement of task efficiency or the safety guarantee under environmental uncertainties. The methods like SEA [14], safe learning-based control [15], RL-CBF [16], backup-CBF [17] and adaptive CBF [18] can provide such real-time safe response with system dynamics parameters uncertainty. The core idea is to increase the safety margin in the control synthesis with respect to the environmental uncertainties.

These challenges constrain one another with numerous trade-offs between long term planning and short term planning. For example, long term planners improve the task efficiency by optimizing the future trajectory in a long time horizon, but this requires longer computation time and results in uncertainty accumulation (from both the environment and the trajectory itself). To ensure safety under those uncertainties, conservative robot motion is needed, which in turn will render the motion inefficient. On the other hand, short term reactive safety modules guarantee safety through high-frequency reactions. However, the resulting trajectory is easier to be trapped into deadlocks where the safe reaction conflicts with task completion. Thus the single scheme planning is not desired, and we desire to jointly address all challenges in a unified framework to maximize the task efficiency of the robot system while satisfying the real-time safety constraint under environmental uncertainties.

This paper proposes a hierarchical long short term safety system (HLSTS), where the upper layer contains (1) a long term planner for global reference trajectory generation, (2) the lower layer contains a short term planner for real-time emergent safety maneuvers, and (3) the middle layer contains the hierarchical coordinator to deal with communication latency. The proposed HLSTS differs from the state-of-the-art methods through a hierarchical framework which jointly addresses the aforementioned challenges to equip the robot system with high task efficiency and provable safety guarantees under environmental uncertainties. The long term planning module aims to tackle the second challenge by equipping the system with the ability to generate a trajectory that is task efficient and safe with respect to predicted obstacles. The short term planning module jointly addresses the first and third challenges by providing the reactive safety guarantees. Additionally, a hierarchical coordinator is proposed to enable smooth coordination of the two layers by compensating the communication delay through trajectory modification. The key contributions of this paper are summarized below:

- We propose a unique parallel planning architecture that integrates previously developed algorithms to handle both safety and efficiency under system uncertainty and computation limits.
- We provide the theoretical proofs to verify that the long term planner can always find a feasible trajectory (feasibility guarantee); the short term planner can also guarantee safety in the probabilistic sense.
- We integrate and test the proposed architecture on industrial settings in both simulations and real robot experiments, which demonstrates that the proposed HLSTS framework not only guarantees safety but also improves task efficiency.

The remainder of the paper is organized as follows Section 2 reviews prior work. Section 3 formulates the problem. Section 4 discusses safe set algorithm (SSA) for short term planning, while Section 5 discusses convex feasible set (CFS) method for long term planning. Section 6 proposes the hierarchical long and short term safety system, where the coordination mechanism is elaborated. Section 7 presents the hypothesis for the HLSTS framework. Section 8 introduces the evaluation experiments and discusses the results. Section 9 concludes the paper. A presents the theoretical results for the system.

2. Prior work

In this section, we are going to discuss prior work in three aspects: (1) long term planning; (2) short term planning; (3) coordination between long term and short term under uncertainties.

2.1. Efficient long term planning

Many long term planning methods have been proposed to ensure efficiency and safety of robot manipulation as well as the trajectory smoothness [19]. There are three types of algorithms for long

term motion planning [20]: sampling-based methods [21], searched-based methods [22] and optimization-based methods [19]. Sampling-based methods plan trajectories by generating random joint space displacements until the goal is reached. Representative methods include probabilistic road maps (PRM) [23] and rapidly-exploring random tree (RRT) [8], both of which can generate collision free trajectories. Building on top of RRT, Keil et al. [24] proposed meta-planning to generate reference trajectories using multiple online planners with offline computation to guarantee safety during tracking. Similar to sampling based method, searched based methods also expand the trajectory through node construction while satisfying predefined heuristic. However, the reference trajectories planned by construction are usually not smooth [25], which may cause chattering motions. Meanwhile, these methods do not scale well with the dimensionality of the state space.

Optimization-based methods, on the other hand, generate much smoother trajectories compared to the sampling-based methods. To generate collision free long term trajectory through optimization, some methods incorporate the safety constraint as penalty into optimization objectives, such as CHOMP algorithm [9] and ITOMP algorithm [10]. However, unsafe reference trajectory may still be generated. It is desired to solve trajectory generation optimization problem while explicitly satisfying the safety constraints, which makes the optimization problem highly nonlinear and non-convex [26]. To solve the nonlinear and non-convex long term planning problem in real-time, a method called Convex Feasible Set (CFS) algorithm [19] is introduced which incorporates domain specific information to speed up the computation, e.g., the geometry of the problem.

The core idea of CFS is to solve a sequence of convex optimizations constrained in the convex feasible sets, which efficiently search the non-convex feasible space defined by the inequality constraints for solutions. The CFS algorithm handles problems that have the following two features: (1) The objective function is strictly convex and smooth. (2) The non-convex safety inequality constraints can be written as $\boldsymbol{s} \in \Lambda$ where $\Lambda = \cap_i \Lambda_i$, and $\Lambda_i = \{\boldsymbol{s} : \pi_i(\boldsymbol{s}) \geq 0\}$ where $\boldsymbol{s}$ is the robot state trajectory and π_i is a continuous, piecewise and semi-convex smooth function. The core idea of the CFS algorithm is to compute a convex feasible set $\mathcal{F} := \mathcal{F}(\boldsymbol{s}_r) \subset \Lambda$ around $\boldsymbol{s}_r$, where $\boldsymbol{s}_r$ is a given reference trajectory. For each constraint Λ_i, CFS algorithm finds a convex feasible set $\mathcal{F}_i$ and constructs the overall convex feasible set as $\mathcal{F}(\boldsymbol{s}_r) = \cap_i \mathcal{F}_i(\boldsymbol{s}_r)$. Liu et al. [19] discussed the complete rules of finding $\mathcal{F}_i$ in three cases: (1) Λ_i is convex, (2) the complementary of Λ_i is convex, and (3) neither Λ_i nor its complementary is convex.

In practice, different variations of the CFS algorithm have been proposed to address planning problems under different contexts. To address planning problems with nonlinear equality constraints, such as nonlinear system dynamics, Slack-CFS [27] is introduced to relax nonlinear equality constraints by introducing slack variables. Fast robot motion planner (FRMP) [28] is proposed to ensure that the generated trajectory is time-optimal. FRMP applies CFS algorithm to solve both trajectory planning problem and the associated velocity/acceleration profile optimization problem.

To apply trajectory generation in real world robotics applications, frequent replanning is required to compensate the imperfect trajectory tracking. Chen et al. [29] proposed the FOAD framework to perform real-time planning and replanning using the CFS algorithm. However, FOAD framework adopts the time driven replanning where replanning is triggered in every time step, which introduces a heavy computation load. This paper will introduce an event-triggered replanning scheme to save the computation effort.

2.2. Provably safe short term planning

In contrast with long term planning, which generates efficient and safe long horizon reference trajectories, short term planning methods are mainly used to address real-time safety. One extreme case of short term planning is reactive safe control [30]. The safe control law guarantees that the unsafe region of the system state space is not reachable. Usually, a scalar energy function, which can also be called as a potential function, a barrier function, or a safety index, is designed to evaluate how far the system state is from the boundary of the safe set. Representative reactive safe control methods include potential field method (PFM) [11], sliding mode algorithm (SMA) [13], control barrier function (CBF) [3], and safe set algorithm (SSA) [12]. Specifically, the design objective for safe control is to maintain the system state in the safe set. The safe set is conventionally defined such that X_S is a 0-sublevel set of a continuously differentiable function $\phi : X \mapsto \mathbb{R}$, i.e., $X_S = \{\boldsymbol{x} : \phi(\boldsymbol{x}) < 0\}$, where $\phi(\boldsymbol{x})$ is known as the safety index as shown in Fig. 1. The safety index maps the subset of "unsafe" states in the state space X to positive real values and "safe" states to zero or negative real values. The safety index ϕ has the following properties,

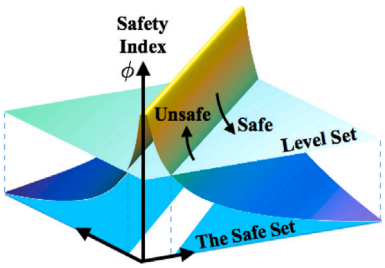


Fig. 1. Illustration of the safe set X_S and the safety index ϕ.

1. ϕ is differentiable with respect to t, i.e. $\dot{\phi} = (\partial\phi/\partial\boldsymbol{x})\dot{\boldsymbol{x}}$ exists everywhere;
2. $\partial\dot{\phi}/\partial\boldsymbol{u} \neq 0$;

The first condition is to ensure that ϕ is smooth. The second condition is to ensure that the robot control input $\boldsymbol{u}$ can always affect the safety index. If a control law is selected such that the closed-loop system dynamics always satisfy the following conditions, the safe set will be forward invariant and globally attractive [31]:

$$\dot{\phi}(\boldsymbol{x}, \boldsymbol{u}) \leq -\eta, \ \forall \ x \in \{x \ : \ \phi(\boldsymbol{x}) \geq 0\}, \tag{1}$$

for some $\eta > 0$. We say the system is safe if the safe set is forward invariant and globally attractive. This means that any system that starts at a state within the safe set will remain within the safe set, and any state which starts outside the safe set will be drawn towards the safe set in finite time $t \leq \phi/\eta$.

2.3. Hierarchical multi-layer systems

Hierarchical systems and the coordination of different system hierarchies have always been an active research area. In learning community, hierarchical reinforcement learning (HRL) [32,33] is proposed to decompose reinforcement learning problems into different levels of hierarchy, where higher level plans the parent problems, e.g. the different motion primitives. Lower level is invoked by higher level to complete the child problems, e.g. execute motion primitives [34]. The coordination of HRL is straightforward through different levels of problem assignment, whereas it is difficult for reinforcement learning methods to satisfy hard constraints, e.g. safety constraints [35] [36]. Compared with HRL system, the hierarchical model predictive control (HMPC) [37] system is more close to our robotics system where tasks are distributed to different layers and executed in different frequencies. However, nonlinear and nonconvex design methods for HMPC are still lacking, and some fundamental problems regarding the synchronization across different layers remain unsolved, such as transmission latency or loss of information. Some existing works adopt the idea of HMPC into hierarchical robotics system, such as the robot safe interaction

system for intelligent industrial co-robots (RSIS) [38] and safe and efficient robot collaboration system (SERoCS) [39]. Empirical results demonstrate that those hierarchical systems can efficiently make robots accomplish manipulation tasks while avoiding environmental obstacles. However, there still lacks principled approaches to design the coordination and to perform extensive evaluation, which are necessary to provide a deeper insight on how to improve the design of hierarchical system.

3. Problem formulation

In this section, we formulate the fundamental problem of safe and efficient robot motion planning in uncertain clustered environments. For simplicity, this paper focuses on the scenario with one robot and multiple moving obstacles. The methodology can be extended to scenarios with multiple robots and multiple moving obstacles.

We denote the continuous robot trajectory from current time t_0 to time $t_0 + T$ as $\boldsymbol{x}(t_0 : t_0 + T)$, where $\boldsymbol{x} \in \mathbb{R}^N$. By sampling the trajectory by time t_s, i.e., let $t_0 = t^{[1]} < t^{[2]} < \cdots < t^{[M]} = t_0 + T$, $t^{[i+1]} - t^{[i]} = t_s, i = 1, 2, \ldots, M-1$, then we have the time-stamped robot trajectory $\boldsymbol{s} = [\boldsymbol{x}^{[1]}; \cdots; \boldsymbol{x}^{[i]}; \cdots; \boldsymbol{x}^{[M]}]$, where $i = 1, 2, \ldots, M$.

The robot control trajectory from current time t to time $t + T$ is denoted as $\boldsymbol{a} = [\boldsymbol{u}^{[1]}; \ldots; \boldsymbol{u}^{[i]}; \ldots; \boldsymbol{u}^{[M]}], \boldsymbol{u} \in \mathbb{R}^N$. The planning horizon T can either be chosen as a fixed number or as a decision variable that should be optimized up to the accomplishment of the task. Similarly, the trajectories of environmental obstacles from t to $t + T$ are denoted as $\boldsymbol{s}_{\mathcal{O}}$, which is the ground truth obstacle trajectory, and is not directly accessible to the robot but can rather be estimated through measurement and prediction. To obtain optimal robot trajectory in uncertain clustered environment, the following optimization problem is considered,

$$\min_{\boldsymbol{s},\boldsymbol{u}} \mathcal{J}(\boldsymbol{s}, \boldsymbol{u}), \tag{2a}$$

$$s.t. \quad \forall i = 1, \ldots, M, \ \boldsymbol{x}^{[i]} \in \Gamma_x, \boldsymbol{u}^{[i]} \in \Gamma_u, \tag{2b}$$

$$\dot{\boldsymbol{x}}^{[i]} = f(\boldsymbol{x}^{[i]}) + g(\boldsymbol{x}^{[i]})\boldsymbol{u}^{[i]}, \tag{2c}$$

$$\boldsymbol{x}^{[i]} \in X_S, \tag{2d}$$

$$\boldsymbol{x}^{[1]} = \boldsymbol{x}(t_0), \boldsymbol{x}^{[M]} = \boldsymbol{x}_{goal}, \tag{2e}$$

where (2a) is the objective function, which evaluates the trajectory smoothness and task efficiency. $\mathcal{J}$ is designed as:

$$\mathcal{J}(\boldsymbol{s}, \boldsymbol{u}) = \sum_{i=t}^{t+T} (w_1^{[i]} \|\boldsymbol{x}^{[i]} - \boldsymbol{x}_{goal}\|^2 + w_2^{[i]} \|\boldsymbol{u}^{[i]}\|^2) \tag{3}$$

where $\boldsymbol{x}_{goal}$ is the goal configuration for the robot state. Smaller difference norm between the current state $\boldsymbol{x}(t)$ and the target configuration indicates higher task efficiency. The control norm penalty aims to minimize the control effort to reduce the chattering during robot operation. $w_1^{[i]}, w_2^{[i]} \in \mathbb{R}^+$ denote weight terms.

Eq. (2b) are the constraints on the robot state space Γ_x (such as joint limits) and control space Γ_u (such as control saturation). Eq. (2c) is the dynamic constraint which is assumed to be affine in the control input. Eq. (2d) is the safety constraint. In this paper, safety constraint is considered as the constraint for collision avoidance. Eqs. (2e) are the initial and end pose constraints.

However, solving (2) is not a trivial task, where both efficiency and safety objectives are coupled together. Note that it is impossible to solve such a long horizon optimization problem in a high frequency due to limited computation capacity of real world robot systems, whereas safety is a hard constraint that should be satisfied in real-time. This fact motivates us to design a system with a principle to separate the efficiency objective and safety objective when operating in different frequencies, such that the safety is guaranteed at the maximum possible frequency and efficiency objective is satisfied at lower frequency allowed by the remaining computation resources.

Following the prescribed design principle, we define (1) a long term objective in terms of efficiency for the whole trajectory which is optimized at a low frequency; and (2) a short term objective in terms of real-time safety which is optimized at a high frequency. In the following discussion, we will introduce the dismantled objectives for long term and short term, separately.

Firstly, we discuss the optimization problem for long term planning. Intuitively, the long term planner will generate a sequence of state trajectory $\boldsymbol{s}$ to guide the short term planner, such that greater task efficiency and trajectory smoothness can be achieved while satisfying safety constraints. Suppose the trajectory generated from long term planner is discretized into M points, we denote the reference trajectory as: $\boldsymbol{s}_r := [\boldsymbol{x}_r^{[1]}; \ldots; \boldsymbol{x}_r^{[M]}]$, where $\boldsymbol{x}_r^{[1]} = \boldsymbol{x}(t_0)$ is the initial robot pose, and the last point of the reference trajectory should reach the goal such that $\boldsymbol{x}_r^{[M]} = \boldsymbol{x}_{goal}$. As described in (3), the system efficiency objective is to minimize the distance between robot state and the target state along the trajectory. Therefore, the most efficient reference trajectory can be generated using direct linear interpolation from the current state to the goal state, where the linear interpolation indicates that $\boldsymbol{x}_r^{[i]} = \frac{i-1}{M-1}(\boldsymbol{x}_{goal} - \boldsymbol{x}_r^{[1]}) + \boldsymbol{x}_r^{[1]}$. We also treat linear interpolation as the default $\boldsymbol{s}_r$, whereas $\boldsymbol{x}_r^{[i]}$ does not necessarily satisfy the safety requirements. Therefore, we design the optimization objective as to encourage the new trajectory to be smooth and close to reference trajectory, and pose the optimization constraints as the safety requirements. Mathematically, the long term optimization problem is formulated as following:

$$\min_{\boldsymbol{s}} \ J(\boldsymbol{s}) = \|\boldsymbol{s} - \boldsymbol{s}_r\|_{Q_r}^2 + \|\boldsymbol{s}\|_{Q_s}^2 \tag{4a}$$

$$s.t. \ \boldsymbol{x}^{[i]} \in X_S, i = 1, 2, \ldots, M \tag{4b}$$

$$\boldsymbol{x}^{[i]} \in \Gamma_x \tag{4c}$$

$$\boldsymbol{x}^{[1]} = \boldsymbol{x}(t_0), \boldsymbol{x}^{[M]} = \boldsymbol{x}_{goal} \tag{4d}$$

where Q_r and Q_s are both positive semi-definite, $\|\boldsymbol{s} - \boldsymbol{s}_r\|_{Q_r}^2 = (\boldsymbol{s} - \boldsymbol{s}_r)^T Q_r (\boldsymbol{s} - \boldsymbol{s}_r)$ penalizes the deviation from the new trajectory to the reference trajectory, and $\|\boldsymbol{s}\|_{Q_s}^2 = \boldsymbol{s}^T Q_s \boldsymbol{s}$ penalizes the properties of the new trajectory itself.

After finishing the long term optimization, the short term planner receives the planned reference trajectory from the long term planner. The short term planner is firstly formulated as a reference trajectory tracking controller to generate a reference input $\boldsymbol{u}_r$ at each control cycle. Then, certain modification on the $\boldsymbol{u}_r$ is needed to ensure that the interaction constraint $\boldsymbol{x} \in X_S$ will be satisfied after applying the new input. The short term planning problem can be formulated as the following optimization,

$$\min_{\boldsymbol{u}} \|\boldsymbol{u} - \boldsymbol{u}_r\|_{Q_u}^2, \tag{5a}$$

$$s.t. \ \boldsymbol{u} \in \Gamma_u, \boldsymbol{x} \in \Gamma_x, \dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}, \tag{5b}$$

$$\boldsymbol{x} \in X_S \tag{5c}$$

where $\|\boldsymbol{u} - \boldsymbol{u}_r\|_{Q_u}^2 = (\boldsymbol{u} - \boldsymbol{u}_r)^\top Q_u (\boldsymbol{u} - \boldsymbol{u}_r)$ penalizes the deviation from the reference input, where Q_u should be designed as a second order approximation of $\mathcal{J}$, e.g. $Q_u \approx d^2\mathcal{J}/d(\boldsymbol{u})^2$. The constraints are the same as the constraints in (2). Note that modifying the control input is equivalent to modifying the reference trajectory. Without loss of generality, it is assumed that $\boldsymbol{x} \in X_S$ implies that $\boldsymbol{x} \in \Gamma_x$. Otherwise, we just take the intersection of the two constraints. The safe set and the robot dynamics impose nonlinear and non-convex constraints which make the problem hard to solve. In Section 4, we will transform the non-convex state space constraint into convex control space constraint using the idea of invariant set.

By solving the dismantled objectives, the long term planner pass as the reference trajectory to the short term planner, which computes the safe control and further control the robot. Building on top of the long term planner and short term planner, we propose a hierarchical long short term safety framework (HLSTS) as demonstrated in Fig. 2. In

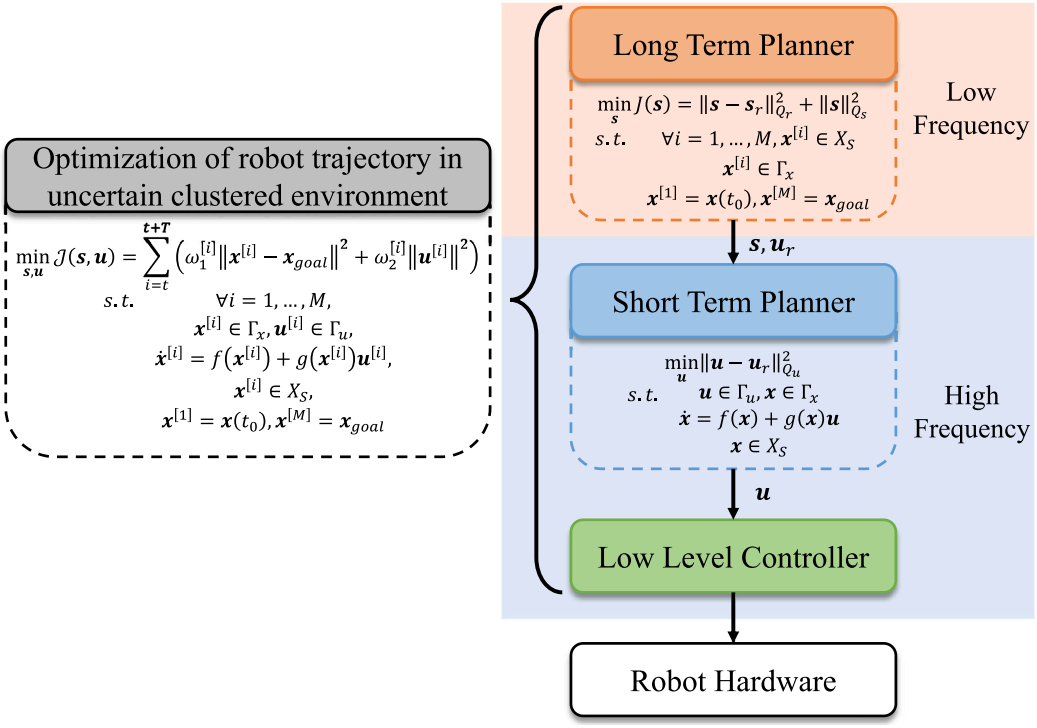


Fig. 2. The designed architecture of the hierarchical long short term safety system. The original optimization problem of robot trajectory in uncertain clustered environment is dismantled into (1) the long term optimization problem and (2) the short term optimization problem. The long term planner generates the long time horizontal reference trajectory s at a low frequency. The short term planner applies real-time reactive safe control u to guarantee safety at a high frequency. Then the low level controller executes the desired control u on the robot hardware.

the following sections, we target to design proper methods to solve (4) and (5), respectively. Additionally, we desire a suitable coordination mechanism such that the HLSTS system maximizes the task efficiency of the robot system while satisfying real-time safety constraint under environmental uncertainties.

4. The safety-oriented short term planning

Firstly, we discusses how to address the safety following the invariant safe set definition in Section 2.2. To ensure safety, the robot control must be chosen from the set of safe control $U_S(t) = \{u(t) : \dot{\phi}(x,u) \leq -\eta(\phi)$ when $\phi(x) \geq 0\}$ where $\eta : \mathbb{R} \mapsto \mathbb{R}^+$ is a general function of ϕ. By the dynamics equation in (5b), the derivative of the safety index can be written as $\dot{\phi} = L_f\phi(x) + L_g\phi(x)u$, where $L_f\phi(x) = \frac{\partial\phi}{\partial x} f(x)$ and $L_g\phi(x) = \frac{\partial\phi}{\partial x} g(x)$. Then

$$U_S(t) = \left\{u(t) : \ L_g\phi \cdot u(t) \leq -\eta - L_f\phi \text{ when } \phi \geq 0\right\}, \tag{6}$$

Due to the introduction of the set of safe control, the non-convex safe set constraint X_S is transformed to a convex control space constraint U_S. Since Γ_u is usually convex, the problem (5) is transformed to a quadratic programming optimization,

$$\min_u \|u - u_r\|^2_Q, \tag{7a}$$

$$s.t. \ u \in \Gamma_u \cap U_S. \tag{7b}$$

The switch condition between $\phi \geq 0$ and $\phi < 0$ may result in oscillation for discrete time implementation. Therefore, a smoothed version of the algorithm is discussed in [40]. To ensure the feasibility of (7b) constraint, i.e. there always exists non-empty $\Gamma_u \cap U_S$, we can synthesize a properly parameterized safety index ϕ as in [41–43], where the problem (7) is guaranteed to be feasible. In the next step, the above-mentioned safe set algorithm (SSA) will be applied in our robotics application and the detailed structural design will be illustrated.

In real world applications, robotics systems usually take joint velocity or joint acceleration/torque as the control input. Thus, depending on the robot control input, the SSA can be classified as Velocity-SSA and Acceleration-SSA. For both types of SSA, we build their detailed structure as follows.

4.1. Velocity-SSA

For the Velocity-SSA, the control input $u = \dot{x}$ is the joint velocity, and the safety index ϕ is defined as

$$\phi(x) = \phi_0(x) = d_m - D(x), \tag{8}$$

where $d_m \geq 0$ is a positive safety margin, $D(x) : \mathbb{R}^N \mapsto \mathbb{R}$ is a metric evaluating the signed distance between the robot and the obstacles.

For most real world robotic systems, the desired control input for the robot is supposed to be smooth. Therefore, we use the sub level safe set algorithm to construct U_S where $\dot{\phi}$ satisfies

$$\dot{\phi} \leq -k_s\phi(x), \quad \text{when } \phi(x) \geq 0 \tag{9}$$

where $k_s \in \mathbb{R}^+$ is a positive gain. In this way, the robot motion during the safe control is smoother than the original constant change rate. The $\dot{\phi}$ can be further represented by

$$\dot{\phi}(x, \dot{x}) = \nabla_x \phi^\top \dot{x} \tag{10}$$

It is assumed that $\Gamma_u = \{u : \ u_{min} \leq u \leq u_{max}\}$ represents the control saturation for the robot, where u_{min} and u_{max} are constant bounds. Suppose the original joint velocity command without SSA control is $\dot{x}_o$. After applying SSA control, the new joint velocity command is $\dot{x}_c$. Thus, the optimization problem (7) can be formulated as

$$\min_{\dot{x}_c} \|\dot{x}_c - \dot{x}_o\|_2 \tag{11a}$$

$$\text{s.t. } \nabla_x \phi^\top \dot{x}_c \leq -k_s\phi(x) \text{ or } \phi(x) < 0 \tag{11b}$$

$$u_{min} \leq \dot{x}_c \leq u_{max} \tag{11c}$$

It is noted that when the control saturation constraint (11c) is not activated, the analytical solution of (11) can be obtained through KKT condition and the resulting Velocity-SSA control law is

$$\dot{\boldsymbol{x}}_c = \begin{cases} \dot{\boldsymbol{x}}_o, & \phi(\boldsymbol{x}) < 0 \\ \dot{\boldsymbol{x}}_o - \frac{k_s\phi + \nabla_{\boldsymbol{x}}\phi^\top \dot{\boldsymbol{x}}_o}{\nabla_{\boldsymbol{x}}\phi^\top \nabla_{\boldsymbol{x}}\phi}\nabla_{\boldsymbol{x}}\phi, & \phi(\boldsymbol{x}) \geq 0 \end{cases} \tag{12}$$

In practice, a suitable safety margin will be used, and the robustness of the SSA controller can be improved by tuning coefficient k_s. Note that the conclusion from (12) does not consider the control saturation constraint (11c). When (11c) is activated, we can synthesize a properly parameterized safety index ϕ, such that there always exists non-empty set of safe control within the control saturation [41,44,45]. Thus, the safe control solution for (11) can be obtained through quadratic programming.

4.2. Acceleration-SSA

For the Acceleration-SSA, the control input $\boldsymbol{u} = \ddot{\boldsymbol{x}}$ is the joint acceleration, and the safety index is defined as in [12]

$$\phi(\boldsymbol{x}, \dot{\boldsymbol{x}}) = d_m^2 - D^2 - k_1\dot{D}, \tag{13}$$

where $k_1 \geq 0$ is a positive scalar. Similar to the Velocity-SSA, a sub level safe set is used so that $\dot{\phi}$ satisfies

$$\dot{\phi} \leq -k_s\phi(\boldsymbol{x}, \dot{\boldsymbol{x}}), \quad \text{when } \phi \geq 0 \tag{14}$$

And $\dot{\phi}$ can be represented as

$$\begin{aligned} \dot{\phi} &= \nabla_{\boldsymbol{x}}\phi^\top \dot{\boldsymbol{x}} + \nabla_{\dot{\boldsymbol{x}}}\phi^\top \ddot{\boldsymbol{x}} \\ \nabla_{\boldsymbol{x}}\phi &= 2D\frac{\partial \phi_0}{\partial \boldsymbol{x}} + k_1\frac{\partial^2 \phi_0}{\partial \boldsymbol{x}^2}\dot{\boldsymbol{x}} \\ \nabla_{\dot{\boldsymbol{x}}}\phi &= k_1\frac{\partial \phi_0}{\partial \boldsymbol{x}} \end{aligned} \tag{15}$$

Suppose the original joint acceleration command without SSA control is $\ddot{\boldsymbol{x}}_o$. After applying SSA control, the new joint acceleration command is $\ddot{\boldsymbol{x}}_c$. We can similarly formulate the optimization problem (7) as

$$\min_{\ddot{\boldsymbol{x}}_c} \|\ddot{\boldsymbol{x}}_c - \ddot{\boldsymbol{x}}_o\|_2 \tag{16a}$$

$$\text{s.t. } \nabla_{\boldsymbol{x}}\phi^\top \dot{\boldsymbol{x}} + \nabla_{\dot{\boldsymbol{x}}}\phi^\top \ddot{\boldsymbol{x}}_c \leq -k_s\phi, \quad \text{if } \phi > 0 \tag{16b}$$

$$\boldsymbol{u}_{min} \leq \ddot{\boldsymbol{x}}_c \leq \boldsymbol{u}_{max} \tag{16c}$$

Similarly, when the control saturation constraint (16c) is not activated, the resulting Acceleration-SSA control law as

$$\ddot{\boldsymbol{x}}_c = \begin{cases} \ddot{\boldsymbol{x}}_o, & \phi < 0 \\ \ddot{\boldsymbol{x}}_o - \frac{k_s\phi + \nabla_{\dot{\boldsymbol{x}}}\phi^\top \ddot{\boldsymbol{x}}_o + \nabla_{\boldsymbol{x}}\phi^\top \dot{\boldsymbol{x}}}{\nabla_{\dot{\boldsymbol{x}}}\phi^\top \nabla_{\dot{\boldsymbol{x}}}\phi}\nabla_{\dot{\boldsymbol{x}}}\phi, & \phi \geq 0 \end{cases} \tag{17}$$

In practice, the choice of Velocity-SSA and Acceleration-SSA depends on the input of the servo system (i.e., the servo controller is working on the velocity-loop or the acceleration-loop) as well as where the constraints are defined (i.e., the velocity constraint or acceleration constraint). Since the joint acceleration command from Acceleration-SSA is integrated to joint velocity, the actual robot motion using Acceleration-SSA is smoother than the robot motion using Velocity-SSA. However, in most industrial applications, the servo controller only provides a joint position/velocity control interface. Thus we can only apply Velocity-SSA, which has lesser computation complexity as well. And for the joint position interface, we can multiply the joint velocity command by control cycle time, and feed the result to the position controller. As long as the servo control frequency is not too slow, this approximation is accurate enough for safe control.

5. Efficiency-oriented long term planning

The proposed efficiency oriented long term planner aims to generate the reference trajectory to optimize the task efficiency and trajectory smoothness while satisfying safety constraints. As discussed in Section 3, $\|\boldsymbol{s} - \boldsymbol{s}_r\|_{Q_r}^2$ penalizes the deviation from the new trajectory to the reference trajectory, and $\|\boldsymbol{s}\|_{Q_s}^2$ penalizes the properties of the new trajectory itself. Specifically, we construct the positive semi-definite matrices $Q_r, Q_s \in \mathbb{R}^{NM\times NM}$ from three components, including the matrices for position, velocity and acceleration [19]. (1) We denote $Q_1 = I_N M$ as the position matrix, where I_N represents the identity matrix with dimension N. (2) We denote $Q_2 = V^\top V$ as the velocity matrix, where $V \in \mathbb{R}^{N(M-1)\times NM}$ is a finite difference operator, such that:

$$V = \frac{1}{t_s}\begin{bmatrix} I & -I & 0 & 0 & \cdots & 0 \\ 0 & I & -I & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & I & -I \end{bmatrix}. \tag{18}$$

(3) We denote $Q_3 = A^\top A$ as the acceleration matrix, and $A \in \mathbb{R}^{N(M-2)\times NM}$ is also a finite difference operator, such that:

$$A = \frac{1}{t_s^2}\begin{bmatrix} I & -2I & I & 0 & \cdots & 0 \\ 0 & I & -2I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & I & -2I & I \end{bmatrix} \tag{19}$$

Then we set $Q_r = \sum_{i=1}^3 c_i^r Q_i$ and $Q_s = \sum_{i=1}^3 c_i^s Q_i$, where c_i^r and c_i^s are the positive weights. Therefore, the minimization of $\|\boldsymbol{s} - \boldsymbol{s}_r\|_{Q_r}^2$ minimizes the distance between the trajectory $\boldsymbol{s}$ and the reference $\boldsymbol{s}_r$, which ensures the task efficiency. The minimization of $\|\boldsymbol{s}\|_{Q_s}^2$ ensures low velocity and acceleration magnitude, which ensures the trajectory smoothness and further benefits the trajectory interpolation and tracking. Constraint $\boldsymbol{x}^{[i]} \in X_S$ requires that the system states (including the robot state and the estimated obstacle state) should be in the safe set at each planning horizon.

However, solving the long term planning problem is not a trivial task, since the corresponding optimization problem (4) is highly nonconvex, where the non-convexity mainly comes from the highly nonlinear inequality constraints (4b) and (4c). It is computationally expensive to obtain a solution using generic nonlinear optimization solvers such as sequential quadratic programming (SQP) [46]. Therefore, it is desired to incorporate the domain specific information, e.g., the geometry of the problem, to improve the computational efficiency while solving the optimization. In this work, we adopt Convex Feasible Set (CFS) algorithm [47] to tackle (4), where we directly convexify the optimization problem using domain knowledge. CFS algorithm has been shown to work successfully in practice and can efficiently find optimal global planning solutions in real-time that are strictly feasible [19]. in the following subsections, we are going to (1) briefly introduce the CFS algorithm to solve (4), and (2) provide an example of long term planning for a six-degrees-of-freedom robot arm.

5.1. Convex feasible set algorithm

To make the computation more efficient, we transform (4) into a sequence of convex optimizations by obtaining a sequence of convex feasible sets inside the non-convex domain X_S. The general methods in constructing the convex feasible set are discussed in [47]. In the following discussions, we will review the major steps to find convex feasible sets.

In this paper, we consider the continuously moving environmental obstacles. In order to meet the safety requirement $\boldsymbol{x}^{[i]} \in X_S$ continuously, we require the robot state at time step i to not be in the infeasible sets from time step i to time step $i+1$, which are denoted as $C_j^{[i:i+1]}$, where $j \in \mathbb{N}^+$ denotes the jth infeasible set. Based on the $C_j^{[i:i+1]}$, we give the following definition of distance constraint:

Definition 1 (*Distance Constraint*)**.** Define the signed distance function from time step i to $i+1$ as:

$$D_j^{[i:i+1]}(\boldsymbol{x}) = \begin{cases} \min_{o\in\partial C_j^{[i:i+1]}} \|\boldsymbol{x}-\boldsymbol{o}\| & \boldsymbol{x}\notin C_j^{[i:i+1]} \\ -\min_{o\in\partial C_j^{[i:i+1]}} \|\boldsymbol{x}-\boldsymbol{o}\| & \boldsymbol{x}\in C_j^{[i:i+1]} \end{cases} \tag{20}$$

where ∂C_j denotes the boundary of the infeasible set C_j. Then the distance constraints from time step i to $i+1$ are $D_j^{[i:i+1]}(\boldsymbol{x}) \geq 0, \forall j \in \mathbb{N}^+$.

Then we can rewrite the constraint (4b) using distance constraints as $D_j^{[i:i+1]}(\boldsymbol{x}) \geq 0, \forall j \in \mathbb{N}^+$.

Next, the convex feasible subsets for (4) can be computed leveraging the geometric property of $C_j^{[i:i+1]}$. For the case where $C_j^{[i:i+1]}$ is convex, the function $D_j^{[i:i+1]}(\boldsymbol{x})$ is also convex [48]. Hence $D_j^{[i:i+1]}(\boldsymbol{x}) \geq D_j^{[i:i+1]}(\boldsymbol{x}_r) + \nabla_{\boldsymbol{x}} D_j^{[i:i+1]}(\boldsymbol{x}_r)(\boldsymbol{x}-\boldsymbol{x}_r)$ for any reference point $\boldsymbol{x}_r$. Then $D_j^{[i:i+1]}(\boldsymbol{x}_r) + \nabla_{\boldsymbol{x}} D_j^{[i:i+1]}(\boldsymbol{x}_r)(\boldsymbol{x}-\boldsymbol{x}_r) \geq 0$ implies that $\boldsymbol{x} \notin C_j^{[i:i+1]}$ and it is feasible. For the case where $C_j^{[i:i+1]}$ is non-convex, we can manually break it into several overlapping infeasible convex subsets $C_{j,q}^{[i:i+1]}$, such that $C_j^{[i:i+1]} = \bigcup_q C_{j,q}^{[i:i+1]}$. The same rule applies for each $C_{j,q}^{[i:i+1]}$.

Therefore, the convex feasible set $\mathcal{F}(\boldsymbol{s}_r)$ for (4) can be constructed as:

$$\begin{aligned} \mathcal{F}(\boldsymbol{s}_r) = \{&\boldsymbol{x} : \boldsymbol{x} \in \Gamma_x, \\ &D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) + \nabla_{\boldsymbol{x}} D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]})(\boldsymbol{x}^{[i]} - \boldsymbol{x}_r^{[i]}) \geq 0, \forall j, i\}, \end{aligned} \tag{21}$$

which is a convex subset of Γ_x. Thus, (4) can be solved iteratively within computed convex feasible set:

$$\boldsymbol{s}^{(k+1)} = \underset{\boldsymbol{s}\in\mathcal{F}(\boldsymbol{s}_r^{(k)})}{\operatorname{argmin}} J(\boldsymbol{s}). \tag{22}$$

where the reference trajectory is updated using solution from last iteration $\boldsymbol{s}_r^{(k)} = \boldsymbol{s}^{(k)}$, and the initial reference trajectory is $\boldsymbol{s}_r^{(1)} = \boldsymbol{s}_r$. The convex optimization (22) will be solved iteratively until either the solution converges or the descent of objective J is small.

It has been proved [47] that the sequence $\{\boldsymbol{s}^{(k)}\}$ converges to a local optimum of problem (4). Moreover, the computation time can be greatly reduced using CFS. This is due to the fact that we directly search for solutions in the feasible area. Hence (1) the computation time per iteration is smaller than existing methods as no linear search is needed, and (2) the number of iterations is reduced as the step size (change of the trajectories between two consecutive steps) is unconstrained.

5.2. Example: Long term planning for a robot arm manipulator

In this subsection, we will discuss how to apply CFS algorithm to robot arm motion planning. Here we consider the scenarios where the six-degrees-of-freedom robot manipulator with initial pose $\boldsymbol{x}_0 \in \mathbb{R}^6$ at time t_0, and the robot is suppose to reach a target position at $\boldsymbol{x}_{goal} \in \mathbb{R}^6$ at time $t_0 + T$. Let $T = Mt_s$, where t_s is the sampling rate, we have the time-stamped robot trajectory as $\boldsymbol{s} = [\boldsymbol{x}^{[1]}; \dots; \boldsymbol{x}^{[i]}; \dots; \boldsymbol{x}^{[M]}]$, where $i \in \{1, 2, \dots, M\}$, where $\boldsymbol{x}^{[1]} = \boldsymbol{x}_0$, and $\boldsymbol{x}^{[M]} = \boldsymbol{x}_{goal}$. Due to the joint limits, the robot state constraints are: $\Gamma_x := \{\boldsymbol{x} : x_p \in [-\frac{\pi}{2}, \frac{\pi}{2}], p = 1, 2, \dots, 6\}$.

In the environment, there are other entities (obstacles) performing routine navigation tasks, which may interfere the robot maneuver. For simplicity, we assume the obstacles yield several convex non-overlapping infeasible sets in the robot configuration space, which are denoted as C_j.

Therefore, the safety constraint $\boldsymbol{x}^{[i]} \in X_S$ is equivalent to that the robot state is within the feasible set at every time step, which can then be represented as the signed distance function to infeasible sets $C_j^{[i:i+1]}$ is greater than $0, \forall j$, i.e. $D_j^{[i:i+1]}(\boldsymbol{x}^{[i]}) \geq 0$. Mathematically, the discretized optimization problem is formulated as:

$$\min_{\boldsymbol{s}} \ \|\boldsymbol{s}-\boldsymbol{s}_r\|_{Q_r}^2 + \|\boldsymbol{s}\|_{Q_s}^2, \tag{23a}$$

$$s.t.\ x_p^{[i]} \in [-\frac{\pi}{2}, \frac{\pi}{2}], p = 1, 2, \dots, 6, \tag{23b}$$

$$D_j^{[i:i+1]}(\boldsymbol{x}^{[i]}) \geq 0, \forall i, \forall j, \tag{23c}$$

$$\boldsymbol{x}^{[1]} = \boldsymbol{x}_0, \boldsymbol{x}^{[M]} = \boldsymbol{x}_{goal} \tag{23d}$$

where the reference trajectory $\boldsymbol{s}_r$ is the linear interpolation from $\boldsymbol{x}_0$ to $\boldsymbol{x}_{goal}$, such that $\boldsymbol{x}_r^{[i]} = \boldsymbol{x}_0 + \frac{i-1}{M-1}(\boldsymbol{x}_{goal} - \boldsymbol{x}_0)$.

Next, we will convexify the safety constraint (23c) according to (21). Given the reference trajectory $\boldsymbol{s}_r$, we have the configuration $\boldsymbol{x}^{[i]}$ at horizon $(i = 2, 3, \dots, M-1)$ should satisfy a linear inequality constraint $\boldsymbol{A}^{[i]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}^{[i]} \leq \boldsymbol{b}^{[i]}(\boldsymbol{x}_r^{[i]})$, where $\boldsymbol{A}^{[i]}(\boldsymbol{x}_r^{[i]}) = \begin{bmatrix} -\nabla_{\boldsymbol{x}} D_1^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) \\ -\nabla_{\boldsymbol{x}} D_2^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) \\ \vdots \end{bmatrix}$ and

$$\boldsymbol{b}^{[i]}(\boldsymbol{x}_r^{[i]}) = \begin{bmatrix} D_1^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) - \nabla_{\boldsymbol{x}} D_1^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}_r^{[i]} \\ D_2^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) - \nabla_{\boldsymbol{x}} D_2^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}_r^{[i]} \\ \vdots \end{bmatrix}.$$

Therefore, we can represent the convex feasible set $\mathcal{F}$ for trajectory $\boldsymbol{s}$ from (21) as $\boldsymbol{A}\boldsymbol{s} \leq \boldsymbol{b}$, where $\boldsymbol{A} = \text{diag}\{\boldsymbol{0}_{1\times 6}, \boldsymbol{A}^{[2]}, \dots, \boldsymbol{A}^{[i]}, \dots, \boldsymbol{A}^{[M-1]}, \boldsymbol{0}_{1\times 6}\}$ and $\boldsymbol{b} = [0; \boldsymbol{b}^{[2]}; \dots; \boldsymbol{b}^{[i]}; \dots; \boldsymbol{b}^{[M-1]}; 0]$.

With the convexified (23c), the motion planning optimization problem (23) is then solved using CFS iteratively until the solution converges or the objective value is small. Note that since both the convexified (23b) and (23c) are linear constraints and the objective (23a) is quadratic, we are actually solving a sequence of quadratic programs, which can be handled efficiently using off-the-shelf quadratic programming solvers.

6. Hierarchical long short term safety system

6.1. System workflow

The HLSTS system workflow is shown in Fig. 3. The system consists of three hardware/software layer, those are (1) the long term planner layer, (2) the coordinator layer, (3) the short term planner layer, and (4) the low-level control layer.

The long term planner and short term planner run in parallel with the control signal and data communication. The long term planner focuses on generating the initial reference trajectory or replanned trajectory $\boldsymbol{s}_r = [\boldsymbol{x}_r^{[1]}; \boldsymbol{x}_r^{[2]}; \dots; \boldsymbol{x}_r^{[M]}]$ by solving (4) for the entire system. The trajectory $\boldsymbol{s}_r$ is then sent to the hierarchical coordinator. The idea of the hierarchical coordinator is to deal with the latency in the communication between long term planner and short term planner. The hierarchical coordinator runs the trajectory interpolation and modification program and generates a modified reference trajectory $\boldsymbol{s}_d = [\boldsymbol{x}_d^{[1]}; \boldsymbol{x}_d^{[2]}; \dots; \boldsymbol{x}_d^{[M_d]}]$ for the short term planner, where $M_d > M$ is the number of the interpolated trajectory. The short term planner uses SSA control law (12) to generate the joint velocity command $\dot{\boldsymbol{x}}_c$. This command will be sent to the low-level controller to a simulated robot or a real robot, and certain servo controllers such as PID controller is required to track the $\dot{\boldsymbol{x}}_c$ command.

The global data are robot position $\boldsymbol{x}$, robot velocity $\dot{\boldsymbol{x}}$, goal position $\boldsymbol{x}_{goal}$, and the nearest obstacle position $\boldsymbol{\xi}$. Leveraging the high loop frequency for short term planner, the distance to obstacle is calculated in the short term planner and the safety index ϕ is then calculated. If $\phi \geq 0$, the SSA control is activated and a `replan_request` signal will be sent to the long term planner. Note that long term planning adopts the "Event Trigger" replanning mechanism, where the long term planner will replan a trajectory from the current state to goal state, if an uninterrupted sequence of `replan_request` are received (e.g., consecutive 3 requests in our 40 Hz implementation). Here, the "uninterrupted requests" is used as the replanning indicator to avoid occasional false requests due to measurement noises. Additionally, since the short term controller is running at a high frequency, it is suitable to serve as the replanning requester who monitors the system safety status (ϕ) in real-time. Through these control signals, the hierarchical system could run in parallel while keeping synchronization between different layers.

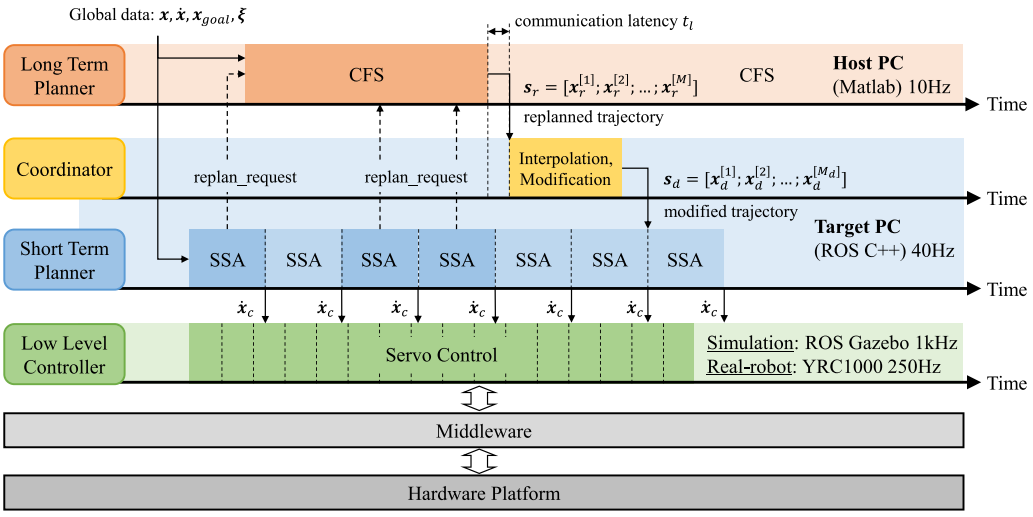


Fig. 3. Hierarchical Long Short Term Safety (HLSTS) system workflow timeline.

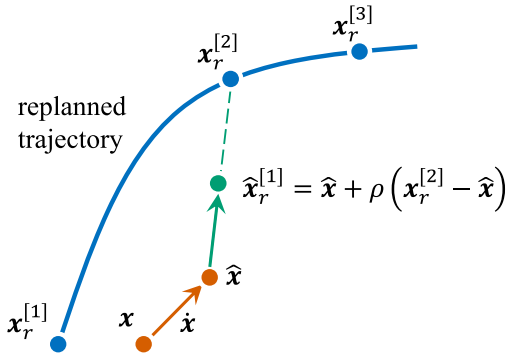


Fig. 4. Replanned trajectory modification in Coordinator.

6.2. Hierarchical coordinator

The Hierarchical Coordinator is used to solve the delay problem caused by two factors. The first one is the computation time in the long term planner. And the second one is the communication latency between long term planner and short term planner as shown in Fig. 3. The Hierarchical Coordinator has two major functions including the trajectory interpolation and trajectory modification. The trajectory generated in long term planner features a low sampling rate while the short term planner has a high sampling rate trajectory. Thus the trajectory interpolation is applied in the Hierarchical Coordinator to match these two different sampling rates. The interpolation will comply with the joint velocity limit.

The trajectory modification is applied when a new plan is generated by the long term planner. Due to the computation time needed for replanning and the communication delay between the long term planner and the short term planner, when the replanned trajectory arrives at the short term planner, the current position of the robot may not be aligned with the initial position of the replanned trajectory. As shown in Fig. 4, we propose a simple trajectory modification method to deal with this "planning-on-the-fly" problem. Suppose the replanned trajectory is $s_r = [x_r^{[1]}, x_r^{[2]}; \dots; x_r^{[M]}]$, the robot's current position is x, and the robot's velocity is $\dot{x}$. The estimated robot position in the next time cycle is $\hat{x} = x + \dot{x}T_p$, where T_p is the low level controller cycle time. Then, we neglect the first point of the new trajectory and take the second point as a reference. The new desired point $\hat{x}_r^{[1]}$ is obtained by

$$\hat{x}_r^{[1]} = \hat{x} + \rho(x_r^{[2]} - \hat{x}) \tag{24}$$

where $\rho \geq 0$ is a smoothing factor. After the trajectory modification, the new reference trajectory $\hat{s}_r = [\hat{x}_r^{[1]}; x_r^{[2]}; \dots; x_r^{[M]}]$ is interpolated into $s_d = [x_d^{[1]}; x_d^{[2]}; \dots; x_d^{[M_d]}]$ and then sent to the short term planner. This trajectory modification method is efficient in practice. It improves the motion smoothness significantly when the replanning is running in a low frequency.

7. HLSTS hypothesis

Based on the hierarchical system architecture and constrained multi-objective optimization, we anticipate that the long term safety planner and short term safety planner both can benefit each other and a hierarchical coordination of both long and short term planners will further improve the overall performance of the hierarchical system. Here we propose three main hypothesis, which will be verified in the experiments to be discussed in the following sections:

Hypothesis 1 (*Long Term Planner Improves Efficiency*)**.** Long terms safety planner can find a global optimal reference trajectory compared to the short term planner, which largely optimize the total operation time by reducing the livelock and deadlock situations.

Hypothesis 2 (*Short Term Planner Guarantees Safe Operation*)**.** Short term safety planner can always drag the robot state to the safe set while considering the obstacle uncertainty, which guarantees the provably safe operation in both simulation and real world application.

Hypothesis 3 (*Coordination of Long and Short Term Planner Increases the Overall Smoothness, Efficiency and Safety*)**.** A coordination mechanism can compensate the latency during the communication between long and short term safety planner, which reduces the undesired zig-zag motion and further increases the overall smoothness, efficiency and safety.

8. Results and discussion

8.1. Evaluation platforms

To evaluate the performance of the proposed HLSTS framework as shown in Fig. 3, a series of decentralized robot manipulation experiments in clustered environments have been conducted, where the robot have no access to the ground truth obstacle movement patterns. The decentralized manufacturing system increases both flexibility and robustness by maintaining the level of productivity [49]. For example, human–robot collaboration system can accomplish non-repetitive

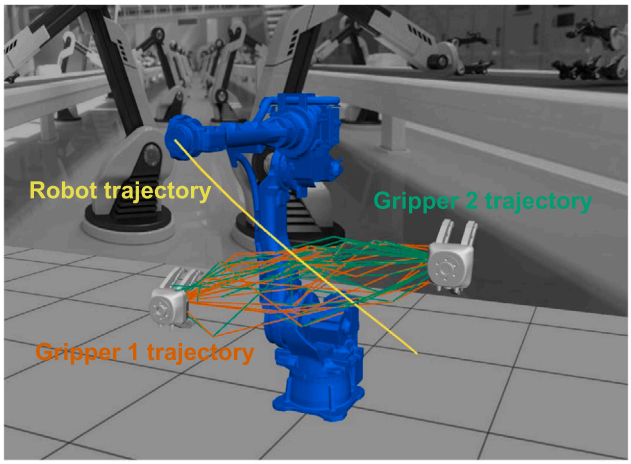


Fig. 5. A robot factory coordination evaluation platform. The robot is performing a goal reaching task and two gripper obstacles are driven along several random trajectories.

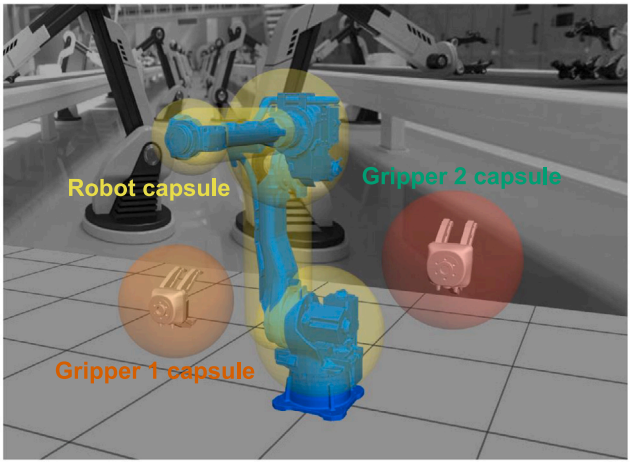


Fig. 6. Robot and obstacle capsules. Several capsules are used to wrap the robot mesh and obstacle mesh for distance calculation.

tasks with human's adaptive and cognitive capability [50], where the robot and human coexist in the same cell and the robot should safely and efficiently complete its tasks without full access of human's intentions [51]. This section introduces the evaluation tasks that we developed.

Robot factory coordination simulation platform

The experiments are first conducted on a simulation platform with a robot factory coordination scenario. A 6-DOF serial robot YASKAWA Motoman GP50 is used in these experiments to perform a goal reaching task as the yellow line shown in Fig. 5. The initial pose of the robot is $\boldsymbol{x}_0 = [0,0,0,0,0,0]^\top$ and the target pose of the robot is $\boldsymbol{x}_{goal} = [\frac{\pi}{4}, -\frac{\pi}{2}, \frac{\pi}{4}, 0, 0, 0]^\top$. Meanwhile, other robots or tools are executing their own tasks, i.e., their end-effectors are moving along specific trajectories and interfering with the GP50 robot in the middle of the goal reaching task. To demonstrate generalizability, we simulate a two-tool scenario and a one-tool scenario and randomized 20 tool trajectories as the orange lines and green lines shown in Fig. 5. These tool trajectories are composed of several line segments. For the one-tool scenario, the tool trajectories are defined by four segment points, where the start point is fixed and the rest points are randomly sampled from a uniform distribution. The segment points in (x, y, z) are

$$\left\{ \begin{bmatrix} 1 \\ 0.1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0.4 \\ 1.1 \end{bmatrix} + 0.2\mathcal{U}, \begin{bmatrix} 1 \\ 0.8 \\ 1.2 \end{bmatrix} + 0.2\mathcal{U}, \begin{bmatrix} 1 \\ 1.2 \\ 1.3 \end{bmatrix} + 0.1\mathcal{U} \right\},$$

where the $\mathcal{U}$ means the uniform distribution from -1 to 1.

Similar to one-tool scenario, for the two-tool scenario, where the trajectories are also defined by four randomly sampled segment points. The tool 1 trajectories segment points in (x, y, z) are

$$\left\{ \begin{bmatrix} 1.1 \\ -0.2 \\ 0.9 \end{bmatrix}, \begin{bmatrix} 1 \\ 0.2 \\ 1.0 \end{bmatrix} + 0.2\mathcal{U}, \begin{bmatrix} 1.1 \\ 0.6 \\ 1.1 \end{bmatrix} + 0.2\mathcal{U}, \begin{bmatrix} 1 \\ 1.0 \\ 1.2 \end{bmatrix} + 0.1\mathcal{U} \right\},$$

and the tool 2 trajectories segment points in (x, y, z) are

$$\left\{ \begin{bmatrix} 1.1 \\ 1.0 \\ 1.2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0.6 \\ 1.1 \end{bmatrix} + 0.2\mathcal{U}, \begin{bmatrix} 1.1 \\ 0.2 \\ 1.0 \end{bmatrix} + 0.2\mathcal{U}, \begin{bmatrix} 1 \\ -0.2 \\ 0.9 \end{bmatrix} + 0.1\mathcal{U} \right\}.$$

The tools are moving along the trajectories with a constant speed 0.03 m/s. In order to calculate the distance between the robot and the obstacles efficiently, several capsules are used to wrap the robot and the obstacles as illustrated in Fig. 6. Thus, the distance calculation between mesh surfaces can be simplified to the distance calculation between line segments, which increase computation efficiency significantly [52].

To investigate the impact of different uncertainty levels on the efficiency and safety performance of HLSTS framework, we design different experiment sets where the robot is supposed to perform the goal reaching task while keeping the obstacle collision avoidance.

Without the access to the ground truth obstacle movement trajectory, the obstacle position estimation is becoming more inaccurate as the time step growing, which will result in accumulated obstacle uncertainties. Therefore, we investigate the performance of the HLSTS framework under different prediction qualities on the obstacle trajectory, including (1) no prediction, (2) local prediction, and (3) global prediction. No prediction assumes obstacle is staying static, which greatly differentiates from ground truth obstacle movement, thus representing large obstacle uncertainty. Local prediction makes inaccurate obstacle movement prediction, which mildly differentiates from ground truth obstacle movement, thus representing mediocre obstacle uncertainty. Finally, global prediction makes accurate obstacle movement prediction, thus representing near-zero obstacle uncertainty. Therefore, we have designed the following experiment cases:

1. *LT no predict*, Long Term planning only, without any prediction on the obstacles.
2. *LT local predict*, Long Term planning only, with inaccurate prediction on the obstacles.
3. *LT global predict*, Long Term planning only, with accurate prediction on the obstacles.
4. *ST*, Short Term planning only, where the current obstacle position is fed to the short term planner. Short term planner is a reactive planner and no prediction is needed.
5. *LSTS no predict*, Hierarchical Long Short Term Safety planning, without any prediction on the obstacles.
6. *LSTS local predict*, Hierarchical Long Short Term Safety planning, with inaccurate prediction on the obstacles.
7. *LSTS global predict*, Hierarchical Long Short Term Safety planning, with accurate prediction on the obstacles.

To perform a fair comparison in time efficiency between these experimental cases, the robot is driven with a maximum joint velocity 0.05 rad/s. The Velocity-SSA in (12) is used in the short term planner for these cases and the coefficient for SSA controller is set as $k_s = 10$. To ensure the provable safety for the short term safety planner under the uncertainties (tracking and measurement, etc.), we choose a

appropriate safe margin d_m and gain k_s for SSA controller, such that the requirements in Theorem 2 are met. To evaluate the efficiency and safety of different cases, we chose the goal reaching time t_f and minimum distance d_{min} as performance indexes.

Real-robot platform

The real robot experiments are carried out on a real 6-DOF YASKAWA Motoman GP50 robot as shown in Fig. 7. The GP50 robot offers outstanding performance for increased production output with high mounting flexibility for a large range of applications.[2] The maximum horizontal reach of GP50 robot is 2.061 m and the maximum vertical reach of GP50 is 3.578 m. To test the performance of the proposed HLSTS framework on real robot, we design a simple obstacle avoidance case using a virtual obstacle. The virtual sphere obstacle with radius 0.3 m is driven from the initial position $(1.0, 0.3, 1.0)$ m with constant speed as the yellow circle shown in Fig. 7. To study the obstacle avoidance ability of the robot system, different obstacle moving speed are chosen as follows

- $v_O = (0, 0.0005, 0.004)$ m/s,
- $v_O = (0, 0.001, 0.008)$ m/s,
- $v_O = (0, 0.005, 0.04)$ m/s.

The joint velocity limit for the real robot is 0.025 rad/s. The Velocity-SSA in (12) is used in the short term planner and the coefficient for SSA controller is set as $k_s = 4$. It is tested in pre-experiments that this choice of k_s guarantees the probabilistic safety in real-time. And the capsule design is the same as in the simulation setup. The overall HLSTS system workflow is the same as in Fig. 3. The long term planner is running on a Matlab-ROS program in host-PC with about 10 Hz frequency. And the short term planner is running on a ROS C++ node in target-PC with 40 Hz frequency. The joint velocity command output from the short term planner is multiplied by cycle time 0.025 s as the joint position increment, since the YRC1000 controller only takes in the joint position command. The control command is sent to the robot through MotoROS application running on the Motoman YRC1000 controller, where the desired position is tracked in the servo control level. The communication latency between the host-PC and the YRC1000 controller is about 50 ms according to preliminary experiments. It is noted that in our experiments, the host-PC and target-PC are the same laptop with Intel Core i7-6700HQ @2.60 GHz CPU. The results is still satisfying even under such limited computational resources, while one can set up the host-PC and target-PC separately to acquire better computational ability.

8.2. Experiment results

Simulation experiments results

Different cases have been carried out in the one-obstacle scenario and the two-obstacle scenario. The goal reaching time t_f – minimum distance d_{min} graph for the one-obstacle scenario and the two-obstacle scenario are shown in Fig. 8(a) and Fig. 8(b), respectively. Each data point is a experiment for corresponding case with respect to certain randomized obstacle trajectory. And the Kernel distribution is used to illustrate the general spreading of the data sets. We could observe the distribution behavior more intuitively in the goal reaching time t_f and minimum distance d_{min} for different cases. Meanwhile, the mean, standard deviation and correlation coefficient values for all experiments are exhibited in Table 1. Besides, the running process for two-obstacle experiments are plotted in Figs. 9 and 15–20 for detailed analyses.

[2] https://www.motoman.com/en-us/products/robots/industrial/assembly-handling/gp-series/gp50

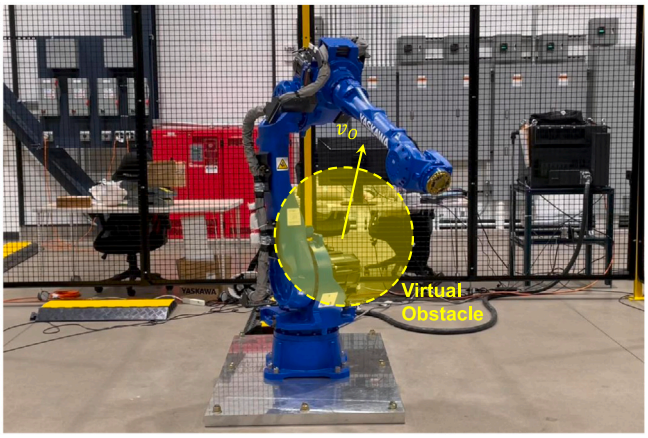


Fig. 7. Real robot experiment platform. The robot is performing a goal reaching task as in the simulation and the virtual obstacle (yellow sphere) is driven from the initial position with constant speed.

One-obstacle results. From the one-obstacle results in Fig. 8(a), we could observe that different cases have a conspicuous difference on minimum distance d_{min} while having an inconspicuous difference on goal reaching time t_f.

For the minimum distance d_{min}, the cases with short term planner, i.e., ST and LSTS, have a larger minimum distance ($d_{min}^{mean} \geq 0.0436$ m), which means the better safety performance. The LT cases have smaller minimum distance ($d_{min}^{mean} \leq 0.0265$ m), however, with the better prediction on the obstacle movement, the safety performance increase significantly ($d_{min}^{mean} = -0.0119$ m for LT no predict, $d_{min}^{mean} = 0.0035$ m for LT local predict and $d_{min}^{mean} = 0.0265$ m for LT global predict). Most data points of LT no predict case have negative minimum distance, while only a few data points of LT global predict case have negative minimum distance. From this result we can conclude that the better prediction on the obstacle movement leads to the better safety, but the best way to ensure safety is using the high-frequency short term planner. Note the goal reaching time of LSTS is only 1 s longer than LT on average, which indicates that LSTS does not generate a more time-consuming trajectory. There are two underlying reasons for this, (1) LT cases reach the target efficiently by sacrificing safety too much, i.e., the average d_{min} is lower than LSTS cases and the variation is larger than LSTS case as in Fig. 8(a); and (2) LSTS cases only need minor modification of the reference trajectory from LT cases via triggering the ST at critical states. Hence, the goal reaching time of LSTS is slightly longer than LT.

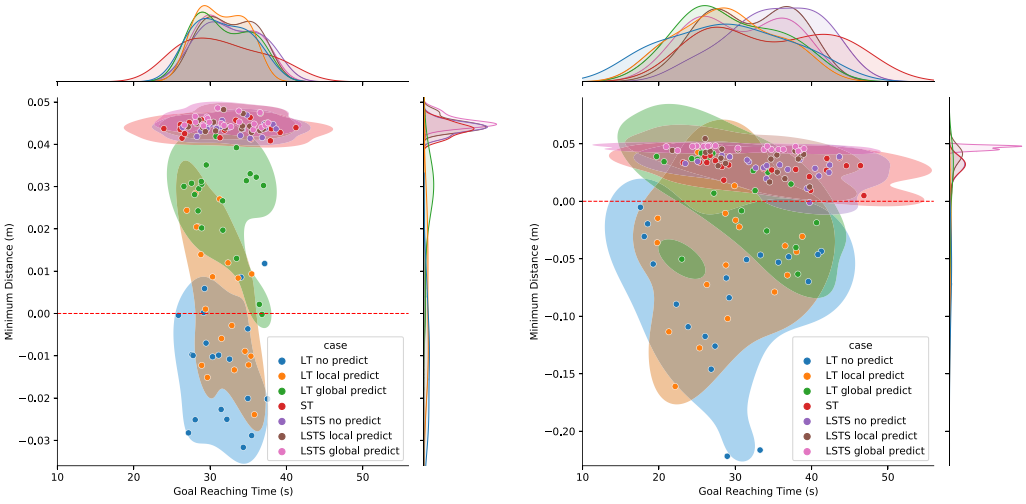
For the goal reaching time t_f, the nominal values of the distribution of different cases are almost the same ($t_f^{mean} \approx 32$ s) while the ST case has larger distribution variance ($t_f^{std} = 5.0889$ s) than the others ($t_f^{std} \leq 3.7314$ s). The reason for this result is that the one-obstacle scenario is too simple to demonstrate the efficiency merits for the long term planner. The robot trajectory from long term planner and the trajectory from short term planner are quite similar in most cases. However, the long term planner tends to find a more time-optimal trajectory in complicated scenario. Thus, we further carried out the two-obstacle experiments.

Two-obstacle results. The two-obstacle results in Fig. 8(b) show obvious difference on both minimum distance d_{min} and goal reaching time t_f.

The minimum distance d_{min} results have similar pattern as the one-obstacle results, i.e., the cases with short term planner have better safety performance ($d_{min} \geq 0.0288$ m for cases with short term planner and $d_{min} \leq 0.0115$ m for cases without short term planner). Besides, the safety performance increases as the prediction accuracy increases. This trend is observed in both LT cases ($d_{min}^{mean} = -0.0823$ m for LT no predict, $d_{min}^{mean} = -0.0470$ m for LT local predict and $d_{min}^{mean} = 0.0115$ m for LT global predict) and LSTS cases ($d_{min}^{mean} = 0.0288$ m for LSTS no

Table 1
Experimental results for one-obstacle and two-obstacle scenarios.

	One-obstacle experiments					Two-obstacle experiments				
	t_f (s)		d_{min} (m)		corr. coef.	t_f (s)		d_{min} (m)		corr. coef.
	mean	std.	mean	std.		mean	std.	mean	std.	
LT no predict	31.5207	3.4869	−0.0119	0.0130	−0.0455	28.9548	7.5948	−0.0823	0.0590	−0.0830
LT local predict	31.4885	2.8746	0.0035	0.0159	−0.5842	29.0144	6.0028	−0.0470	0.0540	0.1669
LT global predict	31.6287	3.4862	0.0265	0.0109	−0.3540	29.4750	6.1475	0.0115	0.0342	−0.5705
ST	31.5999	5.0889	0.0436	0.0016	−0.1119	34.6100	8.0315	0.0289	0.0092	−0.4128
LSTS no predict	32.6966	3.7314	0.0439	0.0014	−0.0772	35.1973	5.8271	0.0288	0.0099	−0.3881
LSTS local predict	32.4600	2.8746	0.0447	0.0159	0.0107	31.9381	5.5963	0.0368	0.0109	−0.5194
LSTS global predict	32.2146	3.4675	0.0454	0.0014	−0.1160	30.7253	6.0510	0.0465	0.0017	−0.5412



(a) One-obstacle results (b) Two-obstacle results

Fig. 8. Goal reaching time t_f – minimum distance d_{min} graph for experimental results. (Each data point represents a randomized obstacle trajectory, and the colored area represents the Kernel distribution for the data set).

predict, $d_{min}^{mean} = 0.0368$ m for LSTS local predict and $d_{min}^{mean} = 0.0465$ m for LSTS global predict) for two-obstacle results while it is not obvious for LSTS cases in one-obstacle results ($d_{min}^{mean} = 0.0439$ m for LSTS no predict, $d_{min}^{mean} = 0.0447$ m for LSTS local predict and $d_{min}^{mean} = 0.0454$ m for LSTS global predict). The reason is that the two-obstacle experiment is more complicated than the one-obstacle experiment. A bad prediction on the obstacle movement will cause the SSA control runs into a severe condition (e.g., two obstacle come from both side and block the path) that a large gain k_s is needed to avoid the collision. However, we cannot set the infinite value for k_s. Thus there are some unsafe cases even there is a short term planner.

Unlike the one-obstacle results, the two-obstacle results have more apparent difference on the goal reaching time t_f. The nominal value of ST case ($t_f^{mean} = 34.6100$ s) and the LSTS no predict case ($t_f^{mean} = 35.1973$ s) is larger than the others ($t_f^{mean} \leq 31.9381$ s). However, the ST case has larger distribution variance ($t_f^{std} = 8.0315$ s) than the LSTS no predict case ($t_f^{std} = 5.8271$ s). And from the distribution plots on the top of Fig. 8(b) we can see that, the ST case has a series of experiment sets those goal reaching time t_f are larger than the others. These results indicate that the ST case is less efficient than the others. The reason for this phenomenon can be explained through the ST case process plot in Fig. 9. It is observed from 8 s to 38 s in Fig. 9 that, some ST cases runs into a situation that one of the obstacle always blocks the robot on the one side. The short term planner can only locally generates a trajectory free from collision while stay at one side of the obstacle. Thus for a long time the robot is moving but the distance to the goal is not decreasing. For such livelock situation, the long term planner could generate a trajectory around the obstacle and get away with the local minima. The more complicated the environment is, the more efficient will the long term planner works.

Joint velocity comparison

The joint velocity profiles for different cases are also compared as in Fig. 10. It is observed that, the velocity profile for the ST case has chattering phenomenon, which is due to the high frequency interaction between the robot and the obstacle. Without the short term planner, the LT only cases (LT no predict, LT local predict and LT global predict) have the smoothest joint velocity profile as illustrated in the top of Fig. 10. And the long term planner benefits the joint velocity smoothness significantly. Compared with the ST cases, the LSTS cases (LSTS no predict, LSTS local predict and LSTS global predict) have smoother joint velocity profile as illustrated in the bottom of Fig. 10.

Besides, for both LT cases and LSTS cases, the local predict cases have smoother joint velocity than no predict cases, and the global predict cases have smoother joint velocity than local predict cases. These results can be explain through two aspects:

1. The reason for LT cases is that as the prediction accuracy increases, the numbers of the replanning decrease. This phenomenon can be observed from Figs. 15–17, where the numbers of replanning decreases from 12 to 5. The replanning is requested when a potential collision is detected. Each replanned trajectory will driven the robot away from current trajectory to avoid the collision. Thus successional replanning keeps changing the robot moving direction and thus causes the unsmooth motion.
2. The reason for LSTS cases is that as the prediction accuracy increases, the time of the short term planner activated decrease. This phenomenon can be observed from Figs. 18–20, where the green SSA activate area decreases. The less time of short term planner activated also means the less numbers of replanning,

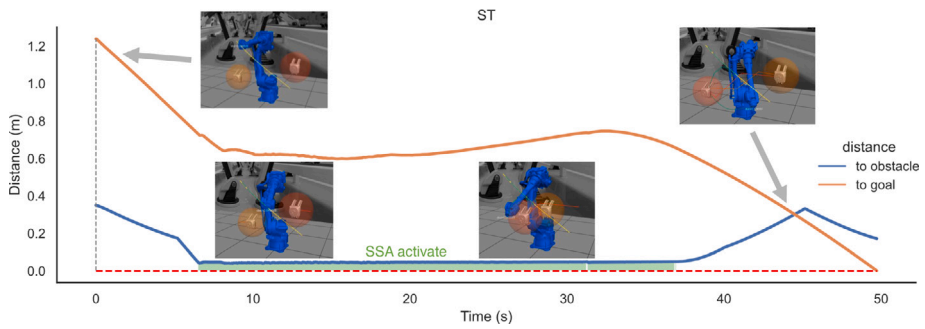


Fig. 9. Two-obstacle ST process. (The blue solid line is the distance from the robot to the obstacle. The orange solid line is the distance from the robot end-effector to the goal. The red dashed line is the safety line with zero distance. The gray dashed lines represent the replanning time. The green zone means the SSA control in short term planner is activated).

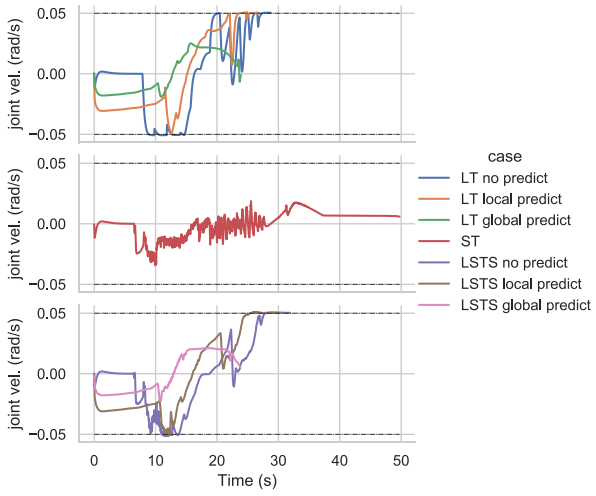


Fig. 10. Joint velocity compare results for different cases under same obstacle trajectory. (The dashed lines on ± 0.05 represent the velocity limit. Only joint 2 velocity profile is plotted for distinctness).

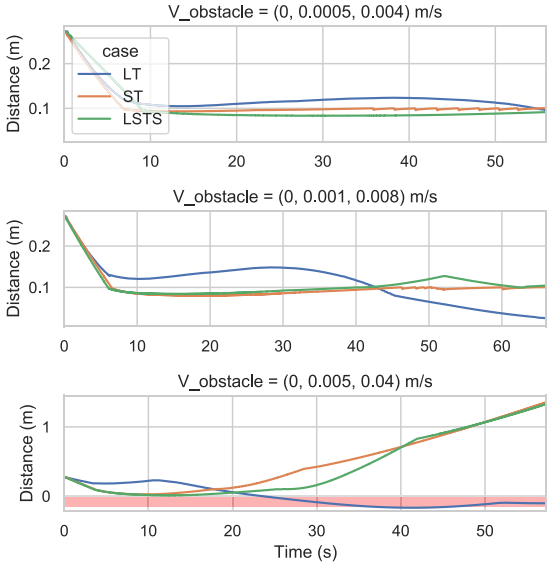


Fig. 11. Obstacle distance results for real robot experiments with different obstacle moving speed. (The area colored in red represents unsafe distance).

which is determined by our system workflow as in Fig. 3. Besides, the SSA control law (12) in the short term planner will cause chattering without the appropriate gain k_s. With the prior-knowledge of the obstacle, the accurate prediction decreases the time of short term planner activated, thus the whole running process is smooth.

Thus the more accurate the prediction is, the smoother will the joint velocity profile be. The above results consistently verify Hypotheses 1 and 2.

Real-robot experiments results

Different obstacle speed are tested and the obstacle distance profile for different cases are shown in Fig. 11. It is observed that for the low obstacle speed case $v_O = (0, 0.0005, 0.004)$ m/s at the top of Fig. 11, all three cases have similar safety performance with the minimum distance approximate 0.1 m. For the medium obstacle speed case $v_O = (0, 0.001, 0.008)$ m/s at the middle of Fig. 11, the LSTS case and ST case have similar safety performance with the minimum distance approximate 0.08 m, while the LT case has worst safety performance where the minimum distance $d_{min} = 0.018$ m. For the high obstacle speed case $v_O = (0, 0.005, 0.04)$ m/s at the bottom of Fig. 11, the LT case is unsafe with $d_{min} = -0.17$ m. The LSTS case and ST case are still safe, but due to the obstacle avoidance ability limitation brought by the robot joint velocity limits, the minimum distance is quite close to 0. These results demonstrate that the proposed HLSTS system can maintain a good safety performance in real robot application.

The whole process for the LSTS experiment with high obstacle speed is shown in Fig. 12. The long term planner starts with a trajectory circumventing the obstacle from its top. As the obstacle moving upwards, the short term planner pushing the robot upward to guarantee the safety. Meanwhile, the long term planner keeps replanning efficient trajectory to avoid the collision. The average replanning time is 0.4728 s as the dashed lines shown in Fig. 12. After a few seconds (at about 15 s), the long term planner find a replanned trajectory with better time efficiency, which circumventing the obstacle from its bottom. Then, more replanning corrects the trajectory and leads the robot to the goal. The whole process shows the coordination between the long term planner and the short term planner. The resulting obstacle avoidance trajectory is efficient to our human knowledge.

Hierarchical coordinator comparison

In Section 6.2, we proposed a hierarchical coordination method through replanned trajectory modification, which improve the coordination between the long term planner and short term planner due to the replanning computation time and the communication latency. In order to illustrate the effectiveness of the proposed hierarchical coordinator, we compare two tasks with coordination on and off using LT no predict case under the same obstacle trajectory. The comparison result is shown in Fig. 13.

It is seen from the robot end-effector trajectory on the top of Fig. 13 that, without coordination, the robot is running with a zig-zag

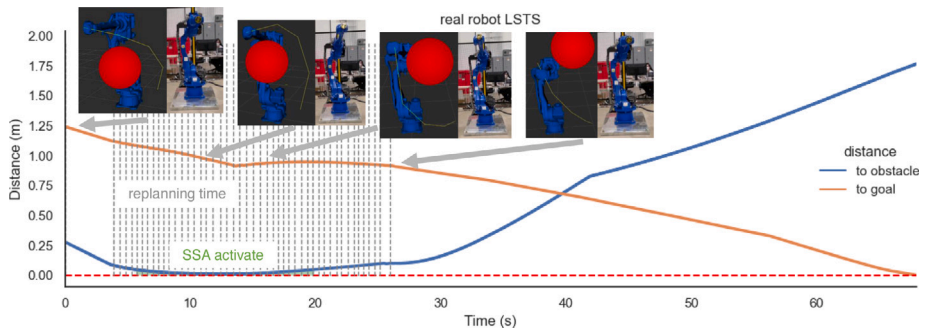


Fig. 12. LSTS process for real robot experiments with obstacle moving speed $v_O = (0, 0.005, 0.04)$ m/s. (The blue solid line is the distance from the robot to the obstacle. The orange solid line is the distance from the robot end-effector to the goal. The red dashed line is the safety line with zero distance. The gray dashed lines represent the replanning time. The green zone means the SSA control in short term planner is activated).

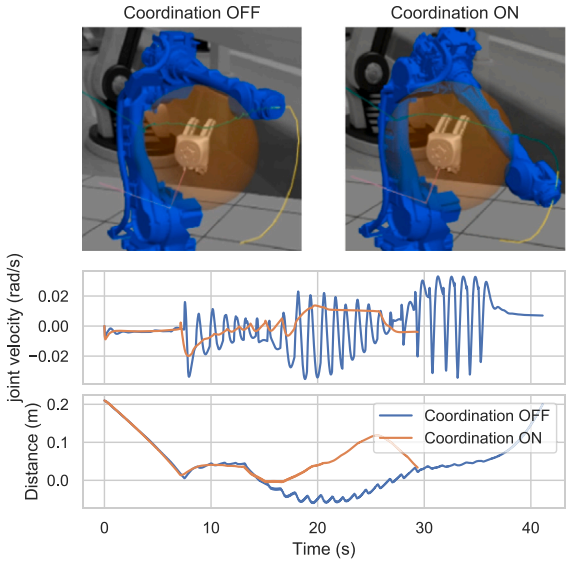


Fig. 13. Coordination compare results under the same obstacle trajectory. (The green curve is the robot end-effector trajectory and the red curve is the reference trajectory. Only joint 2 velocity profile is plotted for distinctness).

motion. While the robot is running smoothly with coordination. This phenomenon can also be illustrated through the joint velocity profile in the middle of Fig. 13. Without coordination, the joint velocity is reversed after receiving a replanned trajectory, which not only lead to the zig-zag motion, but also increase the goal reaching time. The coordination prevents the velocity reverse and lead to smooth motion as well as efficiency improvement.

Besides, the obstacle distance profile at the bottom of Fig. 13 indicates that the coordination also improves the safety. Without coordination, the first point of replanned trajectory is an unsafe point short time ago. The robot is commanded to track this unsafe point thus the safety is violated. With coordination, the modified first point leverage the safe position of the second point of replanned trajectory. Hence the robot is tracking a relatively safe point and the safety performance is better than the uncoordinated case. These results illustrate the effectiveness of the proposed hierarchical coordinator and also verify Hypothesis 3.

Safety index comparison in SSA

To provide a guidance on the safety index choose, two different types of safety indexes used in (8) and (13) are also compared. For the Velocity-SSA, the controller output is joint velocity and the control law is used as in (12). For the Acceleration-SSA, the controller output is joint acceleration and the control law is used as in (17). Different joint hardware interfaces and tracking controllers are implemented accordingly. They are both running on a same obstacle avoidance task, i.e., same obstacle trajectory and moving speed. The parameters used in Velocity-SSA and Acceleration-SSA are

- Velocity-SSA: $k_s = 10$
- Acceleration-SSA: $k_s = 30$, $k_1 = 0.02$

To perform a fair comparison, parameters for Velocity-SSA and Acceleration-SSA are carefully tuned so that the resulting obstacle distance for them are basically the same. Besides, the joint velocity limits are the same 0.05 rad/s. It is observed that the increase on k_1 will lead to the decrease on minimum distance. Thus a larger k_s is used in Acceleration-SSA while the similar minimum distance is obtained. Due to the complexity of the distance function D, we can only calculate $\frac{\partial \phi_0}{\partial \boldsymbol{x}}$ and $\frac{\partial^2 \phi_0}{\partial \boldsymbol{x}^2}$ in discrete form. In (15), we have

$$\begin{aligned} \frac{\partial \phi_0}{\partial \boldsymbol{x}} &= [\frac{\partial \phi_0}{\partial x_1}, \dots, \frac{\partial \phi_0}{\partial x_N}]^\top \\ \frac{\partial^2 \phi_0}{\partial \boldsymbol{x}^2} &= \begin{bmatrix} \frac{\partial^2 \phi_0}{\partial x_1^2} & \cdots & \frac{\partial^2 \phi_0}{\partial x_1 \partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \phi_0}{\partial x_N \partial x_1} & \cdots & \frac{\partial^2 \phi_0}{\partial x_N^2} \end{bmatrix} \end{aligned} \tag{25}$$

Using central difference method, the discrete form can be obtained as

$$\begin{aligned} \frac{\partial \phi_0}{\partial x_i} &\approx \frac{\phi_0(x_i + h) - \phi_0(x_i - h)}{2h} \\ \frac{\partial^2 \phi_0}{\partial x_i^2} &\approx \frac{\phi_0(x_i + h) - 2\phi_0(x_i) + \phi_0(x_i - h)}{h^2} \\ \frac{\partial^2 \phi_0}{\partial x_i \partial x_j} &\approx \frac{\delta_1 - \delta_2 - \delta_3 + \delta_4}{4h^2} \\ \delta_1 &= \phi_0(x_i + h, x_j + h) \\ \delta_2 &= \phi_0(x_i - h, x_j + h) \\ \delta_3 &= \phi_0(x_i + h, x_j - h) \\ \delta_4 &= \phi_0(x_i - h, x_j - h) \end{aligned} \tag{26}$$

where $h > 0$ is a small positive constant, $i, j = 1, \dots, N$ and $i \neq j$.

The resulting obstacle distance profile and the joint velocity profile are plotted in Fig. 14. The top figure of Fig. 14 indicates that the obstacle distance profile for them are basically the same ($d_{min} = 0.0426$ m for Velocity-SSA and $d_{min} = 0.0439$ m for Acceleration-SSA). From the joint velocity profile we can see that the Acceleration-SSA is much more smoother than the Velocity-SSA. One reason is that the system dynamics for Acceleration-SSA is running on the joint acceleration

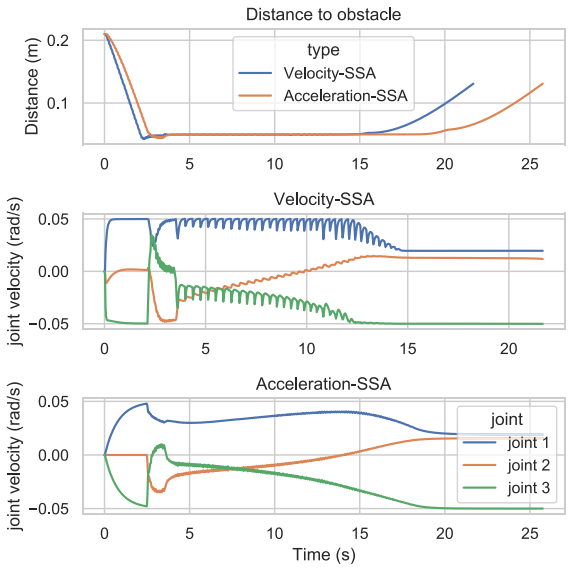


Fig. 14. Velocity-SSA and Acceleration-SSA safety index comparison results. (Joint 4–6 velocity profiles are not plotted due to small movement).

layer, where the joint velocity is the integral of the control command. Note that the smooth joint velocity profile from Acceleration-SSA could be achieved by applying a filter to Velocity-SSA. However, in our experiments, we choose Acceleration-SSA instead of filtered Velocity-SSA due to (1) Acceleration-SSA achieves smooth joint velocity profile while providing provable safety guarantee; and (2) simply filtering Velocity-SSA will compromise the zero safety violation guarantee brought by SSA. Another reason is the penalization on the time derivative of the distance D in the safety index (13). By tuning the parameter k_1 we can tune the smoothness of the joint velocity profile.

8.3. Hypothesis discussion

Hypothesis 1 - long term planner improves the efficiency of short term planner

As the two-obstacle results analyzed in Section 8.2, the average goal reaching time t_f is smaller for cases with long term planner (LSTS and LT cases) compared with the cases without long term planner (ST case). Notice that when the joint velocity of robot is bounded, the smaller goal reaching time means the robot can complete its tasks more efficiently. We also observed that when equipped the long term planner, the livelock situation in Fig. 9 is avoided (**Hypothesis 1**).

Due to the complexity nature of global optimal maneuver trajectory for the robot to reach the target while dodging the obstacle, the short term planner only focuses on the locally reactive trajectory modification, where the local trajectory construction is hard to satisfy the global trajectory objective (smoothness) when starting from a reference trajectory that is a bad local trajectory optima (linear interpolation). On the other hand, the idea of long term planner is solving the optimization problem avoiding the obstacle, minimizing the overall trajectory length, and ensuring trajectory smoothness. Therefore, the trajectory generated by long term planner inherently minimize the undesired behaviors such as livelocks that make the trajectory longer and less smooth. At the same time we also observe that, as the obstacle prediction accuracy increases in the long term planner, the average goal reaching time t_f decreases accordingly. Since better obstacle prediction accuracy enables better obstacle avoidance constraints satisfaction for the reference trajectory, which further improves the reference trajectory optimality and yields better efficiency.

Hypothesis 2 - short term planner guarantees safe operation

From the simulation results in Section 8.2 and real-robot results in , the minimum distance d_{min} between the robot and the obstacle is significantly larger for cases with short term planner (LSTS and ST cases) compared with cases without short term planner (LT cases).

Considering the uncertainty of obstacle position prediction, tracking error of reference trajectory, as well as the limited horizon of reference trajectory from long term planner, the robot state may not constantly remain the safe set during the reference trajectory tracking. However, the short term planner explicitly consider the uncertainty of the obstacle, and pull the robot into safe set whenever it is unsafe, which further guarantees the safe operation during reference trajectory tracking (**Hypothesis 2**).

Hypothesis 3 - coordination of long and short term planner increases the overall smoothness, efficiency and safety

As the results shown in , with the hierarchical coordinator turned on, the goal reaching time t_f is decreased, the average distance between the robot and the obstacle is increased, and the robot motion smoothness is increased significantly.

Due to the execution frequency difference between the long term planner and short term planner, the reference start position of the robot is the snapshot of the robot state a few times ago when replanning is requested, whereas the shorter term planner makes the current robot state closer to the target and more safe. Without a proper coordination mechanism to modify the reference start position after replanning, the robot is commanded to track this unsafe point thus the safety will be violated. At the same time, tracking a point that has already been passed will reverse the velocity of the robot, which leads to the zig-zag motion as well as the increasing the goal reaching time. With a proper designed coordinator, we can prevents the zig-zag motion which further leads to better efficiency and safety (**Hypothesis 3**).

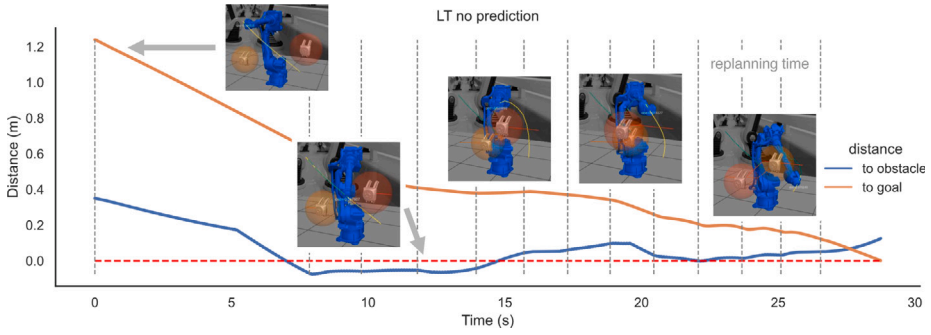


Fig. 15. Two-obstacle LT process without prediction. (The blue solid line is the distance from the robot to the obstacle. The orange solid line is the distance from the robot end-effector to the goal. The red dashed line is the safety line with zero distance. The gray dashed lines represent the replanning time).

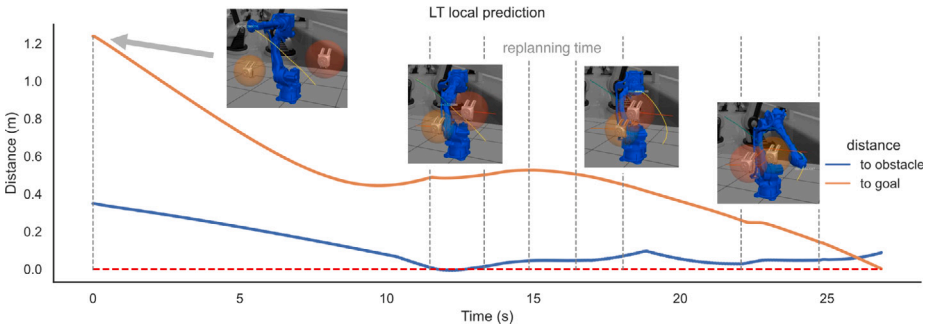


Fig. 16. Two-obstacle LT process with local prediction. (The blue solid line is the distance from the robot to the obstacle. The orange solid line is the distance from the robot end-effector to the goal. The red dashed line is the safety line with zero distance. The gray dashed lines represent the replanning time).

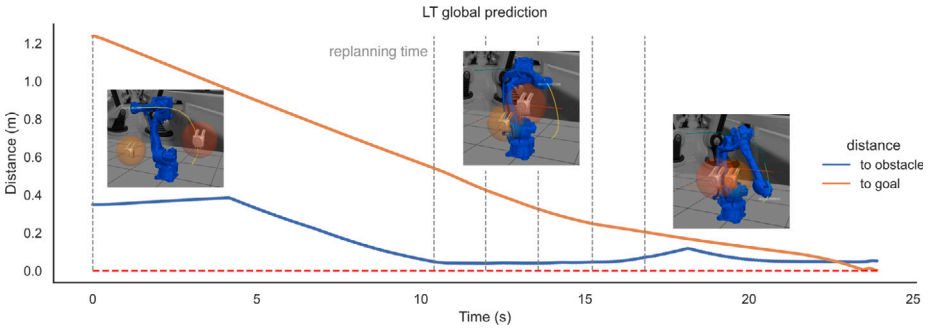


Fig. 17. Two-obstacle LT process with global prediction. (The blue solid line is the distance from the robot to the obstacle. The orange solid line is the distance from the robot end-effector to the goal. The red dashed line is the safety line with zero distance. The gray dashed lines represent the replanning time).

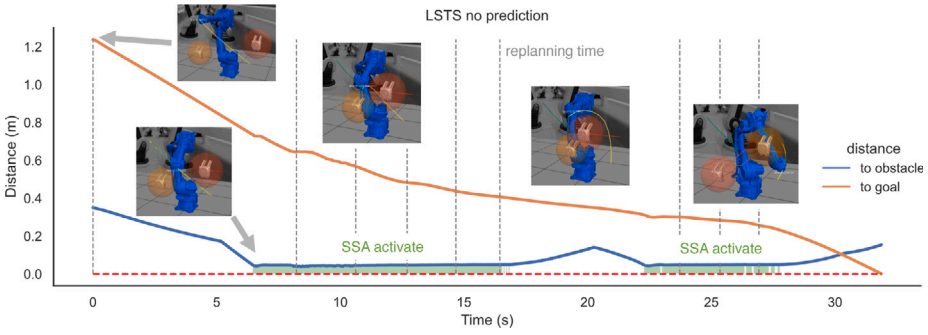


Fig. 18. Two-obstacle LSTS process without prediction. (The blue solid line is the distance from the robot to the obstacle. The orange solid line is the distance from the robot end-effector to the goal. The red dashed line is the safety line with zero distance. The gray dashed lines represent the replanning time. The green zone means the SSA control in short term planner is activated).

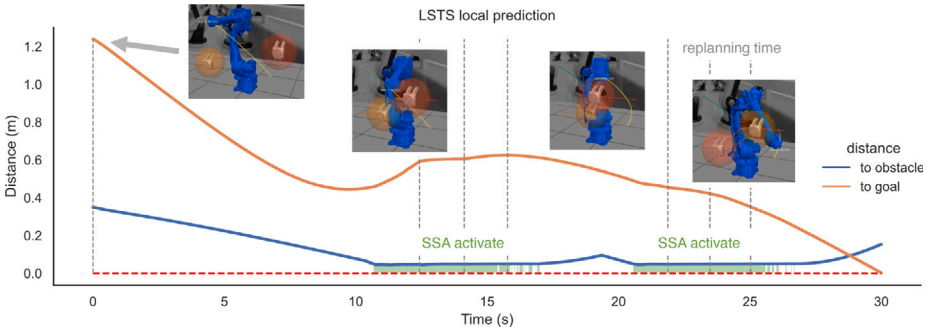


Fig. 19. Two-obstacle LSTS process with local prediction. (The blue solid line is the distance from the robot to the obstacle. The orange solid line is the distance from the robot end-effector to the goal. The red dashed line is the safety line with zero distance. The gray dashed lines represent the replanning time. The green zone means the SSA control in short term planner is activated).

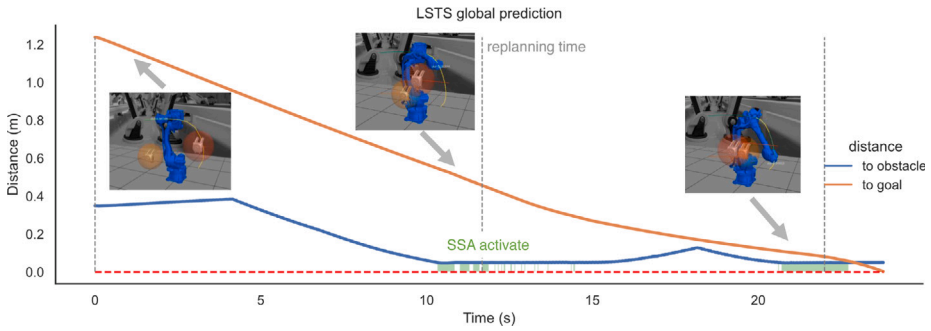


Fig. 20. Two-obstacle LSTS process with global prediction. (The blue solid line is the distance from the robot to the obstacle. The orange solid line is the distance from the robot end-effector to the goal. The red dashed line is the safety line with zero distance. The gray dashed lines represent the replanning time. The green zone means the SSA control in short term planner is activated).

9. Conclusion

The safe and efficient robot manipulation in clustered environment remains challenging due to uncertainties in the system and limited computation resources. This paper proposed a hierarchical long short term safety system (HLSTS), which combines efficiency-oriented long term planner in the upper layer, a safety-oriented short term planner in the lower layer, and a hierarchical coordinator in the middle layer. The long term planner enables the robot to generate global reference trajectory with great optimality. The short term planner guarantees the robot safety by reacting to local environment emergencies in real-time. The coordinator ensures the smooth motion of the robot system by trajectory modification to compensate the communication latency between upper and lower layer. The theoretical results indicate feasibility guarantee of the long term planner as well as the probabilistic safety guarantee of the shot term planner. The proposed architecture has been implemented in our robot factory application on both simulations and real robot experiments, where the robot is performing a goal reaching task while interacting with randomly moving obstacles. Experimental results consistently verify our hypothesis, which indicates that the proposed HLSTS framework not only guarantees the safety but also improves the efficiency.

CRediT authorship contribution statement

Suqin He: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft. **Weiye Zhao:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft. **Chuxiong Hu:** Methodology, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Yu Zhu:** Writing – review & editing. **Changliu Liu:** Conceptualization, Methodology, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.rcim.2022.102522.

Data availability

Data will be made available on request.

Acknowledgment

This work is supported in part by the Beijing Natural Science Foundation under Grant JQ19010, in part by the National Nature Science Foundation of China under Grant 51922059, in part by the National Science Foundation under Grant No. 2144489, in part by the Amazon Research Award, and in part by Subaward No. ARM-TEC-18-01-F-06 from the Advanced Robotics for Manufacturing ("ARM") Institute.

Appendix A. Theoretical analysis

Before we discuss the theoretical analysis for the proposed hierarchical long and short term safety system, we start with definitions for two crucial categories of concepts.

The first category of concepts is trajectory, there are four different definitions summarized as following, and their illustrations are summarized in Fig. A.21:

- **discrete-time trajectory** s_D: The planned trajectory generated by planning algorithm (e.g. convex feasible set algorithm), which includes a sequence of discrete waypoints $\boldsymbol{x}_r^{[i]}$, such that $s_D = [\boldsymbol{x}_r^{[1]}; \boldsymbol{x}_r^{[2]}; \ldots]$. In the following, we use left-superscript j to denote the jth replanned trajectory, e.g., ${}^j s_D$ represents the jth replanned discrete-time trajectory.
- **continuous-time trajectory** s_C: The continuous trajectory that connects all the discrete waypoints from a discrete-time trajectory, such that: $\exists \mathcal{T}_i : [0,1] \to X$ indexed by interval $\mathcal{I} = [0,1]$, such that $\mathcal{T}_i(0) = \boldsymbol{x}_r^{[i]}, \mathcal{T}_i(1) = \boldsymbol{x}_r^{[i+1]}, i = 1, 2, \ldots$. Then $s_C = [\mathcal{T}_1; \mathcal{T}_2; \cdots]$
- **concatenated planning trajectory** s_{Concat}: The continuous trajectory that connects the first waypoint ${}^j\boldsymbol{x}_r^{[1]}$ for a sequence of planned discrete-time trajectory ${}^j s_D, j = 1, 2, \ldots$, such that: $\exists {}^j\mathcal{T}_C : [0,1] \to X$ indexed by interval $\mathcal{I} = [0,1]$, such that ${}^j\mathcal{T}_C(0) = {}^j\boldsymbol{x}_r^{[1]}, {}^j\mathcal{T}_C(1) = {}^{j+1}\boldsymbol{x}_r^{[1]}, j = 1, 2, \ldots$. Then $s_{Concat} = [{}^1\mathcal{T}_C; {}^2\mathcal{T}_C; \cdots]$. And this is the trajectory that is being monitored by the short term controller.
- **executed trajectory** s_{Exec}: The robot executed continuous trajectory.

The second category of concepts is feasibility, there are three different definitions summarized as following:

- **discrete-time feasibility**: every waypoint in discrete-time trajectory s_D satisfies the safety constraints, i.e. $(\forall \boldsymbol{x}_r^{[i]} \in s_D, \forall j \in \mathbb{N}^+), D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) \geq 0$, where $D_j^{[i:i+1]}$ is the safety constraint defined in (20).
- **continuous-time feasibility**: there exists a continuous time trajectory s_C such that every possible point $\boldsymbol{x} \in s_C$ satisfies the safety constraint.

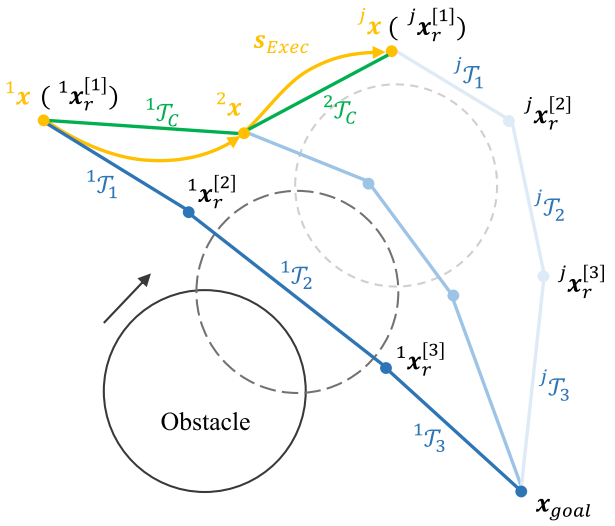


Fig. A.21. Illustration of different types of trajectories. Blue waypoints: discrete-time trajectory s_D. Blue line segments: continuous-time trajectory s_C. Green line segments: concatenated planning trajectory s_{Concat}. Yellow trajectory: executed trajectory s_{Exec}.

- **persistent feasibility**: for each jth replanning step, there always exists a discrete-time trajectory ${}^j s_D$ that satisfies the discrete-time feasibility.

With the clarified two categories of concepts, for the proposed hierarchical long and short term safety system, there are mainly five potential questions regarding its performance:

Q1: If both the delay and tracking error are bounded, and there is no uncertainty, will the planner always generate a feasible discrete-time trajectory plan at every replanning step during execution, i.e., can we ensure persistent feasibility?
Q2: If there is no uncertainty, will there always be a feasible continuous-time trajectory such that every possible point along the trajectory is feasible, i.e. continuous-time feasibility?
Q3: If there is bounded delay, tracking error, and uncertainty, will the safety of the system still be guaranteed?
Q4: If there is bounded delay and uncertainty, will there always be a feasible continuous-time trajectory to track the concatenated planning trajectory s_{Concat} with bounded error during execution?
Q5: Will there be persistent feasibility for the long term planner when there are bounded delay and uncertainty, and if so, will the robot be able to converge to the target position, i.e. $\boldsymbol{x} \to \boldsymbol{x}_{goal}$, safely in finite time?

In this paper, we will answer the first three questions by presenting the following main theorems and the associated assumptions, while the last two questions are left for future work.

The above questions are based on different assumptions regarding delay, tracking error and uncertainties, where (1) the delay is mainly caused by the communication delay between hierarchical layers; (2) the tracking error is mainly caused by system dynamics and modeling errors; (3) the uncertainties are mainly caused by measurement of the robot and prediction of the obstacles.

The first question is answered by Theorem 3 and the second question is answered by Theorem 1. A positive answer to the first question ensures that we will always find a discrete-time long term plan in receding horizon if there is no uncertainty (persistent feasibility guarantee). A positive answer to the second question ensures that the robot can safely track a continuous-time trajectory to the target if there is no delay, tracking error or uncertainty. Note that the dynamic constraints are not considered in long term planning, thus the feasible continuous-time trajectory can only be perfectly tracked by the robot if the low-level controller is perfect, i.e., there is no tracking error regarding the continuous-time trajectory.

The third question is answered by Theorem 2. The result ensures that the system safety can be probabilistically ensured even if the long term plan is not feasible during execution due to prediction error as well as the trajectory tracking error.

The reason why we need to show Theorem 1 and Theorem 3 is that the original proof for CFS algorithm [47] only concerns with discrete-time feasibility in one planning. Discrete-time feasibility in one planning answers what initial reference trajectory can lead to an optimal and feasible discrete-time trajectory through CFS iterations. Theorems 1 and 3 answer continuous time feasibility in one plan as well as persistent feasibility in subsequent plans. Note there are other approaches that plan in discrete time and ensure continuous time feasibility, such as TrajOpt algorithm [53]. However, there are no formal guarantees on convergence for TrajOpt and it remains unclear what initialization can guarantee a solution.

A.1. Assumption and properties

Assumption 1. It is assumed that the infeasible set in the configuration space from time step i to time step $i+1$ can be represented as a set of convex hulls $C_j^{[i:i+1]}, j \in \mathbb{N}^+$ and $j < \infty$, and all the convex hulls have disjoint closures, i.e. $\forall k \neq j, C_j^{[i:i+1]} \bigcap C_k^{[i:i+1]} = \emptyset$.

The Assumption 1 is easy to met in practice. Consider the settings shown in Fig. 6, the infeasible set in the state space for the 6-DOF robot can be wrapped into a set of disjoint convex hulls as shown in Fig. A.22. The Assumption 1 ensures the convexity of the distance function (20). In practice, we can use Joint Tolerance Estimation (JTE) toolbox [54] to find the convex hulls for the infeasible sets by solving the following local positiveness constrained optimization:

$$\min \lambda_j, \text{ s.t. } \forall \lambda_j + \zeta \geq \|\boldsymbol{x} - \boldsymbol{x}^{C_j}\|_\infty > \lambda_j, \boldsymbol{x} \text{ is feasible.} \tag{A.1}$$

where the infeasible convex hull C_j is represented as a hypercube with length λ_j, $\boldsymbol{x}^{C_j}$ is a reference infeasible state sampled in C_j, and $\zeta > 0$ controls the minimum distance between the disjoint infeasible convex hulls. We can also use adversarial optimization [55] to numerically solve the dual problem of (A.1) as following:

$$\max \lambda_j, \text{ s.t. } \exists \lambda_j + \zeta \geq \|\boldsymbol{x} - \boldsymbol{x}^{C_j}\|_\infty > \lambda_j, \boldsymbol{x} \text{ is infeasible.} \tag{A.2}$$

where infeasible convex hull C_j is also represented as a hypercube with length λ_j. Note that (A.2) can be solved efficiently using off-the-shelf nonlinear programming solvers, such as `fmincon` in MATLAB.

Given the distance function definition in (20), $D_j^{[i:i+1]}(\boldsymbol{x})$ is a convex function [48]. Therefore, $D_j^{[i:i+1]}(\boldsymbol{x})$ is Lipschitz continuous and has bounded subgradient everywhere. At the same time, since we are representing $\forall j, C_j$ using polygon-shaped convex hull, which implies the boundary of $\forall j, C_j$ is piece-wise smooth. According to the Theorem 3.1 from [56], the following properties hold:

(P1) $\forall j, D_j^{[i:i+1]}(\boldsymbol{x})$ is Lipschitz continuous in $\boldsymbol{x}$, i.e., $|D_j^{[i:i+1]}(\boldsymbol{x}_1) - D_j^{[i:i+1]}(\boldsymbol{x}_2)| \leq L_1 \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$.
(P2) $\forall j, D_j^{[i:i+1]}(\boldsymbol{x})$ is a L-smooth function in $\boldsymbol{x}$, i.e., $\|\nabla_{\boldsymbol{x}} D_j^{[i:i+1]}(\boldsymbol{x}_1) - \nabla_{\boldsymbol{x}} D_j^{[i:i+1]}(\boldsymbol{x}_2)\| \leq L_2 \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$ almost everywhere.
(P3) $\forall j, \nabla_{\boldsymbol{x}} D_j^{[i:i+1]}$ is bounded, i.e., $\|\nabla_{\boldsymbol{x}} D_j^{[i:i+1]}\| \leq L_3$ almost everywhere.

where $L_1, L_2, L_3 \in \mathbb{R}^+$ are positive constants.

Assumption 2. It is assumed that the distance function (20) at each time step is continuous and smooth with respect to time, i.e.,

(A1) $\forall i, j, \boldsymbol{x}, |D_j^{[i+1:i+2]}(\boldsymbol{x}) - D_j^{[i:i+1]}(\boldsymbol{x})| \leq L_4(t^{[i+1]} - t^{[i]}) = L_4 t_s$.
(A2) $\forall i, j, \boldsymbol{x}, \|\nabla_{\boldsymbol{x}} D_j^{[i+1:i+2]}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} D_j^{[i:i+1]}(\boldsymbol{x})\| \leq L_5(t^{[i+1]} - t^{[i]}) = L_5 t_s$.

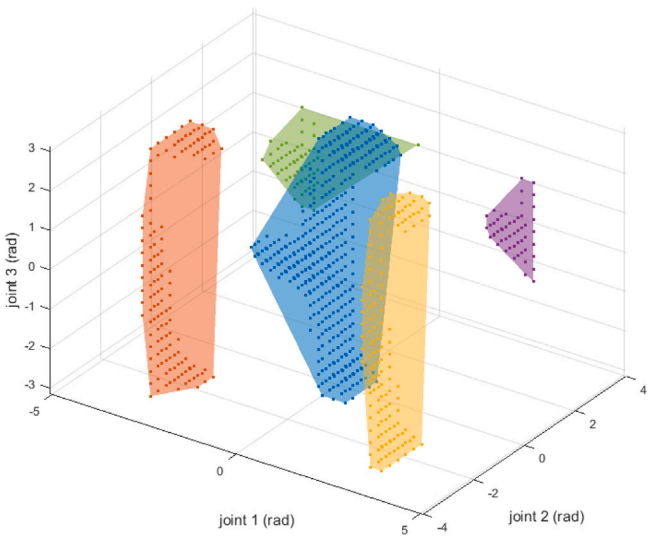


Fig. A.22. Infeasible configurations for a 6-DOF robot. Each point represents an infeasible joint 1-2-3 configuration. These infeasible configurations can be wrapped by a series of convex hulls.

where $L_4, L_5 \in \mathbb{R}^+$ are positive constants.

Since the robot system is continuously evolving and the environmental obstacles are also evolving with bounded velocities in reality, Assumption 2 is easy to be met in practice if we sample the trajectory dense enough. Assumption 2 ensures that the feasible set deforms continuously with respect to time, which will be used in the proof of Theorem 1.

A.2. Continuous-time feasibility theorem

In this subsection, we will first provide the theory and the associated proof to answer **Q2**. Here we define the existence of continuous trajectory that goes through all the reference waypoints as: $\exists \mathcal{T}_i : [0,1] \to X$ indexed by interval $\mathcal{I} = [0,1]$, such that $\mathcal{T}_i(0) = \boldsymbol{x}_r^{[i]}, \mathcal{T}_i(1) = \boldsymbol{x}_r^{[i+1]}, i = 1, 2, \dots, M-1$.

Note that the long term planner plans in discrete time. And during execution, the robot generates a continuous path passing the planned discrete waypoints s_D. Thus, we should prove that there exists a continuous-time path passing through all the waypoints of discrete-time planned trajectory, such that every point along this continuous-time path is safe (continuous-time feasibility).

Theorem 1 (***Continuous-Time Feasibility***). *If the system satisfies Assumptions 1 and 2, start pose and end pose are feasible, i.e.* $\forall j, D_j^{[1:2]}(\boldsymbol{x}^{[1]}) > 0, D_j^{[M-1:M]}(\boldsymbol{x}^{[M]}) > 0$, *then by using a continuous-time reference trajectory* $\boldsymbol{x}_r(t_0 : t_0+T)$ *with high enough sampling rate, i.e.* $t_s = t^{[i+1]} - t^{[i]} \to 0, \forall i = 1, \dots, M-1$, *the long term planner can always find a continuous feasible path* $\mathcal{T}(s,t) : \mathbb{R}^{N\times M} \times \mathbb{R} \mapsto \mathbb{R}^N$, *where* $\mathcal{T}$ *is a continuous function with respect to time* $t \in [t^{[1]}, t^{[M]}]$, *and* $\mathcal{T}(\boldsymbol{s}, t^{[i]}) = \boldsymbol{x}^{[i]}, i = 1, \dots, M$, *such that every point on the path is safe, i.e.* $D_j^{[i:i+1]}(\mathcal{T}(t)) > 0, \forall i = 1, \dots, M-1, j \in \mathbb{N}^+$.

Proof.

According to Assumption 1, from time step i to time step $i+1$, we have that the infeasible set in configuration space are a set of disjoint convex hulls $C_j^{[i:i+1]}, j \in \mathbb{N}^+$. Since that distance constraints in (20) are convex functions [48], according to the Feasibility Lemma (Lemma 5.1) from Liu et al. [57], we have that CFS algorithm can always find non-empty convex feasible set $\mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r)$ for any reference configuration state $\boldsymbol{x}_r$ with respect to the constraint $D_j^{[i:i+1]}(\boldsymbol{x}) > 0$, where $\mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r) = \bigcap_{j\in\mathbb{N}^+} \mathcal{F}_j^{[i:i+1]}(\boldsymbol{x}_r) \neq \emptyset$ and $\forall \boldsymbol{x} \in \mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r), D_j^{[i:i+1]}(\boldsymbol{x}) > 0, \forall j$. According to convex feasible set algorithm described in Section 5.1, we compute $\mathcal{F}^{[i:i+1]}$ with respect to $\boldsymbol{x}_r^{[i]}$ at time step i, and $\mathcal{F}_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) := \{\boldsymbol{x} : D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) + \nabla D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]})(\boldsymbol{x} - \boldsymbol{x}_r^{[i]}) \geq 0\}$.

Therefore, we can represent $\mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r^{[i]})$ in the form of $\boldsymbol{A}^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x} \leq \boldsymbol{b}^{[i:i+1]}(\boldsymbol{x}_r^{[i]})$, such that:

$$\boldsymbol{A}^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) = \begin{bmatrix} \nabla_{\boldsymbol{x}} D_1^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) \\ \nabla_{\boldsymbol{x}} D_2^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) \\ \nabla_{\boldsymbol{x}} D_3^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) \\ \vdots \end{bmatrix} \tag{A.3}$$

$$\boldsymbol{b}^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) = \begin{bmatrix} D_1^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) - \nabla_{\boldsymbol{x}} D_1^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}_r^{[i]} \\ D_2^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) - \nabla_{\boldsymbol{x}} D_2^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}_r^{[i]} \\ D_3^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) - \nabla_{\boldsymbol{x}} D_3^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}_r^{[i]} \\ \vdots \end{bmatrix}$$

In the following discussion, we will show that if the trajectory sampling rate is high enough, i.e., $t_s = t^{[i+1]} - t^{[i]} \to 0$, then $\mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) \bigcap \mathcal{F}^{[i+1:i+2]}(\boldsymbol{x}_r^{[i+1]}) \neq \emptyset$.

Since the reference trajectory $\boldsymbol{x}_r(t_0 : t_0+T)$ is continuous and $t_s \to 0$, we have $\|\boldsymbol{x}_r^{[i+1]} - \boldsymbol{x}_r^{[i]}\| \to 0, i = 1, 2, \dots, M-1$. According to (P2) and (A2), the following condition holds:

$$\begin{aligned} &\forall j, \ \|\nabla_{\boldsymbol{x}} D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i+1]}) - \nabla_{\boldsymbol{x}} D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\| \\ &\leq \|\nabla_{\boldsymbol{x}} D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i+1]}) - \nabla_{\boldsymbol{x}} D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i]})\| \\ &\quad + \|\nabla_{\boldsymbol{x}} D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i]}) - \nabla_{\boldsymbol{x}} D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\| \\ &\quad \leq L_5 t_s + L_2 \|\boldsymbol{x}_r^{[i+1]} - \boldsymbol{x}_r^{[i]}\| \to 0 \end{aligned} \tag{A.4}$$

Given (P1), (P2), (P3), (A1) and (A2), and noting that $\forall i, \|\boldsymbol{x}_r^{[i]}\|$ is bounded, the following condition holds:

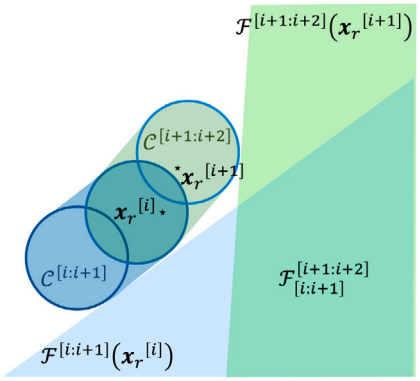
$$\begin{aligned} &\forall j, \ \| \ D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i+1]}) - \nabla_{\boldsymbol{x}^{[i+1]}} D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i+1]})\boldsymbol{x}_r^{[i+1]} \\ &\quad - (D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) - \nabla_{\boldsymbol{x}} D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}_r^{[i]}) \ \| \\ &\leq \| \ D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i+1]}) - D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i]}) \ \| + \\ &\quad \|\nabla_{\boldsymbol{x}} D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i+1]})\boldsymbol{x}_r^{[i+1]} - \nabla_{\boldsymbol{x}} D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}_r^{[i]}\| \\ &\quad + \|\nabla_{\boldsymbol{x}} D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i+1]})\boldsymbol{x}_r^{[i]} - \nabla_{\boldsymbol{x}} D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}_r^{[i]}\| \\ &\quad + \|D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i]}) - D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\| \\ &\quad + \|\nabla_{\boldsymbol{x}} D_j^{[i+1:i+2]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}_r^{[i]} - \nabla_{\boldsymbol{x}} D_j^{[i:i+1]}(\boldsymbol{x}_r^{[i]})\boldsymbol{x}_r^{[i]}\| \\ &\leq L_1\|\boldsymbol{x}_r^{[i+1]} - \boldsymbol{x}_r^{[i]}\| + L_3\|\boldsymbol{x}_r^{[i+1]} - \boldsymbol{x}_r^{[i]}\| \\ &\quad + L_2\|\boldsymbol{x}_r^{[i+1]} - \boldsymbol{x}_r^{[i]}\|\|\boldsymbol{x}_r^{[i]}\| + L_4 t_s + L_5 t_s \|\boldsymbol{x}_r^{[i]}\| \to 0 \end{aligned} \tag{A.5}$$

Based on (A.4) and (A.5), given the fact that the numbers j of C_j is limited, we further have $\mathcal{F}_{[i:i+1]}^{[i+1:i+2]} := \mathcal{F}^{[i+1:i+2]}(\boldsymbol{x}_r^{[i+1]}) \bigcap \mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r^{[i]}) \neq \emptyset$, and we illustrate the existence of $\mathcal{F}_{[i:i+1]}^{[i+1:i+2]}$ in Fig. A.23, where we also demonstrate $\mathcal{F}_{[i:i+1]}^{[i+1:i+2]}$ does not exist if Assumption 2 does not hold . Therefore, we can find a common solution $\boldsymbol{x}^{[i+1]} \in \mathcal{F}_{[i:i+1]}^{[i+1:i+2]}, i = 1, 2, \dots, M-2$.

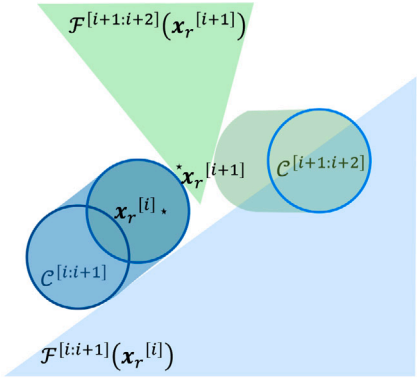
Then we can construct a continuous path in $[t^{[i]}, t^{[i+1]}]$ as

$$\mathcal{T}_i(\tau') = (1-\tau')\boldsymbol{x}^{[i]} + \tau'\boldsymbol{x}^{[i+1]}, \tau' \in [0,1] \tag{A.6}$$

$\mathcal{T}_i$ is actually a straight line connecting $\{\boldsymbol{x}^{[i]}, \boldsymbol{x}^{[i+1]}\}$. Noting that $\mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r^{[i]})$ is convex, $\boldsymbol{x}^{[i]} \in \mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r^{[i]})$, and $\boldsymbol{x}^{[i+1]} \in \mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r^{[i]})$, we have $\forall \tau' \in [0,1], \mathcal{T}_i(\tau') \in \mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r^{[i]})$. Thus, $\forall j, D_j^{[i:i+1]}(\mathcal{T}_i(\tau')) > 0, \forall \tau' \in [0,1]$.



(a) Existence of $\mathcal{F}^{[i+1:i+2]}_{[i:i+1]}$ when Assumption 2 is satisfied.



(b) Nonexistence of $\mathcal{F}^{[i+1:i+2]}_{[i:i+1]}$ when Assumption 2 is not satisfied.

Fig. A.23. Illustration of $\mathcal{F}^{[i+1:i+2]}_{[i:i+1]}$ in Theorem 1.

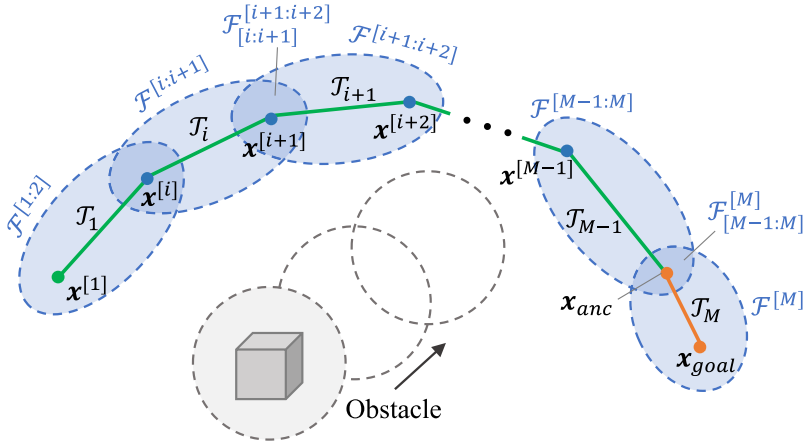


Fig. A.24. The construction of continuous feasible path $\mathcal{T}$.

Then we connect all line segment paths $\mathcal{T}_i$ and construct the complete continuous path as shown in Fig. A.24, and

$$\mathcal{T}(\tau)=\begin{cases} \mathcal{T}_1((M-1)\tau), & \tau\in[0,\frac{1}{M-1}) \\ \vdots & \vdots \\ \mathcal{T}_i((M-1)\tau-(i-1)), & \tau\in[\frac{i-1}{M-1},\frac{i}{M-1}) \\ \vdots & \vdots \\ \mathcal{T}_{M-1}((M-1)\tau-(M-2)), & \tau\in[\frac{M-2}{M-1},1] \end{cases} \tag{A.7}$$

Obviously, every point along the path $\mathcal{T}(\tau)$ are within the convex feasible set, i.e., $\forall j, \forall\tau\in[0,1], D_j^{[i:i+1]}(\mathcal{T}(\tau))>0, i=1,2,\dots,M-1$.

Note that the initial pose and end pose are both feasible, with $\boldsymbol{x}^{[1]}=\boldsymbol{x}(t_0)$ and $\boldsymbol{x}^{[M]}=\boldsymbol{x}_{goal}$. However, we only have $\boldsymbol{x}^{[1]}\in\mathcal{F}^{[1:2]}(\boldsymbol{x}_r^{[1]})$, whereas $\boldsymbol{x}^{[M]}\notin\mathcal{F}^{[M-1:M]}(\boldsymbol{x}_r^{[M]})$. Thus the predefined continuous-time trajectory $\mathcal{T}(\tau)$ only continuously connect $x^{[i]}, i=1,2,\dots,M-1$. To construct a feasible continuous-time trajectory segment for connecting $\boldsymbol{x}^{[M-1]}$ towards $\boldsymbol{x}^{[M]}$, we firstly define $\mathcal{F}^{[M]}(\boldsymbol{x}_r^{[M]}):=\mathcal{F}^{[M:M+1]}(\boldsymbol{x}_r^{[M]})$. Similar to the aforementioned proof, we have $\mathcal{F}^{[M]}_{[M-1:M]}=\mathcal{F}^{[M]}(\boldsymbol{x}_r^{[M]})\bigcap\mathcal{F}^{[M-1:M]}(\boldsymbol{x}_r^{[M-1]})\neq\emptyset$. Next, we find an anchor point $\boldsymbol{x}_{anc}\in\mathcal{F}^{[M]}_{[M-1:M]}$. Then we can construct a straight line $\mathcal{T}_M$ connecting $\boldsymbol{x}_{anc}$ and $\boldsymbol{x}^{[M]}$ as shown in Fig. A.24, and modify the $\mathcal{T}_{M-1}$ in (A.7) to connecting $\boldsymbol{x}^{[M-1]}$ and $\boldsymbol{x}_{anc}$. With these modifications, it is proved that every point along the path the continuous path $\mathcal{T}=\mathcal{T}_1,\dots,\mathcal{T}_{M-1},\mathcal{T}_M$ is safe. □

A.3. Probabilistic safety theorem

In this subsection, we will give the theoretical guarantees of probabilistic safety for the shorter term safety planner under uncertainties.

We first define a metric as D in (8) to evaluate the signed distance between the robot and the obstacle at time t based on Definition 1. Denote $j_m=\text{argmin}_{j\in\mathbb{N}^+}D_j^{[i:i+1]}(\boldsymbol{x}(t)), t\in[t^{[i]},t^{[i+1]}]$, then we define $\boldsymbol{\xi}(t)\in\mathbb{R}^N$ as the closest point on the boundary of $C_{j_m}^{[i:i+1]}$ with respect to $\boldsymbol{x}(t)$ such that $D(\boldsymbol{x}(t),\boldsymbol{\xi}(t))=\begin{cases}\|\boldsymbol{x}(t)-\boldsymbol{\xi}(t)\|, & \boldsymbol{x}(t)\notin C_{j_m}^{[i:i+1]}\\ -\|\boldsymbol{x}(t)-\boldsymbol{\xi}(t)\|, & \boldsymbol{x}(t)\in C_{j_m}^{[i:i+1]}\end{cases}=D_{j_m}^{[i:i+1]}(\boldsymbol{x}(t))$. Therefore, we can straightforwardly obtain the following properties for $D(\boldsymbol{x},\boldsymbol{\xi})$:

- (P4) $D(\boldsymbol{x},\boldsymbol{\xi})$ is Lipschitz continuous in $\boldsymbol{x}$ and $\boldsymbol{\xi}$, i.e., $|D(\boldsymbol{x}_1,\boldsymbol{\xi}_1)-D(\boldsymbol{x}_2,\boldsymbol{\xi}_2)|\le\|\boldsymbol{x}_1-\boldsymbol{x}_2\|+\|\boldsymbol{\xi}_1-\boldsymbol{\xi}_2\|$.
- (P5) $D(\boldsymbol{x},\boldsymbol{\xi})$ is a L-smooth function in $\boldsymbol{x}$, i.e., $\|\nabla_{\boldsymbol{x}}D(\boldsymbol{x}_1,\boldsymbol{\xi})-\nabla_{\boldsymbol{x}}D(\boldsymbol{x}_2,\boldsymbol{\xi})\|\le L_6\|\boldsymbol{x}_1-\boldsymbol{x}_2\|$.
- (P6) $\nabla_{\boldsymbol{x}}D(\boldsymbol{x},\boldsymbol{\xi})$ is bounded, i.e., $\|\nabla_{\boldsymbol{x}}D(\boldsymbol{x},\boldsymbol{\xi})\|\le L_7$.

where $L_6, L_7\in\mathbb{R}^+$ are positive constants.

Theorem 2 (***Probabilistic Safety***). *Suppose there are bounded uncertainties on robot tracking, robot position measurement and obstacle perception, i.e., $\dot{\boldsymbol{x}}=\dot{\boldsymbol{x}}^*+\boldsymbol{\delta}_V$, $\boldsymbol{x}=\boldsymbol{x}^*+\boldsymbol{\delta}_R$ and $\boldsymbol{\xi}=\boldsymbol{\xi}^*+\boldsymbol{\delta}_\xi$, where $\dot{\boldsymbol{x}}^*$, $\boldsymbol{x}^*$ and $\boldsymbol{\xi}^*$ are ground truth robot velocity, velocity and closest point on the boundary of $\mathcal{C}$ with respect to x^*, respectively. $\|\boldsymbol{\delta}_u\|\le\Delta_u$, $\|\boldsymbol{\delta}_x\|\le\Delta_x$ and $\|\boldsymbol{\delta}_\xi\|\le\Delta_\xi$*

are bounded uncertainties with upper bound $\Delta_u, \Delta_x, \Delta_\xi > 0$. Then with the proper defined safety margin d_m in safety index and safe control gain k_s, the short term safety planner can always guarantee safety, i.e., $\mathcal{D}(\boldsymbol{x}^*, \boldsymbol{\xi}^*) > 0$.

Proof. The bounded uncertainty assumptions indicate $\|\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*\| \le \Delta_u$, $\|\boldsymbol{x} - \boldsymbol{x}^*\| \le \Delta_x$ and $\|\boldsymbol{\xi} - \boldsymbol{\xi}^*\| \le \Delta_\xi$.

Firstly, we consider the robot tracking uncertainty $\|\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*\| \le \Delta_u$. According to (10), we have $\dot{\phi}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \nabla_{\boldsymbol{x}}\phi^\top \dot{\boldsymbol{x}}$ and $\dot{\phi}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*) = \nabla_{\boldsymbol{x}^*}\phi^\top \dot{\boldsymbol{x}}^*$. The difference between $\dot{\phi}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ and $\dot{\phi}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*)$ satisfying

$$\begin{aligned}
&|\dot{\phi}(\boldsymbol{x}, \dot{\boldsymbol{x}}) - \dot{\phi}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*)| = |\nabla_{\boldsymbol{x}}\phi^\top \dot{\boldsymbol{x}} - \nabla_{\boldsymbol{x}^*}\phi^\top \dot{\boldsymbol{x}}^*| \\
&= |\nabla_{\boldsymbol{x}}\phi^\top \dot{\boldsymbol{x}} - \nabla_{\boldsymbol{x}}\phi^\top \dot{\boldsymbol{x}}^* + \nabla_{\boldsymbol{x}}\phi^\top \dot{\boldsymbol{x}}^* - \nabla_{\boldsymbol{x}^*}\phi^\top \dot{\boldsymbol{x}}^*| \\
&\le |\nabla\phi_{\boldsymbol{x}}^\top(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*)| + |(\nabla_{\boldsymbol{x}}\phi - \nabla_{\boldsymbol{x}^*}\phi)^\top \dot{\boldsymbol{x}}^*| \\
&\le \|\nabla_{\boldsymbol{x}}\phi\|\|\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*\| + \|\nabla_{\boldsymbol{x}}\phi - \nabla_{\boldsymbol{x}^*}\phi\|\|\dot{\boldsymbol{x}}^*\|
\end{aligned} \tag{A.8}$$

According to the definition of ϕ in (8), and based on (P5)(P6), we have

$$\begin{aligned}
&\|\nabla_{\boldsymbol{x}}\phi\| \le L_7 \\
&\|\nabla_{\boldsymbol{x}}\phi - \nabla_{\boldsymbol{x}^*}\phi\| \le L_6\|\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*\|
\end{aligned} \tag{A.9}$$

The true robot velocity $\dot{\boldsymbol{x}}^*$ is the commanded joint velocity, thus by limiting the commanded joint velocity, we have $\|\dot{\boldsymbol{x}}^*\| \le V_b$, where $V_b > 0$ is velocity bound. Therefore, the (A.8) can be represented as

$$|\dot{\phi}(\boldsymbol{x}, \dot{\boldsymbol{x}}) - \dot{\phi}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*)| \le (L_6 + L_7 V_b)\Delta_u \tag{A.10}$$

The bounded difference between $\dot{\phi}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ and $\dot{\phi}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*)$ indicates that we can always guarantee $\dot{\phi}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*) < 0$ by controlling $\dot{\phi}(\boldsymbol{x}, \dot{\boldsymbol{x}}) < -k_s$, where $k_s \ge (L_6 + L_7 V_b)\Delta_u$. Thus the safety under robot tracking uncertainty can be guaranteed.

Now we consider effects on robot position uncertainty $\|\boldsymbol{x} - \boldsymbol{x}^*\| \le \Delta_x$ and obstacle position uncertainty $\|\boldsymbol{\xi} - \boldsymbol{\xi}^*\| \le \Delta_\xi$. According to (P4), we have

$$\begin{aligned}
|\mathcal{D}(\boldsymbol{x}, \boldsymbol{\xi}) - \mathcal{D}(\boldsymbol{x}^*, \boldsymbol{\xi}^*)| &\le \|\boldsymbol{x} - \boldsymbol{x}^*\| + \|\boldsymbol{\xi} - \boldsymbol{\xi}^*\| \\
&\le \Delta_x + \Delta_\xi
\end{aligned} \tag{A.11}$$

Under the short term planner control law (12), we have $\phi(\boldsymbol{x}) < 0$, i.e., $\mathcal{D}(\boldsymbol{x}, \boldsymbol{\xi}) > d_m$ according to (8). By choosing a large enough safety margin $d_m > \Delta_x + \Delta_\xi$, short term safety planner can always guarantee $\mathcal{D}(\boldsymbol{x}^*, \boldsymbol{\xi}^*) > 0$. In summary, once $k_s \ge (L_6 + L_7 V_b)\Delta_u$ and $d_m > \Delta_x + \Delta_\xi$ are satisfied in the designed safety controller, the short term safety planner can guarantee the safety even under all possible robot tracking, robot position measurement and obstacle perception uncertainties. □

A.4. Persistent feasibility theorem

In this subsection, with the help of Theorem 2, we will finally provide the theory and the associated proof to answer **Q1**.

Theorem 3 (***Persistent Feasibility***). *If there is (1) bounded delay and bounded tracking error, (2) no uncertainty, (3) the system satisfies Assumption 1, and (4) the end pose is feasible, i.e. $\forall j, D_j^{[M-1:M]}(\boldsymbol{x}^{[M]}) > 0$, then the long term planner can always find a feasible discrete-time trajectory ($\boldsymbol{s}_D$) at all replanning steps during the execution.*

Proof. There are many factors can contribute to tracking error, such as modeling mismatch, actuation limit and servo control tuning, as long as there is bounded delay and tracking error, and no uncertainty, we have that $\min_{j\in\mathbb{N}^+} D_j(\boldsymbol{x}^*) = \mathcal{D}(\boldsymbol{x}^*, \boldsymbol{\xi}^*) > 0$ for every time step during execution according to Theorem 2, which indicates the start pose $\boldsymbol{x}^{[1]}$ for a replanned trajectory is always feasible during execution.

According to Theorem 1, if the system satisfies Assumption 1, we have that CFS algorithm can always find non-empty convex feasible set $\forall i, \mathcal{F}^{[i:i+1]}(\boldsymbol{x}_r^{[i]})$ for any reference configuration state $\boldsymbol{x}_r^{[i]}$ with respect to the constraint $D_j^{[i:i+1]}(\boldsymbol{x}) > 0$. By the Theorem 4.1 from [57], we have that for $i = 2, 3, \ldots, M-1$, $\boldsymbol{x}^{[i]}$ will converge to a feasible solution, where $D_j^{[i:i+1]}(\boldsymbol{x}^{[i]}) > 0, j \in \mathbb{N}^+$. Since the end of pose $\boldsymbol{x}^{[M]}$ is also feasible, we have long term planner can always find a feasible discrete-time trajectory $\boldsymbol{s}_D = [\boldsymbol{x}^{[1]}, \boldsymbol{x}^{[2]}, \ldots, \boldsymbol{x}^{[M]}]$ at any replanning step during the execution. □

So far, we have discussed the assumptions and theorems to show that (1) if there are bounded delay and tracking error, and there is no uncertainty, then there always exists a feasible discrete long term plan at all replanning steps during execution; (2) If there is no delay and no uncertainty, there always exists a feasible continuous-time trajectory such that every possible point along the trajectory is feasible; (3) If there is bounded uncertainty, the safety of the system is still provably guaranteed.

In order to guarantee the stability and optimality of HLSTS, we still need to prove (1) if there is delay and uncertainty, there exists a feasible continuous trajectory to track the long term plan with bounded error; (2) The robot can converge to the target position safely in finite time, which are left for future works. In the following section, we will validate the effectiveness of HLSTS for safe and efficient manipulation in uncertain clustered environment with both simulation and real world experiments.

References

[1] P.A. Lasota, T. Fong, J.A. Shah, et al., A Survey of Methods for Safe Human-Robot Interaction, Now Publishers, 2017.

[2] J.S. Grover, C. Liu, K. Sycara, Deadlock analysis and resolution for multi-robot systems, in: Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics, Vol. 14, Springer International Publishing, 2021, pp. 294–312.

[3] A.D. Ames, J.W. Grizzle, P. Tabuada, Control barrier function based quadratic programs with application to adaptive cruise control, in: 53rd IEEE Conference on Decision and Control, IEEE, 2014, pp. 6271–6278.

[4] S.L. Herbert, M. Chen, S. Han, S. Bansal, J.F. Fisac, C.J. Tomlin, FaSTrack: A modular framework for fast and guaranteed safe motion planning, in: 2017 IEEE 56th Annual Conference on Decision and Control, CDC, IEEE, 2017, pp. 1517–1522.

[5] S. Magdici, M. Althoff, Fail-safe motion planning of autonomous vehicles, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2016, pp. 452–458.

[6] W. Zhan, C. Liu, C.-Y. Chan, M. Tomizuka, A non-conservatively defensive strategy for urban autonomous driving, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2016, pp. 459–464.

[7] J. Grover, C. Liu, K. Sycara, Why does symmetry cause deadlocks? IFAC-PapersOnLine 53 (2) (2020) 9746–9753.

[8] S.M. LaValle, Rapidly-Exploring Random Trees: a New Tool for Path Planning, Citeseer, 1998.

[9] N. Ratliff, M. Zucker, J.A. Bagnell, S. Srinivasa, CHOMP: Gradient optimization techniques for efficient motion planning, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 2009, pp. 489–494.

[10] C. Park, J. Pan, D. Manocha, ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments, in: Twenty-Second International Conference on Automated Planning and Scheduling, 2012.

[11] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: Autonomous Robot Vehicles, Springer, 1986, pp. 396–404.

[12] C. Liu, M. Tomizuka, Control in a safe set: Addressing safety in human-robot interactions, in: Dynamic Systems and Control Conference. Vol. 46209, American Society of Mechanical Engineers, 2014, V003T42A003.

[13] L. Gracia, F. Garelli, A. Sala, Reactive sliding-mode algorithm for collision avoidance in robotic systems, IEEE Trans. Control Syst. Technol. 21 (6) (2013) 2391–2399.

[14] C. Liu, M. Tomizuka, Safe exploration: Addressing various uncertainty levels in human robot interactions, in: Proceedings of the American Control Conference, ACC, 2015, pp. 465–470.

[15] J.F. Fisac, A.K. Akametalu, M.N. Zeilinger, S. Kaynama, J. Gillula, C.J. Tomlin, A general safety framework for learning-based control in uncertain robotic systems, IEEE Trans. Automat. Control 64 (7) (2019) 2737–2752, http://dx.doi.org/10.1109/TAC.2018.2876389.

[16] R. Cheng, G. Orosz, R.M. Murray, J.W. Burdick, End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks, 2019, CoRR. arXiv:1903.08792.

[17] S. He, C. Hu, S. Lin, Y. Zhu, An online time-optimal trajectory planning method for constrained multi-axis trajectory with guaranteed feasibility, IEEE Robotics and Automation Letters 7 (3) (2022) 7375–7382, http://dx.doi.org/10.1109/LRA.2022.3183536.

[18] A.J. Taylor, A.D. Ames, Adaptive safety with control barrier functions, in: 2020 American Control Conference, ACC, 2020, pp. 1399–1405, http://dx.doi.org/10.23919/ACC45564.2020.9147463.

[19] C. Liu, C.-Y. Lin, Y. Wang, M. Tomizuka, Convex feasible set algorithm for constrained trajectory smoothing, in: 2017 American Control Conference, ACC, IEEE, 2017, pp. 4177–4182.

[20] H.-y. Zhang, W.-m. Lin, A.-x. Chen, Path planning for the mobile robot: A review, Symmetry 10 (10) (2018) 450.

[21] B. Kim, T.T. Um, C. Suh, F.C. Park, Tangent bundle RRT: A randomized algorithm for constrained motion planning, Robotica 34 (1) (2016) 202–225.

[22] B.J. Cohen, S. Chitta, M. Likhachev, Search-based planning for manipulation with motion primitives, in: 2010 IEEE International Conference on Robotics and Automation, IEEE, 2010, pp. 2902–2908.

[23] L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Autom. 12 (4) (1996) 566–580.

[24] D. Fridovich-Keil, S.L. Herbert, J.F. Fisac, S. Deglurkar, C.J. Tomlin, Planning, fast and slow: A framework for adaptive real-time safe trajectory planning, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 387–394.

[25] W. Li, R. Xiong, Dynamical obstacle avoidance of task-constrained mobile manipulation using model predictive control, IEEE Access 7 (2019) 88301–88311.

[26] W.-Y. Zhao, S. He, C. Wen, C. Liu, Contact-rich trajectory generation in confined environments using iterative convex optimization, in: Dynamic Systems and Control Conference, Vol. 84287, American Society of Mechanical Engineers, 2020, V002T31A002.

[27] C. Liu, M. Tomizuka, Real time trajectory optimization for nonlinear robotic systems: Relaxation and convexification, Systems Control Lett. 108 (2017) 56–63.

[28] H.-C. Lin, C. Liu, M. Tomizuka, Fast robot motion planning with collision avoidance and temporal optimization, in: 2018 15th International Conference on Control, Automation, Robotics and Vision, ICARCV, IEEE, 2018, pp. 29–35.

[29] J. Chen, C. Liu, M. Tomizuka, Foad: Fast optimization-based autonomous driving motion planner, in: 2018 Annual American Control Conference, ACC, IEEE, 2018, pp. 4725–4732.

[30] T. Wei, C. Liu, Safe control algorithms using energy functions: A uni ed framework, benchmark, and new directions, in: 2019 IEEE 58th Conference on Decision and Control, CDC, IEEE, 2019, pp. 238–243.

[31] J. Rauch, J.A. Smoller, Qualitative Theory of the Fitzhugh-Nagumo Equations, Elsevier, 1978.

[32] A.G. Barto, S. Mahadevan, Recent advances in hierarchical reinforcement learning, Discrete Event Dyn. Syst. 13 (1) (2003) 41–77.

[33] J. Wang, C. Hu, Y. Zhu, Cpg-based hierarchical locomotion control for modular quadrupedal robots using deep reinforcement learning, IEEE Robotics and Automation Letters 6 (4) (2021) 7193–7200, http://dx.doi.org/10.1109/LRA.2021.3092647.

[34] T.D. Kulkarni, K. Narasimhan, A. Saeedi, J. Tenenbaum, Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, Adv. Neural Inf. Process. Syst. 29 (2016) 3675–3683.

[35] J. Garcıa, F. Fernández, A comprehensive survey on safe reinforcement learning, J. Mach. Learn. Res. 16 (1) (2015) 1437–1480.

[36] A. Ray, J. Achiam, D. Amodei, Benchmarking safe exploration in deep reinforcement learning, 2019, arXiv preprint arXiv:1910.01708.

[37] R. Scattolini, Architectures for distributed and hierarchical model predictive control–a review, J. Process Control 19 (5) (2009) 723–731.

[38] C. Liu, M. Tomizuka, Robot safe interaction system for intelligent industrial co-robots, 2018, arXiv preprint arXiv:1808.03983.

[39] C. Liu, T. Tang, H.-C. Lin, Y. Cheng, M. Tomizuka, Serocs: safe and efficient robot collaborative systems for next generation intelligent industrial co-robots, arXiv preprint arXiv:1809.08215 (2018).

[40] H.-C. Lin, C. Liu, Y. Fan, M. Tomizuka, Real-time collision avoidance algorithm on industrial manipulators, in: Control Technology and Applications (CCTA), 2017 IEEE Conference on, IEEE, 2017, pp. 1294–1299.

[41] W. Zhao, T. He, C. Liu, Model-free safe control for zero-violation reinforcement learning, in: 5th Annual Conference on Robot Learning, 2021, URL: https://openreview.net/forum?id=UGp6FDaxB0f.

[42] S. He, C. Hu, S. Lin, Y. Zhu, M. Tomizuka, Real-time time-optimal continuous multi-axis trajectory planning using the trajectory index coordination method, ISA Transactions (ISSN: 0019-0578) 131 (2022) 639–649, http://dx.doi.org/10.1016/j.isatra.2022.05.016, https://www.sciencedirect.com/science/article/pii/S0019057822002506.

[43] W. Zhao, T. He, C. Liu, Probabilistic safeguard for reinforcement learning using safety index guided gaussian process models, arXiv preprint arXiv:2210.01041 (2022).

[44] W. Zhao, T. He, T. Wei, S. Liu, C. Liu, Safety index synthesis via sum-of-squares programming, arXiv preprint arXiv:2209.09134 (2022).

[45] T. Wei, S. Kang, W. Zhao, C. Liu, Persistently feasible robust safe control by safety index synthesis and convex semi-infinite programming, IEEE Control Systems Letters (2022).

[46] P.T. Boggs, J.W. Tolle, Sequential quadratic programming, Acta Numer. 4 (1995) 1–51.

[47] C. Liu, C.-Y. Lin, M. Tomizuka, The convex feasible set algorithm for real time optimization in motion planning, SIAM J. Control Optim. 56 (4) (2018) 2712–2733.

[48] M. Safdari, The distance function from the boundary of a domain with corners, Nonlinear Anal. 181 (2019) 294–310.

[49] L. Bochmann, T. Bänziger, A. Kunz, K. Wegener, Human-robot collaboration in decentralized manufacturing systems: An approach for simulation-based evaluation of future intelligent production, Procedia CIRP 62 (2017) 624–629.

[50] Y. Cheng, W. Zhao, C. Liu, M. Tomizuka, Human motion prediction using semi-adaptable neural networks, in: 2019 American Control Conference (ACC), IEEE, 2019, pp. 4884–4890.

[51] P. Tsarouchi, A.-S. Matthaiakis, S. Makris, G. Chryssolouris, On a human-robot collaboration in an assembly cell, Int. J. Comput. Integr. Manuf. 30 (6) (2017) 580–589.

[52] C. Liu, M. Tomizuka, Algorithmic safety measures for intelligent industrial co-robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 2016, pp. 3095–3102.

[53] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, P. Abbeel, Motion planning with sequential convex optimization and convex collision checking, Int. J. Robot. Res. 33 (9) (2014) 1251–1270.

[54] W. Zhao, S. He, C. Liu, Provably safe tolerance estimation for robot arms via sum-of-squares programming, 2021, arXiv preprint arXiv:2104.08896.

[55] Y. Dong, Q.A. Fu, X. Yang, T. Pang, H. Su, Z. Xiao, J. Zhu, Benchmarking adversarial robustness on image classification, in: 2020 IEEE/CVF CVPR, 2020, pp. 318–328, http://dx.doi.org/10.1109/CVPR42600.2020.00040.

[56] H. Luo, X. Wang, B. Lukens, Variational analysis on the signed distance functions, J. Optim. Theory Appl. 180 (3) (2019) 751–774.

[57] C. Liu, C.-Y. Lin, M. Tomizuka, The convex feasible set algorithm for real time optimization in motion planning, SIAM J. Control Optim. 56 (4) (2018) 2712–2733.