KDF: Kinodynamic Motion Planning via Geometric Sampling-based Algorithms and Funnel Control

Christos K. Verginis, *Member, IEEE,* Dimos V. Dimarogonas, *Senior Member, IEEE,* and Lydia E. Kavraki, *Fellow, IEEE*

***Abstract*—We integrate sampling-based planning techniques with funnel-based feedback control to develop KDF, a new framework for solving the kinodynamic motion-planning problem via funnel control. The considered systems evolve subject to complex, nonlinear, and uncertain dynamics (aka differential constraints). Firstly, we use a *geometric* planner to obtain a high-level safe path in a user-defined extended free space. Secondly, we develop a low-level funnel control algorithm that guarantees safe tracking of the path by the system. Neither the planner nor the control algorithm use information on the underlying dynamics of the system, which makes the proposed scheme easily distributable to a large variety of different systems and scenarios. Intuitively, the funnel control module is able to implicitly accommodate the dynamics of the system, allowing hence the deployment of purely geometrical motion planners. Extensive computer simulations and hardware experiments with a 6-DOF robotic arm validate the proposed approach.**

***Index Terms*—kinodynamic motion planning, uncertain dynamics, funnel control**

I. Introduction

MOTION planning of autonomous systems is one of the most fundamental problems in robotics, with numerous applications such as exploration, autonomous driving, robotic manipulation, autonomous warehouses, and multi-robot coordination [1], [2]. It has been extensively studied in the related literature; works have been continuously developed for the last three decades, exploring plenty of variations, including feedback control, discrete planning, uncertain environments, and multi-agent systems. One important and active area of research consists of *kinodynamic* motion planning, i.e., when the planning algorithm takes into account the underlying system dynamics, also known as differential constraints [1].

In this paper we develop KDF, an algorithmic framework for the kinodynamic motion-planning problem by integrating sampling-based algorithms with intelligent feedback control. We consider systems that evolve subject to high-dimensional dynamics, which are highly nonlinear and uncertain. The proposed framework is the integration of the following three modules. The first module is the KDF sampling-based motion planners (KDF-MP), which is a family of geometric planners that produce a path in an "extended" free space. By "extended" we mean that the obtained path has some clearance with respect to the workspace obstacles. The second module is the smoothening of the derived path and its endowment with time constraints in order to produce a smooth time-varying trajectory. The third module is a funnel-based, feedback-control scheme that achieves safe tracking of this trajectory within the clearance of the extended free space. Neither of the aforementioned modules uses any information on the dynamics of the system. The proposed framework guarantees that the system will follow safely the derived path, free from collisions. Loosely speaking, we augment geometric motion planning algorithms with extended free-space capabilities and intelligent feedback control to provide a new solution to the kinodynamic motion-planning problem. The incorporation of the control scheme relieves the sampling-based motion planner from the system dynamics and their uncertainties.

Feedback control is a popular methodology to tackle motion-planning problems, since it simultaneously solves the planning and control problems, offering a closed-form policy for the control input of the system. Artificial potential fields constitute the main tool of closed-form feedback control methods. Early works develop the so-called "navigation functions" [3], appropriately constructed terms whose gradient constitutes a vector field that takes the system safely to the goal configuration from almost all initial conditions (except for a set of initial conditions that has measure zero). Navigation functions can accommodate sphere worlds (spherical obstacles), as well as star-shaped obstacles via appropriate diffeomorphic transformations [4]. Several works build on the notion of navigation functions, and propose harmonic-based potential fields as well as point-world transformations [5], [6]. Potential field-based feedback control schemes have also accommodated multi-robot systems as well as higher order dynamics and model uncertainties [7]–[9]. Optimization-based feedback control techniques, such as Model Predictive Control (MPC) and barrier functions have also been employed to tackle the motion planning problem [10]–[13].

Although feedback control is a promising and convenient tool for motion planning problems, it is usually restricted to simple robot shapes, such as spheres or ellipsoids. For more complex structures, such as high-dimensional robotic manipulators, the aforementioned strategies can guarantee safety but

C. K. Verginis is with the Division of Signals and Systems, Department of Electrical Engineering, Uppsala University, Uppsala, Sweden. e-mail: christos.verginis@angstrom.uu.se.

D. V. Dimarogonas is with the School of Electrical and Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden, e-mail: dimos@kth.se.

L. E. Kavraki is with the Department of Computer Science at Rice University, Houston, TX, USA, e-mail: kavraki@rice.edu.

This work was supported by the H2020 ERC Starting Grant BUCOPHSYS, the European Union's Horizon 2020 Research and Innovation Programme under the GA No. 731869 (Co4Robots), the Swedish Research Council (VR), the Knut och Alice Wallenberg Foundation (KAW), the Swedish Foundation for Strategic Research (SSF), and the National Science Foundation project NSF 2008720 (LEK).

suffer from local minima configurations, or high computation times that render them impractical. Randomized planning has been introduced to tackle such scenarios; [14] and [15], [16] develop the notions of probabilistic roadmaps (PRM) and trees (Expansive Space Trees - EST, Rapidly Exploring Random Trees - RRT), constituting efficient and probabilistically complete geometric solutions to high-dimensional motion-planning problems. These methodologies build a discrete graph that spans the free space by incremental sampling, providing thus a safe path to be followed by the robot. Variants of sampling-based algorithms have been also proposed in order to improve their attributes; RRT-connect [17] computes two trees in the free space (from the initial and goal configuration, respectively), speeding up the convergence to the goal, and asymptotically optimal algorithms, such as RRT*, PRM*, provide a path whose length becomes shorter (more optimal) as the number of samples increases [18]. The initial versions of the aforementioned algorithms are geometrical, without accommodating the dynamics (differential constraints) of the system. To that end, tree-based algorithms such as RRTs and ESTs have been extended to kinodynamic planning [15], [19], [20]. Kinodynamic planning algorithms simulate forward the dynamics of the system by randomly sampling control inputs, in order to find a feasible path in the state space. Moreover, similarly to PRM, [21] and [22] introduced the framework of LQR-trees, i.e., trees of trajectories that probabilistically cover the free space. By linearizing the system dynamics and using optimal control techniques, every point of the free space is assigned a funnel corresponding to its region of attraction with respect to the goal configuration. Similar ideas are used in [23], where dynamics linearization and reachability sets are employed to develop an optimal kinodynamic algorithm. Sampling-based algorithms have been also integrated with receding horizon optimization techniques [24] whereas [25] develops a Hamilton-Jacobi-Bellman approach. In this work we leverage geometric sampling-based motion planning-techniques and feedback control; we integrate the two, efficiently combining and exploiting their benefits and avoiding thus high computation times and undesired local minima configurations.

Another important disadvantage of the majority of the related works in motion planning is their strong dependence on the considered model of the system dynamics; the respective algorithms use partial or full information on the underlying dynamic models. Optimization-based algorithms usually employ dynamics linearization or simulate forward the dynamical model to obtain an optimal control input. The latter is similar to kinodynamic sampling-based motion planning algorithms, which simulate forward the model using random inputs to obtain feasible samples in the free space. The accurate identification of the system dynamics of real robots is a tedious procedure, due to the high uncertainty in the several components (dynamic parameters, friction terms) and unknown exogenous disturbances. Hence, the considered dynamic models used in the aforementioned algorithms do not match sufficiently enough the dynamics of the actual system. As a result, the actual trajectories of the robotic system might deviate from the predicted/planned ones, jeopardizing thus safety and degrading performance.

Similar to the approach developed in this paper, the works [26]–[35] develop two-stage algorithms, combining geometric planning in an extended free space with control procedures. However, [26]–[29] do not consider any uncertainties in the robot dynamics, while [31], [32], [35] consider stochastic uncertainties, modeled via Gaussian distributions and belief trees and spaces; [32] develops a receding-horizon controlled based on such spaces. These approaches, however, usually deal with linearized dynamics, and/or propagate the uncertainties on the planning horizon, constraining thus the free space excessively. Additionally, for high-dimensional articulated robots, such as robotic manipulators, and complicated obstacle-cluttered workspaces, receding-horizon approaches might be too computationally expensive and result in local-minima configurations. The work [30] proposes an algorithm that builds trees of funnels based on the (known) bounds of model disturbances, restricted however to polynomial robot dynamics. Similarly, the works [33], [34] consider dynamic uncertainties that are, however, restricted to uniformly bounded disturbances that evolve in a priori known sets. Additionally, the aforementioned works cannot guarantee the evolution of the system in *predefined* bounds that are independent of the dynamic uncertainties. In this paper, we consider systems with high-order nonlinear dynamics. The dynamics consist of state-dependent terms and time-varying disturbances that are *unknown*. In contrast to the aforementioned works (e.g., [30], [34]), we consider that these time-varying disturbances evolve in bounded but *unknown* sets, whereas the state-dependent terms might grow unbounded. Neither the planning nor the feedback-control module use any information on the underlying system dynamics or their bounds, providing thus robustness to model uncertainties and unknown external disturbances, and applicability to a large variety of different systems and scenarios. More specifically, the proposed framework exhibits the following important characteristics:

1) The (unknown) dynamics of the system are not simulated forward in time and are hence decoupled from the motion planner. This results in the latter being purely geometrical, depending on the geometry of the configuration space and user-defined funnel bounds that are set a priori and define the extended free space.
2) Even though kth-order dynamics are considered, the motion planner searches for a path only in the configuration space, since the dynamics are appropriately compensated by the designed feedback control protocol.
3) We do not resort to linearization of the dynamics and computation of basins of attraction around the output trajectories, since the designed feedback control protocol applies directly to the nonlinear model.

Note that, since the sampling-based motion planner involved in our framework is purely geometric, it is expected to yield lower complexity than standard kinodynamic planning algorithms. Such algorithms sample points in a space of larger dimension, including random states and control inputs, and simulate forward the underlying dynamics; hence they usually require more computational resources than geometric planners.

It should be noted that similar ideas were pursued in [36], [37], without however considering the complex unknown systems adopted in this work. We validate the proposed methodology an Unmanned Aerial Vehicle (UAV) and a 6DOF UR5 manipulator in CoppeliaSim environment [38], as well as a 6DOF Hebi-Robotics manipulator. This paper is an extension of our recent work [39] along the following directions: firstly, we generalize our framework to a family of geometric sampling-based motion planners (as opposed to just RRT used in [39]); secondly, we use a different control algorithm that can handle more general robot dynamics and uncertainties; thirdly, the bounds that define the funnel where the system evolves in, which also define the extended free space in the developed motion planner, are *a priori user-defined*. This is in contrast to [39], where the bounds depended on the system dynamics and gain tuning was needed to shrink the funnel and produce less conservative trajectories. Finally, we use extensive hardware experiments to validate the efficiency of the proposed framework.

II. Problem Formulation

Consider a robotic system characterized by the configuration vector $q_1 \in \mathbb{T} \subset \mathbb{R}^n$, $n \in \mathbb{N}$. Usual robotic structures (e.g., robotic manipulators) might consist of translational and rotational joints, which we define here as $q^{\mathfrak{t}} = [q_1^{\mathfrak{t}}, \dots, q_{n_{tr}}^{\mathfrak{t}}]^\top \in \mathbb{R}^{n_{tr}}$ and $q^{\mathfrak{r}} = [q_1^{\mathfrak{r}}, \dots, q_{n_r}^{\mathfrak{r}}]^\top \in [0, 2\pi)^{n_r}$, respectively, with $n_{tr} + n_r = n$, and hence $\mathbb{T} \coloneqq \mathcal{W}_{tr} \times [0, 2\pi)^{n_r}$, where $\mathcal{W}_{tr}$ is a closed subset of $\mathbb{R}^{n_{tr}}$. Without loss of generality, we assume that $q_1 = [(q^{\mathfrak{t}})^\top, (q^{\mathfrak{r}})^\top]^\top$.

We consider kth-order systems, with $k \geq 2$, of the form

$$\dot{q}_i = f_i(\bar{q}_i, t) + g_i(\bar{q}_i, t) q_{i+1}, \quad \forall i \in \{1, \dots, k-1\} \tag{1a}$$

$$\dot{q}_k = f_k(\bar{q}_k, t) + g_k(\bar{q}_k, t) u, \tag{1b}$$

where $\bar{q}_i \coloneqq [q_1^\top, \dots, q_i^\top]^\top \in \mathbb{T} \times \mathbb{R}^{n(i-1)}$, for all $i \in \{1, \dots, k\}$, and $u \in \mathbb{R}^n$ is the control input of the system. Note that the kth-order model (1) generalizes the simpler 2nd-order Lagrangian dynamics, which is commonly used in the related literature.

The vector fields f_i, g_i, which represent various terms in robotic systems (inertia, Coriolis, friction, gravity, centrifugal) are considered to be completely unknown to the designer/planner, for all $i \in \{1, \dots, k\}$. The only assumptions we make for the system are mild continuity and controllability conditions, as follows:

Assumption 1. *The maps $\bar{q}_i \mapsto f_i(\bar{q}_i, t) : \mathbb{T} \times \mathbb{R}^{n(i-1)} \to \mathbb{R}^n$, $\bar{q}_i \mapsto g_i(\bar{q}_i, t) : \mathbb{T} \times \mathbb{R}^{n(i-1)} \to \mathbb{R}^{n \times n}$ are continuously differentiable for each fixed $t \in \mathbb{R}_{\geq 0}$ and the maps $t \mapsto f_i(\bar{q}_i, t) : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, $t \mapsto g_i(\bar{q}_i, t) : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n \times n}$ are piecewise continuous and uniformly bounded for each fixed $\bar{q}_i \in \mathbb{T} \times \mathbb{R}^{n(i-1)}$, for all $i \in \{1, \dots, k\}$, by unknown bounds.*

Assumption 2. *It holds that*

$$\lambda_{\min}\left(g_i(\bar{q}_i, t) + g_i(\bar{q}_i, t)^\top \right) \geq \underline{\lambda} > 0,$$

for a positive constant $\underline{\lambda}$, for all $\bar{q}_i \in \mathbb{R}^{n(i-1)}$, $t \geq 0$, $i \in \{1, \dots, k\}$, where $\lambda_{\min}(\cdot)$ is the minimum eigenvalue of a matrix.

Assumption 1 intuitively states that the terms $f_i(\cdot)$, $g_i(\cdot)$ are sufficiently smooth in the state $\bar{q}_i$ and bounded in time t. The smoothness in $\bar{q}_i$ is satisfied by standard terms that appear in the dynamics of robotic systems (inertia, Coriolis, gravity); friction terms might pose an exception, since they are usually modeled by *discontinuous* functions of the state [40]. Although smooth friction approximations can be employed [41], the proposed control design can be adapted to account for discontinuous dynamics (as, e.g., in [42]); we consider smooth terms for ease of exposition.

Note that, unlike existing methodologies in the control-design literature, such as feedback linearization [43], traditional adaptive control [39], control-barrier-function design [44], or model-predictive control [45], we consider that the dynamic terms $f_i(\cdot)$ and $g_i(\cdot)$ are completely unknown. Further, the incorporation of time dependence in $f_i(\cdot)$, $g_i(\cdot)$ reflects time-varying and bounded external disturbances (e.g., wind or adversarial perturbations). However, unlike existing works in motion planning and control (e.g., [30], [34]), the bounds of such disturbances are considered unknown. Finally, note that we assume that the terms $f_i(\cdot)$, $g_i(\cdot)$ are continuously differentiable but not upper-bounded with respect to $\bar{q}_i$. The proposed algorithm *guarantees* the boundedness of $\bar{q}_i$ and, consequently, of $f_i(\cdot)$, $g_i(\cdot)$.

Assumption 2 is a sufficiently controllability condition for (1); intuitively, it states that the input matrices g_i do not change the direction imposed to the system by q_{i+1} when the latter are viewed as inputs (with $q_{k+1} = u$). Note that standard holonomic Lagrangian systems satisfy this condition. Examples include robotic manipulators, omnidirectional mobile robots, and fully actuated aerial vehicles. Systems not covered by (1) consist of underactuated or non-holonomic robots, such as unicycles, underactuated aerial or underwater vehicles. Each of these systems requires special attention and cannot be framed into the general framework presented in this work. However, funnel control can be applied for such systems (see, e.g., [46]–[48]) and the proposed methodology could be extended to account for non-holonomic and underactuation constraints.

We consider that the robot operates in a workspace $\mathcal{W} \subset \mathbb{R}^3$ filled with obstacles occupying a closed set $\mathcal{O} \subset \mathbb{R}^3$. We denote the set of points that consist the volume of the robot at configuration q_1 as $\mathcal{A}(q_1) \subset \mathbb{R}^3$. The collision-free space is defined as the open set $\mathcal{A}_{\text{free}} \coloneqq \{q_1 \in \mathbb{T} : \mathcal{A}(q_1) \cap \mathcal{O} = \emptyset\}$. Our goal is to achieve safe navigation of the robot to a predefined goal region $Q_g \subset \mathcal{A}_{\text{free}}$ from an initial configuration $q_1(0) \in \mathcal{A}_{\text{free}}$ via a path $\boldsymbol{q}_{\text{p}} : [0, \sigma] \to \mathcal{A}_{\text{free}}$ satisfying $\boldsymbol{q}_{\text{p}}(0) = q_1(0)$ and $\boldsymbol{q}_{\text{p}}(\sigma) \in Q_g$, for some positive σ.

The problem we consider is the following:

Problem 1. *Given $q_1(0) \in \mathcal{A}_{free}$ and $Q_g \subset \mathcal{A}_{free}$, respectively, design a control trajectory $u : [0, t_f] \to \mathbb{R}^n$, for some finite $t_f > 0$, such that the solution $q^*(t)$ of (1) satisfies $q_1^*(t) \in \mathcal{A}_{free}$, for all $t \in [0, t_f]$, and $q_1^*(t_f) \in Q_g$.*

The feasibility of Problem 1 is established in the following assumption.

Assumption 3. *There exists a (at least twice differentiable) path $\boldsymbol{q}_{\text{p}} : [0, \sigma] \to \mathcal{A}_{\text{free}}$ such that $\boldsymbol{q}_{\text{p}}(0) = q(0)$ and $\boldsymbol{q}_{\text{p}}(\sigma) \in$*

Q_g.

III. Main Results

We present here the proposed solution for Problem 1. Our methodology follows a two-layer approach, consisting of a robust trajectory-tracking control design and a higher-level sampling-based motion planner. Firstly, we design an adaptive control protocol that compensates for the uncertain dynamical parameters of the robot and forces the system to evolve in a funnel around a desired trajectory, whose size can be a priori chosen by the user/designer, and is completely independent from the system dynamics (1). We stress that this constitutes the main difference from our previous work [9], where the derived funnel depends on the bounds of the various dynamic terms and the external disturbances. Secondly, we develop a geometric sampling-based motion planner that uses this funnel to find a collision free trajectory from the initial to the goal configuration. Intuitively, the robust control design helps the motion planner procedure, which does not have to take into account the complete dynamics (1). Section III-A gives some preliminary background on funnel control and provides the control design, while Section III-B provides the motion planner.

A. Control Design

In order to tackle the unknown dynamics of (1) we use the methodology of funnel control [49], [50]. Funnel control aims at achieving containment of a scalar tracking error $e(t)$ in a user-prespecified time-varying set, defined by certain functions of time, as

$$-\rho(t) < e(t) < \rho(t), \ \forall t \geq 0, \tag{2}$$

where $\rho(t)$ denotes a smooth and bounded function of time, with bounded derivatives, that satisfies $\rho(t_0) > |e(t_0)|$ and $\rho(t) > 0$, for all $t \geq t_0$, called funnel function (or performance function in [49]). Fig. 1 illustrates the aforementioned statements. Since the funnel set is user defined a priori, it can be set to converge to an arbitrarily small residual set with speed no less than a prespecified value, e.g., by using the funnel function $\rho(t) \coloneqq (\rho_0 - \rho_\infty)e^{-\lambda t} + \rho_\infty$. The parameter $\rho_\infty \coloneqq \lim_{t\to\infty} \rho(t) > 0$ represents the maximum allowable value of the steady state error and can be set to a value reflecting the resolution of the measurement device, so that the error $e(t)$ practically converges to zero. Moreover, the constant λ determines the decreasing rate of $\rho(t)$ and thus is used to set a lower bound on the convergence rate of $e(t)$. Therefore, the appropriate selection of the function $\rho(t)$ imposes certain transient and steady state performance characteristics on the tracking error $e(t)$. Intuitively, larger λ and small ρ_∞ improve the performance of the system, yielding fast convergence close to zero. Although these constants can be arbitratrily set by a user, their values affect significantly the stress imposed on the system, and hence they should be chosen according to the system's capabilities.

The key point in funnel control is a transformation of the tracking error $e(t)$ that modulates it with respect to the corresponding funnel specifications, encapsulated in the function $\rho(t)$. This is achieved by converting the constrained problem to an unconstrained one via a transformation of the form $\mathsf{T}\left(\frac{e(t)}{\rho(t)}\right)$, where $\mathsf{T} : (-1, 1) \to (-\infty, \infty)$ is a strictly increasing, odd and bijective mapping. Then the funnel specifications are met by simply preserving the boundedness of $\mathsf{T}\left(\frac{e(t)}{\rho(t)}\right)$. Most funnel control schemes do not employ any information on the system dynamics, using a high-gain approach. That is, the control action approaches infinity as the state approaches the funnel boundary, "pushing" thus the system to remain inside the funnel. In this work we extend the funnel control design to apply for the system (1) and the manifold $\mathbb{T}$, and we show how such a design can be used in the motion planning of the uncertain robotic system (1).

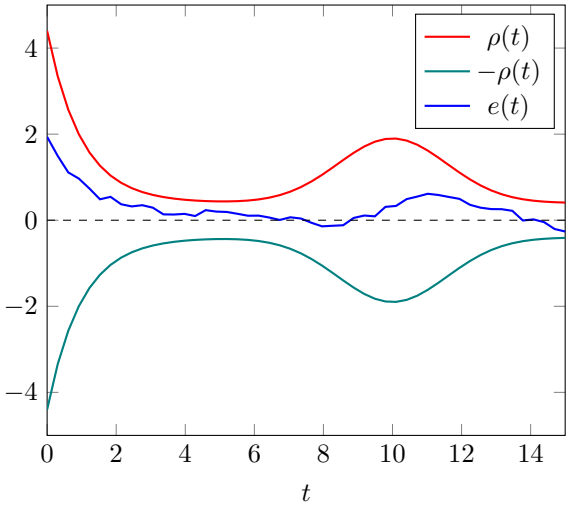


Fig. 1: Illustration of funnel control, where the error $e(t)$ is confined in the prescribed funnel defined by the function $\rho(t)$.

Let $q_\mathrm{d} \coloneqq [(q_\mathrm{d}^\mathrm{t})^\top, (q_\mathrm{d}^\mathrm{r})^\top]^\top \coloneqq [q_{\mathrm{d}_1}^\mathrm{t}, \ldots, q_{\mathrm{d}_{n_{tr}}}^\mathrm{t}, q_{\mathrm{d}_1}^\mathrm{r}, \ldots, q_{\mathrm{d}_{n_r}}^\mathrm{r}]^\top : [t_0, t_0 + t_f] \to \mathbb{T}$ be a smooth (at least k-times continuously differentiable) reference trajectory, with $q_\mathrm{d}^\mathrm{t} \in \mathbb{R}^{n_{tr}}$ and $q_\mathrm{d}^\mathrm{r} \in [0, 2\pi)^{n_r}$ being its translational and rotational parts, respectively. Such a trajectory will be derived by smoothening and adding time constraints to the output path of the sampling-based motion-planning algorithm that will be developed in the next section. Note that the smoothness assumption on q_d is not restrictive, since the smoothening of geometric paths eases the resulting robot motion and is hence common practice in real applications. Nevertheless, we stress that the proposed control algorithm can be applied separately on the raw path segments produced by the local collision-checking planner of the motion-planning algorithm, without requiring any smoothening. This might, however, induce discontinuities on the control algorithm, which might be problematic for robot actuators.

We wish to design the control input u of (1) such that $q(t)$ converges to $q_\mathrm{d}(t)$, despite the unknown terms f_i, g_i. We start by defining the appropriate error metric between q_1 and q_d, which represents their distance. Regarding the translational part, we define the standard Euclidean error $e^\mathrm{t} \coloneqq q^\mathrm{t} - q_\mathrm{d}^\mathrm{t}$. For the rotation part, however, the same error $e^\mathrm{r} \coloneqq q^\mathrm{r} - q_\mathrm{d}^\mathrm{r}$ does not

represent the minimum distance metric, since $q^{\mathfrak{r}}$ evolves on the n_r-dimensional sphere, and its use might cause conservative or infeasible results in the planning layer. Hence, unlike standard control schemes, which drive the Euclidean difference $e^{\mathfrak{r}}$ to zero (e.g., [51]), we use the chordal metric

$$d_C(x) \coloneqq 1 - \cos(x) \in [0, 2], \forall x \in [0, 2\pi),$$

extended for vector arguments $x = [x_1, \dots, x_{\mathfrak{n}}] \in [0, 2\pi)^{\mathfrak{n}}$ to

$$\bar{d}_C(x) \coloneqq \sum_{\ell \in \{1,\dots,\mathfrak{n}\}} d_C(x_j). \tag{3}$$

Note that the chordal metric induces a limitation with respect to tracking on the unit sphere. Consider

$$\eta_\ell^{\mathfrak{r}} \coloneqq d_C(e_\ell^{\mathfrak{r}}) = 1 - \cos(e_\ell^{\mathfrak{r}}),$$

where $e_\ell^{\mathfrak{r}} \coloneqq q_\ell^{\mathfrak{r}} - q_{\mathfrak{d}_\ell}^{\mathfrak{r}}$ is the ℓth element of $e^{\mathfrak{r}}$, $\ell \in \{1, \dots, n_r\}$. Differentiation yields

$$\dot{d}_C(e_\ell^{\mathfrak{r}}) = \sin(e_\ell^{\mathfrak{r}}) \dot{e}_\ell^{\mathfrak{r}},$$

for all $\ell \in \{1, \dots, n_r\}$, which is zero when $e_\ell^{\mathfrak{r}} = 0$ or $e_\ell^{\mathfrak{r}} = \pi$. The second case is an undesired equilibrium, which implies that the point $e_\ell^{\mathfrak{r}} = 0$ cannot be stabilized from *all* initial conditions using a continuous controller. This is an inherent property of dynamics on the unit sphere due to topological obstructions [52]. In the following, we devise a control scheme that, except for guaranteeing that $e_\ell^{\mathfrak{r}}(t)$ evolves in a predefined funnel, guarantees that $e_\ell^{\mathfrak{r}}(t) \neq \pi$, for all $t \in (t_0, t_f]$, provided that $e_\ell^{\mathfrak{r}}(t_0) \neq \pi$, for all $\ell \in \{1, \dots, n_r\}$.

The funnel is defined by the functions $\rho_j^{\mathfrak{t}} : [t_0, t_0 + t_f] \to [\underline{\rho}_j^{\mathfrak{t}}, \bar{\rho}_j^{\mathfrak{t}}]$, $\rho_\ell^{\mathfrak{r}} : [t_0, t_0 + t_f] \to [\underline{\rho}_\ell^{\mathfrak{r}}, \bar{\rho}_\ell^{\mathfrak{r}}]$ with initial and final values $\bar{\rho}_j^{\mathfrak{t}}$, $\bar{\rho}_\ell^{\mathfrak{r}}$, and $\underline{\rho}_j^{\mathfrak{t}}$, $\underline{\rho}_\ell^{\mathfrak{r}}$, respectively, i.e.,

$$\rho_j^{\mathfrak{t}}(t_0) = \bar{\rho}_j^{\mathfrak{t}}, \rho_\ell^{\mathfrak{r}}(t_0) = \bar{\rho}_\ell^{\mathfrak{r}} \tag{4a}$$

$$\rho_j^{\mathfrak{t}}(t_0 + t_f) = \underline{\rho}_j^{\mathfrak{t}}, \rho_\ell^{\mathfrak{r}}(t_0 + t_f) = \underline{\rho}_\ell^{\mathfrak{r}} \tag{4b}$$

$$0 < \underline{\rho}_j^{\mathfrak{t}} \leq \bar{\rho}_j^{\mathfrak{t}}, 0 < \underline{\rho}_\ell^{\mathfrak{r}} \leq \bar{\rho}_\ell^{\mathfrak{r}} < 2 \tag{4c}$$

and being consistent with the errors initially, i.e.,

$$|e_j^{\mathfrak{t}}(t_0)| < \bar{\rho}_j^{\mathfrak{t}}, \quad \eta_\ell^{\mathfrak{r}}(t_0) < \bar{\rho}_\ell^{\mathfrak{r}} \tag{4d}$$

for all $j \in \{1, \dots, n_{tr}\}$, $\ell \in \{1, \dots, n_r\}$. The functions $\rho_j^{\mathfrak{t}}$, $\rho_\ell^{\mathfrak{r}}$ are assumed to be smooth, with bounded derivatives, for all $j \in \{1, \dots, n_{tr}\}$, $\ell \in \{1, \dots, n_r\}$. Our aim is to design a control protocol such that

$$|e_j^{\mathfrak{t}}(t)| < \rho_j^{\mathfrak{t}}(t), \quad \forall j \in \{1 \dots, n_{tr}\}, \tag{5a}$$

$$\eta_\ell^{\mathfrak{r}}(t) < \rho_\ell^{\mathfrak{r}}(t), \quad \forall \ell \in \{1 \dots, n_r\}, \tag{5b}$$

for all $t \in [t_0, t_0 + t_f]$. Note that, since $\bar{\rho}_{\mathfrak{r}_\ell} < 2$, guaranteeing (5b) ensures that $\eta_\ell^{\mathfrak{r}}(t) < 2$, i.e., $e_\ell^{\mathfrak{r}}(t) \neq \pi$, for all $t \in [t_0, t_0 + t_f], \ell \in \{1, \dots, n_r\}$ and avoidance of the respective singularity. The funnel functions can be defined a priori by a user, specifying the performance of the system in terms of overshoot and steady-state value of the errors $e_j^{\mathfrak{t}}$, $e_\ell^{\mathfrak{r}}$. For instance, for the exponentially decaying $\rho_j^{\mathfrak{t}}(t) = (\bar{\rho}_j^{\mathfrak{t}} - \underline{\rho}_j^{\mathfrak{t}}) \exp(-\lambda_j t) + \underline{\rho}_j^{\mathfrak{t}}$, $\forall t \in [t_0, t_0 + t_f]$, a user can choose the constants $\underline{\rho}_j^{\mathfrak{t}}$, $\bar{\rho}_j^{\mathfrak{t}}$, λ_j dictating the maximum error steady-state value, overshoot, and speed of convergence. The only hard condition is property (4d) above, stating that the errors needs to respect the funnel constraints initially. Note also that the funnel functions do not depend on the robot dynamics, and can converge to values $\underline{\rho}_j^{\mathfrak{t}}$, $\underline{\rho}_\ell^{\mathfrak{r}}$ arbitrarily close to zero at $t_0 + t_f$, achieving thus practical stability. We provide more details on the choice of the funnels after the control-design algorithm, which is described next.

Let us define first the normalized errors as

$$\xi_j^{\mathfrak{t}} \coloneqq \frac{e_j^{\mathfrak{t}}}{\rho_j^{\mathfrak{t}}}, \quad \forall j \in \{1 \dots, n_{tr}\}, \tag{6a}$$

$$\xi_\ell^{\mathfrak{r}} \coloneqq \frac{\eta_\ell^{\mathfrak{r}}}{\rho_\ell^{\mathfrak{r}}}, \quad \forall \ell \in \{1 \dots, n_r\}. \tag{6b}$$

Note that, in order for the errors $e_j^{\mathfrak{t}}$ and $\eta_\ell^{\mathfrak{r}}$ to satisfy the funnel constraints, the control design must guarantee that $\xi_j^{\mathfrak{t}} \in (-1, 1)$ and $\xi_\ell^{\mathfrak{r}} \in [0, 1)$. In order to do that, we define the transformed errors and signals

$$\varepsilon_j^{\mathfrak{t}} \coloneqq \ln\left(\frac{1 + \xi_j^{\mathfrak{t}}}{1 - \xi_j^{\mathfrak{t}}}\right), \quad \forall j \in \{1 \dots, n_{tr}\}, \tag{7a}$$

$$\varepsilon_\ell^{\mathfrak{r}} \coloneqq \ln\left(\frac{1}{1 - \xi_\ell^{\mathfrak{r}}}\right), \quad \forall \ell \in \{1 \dots, n_r\}, \tag{7b}$$

$$r_j^{\mathfrak{t}} \coloneqq \frac{\partial \varepsilon_j^{\mathfrak{t}}}{\partial \xi_j^{\mathfrak{t}}} = \frac{2}{1 - (\xi_j^{\mathfrak{t}})^2}, \quad \forall j \in \{1 \dots, n_{tr}\}, \tag{7c}$$

$$r_\ell^{\mathfrak{r}} \coloneqq \frac{\partial \varepsilon_\ell^{\mathfrak{r}}}{\partial \xi_\ell^{\mathfrak{r}}} = \frac{1}{1 - \xi_\ell^{\mathfrak{r}}}, \quad \forall \ell \in \{1 \dots, n_r\}. \tag{7d}$$

Note that $\varepsilon_j^{\mathfrak{t}}$, $r_j^{\mathfrak{t}}$, and $\varepsilon_\ell^{\mathfrak{r}}$, $r_\ell^{\mathfrak{r}}$ diverge to infinity as $\xi_j^{\mathfrak{t}}$ and $\xi_\ell^{\mathfrak{r}}$ approach 1, respectively. The control design exploits this property; it aims to keep these signals bounded in order to achieve $\xi_j^{\mathfrak{t}}(t) \in (-1, 1)$ and $\xi_\ell^{\mathfrak{r}}(t) \in [0, 1)$. We now proceed with a back-stepping methodology [43]. Since q_2 is part of the system state and cannot be designed, we set a desired reference signal that we want q_2 to track. In particular, we define the reference signal for q_2 as $\alpha_1 \coloneqq [(\alpha^{\mathfrak{t}})^\top, (\alpha^{\mathfrak{r}})^\top]^\top$, with

$$\alpha^{\mathfrak{t}} \coloneqq -\begin{bmatrix} \frac{k_1^{\mathfrak{t}}}{\rho_1^{\mathfrak{t}}} r_1^{\mathfrak{t}} \varepsilon_1^{\mathfrak{t}}, & \dots, & \frac{k_{n_{tr}}^{\mathfrak{t}}}{\rho_{n_{tr}}^{\mathfrak{t}}} r_{n_{tr}}^{\mathfrak{t}} \varepsilon_{n_{tr}}^{\mathfrak{t}} \end{bmatrix}^\top \tag{8a}$$

$$\alpha^{\mathfrak{r}} \coloneqq -\begin{bmatrix} \frac{k_1^{\mathfrak{r}}}{\rho_1^{\mathfrak{r}}} r_1^{\mathfrak{r}} \sin(e_1^{\mathfrak{r}}), & \dots, & \frac{k_{n_r}^{\mathfrak{r}}}{\rho_{n_r}^{\mathfrak{r}}} r_{n_r}^{\mathfrak{r}} \sin(e_{n_r}^{\mathfrak{r}}) \end{bmatrix}^\top \tag{8b}$$

where $k_j^{\mathfrak{t}}$, $k_\ell^{\mathfrak{r}}$ are positive gain constants, $j \in \{1, \dots, n_{tr}\}$, $\ell \in \{1, \dots, n_r\}$.

The rest of the algorithm proceeds recursively: for $i \in \{2, \dots, k\}$, we define the error

$$e_i \coloneqq \begin{bmatrix} e_{i_1}, \dots, e_{i_n} \end{bmatrix}^\top \coloneqq q_i - \alpha_{i-1} \in \mathbb{R}^n, \tag{9}$$

where α_{i-1} will be given subsequently in (11). We design funnel functions $\rho_{i_m} : [t_0, t_0 + t_f] \to [\underline{\rho}_{i_m}, \bar{\rho}_{i_m}]$, $\underline{\rho}_{i_m} \leq \bar{\rho}_{i_m}$, such that $\rho_{i_m}(t_0) = \bar{\rho}_{i_m} > |e_{i_m}(t_0)|$[1], for all $m \in \{1, \dots, n\}$, and define

$$\xi_i \coloneqq \begin{bmatrix} \xi_{i_1}, \dots, \xi_{i_n} \end{bmatrix}^\top \coloneqq \rho_i^{-1} e_i, \tag{10a}$$

[1]Note that $e_{i_m}(t_0)$ can be measured at the time instant t_0 and the functions ρ_{i_m} can be designed accordingly.

$$\varepsilon_i \coloneqq \left[\varepsilon_{i_1},\ldots,\varepsilon_{i_n}\right]^\top \coloneqq \left[\ln\left(\tfrac{1+\xi_{i_1}}{1-\xi_{i_1}}\right),\ldots,\ln\left(\tfrac{1+\xi_{i_n}}{1-\xi_{i_n}}\right)\right]^\top \tag{10b}$$

$$r_i \coloneqq \operatorname{diag}\left\{\left[\frac{\partial \varepsilon_{i_m}}{\partial \xi_{i_m}}\right]_{m\in\{1,\ldots,n\}}\right\}, \tag{10c}$$

where $\rho_i \coloneqq \operatorname{diag}\{[\rho_{i_m}]_{i\in\{1,\ldots,n\}}\} \in \mathbb{R}^{n\times n}$. Finally, we design the intermediate reference signals as

$$\alpha_i \coloneqq -K_i\rho_i^{-1}r_i\varepsilon_i, \forall i \in \{2,\ldots,k-1\}, \tag{11}$$

and the control law

$$u = -K_k\rho_k^{-1}r_k\varepsilon_k, \tag{12}$$

where $K_i \in \mathbb{R}^{n\times n}$, $i \in \{2,\ldots,k\}$, are positive definite z matrices.

Note that the funnels defined by the functions ρ_i are defined initially (at $t = t_0$) since they only require the measured value of $e_i(t_0)$ in order to guarantee $\rho_{i_m}(t_0) > |e_{i_m}(t_0)|$, for $m \in \{1,\ldots,n\}$; the type and structure of the functions ρ_i, however, can be determined a priori by a user. A standard choice is exponentially decaying $\rho_{i_m}(t) = (\bar{\rho}_{i_m} - \underline{\rho}_{i_m})\exp(-\lambda_{i_m}t) + \underline{\rho}_{i_m}$, or constant ones $\rho_{i_m}(t) = \bar{\rho}_{i_m}$, with the rule $\rho_{i_m}(t_0) = \bar{\rho}_{i_m} = |e_{i_m}(t_0)| + \alpha$, for some $\alpha > 0$. Finally, although tight funnels $\rho_j^{\mathfrak{t}}$, $\rho_\ell^{\mathfrak{r}}$ might be desired to achieve close proximity of $q_1(t)$ to $q_{\mathrm{d}}(t)$, the funnels defined by ρ_i are only required to be bounded; convergence to very small values (e.g., by choosing very small values $\underline{\rho}_{i_m}$ for exponentially-decaying funnels) does not have a direct physical interpretation in the system's configuration space and can overstress the system causing unnecessarily large control inputs.

Remark 1. *The control algorithm* (6)-(12) *resembles the function of reciprocal barriers used in optimization. That is, the intermediate reference and control signals* (8), (11), (12) *approach infinity as the errors* $|e_j^{\mathfrak{t}}|$, $1 - e_\ell^{\mathfrak{r}}$, $|e_{i_m}|$, $i \in \{2,\ldots,k-1\}$ *approach the respective funnel functions* $\rho_j^{\mathfrak{t}}$, $\rho_\ell^{\mathfrak{r}}$, ρ_{i_m}, $j \in \{1,\ldots,n_{tr}\}$, $\ell \in \{1,\ldots,n_r\}$, $m \in \{1,\ldots,n\}$, $i \in \{2,\ldots,k\}$. *Intuitively, this forces these errors to remain inside their respective funnels, by compensating for the unknown dynamic terms of* (1), *which are assumed to be continuous and hence bounded in these funnels. In addition, note that the control algorithm does not use any information on the state- and time-dependent system dynamics* $f_i(\cdot)$, $g_i(\cdot)$, *giving rise to two important properties; firstly, it can be easily applied to a large variety of systems with different dynamic parameters; secondly, it is robust against unknown, possibly adversarial, time-varying disturbances. The latter is clearly illustrated in the performed experiments of Section IV.*

Remark 2 (Control gain selection and control input bounds)**.** *The control gain matrices* $K^{\mathfrak{t}}$, $K^{\mathfrak{r}}$, K_i, $i \in \{2,\ldots,k\}$ *are chosen by the user and can be any positive definite matrices. It should be noted, however, that their choice affects both the quality of evolution of the errors inside the funnel envelopes as well as the control input characteristics (e.g., decreasing the gain values leads to increased oscillatory behavior within, which is improved when adopting higher values, enlarging, however, the control effort both in magnitude and rate). Additionally, fine tuning might be needed in real-time scenarios, to retain the required control input signals within the feasible range that can be implemented by the actuators. In fact, by following the proof of correctness of the proposed control algorithm (in the Appendix), we can derive expressions connecting the control input magnitude with the control gains. More specifically, the proof of Theorem 1 provides positive constants* $\bar{\varepsilon}$, $\bar{\xi}^{\mathfrak{t}} < 1$, $\bar{\xi}^{\mathfrak{r}} < 1$, $\bar{\varepsilon}_i$, $\bar{\xi}_i < 1$ *that satisfy* $|\varepsilon_j^{\mathfrak{t}}(t)| \le \bar{\varepsilon}$, $\varepsilon_\ell^{\mathfrak{r}}(t) \le \bar{\varepsilon}$, $|\xi_j^{\mathfrak{t}}(t)| \le \bar{\xi}^{\mathfrak{t}}$, $|\xi_\ell^{\mathfrak{r}}(t)| \le \bar{\xi}^{\mathfrak{r}}$, $\|\varepsilon_i(t)\| \le \bar{\varepsilon}_i$, $\|\xi_i(t)\| \le \bar{\xi}_i$, *for all* $t \ge t_0$, $j \in \{1,\ldots,n_{tr}\}$, $\ell \in \{1,\ldots,n_r\}$, $i \in \{2,\ldots,k\}$. *Subsequently, according to* (7) *and* (10)*, we can derive positive constants* $\bar{r}^{\mathfrak{t}}$, $\bar{r}^{\mathfrak{r}}$, $\bar{r}_i$ *such that* $r_j^{\mathfrak{t}}(t) \le \bar{r}^{\mathfrak{t}}$, $r_\ell^{\mathfrak{r}}(t) \le \bar{r}^{\mathfrak{r}}$, $\|r_i(t)\| \le \bar{r}_i$, *for all* $t \ge t_0$, $j \in \{1,\ldots,n_{tr}\}$, $\ell \in \{1,\ldots,n_r\}$, $i \in \{2,\ldots,k\}$. *Moreover, in view of* (12)*, it holds that* $\|u(t)\| \le K_k\bar{\rho}\bar{r}_k\bar{\varepsilon}_k$, *for all* $t \ge t_0$, *where* $\bar{\rho}$ *is the upper bound of* $\|\rho_k^{-1}\|$. *Consequently,* K_k *can be tuned in order to achieve* $\|u(t)\| \le \bar{u}$ *for some pre-defined saturation bound* $\bar{u}$*; an explicit derivation can be found in [53]. Nevertheless, it should be noted that the aforementioned constants involve upper bounds of the unknown dynamic terms* $f_i(\cdot)$, $g_i(\cdot)$ *in the part of the state space where* $|\xi_j^{\mathfrak{t}}| < 1$, $\xi_r^{\mathfrak{r}} < 1$, $|\xi_{i_m}| < 1$, $j \in \{1,\ldots,n_{tr}\}$, $\ell \in \{1,\ldots,n_r\}$, $i \in \{2,\ldots,k\}$, $m \in \{1,\ldots,n\}$ *(i.e., inside the respective funnels). Therefore, such bounds must be known for the potential confinement* $\|u(t)\| \le \bar{u}$. *In the same spirit, potential known parts of the dynamics can be leveraged and used in the control design; such parts will then be explicitly cancelled in the closed-loop system, which is expected to yield a smoother control-input trajectory and making it easier to enforce explicit bounds on the control input. In case the dynamics or their bounds are unknown, however, the problem of guaranteeing containment in a user-defined funnel for a system with unknown high-order dynamics while at the same time complying with explicit control-input bounds is significantly challenging. Potential solutions include on-the-fly relaxation of the funnel functions* $\rho_j^{\mathfrak{t}}$, $\rho_\ell^{\mathfrak{r}}$, ρ_i, $j \in \{1,\ldots,n_{tr}\}$, $\ell \in \{1,\ldots,n_r\}$, $i \in \{2,\ldots,k\}$, *or modifications of the reference trajectory* q_{d}. *Finally, the discrete nature of the microcontroller units of robot actuators prevents the continuous application of the control law* (12)*, which might hinder the performance of the overall scheme. Therefore, tuning of the control gains towards optimal performance should be performed off-line or using a simulator.*

The next theorem guarantees the correctness of the proposed protocol.

Theorem 1. *Let the dynamics* (1) *as well as prescribed funnels* $\rho_j^{\mathfrak{t}}$, $j \in \{1,\ldots,n_{tr}\}$, $\rho_\ell^{\mathfrak{r}}$, $\ell \in \{1,\ldots,n_r\}$ *satisfying the prescribed initial constraints* (4). *Then the control protocol* (6)-(12) *guarantees that*

$$|e_j^{\mathfrak{t}}(t)| < \rho_j^{\mathfrak{t}}(t), \forall j \in \{1,\ldots,n_{tr}\}, \tag{13a}$$

$$\eta_\ell^{\mathfrak{r}}(t) = 1 - \cos(e^{\mathfrak{r}}(t)) < \rho_\ell^{\mathfrak{r}}(t), \forall \ell \in \{1,\ldots,n_r\} \tag{13b}$$

as well as the boundedness of all closed loop signals, for all $t \in [t_0, t_0 + t_f]$.

Proof. The proof is given in the Appendix. □

Remark 3 (Funnel properties)**.** *Theorem 1 establishes a funnel around the desired trajectory q_d where the state $q(t)$ will evolve in. This funnel will be used as clearance in the motion planner of the subsequent section to derive a collision-free path to the goal region. We stress that this funnel can be a priori chosen by a user, in contrast to our previous work [39], where the corresponding funnel depends on the system's dynamic terms that are unknown to the user. The only hard constraint is the one imposed by* (4d) *at t_0, i.e., $|e_j^{\mathfrak{t}}(t_0)| < \bar{\rho}_j^{\mathfrak{t}}$, $\eta_\ell^{\mathfrak{r}}(t_0) < \bar{\rho}_\ell^{\mathfrak{r}}$, for $j \in \{1,\dots,n_{tr}\}$, $\ell \in \{1,\dots,n_r\}$. Note, however, that the collision-free geometric trajectory q_d of the motion planner will connect the initial condition $q(0)$ to the goal and hence it is reasonable to enforce $q_{\mathrm{d}}(0) = q(0)$, which implies that the aforementioned constraint is trivially satisfied. Moreover, the selection of $\rho_j^{\mathfrak{t}}$, $\rho_\ell^{\mathfrak{r}}$ can be chosen such that the respective funnels are arbitrarily small implying that the system evolves arbitrarily close to the derived trajectory q_d. It should be noted, nevertheless, that too shrunk or very fast-converging funnels might yield excessive control inputs that cannot be realized by the actuators in realistic systems. Therefore, the funnel characteristics must be always chosen in accordance to the capabilities of the system. Similarly to the control gains (see Remark 2), the funnel characteristics can be explicitly connected to the system's control input, requiring, however, upper bounds of the unknown dynamics $f_i(\cdot)$, $g_i(\cdot)$. Hence, tuning can be attempted off-line or using a simulator. As an example, in our results of Section IV, where we perform computer simulations and hardware experiments using 6-DOF robotic manipulators, we choose the funnels as follows. For the computer simulations, we choose $\rho_j^{\mathfrak{t}}(t) = 0.05\exp(-0.01t) + 0.1$, $\rho_j^{\mathfrak{r}}(t) = 0.005\exp(-0.01t) + 0.005$, implying shrinking funnels from 0.1 to 0.05 and from 0.01 to 0.005 (rad), respectively, with exponential convergence dictated by $\exp(-0.01t)$, while for the hardware experiments we choose constant funnels ranging from 0.2 to 0.4.*

B. Motion Planner

We introduce now the framework of *KinoDynamic motion planning via Funnel control*, or *KDF* motion-planning framework; The framework uses the control design of Section III-A to augment geometric sampling-based motion-planning algorithms and solve the kinodynamic motion-planning problem.

Before presenting the framework, we define the extended-free space, which will be used to integrate the results from the feedback control of the previous subsection. In order to do that, we define first the open polyhedron as

$$\begin{aligned}\mathcal{P}(\mathsf{z},\bar{\rho}) \coloneqq \{\mathsf{y}\in\mathbb{T} :& |\mathsf{y}_j^{\mathfrak{t}} - \mathsf{z}_j^{\mathfrak{t}}| < \bar{\rho}_j^{\mathfrak{t}},\\ & 1-\cos(\mathsf{y}_\ell^{\mathfrak{r}} - \mathsf{z}_\ell^{\mathfrak{r}}) < \bar{\rho}_\ell^{r},\\ & \forall j\in\{1,\dots,n_{tr}\}, \ell\in\{1,\dots,n_r\}\} \end{aligned} \tag{14}$$

where $\mathsf{y},\mathsf{z}\in\mathbb{T}$ consist of translational and rotational terms (similarly to q_1), and $\bar{\rho} \coloneqq [\bar{\rho}_1^{\mathfrak{t}},\dots,\bar{\rho}_{n_{tr}}^{\mathfrak{t}},\bar{\rho}_1^{\mathfrak{r}},\dots,\bar{\rho}_{n_r}^{\mathfrak{r}}]^\top \in \mathbb{R}^{n_{tr}+n_r}$ is the vector of maximum funnel values. We define now, similarly to [9], the $\bar{\rho}$-extended free space

$$\bar{\mathcal{A}}_{\text{free}}(\bar{\rho}) \coloneqq \{\mathsf{z}\in\mathbb{T} : \bar{\mathcal{A}}(\mathsf{z},\bar{\rho})\cap\mathcal{O} = \emptyset\}, \tag{15}$$

where $\bar{\mathcal{A}}(\mathsf{z},\bar{\rho}) \coloneqq \bigcup_{\mathsf{y}\in\mathcal{P}(\mathsf{z},\bar{\rho})}\mathcal{A}(\mathsf{y})$. Note that, for vectors ρ_1 and $\rho_2\in\mathbb{R}^{n_{tr}+n_r}$, with $\rho_1 \succeq \rho_2$, with $\succeq$ denoting element-wise inequality, it holds that $\bar{\mathcal{A}}_{\text{free}}(\rho_1)\subseteq\bar{\mathcal{A}}_{\text{free}}(\rho_2)$.

In addition, we need a distance metric that captures accurately the proximity of $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})\subset\mathbb{T}$. As elaborated in the previous section, the Euclidean distance is not an appropriate distance metric in $\mathbb{T}$ due to the fact that the rotational part $\mathfrak{r}$ evolves on the n_r-dimensional sphere. Hence, having already defined the chordal metric $\bar{d}_C$ in (3), we define a suitable distance metric for vectors $x = [(x^{\mathfrak{t}})^\top,(x^{\mathfrak{r}})^\top]^\top$, $y = [(y^{\mathfrak{t}})^\top,(y^{\mathfrak{r}})^\top]^\top\in\mathbb{T}$ as $d_{\mathbb{T}}:\mathbb{T}^2\to\mathbb{R}_{\geq 0}$, with

$$d_{\mathbb{T}}(x,y) = \|x^{\mathfrak{t}} - y^{\mathfrak{r}}\|^2 + \bar{d}_C(x^{\mathfrak{r}} - y^{\mathfrak{r}}). \tag{16}$$

The intuition behind the KDF framework is as follows. The control scheme of the previous subsection guarantees that the robot can track a trajectory within the bounds (13). In other words, given a desired trajectory signal $q_{\mathrm{d}} : [t_0, t_0 + t_f] \to \mathbb{T}$, the control algorithm (6)-(12) guarantees that $q_1(t)\in\mathcal{P}(q_{\mathrm{d}}(t),\bar{\rho})$, for all $t\in[t_0,t_0+t_f]$. Therefore, by the construction of $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, if $q_{\mathrm{d}}(t)$ belongs to $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, $q_1(t)$ belongs to $\mathcal{A}_{\text{free}}$. The proposed sampling-based framework aims at finding a path in $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$; this path will be then endowed with time constraints in order to form the trajectory $q_{\mathrm{d}} : [t_0,t_0+t_f]\to\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, which will then safely tracked by the system using the designed controller.

Common geometric sampling-based motion-planning algorithms follow a standard iterative procedure that build a discrete network $\mathcal{G} = (\mathcal{V},\mathcal{E})$, (tree, roadmap) of points in the free space connecting the initial configuration to the goal; $\mathcal{V}$ and $\mathcal{E}$ denote the nodes (points) and edges, respectively, of the network. Standard functions involved in such algorithms include $\mathsf{Sample}()$, $\mathsf{Nearest}(\mathcal{G},\mathsf{y})$, $\mathsf{Closest}(\mathcal{G},\mathsf{y},K)$, $\mathsf{Steer}(\mathsf{y},\mathsf{z})$, and $\mathsf{ObstacleFree}(\mathsf{y},\mathsf{z})$; $\mathsf{Sample}()$ samples a random point from a distribution in $\mathcal{A}_{\text{free}}$; $\mathsf{Nearest}(\mathcal{G},\mathsf{y})$ and $\mathsf{Closest}(\mathcal{G},\mathsf{y},K)$ find the closest and K closest, respectively, nodes of $\mathcal{G}$ to y, according to some distance metric; $\mathsf{Steer}(\mathsf{y},\mathsf{z})$ computes a point lying on line from z to y and $\mathsf{ObstacleFree}(\mathsf{y},\mathsf{z})$ checks whether the path from y to z belongs to the free space $\mathcal{A}_{\text{free}}$ (i.e., collision-free).

The framework we propose in this work *modifies* the functions $\mathsf{Sample}()$ and $\mathsf{ObstacleFree}(\mathsf{y},\mathsf{z})$, which constitute the main building blocks of all sampling-based motion-planning algorithms. Consequently, we create a basis for a new class of sampling-based motion-planning algorithms that, in combination with the funnel controller of Section III-A, are able to solve the *kinodynamic* motion-planning problem only by sampling in an extended free *geometric configuration* space, without using any information on the system dynamics or resorting to sampling of control inputs.

As stated before, we aim to find a path in the extended free space $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$. To this end, we need to sample points and perform collision checking in $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$. Therefore, we define the functions $\mathsf{SampleExt}(\bar{\rho})$ and $\mathsf{ObstacleFreeExt}(\mathsf{y},\mathsf{z},\bar{\rho})$; $\mathsf{SampleExt}(\bar{\rho})$ samples a point from a uniform distribution in the extended free space $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$; $\mathsf{ObstacleFreeExt}(\mathsf{y},\mathsf{z},\bar{\rho})$ checks whether the path $X_{\text{Line}} : [0,\sigma]\to\mathbb{T}$, for some positive σ, from y to z is collision free with respect to the extended

free space, i.e., check whether $\mathsf{y}' \in \bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, $\forall \mathsf{y}' \in X_{\text{Line}}$. We elaborate on the collision checking procedure in Remark 6.

The new functions $\mathsf{SampleExt}(\cdot)$ and $\mathsf{ObstacleFreeExt}(\cdot)$ can be used in any geometric sampling-based motion planning algorithm, giving thus rise to a new family of algorithms, which produce a safe path in an extended free space $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$. This path is then tracked by the system using the control algorithm of Section III-A. Note, however, that the control algorithm guarantees tracking of a *time-varying* smooth (at least k-times continuously differentiable) trajectory $q_{\text{d}}(t)$, whereas the output of the respective motion planning algorithm is a path, i.e., a sequence of points in $\mathbb{T}$. Therefore, we smoothen this path and endow it with time constraints, producing hence a time-varying trajectory. The aforementioned steps, namely the family of geometric sampling-based motion planning algorithms in $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, the addition of time constraints, and the funnel-control algorithm of Section III-A, constitute the framework of *KinoDynamic motion planning via Funnel control*, or *KDF* motion-planning (KDF-MP) framework. Further, note that the smoothening of the output path is not required to be performed by the sampling-based algorithm; it is required by the overall KDF framework due to to the smooth-trajectory requirement of the funnel controller. We describe in Remark 4 how such a restriction can be relaxed.

The vector $\bar{\rho}$, which forms the extended free space $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$ (see (15)), is the connection of the KDF-MP algorithms to the control module of Section III-A and can be chosen by the user. Intuitively, smaller funnel values lead to a larger extended free space $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$ (note that if $\bar{\rho}$ consists of zeros, then $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho}) = \mathcal{A}_{\text{free}}$), giving the chance to navigate through potential narrow passages or with larger distance from the obstacles. Moreover, note that the goal Q_g must belong to $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$. In view of Assumption 3, Q_g belongs to the open set $\mathcal{A}_{\text{free}}$. Therefore, by invoking continuity properties of the free space, we conclude that there exists a $\rho_\varepsilon \in \mathbb{R}^{n_{tr}+n_r}$ such that $Q_g \in \bar{\mathcal{A}}_{\text{free}}(\rho_\varepsilon)$[2]. Hence, by choosing $\bar{\rho}$ such that $\rho_\varepsilon \succeq \bar{\rho}$, one can achieve $\bar{\mathcal{A}}_{\text{free}}(\rho_\varepsilon) \subseteq \bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$ and hence $Q_g \in \bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$. As stressed before, however, tight funnels might need excessively large control inputs that might not be realizable by real actuators. Therefore, one must take into account the capabilities of the system when choosing $\bar{\rho}$ and the funnel functions of Section III-A, as mentioned in Remark 3. If it is not possible to select $\bar{\rho}$ such that $\rho_\varepsilon \succeq \bar{\rho}$ (e.g., if the goal Q_g is too close to an obstacle), then one can consider a new goal Q_g' that is close to Q_g and belongs to $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$. That is, $Q_g' := \arg\min_{q \in \mathbb{A}} d_{\mathbb{T}}(q, Q_g)$, where $\mathbb{A}$ is a compact subset of $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$ for a chosen $\bar{\rho}$. The probabilistic completeness of the KDF-MP algorithms follows from that of their original counterparts (see [39]).

The control protocol of Section III-A guarantees tracking of a *time-varying* smooth (at least k-times continuously differentiable) trajectory $q_{\text{d}}(t)$, whereas the output of a KDF-MP algorithm is a path, i.e., a sequence of points in $\mathbb{T}$. Therefore, we endow the latter with a time behavior, as follows.

The output path is first converted to a smooth (at least k-times continuously differentiable) one. This is needed to smoothly interpolate the connecting points of the consecutive edges of the solution path that is obtained from a KDF-MP algorithm. This smoothening procedure must be performed in accordance to the extended free space $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, so that the smoothed path still belongs in $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$. Time constraints are then enforced on the smooth path to create a timed trajectory $q_{\text{d}} : [0, t_f] \to \bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, for some $t_f > 0$, which is then given to the control protocol of Section III-A as the desired trajectory input. Note that $q_{\text{d}}(0)$ must satisfy the funnel constraints (4).

Remark 4. *It is also possible to use the output path of the KDF motion-planning algorithm without any post-processing steps, i.e., the raw segments that correspond to the edges of the respective data structure (tree, graph). Each one of these segments can be endowed with time constraints, as well as separate funnel functions. The control algorithm of Section III-A is then applied separately for these segments, possibly with discontinuities at the connecting points. Although one avoids the use of post-processing steps on the output path, such discontinuities might be problematic for the actuators and might jeopardize the safety of the system.*

Remark 5. *Note that the duration of the resulting trajectory t_f, and hence the respective velocity $\dot{q}_{\text{d}}$ can be a priori chosen by a user and hence the robotic system can execute the path in a predefined time interval. There is also no constraint on this duration, since the control protocol of Section III-A guarantees funnel confinement with respect to any arbitrarily fast time trajectory. Nevertheless, the physical limits of the system's actuators prevent the achievement of any time trajectory, and the latter should be properly defined in accordance to any such limits, similarly to the selection of the control gains and the funnel characteristics (see Remarks 2 and 3).*

Algorithm 1 provides the overall KDF framework, including the KDF-MP algorithm, the conversion to a time-varying trajectory, and the application of the funnel control algorithm of Section III-A. The algorithm extracts first the bounds $\bar{\rho} = [\bar{\rho}_1^{\mathfrak{t}}, \ldots, \bar{\rho}_{n_{tr}}^{\mathfrak{t}}, \bar{\rho}_1^{\mathfrak{r}}, \ldots, \bar{\rho}_{n_r}^{\mathfrak{r}}]^\top$ (line 2) which are used in a KDF-MP algorithm (e.g., an appropriately modified RRT or PRM), along with the free space $\mathcal{A}_{\text{free}}$, and other potential arguments such as the goal Q_g or a desired number of nodes N (3). The output path is converted to a smooth time-varying trajectory with the desired duration t_f (line 4), which is then tracked by the system using the funnel control algorithm (line 5).

Algorithm 1 KDF

Input: $\mathcal{A}_{\text{free}}$, Q_g, N, t_f, k, $q_1(0)$, q_{d}, $\rho_j^{\mathfrak{t}}$, $\rho_\ell^{\mathfrak{r}}$, $K^{\mathfrak{t}}$, $K^{\mathfrak{r}}$, K_i, $j \in \{1, \ldots, n_{tr}\}$, $\ell \in \{1, \ldots, n_r\}$
Output: $u(t)$

1: **procedure** KDF
2: $\bar{\rho} \leftarrow \mathsf{Bounds}(\rho_j^{\mathfrak{t}}, \rho_\ell^{\mathfrak{r}})$;
3: $\mathbf{q}_p \leftarrow \mathsf{KDF-MP}(\bar{\rho}, \mathcal{A}_{\text{free}}, Q_g, q_1(0), N)$;
4: $q_{\text{d}} \leftarrow \mathsf{TimeTraj}(\mathbf{q}_p, t_f)$;
5: $u \leftarrow \mathsf{FunnelControl}(t_f, k, q_{\text{d}}, \rho_j^{\mathfrak{t}}, \rho_\ell^{\mathfrak{r}}, K^{\mathfrak{t}}, K^{\mathfrak{r}}, K_i)$;

[2]Since the free space $\mathcal{A}_{\text{free}}$ and the goal configuration Q_g are known, such a ρ_ε can be explicitly found.

Remark 6 (Collision Checking in $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$). *The proposed feedback control scheme guarantees that $q(t) \in \mathcal{P}(q_{\text{d}}(t), \bar{\rho})$ for any trajectory $q_{\text{d}}(t)$, formed by the sampled points q_s of a KDF-MP algorithm. Therefore, checking whether the points q_s belong to $\mathcal{A}_{free}$, as in standard motion planners [14], is not sufficient. For each such point q_s, one must check whether $z \in \mathcal{A}_{free}$, for all $z \in \mathcal{P}(q_s, \bar{\rho})$, which is equivalent to checking if $q_s \in \bar{\mathcal{A}}_{free}(\bar{\rho})$. For simple robotic structures, such as mobile robots or UAV (see Section IV-A), whose volume can be bounded by convex shapes, one can enlarge the robot's or the obstacles' volume by $\bar{\rho}$ and perform the collision checking procedure in the remaining free space. However, more complex structures, such as robotic manipulators (see Section IV), necessitate a more sophisticated approach, since they can assume nonconvex complex shapes in various configurations. For such systems, there are two procedures one could follow. Firstly, for each q_s, a finite number of points z can be sampled from a uniform distribution in $\mathcal{P}(q_s, \bar{\rho})$ and separately checked for collision. In the experimental results of Section IV, we apply such a procedure for a robotic manipulator with 6 degrees of freedom; in particular, for each sampled point q_s in the free space, we uniformly sample up to 50 more points in $\mathcal{P}(q_s, \bar{\rho})$ and check whether they belong to $\mathcal{A}_{free}$. Note that the more points we sample from $\mathcal{P}(q_s, \bar{\rho})$, the better exploration of $\bar{\mathcal{A}}_{free}(\bar{\rho})$ we achieve, guaranteeing hence that $q_s \in \bar{\mathcal{A}}_{free}(\bar{\rho})$. Many practical applications, however, entail workspace obstacles that have a so-called "fat"-structure. Such obstacles consist of objects without protuberances (excluding, e.g., long and skinny obstacles such as wires, cables and tree branches); a formal definition can be found in [54], which shows that the complexity of the free space is linear when the obstacles are "fat". Therefore, in such cases, sampling a finite number of points in $\mathcal{P}(q_s, \bar{\rho})$ can be considered to be complete, i.e., the resulting path will belong to the extended free space $\bar{\mathcal{A}}_{free}(\bar{\rho})$. The second procedure consists of calculating the convex hull of the robot around the configuration point q_s. First, one calculates the limit poses of each link of the robot around q_s. Such limits are calculated by combining the lower and upper bounds of the joints that affect the link (i.e., joints that come before the link in the articulated robotic structure). These lower and upper bounds are defined by $\bar{\rho}$. After deriving the limit poses, one computes their convex hull, which is expanded by an appropriate constant to yield an over-approximation of the swept volume of the potential motion of the link, as described in [55]. The resulting shape is then checked for collisions for each link separately.*

IV. Experimental Results

This section is devoted to experimental results that validate the theoretical findings. Firstly, we present computer simulation results from the application of the KDF framework to a UAV moving in $\mathbb{R}^3$, as well as a UR5 robot, in obstacle-cluttered environments. We use KDF-RRT as the motion planner and we compare the efficiency with a standard geometric and kinodynamic RRT algorithms.

Secondly, we present experimental results using the KDF framework on a 6DOF HEBI manipulator. We compare the performance of the proposed funnel control algorithm with our previous work [39] as well as a standard PID controller.

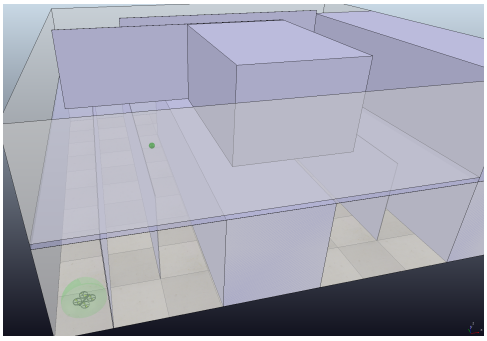


Fig. 2: The obstacle-cluttered 3D workspace and the starting position of the UAV, along with its augmented volume (green sphere around the UAV) to account for $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, and the goal configuration (smaller green sphere).

A. Computer Simulations

We apply here the KDF framework by using a KDF-RRT motion planner and the funnel control algorithm presented in Sections III-B and III-A, respectively, in two computer simulated scenarios by using the CoppeliaSim robotic simulator [38]. In both cases, the KDF-RRT was implemented using the algorithms of the OMPL library [56], which was appropriately interfaced with CoppeliaSim. The control algorithm was implemented via a ROS node in MATLAB environment, communicating with the CoppeliaSim scenes using ROS messages at a frequency of 100Hz. The CoppeliaSim scenes were updated at a frequency of 1kHz.

Unmanned Autonomous Vehicle

The first case consists of a UAV moving in an obstacle-cluttered 3D space, as shown in Fig. 2. In order to comply with the dynamic model of Section II, we view the UAV as a fully actuated rigid body with dynamics

$$\dot{q}_1 = f_1(q_1, t) + g_1(q_1, t) q_2 \tag{17a}$$

$$\dot{q}_2 = f_2(q_1, q_2, t) + g_2(q_1, q_2, t) u \tag{17b}$$

where $q_1 = [q_1^{\text{t}}, q_2^{\text{t}}, q_3^{\text{t}}]^\top$, $q_2 \in \mathbb{R}^3$ are the linear position and velocity, u is the 3D force, acting as the control input, and f_1, g_1, f_2, g_2 are unknown functions satisfying Assumption 1.

The UAV aims to navigate safely to a goal position $Q_g = [-3, -4, 3]^\top$, starting from $q_1(0) = [-4.5, -4.2, 0]^\top$ within 90 seconds and space bounds $(-5, 5)$, $(-5, 5)$, $(0, 4)$ in x-, y-, and z-dimensions respectively (the x- and y- dimensions correspond to the floor dimensions of Fig. 2). For the safe path tracking, we choose the exponentially decaying funnel functions $\rho_1^{\text{t}}(t) = \rho_2^{\text{t}}(t) = \rho_3^{\text{t}}(t) = \rho(t) \coloneqq 0.15 \exp(-0.1t) + 0.05 \in [0.05, 0.2]$ (meters), implying $\bar{\rho} = 0.2[1, 1, 1]^\top$, as well as $\rho_{2_j}(t) \coloneqq (\max\{2|e_{2_j}(0)|, 0.5\} - 0.1) \exp(-0.1t) + 0.1 \in [0.1, \max\{2|e_{2_j}(0)|, 0.5\}]$ (meters/second) for $j \in \{1, \ldots, 3\}$. Hence, the minimum distance from the obstacles and the path output by the KDF-RRT algorithm must be larger than 0.2

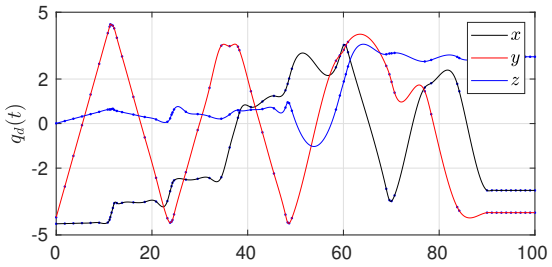


Fig. 3: The 3D (x,y,z) output path of the KDF-RRT algorithm (blue points), along with the smoothened time-varying trajectory, in meters, for the UAV scenario.

meters. In the conducted simulation, this was achieved by enlarging the radius of the UAV volume sphere by 0.2, which was then checked for collision (green sphere in Fig. 2).

The obtained path consists of 50 points in $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$ and is converted to a smooth time trajectory as follows. We construct $q_{\text{d}} : [0, 100] \rightarrow \mathbb{R}^3$, such that $q_{\text{d}}(0) = q_1(0)$ and $q_{\text{d}}(t) = Q_g$, for $t \in [90, 100]$, using a standard fitting procedure. For the construction of $q_{\text{d}}(t)$, each pair of two path points $\mathsf{h} = [\mathsf{h}_1, \mathsf{h}_2, \mathsf{h}_3]^\top$, $\mathsf{w} = [\mathsf{w}_1, \mathsf{w}_2, \mathsf{w}_3]^\top \in \bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$ is endowed with a time duration proportional to their distance, i.e., equal to $\frac{1}{90} \max_{i\in\{1,2,3\}}\{|\mathsf{h}_i - \mathsf{w}_i|\}$. For the specific control and scene update frequencies and chosen funnels, the control gains that yield satisfactory behavior (reasonable control inputs and avoidance of oscillations) were found via offline tuning to be $K^{\text{t}} = \text{diag}\{[k_j^{\text{t}}]_{j\in\{1,2,3\}}\} = 2I_3$ and $K_2 = 35I_3$.

The signals of the resulting motion of the UAV for the two different cases are depicted in Figs. 4 and 5. In particular, Fig. 4 shows the evolution of the errors $e_j^{\text{t}}(t)$, $e_{2_m}(t)$ (in meters and meters/second, respectively) along with the respective funnel functions $\rho_j^{\text{t}}(t)$, $\rho_{2_m}(t)$, It can be verified that the errors always respect the respective funnels, guaranteeing thus the successful execution of the respective trajectories. Moreover, Fig. 5 depicts the distance of the UAV from the obstacles $D_{UAV}(t)$ (in meters) as well as the resulting control inputs $u = [u_1, u_2, u_3]^\top$ for the three spatial dimensions (in Newton). Although the output path was smoothened without taking into account the extended free space $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, the UAV was able to successfully navigate to the goal configuration safely. Moreover, the funnel controller produced reasonable control inputs, without excessive oscillations or magnitude.

UR5 Robotic Manipulator

The second case consists of a UR5 6DOF robotic manipulator, whose dynamics are considered to have the form (17) and whose end-effector aims to sequentially navigate to four points in $\mathbb{R}^6$ (position, orientation), as pictured in Fig. 6. By using inverse kinematics algorithms, we translate these points to desired points for the joint variables of the manipulator, which are then used in a sequential application of the proposed scheme. We consider here that the base joint of the manipulator operates in the unit circle $[0, 2\pi)$, whereas the rest of the joints operate in $[-\pi, \pi] \subset \mathbb{R}$ defined by mechanical and structural limits, resulting in $q_1 = [q_{1_1}, \ldots, q_{1_6}]^\top = [q_1^{\text{t}}, \ldots, q_5^{\text{t}}, q_1^{\text{r}}]^\top$,

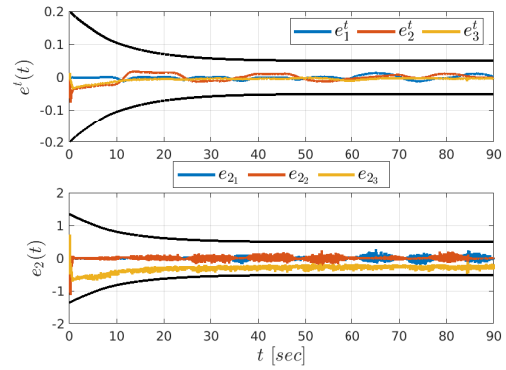


Fig. 4: Top: the evolution of the errors $e_1^{\text{t}}(t)$, $e_2^{\text{t}}(t)$, $e_3^{\text{t}}(t)$ (in meters), along with the funnels $\rho_1^{\text{t}}(t) = \rho_2^{\text{t}}(t) = \rho_3^{\text{t}}(t)$, shown in black, for $t \in [0, 90]$ seconds. Bottom: the evolution of the velocity errors $e_{2_1}(t)$, $e_{2_2}(t)$, $e_{2_3}(t)$ (in meters/second), along with the funnels $\rho_{2_1}(t) = \rho_{2_2}(t) = \rho_{2_3}(t)$, shown in black, for $t \in [0, 90]$ seconds.

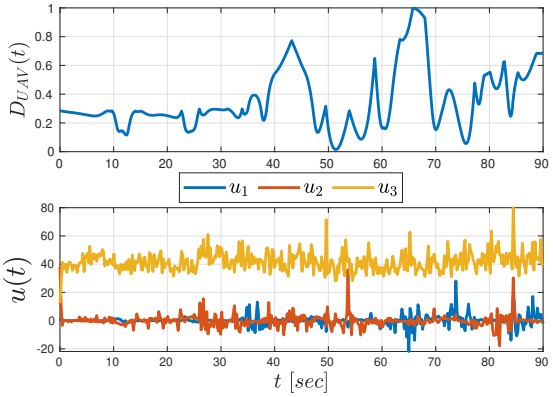


Fig. 5: Top: the distance $D_{UAV}(t)$ (in meters) of the UAV from the obstacles for $t \in [0, 90]$ seconds. Bottom: the evolution of the control inputs $u_1(t), u_2(t), u_3(t)$ (in Newton) for the three spatial dimensions and $t \in [0, 90]$ seconds.

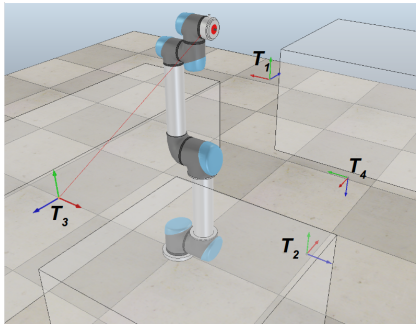


Fig. 6: The initial configuration of the UR5 robot in an obstacle-cluttered environment with four targets.

based on the notation of Section II.

We consider that the robot end-effector has to sequentially navigate from its initial configuration $q_0 = [0, 0, 0, 0, 0, 0]^\top$ to the following four target points (shown in Fig. 6).

- Target 1: $T_1 = [-0.15, -0.475, 0.675]^\top$ and Euler-angle orientation $[\frac{\pi}{2}, 0, 0]^\top$, which yields the configuration $q_{T_1} = [-0.07, -1.05, 0.45, 2.3, 1.37, -1.33]^\top$.
- Target 2: $T_2 = [-0.6, 0, 2.5]^\top$ and Euler-angle orientation $[0, -\frac{\pi}{2}, -\frac{\pi}{2}]^\top$, which yields the configuration $q_{T_2} = [1.28, 0.35, 1.75, 0.03, 0.1, -1.22]^\top$
- Target 3: $T_3 = [-0.025, 0.595, 0.6]^\top$ and Euler-angle orientation $[-\frac{\pi}{2}, 0, \pi]^\top$, which yields the configuration $q_{T_3} = [-0.08, 0.85, -0.23, 2.58, 2.09, -2, 36]^\top$
- Target 4: $T_4 = [-0.525, -0.55, 0.28]^\top$ and Euler-angle orientation $[\pi, 0, -\frac{\pi}{2}]^\top$, which yields the configuration $q_{T_4} = [-0.7, -0.76, -1.05, -0.05, -3.08, 2.37]\top$

The target points where chosen such that they yield increasing difficulty with respect to the navigation path of the robot. The paths for each pair are computed on the fly using the KDF-RRT algorithm, after the manipulator reaches each target. For the safe tracking of the four output paths, we choose the funnel functions such that $\bar{\rho} = 0.01[1, 15, 15, 15, 15, 15]$, as will be elaborated later. Regarding the collision checking in $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$ of KDF-RRT, we check a finite number of samples around each point of the resulting path for collision. We run KDF-RRT with 10 and 50 such samples and we compared the results to a standard geometric RRT algorithm in terms of time per number of nodes. The results for 30 runs of the algorithms are given in Figs. 7-8 for the four paths, in logarithmic scale. One can notice that the average nodes created do not differ significantly among the different algorithms. As expected, however, KDF-RRT requires more time than the standard geometric RRT algorithm, since it checks the extra samples in $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$ for collision. One can also notice that the time increases with the number of samples. However, more samples imply greater coverage of $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$ and hence the respective solutions are more likely to be complete with respect to collisions.

Since, in contrast to the standard geometric RRT, KDF-RRT implicitly takes into account the robot dynamics (17) through the designed tracking control scheme and the respective extended free space $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, we compare the results to a standard kinodynamic RRT algorithm that simulates forward the robot dynamics, assuming known dynamical parameters. In particular, we run the algorithm only for the first two joints, with initial configuration $[0, 0]^\top$ and a randomly chosen goal configuration at $[-\frac{\pi}{18}, \frac{\pi}{4}]^\top$ rad, while keeping the other joints fixed at 0. For the forward simulation of the respective dynamics we chose a sampling step of 10^{-3} sec and total simulation time 30 sec for each constant control input. The termination threshold distance was set to 0.25 (with respect to the distance $d_{\mathbb{T}}$), i.e., the algorithm terminated when the forward simulation reached a configuration closer than 0.25 units to the goal configuration. The results for 10 runs of the algorithm are depicted in Fig. 9, which provides the execution time and number of nodes created in logarithmic scale. Note that, even for this simple case (planning for only two joints), the execution time is comparable to the KDF-RRT case of 50 samples in the fourth path scenario $q_3 \to q_4$. Running the kinodynamic RRT for more than two joints resulted in unreasonably large execution times (more than 1 hour) and hence they are not included in the results. This can be attributed to the randomized inputs and complex robot dynamics; the bias-free random sample of constant inputs and forward simulation of the complex robot dynamics requires a significant amount of time to sufficiently explore the 12-dimensional state space. One may argue that other accelerated algorithms can be used (e.g., BIT [57]), however it is not trivial to reduce such long running time.

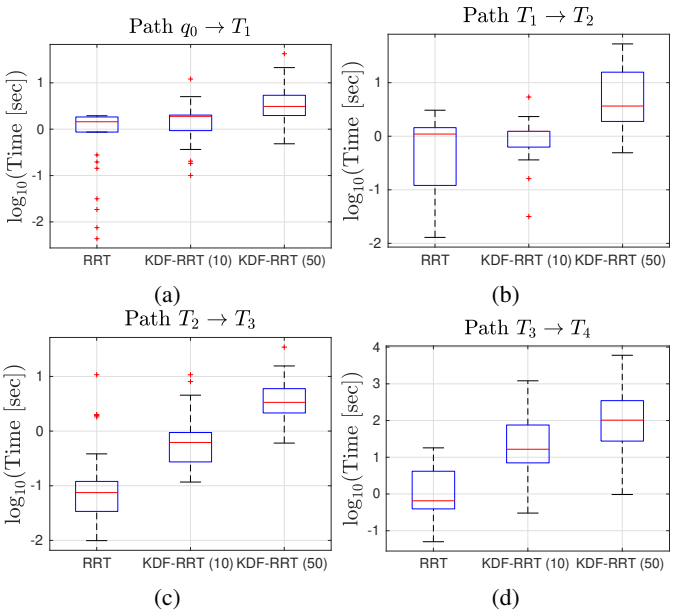


Fig. 7: Box plots showing the execution time of the three algorithms (in logarithmic scale) for the four paths; '+' indicate the outliers.

Next, we illustrate the motion of the robot through the target points via the control design of Section III-A. For each sub-path $T_i \to T_{i+1}$, with $T_0 = q_0$ we fit trajectories $q_{\mathrm{d}_1}^{\mathrm{r},i}$, $q_{\mathrm{d}_j}^{\mathrm{t},i}$, $j \in \{1, \ldots, 5\}$, with time duration $t_f^i = 11$ seconds, as depicted in Fig. 10, where the extra superscript $i \in \{0, \ldots, 3\}$ stands for the path. For safe tracking, we choose the exponentially decaying functions $\rho_j^{\mathrm{t},i}(t) = 0.05\exp(-0.01(t - t_s)) + 0.1 \in [0.1, 0.15]$ (rad), for all $j \in \{1, \ldots, 5\}$, and $\rho_1^{\mathrm{r},i}(t) = 0.005\exp(-0.01(t - t_{\mathsf{p}_i})) + 0.005 \in [0.005, 0.01]$, (implying $\bar{\rho} = 0.01[1, 15, 15, 15, 15, 15]$), as well as $\rho_{2_j}^i(t) = 2\max\{\max_{j\in\{1,\ldots,6\}}\{|e_{2_j}(t_{\mathsf{p}_i})|\}, 0.25\}$, where $\{t_{\mathsf{p}_0}, t_{\mathsf{p}_1}, t_{\mathsf{p}_2}, t_{\mathsf{p}_3}\} := \{0, 11, 22, 33, 44\}$ are the starting times of the four paths. We further chose $k_1^{\mathrm{r}} = 1$, $K^{\mathrm{t}} = \mathrm{diag}\{[k_j^{\mathrm{t}}]_{j\in\{1,\ldots,5\}}\} = \mathrm{diag}(1, 1, 5, 5, 5)$ and $K_2 = 0.1I_6$ and the control-input saturation bounds for the 6 joints of the UR5 robot are $[150, 150, 150, 12, 12, 12]$ Newton·meters.

The results of the experiment are depicted in Figs. 11-13. In particular, Fig. 11 depicts the evolution of the errors $\eta_1^{\mathrm{r}}(t)$ $e_j^{\mathrm{t}}(t)$ (top and bottom in cos(rad) and rad, respectively), which always satisfy the funnels defined by the respective funnel functions $\rho_1^{\mathrm{r}}(t)$, $\rho_j^{\mathrm{t}}(t)$, $j \in \{1, \ldots, 5\}$. Similarly, Fig. 12 depicts the evolution of the errors $e_{2_j}(t)$ (in rad/seconds),

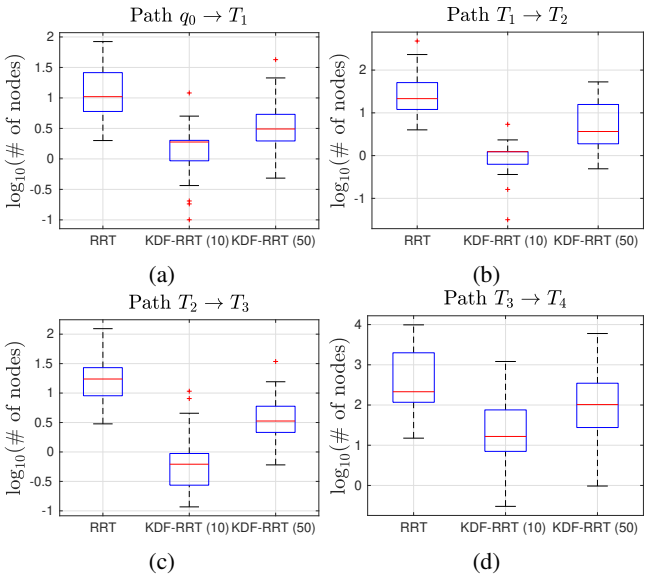


Fig. 8: Box plots showing the number of nodes created in the three algorithms in logarithmic scale for the four paths; '+' indicate the outliers.

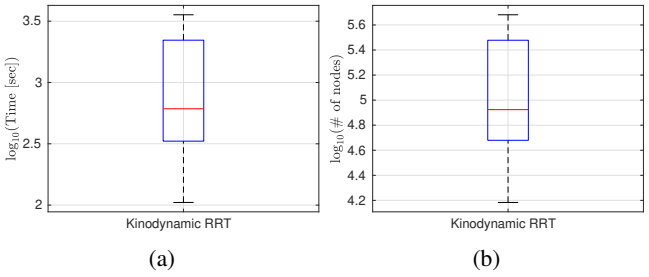


Fig. 9: Box plots showing the execution time (a) and number of nodes (b) created for the kinodynamic RRT in logarithmic scale (for the first two joints).

evolving inside the funnel defined by $\rho_{2_j}(t)$, $j \in \{1, \ldots, 6\}$. Finally, Fig. 13 illustrates the minimum distance (in meters) of the UR5 from the obstacles in the environment (top), which is always positive and verifies thus the safety of the framework, and the evolution of the control inputs $u(t) = [u_1, \ldots, u_6]^\top$ (in Newton $\cdot$ meters) for the six joint actuators (bottom). It is evidently concluded that the control-input signals satisfy the aforementioned saturation bounds $[150, 150, 150, 12, 12, 12]$.

B. Comparative Simulations

In order to further evaluate the proposed algorithm, we perform additional numerical experiments of the previous section's scenarios comparing the funnel controller with a control-barrier-function method and a model-predictive controller. We use a more controlled simulation by explicitly simulating the system dynamics in the MATLAB environment, which allows us to inject extra uncertainty and disturbance terms.

Unmanned Autonomous Vehicle

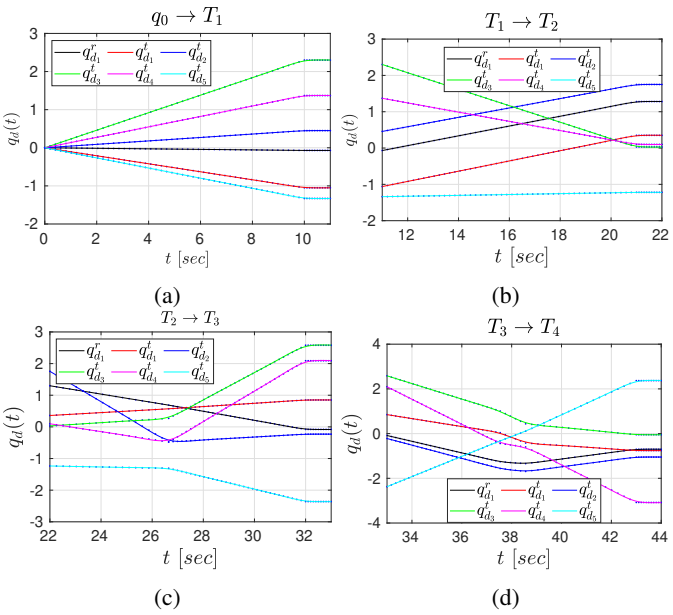


Fig. 10: The output paths and the respective time-varying trajectories for the four paths.

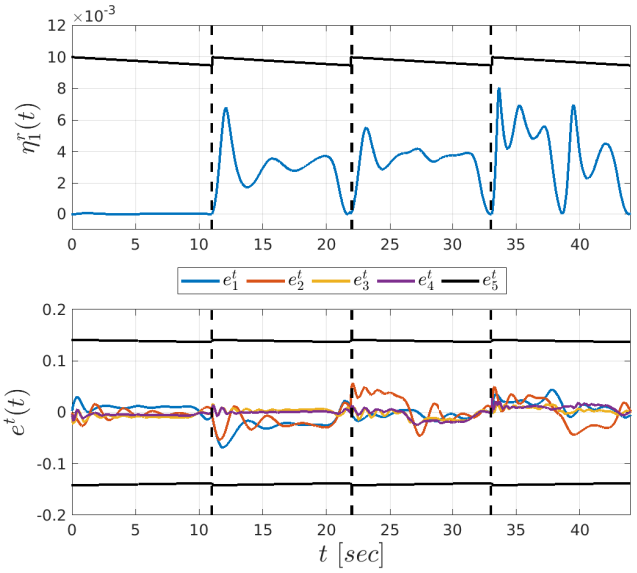


Fig. 11: Top: the evolution of the error $\eta_1^{\mathfrak{r}}(t)$ (in cos(rad)), along with the respective funnel $\rho_1^{\mathfrak{r}}(t)$, shown in black, for the four paths. Bottom: the evolution of the errors $e_j^{\mathfrak{t}}(t)$ (in rad), along with the respective funnel $\rho_j^{\mathfrak{t}}(t)$, shown in black, for the four paths.

We first consider the UAV scenario described in the previous section. We use the simulated dynamics:

$$\dot{q}_1 = q_2$$
$$\dot{q}_2 = \frac{1}{m}(d_1(q_2) + d_2(t)) + \frac{1}{m}u$$

where $m = 1$ is the UAV's mass, $d_1(q_2) = -0.5q_2 - 0.25\text{diag}\{q_2\}|q_2|$, $d_2(t) = [\sin(10t + \frac{\pi}{6}), 0.5\cos(5t - \frac{\pi}{4}), 0.75\sin(10t + \frac{\pi}{3})]^\top$ are unknown vectors of state and time,

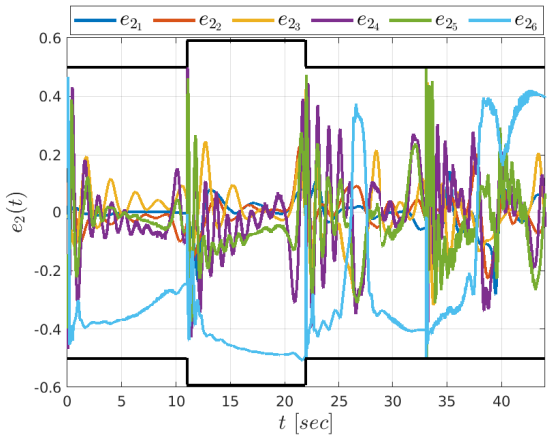


Fig. 12: The evolution of the velocity errors $e_{2_j}(t)$ (in rad/seconds), along with the respective funnels $\rho_{2_j}(t)$ (in black), $j \in \{1, \dots, 6\}$, for the four paths.

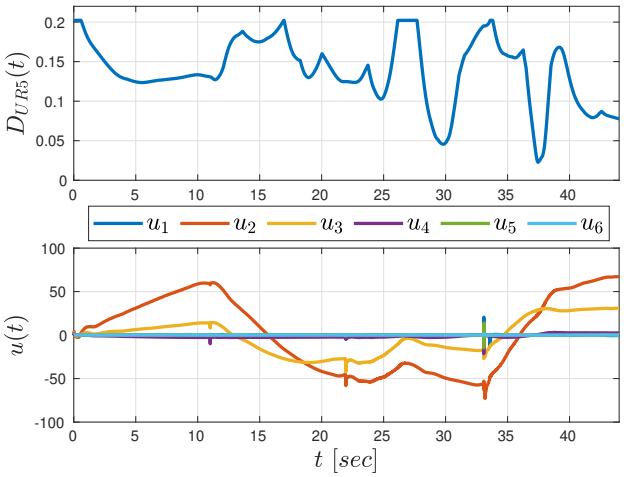


Fig. 13: Top: the distance $D_{UAV}(t)$ (in meters) of the robot from the obstacles for the four paths. Bottom: the evolution of the control inputs $u(t) = [u_1(t), \dots, u_6(t)]^\top$ (in Newton $\cdot$ meters) for the four paths.

representing uncertainties (e.g., aerodynamic terms) and time-varying disturbances, respectively; diag$\{q_2\}$ denotes the diagonal matrix with the elements of q_2 along its main diagonal, and $|q_2|$ denotes the vector of absolute values of q_2. In order to simulate a realistic scenario, we use a periodic-control setting, where the control input is applied at a frequency of 200Hz, i.e., every 0.005 seconds. We further consider control-input bounds of 15 Newton·meters.

We apply the proposed KDF algorithm under the same settings with the previous section (initial and goal positions, funnel functions, and control gains). The results are depicted in Fig. 14, which depicts the evolution of the errors $e_j^{\mathrm{t}}(t)$, along with the funnel functions $\rho_j^{\mathrm{t}}(t)$, $j \in \{1, 2, 3\}$, as well as the control inputs $u(t)$. One can verify the successful containment of the errors in the funnel and the compliance with the input saturation constraints. The chattering in the depicted signals is attributed to the rapid oscillatory disturbances $d_2(t)$, whose magnitude reaches the UAV's mass. Note, however, that such disturbances do not affect the algorithm's performance.

Next, we compare the proposed KDF algorithm with the high-level control-barrier-function (CBF) methodology proposed in [58] and a model-predictive controller (MPC) [59]. CBF methodologies guarantee invariance in a set $\mathcal{C} = \{q \in \mathbb{R}^n : \psi(q) > 0\}$, for some sufficiently smooth function ψ. In order to apply such a methodology to the considered environment, shown in Fig. 2, we set $\psi_{j_1}(e_j^{\mathrm{t}}) = \bar{\rho}_M - e_j^{\mathrm{t}}$, $\psi_{j_2}(e_j^{\mathrm{t}}) = \bar{\rho}_M + e_j^{\mathrm{t}}$ $j \in \{1, 2, 3\}$, aiming to retain the system in the safe set $\{e^{\mathrm{t}} \in \mathbb{R}^3 : \bar{\rho}_M > |e_j^{\mathrm{t}}|, j \in \{1, 2, 3\}\}$, where $\bar{\rho}_M = 0.2$, defining the extended free space of the KDF-RRT. Since the UAV model has order 2, we apply the methodology proposed in [58]. In particular, we define $h_{j_k}(q, t) = \dot{\psi}_{j_k}(e_j^{\mathrm{t}}) + \psi_{j_k}(e_j^{\mathrm{t}})^2$, $j \in \{1, 2, 3\}$, $k \in \{1, 2\}$, and choose the control input as the solution of the optimization problem

$$\min_{u \in U} \|u\|^2$$
$$\text{s.t. } \frac{\partial h_{j_k}}{\partial t} + \frac{\partial h_{j_k}}{\partial q_1}^\top q_2 + \frac{\partial h_{j_k}}{\partial q_2}^\top \frac{1}{\hat{m}} u + h_{j_k}^2 > 0,$$
$$\forall j \in \{1, 2, 3\}, k \in \{1, 2\}$$

with $U = (-15, 15)^3$. The controller is agnostic to the uncertainties and disturbances $d_1(q)$, $d_2(t)$, and it uses an estimated mass value of $\hat{m} = 1.5m = 1.5$. Note that the proposed funnel controller does not use any information on the mass m or the functions d_1, d_2. The results of the CBF methodology are given in the top part of Fig. 15, which depicts the evolution of the errors $e_j^{\mathrm{t}}(t)$, $j \in \{1, 2, 3\}$, along with the funnel safety constant $\bar{\rho}$ for 10 seconds. One concludes from the results that the errors do not comply with the constraint $|e_j^{\mathrm{t}}(t)| < \bar{\rho}_M$, showing the large dependence of the CBF method to the system dynamics.

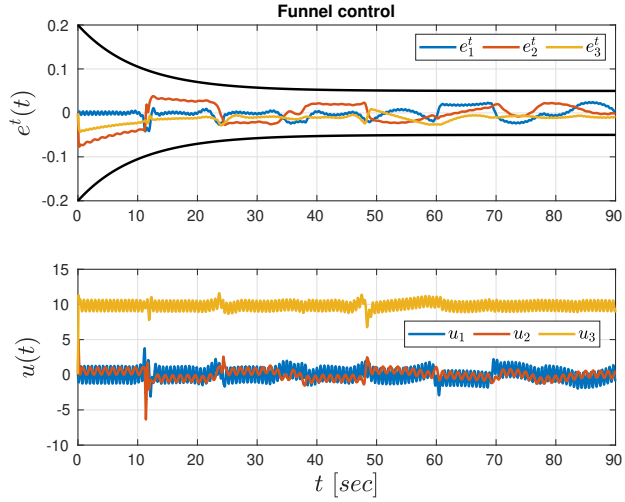


Fig. 14: Results of the funnel-control algorithm for the UAV scenario of the comparative simulation studies. Top: the evolution of the errors $e_1^{\mathrm{t}}(t)$, $e_2^{\mathrm{t}}(t)$, $e_3^{\mathrm{t}}(t)$ (in meters), along with the funnels $\rho_1^{\mathrm{t}}(t) = \rho_2^{\mathrm{t}}(t) = \rho_3^{\mathrm{t}}(t)$, shown in black, for $t \in [0, 90]$ seconds. Bottom: the evolution of the control inputs $u_1(t), u_2(t), u_3(t)$ (in Newton) for $t \in [0, 90]$ seconds.

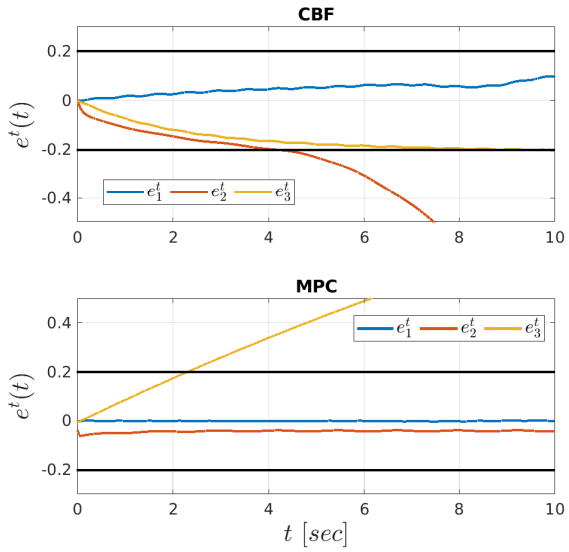


Fig. 15: The evolution of the errors $e_1^{\mathfrak{t}}(t)$, $e_2^{\mathfrak{t}}(t)$, $e_3^{\mathfrak{t}}(t)$ (in meters), along with the funnel constant $\bar{\rho}_M = 0.2$, shown in black, for the CBF methodology (top) and the MPC algorithm (bottom), for $t \in [0, 10]$ seconds.

Finally, we apply an MPC algorithm [59] to the aforementioned UAV scenario. More specifically, the controller is computed by solving online the receding-horizon optimization problem

$$\min \quad \int_t^{t+T_p} (\bar{e}(\tau)^\top Q \bar{e}(\tau) + \bar{u}(\tau)^\top R \bar{u}(\tau)) \mathrm{d}\tau + \bar{e}(t+T_p)^\top P \bar{e}(t+T_p), \quad (18a)$$

$$\text{s.t.} \quad \ddot{\bar{q}}_1 = \frac{1}{\hat{m}} \bar{u}, \qquad \bar{q}_1(t) = q_1(t), \bar{q}_2(t) = q_2(t) \quad (18b)$$

$$|\bar{e}_j^{\mathfrak{t}}(\tau)| < \bar{\rho}_M, \quad \tau \in [t, t+T_p], j \in \{1,2,3\} \quad (18c)$$

$$\bar{u}(\tau) \in U, \qquad \tau \in [t, t+T_p] \quad (18d)$$

$$\bar{e}(t+T_p) \in \Omega \quad (18e)$$

where T_p is the prediction horizon, $e = [(e^{\mathfrak{t}})^\top, (\dot{e}^{\mathfrak{t}})^\top]^\top$, $Q \in \mathbb{R}^{6\times 6}$, $R \in \mathbb{R}^{6\times 6}$, and $P \in \mathbb{R}^{6\times 6}$ are positive-definite matrices, Ω is a terminal region, and the barred variables denote the predicted signals driven by the optimization solution $\bar{u}(t)$. A common choice for MPC control is to encode the obstacle-avoidance specifications in the optimization constraints of (18) and aim to minimize the error with respect to the goal configuration Q_g. However, the labyrinth-like environment of the UAV scenario would require a significantly large prediction horizon to avoid local-minima configurations; such a horizon would prevent the MPC to be applied in real time. Therefore, (18) aims to minimize the error e with respect to the trajectory computed by the KDF-RRT, while enforcing the funnel specification $|e_j^{\mathfrak{t}}(\tau)| < \bar{\rho}_M$, $j \in \{1,2,3\}$, through the optimization constraints. Similarly to the CBF controller, MPC is agnostic to the model uncertainties and disturbances, but uses an estimated mass value $\hat{m} = 1.5m$. We further choose $T_p = 20$, $Q = P = 100 I_6$, $R = I_3$, $\Omega = \{\mathsf{x} \in \mathbb{R}^3 : \|\mathsf{x}\| \leq 0.1\}$, and a sampling frequency of 200Hz. The results are depicted in the bottom part of Fig. 15 for 10 seconds. One concludes that the closed-loop system is unstable since the vertical component grows unbounded; such a behiavor is attributed to the deviation between the actual UAV dynamics and the ones used in (18).

UR5 Robotic Manipulator

Next, we perform comparative simulations for the path $T_2 \to T_3$ of the UR5 scenario described in the previous section. We use the simulated dynamics:

$$\dot{q}_1 = q_2$$
$$\dot{q}_2 = B(q_1)^{-1}(u - C(q_1, \dot{q}_1)\dot{q}_1 - g(q_1) + d_1(q_2) + d_2(t)),$$

where $B \in \mathbb{R}^{6\times 6}$ is the positive definite inertia matrix, $C \in \mathbb{R}^{6\times 6}$ is the Coriolis matrix, $g \in \mathbb{R}^6$ is the gravity vector, and $d_1(q_2) = -q_2 - \mathrm{diag}\{q_2\}|q_2|$, $d_2(t) = [\sin(10t + \frac{\pi}{6}), 0.5\cos(5t - \frac{\pi}{4}), 0.75\sin(10t + \frac{\pi}{3}), \sin(3t - \frac{\pi}{6}), 0.5\cos(2t + \frac{\pi}{4}), 0.5\cos(2t)]^\top$ are vectors representing model uncertainties (e.g., friction) and time-varying disturbances, respectively.

We apply the proposed KDF algorithm with the initial and goal configurations q_{T_2} and q_{T_3}, respectively, using the same funnel functions as in the previous section, and control gains $k_1^{\mathfrak{r}} = 1$, $K^{\mathfrak{t}} = \mathrm{diag}\{[k_j^{\mathfrak{t}}]_{j\in\{1,\ldots,5\}}\} = I_5$, $K_2 = 10 I_6$. The results are depicted in Fig. 16, which shows the evolution of the errors $\eta_1^{\mathfrak{r}}(t)$, $e_j^{\mathfrak{t}}(t)$, along with the respective funnel functions $\rho_1^{\mathfrak{r}}(t)$, $\rho_j^{\mathfrak{t}}(t)$, $j \in \{1, \ldots, 5\}$, as well as the resulting control inputs $u(t)$. One concludes the containment of the errors in the prescribed funnels and the compliance of the control inputs with the UR5 saturation bounds $[150, 150, 150, 12, 12, 12]$.

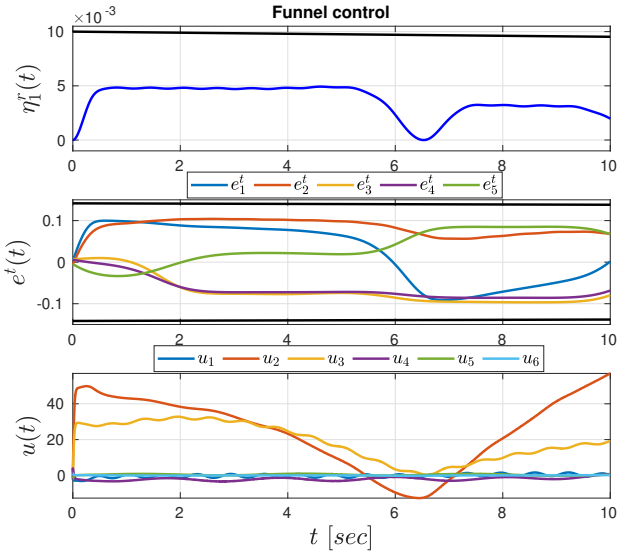


Fig. 16: Results of the funnel-control algorithm for the UR5 navigation $T_2 \to T_3$ of the comparative simulation studies. Top: the evolution of the error $\eta_1^{\mathfrak{r}}(t)$ (in cos(rad)), along with the funnel $\rho_1^{\mathfrak{r}}(t)$, shown in black. Middle: the evolution of the errors $e_j^{\mathfrak{t}}(t)$ (in rad), along with the funnel $\rho_j^{\mathfrak{t}}(t)$, shown in black. Bottom: the evolution of the control inputs $u(t) = [u_1(t), \ldots, u_6(t)]^\top$ (in Newton $\cdot$ meters).

Next, we apply the CBF methodology from [58], as in the UAV scenario. We set the barrier functions $h^{\mathfrak{r}}(q,t) = \dot{\psi}^{\mathfrak{r}}(\eta_1^{\mathfrak{r}}) +$

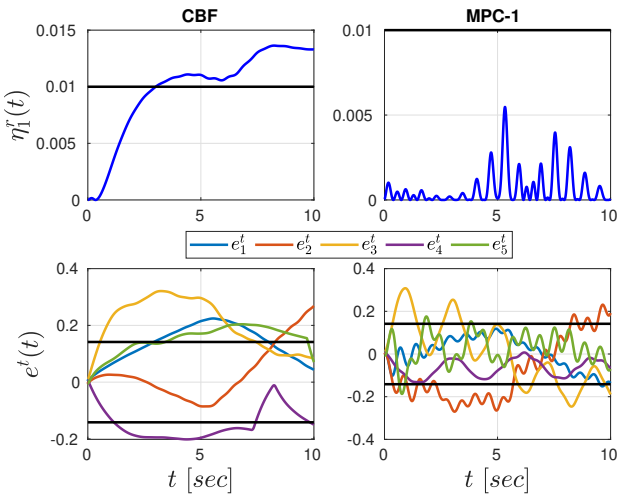


Fig. 17: The evolution of the errors $\eta_1^{\mathfrak{r}}(t)$, in cos(rad) (top) and $e_j^{\mathfrak{t}}(t)$, $j \in \{1,\dots,5\}$, in rad (bottom), along with the funnel constant $\bar{\rho}_r = 0.01$, $\bar{\rho}_{tr} = 0.15$, shown in black, for the CBF methodology (left) and the first MPC algorithm (right).

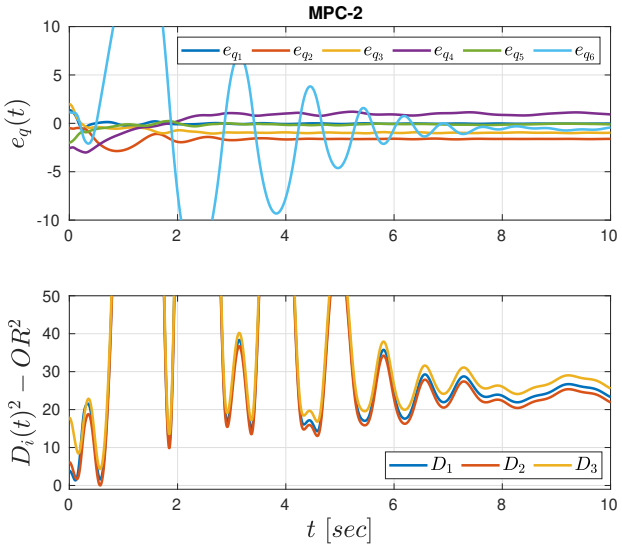


Fig. 18: Results of the second MPC algorithm of the UR5 comparative simulation studies. Top: the evolution of the error $e_q(t) = q_1(t) - q_{T_3}$ (in rad). Bottom: the evolution of the metrics $D_i(t)^2 - \text{OR}^2$ (in rad^2), for $i \in \{1,2,3\}$.

$\psi^{\mathfrak{r}}(\eta_1^{\mathfrak{r}})^2$ and $h_{j_k}^{\mathfrak{t}}(q,t) = \dot{\psi}_{j_k}^{\mathfrak{t}}(e_j^{\mathfrak{t}}) + \psi_{j_k}^{\mathfrak{t}}(e_j^{\mathfrak{t}})^2$, where $\psi^{\mathfrak{r}}(\eta_1^{\mathfrak{r}}) = \bar{\rho}_r - \eta_1^{\mathfrak{r}}$, $\psi_{j_1}^{\mathfrak{t}}(e_j^{\mathfrak{t}}) = \bar{\rho}_{tr} - e_j^{\mathfrak{t}}$, $\psi_{j_2}^{\mathfrak{t}}(e_j^{\mathfrak{t}}) = \bar{\rho}_{tr} + e_j^{\mathfrak{t}}$, $\bar{\rho}_r = 0.01$, $\bar{\rho}_{tr} = 0.15$. The control input is chosen as the solution of the optimization problem

$$
\begin{aligned}
&\min_{u\in U} \|u\|^2 \\
&\text{s.t. } \frac{\partial h^{\mathfrak{r}}}{\partial t} + \left(\frac{\partial h^{\mathfrak{r}}}{\partial q_1}\right)^\top q_2 + \left(\frac{\partial h^{\mathfrak{r}}}{\partial q_2}\right)^\top \left[\hat{B}(q_1)^{-1}(u - \hat{g}(q_1)\right. \\
&\left. - \hat{C}(q_1,q_2)q_2)\right] + (h^{\mathfrak{r}})^2 > 0 \\
&\text{s.t. } \frac{\partial h_{j_k}^{\mathfrak{t}}}{\partial t} + \left(\frac{\partial h_{j_k}^{\mathfrak{t}}}{\partial q_1}\right)^\top q_2 + \left(\frac{\partial h_{j_k}^{\mathfrak{t}}}{\partial q_2}\right)^\top \left[\hat{B}(q_1)^{-1}(u - \hat{g}(q_1)\right. \\
&\left. - \hat{C}(q_1,q_2)q_2)\right] + (h_{j_k}^{\mathfrak{t}})^2 > 0,\ \forall j \in \{1,\dots,6\}, k \in \{1,2\}
\end{aligned}
$$

Similarly to the UAV scenario, in order to account for uncertainty, the controller is agnostic to the terms $d_1(q_2)$ and $d_2(t)$. Further, the terms $\hat{B}(q_1)$, $\hat{C}(q_1,q_2)$, and $\hat{g}(q_1)$ are estimations of $B(q_1)$, $C(q_1,q_2)$, and $g(q_1)$, respectively, and are computed based on a deviation of the actual masses and moments of inertia; the deviation is sampled from the interval $[75\%, 125\%]$. The results of the CBF methodology are given in the left part of Fig. 17, which depicts the evolution of the errors $\eta_1^{\mathfrak{r}}(t)$, $e_j^{\mathfrak{t}}(t)$, along with the funnel safety constants $\bar{\rho}_r$ and $\bar{\rho}_{tr}$. Similar to the UAV scenario, one concludes the strong dependence of the methodology to the system dynamics, since the errors are not contained in the region defined by the constants $\bar{\rho}_r$ and $\bar{\rho}_{tr}$.

Finally, we apply two MPC algorithms [59] for the safe navigation from q_{T_2} to q_{T_3}. The first one is similar to the one used for the UAV scenario and consists of the receding-horizon optimization problem

$$
\begin{aligned}
\min \quad & \int_t^{t+T_p} (\bar{e}(\tau)^\top Q \bar{e}(\tau) + \bar{u}(\tau)^\top R \bar{u}(\tau))\mathrm{d}\tau \\
& + \bar{e}(t+T_p)^\top P \bar{e}(t+T_p), \\
\text{s.t.} \quad & \ddot{\bar{q}}_1 = \hat{B}(q_1)^{-1}(u - \hat{C}(q_1,q_2)q_2 - \hat{g}(q_1)), \\
& \bar{q}_1(t) = q_1(t), \bar{q}_2(t) = q_2(t) \\
& \bar{\eta}_1^{\mathfrak{r}}(\tau) < \bar{\rho}_r, \quad \tau \in [t, t+T_p] \\
& |\bar{e}_j^{\mathfrak{t}}(\tau)| < \bar{\rho}_{tr}, \quad \tau \in [t, t+T_p], j \in \{1,\dots,5\} \\
& \bar{u}(\tau) \in U, \quad \tau \in [t, t+T_p] \\
& \bar{e}(t+T_p) \in \Omega
\end{aligned}
$$

where we impose the funnel specification with respect to the trajectory obtained by KDF-RRT through the optimization constraints. We choose $T_p = 20$, $Q = P = 100 I_6$, $R = I_3$, $\Omega = \{\mathsf{x} \in \mathbb{R}^3 : \|\mathsf{x}\| \leq 0.1\}$, and a sampling frequency of 200Hz. The results are given in the right part of Fig. 17, which depicts the evolution of the errors $\eta_1^{\mathfrak{r}}(t)$, $e_j^{\mathfrak{t}}(t)$, along with the funnel safety constants $\bar{\rho}_r$ and $\bar{\rho}_{tr}$. It is clear that the errors are not contained in the region defined by $\bar{\rho}_r$ and $\bar{\rho}_{tr}$, verifying the strong dependence of MPC to the system dynamics.

In the second MPC methodology, we aim to minimize the deviation of the UR5 configuration with respect to the goal configuration T_3 and we explicitly encode the collision avoidance in the optimization constraints, i.e.,

$$
\begin{aligned}
\min \quad & \int_t^{t+T_p} (\bar{e}_g(\tau)^\top Q \bar{e}_g(\tau) + \bar{u}(\tau)^\top R \bar{u}(\tau))\mathrm{d}\tau \\
& + \bar{e}_g(t+T_p)^\top P \bar{e}_g(t+T_p), \\
\text{s.t.} \quad & \ddot{\bar{q}}_1 = \hat{B}(q_1)^{-1}(u - \hat{C}(q_1,q_2)q_2 - \hat{g}(q_1)), \\
& \bar{q}_1(t) = q_1(t), \bar{q}_2(t) = q_2(t) \\
& q_1(\tau) \in \mathcal{S}, \quad \tau \in [t, t+T_p] \\
& \bar{u}(\tau) \in U, \quad \tau \in [t, t+T_p] \\
& \bar{e}_g(t+T_p) \in \Omega
\end{aligned}
$$

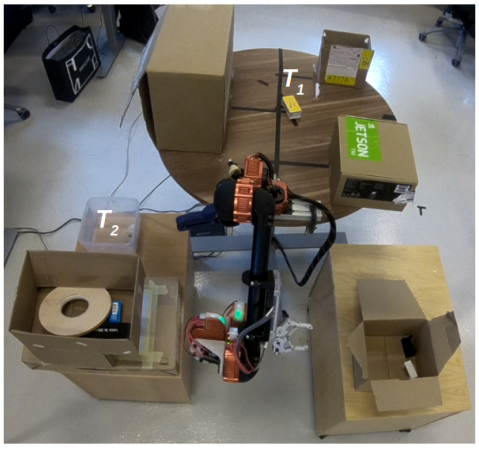
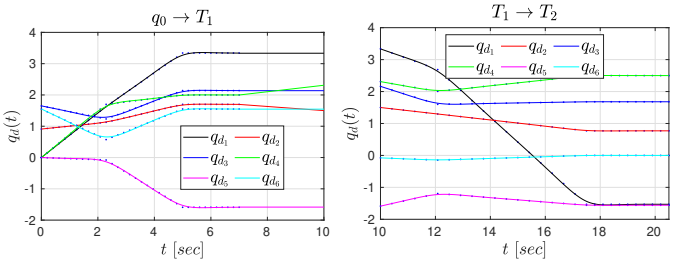


Fig. 19: The initial configuration of the HEBI robot in an obstacle-cluttered environment.

where $e_g = [e_q^\top, \dot{e}_q^\top]^\top = [(q_1 - q_{T_3})^\top, q_2^\top]^\top$ and the set $\mathcal{S}$ captures the safe part of the workspace, modeled as the exterior set of three spherical regions, i.e., $\mathcal{S} = \bigcup_{k\in\{1,2,3\}}\{q_1 \in \mathbb{R}^6 : D_k = \|q_1 - \text{OC}_k\|^2 > \text{OR}^2\}$, with $\text{OC}_1 = [\frac{\pi}{2}, -\frac{2\pi}{9}, -\frac{\pi}{6}, 0, 0, 0]^\top$, $\text{OC}_2 = [-\frac{\pi}{4}, -\frac{\pi}{4}, 0, 0, 0, 0]^\top$, $\text{OC}_3 = [-\frac{3\pi}{4}, -\frac{\pi}{3}, -\frac{\pi}{6}, 0, 0, 0]^\top$, and $\text{OR} = 2$ (rad). Further, we choose $T_p = 20$, $Q = P = 100I_6$, $R = I_3$, $\Omega = \{\mathsf{x} \in \mathbb{R}^3 : \|\mathsf{x}\| \leq 0.1\}$. The results are given in Fig. 18, which depicts the evolution of the errors $e_q(t) = q_1(t) - q_{T_3}$ and the metrics $D_k^2 - \text{OR}^2$, $k \in \{1, 2, 3\}$. Although the distances satisfy the safety constraint $D_k(t) > \text{OR}^2$, the errors e_q exhibit a significant steady-state error, which is attributed to the dynamic uncertainties. Additionally, we stress that the imposed safe set defined by $\|q_1 - \text{OC}_k\|^2 > \text{OR}^2 = 4$ is a superset of the actual safe set shown in Fig. 6; larger values of OR, which would represent more accurately the actual safe set, led to excessive computational times for the MPC optimization that would prevent it from being applied in real time.

C. Hardware Experiments



(a) (b)

Fig. 20: The output sequence of points and the respective trajectories for the two paths of the hardware experiment.

This section is devoted to the experimental validation of the proposed framework using a 6DOF manipulator from HEBI-Robotics subject to 2nd-order dynamics (as in (17)), which consists of 6 rotational joints (see Fig. 19) operating in $[-\pi, \pi]$, resulting in $q_1 = [q_1^{\mathsf{t}}, \dots, q_6^{\mathsf{t}}]^\top$.

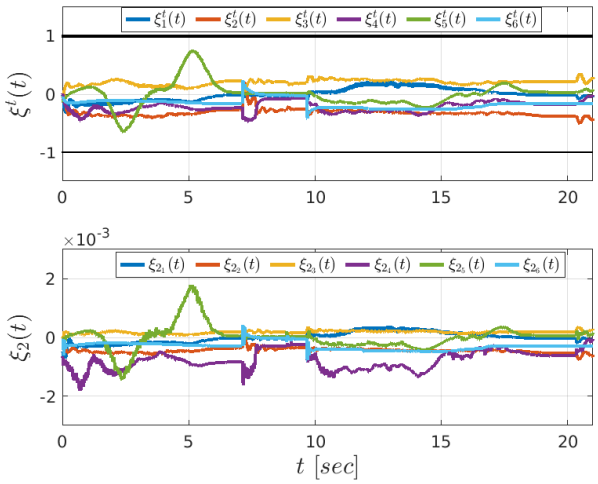


Fig. 21: The evolution of the normalized errors ξ_j^{t} (top) and $\xi_{2_j}(t)$, for $j \in \{1, \dots, 6\}$, of the hardware experiment.

We consider that the robot has to perform a pick-and-place task, where it has to pick an object from T_1 and deliver it in T_2 (see Fig. 19). We use the KDF-RRT algorithm, with $\bar{\rho} = [0.15, 0.1, 0.1, 0.2, 0.2, 0.2]$ rad, to generate two paths: from the initial configuration to a point close to T_1 (to avoid collision with the object), and from T_1 to T_2. Regarding the collision checking in $\bar{\mathcal{A}}_{\text{free}}(\bar{\rho})$, we check 10 samples around each point of the resulting path for collision. We next fit smooth trajectories for the two paths $q_{\text{d}}^{\mathsf{t},1}(t)$, $q_{\text{d}}^{\mathsf{t},2}(t)$, with duration of $t_{f_1} = 7$ and $t_{f_2} = 11$ seconds, respectively, as shown in Fig. 20. For grasping the object, we use linear interpolation to create an additional trajectory segment to T_1 with duration of 3 seconds (see Fig. 20(a) for $t \in [7, 10]$).

For the execution of the control algorithm, we choose constant funnel functions $\rho^{\mathsf{t},i} = [\rho_1^{\mathsf{t},i}, \dots, \rho_6^{\mathsf{t},i}]^\top = \bar{\rho} = [0.15, 0.1, 0.1, 0.2, 0.2, 0.2]$ rad, for the two paths $i \in \{1, 2\}$. Moreover, we choose $\rho_{2_j} = 15$ for all $j \in \{1, \dots, 6\}$, and the control gains as $K^{\mathsf{t}} = \text{diag}(1.25, 1.5, 1, 2, 1, 1)$, $K_2 = \text{diag}(250, 200, 150, 50, 20, 10)$.

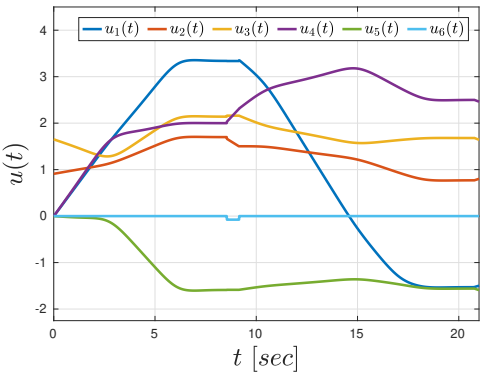
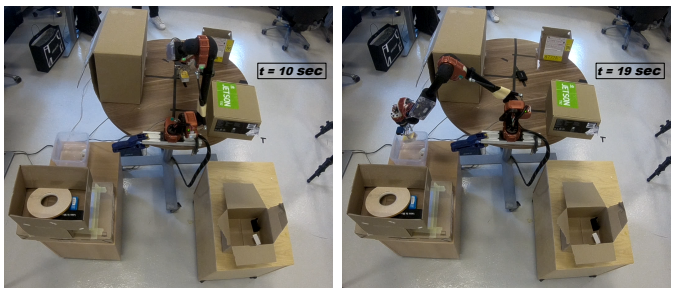


Fig. 22: The evolution of the control inputs $u(t) = [u_1(t), \dots, u_6(t)]^\top$ of the hardware experiment.

The results of the experiment are depicted in Figs 21 and 22. In particular, Fig. 20 depicts the normalized signals $\xi^{\mathsf{t}}(t) =$

$[\xi_1^{\mathrm{t}}, \ldots, \xi_6^{\mathrm{t}}]^\top$ and $\xi_2(t) = [\xi_{2_1}, \ldots, \xi_{2_6}]^\top$ (top and bottom, respectively) for $t \in [0, 21]$ seconds. It can be observed that for the entire motion, it holds that $\xi_j^{\mathrm{t}} \in (-1, 1)$, $\xi_{2_j}(t) \in (-1, 1)$, for all $j \in \{1, \ldots, 6\}$, which implies that $-\rho_j^{\mathrm{t}} < e_j^{\mathrm{t}}(t) = q_1(t) - q_{\mathrm{d}}(t) < \rho_j^{\mathrm{t}}$, $-\rho_{2_j} < e_2(t) = q_{2_j}(t) - \alpha_{1_j}(t) < \rho_j^{\mathrm{t}}$, for all $j \in \{1, \ldots, 6\}$ and $t \in [0, 21]$ seconds, with α_1 as in (8). Therefore, we conclude that the robot tracks the path output by the KDF-RRT algorithm within the prescribed funnel, avoiding thus collisions. Snapshots of the path execution are given in Fig. 23 for $t = 10$ and $t = 19$ seconds.



(a) (b)

Fig. 23: Snapshots of the hardware experiment at $t = 10$ (a), and $t = 19$ (b) seconds.

We further test the robustness of the proposed control scheme against *adversarial* disturbances. In particular, we disturb the manipulator using a rod three times during the execution of the aforementioned trajectory (see Fig. 24). In order to prevent the control scheme from having invalid values (see the domain of definition of (7) and (10)), we set $\xi_j^{\mathrm{t}} = \max\left\{\min\left\{1, \frac{e_j^{\mathrm{t}}}{\rho_j^{\mathrm{t}}}\right\}, -\frac{e_j^{\mathrm{t}}}{\rho^{\mathrm{t}}{}_j^{\mathrm{t}}}\right\}$, $\xi_{2_j} = \max\left\{\min\left\{1, \frac{e_{2_j}}{\rho_{2_j}}\right\}, -\frac{e_{2_j}}{\rho_{2_j}}\right\}$ for all $j \in \{1, \ldots, 6\}$. The evolution of the signals $\xi^{\mathrm{t}}(t)$, $\xi_2(t)$ are depicted in Fig. 25 for 21 seconds, with vertical black dashed lines depicting the instants of the disturbance, which affects mostly the first joint of the system; note from Fig. 25 that ξ_1^{t} and ξ_{1_1} are excessively increased with respect to their nominal values shown in Fig. 21, implying a large increase in the respective errors e_1^{t} and e_{2_1}. Nevertheless, one can conclude that, despite the presence of adversarial disturbances, the system manages to successfully recover and complete the derived path.

In order to further evaluate the proposed control algorithm, we compared our results with a standard well-tuned PID controller as well as the parametric adaptive control scheme (PAC) of our previous work [39]. The signals ξ^{t} for these two control schemes are depicted in Fig. 26. Note that the controllers fail to retain the normalized errors $\xi_j^{\mathrm{t}}(t)$ in the interval $(-1, 1)$. Although in the particular instance this did not lead to collisions, it jeopardizes the system motion, since it does not comply with the bounds set in the KDF-RRT.

This experimental section helps to verify the validity and effectiveness of the proposed algorithm in a practical setting. In particular, note that the considered model in eq. (1) assumes continuous measurements of the system state (e.g., the joint angles in the case of a robotic manipulator) and continuous application of the control input. However, the sensors and

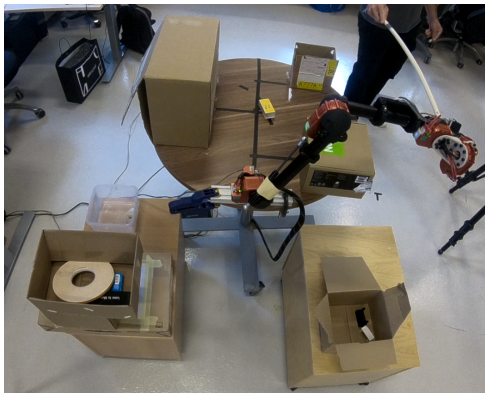


Fig. 24: Application of adversarial disturbances in the hardware experiment.

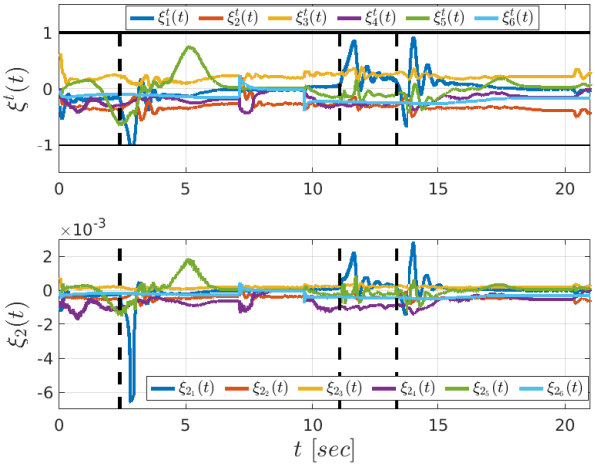


Fig. 25: The evolution of the normalized errors ξ_j^{t} (top) and $\xi_{2_j}(t)$, for $j \in \{1, \ldots, 6\}$, of the hardware experiment, in the case of adversarial disturbances. The time instants of the disturbance application are shown with vertical dashed lines.

motors in a real system, as well as the communication between them, operate at discrete sampling instants. The successful application of the proposed algorithm in such a system shows its robustness against such differences and its applicability in a real-world scenario. Furthermore, the numerical experiments show the robustness of the proposed algorithm against manually added perturbations.

V. Conclusion

We develop KDF, a new framework for solving the kinodynamic motion-planning problem for complex systems with uncertain dynamics. The framework comprises of three modules: first, a family of geometric sampling-based motion planners that produce a path in an extended free space; secondly, a smoothening and time endowment procedure that converts the path into a smooth time-varying trajectory; and finally, a funnel-based feedback control scheme that guarantees safe tracking of the trajectory. Neither of the modules uses any information on the system dynamics. Experimental results demonstrate the effectiveness of the proposed method. Future

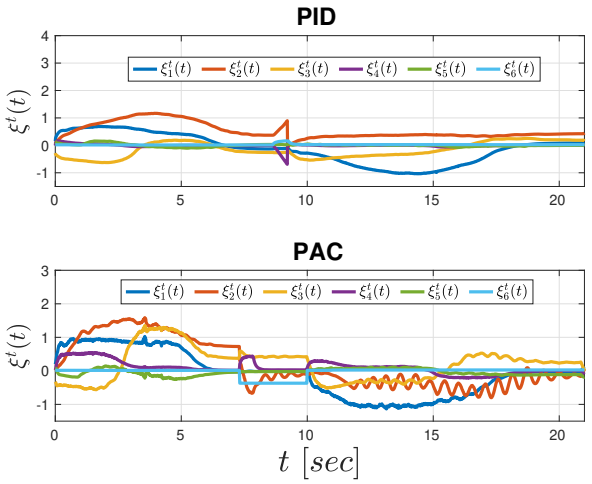


Fig. 26: The evolution of the normalized errors $\xi_j^{\mathfrak{t}}$ and $\xi_{2_j}(t)$, for $j \in \{1, \ldots, 6\}$, of the hardware experiment, when using a PID controller (top), and the adaptive control algorithm from [39] (bottom).

directions will focus on extending KDF to systems with non-holonomic and underactuated dynamics and taking into account explicit input constraints.

APPENDIX

Proof of Theorem 1. : Consider the non-empty open set

$$\begin{aligned}\Omega := \big\{&(\bar{q}, t) \in \mathbb{T} \times \mathbb{R}^{n(k-1)} \times [t_0, t_0 + t_f) : \xi_j^{\mathfrak{t}} \in (-1, 1),\\ &\xi_\ell^{\mathfrak{r}} \in [0, 1), \xi_i \in (-1, 1)^n, \forall j \in \{1, \ldots, n_{tr}\},\\ &\ell \in \{1, \ldots, n_r\}, i \in \{2, \ldots, k\}\big\},\end{aligned} \tag{19}$$

where we implicitly write the ξ variables as function of $\bar{q}$ and time t. The constraints (4) imply that $(\bar{q}(t_0), t_0) \in \Omega$. By substituting the control law (12) in the dynamics (1), we obtain a closed-loop system $\dot{\bar{q}} = f_{\text{cl}}(\bar{q}, t)$ and one can verify, based on Assumption 1, that f_{cl} is continuously differentiable in $\bar{q}$ and continuous t on Ω. Therefore, the conditions of [60, Theorem 2.1.3] are satisfied and we conclude the existence of a *maximal* solution $\bar{q}(t)$ for $t \in I_t := [t_0, t_0 + t_{\max})$, with $t_{\max} > 0$, satisfying $(\bar{q}(t), t) \in \Omega$ for all $t \in I_t$.

Hence, for $t \in I_t$, the transformed errors $\varepsilon_j^{\mathfrak{t}}$, $\varepsilon_\ell^{\mathfrak{r}}$, ε_i are well defined. We proceed inductively with the following steps.

Step 1. Consider the positive definite and radially unbounded candidate Lyapunov function $V_1 := \frac{1}{2}(\varepsilon^{\mathfrak{t}})^\top K^{\mathfrak{t}} \varepsilon^{\mathfrak{t}} + \sum_{\ell \in \mathcal{L}_r} k_\ell^{\mathfrak{r}} \varepsilon_\ell^{\mathfrak{r}}$, where $\mathcal{L}_r := \{1, \ldots, n_r\}$, and $K^{\mathfrak{t}} := \text{diag}\{[k_j^{\mathfrak{t}}]_{j \in \{1, \ldots, n_{tr}\}}\} \in \mathbb{R}^{n_{tr} \times n_{tr}}$, $k_\ell^{\mathfrak{r}}$ are gains introduced in (8). Let also the first equation of (1) be partitioned as

$$\begin{bmatrix}\dot{q}^{\mathfrak{t}} \\ \dot{q}^{\mathfrak{r}}\end{bmatrix} = \begin{bmatrix}f^{\mathfrak{t}}(q_1, t) \\ f^{\mathfrak{r}}(q_1, t)\end{bmatrix} + \begin{bmatrix}g_{11}(q_1, t) & g_{12}(q_1, t) \\ g_{13}(q_1, t) & g_{14}(q_1, t)\end{bmatrix} q_2.$$

Differentiating V_1, using $q_2 = \alpha_1 + e_2$ from (9) and substituting (8), we obtain

$$\begin{aligned}\dot{V}_1 = &- \sigma^\top K \widetilde{R} \widetilde{\rho}^{-1} \widetilde{S} g_1 \widetilde{S} \widetilde{\rho}^{-1} \widetilde{R} K \sigma + \\ &\sigma^\top K \widetilde{R} \widetilde{\rho}^{-1} \left[\widetilde{S}\big(f_1 + g_1 e_2 - \dot{q}_{\text{d}}\big) - \dot{\widetilde{\rho}} \xi\right] =: T_n + T_b,\end{aligned}$$

where we further define $\xi := [(\xi^{\mathfrak{t}})^\top, (\xi^{\mathfrak{r}})^\top]^\top$, $\sigma := [(\varepsilon^{\mathfrak{t}})^\top, (r^{\mathfrak{r}})^\top]^\top$, $K := \text{blkdiag}\{K^{\mathfrak{t}}, K^{\mathfrak{r}}\}$, $\widetilde{R} := \text{blkdiag}\{\widetilde{r}^{\mathfrak{t}}, I\}$, $\widetilde{S} := \text{blkdiag}\{I, \widetilde{s}^{\mathfrak{r}}\}$, $\widetilde{\rho} := \text{blkdiag}\{\widetilde{\rho}^{\mathfrak{t}}, \widetilde{\rho}^{\mathfrak{r}}\}$, $\xi^{\mathfrak{t}} := [\xi_1^{\mathfrak{t}}, \ldots, \xi_{n_{tr}}^{\mathfrak{t}}]^\top$, $\xi^{\mathfrak{t}} := [\xi_1^{\mathfrak{r}}, \ldots, \xi_{n_r}^{\mathfrak{r}}]^\top$, $\widetilde{r}^{\mathfrak{t}} := \text{diag}\{[r_j^{\mathfrak{t}}]_{j \in \{1, \ldots, n_{tr}\}}\}$, $r^{\mathfrak{r}} := [r_1^{\mathfrak{r}}, \ldots, r_{n_r}^{\mathfrak{r}}]^\top$, $\widetilde{\rho}^{\mathfrak{t}} := \text{diag}\{[\rho_j^{\mathfrak{t}}]_{j \in \{1, \ldots, n_{tr}\}}\}$, $\widetilde{\rho}^{\mathfrak{r}} := \text{diag}\{[\rho_\ell^{\mathfrak{r}}]_{\ell \in \{1, \ldots, n_r\}}\}$, $\varepsilon^{\mathfrak{t}} := [\varepsilon_1^{\mathfrak{t}}, \ldots, \varepsilon_{n_{tr}}^{\mathfrak{t}}]^\top$, $\widetilde{s}^{\mathfrak{r}} := \text{diag}\{[\sin(e_\ell^{\mathfrak{r}})]_{\ell \in \{1, \ldots, n_r\}}\}$, and $K_{\mathfrak{r}} := \text{diag}\{[k_\ell^{\mathfrak{r}}]_{\ell \in \{1, \ldots, n_r\}}\} \in \mathbb{R}^{n_r \times n_r}$. Since T_n is a quadratic form, it holds that $T_n = -\frac{1}{2}\sigma^\top K \widetilde{R} \widetilde{\rho}^{-1} \widetilde{S}(g_1 + g_1^\top) \widetilde{S} \widetilde{\rho}^{-1} \widetilde{R} K \sigma$, and in view of Assumption 2, $T_n \leq -\underline{g} \|K \widetilde{R} \widetilde{\rho}^{-1} \widetilde{S} \sigma\|^2$, where $\underline{g} := \frac{1}{2}\lambda_{\min}(g_1 + g_1^\top) > 0$. Moreover, we obtain from (6) $1 - \cos(e_\ell^{\mathfrak{r}}) = \rho_\ell^{\mathfrak{r}} \xi_\ell^{\mathfrak{r}}$ implying $\sin^2(e_\ell^{\mathfrak{r}}) = \rho_\ell^{\mathfrak{r}} \xi_\ell^{\mathfrak{r}}(1 + \cos(e_\ell^{\mathfrak{r}}))$, for all $\ell \in \mathcal{L}_r$. By further defining $\underline{k} := \underline{g} \lambda_{\min}(K \widetilde{\rho})$, we obtain

$$T_n \leq -\underline{k} \|\widetilde{r}^{\mathfrak{t}} \varepsilon^{\mathfrak{t}}\|^2 - \underline{k} \sum_{\ell \in \mathcal{L}_r} (r_\ell^{\mathfrak{r}})^2 \xi_\ell^{\mathfrak{r}}(1 + \cos(e_\ell^{\mathfrak{r}})).$$

Note that, for $t \in I_t$, it holds that $\xi_\ell^{\mathfrak{t}} \in (-1, 1)$ and hence $\eta_\ell^{\mathfrak{r}}(t) = 1 - \cos(e_\ell^{\mathfrak{r}}(t)) < \rho_\ell^{\mathfrak{r}}(t) \leq \bar{\rho}_\ell^{\mathfrak{r}} < 2$, for all $\ell \in \mathcal{L}_r$ (see (4)). Therefore, it holds that $1 + \cos(e_\ell^{\mathfrak{r}}) \geq 2 - \bar{\rho}_\ell^{\mathfrak{r}} =: \underline{e}_\ell^{\mathfrak{r}} > 0$, $\ell \in \mathcal{L}_r$. By further defining $\underline{e}^{\mathfrak{r}} := \min_{\ell \in \mathcal{L}_r}\{\underline{e}_\ell^{\mathfrak{r}}\}$, we obtain

$$T_n \leq -\underline{k} \|\widetilde{r}^{\mathfrak{t}} \varepsilon^{\mathfrak{t}}\|^2 - \underline{k} \underline{e}^{\mathfrak{r}} \sum_{\ell \in \mathcal{L}_r} (r_\ell^{\mathfrak{r}})^2 \xi_\ell^{\mathfrak{r}} \leq -\underline{m} \|\kappa\|^2,$$

where $\kappa := [(\widetilde{r}^{\mathfrak{t}} \varepsilon^{\mathfrak{t}})^\top, r_1^{\mathfrak{r}} \sqrt{\xi_1^{\mathfrak{r}}}, \ldots, r_{n_r}^{\mathfrak{r}} \sqrt{\xi_{n_r}^{\mathfrak{r}}}]^\top$, and $\underline{m} := \min\{\underline{k}, \underline{k} \underline{e}^{\mathfrak{r}}\}$. Moreover, the fact that $q_{\text{d}}(t)$ is bounded and $(\bar{q}(t), t) \in \Omega$ for $t \in I_t$ implies that $q_1^{\mathfrak{t}}(t)$ is bounded as $\|q^{\mathfrak{t}}(t)\| \leq \sup_{t \geq t_0} \|q_{\text{d}}^{\mathfrak{t}}(t)\| + \sqrt{n_{tr}} \max_{j \in \{1, \ldots, n_{tr}\}}\{\bar{\rho}_j^{\mathfrak{t}}\}$ and $\|e_2(t)\| \leq \sqrt{n} \max_{m \in \{1, \ldots, n\}}\{\bar{\rho}_{2_m}\}$, for $t \in I_t$. Note that the aforementioned bounds do not depend on $t_{\max}$. Hence, we conclude by Assumption 1 that $f_1(q_1(t), t)$, $g_1(q_1(t), t)$ are bounded in I_t, by bounds independent of $t_{\max}$. Next, owing to the boundedness of $q_1^{\mathfrak{r}}(t)$ and $\dot{q}_{\text{d}}$, $\widetilde{\rho}^{-1}$ (by definition and assumption, respectively), as well as by using $\xi_\ell^{\mathfrak{r}} < \sqrt{\xi_\ell^{\mathfrak{r}}} < 1$, for all $\ell \in \mathcal{L}_r$, we conclude that there exists a positive finite constant $\bar{B}_1$, independent of I_t, satisfying $T_b \leq \bar{B}_1 \|\kappa\|$, for all $t \in I_t$. Therefore, $\dot{V}_1$ becomes $\dot{V}_1 \leq -\underline{m}\|\kappa\|^2 + \bar{B}_1 \|\kappa\|$ for all $t \in I_t$ and is negative when $\|\kappa\| > \frac{\bar{B}_1}{\underline{m}}$, i.e.,

$$\sqrt{\sum_{j \in \mathcal{L}_t} (r_j^{\mathfrak{t}} \varepsilon_j^{\mathfrak{t}})^2 + \sum_{\ell \in \mathcal{L}_r} (r_\ell^{\mathfrak{r}})^2 \xi_\ell^{\mathfrak{r}}} > \frac{\bar{B}_1}{\underline{m}}, \tag{20}$$

with $\mathcal{L}_t := \{1, \ldots, n_{tr}\}$. From the definition of $r_j^{\mathfrak{t}}$ in (7), it holds that $r_j^{\mathfrak{t}}(t) \geq 2$, for all $j \in \mathcal{L}_t$ and $\forall t \in I_t$. Moreover, one can conclude by inspection that the function $\mathsf{f}(\mathsf{x}) = \frac{1}{(1-\mathsf{x})^2}\mathsf{x} - \ln\left(\frac{1}{1-\mathsf{x}}\right)$ is positive for positive x, Therefore, since by definition $\xi_\ell^{\mathfrak{r}} \geq 0$ it holds that $(r_\ell^{\mathfrak{r}})^2 \xi_\ell^{\mathfrak{r}} \geq \varepsilon_\ell^{\mathfrak{r}}$, for all $\ell \in \mathcal{L}_r$. Therefore, it holds that $\sqrt{\sum_{j \in \mathcal{L}_t} (r_j^{\mathfrak{t}} \varepsilon_j^{\mathfrak{t}})^2 + \sum_{\ell \in \mathcal{L}_r} (r_\ell^{\mathfrak{r}})^2 \xi_\ell^{\mathfrak{r}}} \geq \sqrt{\sum_{j \in \mathcal{L}_t} (\varepsilon_j^{\mathfrak{t}})^2 + \sum_{\ell \in \mathcal{L}_r} \varepsilon_\ell^{\mathfrak{r}}}$ and a sufficient condition for $\dot{V}_1$ to be negative is $\sqrt{\sum_{j \in \mathcal{L}_t} (\varepsilon_j^{\mathfrak{t}})^2 + \sum_{\ell \in \mathcal{L}_r} \varepsilon_\ell^{\mathfrak{r}}} > \frac{\bar{B}_1}{\underline{m}_1}$, from which we conclude, by applying Theorem 4.18 of [43], that

there exists a positive constant $\bar{\varepsilon}$ satisfying $|\varepsilon_j^{\mathfrak{t}}(t)| \leq \bar{\varepsilon}$ and $\varepsilon_\ell^{\mathfrak{r}}(t) \leq \bar{\varepsilon}$, which implies via (7) that

$$|\xi_j^{\mathfrak{t}}(t)| \leq \bar{\xi}^{\mathfrak{t}} \coloneqq \frac{\exp(\bar{\varepsilon})-1}{\exp(\bar{\varepsilon}+1} < 1 \tag{21}$$

$$\xi_\ell^{\mathfrak{r}}(t) \leq \bar{\xi}^{\mathfrak{r}} \coloneqq \frac{\exp(\bar{\varepsilon})-1}{\exp(\bar{\varepsilon})} < 1, \tag{22}$$

for all $t \in I_t$, $j \in \mathcal{L}_t$, $\ell \in \mathcal{L}_r$. Hence, $\alpha_1(t)$, as designed in (8), is bounded, for all $t \in I_t$, from which we also conclude the boundedness of $q_2 = e_2 + \alpha_1$, since $\|e_2(t)\| = \|\rho_i(t)\xi_2(t)\| \leq \sqrt{n}\max_{m\in\{1,\ldots,n\}}\{\bar{\rho}_{2_m}\}$ for all $t \in I_t$. Moreover, by invoking (21), we conclude the boundedness of $\dot{\alpha}_1$, for all $t \in I_t$.

Step $i \in \{2,\ldots,k\}$: We apply recursively the aforementioned line proof for the remaining step. By considering the function $V_i = \frac{1}{2}\varepsilon_i^\top K_i \varepsilon$, we obtain

$$\begin{aligned}\dot{V}_i \leq &-\varepsilon_i^\top r_i\rho_i^{-1}K_i g_i K_i \rho_i^{-1} r_i \varepsilon_i \\ &+ \|r_i\rho_i^{-1}K_i\varepsilon_i\|\|f_i + g_i e_{i+1} - \dot{\alpha}_{i-1} - \dot{\rho}_i\xi_i\|,\end{aligned}$$

for $i \in \{2,\ldots,k-1\}$, and

$$\begin{aligned}\dot{V}_k \leq &-\varepsilon_k^\top r_k\rho_k^{-1}K_k g_k K_k \rho_k^{-1} r_k \varepsilon_k \\ &+ \|r_k\rho_k^{-1}K_k\varepsilon_k\|\|f_k - \dot{\alpha}_{k-1} - \dot{\rho}_k\xi_k\|,\end{aligned}$$

from which we conclude the boundedness of ε_i and ξ_i as

$$\|\varepsilon_i(t)\| \leq \bar{\varepsilon}_i \Rightarrow \|\xi_i(t)\| \leq \bar{\xi}_i \coloneqq \frac{\exp(\bar{\varepsilon}_i)-1}{\exp(\bar{\varepsilon}_i)+1}, \tag{23}$$

for all $t \in I_t$ for positive finite constants $\bar{\varepsilon}_i$. As a consequence, all intermediate signals α_i and system states q_{i+1}, $i \in \{2,\ldots,k-1\}$, as well as the control law (12) remain bounded for all $t \in I_t$.

What remains to be shown is $t_{\max} = \infty$. Notice that (21) and (23) imply that the system remains bounded in a compact subset of Ω, i.e., $(\bar{q}(t),t) \in \bar{\Omega} \subset \Omega$, for all $t \in I_t$. Since $\bar{q}(t)$ has been proven bounded, the conditions of [60, Theorem 2.1.4] hold and we conclude hence that $\tau_{\max} = \infty$. □

ACKNOWLEDGMENT

The authors would like to thank Robin Baran for his decisive help in the hardware experiments.

REFERENCES

[1] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
[2] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.
[3] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transaction on Robotics and Automation*, vol. 8, pp. 501–518, 1992.
[4] ——, "The construction of analytic diffeomorphisms for exact robot navigation on star worlds," *Transactions of the American Mathematical Society*, vol. 327, no. 1, pp. 71–116, 1991.
[5] S. G. Loizou, "The navigation transformation," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1516–1523, 2017.
[6] P. Vlantis, C. Vrohidis, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Robot navigation in complex workspaces using harmonic maps," *2018 IEEE International Conference on Robotics and Automation*, pp. 1726–1731, 2018.
[7] S. G. Loizou, "The multi-agent navigation transformation: Tuning-free multi-robot navigation." *Robotics: Science and Systems*, vol. 6, pp. 1516–1523, 2014.
[8] C. K. Verginis, Z. Xu, and D. V. Dimarogonas, "Decentralized motion planning with collision avoidance for a team of uavs under high level goals," *IEEE International Conference on Robotics and Automation*, pp. 781–787, 2017.
[9] C. K. Verginis and D. V. Dimarogonas, "Adaptive robot navigation with collision avoidance subject to 2nd-order uncertain dynamics," *Automatica*, vol. 123, p. 109303, 2021.
[10] C. K. Verginis, A. Nikou, and D. V. Dimarogonas, "Communication-based decentralized cooperative object transportation using nonlinear model predictive control," *European Control Conference*, pp. 733–738, 2018.
[11] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.
[12] D. Panagou, D. M. Stipanović, and P. G. Voulgaris, "Distributed coordination control for multi-robot networks using lyapunov-like barrier functions," *IEEE Transactions on Automatic Control*, vol. 61, no. 3, pp. 617–632, 2015.
[13] C. K. Verginis and D. V. Dimarogonas, "Closed-form barrier functions for multi-agent ellipsoidal systems with uncertain lagrangian dynamics," *IEEE Control Systems Letters (L-CSS)*, 2019.
[14] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
[15] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Proceedings of International Conference on Robotics and Automation*, vol. 3, pp. 2719–2726, 1997.
[16] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.
[17] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 995–1001, 2000.
[18] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
[19] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.
[20] E. Vidal, M. Moll, N. Palomeras, J. D. Hernández, M. Carreras, and L. E. Kavraki, "Online multilayered motion planning with dynamic constraints for autonomous underwater vehicles," *IEEE International Conference on Robotics and Automation*, pp. 8936–8942, 2019.
[21] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.
[22] P. Reist, P. Preiswerk, and R. Tedrake, "Feedback-motion-planning with simulation-based lqr-trees," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1393–1416, 2016.
[23] A. Wu, S. Sadraddini, and R. Tedrake, "R3t: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems," *2020 IEEE International Conference on Robotics and Automation*, pp. 4245–4251, 2020.
[24] H. G. Tanner and J. L. Piovesan, "Randomized receding horizon navigation," *IEEE Transactions on Automatic Control*, vol. 55, no. 11, pp. 2640–2644, 2010.
[25] D. S. Yershov and E. Frazzoli, "Asymptotically optimal feedback planning using a numerical hamilton-jacobi-bellman solver and an adaptive mesh refinement," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 565–584, 2016.
[26] S. H. Kim and R. Bhattacharya, "Multi-layer approach for motion planning in obstacle rich environments," *AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 6603, 2007.
[27] J. Denny, R. Sandström, A. Bregger, and N. M. Amato, "Dynamic region-biased rapidly-exploring random trees," *Algorithmic Foundations of Robotics XII*, pp. 640–655, 2020.
[28] O. B. Bayazit, D. Xie, and N. M. Amato, "Iterative relaxation of constraints: A framework for improving automated motion planning," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3433–3440, 2005.
[29] J. Denny, E. Greco, S. Thomas, and N. M. Amato, "Marrt: Medial axis biased rapidly-exploring random trees," pp. 90–97, 2014.
[30] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[31] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," *2011 IEEE international conference on robotics and automation*, pp. 723–730, 2011.
[32] A.-A. Agha-Mohammadi, S. Agarwal, A. Mahadevan, S. Chakravorty, D. Tomkins, J. Denny, and N. M. Amato, "Robust online belief space planning in changing environments: Application to physical mobile robots," *IEEE International Conference on Robotics and Automation*, pp. 149–156, 2014.
[33] V. Rubies-Royo, D. Fridovich-Keil, S. Herbert, and C. J. Tomlin, "A classification-based approach for approximate reachability," *International Conference on Robotics and Automation*, pp. 7697–7704, 2019.
[34] M. Chen, S. L. Herbert, H. Hu, Y. Pu, J. F. Fisac, S. Bansal, S. Han, and C. J. Tomlin, "Fastrack: a modular framework for real-time motion planning and guaranteed safe tracking," *IEEE Transactions on Automatic Control*, vol. 66, no. 12, pp. 5861–5876, 2021.
[35] È. Pairet, J. D. Hernández, M. Lahijanian, and M. Carreras, "Uncertainty-based online mapping and motion planning for marine robotics guidance," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2367–2374, 2018.
[36] J. Le Ny and G. J. Pappas, "Sequential composition of robust controller specifications," *IEEE International Conference on Robotics and Automation*, pp. 5190–5195, 2012.
[37] B. D. Luders, S. Karaman, E. Frazzoli, and J. P. How, "Bounds on tracking error using closed-loop rapidly-exploring random trees," *American Control Conference*, pp. 5406–5412, 2010.
[38] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: a versatile and scalable robot simulation framework," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.
[39] C. Verginis, D. V. Dimarogonas, and L. Kavraki, "Sampling-based motion planning for uncertain high-dimensional systems via adaptive control," *International Workshop on the Algorithmic Foundations of Robotics*, 2021.
[40] C. C. De Wit, H. Olsson, K. J. Astrom, and P. Lischinsky, "A new model for control of systems with friction," *IEEE Transactions on automatic control*, vol. 40, no. 3, pp. 419–425, 1995.
[41] C. Makkar, W. Dixon, W. Sawyer, and G. Hu, "A new continuously differentiable friction model for control systems design," *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics.*, pp. 600–605, 2005.
[42] C. Verginis and D. V. Dimarogonas, "Asymptotic tracking of second-order nonsmooth feedback stabilizable unknown systems with prescribed transient response," *IEEE Transactions on Automatic Control*, 2020.
[43] H. K. Khalil, "Nonlinear Systems," *Prentice Hall*, 2002.
[44] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Robust adaptive control barrier functions: An adaptive and data-driven approach to safety," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 1031–1036, 2020.
[45] ——, "Dynamic tube mpc for nonlinear systems," *American Control Conference*, pp. 1655–1662, 2019.
[46] M. Zambelli, Y. Karayiannidis, and D. V. Dimarogonas, "Posture regulation for unicycle-like robots with prescribed performance guarantees," *IET Control Theory & Applications*, vol. 9, no. 2, pp. 192–202, 2014.
[47] C. P. Bechlioulis, G. C. Karras, S. Heshmati-Alamdari, and K. J. Kyriakopoulos, "Trajectory tracking with prescribed performance for underactuated underwater vehicles under model uncertainties and external disturbances," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 429–440, 2016.
[48] C. K. Verginis, C. P. Bechlioulis, A. G. Soldatos, and D. Tsipianitis, "Robust trajectory tracking control for uncertain 3-dof helicopters with prescribed performance," *IEEE/ASME Transactions on Mechatronics*, 2022.
[49] C. Bechlioulis and G. Rovithakis, "Robust Adaptive Control of Feedback Linearizable MIMO Nonlinear Systems with Prescribed Performance," *IEEE Transactions on Automatic Control (TAC)*, vol. 53, no. 9, pp. 2090–2099, 2008.
[50] A. Ilchmann, E. P. Ryan, and P. Townsend, "Tracking with prescribed transient behavior for nonlinear systems of known relative degree," *SIAM Journal on Control and Optimization*, vol. 46, no. 1, pp. 210–230, 2007.
[51] J.-J. E. Slotine and W. Li, "On the adaptive control of robot manipulators," *The international journal of robotics research*, vol. 6, no. 3, pp. 49–59, 1987.
[52] S. Bhat and D. Bernstein, "A Topological Obstruction to Continuous Global Stabilization of Rotational Motion and the Unwinding Phenomenon," *Systems and Control Letters*, vol. 39, no. 1, pp. 63–70, 2000.
[53] C. K. Verginis, M. Mastellaro, and D. V. Dimarogonas, "Robust cooperative manipulation without force/torque measurements: Control design and experiments," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 713–729, 2019.
[54] A. F. van der Stappen, D. Halperin, and M. H. Overmars, "The complexity of the free space for a robot moving amidst fat obstacles," *Computational Geometry*, vol. 3, no. 6, pp. 353–373, 1993.
[55] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.
[56] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.
[57] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," *IEEE international conference on robotics and automation*, pp. 3067–3074, 2015.
[58] X. Tan, W. S. Cortez, and D. V. Dimarogonas, "High-order barrier functions: Robustness, safety and performance-critical control," *IEEE Transactions on Automatic Control*, 2021.
[59] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.
[60] A. Bressan and B. Piccoli, *Introduction to the Mathematical Theory of Control*. American institute of mathematical sciences Springfield, 2007, vol. 2.