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
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 \( q^t = [q_1^t, \ldots, q_n^t]^T \in \mathbb{R}^{n_1} \) and \( q^r = [q_1^r, \ldots, q_n^r]^T \in \{0, 2\pi\}^{n_r} \), respectively, with \( n_1 + n_r = n \), and hence \( \mathbb{T} := \mathbb{W}_{n_r} \times [0, 2\pi)^{n_r} \), where \( \mathbb{W}_{n_r} \) is a closed subset of \( \mathbb{R}^{n_{r}} \). Without loss of generality, we assume that \( q_1 = [(q^t)^T, (q^r)^T]^T \).

We consider \( k \)-th order systems, with \( k \geq 2 \), of the form

\[
\dot{q}_i = f_i(q_i, t) + g_i(q_i, t)\dot{q}_{i+1}, \quad \forall i \in \{1, \ldots, k - 1\} \quad (1a)
\]

\[
\dot{q}_k = f_k(q_k, t) + g_k(q_k, t)u, \quad (1b)
\]

where \( \dot{q}_i := [q_i^T, \ldots, q_i^{T(i-1)}]^T \in \mathbb{T} \times \mathbb{R}^{n(i-1)} \), for all \( i \in \{1, \ldots, k\} \), and \( u \in \mathbb{R}^n \) is the control input of the system. Note that the \( k \)-th 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 \( i \in \{1, \ldots, k\} \). The only assumptions we make for the system are mild continuity and controllability conditions, as follows:

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

**Assumption 2.** It holds that

\[
\lambda_{\min}\left(g_i(q_i, t) + g_i(q_i, t)^T\right) \geq \lambda > 0,
\]

for a positive constant \( \lambda \), for all \( q_i \in \mathbb{R}^{n(i-1)}, t \geq 0, i \in \{1, \ldots, k\} \), 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 \( \dot{q}_i \) and bounded in time \( t \). The smoothness in \( \dot{q}_i \) 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 \( \dot{q}_i \). The proposed algorithm guarantees the boundedness of \( \dot{q}_i \) and, consequently, of \( f_i(\cdot), g_i(\cdot) \).

**Assumption 2.** It holds that

\[
\lambda_{\min}\left(g_i(q_i, t) + g_i(q_i, t)^T\right) \geq \lambda > 0,
\]

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

\[
-\rho(t) < e(t) < \rho(t), \quad \forall t \geq 0,
\]

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 \( \rho(t) := (\rho_0 - \rho_\infty)e^{-\lambda t} + \rho_\infty \). The parameter \( \rho_\infty := \lim_{t\to\infty} \rho(t) > 0 \) 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 \( \lambda \) 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 \( \lambda \) and small \( \rho_\infty \) improve the performance of the system, yielding fast convergence close to zero. Although these constants can be arbitrarily 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 \( T \left( \frac{e(t)}{\rho(t)} \right) \), where \( T : (-1, 1) \to (-\infty, \infty) \) is a strictly increasing, odd and bijective mapping. Then the funnel specifications are met by simply preserving the boundedness of \( T \left( \frac{e(t)}{\rho(t)} \right) \). 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 \( T \), and we show how such a design can be used in the motion planning of the uncertain robotic system (1).

Let \( q_\Delta := [(q_d^1)^T, (q_d^2)^T]^T := [q_{d_1}, \ldots, q_{d_n}, q_{d_1}, \ldots, q_{d_n}]^T : [t_0, t_0 + t_f] \to T \) be a smooth (at least \( k \)-times continuously differentiable) reference trajectory, with \( q_{d_1} \in \mathbb{R}^{n_{tr}} \) and \( q_{d_2} \in [0, 2\pi)^{n_{ro}} \) 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 \( q_\Delta \) 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 \( q_\Delta(t) \), despite the unknown terms \( f_i, g_i \). We start by defining the appropriate error metric between \( q_i \) and \( q_\Delta \), which represents their distance. Regarding the translational part, we define the standard Euclidean error \( e^t := q^t - q_{d_1}^t \). For the rotation part, however, the same error \( e^r := q^r - q_{d_2}^r \) does not
represent the minimum distance metric, since \( q^* \) evolves on the \( n_x \)-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 \( e^2 \) to zero (e.g., [51]), we use the chordal metric

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

extended for vector arguments \( x = [x_1, \ldots, x_n] \in [0, 2\pi)^n \) to

\[
d_C(x) := \sum_{\ell \in \{1, \ldots, n\}} d_C(x_j). \tag{3}
\]

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

\[
\eta^\ell := d_C(e^\ell) = 1 - \cos(e^\ell),
\]

where \( e^\ell := q^\ell - q^\ell_0 \) is the \( \ell \)th element of \( e^r \), \( \ell \in \{1, \ldots, n_r\} \). Differentiation yields

\[
d_C(e^\ell) = \sin(e^\ell)e^\ell, \tag{4d}
\]

for all \( \ell \in \{1, \ldots, n_r\} \), which is zero when \( e^\ell = 0 \) or \( e^\ell = \pi \). The second case is an undesired equilibrium, which implies that the point \( e^\ell = 0 \) 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 \( e^\ell(t) \) evolves in a predefined funnel, guarantees that \( e^\ell(t) \neq \pi \), for all \( t \in (t_0, t_f) \), provided that \( e^\ell(t_0) \neq \pi \), for all \( \ell \in \{1, \ldots, n_r\} \).

The funnel is defined by the functions \( \rho^j_1 : [t_0, t_0 + t_f] \rightarrow [-\rho^j_1, \rho^j_1], \rho^j_2 : [t_0, t_0 + t_f] \rightarrow [-\rho^j_2, \rho^j_2] \) with initial and final values \( \rho^j_1(0), \rho^j_1(t_f) \) and \( \rho^j_2, \rho^j_2 \), respectively, i.e.,

\[
\begin{align*}
\rho^j_1(t_0) &= \rho^j_1(0) = \rho^j_1, \\
\rho^j_2(t_0 + t_f) &= \rho^j_2, \rho^j_2(t_0 + t_f) = \rho^j_2, \\
0 &< \rho^j_2 \leq \rho^j_1, 0 < \rho^j_2 \leq \rho^j_2 < 2,
\end{align*} \tag{4a,b,c}
\]

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

\[
|e^\ell_j(t_0)| < \rho^\ell_1, \quad \eta^\ell_j(t_0) < \rho^\ell_2, \tag{4d}
\]

for all \( j \in \{1, \ldots, n_r\}, \ell \in \{1, \ldots, n_r\} \). The functions \( \rho^j_1, \rho^j_2 \) are assumed to be smooth, with bounded derivatives, for all \( j \in \{1, \ldots, n_r\}, \ell \in \{1, \ldots, n_r\} \). Our aim is to design a control protocol such that

\[
\begin{align*}
|e^\ell_j(t)| &< \rho^\ell_1, \forall j \in \{1, \ldots, n_r\}, \tag{5a} \\
\eta^\ell_j(t) &< \rho^\ell_2, \forall \ell \in \{1, \ldots, n_r\}. \tag{5b}
\end{align*}
\]

for all \( t \in [t_0, t_0 + t_f] \). Note that, since \( \rho^\ell_2 < 2 \), guaranteeing (5b) ensures that \( \eta^\ell_j(t) < 2 \), i.e., \( e^\ell_j(t) \neq \pi \), for all \( t \in [t_0, t_0 + t_f], \ell \in \{1, \ldots, n_r\} \) and avoidance of the respective singularities. 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 \( e^\ell_j, e^\ell_j \). For instance, for the exponentially decaying \( \rho^\ell_j(t) = (\rho^\ell_1 - \rho^\ell_1) \exp(-\lambda_j t) + \rho^\ell_1, \forall t \in [t_0, t_0 + t_f], \) a user can choose the constants \( \rho^\ell_1, \rho^\ell_2, \lambda_j \) 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 \( \rho^\ell_1, \rho^\ell_2 \) 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

\[
\xi^\ell_j := \frac{e^\ell_j}{\rho^\ell_j}, \quad \forall j \in \{1, \ldots, n_r\}, \tag{6a}
\]

\[
\xi^\ell_\ell := \frac{\eta^\ell_j}{\rho^\ell_j}, \quad \forall \ell \in \{1, \ldots, n_r\}. \tag{6b}
\]

Note that, in order for the errors \( e^\ell_j \) and \( \eta^\ell_j \) to satisfy the funnel constraints, the control design must guarantee that \( \xi^\ell_j \in (-1,1) \) and \( \xi^\ell_\ell \in [0,1) \). In order to do that, we define the transformed errors and signals

\[
\begin{align*}
\varepsilon^\ell_j &= \ln \left( 1 + \frac{\xi^\ell_j}{1 - \xi^\ell_j} \right), \quad \forall j \in \{1, \ldots, n_r\}, \tag{7a} \\
\varepsilon^\ell_\ell &= \ln \left( 1 + \frac{\xi^\ell_\ell}{1 - \xi^\ell_\ell} \right), \quad \forall \ell \in \{1, \ldots, n_r\}, \tag{7b} \\
r^\ell_1 &= \frac{\partial \varepsilon^\ell_j}{\partial \xi^\ell_j} = 2 \left( 1 - (\xi^\ell_j)^2 \right), \quad \forall j \in \{1, \ldots, n_r\}, \tag{7c} \\
r^\ell_\ell &= \frac{\partial \varepsilon^\ell_\ell}{\partial \xi^\ell_\ell} = \frac{1}{1 - \xi^\ell_\ell}, \quad \forall \ell \in \{1, \ldots, n_r\}. \tag{7d}
\end{align*}
\]

Note that \( \varepsilon^\ell_j, r^\ell_1 \), and \( \varepsilon^\ell_\ell, r^\ell_\ell \) diverge to infinity as \( \xi^\ell_j \) and \( \xi^\ell_\ell \) approach 1, respectively. The control design exploits this property; it aims to keep these signals bounded in order to achieve \( \xi^\ell_j(t) \in (-1,1) \) and \( \xi^\ell_\ell(t) \in [0,1) \). 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 \( \alpha_2 := [(\alpha^r)^T, (\alpha^\ell)^T]^T \), with

\[
\begin{align*}
\alpha^r := -\begin{bmatrix} k_1^r r_1^r \varepsilon^r_1, \ldots, k_1^r m \rho^r_{n_r} r_1^r \varepsilon^r_{n_r} \end{bmatrix}^T, & \quad \tag{8a} \\
\alpha^\ell := -\begin{bmatrix} k_1^\ell \rho^\ell_1 \sin(e^\ell_1), \ldots, k_1^\ell \rho^\ell_1 \rho^\ell_{n_r} \sin(e^\ell_{n_r}) \end{bmatrix}^T, & \quad \tag{8b}
\end{align*}
\]

where \( k_1^r, k_1^\ell \) are positive gain constants, \( j \in \{1, \ldots, n_r\}, \ell \in \{1, \ldots, n_r\} \).

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

\[
e_i := \begin{bmatrix} e_i, \ldots, e_i \end{bmatrix}^T := q_i - q_{i-1} \in \mathbb{R}^n, \tag{9}
\]

where \( q_{i-1} \) will be given subsequently in (11). We design funnel functions \( \rho^i_m : [t_0, t_0 + t_f] \rightarrow [\rho^i_m, \tilde{\rho}^i_m], \rho^i_m \leq \tilde{\rho}^i_m \), such that \( \rho^i_m(t_0) = \tilde{\rho}^i_m > |e^i_{m}(t_0)| \) for all \( m \in \{1, \ldots, n\} \), and define

\[
\xi_i := \begin{bmatrix} \xi_i, \ldots, \xi_i \end{bmatrix}^T := \rho^{-1}_i e_i, \tag{10a}
\]

1Note that \( e^i_{m}(t_0) \) can be measured at the time instant \( t_0 \) and the functions \( \rho^i_m \) can be designed accordingly.
\[ e_i := [e_{i1}, \ldots, e_{in}]^T := \left[ \ln \left( \frac{1 + \xi_{i1}}{1 - \xi_{i1}} \right), \ldots, \ln \left( \frac{1 + \xi_{in}}{1 - \xi_{in}} \right) \right]^T \]  
Eq. (10b)

\[ r_i := \text{diag} \left\{ \left[ \frac{\partial e_{im}}{\partial x_{im}} \right]_{m \in \{ 1, \ldots, n \}} \right\}, \]  
Eq. (10c)

where \( \rho_i := \text{diag} \left\{ [\rho_{im}]_{i \in \{ 1, \ldots, n \}} \right\} \in \mathbb{R}^{n \times n}. \) Finally, we design the intermediate reference signals as

\[ \alpha_i := -K_i \rho_i^{-1} r_i e_i, \quad \forall i \in \{ 2, \ldots, k - 1 \}, \]  
Eq. (11)

and the control law

\[ u = -K_k \bar{\rho}_k^{-1} r_k e_k, \]  
Eq. (12)

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

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 \( |e_{i1}|, 1 - e_{i1}, |e_{in}|, i \in \{ 2, \ldots, k - 1 \} \) approach the respective funnel functions \( \rho_f^1, \rho_f^2, \) \( \rho_{im}, \quad j \in \{ 1, \ldots, n_r \}, \ell \in \{ 1, \ldots, n_r \}, \quad m \in \{ 1, \ldots, n \}, \quad i \in \{ 2, \ldots, k \}. \) 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 \( f_i(\cdot), \quad g_i(\cdot), \) 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 \( K^\xi, K^\rho, K, \quad i \in \{ 2, \ldots, k \} \) 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 \( \bar{\epsilon}, \xi^t < 1, \xi^r < 1, \xi^e < 1 \) that satisfy

\[ |e_{i1}^t(t)| \leq \bar{\epsilon}, |e_{i1}^r(t)| \leq \xi^t, |e_{i1}^e(t)| \leq \xi^e, \]  
Eq. (13a)

\[ n_{i1}^t(t) = 1 - \cos(e^{e^t}(t)) < \rho_{i1}^e(t), \quad \forall \ell \in \{ 1, \ldots, n_r \} \]  
Eq. (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. \( \square \)
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., $|e_j^T(t_0)| < \rho_j^r$, $j \in \{1, \ldots, n_r\}$, $\ell \in \{1, \ldots, n_r\}$. 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 $q_d(0) = q(0)$, which implies that the aforementioned constraint is trivially satisfied. Moreover, the selection of $\rho_j^r$, $\rho_j^s$ 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 $\rho_j^r(t) = 0.05 \exp(-0.01 t) + 0.1$, $\rho_j^s(t) = 0.005 \exp(-0.01 t) + 0.005$, implying shrinking funnels from 0.1 to 0.05 and from 0.01 to 0.005 (rad), respectively, with exponential convergence dictated by $\exp(-0.01 t)$, 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

$$\mathcal{P}(z, \bar{\rho}) := \{ y \in \mathbb{T} : |y_j^r - z_j^r| < \rho_j^r, 1 - \cos(y_j^s - z_j^s) < \rho_j^s, \forall j \in \{1, \ldots, n_r\}, \ell \in \{1, \ldots, n_r\} \}$$

where $y, z \in \mathbb{T}$ consist of translational and rotational terms (similarly to $q_1$), and $\bar{\rho} := [\bar{\rho}_1^r, \ldots, \bar{\rho}_m^r, \bar{\rho}_1^s, \ldots, \bar{\rho}_n_s]^T \in \mathbb{R}^{n_r + n_r}$ is the vector of maximum funnel values. We define now, similarly to [9], the $\bar{\rho}$-extended free space

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

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

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 $r$ evolves on the $n_r$-dimensional sphere. Hence, having already defined the chordal metric $\tilde{d}_C$ in (3), we define a suitable distance metric for vectors $x = [(x_1^T, (x_2^T)]^T$, $y = [(y_1^T, (y_2^T)]^T \in \mathbb{T}$ as $d_T(x, y) = \|x_1^T - y_1^T\|^2 + \tilde{d}_C(x_2^T - y_2^T)$. \hfill (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 $q_d : [t_0, t_0 + t_f] \to \mathbb{T}$, the control algorithm (6)-(12) guarantees that $q_1(t) \in \mathcal{P}(q_2(t), \bar{\rho})$ for all $t \in [t_0, t_0 + t_f]$. Therefore, by the construction of $\mathcal{A}_{\text{free}}(\bar{\rho})$, if $q_d(t)$ belongs to $\mathcal{A}_{\text{free}}(\bar{\rho})$, $q_1(t)$ belongs to $\mathcal{A}_{\text{free}}$. The proposed sampling-based framework aims at finding a path in $\mathcal{A}_{\text{free}}(\bar{\rho})$; this path will be then endowed with time constraints in order to form the trajectory $q_d : [t_0, t_0 + t_f] \to \mathcal{A}_{\text{free}}(\bar{\rho})$, 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 $G = (\mathcal{V}, \mathcal{E})$, (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 Sample(), Nearest($\mathcal{G}$, $y$), Closest($\mathcal{G}$, $y$, $K$), Steer($y$, $z$), and ObstacleFree($y$, $z$); Sample() samples a random point from a distribution in $\mathcal{A}_{\text{free}}$; Nearest($\mathcal{G}$, $y$) and Closest($\mathcal{G}$, $y$, $K$) find the closest and $K$ closest, respectively, nodes of $\mathcal{G}$ to $y$, according to some distance metric; Steer($y$, $z$) computes a point lying on line from $z$ to $y$ and ObstacleFree($y$, $z$) 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 Sample() and ObstacleFree($y$, $z$), 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 $\mathcal{A}_{\text{free}}(\bar{\rho})$. To this end, we need to sample points and perform collision checking in $\mathcal{A}_{\text{free}}(\bar{\rho})$. Therefore, we define the functions SampleExt($\bar{\rho}$) and ObstacleFreeExt($y$, $z$, $\bar{\rho}$); SampleExt($\bar{\rho}$) samples a point from a uniform distribution in the extended free space $\mathcal{A}_{\text{free}}(\bar{\rho})$; ObstacleFreeExt($y$, $z$, $\bar{\rho}$) checks whether the path $X_{\text{line}} : [0, \sigma] \to \mathbb{T}$, for some positive $\sigma$, from $y$ to $z$ is collision free with respect to the extended
free space, i.e., check whether \( y' \in \mathcal{A}_{\text{free}}(\bar{\rho}), \forall y' \in X_{\text{line}} \). We elaborate on the collision checking procedure in Remark 6.

The new functions \( \text{SampleExt}() \) and \( \text{ObstacleFreeExt}() \) can be used in any geometric sampling-based motion planning algorithm, giving rise to a new family of algorithms, which produce a safe path in an extended free space \( \mathcal{A}_{\text{free}}(\bar{\rho}) \). 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 \textit{time-varying} smooth (at least \( k \)-times continuously differentiable) trajectory \( q_d(t) \), whereas the output of the respective motion planning algorithm is a path, i.e., a sequence of points in \( \mathbb{T} \). Therefore, we smooth 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 \( \mathcal{A}_{\text{free}}(\bar{\rho}) \), the addition of time constraints, and the funnel-control algorithm of Section III-A, constitute the framework of \textit{KinoDynamic motion planning via Funnel control}, or KDF motion-planning (KDF-MP) framework. Further, note that the smoothing 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 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 \( \mathcal{A}_{\text{free}}(\bar{\rho}) \) (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 \( \mathcal{A}_{\text{free}}(\bar{\rho}) \) (note that if \( \rho \) consists of zeros, then \( \mathcal{A}_{\text{free}}(\bar{\rho}) = \mathcal{A}_{\text{free}} \)), 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 \( \mathcal{A}_{\text{free}}(\bar{\rho}) \). 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 \( \rho_c \in \mathbb{R}^{n_x+\ell r} \) such that \( Q_g \in \mathcal{A}_{\text{free}}(\rho_c)_i^2 \). Hence, by choosing \( \rho_c \) such that \( \rho_c \geq \bar{\rho} \), one can achieve \( \mathcal{A}_{\text{free}}(\rho_c) \subseteq \mathcal{A}_{\text{free}}(\bar{\rho}) \) and hence \( Q_g \in \mathcal{A}_{\text{free}}(\bar{\rho}) \). 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 \( \rho_c \geq \bar{\rho} \) (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 \( \mathcal{A}_{\text{free}}(\bar{\rho}) \). That is, \( Q'_g := \text{arg min}_{q \in \mathcal{A}_{\text{free}}(\bar{\rho})} d_T(q, Q_g) \), where \( \mathcal{A} \) is a compact subset of \( \mathcal{A}_{\text{free}}(\bar{\rho}) \) 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 \textit{time-varying} smooth (at least \( k \)-times continuously differentiable) trajectory \( q_d(t) \), 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 \( \mathcal{A}_{\text{free}}(\bar{\rho}) \), so that the smoothed path still belongs in \( \mathcal{A}_{\text{free}}(\bar{\rho}) \). Time constraints are then enforced on the smooth path to create a timed trajectory \( q_d : [0, t_f] \rightarrow \mathcal{A}_{\text{free}}(\bar{\rho}), \) for some \( t_f > 0 \), which is then given to the control protocol of Section III-A as the desired trajectory input. Note that \( q_d(0) \) 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 \( \dot{q}_d \) 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 \( \bar{\rho} = \left[ \bar{\rho}_1^1, \ldots, \bar{\rho}_n^1, \bar{\rho}_1^r, \ldots, \bar{\rho}_n^r \right]^T \) (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 \( N \). 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:** \( \mathcal{A}_{\text{free}}, Q_g, N, t_f, k, q_1(0), q_a, \rho_j^1, \rho_j^r, K^1, K^r, K_i \), \( j \in \{1, \ldots, n_r\}, \ell \in \{1, \ldots, n_r\} \)**

**Output:** \( u(t) \)

```plaintext
1: procedure KDF
2: \( \bar{\rho} \leftarrow \text{Bounds}(\rho_1^1, \rho_1^r); \)
3: \( q_p \leftarrow \text{KDF-MP}(\bar{\rho}, \mathcal{A}_{\text{free}}, Q_g, q_1(0), N); \)
4: \( q_d \leftarrow \text{TimeTraj}(q_p, t_f); \)
5: \( u \leftarrow \text{FunnelControl}(t_f, k, q_a, \rho_j^1, \rho_j^r, K^1, K^r, K_i); \)
```

Since the free space \( \mathcal{A}_{\text{free}} \) and the goal configuration \( Q_g \) are known, such a \( \rho_c \) can be explicitly found.
Remark 6 (Collision Checking in $\tilde{A}_{\text{free}}(\bar{\rho})$). The proposed feedback control scheme guarantees that $q(t) \in P(q_0(t), \bar{\rho})$ for any trajectory $q_0(t)$, formed by the sampled points $q_s$ of a KDF-MP algorithm. Therefore, checking whether the points $q_s$ belong to $\tilde{A}_{\text{free}}$, as in standard motion planners [14], is not sufficient. For each such point $q_s$, one must check whether $z \in \tilde{A}_{\text{free}}$. For all $z \in P(q_s, \bar{\rho})$, which is equivalent to checking if $q_s \in \tilde{A}_{\text{free}}(\bar{\rho})$. 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 $P(q_s, \bar{\rho})$ 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 from $P(q_s, \bar{\rho})$ and check whether they belong to $\tilde{A}_{\text{free}}$. Note that the more points we sample from $P(q_s, \bar{\rho})$, the better exploration of $\tilde{A}_{\text{free}}(\bar{\rho})$ we achieve, guaranteeing hence that $q_s \in \tilde{A}_{\text{free}}(\bar{\rho})$. 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 $P(q_s, \bar{\rho})$ can be considered to be complete, i.e., the resulting path will belong to the extended free space $\tilde{A}_{\text{free}}(\bar{\rho})$. 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.

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

$$q_1 = f_1(q_1, t) + g_1(q_1, t)u$$
$$q_2 = f_2(q_1, q_2, t) + g_2(q_1, q_2, t)u$$

where $q_1 = [q_1^1, q_1^2, q_1^3]^T$, $q_2 \in \mathbb{R}^3$ 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 $Q_g = [-3, -4, 3]^T$, starting from $q_1(0) = [-4.5, -4.2, 0]^T$ 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 $\rho_1(t) = \rho_2(t) = \rho_3(t) = \rho(t) = 0.15 \exp(-0.1t) + 0.05 \in [0.05, 0.2]$ (meters), implying $\bar{\rho} = 0.2[1, 1, 1]^T$, as well as $\rho_2(t) := (\max\{2|e_2(0)|, 0.5\} - 0.1) \exp(-0.1t) + 0.1 \in [0.1, \max\{2|e_2(0)|, 0.5\}]$ (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
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 \( \mathcal{A}_{\text{free}}(\bar{\rho}) \) and is converted to a smooth time trajectory as follows. We construct \( q_\theta: [0, 100] \rightarrow \mathbb{R}^3 \), such that \( q_\theta(0) = q_1(0) \) and \( q_\theta(t) = Q_\theta \), for \( t \in [90, 100] \), using a standard fitting procedure. For the construction of \( q_\theta(t) \), each pair of two path points \( h = [h_1, h_2, h_3]^\top, w = [w_1, w_2, w_3]^\top \in \mathcal{A}_{\text{free}}(\bar{\rho}) \) is endowed with a time duration proportional to their distance, i.e., equal to \( \frac{1}{2} \max_{i\in\{1,2,3\}} \{h - w_i\} \). 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 \( K_1^\top = \text{diag} \{ [k_{ij}^1]_{i\in\{1,2,3\}} \} = 2I_3 \) 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 \( e_1^j(t), e_2^j(t), e_3^j(t) \) (in meters and meters/second, respectively) along with the respective funnel functions \( \rho_1^j(t), \rho_2^j(t), \rho_3^j(t) \). 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 \( D_{\text{UA V}}(t) \) (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 \( \mathcal{A}_{\text{free}}(\bar{\rho}) \), 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 \( q_1 = [q_{11}, \ldots, q_{1n}]^\top = [q_1^4, q_2, q_3]^\top \),

---

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

**Fig. 4:** Top: the evolution of the errors \( e_1^j(t), e_2^j(t), e_3^j(t) \) (in meters), along with the funnels \( \rho_1^j(t), \rho_2^j(t), \rho_3^j(t) \), shown in black, for \( t \in [0, 90] \) seconds. Bottom: the evolution of the velocity errors \( e_1^j(t), e_2^j(t), e_3^j(t) \) (in meters/second), along with the funnels \( \rho_1^j(t), \rho_2^j(t), \rho_3^j(t) \), shown in black, for \( t \in [0, 90] \) seconds.

**Fig. 5:** Top: the distance \( D_{\text{UA V}}(t) \) (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 \( q_0 = [0, 0, 0, 0, 0, 0]^T \) to the following four target points (shown in Fig. 6).

- **Target 1:** \( T_1 = [-0.15, -0.475, 0.675]^T \) and Euler-angle orientation \([\pi/2, 0, 0]^T\), which yields the configuration \( q_{T_1} = [-0.07, -1.05, 0.45, 2.3, 1.37, -1.33]^T\).
- **Target 2:** \( T_2 = [-0.6, 0.25]^T \) and Euler-angle orientation \([0, -\pi/2, -\pi/2]^T\), which yields the configuration \( q_{T_2} = [1.28, 0.35, 1.75, 0.03, 0.1, -1.22]^T\).
- **Target 3:** \( T_3 = [-0.025, 0.595, 0.6]^T \) and Euler-angle orientation \([ -\pi, 0, 0]^T\), which yields the configuration \( q_{T_3} = [-0.08, 0.85, -0.23, 2.58, 2.09, -2.36]^T\).
- **Target 4:** \( T_4 = [-0.525, -0.55, 0.28]^T \) and Euler-angle orientation \([\pi, 0, -\pi/2]^T\), which yields the configuration \( q_{T_4} = [-0.7, -0.76, -1.05, -0.05, -3.08, 2.37]^T\).

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 \( \rho = 0.01 [1, 15, 15, 15, 15] \), as will be elaborated later. Regarding the collision checking in \( \mathcal{A}_{\text{free}}(\rho) \) 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 \( \mathcal{A}_{\text{free}}(\rho) \) for collision. One can also notice that the time increases with the number of samples. However, more samples imply greater coverage of \( \mathcal{A}_{\text{free}}(\rho) \) 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 \( \mathcal{A}_{\text{free}}(\rho) \), 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 \([0, 0]^T\) and a randomly chosen goal configuration at \([ -\pi/2, \pi/2]^T\) 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 \( d_g \)), 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.

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 \( q_{d,i}^j, q_{l,i}^j, j \in \{1, \ldots, 5\} \), with time duration \( t_f^j = 11 \) 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 \( \rho_j^{1,i}(t) = 0.05 \exp(-0.01(t - t_s)) + 0.1 \in [0.1, 0.15] \) (rad), for all \( j \in \{1, \ldots, 5\} \), and \( \rho_3^{1,i}(t) = 0.005 \exp(-0.01(t - t_p)) + 0.005 \in [0.005, 0.01] \), (implying \( \bar{\rho} = 0.01\) [1, 15, 15, 15, 15]), as well as \( \rho_2^{1,i}(t) = 2 \max\{\max_{j \in \{1, \ldots, 5\}}\{ |e_2(j, t_p)| \}, 0.25\} \), where \( \{p_0, t_{p1}, t_{p2}, t_{p3}\} := \{0, 11, 22, 33, 44\} \) are the starting times of the four paths. We further chose \( K_1^* = 1, K^1 = \text{diag}\{[k^1_{d,j}]_{j \in \{1, \ldots, 5\}}\} = \text{diag}(1, 1.5, 5, 5, 5) \) and \( K_2 = 0.116 \) 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 \( \eta^i_j(t) e_j^i(t) \) (top and bottom in \( \cos(\text{rad}) \) and rad, respectively), which always satisfy the funnel conditions defined by the respective funnel functions \( \rho_j^i(t) \), \( j \in \{1, \ldots, 5\} \). 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).

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

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.

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

\[
\dot{q}_1 = q_2 \\
\dot{q}_2 = \frac{1}{m} \left( d_1(q_2) + d_2(t) \right) + \frac{1}{m} u
\]

where \( m = 1 \) is the UAV’s mass, \( d_1(q_2) = -0.5q_2 - 0.25\text{diag}\{q_2\}q_2 \), \( d_2(t) = \left[ \sin(10t + \frac{\pi}{6}), 0.5\cos(5t - \frac{\pi}{4}), 0.75\sin(10t + \frac{\pi}{3}) \right]^T \) are unknown vectors of state and time,
representing uncertainties (e.g., aerodynamic terms) and time-varying disturbances, respectively; \( \text{diag} \{ q_2 \} \) 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 \( e_j^i(t) \), along with the funnel functions \( \rho^i_j(t) \), \( j \in \{1, 2, 3\} \), 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 \( C = \{ q \in \mathbb{R}^n : \psi(q) > 0 \} \), for some sufficiently smooth function \( \psi \). In order to apply such a methodology to the considered environment, shown in Fig. 2, we set \( \psi_{jk}(e_j^i) = \bar{\rho}_M - e_j^i \), \( j \in \{1, 2, 3\} \), aiming to retain the system in the safe set \( \{ e_j^i, j \in \{1, 2, 3\} \} \), 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 \( h_{jk}(q, t) = \psi_{jk}(e_j^i) + \psi_{jk}(e_j^3)^2 \), \( j \in \{1, 2, 3\} \), \( k \in \{1, 2\} \), and choose the control input as the solution of the optimization problem

\[
\begin{align*}
\min_{u \in U} & \|u\|^2 \\
\text{s.t.} & \quad \frac{\partial h_{jk}}{\partial t} + \frac{\partial h_{jk}}{\partial q} q_2 + \frac{\partial h_{jk}}{\partial q_2} \sqrt{\frac{1}{m}} u + h_{jk} > 0, \\
& \quad \forall j \in \{1, 2, 3\}, k \in \{1, 2\}
\end{align*}
\]

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 \( e_j^i(t) \), \( j \in \{1, 2, 3\} \), 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 \( |e_j^i(t)| < \bar{\rho}_M \), showing the large dependence of the CBF method to the system dynamics.

Fig. 12: The evolution of the velocity errors \( e_j^2(t) \) (in rad/seconds), along with the respective funnels \( \rho_j^2(t) \) (in black), \( j \in \{1, \ldots, 6\} \), 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 \( u(t) = [u_1(t), \ldots, u_6(t)]^T \) (in Newton · meters) for the four paths.

Fig. 14: Results of the funnel-control algorithm for the UAV scenario of the comparative simulation studies. Top: the evolution of the errors \( e_j^1(t) \), \( e_j^2(t) \), \( e_j^3(t) \) (in meters), along with the funnels \( \rho_j^1(t) = \rho_j^2(t) = \rho_j^3(t) \), 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: method to the system dynamics.
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

$$\begin{align*}
\min_{\tilde{u}} & \int_t^{t+T_p} (\tilde{e}(\tau)^T Q \tilde{e}(\tau) + \tilde{u}(\tau)^T R \tilde{u}(\tau)) d\tau \\
& + \tilde{e}(t+T_p)^T P \tilde{e}(t+T_p), \\
n \text{s.t.} & \quad \ddot{q}_1 = \frac{1}{m} \tilde{u}, \\
& \quad \ddot{q}_1(t) = q_1(t), \quad \ddot{q}_2(t) = q_2(t) \\
& \quad |\tilde{e}_j(\tau)| < \tilde{\rho}_M, \quad \tau \in [t, t+T_p], \quad j \in \{1, 2, 3\} \\
& \quad \tilde{u}(\tau) \in U, \quad \tau \in [t, t+T_p] \\
& \quad \tilde{e}(t+T_p) \in \Omega
\end{align*}$$

where $T_p$ is the prediction horizon, $e = [(e_1)^T, (e_1)^T]^T$, $Q \in \mathbb{R}^{6 \times 6}$, $R \in \mathbb{R}^{6 \times 6}$, and $P \in \mathbb{R}^{6 \times 6}$ are positive-definite matrices, $\Omega$ is a terminal region, and the barred variables denote the predicted signals driven by the optimization solution $\tilde{u}(t)$. 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 $|e_j(\tau)| < \tilde{\rho}_M$, $j \in \{1, 2, 3\}$, through the optimization constraints. Similarly to the CBF controller, MPC is agnostic to the model uncertainties and disturbances, but uses an estimated mass value $\tilde{m} = 1.5m$.

We further choose $T_p = 20$, $Q = P = 100I_6$, $R = I_3$, $\Omega = \{x \in \mathbb{R}^3 : \|x\| \leq 0.1\}$, and a sampling frequency of $200\text{Hz}$. 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 behavior 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 $T_2 \rightarrow T_3$ of the UR5 scenario described in the previous section. We use the simulated dynamics:

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

where $B \in \mathbb{R}^{6 \times 6}$ is the gravity vector, and $d_1(q_2) = -q_2 - \text{diag}\{q_2\}q_2$, $d_2(t) = \left[\sin(t0 + \frac{\pi}{6}), 0.5 \cos(5t - \frac{\pi}{6}) \right]$, $\frac{\pi}{6}$. The results are depicted in Fig. 16, which shows the evolution of the errors $e_1(t), e_2(t)$, along with the respective funnel functions $\rho_j^f(t)$, $j \in \{1, \ldots, 5\}$, 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. 15: The evolution of the errors $e_1(t)$, $e_2(t)$, $e_3(t)$ (in meters), along with the funnel constant $\tilde{\rho}_M = 0.2$, shown in black, for the CBF methodology (top) and the MPC algorithm (bottom), for $t \in [0, 10]$ seconds.](image)

![Fig. 16: Results of the funnel-control algorithm for the UR5 Robotic Manipulator](image)
s.t. \( \frac{\partial h_j}{\partial t} + \left( \frac{\partial h_j}{\partial q_1} \right)^T q_2 + \left( \frac{\partial h_j}{\partial q_2} \right)^T [\hat{B}(q_1)]^{-1}(u - \hat{g}(q_1)) - \hat{C}(q_1, q_2)] + (h_j)^2 > 0, \forall j \in \{1, \ldots, 6\}, k \in \{1, 2\} \)

Similarly to the UAV scenario, in order to account for uncertainty, the controller is agnostic to the terms \( d_1(t) \) and \( d_2(t) \). Further, the terms \( \hat{B}(q_1), \hat{C}(q_1, q_2), \text{ and } \hat{g}(q_1) \) are estimations of \( B(q_1), C(q_1, q_2), \text{ and } g(q_1) \), 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 \( \eta^1(t), e_j^1(t) \), along with the funnel safety constants \( \tilde{\rho}_r \) and \( \tilde{\rho}_{tr} \). 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 \( \tilde{\rho}_r \) and \( \tilde{\rho}_{tr} \).

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

\[
\min \int_{t}^{t+T_p} (\bar{e}(\tau)^T Q \bar{e}(\tau) + \bar{u}(\tau)^T R \bar{u}(\tau)) d\tau \\
+ \bar{e}(t + T_p)^T P \bar{e}(t + T_p),
\]

s.t. \( \tilde{q}_1 = \hat{B}(q_1)^{-1}(u - \hat{C}(q_1, q_2) - \hat{g}(q_1)), \) \( \quad \tilde{q}_1(t) = q_1(t), q_2(t) = q_2(t) \)

\( \eta^1_k(t) < \tilde{\rho}_r, \quad \tau \in [t, t + T_p] \)

\( |e_j^1(\tau)| < \tilde{\rho}_{tr}, \quad \tau \in [t, t + T_p], j \in \{1, \ldots, 5\} \)

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

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

where we impose the funnel specification with respect to the trajectory obtained by KDF-RRT through the optimization constraints. We choose \( T_p = 20, Q = P = 100I_6, R = I_3, \)

\( \Omega = \{x \in \mathbb{R}^6 : ||x|| \leq 0.1\} \), and a sampling frequency of 200Hz. The results are given in the right part of Fig. 17, which depicts the evolution of the errors \( \eta^1_k(t), e_j^1(t) \), along with the funnel safety constants \( \tilde{\rho}_r \) and \( \tilde{\rho}_{tr} \). It is clear that the errors are not contained in the region defined by \( \tilde{\rho}_r \) and \( \tilde{\rho}_{tr} \), 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.,

\[
\min \int_{t}^{t+T_p} (\bar{e}_g(\tau)^T Q \bar{e}_g(\tau) + \bar{u}(\tau)^T R \bar{u}(\tau)) d\tau \\
+ \bar{e}_g(t + T_p)^T P \bar{e}_g(t + T_p),
\]

s.t. \( \tilde{q}_1 = \hat{B}(q_1)^{-1}(u - \hat{C}(q_1, q_2) - \hat{g}(q_1)), \) \( \quad \tilde{q}_1(t) = q_1(t), q_2(t) = q_2(t) \)

\( q_1(\tau) \in \mathcal{S}, \quad \tau \in [t, t + T_p] \)

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

\( \bar{e}_g(t + T_p) \in \Omega \)
where \( e_q = [q^T \dot{q}^T]^T = [(q_1 - q_{T_3})^T, q_2^T]^T \) and the set \( \mathcal{S} \) captures the safe part of the workspace, modeled as the exterior set of three spherical regions, i.e., \( \mathcal{S} = \bigcup_{k \in \{1, 2, 3\}} \{q_1 \in \mathbb{R}^6 : D_k = \|q_1 - OC_k\|^2 > OR^2 \} \), with \( OC_1 = [\frac{\pi}{4}, -\frac{\pi}{4}, \frac{\pi}{4}, 0, 0, 0]^T \), \( OC_2 = [-\frac{\pi}{2}, -\frac{\pi}{2}, 0, 0, 0, 0]^T \), \( OC_3 = [-\frac{3\pi}{4}, -\frac{\pi}{4}, 0, 0, 0, 0]^T \), and \( OR = 2 \) (rad). Furthermore, we choose \( T_p = 20 \), \( Q = P = 100I_6 \), \( R = I_3 \), \( \Omega = \{x \in \mathbb{R}^3 : \|x\| \leq 0.1 \} \). 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 \( D^2_k > OR^2 \), \( k \in \{1, 2, 3\} \). Although the distances satisfy the safety constraint \( D_k(t) > OR^2 \), 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 \( \|q_1 - OC_k\|^2 > OR^2 \) 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

![Fig. 19: The initial configuration of the HEBI robot in an obstacle-cluttered environment.](image)

![Fig. 20: The output sequence of points and the respective trajectories for the two paths of the hardware experiment.](image)

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 \( q_1 = [q_1^0, \ldots, q_6^0]^T \).

![Fig. 21: The evolution of the normalized errors \( \xi_j^1 \) (top) and \( \xi_j^2 \) (bottom), for \( j \in \{1, \ldots, 6\} \), of the hardware experiment.](image)

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 \( A_{free}(\bar{\rho}) \), we check 10 samples around each point of the resulting path for collision. We next fit smooth trajectories for the two paths \( q^1(t), q^2(t) \), 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 \( \rho^{1, 2} = [\rho_1^0, \ldots, \rho_6^0]^T = \bar{\rho} = [0.15, 0.1, 0.1, 0.2, 0.2, 0.2] \) rad, for the two paths \( i \in \{1, 2\} \). Moreover, we choose \( \rho_{2j} = 15 \) for all \( j \in \{1, \ldots, 6\} \), and the control gains as \( K^i = diag(1.25, 1.5, 1.2, 1.1) \), \( K_2 = diag(250, 200, 150, 50, 20, 10) \).

![Fig. 22: The evolution of the control inputs \( u(t) = [u_1(t), \ldots, u_6(t)]^T \) of the hardware experiment.](image)

The results of the experiment are depicted in Figs 21 and 22. In particular, Fig. 20 depicts the normalized signals \( \xi_j^i(t) \) =
\( [\xi_1, \ldots, \xi_n]^T \) and \( [\xi_2, \ldots, \xi_6]^T \) (top and bottom, respectively) for \( t \in [0, 21] \) seconds. It can be observed that for the entire motion, it holds that \( \xi_j(t) \in (-1, 1) \), for all \( j \in \{1, \ldots, 6\} \), which implies that \(-p_j^1 < e_j(t) = q_1(t) - q_2(t) < p_j^1\), \(-p_j^2 < e_2(t) = q_2(t) - \alpha_j(t) < p_j^2\), for all \( j \in \{1, \ldots, 6\} \) and \( t \in [0, 21] \) seconds, with \( \alpha_1 \) 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.

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 
\[
\xi_j = \max \left\{ \min \left\{ 1, \frac{e_j}{p_j^1} \right\}, -\frac{e_j}{p_j^1} \right\}, \quad \xi_j = \max \left\{ \min \left\{ 1, \frac{e_j}{p_j^2} \right\}, -\frac{e_j}{p_j^2} \right\}
\]
for all \( j \in \{1, \ldots, 6\} \). The evolution of the signals \( \xi_j(t) \), \( \xi_j(t) \) is 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 \( \xi_1^j \) and \( \xi_2^j \) are excessively increased with respect to their nominal values shown in Fig. 21, implying a large increase in the respective errors \( e_1^j \) and \( e_2^j \). 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 \( \xi^j \) for these two control schemes are depicted in Fig. 26. Note that the controllers fail to retain the normalized errors \( \xi_j^j(t) \) 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 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
where we further define \( \xi := [(\xi_1^T, \xi_2^T)^T, \sigma := [(\varepsilon_1^T, \varepsilon_2^T)^T, K := \text{blkdiag}\{K_1, K_2\}, R := \text{blkdiag}\{r_1, r_2\}, S := \text{blkdiag}\{I, \Sigma^2\}, \rho := \text{blkdiag}\{\bar{\sigma}, \bar{\sigma}\}, \xi^1 := \left[\xi_{1,1}, \xi_{1,2} \right]^T, \xi^2 := \left[\xi_{1,3}, \xi_{1,4} \right]^T, r^1 := \left[r_{1,1}, r_{1,2} \right]^T, r^2 := \left[r_{2,1}, r_{2,2} \right]^T, \tilde{\sigma} := \text{diag}\{\bar{\sigma}_{1,1}, \bar{\sigma}_{1,2} \}, \bar{\sigma} := \text{diag}\{\bar{\sigma}_{2,1}, \bar{\sigma}_{2,2} \}, S := \text{diag}\{\sin(\varepsilon_{1,1})\}, \bar{K} := \text{diag}\{\bar{K}_{1,1}, \bar{K}_{2,2} \} \). Since \( T_n \) is a quadratic form, it holds that \( T_n = -\frac{1}{2} \tilde{\sigma}^T K^T R^2 \bar{\sigma} + \sum_{j \in \mathcal{E}_r} (r_j^1)^2 \varepsilon_{j,1}^2 (1 + \cos(\varepsilon_{j,1}^2)), \)

Note that, for \( t \in I_t \), it holds that \( \xi^j_1 \in (-1, 1) \) and hence \( \eta_j(t) = 1 - \cos(\varepsilon_j^2(t)) < \rho_j(t) \leq \bar{\rho}_j < 2 \), for all \( \ell \in \mathcal{L}_r \) (see (4)). Therefore, it holds that \( 1 + \cos(\varepsilon_j^2) \geq 2 - \bar{\rho}_j =: \tilde{\rho}_j > 0 \), \( \ell \in \mathcal{L}_r \). By further defining \( \epsilon_j^2 := \min_{\ell \in \mathcal{L}_r} (\epsilon_j^2) \), we obtain

\[
T_n \leq -\tilde{\kappa} \|t\|e^2 \|t\| - \epsilon_j^2 \sum_{\ell \in \mathcal{L}_r} (r_j^1)^2 \varepsilon_{j,1}^2 \leq -m \|\kappa\|^2,
\]

where \( \kappa := [(r_j^1)^T, r_j^2 \sqrt{\varepsilon_j^2}, \ldots, r_j^2 \sqrt{\varepsilon_j^2}]^T \), and \( m := \min_{\ell \in \mathcal{L}_r} (\epsilon_j^2) \). Moreover, the fact that \( q_{\min}^j(t) \) is bounded and \( \eta_j(t) \in \Omega \) for \( t \in I_t \) implies that \( q_1^j(t) \) is bounded as \( \|q_1^j(t)\| \leq \sup_{t \geq 0} q_{\min}^j(t) + \sqrt{m} \max_{j \in \mathcal{L}_r} (\bar{\rho}_j) \) and \( \|e_2(t)\| \leq \sqrt{n} \max_{m \in \mathcal{L}_r} (\bar{\rho}_m) \), 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 \( q_1^j(t) \) and \( q_{\min}^j \) (by definition and assumption, respectively), as well as by using \( \xi_j^2 < \sqrt{\tilde{\rho}_j} \), for all \( \ell \in \mathcal{L}_r \), we conclude that there exists a positive finite constant \( B_1 \), independent of \( I_t \), satisfying \( B_1 \leq B_1 \|\kappa\| \), for all \( t \in I_t \). Therefore, \( V_1 \) becomes \( \tilde{V}_1 \leq -m \|\kappa\|^2 + B_1 \|\kappa\| \) for all \( t \in I_t \) and is negative when \( \|\kappa\| > B_1/m \), i.e.,

\[
\sqrt{\sum_{j \in \mathcal{E}_r} (r_j^1)^2 \varepsilon_{j,1}^2 + \sum_{\ell \in \mathcal{L}_r} (r_j^1)^2 \varepsilon_{j,2}^2} \geq \frac{B_1}{m},
\]

with \( L_t = \{1, \ldots, n_t\} \). From the definition of \( r_j^1 \) in (7), it holds that \( r_j^1(t) \geq 2 \), for all \( j \in \mathcal{L}_r \) and \( \forall t \in I_t \). Moreover, one can conclude by inspection that the function \( f(x) = \frac{1}{(1-x^2)} - \ln \left( \frac{1}{1-x} \right) \) is positive for positive \( x \). Therefore, since by definition \( \xi_j^2 \geq 0 \) it holds that \( (r_j^1)^2 \xi_j^2 \geq \xi_j^2 \), for all \( \ell \in \mathcal{L}_r \). Therefore, it holds that \( \sqrt{\sum_{j \in \mathcal{E}_r} (r_j^1)^2 \varepsilon_{j,1}^2 + \sum_{\ell \in \mathcal{L}_r} (r_j^1)^2 \varepsilon_{j,2}^2} \geq \sqrt{\sum_{j \in \mathcal{E}_r} (\xi_j^2)^2 + \sum_{\ell \in \mathcal{L}_r} \varepsilon_{j,2}^2} \) and a sufficient condition for \( \tilde{V}_1 \) to be negative is \( \sqrt{\sum_{j \in \mathcal{E}_r} (\xi_j^2)^2 + \sum_{\ell \in \mathcal{L}_r} \varepsilon_{j,2}^2} \geq \frac{B_1}{m} \), from which we conclude, by applying Theorem 4.18 of [43], that
there exists a positive constant $\varepsilon$ satisfying $|\varepsilon_j^i(t)| \leq \varepsilon$ and $\varepsilon_i^i(t) \leq \varepsilon_i$, which implies via (7) that

$$
|\varepsilon_j^i(t)| \leq \varepsilon_i^i \triangleq \frac{\exp(\varepsilon) - 1}{\exp(\varepsilon + 1)} < 1 \quad (21)
$$

and (23) imply that the system remains bounded in a compact space $\overline{q}$ is bounded, for all $t \in I_i$, as designed in (8), and system states $q_{i+1}$, $i \in \{2, \ldots, k - 1\}$, as well as the control law (12) remain bounded for all $t \in I_i$. What remains to be shown is $t_{\text{max}} = \infty$. Notice that (21) and (23) imply that the system remains bounded in a compact subset of $\Omega$, i.e., $(\overline{q}(t), t) \in \Omega \subset \Omega$, for all $t \in I_i$. Since $\overline{q}(t)$ has been proven bounded, the conditions of [60, Theorem 2.1.4] hold and we conclude hence that $\tau_{\text{max}} = \infty$. $\square$

ACKNOWLEDGMENT

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

REFERENCES


