Kinodynamic Motion Planning with Continuous-Time Q-Learning: An Online, Model-Free, and Safe Navigation Framework

George P. Kontoudis [1], *Student Member, IEEE,* Kyriakos G. Vamvoudakis [2], *Senior Member, IEEE*

***Abstract*—This paper presents an online kinodynamic motion planning algorithmic framework using RRT* and continuous-time Q-learning, which we term as RRT-Q*. We formulate a model-free Q-based advantage function and we utilize integral reinforcement learning to develop tuning laws for the online approximation of the optimal cost and the optimal policy of continuous-time linear systems. Moreover, we provide rigorous Lyapunov-based proofs for the stability of the equilibrium point, that results to asymptotic convergence properties. A terminal state evaluation procedure is introduced to facilitate the online implementation. We propose a static obstacle augmentation and a local re-planning framework, that is based on topological connectedness, to locally re-compute the robot's path and ensure collision-free navigation. We perform simulations and a qualitative comparison to evaluate the efficacy of the proposed methodology.**

***Index Terms*—Q-learning, online motion planning, actor/critic network, asymptotic optimality.**

I. Introduction

The field of motion planning has become a significant research area [1]–[4], towards achieving autonomous navigation of various dynamical systems in structured environments. Unambiguously, the overall complexity of the kinodynamic motion planning [5], and the computational requirements constitute a challenging problem, especially for a real-time implementation [6]–[8]. Optimality requires extensive offline computation and complete knowledge of the system dynamics [9]. Yet, the dynamics are often difficult to derive and when obtained they are unreliable and inaccurate, because disturbances and parameter uncertainties may affect the physics of the system [10]. Also, the mathematical derivation of the system dynamics varies for every robot. To deal with such problems, a solution is to employ simplified dynamical models, but still compute the optimal solution offline. Such an approach, may lead to unexpected, and inadequate performance. Undoubtedly, dynamical systems in practice propagate to the continuous-time domain. Moreover, feedback from sensors may output in high frequency, which dictates continuous-time implementation to capture full information from measurements. As a result, the vast majority of physical control tasks in robotics necessitate continuous-time action space [11], [12]. The objective of this work is to provide an online and safe kinodynamic motion planning algorithmic framework with completely unknown/uncertain linear dynamics, based on continuous-time Q-learning.

[1]G. P. Kontoudis is with the Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24060, USA, email: gpkont@vt.edu.

[2]K. G. Vamvoudakis is with the Daniel Guggenheim School of Aerospace Engineering, Georgia Tech, Atlanta, GA 30332, USA, email: kyriakos@gatech.edu.

This work was supported in part by an NSF CAREER under grant No. CPS-1851588, in part by NATO under grant No. SPS G5176, and in part by ONR Minerva under grant No. N00014-18-1-2160.

Related work

The problem of motion planning in high-dimensional systems has been tackled with incremental sampling algorithms, such as probabilistic road-maps (PRM) [1] and rapidly-exploring random trees (RRT) [2], which are probabilistically complete. The work of [4] proposed a asymptotically optimal approach, namely RRT*. The aforementioned approaches can be applied to systems with simple dynamics as they cannot deal with differential constraints.

On the other hand, the authors in [13] introduced the kinodynamic RRT, yet the control has been obtained either by taking many actions and selecting the best, or by randomly picking an action. LQR-trees have been developed for kinodynamic motion planning in [3]. This approach offers a feedback motion planning algorithm by employing the sum of squares decomposition. More specifically, they employ Lyapunov functions to generate appropriate funnels, that are based on a region of attraction. However, LQR-trees require full information of the system dynamics to solve the Riccati equation, that dictates extensive offline computation. In [14], the authors proposed the model-based LQR-RRT* that solves a free final state, infinite horizon optimal control problem with minimum energy cost and derives a heuristic extension of the RRT*. The work of [15] employed a finite horizon optimal control approach to measure and extend the RRT*. The focus was on systems with known nonlinear dynamics and the solution was obtained offline. The authors of [16] presented the kinodynamic RRT* that performs asymptotically optimal motion planning for systems with known linear dynamics. This technique formulates the finite horizon optimal control problem, with fixed final state, and free final time for a two point boundary value problem (TPBVP). In order to find a closed-form solution to the optimal control problem they used a minimum fuel-time performance, yet this approach yields an open-loop controller. Also, the derivation of the continuous reachability Gramian enforces significant offline computation. The authors in [17] proposed a near optimal kinodynamic motion planning technique, that is named NoD-RRT. The methodology utilizes neural network approximation and RRT. NoD-RRT achieved reduced computational complexity and enhanced performance for nonlinear systems, comparing to RRT and kinodynamic RRT*. Yet, their framework is model-based and requires offline computation.

A real-time kinodynamic motion planning was presented in [6]. The methodology consists of: an offline sampling-based motion planning; an offline machine learning technique to

pre-compute the optimal solutions of the TPBVP with direct optimization; and an online execution of the motion planner. The framework achieves fast response, but it requires complete information about the physics of the system. In [8], a kinodynamic planning without the need of trajectory optimization was presented and named RRT-CoLearn. The work of [18] employed motion primitives to develop a funnel library for LQR-trees. Such a methodology allows for online kinodynamic motion planning, after extensive offline computation.

Safe navigation considers the system's differential constraints to design collision-free paths, that compensate for the motion of an agent. In [19], the authors proposed a motion planning framework, which they term as FaSTrack. Their methodology tracks a trajectory by solving offline a pursuit-evasion game to find the largest relative distance. They employed the system dynamics to produce a safe area around the agent by using reachable sets, and then performed collision-free motion planning. The authors in [20] proposed an asymptotically optimal sampling-based motion planning algorithm that can be applied either in static or in dynamic environments, that they name as RRT$^{\text{X}}$. The latter consists of a local re-planning framework that refines the initial graph, but it requires the system dynamics for motion planning.

Optimal control [9] along with adaptive control [21] can be efficiently connected by employing the principles of reinforcement learning [22], [23], and actor/critic network structures [24]–[27]. More specifically, the critic evaluates the cost and the actor performs a policy improvement. In [28], a discrete-time Q-learning formulation was used to solve controlled Markovian systems. A hierarchical method for motion planning that combines sampling-based path planning and discrete reinforcement learning was proposed in [29]. However, the majority of real engineered systems require a continuous-time approach. Yet, for continuous-time systems the problem is nontrivial. In [30], a relation of Q-learning with nonlinear control was established, based on the observation that the Q-function is related to the Hamiltonian that appears in the minimum principle. The work of [31] presented a policy iteration approach to find online the adaptive optimal controller in a similar fashion with a Q-learning structure. The authors in [32] introduced a partially model-free algorithm based on policy iteration, that managed to solve the optimal control problem online. In [33], a Q-learning approach for solving the model-free infinite horizon optimal control for continuous-time systems was presented. The latter approach, composed of an advantage function with respect to the Hamiltonian and the optimal value function.

Contributions: The contribution of this paper is fourfold. First, we formulate the model-free finite horizon optimal control problem with free final state by employing a continuous-time Q-learning framework. We employ the global path, that is computed by the RRT*, to apply the kinodynamic motion planning algorithm RRT-Q*. The kinodynamic algorithmic framework RRT-Q* learns the optimal policy without any information of the system dynamics by using an actor/critic network, at every TPBVP. Since the kinodynamic algorithmic framework RRT-Q* does not require the solution of the differential Riccati equation, it is computationally feasible to be performed online. Next, we provide rigorous Lyapunov-based proofs for global asymptotic stability of the equilibrium point. Thus, our proposed framework can find online the optimal policy with asymptotic convergence properties and with completely unknown physics of the system. We also introduce a terminal state evaluation framework to further reduce the computational effort of the algorithmic framework and facilitate the online implementation. Finally, we propose a local static obstacle augmentation and a local re-planning framework, that is based on topological connectedness, to guarantee safe kinodynamic motion planning.

The remainder of this paper is organized as follows. Section II focuses on the problem formulation. In section III, we discuss the optimal control TPBVP. Section IV provides a model-free formulation based on Q-learning, and section V presents the structure and the algorithmic framework of the RRT-Q*. Section VI shows the efficacy of our approach through simulations and with a qualitative comparison. Section VII discusses the computational complexity, the implementation details, and the limitations of the methodology, while section VIII concludes the paper.

Notation: The notation here is standard. The $\mathbb{R}$ denotes the set of real numbers, $\mathbb{R}^+$ is the set of all positive real numbers, $\mathbb{R}^{n\times m}$ is the set of $n \times m$ real matrices, and $\mathbb{N}$ is the set of natural numbers. The $(\cdot)^\intercal$ and $(\cdot)^{-1}$ denote the transpose and inverse operator respectively. The I_n or I is the $n \times n$ identity matrix. The superscript $\star$ denotes the optimal solutions of a minimization problem. The $\underline{\lambda}(A)$ and the $\overline{\lambda}(A)$ are the minimum and the maximum eigenvalues of the matrix A respectively. The $\overline{K}$, the $|K|$, and the ∂K denote the closure, the cardinality, and the limit points of the set K respectively. We denote $\|\cdot\|_p$ as the p-norm of a vector. The vech(A), the vec(A), and the mat(A) are the half-vectorization, the vectorization, and the matrization of a matrix A respectively. The $U \otimes V$ denotes the Kronecker product of two vectors. The $\wedge$ denotes the "logical and" operation, and the $\oplus$ is the Minkowski sum of two sets.

II. Problem Formulation

Consider the following linear time-invariant continuous-time system of an agent performing motion forward in time,

$$\dot{x}(t) = Ax(t) + Bu(t),\ x(0) = x_0,\ t \geq 0,$$

where $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ is a measurable kinodynamic state vector, $u(t) \in \mathbb{R}^m$ is the control input, and $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ are the unknown plant and input matrix respectively.

For driving our system from an initial state x_0 to a final state $x(T) = x_{\text{r}}$, we define the difference between the state $x(t)$ and the state x_{r}, as our new state $\bar{x}(t) \coloneqq x(t) - x_{\text{r}}$. The final time is denoted by $T \in \mathbb{R}^+$. Similarly, we define our new control as, $\bar{u}(t) \coloneqq u(t) - u_{\text{r}}$, with $u_{\text{r}} = u(T)$. The new system becomes,

$$\begin{aligned}\dot{\bar{x}}(t) &= \dot{x}(t) - \dot{x}_{\text{r}} \\ &= A\bar{x}(t) + B\bar{u}(t),\ \bar{x}_0 = x_0 - x_{\text{r}},\ t \geq 0. \end{aligned} \tag{1}$$

Remark 1: The system (1) can be either time-varying or time-invariant. For simplicity in this paper, we have removed

the dependence on time, but our framework can be applicable to time-varying systems with minor modifications. □

We define an energy cost functional,

$$J(\bar{x};\bar{u};t_0,T) = \phi(T) + \frac{1}{2}\int_{t_0}^{T} (\bar{x}^\intercal M\bar{x} + \bar{u}^\intercal R\bar{u})\,\mathrm{d}\tau,\ \forall t_0, \quad (2)$$

where $\phi(T) \coloneqq \frac{1}{2}\bar{x}^\intercal(T)P(T)\bar{x}(T)$ is the terminal cost with $P(T) \coloneqq P_\mathrm{T} \in \mathbb{R}^{n\times n} \succ 0$ the final Riccati matrix, $M \in \mathbb{R}^{n\times n} \succeq 0$ and $R \in \mathbb{R}^{m\times m} \succ 0$ user defined matrices that penalize the states and the control input respectively.

Assumption 1: We assume that the unknown pairs (A,B) and ($\sqrt{M}$, A) are controllable and detectable respectively. □

We are interested in finding an optimal control $\bar{u}^\star$ such that, $J(\bar{x};\bar{u}^\star;t_0,T) \leq J(\bar{x};\bar{u};t_0,T),\ \forall \bar{x},\bar{u}$, which can be described by the minimization problem $J(\bar{x};\bar{u}^\star;t_0,T) = \min_u J(\bar{x};\bar{u};t_0,T)$ subject to (1). In other words, we want to obtain the optimal value function $V^\star$ that is defined by,

$$V^\star(\bar{x};t_0,T) \coloneqq \min_{\bar{u}} \Big(\phi(T) + \frac{1}{2}\int_{t_0}^{T} (\bar{x}^\intercal M\bar{x} + \bar{u}^\intercal R\bar{u})\,\mathrm{d}\tau\Big), \quad (3)$$

but without any information about the system dynamics.

Consider the closed obstacle space $\mathcal{X}_\mathrm{obs} \subset \mathcal{X}$. The free space is defined as the relative complement of the obstacle space in the state space, $\mathcal{X}_\mathrm{free} \coloneqq (\mathcal{X}_\mathrm{obs})^\complement = \mathcal{X}\backslash\mathcal{X}_\mathrm{obs}$, which is an open set [34]. The output of an algorithm is designed to efficiently search non-convex, high-dimensional spaces by randomly building a space-filling tree (e.g., RRT*) that will produce the global path $\pi(x_{0,i},x_{\mathrm{r},i}) \in \mathbb{R}^{2(k\times n)}$, for $i = 1,\ldots,k$ with $i \in \mathbb{N}$. The global path $\pi(x_{0,i},x_{\mathrm{r},i})$ will include the initial states $\mathcal{X}_0 = x_{0,i}$ for all i, with $\mathcal{X}_0 \in \mathbb{R}^{k\times n} \subset \mathcal{X}_\mathrm{free}$, and the final states $\mathcal{X}_\mathrm{G} = x_{\mathrm{r},i}$ for all i, with $\mathcal{X}_\mathrm{G} \in \mathbb{R}^{k\times n} \subset \mathcal{X}_\mathrm{free}$. The algorithm will also provide an initial graph $\mathcal{G} = (V,E)$, where V is the initial set of nodes, $V_\mathcal{G} = |V|$ its cardinality, and E the initial set of edges. With a slight abuse of notation, we will refer to the nodes $v \in V$ as states $x \in \mathcal{X}$. A direct relation of the global path π in the initial graph is given by the tree $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T}) \subseteq \mathcal{G}$. Furthermore, the augmented obstacle space $\mathcal{X}_\mathrm{obs}^\mathrm{aug} = f(t;D_\mathrm{rob})$ will change through time depending on the kinodynamic distance D_rob of the robot motion at every i-TPBVP. Similarly, the diminished open free space is defined as $\mathcal{X}_\mathrm{free}^\mathrm{dim} \coloneqq (\mathcal{X}_\mathrm{obs}^\mathrm{aug})^\complement = \mathcal{X}\backslash\mathcal{X}_\mathrm{obs}^\mathrm{aug}$.

Since we are solving a finite horizon optimal control problem with a free final state, and we are also setting a new state $\bar{x}(t)$, as the difference of the current state $x(t)$ with the desired state x_r, we can make the following approximation $x(T) \approx x_\mathrm{r}$. This means that the final state $x(T)$ may not obtain the exact desired state x_r value. Yet, the system may be fast enough to approximate the desired state, and stay there, until the end of the fixed finite horizon T. To address this problem we define the initial distance as the n-norm of the initial state x_0 and the desired state x_r,

$$D_0(\bar{x}_0) \coloneqq \|\bar{x}_0\|_n,\ \forall \bar{x}_0. \quad (4)$$

We shall then measure the relative distance at time $t \geq 0$ with the n-norm of the current state $x(t)$ and the desired state x_r,

$$D(\bar{x};t) \coloneqq \|\bar{x}(t)\|_n,\ \forall x,\ t. \quad (5)$$

Let us also define the distance error of (4) and (5) as,

$$e_\mathrm{d}(\bar{x}_0,\bar{x};t) \coloneqq |D_0(\bar{x}_0) - D(\bar{x},t)|. \quad (6)$$

Our work will formulate an online implementation framework of safe kinodynamic motion planning, given the global path and the initially randomly-sampled graph, with completely unknown dynamics as described in (1).

III. Finite Horizon Optimal Control

Define the Hamiltonian with respect to (1), and (3) as,

$$\mathcal{H}(\bar{x};\bar{u};\lambda) = \frac{1}{2}(\bar{x}^\intercal M\bar{x} + \bar{u}^\intercal R\bar{u}) + \lambda^\intercal(A\bar{x} + B\bar{u}),\ \forall \bar{x},\bar{u},\lambda.$$

In order to solve the finite horizon optimal control problem (3), by using the sweep method [35] and $\lambda = \frac{\partial V^\star}{\partial \bar{x}}$, we derive the Hamilton-Jacobi-Bellman (HJB) equation,

$$-\frac{\partial V^\star}{\partial t} = \frac{1}{2}(\bar{x}^\intercal M\bar{x} + \bar{u}^{\star\intercal} R\bar{u}^\star) + \frac{\partial V^{\star\,\intercal}}{\partial \bar{x}}(A\bar{x} + B\bar{u}^\star),\ \forall \bar{x}.$$

Since our system (1) is linear, we can write the value function in a quadratic in the state $\bar{x}$ form as,

$$V^\star(\bar{x};t) = \frac{1}{2}\bar{x}^\intercal P(t)\bar{x},\ \forall \bar{x}, t \geq t_0, \quad (7)$$

where $P(t) \in \mathbb{R}^{n\times n} \succ 0$ solves the Riccati equation,

$$-\dot{P}(t) = M + P(t)A + A^\intercal P(t) - P(t)BR^{-1}B^\intercal P(t). \quad (8)$$

Hence, the optimal control is,

$$\bar{u}^\star(\bar{x};t) = -R^{-1}B^\intercal P(t)\bar{x},\ \forall \bar{x},\ t. \quad (9)$$

Theorem 1: Suppose that there exists a $P(t)$ that satisfies the Riccati equation (8) with a final condition given by P_T, and the control obtained by,

$$\bar{u}(\bar{x};t) = -R^{-1}B^\intercal P(t)\bar{x}. \quad (10)$$

Then, the control input (10) minimizes the cost given in (3), and the origin is a globally uniformly asymptotically stable equilibrium point of the closed-loop system.

Proof: The closed-loop system becomes,

$$\dot{\bar{x}}^{(\mathrm{cl})} \coloneqq A\bar{x} - BR^{-1}B^\intercal P(t)\bar{x}. \quad (11)$$

We choose the $\mathcal{V}(\bar{x};t) = \frac{1}{2}\bar{x}^\intercal P(t)\bar{x}$, as a Lyapunov candidate with $\mathcal{V} : \mathbb{R}^n \times D \to \mathbb{R}$, and $D = \{P \in \mathbb{R}^{n\times n} \mid P \succ 0\}$. The orbital derivative yields for $t \geq 0$,

$$\begin{aligned}\dot{\mathcal{V}}(\bar{x};t) =& \frac{\partial \mathcal{V}}{\partial t} + \frac{\partial \mathcal{V}}{\partial \bar{x}}\dot{\bar{x}}\\ =& \frac{1}{2}\bar{x}^\intercal \dot{P}\bar{x} + \bar{x}^\intercal P\dot{\bar{x}}.\end{aligned}$$

By using (8) and (11) we obtain,

$$\begin{aligned}\dot{\mathcal{V}}(\bar{x};t) =& \frac{1}{2}\bar{x}^\intercal(-M - P(t)A - A^\intercal P(t) + P(t)BR^{-1}B^\intercal P(t))\bar{x}\\ &+ \bar{x}^\intercal P(t)(A\bar{x} - BR^{-1}B^\intercal P(t)\bar{x})\\ =& -\frac{1}{2}\bar{x}^\intercal(M + P(t)BR^{-1}B^\intercal P(t))\bar{x}\\ =& -\bar{x}^\intercal N(t)\bar{x}\\ =& -W_3(\bar{x};t)\\ \leq& -\underline{\lambda}(N(t))\|\bar{x}\|^2,\ \forall \bar{x},\ t. \end{aligned} \quad (12)$$

Note that, since $P(t)BR^{-1}B^\intercal P(t) \succ 0$, $M \succeq 0$, and thus $N(t) \succ 0$ is positive definite. We have proved that $\dot{\mathcal{V}}(\bar{x};t)$ is non-positive for all $\bar{x}$ and $t \geq t_0$.

Let us now consider, $W_1(\bar{x};t) = W_2(\bar{x};t) = \frac{1}{2}\bar{x}^\intercal P(t)\bar{x} > 0$. Obviously, we have $W_1(\bar{x};t) \leq \mathcal{V}(\bar{x};t) \leq W_2(\bar{x};t)$, and thus, we can conclude that the origin is uniformly stable. Since $\mathcal{V}(\bar{x};t)$ is lower-bounded, and non-increasing, the inequality (12) is also bounded, which implies that $\mathcal{V}(\bar{x};t)$ is uniformly continuous. According to Barbalat's Lemma [36], $\mathcal{V}(\bar{x};t) \to 0$ as $t \to \infty$. The function $W_3(\bar{x};t) = \underline{\lambda}(N(t))\|\bar{x}\|^2$ is positive definite, hence asymptotic stability of the equilibrium point (origin) holds from the Lyapunov stability theorem. Moreover, $W_1(\bar{x};t)$ is radially unbounded with respect to $\|\bar{x}\|$ and global properties also hold. Therefore, the equilibrium at the origin $\bar{x}_\mathrm{e} \coloneqq 0$ is globally uniformly asymptotically stable. ∎

IV. Model-Free TPBVP Formulation

Let us now define the following advantage function,

$$\begin{aligned}\mathcal{Q}(\bar{x};\bar{u};t) \coloneqq & V^\star(\bar{x};t) + \mathcal{H}(\bar{x};\bar{u};\frac{\partial V^\star}{\partial t},\frac{\partial V^\star}{\partial \bar{x}}) \\ =& V^\star(\bar{x};t) + \frac{1}{2}\bar{x}^\intercal M\bar{x} + \frac{1}{2}\bar{u}^\intercal R\bar{u} \\ & + \bar{x}^\intercal P(t)(A\bar{x}+B\bar{u}) + \frac{1}{2}\bar{x}^\intercal \dot{P}(t)\bar{x},\ \forall \bar{x},\bar{u},t, \end{aligned} \tag{13}$$

where $\mathcal{Q}(\bar{x};\bar{u};t) \in \mathbb{R}^+$ is an action-dependent value.

Next, we define $U \coloneqq [\bar{x}^\intercal\ \bar{u}^\intercal]^\intercal \in \mathbb{R}^{(n+m)}$ to express the Q-function (13) in a compact form as,

$$\mathcal{Q}(\bar{x};\bar{u};t) = \frac{1}{2}U^\intercal \begin{bmatrix} Q_{\mathrm{xx}}(t) & Q_{\mathrm{xu}}(t) \\ Q_{\mathrm{ux}}(t) & Q_{\mathrm{uu}} \end{bmatrix} U \coloneqq \frac{1}{2}U^\intercal \bar{\mathcal{Q}}(t) U, \tag{14}$$

where $Q_{\mathrm{xx}}(t) = \dot{P}(t) + P(t) + M + P(t)A + A^\intercal P(t) + P(t)B$, $Q_{\mathrm{xu}}(t) = Q_{\mathrm{ux}}^\intercal(t) = P(t)B$, and $Q_{\mathrm{uu}} = R$, with $\mathcal{Q} : \mathbb{R}^{n+m} \times \mathbb{R}^{(n+m)\times(n+m)} \to \mathbb{R}^+$. Note that, since the Riccati matrix is symmetric, $\bar{x}^\intercal P(t)A\bar{x} = \frac{1}{2}\bar{x}^\intercal(P(t)A + A^\intercal P(t))\bar{x}$, and $\bar{x}^\intercal P(t)B\bar{u} = \frac{1}{2}\bar{x}^\intercal(P(t)B + B^\intercal P(t))\bar{u}$ [9].

We can find a model-free formulation of $\bar{u}^\star$ given in (9) by using the stationarity condition $\frac{\partial \mathcal{Q}(\bar{x};\bar{u};t)}{\partial \bar{u}} = 0$, to obtain,

$$\bar{u}^\star(\bar{x};t) = \arg\min_{\bar{u}} \mathcal{Q}(\bar{x};\bar{u};t) = -Q_{\mathrm{uu}}^{-1}Q_{\mathrm{ux}}(t)\bar{x}. \tag{15}$$

Lemma 1: The value of the minimization $\mathcal{Q}^\star(\bar{x};\bar{u}^\star;t) \coloneqq \min_{\bar{u}} \mathcal{Q}(\bar{x};\bar{u};t)$ is the same with the optimal value $V^\star$ in (7) of the minimization problem (3), where $P(t) \succ 0$ is the Riccati matrix found from (8).

Proof: Substitute the optimal control $\bar{u}^\star$ in the Q-function (13) to get (8), which yields $0 = \dot{P}(t) + M + P(t)A + A^\intercal P(t) + P(t)B - P(t)BR^{-1}B^\intercal P(t)$. Therefore, $\mathcal{Q}^\star(\bar{x};\bar{u}^\star;t) = V^\star(\bar{x};t)$. ∎

A. Actor/Critic Network Structure

We design a critic approximator to approximate the Q-function in (14) as,

$$\mathcal{Q}^\star(\bar{x};\bar{u}^\star;t) = \frac{1}{2}U^\intercal\bar{\mathcal{Q}}(t) \coloneqq U\frac{1}{2}\mathrm{vech}(\bar{\mathcal{Q}}(t))^\intercal(U\otimes U),$$

where $\mathrm{vech}(\bar{\mathcal{Q}}(t)) \in \mathbb{R}^{\frac{(n+m)(n+m+1)}{2}}$. This structure exploits the symmetric properties of the $\bar{\mathcal{Q}}$ matrix to reduce the computational complexity.

Remark 2: We can employ the half-vectorization operation $\mathrm{vech}(\bar{\mathcal{Q}}(t))$, because $\bar{\mathcal{Q}}(t)$ is symmetric. □

Then, by using $\nu(t)^\intercal W_\mathrm{c} \coloneqq \frac{1}{2}\mathrm{vech}(\bar{\mathcal{Q}}(t))$ we have,

$$\mathcal{Q}^\star(\bar{x};\bar{u}^\star;t) = W_\mathrm{c}^\intercal\nu(t)(U\otimes U),$$

where $W_\mathrm{c} \in \mathbb{R}^{\frac{(n+m)(n+m+1)}{2}}$ are the critic weight estimates, and $\nu(t) \in \mathbb{R}^{\frac{(n+m)(n+m+1)}{2}\times\frac{(n+m)(n+m+1)}{2}}$ is a radial basis function of appropriate dimensions that depends explicitly on time.

Since the ideal weight estimates are unknown, we employ an adaptive estimation technique [21] by using current weights. Thus we have,

$$\hat{\mathcal{Q}}(\bar{x};\bar{u};t) = \hat{W}_\mathrm{c}^\intercal\nu(t)(U\otimes U), \tag{16}$$

where $\hat{W}_\mathrm{c}\nu(t) \coloneqq \frac{1}{2}\mathrm{vech}(\hat{\bar{\mathcal{Q}}}(t))$.

By using a similar way of thinking for the actor we will assign $\mu(t)^\intercal W_\mathrm{a} \coloneqq -Q_{\mathrm{uu}}^{-1}Q_{\mathrm{ux}}(t)$ to write,

$$\bar{u}^\star(\bar{x};t) = W_\mathrm{a}^\intercal\mu(t)\bar{x},$$

where $W_\mathrm{a} \in \mathbb{R}^{n\times m}$ are the actor weight estimates, $\mu(t) \in \mathbb{R}^{n\times n}$ is a radial basis function of appropriate dimensions that depends explicitly on time.

The actor by using current weight estimates is,

$$\hat{\bar{u}}(\bar{x};t) = \hat{W}_\mathrm{a}^\intercal\mu(t)\bar{x}. \tag{17}$$

Fact 1: The radial basis functions $\mu(t)$, $\nu(t)$ are bounded.

Remark 3: The critic and the actor approximators described in (16) and (17) respectively, do not include any approximation errors. Therefore, we use the whole space and not just a compact set. With this structure, the approximations will converge to the optimal policies, and hence the superscript $\star$, that denotes the ideal values of the adaptive weight estimation, render similarly with the optimal solutions.

We adopt the integral reinforcement learning approach from [27] that will express the Bellman equation as,

$$\begin{aligned} V^\star(\bar{x}(t);t) =& V^\star(\bar{x}(t-\Delta t);t-\Delta t) \\ & - \frac{1}{2}\int_{t-\Delta t}^{t}(\bar{x}^\intercal M\bar{x} + \hat{\bar{u}}^\intercal R\hat{\bar{u}})\,\mathrm{d}\tau, \end{aligned} \tag{18}$$

$$V^\star(\bar{x}(T);T) = \frac{1}{2}\bar{x}^\intercal(T)P(T)\bar{x}(T), \tag{19}$$

where $\Delta t \in \mathbb{R}^+$ is a small fixed value.

By following Lemma 1, where we have proved that $\mathcal{Q}^\star(\bar{x};\hat{\bar{u}}^\star;t) = V^\star(\bar{x};t)$, we can write (18) and (19) as,

$$\begin{aligned} \mathcal{Q}^\star(\bar{x}(t);\hat{\bar{u}}^\star(t);t) =& \mathcal{Q}^\star(\bar{x}(t-\Delta t);\hat{\bar{u}}^\star(t-\Delta t);t-\Delta t) \\ & - \frac{1}{2}\int_{t-\Delta t}^{t}(\bar{x}^\intercal M\bar{x} + \hat{\bar{u}}^\intercal R\hat{\bar{u}})\,\mathrm{d}\tau, \\ \mathcal{Q}^\star(\bar{x}(T);T) =& \frac{1}{2}\bar{x}^\intercal(T)P(T)\bar{x}(T). \end{aligned}$$

Next, we select the errors $e_{\mathrm{c}_1}, e_{\mathrm{c}_2} \in \mathbb{R}$, that we would like to drive to zero by appropriately tuning the critic weights of

(16). Define the first critic error e_{c_1} as,

$$\begin{aligned}e_{c_1} :=&\hat{\mathcal{Q}}(\bar{x}(t);\hat{u}(t);t) - \hat{\mathcal{Q}}(\bar{x}(t-\Delta t);\hat{u}(t-\Delta t);t-\Delta t)\\ &+\frac{1}{2}\int_{t-\Delta t}^{t}(\bar{x}^\intercal M\bar{x}+\hat{u}^\intercal R\hat{u})\,\mathrm{d}\tau\\ =&\hat{W}_c^\intercal\nu(t)\Big((\hat{U}(t)\otimes\hat{U}(t)) - (\hat{U}(t-\Delta t)\otimes\hat{U}(t-\Delta t)\Big)\\ &+\frac{1}{2}\int_{t-\Delta t}^{t}(\bar{x}^\intercal M\bar{x}+\hat{u}^\intercal R\hat{u})\,\mathrm{d}\tau,\end{aligned}$$

with $\hat{U} = [\bar{x}^\intercal\ \hat{u}^\intercal]^\intercal$ the augmented state that is comprised from the measurable full state vector, and the estimated control action. Next, we define the second critic error as,

$$e_{c_2} := \frac{1}{2}\bar{x}^\intercal(T)P(T)\bar{x}(T) - \hat{W}_c^\intercal\nu(T)(\hat{U}(T)\otimes\hat{U}(T)).$$

The actor approximator error $e_a\in\mathbb{R}^m$ is defined by,

$$e_a := \hat{W}_a^\intercal\mu(t)\bar{x} + \hat{Q}_{uu}^{-1}\hat{Q}_{ux}(t)\bar{x},$$

where $\hat{Q}_{uu}$, $\hat{Q}_{ux}$ will be obtained from the critic weight matrix estimation $\hat{W}_c$. By employing adaptive control techniques [21], we define the squared-norm of errors as,

$$K_1(\hat{W}_c,\hat{W}_c(T)) = \frac{1}{2}\|e_{c_1}\|^2+\frac{1}{2}\|e_{c_2}\|^2, \tag{20}$$

$$K_2(\hat{W}_a) = \frac{1}{2}\|e_a\|^2. \tag{21}$$

B. Learning Framework

The weights of the critic estimation matrix are obtained by applying a normalized gradient descent algorithm in (20),

$$\begin{aligned}\dot{\hat{W}}_c =&-\alpha_c\frac{\partial K_1}{\partial\hat{W}_c}\\ =&-\alpha_c\Big(\frac{1}{(1+\sigma^\intercal\sigma)^2}\sigma e_{c_1}+\frac{1}{(1+\sigma_f^\intercal\sigma_f)^2}\sigma_f e_{c_2}\Big),\end{aligned} \tag{22}$$

where $\sigma(t) := \nu(t)\Big(\hat{U}(t)\otimes\hat{U}(t)-\hat{U}(t-\Delta t)\otimes\hat{U}(t-\Delta t)\Big)$, $\sigma_f(T) = \nu(T)\Big(U(T)\otimes U(T)\Big)$, and $\alpha_c\in\mathbb{R}^+$ is a constant gain that specifies the convergence rate. The critic tuning (22) guarantees that as $e_{c_1}\to 0$ and $e_{c_2}\to 0$ then $\hat{W}_c\to W_c$ and $\hat{W}_c(T)\to W_c(T)$.

Similarly, the weights of the actor estimation matrix $\hat{W}_a$ by applying a gradient descent algorithm in (21) yield,

$$\dot{\hat{W}}_a = -\alpha_a\frac{\partial K_2}{\partial\hat{W}_a} = -\alpha_a\bar{x}e_a^\intercal, \tag{23}$$

where $\alpha_a\in\mathbb{R}^+$ is a constant gain that specifies the convergence rate. The actor estimation algorithm (23) guarantees that as $e_a\to 0$ then $\hat{W}_a\to W_a$.

For the theoretical analysis we introduce the weight estimation error for the critic $\tilde{W}_c := W_c - \hat{W}_c$ and for the actor $\tilde{W}_a := W_a - \hat{W}_a$, with $\tilde{W}_c\in\mathbb{R}^{\frac{(n+m)(n+m+1)}{2}}$, $\tilde{W}_a\in\mathbb{R}^{n\times m}$.

The estimation error dynamics of the critic yields,

$$\begin{aligned}\dot{\tilde{W}}_c =&\Big(\frac{\partial W_c}{\partial e_{c_1}}\frac{\mathrm{d}e_{c_1}}{\mathrm{d}t}+\frac{\partial W_c}{\partial e_{c_2}}\frac{\mathrm{d}e_{c_2}}{\mathrm{d}t}\Big)-\Big(\frac{\partial\hat{W}_c}{\partial e_{c_1}}\frac{\mathrm{d}e_{c_1}}{\mathrm{d}t}+\frac{\partial\hat{W}_c}{\partial e_{c_2}}\frac{\mathrm{d}e_{c_2}}{\mathrm{d}t}\Big)\\ =&\frac{\partial W_c}{\partial e_{c_1}}\frac{\mathrm{d}e_{c_1}}{\mathrm{d}t}-\frac{\partial\hat{W}_c}{\partial e_{c_1}}\frac{\mathrm{d}e_{c_1}}{\mathrm{d}t} = -\alpha_c\frac{1}{(1+\sigma^\intercal\sigma)^2}\sigma\sigma^\intercal\tilde{W}_c,\end{aligned}$$

and the estimation error dynamics of the actor becomes,

$$\dot{\tilde{W}}_a = -\alpha_a\bar{x}\bar{x}^\intercal\mu(t)^\intercal\tilde{W}_a - \alpha_a\bar{x}\bar{x}^\intercal\frac{\mu(t)\tilde{Q}_{xu}R^{-1}}{\|1+\mu(t)^\intercal\mu(t)\|^2}, \tag{24}$$

where $\tilde{Q}_{xu} := \mathrm{mat}(\tilde{W}_c[\frac{n(n+1)}{2}+1:\frac{n(n+1)}{2}+nm])$.

Lemma 2: For any given control input $u(t)\in\mathcal{U}$ the estimation error dynamics of the critic (24) have an exponentially stable equilibrium point at the origin as follows,

$$\|\tilde{W}_c\|\leq\|\tilde{W}_c(t_0)\|\kappa_1 e^{-\kappa_2(t-t_0)},$$

where $\kappa_1,\kappa_2\in\mathbb{R}^+$. In order to establish exponential stability, we require the signal $\Delta(t) := \frac{\sigma(t)}{1+\sigma(t)^\intercal\sigma(t))}$ to be persistently exciting (PE) at $[t,t+T_{PE}]$, where $T_{PE}\in\mathbb{R}^+$ the excitation period, if there exists a $\beta\in\mathbb{R}^+$ such that $\beta I\leq\int_t^{t+T_{PE}}\Delta(\tau)\Delta^\intercal(\tau)\mathrm{d}\tau$, where I is an identity matrix of appropriate dimensions.

Proof: The proof follows from [33]. ∎

Next, we provide the main stability Theorem for the proposed Q-learning framework.

Theorem 2: Consider the linear time-invariant continuous-time system (1), the critic, and the actor approximators given by (16), and (17) respectively. The weights of the critic, and the actor estimators are tuned by (22), and (23) respectively. The origin with state $\psi = [\bar{x}^\intercal\ \tilde{W}_c^\intercal\ \tilde{W}_a^\intercal]^\intercal$ is a globally uniformly asymptotically stable equilibrium point of the closed-loop system and for all initial conditions $\psi(0)$, given that the critic gain α_c is sufficiently larger than the actor gain α_a and the following inequality holds,

$$0<\alpha_a<\frac{\underline{\lambda}(M+Q_{xu}R^{-1}Q_{xu}^\intercal)-\overline{\lambda}(Q_{xu}Q_{xu}^\intercal)}{\delta\overline{\lambda}\Big(\frac{\mu(t)R^{-1}}{\|1+\mu(t)^\intercal\mu(t)\|^2}\Big)}, \tag{25}$$

with δ a constant of unity order.

Proof: Let us consider the following Lyapunov function,

$$\mathcal{L}(\psi;t) = V^\star(\bar{x};t)+\frac{1}{2}\|\tilde{W}_c\|^2+\frac{1}{2}\mathrm{tr}\{\tilde{W}_a^\intercal\tilde{W}_a\}>0,$$

for all $t\geq 0$, where $\psi = [\bar{x}^\intercal\ \tilde{W}_c^\intercal\ \tilde{W}_a^\intercal]^\intercal$ is the augmented state, and $\mathcal{L}:\mathbb{R}^n\times\mathbb{R}^{\frac{(n+m)(n+m+1)}{2}}\times\mathbb{R}^{n\times m}\to\mathbb{R}$. The orbital derivative for the closed-loop dynamics by using $\hat{u}$ yields,

$$\begin{aligned}\dot{\mathcal{L}}(\psi;t) =&\frac{\partial V^\star}{\partial t}+\frac{\partial V^{\star\intercal}}{\partial\bar{x}}\dot{\bar{x}}+\tilde{W}_c^\intercal\dot{\tilde{W}}_c+\mathrm{tr}\{\tilde{W}_a^\intercal\dot{\tilde{W}}_a\}\\ =&\frac{1}{2}\bar{x}^\intercal\dot{P}(t)\bar{x}+\bar{x}^\intercal P(t)(A\bar{x}+B\hat{u})\\ &-\alpha_c\tilde{W}_c^\intercal\frac{1}{(1+\sigma^\intercal\sigma)^2}\sigma\sigma^\intercal\tilde{W}_c\\ &-\alpha_a\mathrm{tr}\Big\{\tilde{W}_a^\intercal\bar{x}\bar{x}^\intercal\mu(t)^\intercal\tilde{W}_a-\tilde{W}_a^\intercal\bar{x}\bar{x}^\intercal\frac{\mu(t)\tilde{Q}_{xu}R^{-1}}{\|1+\mu(t)^\intercal\mu(t)\|^2}\Big\}.\end{aligned} \tag{26}$$

We can rewrite (26) as $\dot{\mathcal{L}} = T_1+T_2+T_3$, where,

$$T_1 = \frac{1}{2}\bar{x}^\intercal\dot{P}(t)\bar{x}+\bar{x}^\intercal P(t)(A\bar{x}+B\hat{u}), \tag{27}$$

$$T_2 = -\alpha_c\tilde{W}_c^\intercal\frac{1}{(1+\sigma^\intercal\sigma)^2}\sigma\sigma^\intercal\tilde{W}_c,$$

$$T_3 = -\alpha_{\mathrm{a}}\mathrm{tr}\Big\{\tilde{W}_{\mathrm{a}}^{\intercal}\bar{x}\bar{x}^{\intercal}\mu(t)^{\intercal}\tilde{W}_{\mathrm{a}} + \tilde{W}_{\mathrm{a}}^{\intercal}\bar{x}\bar{x}^{\intercal}\frac{\mu(t)\tilde{Q}_{\mathrm{xu}}R^{-1}}{\|1+\mu(t)^{\intercal}\mu(t)\|^2}\Big\}.$$

We use Lemma 2 and PE to obtain,

$$T_2 \leq -\frac{\alpha_{\mathrm{c}}}{4}\|\tilde{W}_{\mathrm{c}}\|^2, \tag{28}$$

and,

$$T_3 \leq -\frac{\alpha_{\mathrm{a}}}{2}\|\bar{x}^{\intercal}\sqrt{\mu(t)}^{\intercal}\tilde{W}_{\mathrm{a}}\|^2 + \frac{\alpha_{\mathrm{a}}\delta}{2}\overline{\lambda}\Big(\frac{\mu(t)R^{-1}}{\|1+\mu(t)^{\intercal}\mu(t)\|^2}\Big)\|\bar{x}\|^2. \tag{29}$$

The estimated control action $\hat{\bar{u}}$ can be written as,

$$\begin{aligned}\hat{\bar{u}} &= \hat{W}_{\mathrm{a}}^{\intercal}\mu(t)\bar{x}\\ &= -(Q_{\mathrm{xu}}Q_{\mathrm{uu}}^{-1} + \mu(t)^{\intercal}\tilde{W}_{\mathrm{a}})^{\intercal}\bar{x}\\ &= -Q_{\mathrm{uu}}^{-1}Q_{\mathrm{ux}}\bar{x} - \tilde{W}_{\mathrm{a}}^{\intercal}\mu(t)\bar{x}\\ &= \hat{\bar{u}}^{\star} - \tilde{W}_{\mathrm{a}}^{\intercal}\mu(t)\bar{x}.\end{aligned} \tag{30}$$

Using the Riccati equation (8), and the estimated control input (30), then the (27) takes the form of,

$$T_1 = -\frac{1}{2}\bar{x}^{\intercal}(M + Q_{\mathrm{xu}}R^{-1}Q_{\mathrm{xu}}^{\intercal})\bar{x},$$

which after using Young's inequality becomes,

$$T_1 \leq -\frac{1}{2}\Big(\underline{\lambda}(M + Q_{\mathrm{xu}}R^{-1}Q_{\mathrm{xu}}^{\intercal}) - \frac{1}{2}\overline{\lambda}(Q_{\mathrm{xu}}Q_{\mathrm{xu}}^{\intercal})\Big)\|\bar{x}\|^2. \tag{31}$$

Next, from (28), (29), and (31) we obtain,

$$\begin{aligned}\dot{\mathcal{L}}(\psi;t) \leq &-\Big(\frac{1}{2}\underline{\lambda}(M + Q_{\mathrm{xu}}R^{-1}Q_{\mathrm{xu}}^{\intercal}) - \frac{1}{2}\overline{\lambda}(Q_{\mathrm{xu}}Q_{\mathrm{xu}}^{\intercal})\\ &- \frac{\alpha_a\delta}{2}\overline{\lambda}\Big(\frac{\mu(t)R^{-1}}{\|1+\mu(t)^{\intercal}\mu(t)\|^2}\Big)\Big)\|\bar{x}\|^2\\ &- \frac{\alpha_{\mathrm{c}}}{4}\|\tilde{W}_{\mathrm{c}}\|^2 - \frac{\alpha_{\mathrm{a}}}{2}\|\bar{x}^{\intercal}\sqrt{\mu(t)}^{\intercal}\tilde{W}_{\mathrm{a}}\|^2.\end{aligned} \tag{32}$$

By taking into account the inequality in (25) then $\dot{\mathcal{L}}(\psi;t)$ is non-positive for all ψ and $t \geq t_0$. Consider now $W_1(\psi;t) = W_2(\psi;t) = V^{\star}(\bar{x};t) + \frac{1}{2}\|\tilde{W}_{\mathrm{c}}\|^2 + \frac{1}{2}\mathrm{tr}\{\tilde{W}_{\mathrm{a}}^{\intercal}\tilde{W}_{\mathrm{a}}\} > 0$, then we have $W_1(\psi;t) \leq \mathcal{L}(\psi;t) \leq W_2(\psi;t)$. In this way, we can conclude that the origin $\psi_{\mathrm{e}} \coloneqq 0$ is uniformly stable according to the Lyapunov stability theorem. Since $\mathcal{L}(\psi;t)$ is lower-bounded and non-increasing, inequality (32) is also bounded, which implies that $\mathcal{L}(\psi;t)$ is uniformly continuous. According to Barbalat's lemma, $\mathcal{L}(\psi;t) \to 0$ as $t \to \infty$. Next, the function $W_3(\psi;t) = (\frac{1}{2}\underline{\lambda}(M + Q_{\mathrm{xu}}R^{-1}Q_{\mathrm{xu}}^{\intercal}) - \frac{1}{2}\overline{\lambda}(Q_{\mathrm{xu}}Q_{\mathrm{xu}}^{\intercal}) - \frac{\alpha_{\mathrm{a}}\delta}{2}\overline{\lambda}(\frac{\mu(t)R^{-1}}{\|1+\mu(t)^{\intercal}\mu(t)\|^2}))\|\bar{x}\|^2 + \frac{\alpha_{\mathrm{c}}}{4}\|\tilde{W}_{\mathrm{c}}\|^2 - \frac{\alpha_{\mathrm{a}}}{2}\|x^{\intercal}\sqrt{\mu(t)}^{\intercal}\tilde{W}_{\mathrm{a}}\|^2$ needs to be positive definite. Thus, we obtain,

$$\begin{aligned}0 < &\frac{1}{2}\underline{\lambda}(M + Q_{\mathrm{xu}}R^{-1}Q_{\mathrm{xu}}^{\intercal}) - \frac{1}{2}\overline{\lambda}(Q_{\mathrm{xu}}Q_{\mathrm{xu}}^{\intercal})\\ &- \frac{\alpha_{\mathrm{a}}\delta}{2}\overline{\lambda}(\frac{\mu(t)R^{-1}}{\|1+\mu(t)^{\intercal}\mu(t)\|^2})\\ \alpha_{\mathrm{a}} < &\frac{\underline{\lambda}(M + Q_{\mathrm{xu}}R^{-1}Q_{\mathrm{xu}}^{\intercal}) - \overline{\lambda}(Q_{\mathrm{xu}}Q_{\mathrm{xu}}^{\intercal})}{\delta\overline{\lambda}\Big(\frac{\mu(t)R^{-1}}{\|1+\mu(t)^{\intercal}\mu(t)\|^2}\Big)},\end{aligned} \tag{33}$$

and,

$$\alpha_{\mathrm{a}} > 0. \tag{34}$$

The inequalities (33), (34) form (25) and thus the $W_3(\psi;t) > 0$ is positive definite, so asymptotic stability of the equilibrium point holds from the Lyapunov stability theorem. Moreover, $W_1(\psi;t)$ is radially unbounded with respect to $\|\bar{x}\|$, $\|\tilde{W}_{\mathrm{c}}\|$, $\|\tilde{W}_{\mathrm{a}}\|$, and global properties also hold. Therefore, the equilibrium at the origin $\psi_{\mathrm{e}} = 0$ is globally uniformly asymptotically stable [36]. ■

V. RRT-Q* Algorithmic Framework

In this section, we discuss the structure of the proposed model-free, online motion planning algorithm with Q-learning and optimal sampling-based path-planners. We also present the algorithmic framework of the proposed RRT-Q*.

A. RRT-Q* Structure

The structure of the proposed motion planning RRT-Q* is presented in Figure 1. The RRT-Q* consists of an offline global RRT* computation; an online actor/critic structure; an online terminal state evaluation; an online static obstacle augmentation; and an online local re-planning.

First, we compute offline the global path $\pi(x_{0,i}, x_{\mathrm{r},i})$ by using the RRT* algorithm. Then, we continue with the online model-free learning of the optimal policy. More specifically, the policy evaluation is assessed by the critic and the policy improvement is performed by the actor. The actor is an inner-loop feedback controller that drives the system with $\hat{\bar{u}}$ according to (17), where $\hat{W}_{\mathrm{a}}$ are the actor parameters that can be found online by (23). The critic's objective is to estimate the Q-function, which according to Lemma 1 is the value function that follows from the Bellman equations (18), (19). The critic approximates the $\hat{\mathcal{Q}}$ by using (16), where $\hat{W}_{\mathrm{c}}$ are the critic parameters that can be computed online by (22). The critic's parameters include intrinsic dynamics, which can be obtained by computing the time derivative that yields,

$$\begin{aligned}\dot{p} =&\bar{x}^{\intercal}(t)M\bar{x}(t) - \bar{x}^{\intercal}(t-\Delta t)M\bar{x}(t-\Delta t) + \hat{\bar{u}}^{\intercal}(t)R\hat{\bar{u}}(t)\\ &- \hat{\bar{u}}^{\intercal}(t-\Delta t)R\hat{\bar{u}}(t-\Delta t).\end{aligned} \tag{35}$$

A distance metric will be used to evaluate the terminal condition x_{r}. The initial distance $D_0(\bar{x}_0)$ is computed by (4). Next, the relative distance $D(\bar{x};t)$ is obtained online at every iteration Δt by (5). In case that the distance error (6) decreases below an admissible value of the initial distance $e_{\mathrm{d}} \leq \beta D_0$, $\beta \in \mathcal{B} = \{\beta \in \mathbb{R} \mid 0 \leq \beta \leq 1\}$, we continue to the next i-TPBVP of the RRT*, by assigning the current state value as the new initial state $x_{0,i+1} = x(t)$. It is to be noted that the i-TPBVP is specified by the i-set of the initial and the final states x_0, x_{r}, which were initially provided by the global planning with RRT*.

The RRT* algorithm is proved to compute the optimal path, which most of the times passes very close to the obstacles, that is potentially unsafe. Inherently, in kinodynamic motion planning we cannot track straight lines due to the kinodynamic constraints imposed by the physics of the system. Therefore, when the robot navigates close to the obstacle, and deviates from the given RRT* path, then a collision may occur with the obstacle. To address this problem, we propose a static augmentation of the obstacle space and a local re-planning strategy.

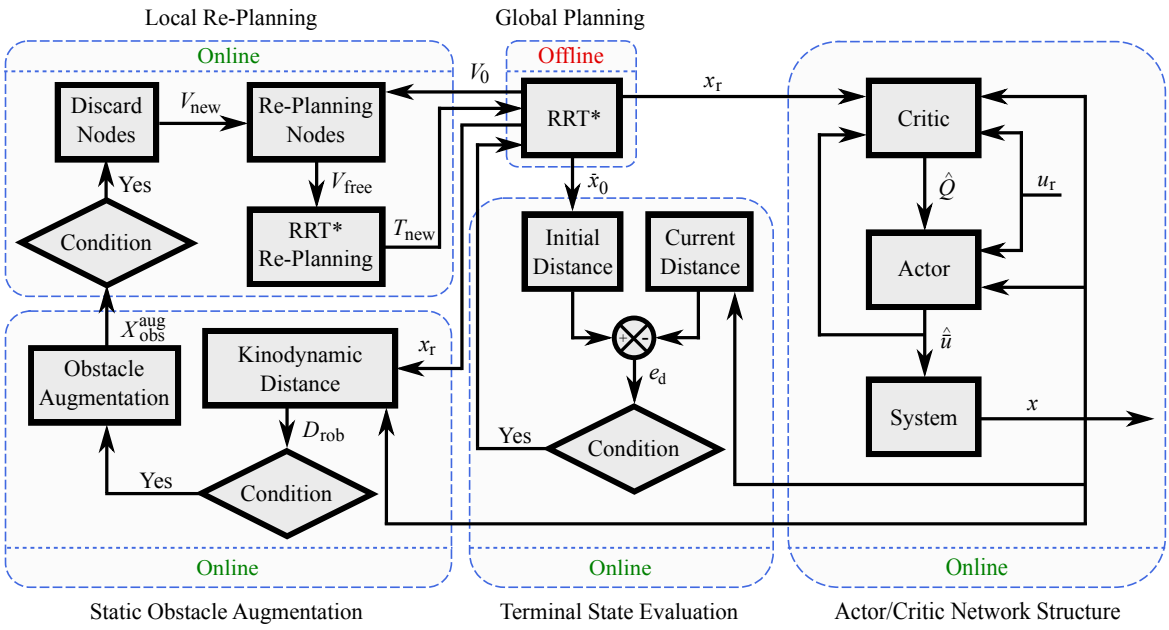

Fig. 1. The motion planning RRT-Q* structure. The RRT-Q* incorporates five stages, 1) the offline global RRT* computation, 2) the online actor/critic network structure, 3) the online terminal state evaluation, 4) the static obstacle augmentation, and 5) the online local RRT* re-planning.

For the static obstacle augmentation, we compute the maximum deviation of the robot from the straight line at every TPBVP, that we term as kinodynamic distance $D_{\text{rob}}(\bar{x}_0, \bar{x})$. The kinodynamic distance is computed by,

$$D_{\text{rob}}(\bar{x}_0, \bar{x}) = \frac{|\bar{x}_0 \times \bar{x}|}{D_0}. \tag{36}$$

Next, if the kinodynamic distance is greater than the previously measured deviations of motion $D_{\text{rob},i} > \max\{D_{\text{rob},1}, \ldots, D_{\text{rob},i-1}\}$, we compute the augmented obstacle space,

$$\mathcal{X}_{\text{obs}}^{\text{aug}} = \mathcal{X}_{\text{obs}} \oplus \mathcal{X}_{\text{rob}},$$

where $\mathcal{X}_{\text{rob}} \in \mathbb{R}^2$ is the kinodynamic distance space that is constructed as a rectangle with sides $\delta = 2D_{\text{rob}}$. That is a conservative approach, as we limit the navigation considering the maximum kinodynamic distance. Since we tackle the model-free problem, the model of the system is unknown, and hence we cannot perform offline computations. Therefore, the agent may deviate from the optimal path, yet the proposed method ensures collision-free navigation.

We continue on the local re-planning stage that will provide a safe path in the open diminished free space $\mathcal{X}_{\text{free}}^{\text{dim}} := (\mathcal{X}_{\text{obs}}^{\text{aug}})^{\complement} = \mathcal{X} \backslash \mathcal{X}_{\text{obs}}^{\text{aug}}$. We start by evaluating whether the global path $\pi(x_{0,i}, x_{\text{r},i})$ collides with the augmented obstacle space $\mathcal{X}_{\text{obs}}^{\text{aug}}$. Then, if a collision occurs, we prune the graph $\mathcal{G}(V, E)$ by discarding the nodes in the augmented obstacle space $V_{\text{aug}} = V \in \mathcal{X}_{\text{obs}}^{\text{aug}}$ from the initial set of nodes, $V_{\text{new}} = V \backslash V_{\text{aug}}$. Since the proposed algorithm operates online we cannot afford computationally to perform the RRT* even in the diminished free state-space $\mathcal{X}_{\text{free}}^{\text{dim}}$. Therefore, a significantly reduced free state-space needs to be specified.

The underlying principle to narrow down the local path planning problem exploits the precomputed nodes V and the initial global path π, towards defining a new local free state space $\mathcal{X}_{\text{free}}^{\text{loc}}$. First, we search for the two closest states of the initial global path π outside the area of collision with the augmented obstacle space $\mathcal{X}_{\text{obs}}^{\text{aug}}$. These two states will serve as the local start state $x_{\text{start}}^{\text{loc}}$ and the local goal state $x_{\text{goal}}^{\text{loc}}$, while the rest path will not be affected. If any states of the path π are located in the augmented obstacle space $\mathcal{X}_{\text{obs}}^{\text{aug}}$ we discard them from the updated set of nodes V_{new}. Next, we establish a circle with center point at $O_{\text{loc}} = (x_{\text{start}}^{\text{loc}} + x_{\text{goal}}^{\text{loc}})/2$ and radius $r_{\text{loc}} = \|x_{\text{start}}^{\text{loc}} - x_{\text{goal}}^{\text{loc}}\|$, that forms the local circular space $\mathcal{X}_{\text{circle}}^{\text{loc}} := \{x \in \mathcal{X} \mid \|x - O_{\text{loc}}\|^2 \leq r_{\text{loc}}^2\}$, as shown in Figure 2. Then, the local candidate path planning space is defined as the relative complement of the augmented obstacle space in the local circular space, $\mathcal{X}_{\text{cand}}^{\text{loc}} = \mathcal{X}_{\text{circle}}^{\text{loc}} \backslash \mathcal{X}_{\text{obs}}^{\text{aug}}$. To assess the local candidate space $\mathcal{X}_{\text{cand}}^{\text{loc}}$ we introduce the following definitions [34].

Definition 1: If A is a subset of a metric space X, and if ∂A denotes the set of all its limit points, then $\overline{A}$ said to be closure of A if $\overline{A} = A \cup \partial A$. □

Definition 2: Two subsets A and B of a metric space X are separated if both $A \cap \overline{B} = \emptyset$ and $\overline{A} \cap B = \emptyset$ hold. □

Definition 3: A set A is connected if it is not the union of two separated sets. □

We analyze whether the local candidate path planning space $\mathcal{X}_{\text{cand}}^{\text{loc}}$ has separated subspaces, as this notion can be handled easier than the connected space. According to the structure of the environment (i.e., obstacle space and free state space) the local candidate path planning space $\mathcal{X}_{\text{cand}}^{\text{loc}}$ may result to be separated, with one space that contain the local goal state $x_{\text{goal}}^{\text{loc}}$ and another space with the local start state $x_{\text{start}}^{\text{loc}}$, as depicted in Figure 3.

Lemma 3: For a given set of states in the diminished free space $\mathcal{X}_{\text{free}}^{\text{dim}}$, the local start state $x_{\text{start}}^{\text{loc}}$, and the local goal state $x_{\text{goal}}^{\text{loc}}$, if there exists a sufficient, connected, and closed local free space $\mathcal{X}_{\text{free}}^{\text{loc}}$ that forms a ring, based on the fixed

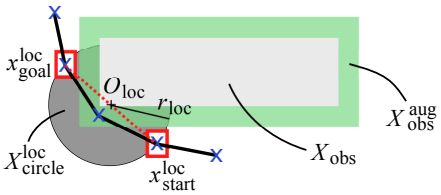

Fig. 2. The construction of the local circular space $\mathcal{X}^{\text{loc}}_{\text{circle}}$. We employ the two closest states $x^{\text{loc}}_{\text{start}}$, $x^{\text{loc}}_{\text{goal}}$ of the initial global path π that do not collide with the augmented obstacle space $\mathcal{X}^{\text{aug}}_{\text{obs},i}$, to construct a circle at center O_{loc} and with radius r_{loc}.

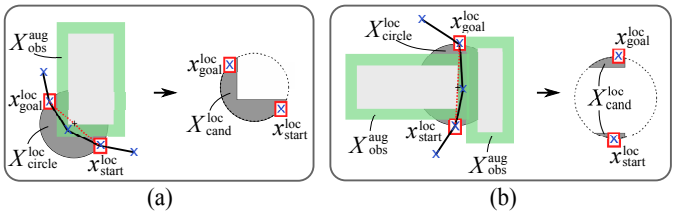

Fig. 3. The proposed procedure to extract the local candidate free space $\mathcal{X}^{\text{loc}}_{\text{cand}}$. (a) A connected local candidate free space, that contains the local start state $x^{\text{loc}}_{\text{start}}$ and the local goal state $x^{\text{loc}}_{\text{goal}}$. (b) Separated local candidate free subspaces.

incremental distance ϵ of the RRT*, then we can obtain a collision-free path with the local re-planning framework.

Proof: For the justification of the connected space, we assume that there exists two separated subspaces, one that contains the local start state $x^{\text{loc}}_{\text{start}}$ and another with the local goal state $x^{\text{loc}}_{\text{goal}}$. According to Definition 1 the closure of a set is related with its boundaries. Since we defined the local candidate space in a circular geometry, we can exploit the circular boundaries with center at O_{loc} and radius r_{loc}. To algorithmically evaluate the boundaries let us mesh the circle with finite many points $l \in \mathbb{N}$ and employ the equations that govern the semicircle $y_{\text{test},l} = \pm\sqrt{r^2_{\text{loc}} - (x_{\text{test},l} - x_{\text{c}})^2} + y_{\text{c}}$, where $O_{\text{loc}} = (x_{\text{c}}, y_{\text{c}})$. If any point lies on the circumference of the semicircle $(x_{\text{test},l}, y_{\text{test},l}) \in \partial\mathcal{X}^{\text{loc}}_{\text{cand}}$, and it is not in the local candidate free space $(x_{\text{test},l}, y_{\text{test},l}) \notin \mathcal{X}^{\text{loc}}_{\text{cand}}$ for both directions, then according to the Definitions 2, and 3 the local candidate free space $\mathcal{X}^{\text{loc}}_{\text{cand}}$ is indeed separated. Otherwise, by contradiction the local candidate free space is connected.

However, to guarantee feasible local re-planning we need also to consider the nature of the RRT*. More specifically, the RRT* employs a fixed incremental distance ϵ which cannot follow circular paths, but only straight lines. As a result, we need to ensure that not only the local candidate free space $\mathcal{X}^{\text{loc}}_{\text{cand}}$ is connected, but also that there exists a sufficiently large ring space to accommodate the fixed incremental distance ϵ, as presented in Figure 4. Let as consider the fixed incremental distance as the length of a chord $c = \epsilon$. Then, the sagitta (height of an arc) is $h = r_{\text{loc}} - \sqrt{{r_{\text{loc}}}^2 - \frac{c^2}{4}}$, and the radius of the internal circle, $r^{\text{int}}_{\text{loc}} = r_{\text{loc}} - h$. Similarly, we evaluate the separation of the local internal space $\mathcal{X}^{\text{loc}}_{\text{int}} := \{x \in \mathcal{X} \mid \|x - O_{\text{loc}}\|^2 \leq {r^{\text{int}}_{\text{loc}}}^2\}$. If both the local candidate space and the local internal space are connected then we assign, $\mathcal{X}^{\text{loc}}_{\text{free}} = \mathcal{X}^{\text{loc}}_{\text{cand}}$. ■

Remark 4: The determination of a significantly small local free space $\mathcal{X}^{\text{loc}}_{\text{free}}$ that is connected, and guarantees the existence of a local path π^{loc} from the local initial state $x^{\text{loc}}_{\text{start}}$ to the local goal state $x^{\text{loc}}_{\text{goal}}$ is a challenging problem. This difficulty lies in the unknown kinodynamic distances D_{rob} due to the model-free approach that augments the obstacle space, the unknown number of states that will be discarded from the initial global path π, and the requirements for a reduced computational effort that will allow the online implementation of the algorithm. In this paper, we assess the local candidate path planning space $\mathcal{X}^{\text{loc}}_{\text{cand}}$, and we discuss the case of a connected space. □

Since we obtained a relatively small local free space $\mathcal{X}^{\text{loc}}_{\text{free}}$ that is guaranteed to contain the local start state $x^{\text{loc}}_{\text{start}}$, the local goal state $x^{\text{loc}}_{\text{goal}}$, and sufficient space for the implementation of the path planning with incremental distance ϵ, we can move on the next step, that is the local re-planning with RRT*. We locate the free re-planning nodes from the updated set of nodes V_{new} that lie in the local free space $\mathcal{X}^{\text{loc}}_{\text{free}}$, which we term as $V_{\text{free}} = V_{\text{new}} \in \mathcal{X}^{\text{loc}}_{\text{free}}$. The coordinates of the free re-planning nodes V_{free} will further reduce the computational effort, as no random sampling is required in the local free space $\mathcal{X}^{\text{loc}}_{\text{free}}$. The output is a local path π^{loc} that connects the local start state $x^{\text{loc}}_{\text{start}}$ with the local goal state $x^{\text{loc}}_{\text{goal}}$, which along with the previously computed global path π produces the new tree $\mathcal{T}_{\text{new}}$.

B. RRT-Q* Algorithm

The algorithmic framework of the RRT-Q* consists of five phases, the offline computation of the global path planning; the online motion planning with Q-learning; a terminal state evaluation framework; the computation of the augmented obstacle space; and the local re-planning procedure.

The RRT-Q* is presented in Algorithm 1 and its subroutines in Algorithms 6 and 7. The "standard" routines of the RRT* are presented in Algorithms 2 to 5. The global graph $\mathcal{G}(V, E)$ is obtained offline by the `RRT*` as shown in Algorithms 2 to 4. Next, we continue with the online implementation. The function `NoCollision` monitors if there exists a collision in the entire augmented obstacle space $\mathcal{X}^{\text{aug}}_{\text{obs}}$ with the global path π through the whole procedure and returns a binary value. The function `InitialDistance` calculates the distance of the initial and the final state according to (4). Then, follows the online approximation of the optimal policy with full state feedback (lines 8-12). The function `Critic` estimates the critic parameters from (22). This includes internal dynamics as given in (35). The function `EstimateQ` estimates the parameters of the $\hat{\mathcal{Q}}$ from (16). The `Actor` calculates the actor parameters from (23), that lead the function `Control` to produce the control action $\hat{u}$ from (17). Next, we perform the terminal state evaluation (lines 13-18). The function `KinodynamicDistance` returns the deviation of the agent from the straight line that connects the the initial and final states, by employing (36). The distance error is calculated by the function `DistanceError`, which allow the terminal state evaluation to proceed to the next i-TPBVP. The primitive `Augment` inflates the obstacle space by comparing the maximum distance of the previously obtained deviations

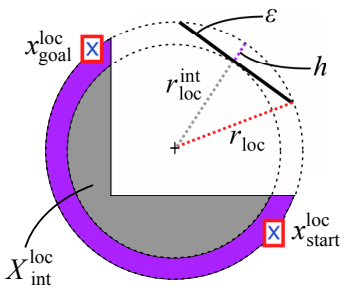

Fig. 4. The local internal space $\mathcal{X}^{\text{loc}}_{\text{int}}$ is illustrated in gray. The internal circle is constructed based on the sagitta h, and the chord c. The chord is related with the incremental distance of the RRT*, that is $c = \epsilon$. The ring space is shown in purple and ensures that the local path planning algorithm has a feasible solution.

$\max\{D_{\text{rob},1}, \ldots, D_{\text{rob},i-1}\}$, with the current kinodynamic distance $D_{\text{rob},i}$.

When a collision of the path π occurs with the augmented obstacle space, then the algorithm continues to the next phase of the online local re-planning. A critical aspect for the feasibility of the online implementation, is to perform the re-planning procedure sufficiently fast. To this end, we narrow down the local free space with `LocalNodes`, that provides the free nodes V_{free} according to the fixed incremental ϵ of the RRT* for feasible local re-planning. Then, the `RRT*` provides the local graph $\mathcal{G}_{\text{new}}$, with a given set of nodes V_{free} that reduce the computational effort even further, as no random sampling is required. Lastly, the primitive `Connect` employs the global graph $\mathcal{G}$ and the locally graph $\mathcal{G}_{\text{new}}$ to find a safe tree $\mathcal{T}_{\text{new}}$ with respect to the kinodynamic constraints.

The `RRT*` either runs the "standard" RRT* routine for the global planning or refers to `ReplanningRRT*` for the local re-planning as presented in Algorithm 5. Algorithm 2 is similar to RRT*, yet with the addition of a condition on line 1 and lines 15, and 16. Algorithm 5 is also similar to RRT*, but with the variation on line 2 given a precomputed set of nodes from the global planning. The new features in Algorithms 2 to 5 are emphasized with a green font. The `Sample` returns independent uniformly distributed random samples in the state space x_{rand}. The function `Nearest` provides the closest state x_{nearest} in the set of nodes V. The function `Steer` provides a new state x_{new} that is closer to the starting state x_{nearest}. The function `Near` produces a set of states X_{near} according to $X_{\text{near}} = \{x_{\text{near}} \in V \mid \|x_{\text{new}} - x_{\text{near}}\| \leq \gamma_{\text{RRT}^\star}(\frac{\log N}{N})^{1/n}\}$, where $\gamma_{\text{RRT}^\star} \in \mathbb{R}^+$, N is the number of samples, and n is the state space dimension. The function `Line` connects two states with a straight line. The `Parent` provides a parent state x_{min} and a cost-to-go c_{min} to the random state x_{rand} through the set X_{near} as presented in Algorithm 3. The algorithm `Rewire` finds the path with the minimum cost by creating and/or discarding edges from the graph, as described in Algorithm 4. In Algorithm 3 the framework for finding the parent with the least cost is described. The function `Cost` computes the cost-to-go to a node v, and the $c(\cdot)$ finds the cost of two nodes that form a straight line. The rewiring process is described in the Algorithm 4. In Algorithm 6, the function `Circle` establishes a local candidate space $\mathcal{X}^{\text{cand}}_{\text{loc}}$ based on a circle with radius

Algorithm 1 RRT_Q*$(T, \Delta t, M, R, P(T), \alpha_{\text{c}}, \alpha_{\text{a}}, \beta, \mathcal{X}_{\text{obs}}, \mathcal{G}(V, E))$

$V_{\text{free}} \leftarrow \emptyset$; $\mathcal{X}^{\text{aug}}_{\text{obs}} \leftarrow \mathcal{X}_{\text{obs}}$;
$\mathcal{G}, \pi(x_0, x_{\text{r}}) \leftarrow$ `RRT*`$(\mathcal{G}, N, V_{\text{free}})$;
$D^{\text{kin}}_{\text{rob}} \leftarrow \emptyset$;
while `NoCollision`(π) **do**
  for $i = 1$ to k **do**
    $D_0 \leftarrow$`InitialDistance`(x_0);
    for $t \in T$ **do**
      $\hat{W}_{\text{c}} \leftarrow$ `Critic`$(\bar{x}, \bar{u}, \alpha_{\text{c}}, M, R, \Delta t, P(T))$;
      $\hat{\mathcal{Q}} \leftarrow$ `EstimateQ`$(\bar{x}, \bar{u}, \hat{W}_{\text{c}})$;
      $\hat{W}_{\text{a}} \leftarrow$ `Actor`$(\bar{x}, \bar{u}, \hat{\mathcal{Q}}, \alpha_{\text{a}})$;
      $\hat{u} \leftarrow$ `Control`$(\bar{x}, \bar{u}, \hat{W}_{\text{a}})$;
      Return $\hat{u}$;
      $D_{\text{rob}} \leftarrow$ `KinodynamicDistance`$(\bar{x}, D_0)$;
      $e_d \leftarrow$ `DistanceError`$(\bar{x}, D_0)$;
      if $e_d \leq \beta D_0$ **then**
        $x_{0,i+1} \leftarrow x(t)$;
        break;
      end if
    end for
    if $D_{\text{rob}} > D^{\text{kin}}_{\text{rob}}$ **then**
      $\mathcal{X}^{\text{aug}}_{\text{obs}} \leftarrow$ `Augment`$(\mathcal{X}_{\text{obs}})$; $D^{\text{kin}}_{\text{rob}} \leftarrow D_{\text{rob}}$;
    end if
  end for
end while
$V_{\text{free}} \leftarrow$ `LocalNodes`$(\pi, \mathcal{X}^{\text{aug}}_{\text{obs}}, \epsilon)$;
$\mathcal{G}_{\text{new}} \leftarrow$ `RRT*`(V_{free});
$\mathcal{T}_{\text{new}} \leftarrow$ `Connect`$(\mathcal{G}, \mathcal{G}_{\text{new}})$;
Return $\mathcal{T}_{\text{new}}$;

Algorithm 2 RRT* $(\mathcal{G}, N, \mathcal{G}_{\text{free}}, V_{\text{free}})$

1: **if** $V_{\text{free}} \leftarrow \emptyset$ **then**
2: $V \leftarrow x_{\text{start}}$; $E \leftarrow \emptyset$;
3: **for** $n = 1$ to N **do**
4: $x_{\text{rand}} \leftarrow$ `Sample`$(\mathcal{X}_{\text{free}}, N)$;
5: $x_{\text{nearest}} \leftarrow$ `Nearest`(V, x_{rand}); $x_{\text{new}} \leftarrow$ `Steer`$(x_{\text{nearest}}, x_{\text{rand}})$;
6: **if** `NoCollision`$(x_{\text{nearest}}, x_{\text{new}})$ **then**
7: $X_{\text{near}} \leftarrow$ `Near`(V, x_{new}); $\eta \leftarrow$ `Line`$(x_{\text{nearest}}, x_{\text{new}})$;
8: $(x_{\text{min}}, c_{\text{min}}) \leftarrow$ `Parent`$(x_{\text{rand}}, X_{\text{near}}, x_{\text{new}}, \eta)$;
9: $V \leftarrow V \cup \{x_{\text{new}}\}$;
10: $E \leftarrow E \cup \{(x_{\text{min}}, x_{\text{new}})\}$; $\mathcal{G} \leftarrow$ `Rewire`$(\mathcal{G}, X_{\text{near}}, x_{\text{new}})$;
11: **end if**
12: **end for**
13: Return $\mathcal{G}$;
14: **else**
15: $\mathcal{G}_{\text{new}} \leftarrow$ `ReplanningRRT*`$(\mathcal{G}_{\text{free}}(V_{\text{free}}, E_{\text{free}}))$;
16: Return $\mathcal{G}_{\text{new}}$;
17: **end if**

r_{loc} and at center O_{loc}. The function `SemiCircle` attempts to evaluate the connectedness of the $\mathcal{X}^{\text{cand}}_{\text{loc}}$ with respect to the fixed incremental distance ϵ of the RRT*. The `Edge` returns the closest states to the area of collision that we set as the local start state $x^{\text{loc}}_{\text{start}}$ and the local goal state $x^{\text{loc}}_{\text{goal}}$.

VI. Simulations

In this section, we present simulations to demonstrate the efficacy of the proposed online motion planning framework. We study an aircraft model with the proposed online, model-free approximation of the optimal policy in a single TPBVP. Next, we perform motion planning simulations in various obstacle environments using the RRT-Q* algorithm, while also altering the dynamics at every TPBVP. Furthermore, we validate the efficiency of the terminal state evaluation by providing a comparison study. Lastly, we perform a qualitative comparison of the RRT-Q* with other motion planning algorithms.

Algorithm 3 Parent ($x_{\text{rand}}, X_{\text{near}}, x_{\text{new}}, \eta$)

$x_{\min} \leftarrow x_{\text{nearest}}$; $c_{\min} \leftarrow \texttt{Cost}(x_{\text{nearest}})+c(\eta)$;
for $x_{\text{near}} \in X_{\text{near}}$ **do**
  $\eta \leftarrow \texttt{Line}(x_{\text{near}}, x_{\text{new}})$;
  if $\texttt{NoCollision}(\eta) \wedge \texttt{Cost}(x_{\text{near}})+c(\eta) < c_{\min}$ **then**
    $c_{\min} \leftarrow \texttt{Cost}(x_{\text{near}})+c(\eta)$; $x_{\min} \leftarrow x_{\text{near}}$;
  end if
end for
Return $(x_{\min}, c_{\min})$

Algorithm 4 Rewire ($\mathcal{G}, X_{\text{near}}, x_{\text{new}}$)

for $x_{\text{near}} \in X_{\text{near}}$ **do**
  $\eta \leftarrow \texttt{Line}(x_{\text{near}}, x_{\text{new}})$;
  if $\texttt{NoCollision}(\eta) \wedge \texttt{Cost}(x_{\text{new}})+c(\eta) < \texttt{Cost}(x_{\text{near}})$ **then**
    $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
    $E_0 \leftarrow (E_0 \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$;
  end if
end for
Return $\mathcal{G}$;

Algorithm 5 ReplanningRRT* ($\mathcal{G}_{\text{free}}(V_{\text{free}}, E_{\text{free}})$)

$E_{\text{free}} \leftarrow \emptyset$;
for each $v \in V_{\text{free}}$ **do**
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(V_{\text{free}}, v)$; $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, v)$;
  if $\texttt{NoCollision}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow \texttt{Near}(V_{\text{free}}, x_{\text{new}})$; $\eta \leftarrow \texttt{Line}(x_{\text{nearest}}, x_{\text{new}})$;
    $(x_{\min}, c_{\min}) \leftarrow \texttt{Parent}(x_{\text{rand}}, X_{\text{near}}, x_{\text{new}}, \eta)$;
    $V_{\text{free}} \leftarrow V_{\text{free}} \cup \{x_{\text{new}}\}$;
    $E_{\text{free}} \leftarrow E_{\text{free}} \cup \{(x_{\min}, x_{\text{new}})\}$
    $\mathcal{G}_{\text{new}} \leftarrow \texttt{Rewire}(\mathcal{G}_{\text{free}}, X_{\text{near}}, x_{\text{new}})$;
  end if
end for
Return $\mathcal{G}_{\text{new}}$;

Algorithm 6 LocalNodes ($\pi, \mathcal{X}_{\text{obs}}^{\text{aug}}, \epsilon$)

$V_{\text{new}} \leftarrow V - \{v \in \mathcal{X}_{\text{obs}}^{\text{aug}}\}$;
$(x_{\text{start}}^{\text{loc}}, x_{\text{goal}}^{\text{loc}}) \leftarrow \texttt{Edge}(\mathcal{X}_{\text{obs}}^{\text{aug}}, \pi)$;
$(O_{\text{loc}}, r_{\text{loc}}, \mathcal{X}_{\text{loc}}^{\text{cand}}) \leftarrow \texttt{Circle}(x_{\text{start}}^{\text{loc}}, x_{\text{goal}}^{\text{loc}})$;
for $j = 1$ to l **do**
  $x_{\text{test}} \leftarrow \texttt{SemiCircle}(O_{\text{loc}}, r_{\text{loc}}, \epsilon)$
  if $x_{\text{test}} \in \mathcal{X}_{\text{obs}}^{\text{aug}}$ **then**
    $V_{\text{free}} \leftarrow \emptyset$;
  else
    $V_{\text{free}} \leftarrow V_{\text{new}}$;
  end if
end for
Return V_{free};

Algorithm 7 Edge ($\pi, \mathcal{X}_{\text{obs}}^{\text{aug}}$)

for $i = 1$ to k **do**
  if $x_0 \in \mathcal{X}_{\text{obs}}^{\text{aug}}$ **then**
    $x_{\text{start}}^{\text{loc}} \leftarrow x_0^{\text{free}}$; break;
  end if
  $x_0 \leftarrow x_0^{\text{free}}$;
end for
for $i = 1$ to k **do**
  if $x_{\text{r}} \in \mathcal{X}_{\text{obs}}^{\text{aug}}$ **then**
    $x_{\text{goal}}^{\text{loc}} \leftarrow x_{\text{r}}^{\text{free}}$;
  end if
  $x_{\text{r}} \leftarrow x_{\text{r}}^{\text{free}}$;
end for
Return $(x_{\text{start}}^{\text{loc}}, x_{\text{goal}}^{\text{loc}})$;

A. An Aircraft Example

Consider the linear time invariant continuous-time F16 aircraft system as in [37], which has the form of,

$$\dot{x} = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u. \quad (37)$$

We set the user-defined $M = I_3$, and $R = 0.1$. The finite horizon is $T = 45$ s. The initial and final states of the TPBVP are $x_0 = [1\ 5\ 1]^\intercal$, $x_{\text{r}} = [2\ 7\ 3]^\intercal$. The final Riccati matrix is $P(T) = 0.5I_3$ and the final control action is $u(T) = 0.001$. We select the weights of the gradient descent method $\alpha_{\text{c}} = 90$, and $\alpha_{\text{a}} = 2.5$. The small fixed value of the internal dynamics is $\Delta t = 0.05$ s. The initial values of $\hat{W}_{\text{c}}$, and $\hat{W}_{\text{a}}$ are randomly selected, except of the last element of $\hat{W}_{\text{c}}$ that needs to be other than zero. This elements is related with the Q_{uu} value that is inverted in (15). We apply the Algorithm 1 (Lines 7-12) of the RRT-Q* for a single run $K = 1$, which guarantees asymptotic convergence according to Theorem 2. The evolution of the states is presented in Figure 5. It is to be noted that the optimal solution was obtained after solving the (8) backwards in time with the aforementioned final state, and then we applied the solution to (9) as described in [9].

B. Motion Planning RRT-Q* Simulations

Consider now the system with dynamics including a double integrator system adopted from [16] along with the Maxwell-slip model [38]. The robot slips on a frictioned flat surface and its mass drifts similarly to [39] at every i-TPBVP. While the mass m is translating along the x-axis and the y-axis direction, a spring-damper system models the friction with coefficients k_{x}, c_{x}, and k_{y}, c_{y} respectively. The system is described by,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \end{bmatrix} = A \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} + B \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}, \quad (38)$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_{\text{x}}}{m(i)} & 0 & -\frac{c_{\text{x}}}{m(i)} & 0 \\ 0 & -\frac{k_{\text{y}}}{m(i)} & 0 & -\frac{c_{\text{y}}}{m(i)} \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m(i)} & 0 \\ 0 & \frac{1}{m(i)} \end{bmatrix},$$

where x_1, y_1 are the translations, $\dot{x}_1 = x_2$, $\dot{y}_1 = y_2$ are the velocities, and $\ddot{x}_1 = \dot{x}_2$, $\ddot{y}_1 = \dot{y}_2$ are the accelerations along the x and y axes respectively. The vector $\begin{bmatrix} f_1 & f_2 \end{bmatrix}^\intercal$ is the input force. The mass decreases at every i-TPBVP due to fuel consumption. The drift mass model is described as,

$$m(i) = m_{\text{f}} \text{e}^{-\alpha i} + m_{\text{n}}. \quad (39)$$

where m_{f} is the fuel mass that is consumed at every i-step, m_{n} is the net mass of the robot without fuel, and α is the fuel decay rate. We set the finite horizon $T = 10$ s for every run and the admissible window $\beta = 5\%$. The user-defined matrices are $M = I_4$, and $R = 0.1I_2$. The final Riccati matrix is $P(T) = 0.5I_4$ and the final control action is $u(T) = 0.001$. We set $\alpha_{\text{c}} = 50$, and $\alpha_{\text{a}} = 2.5$ by following the Theorem 2. The small fixed value of the internal dynamics is $\Delta t = 0.05$ s. The initial values of $\hat{W}_{\text{c}}$, and $\hat{W}_{\text{a}}$ are randomly selected, except of the last three elements of $\hat{W}_{\text{c}}$ that need to be other than zero. These elements are related with the Q_{uu} values that are inverted in (15). Note that there are three elements, because the user

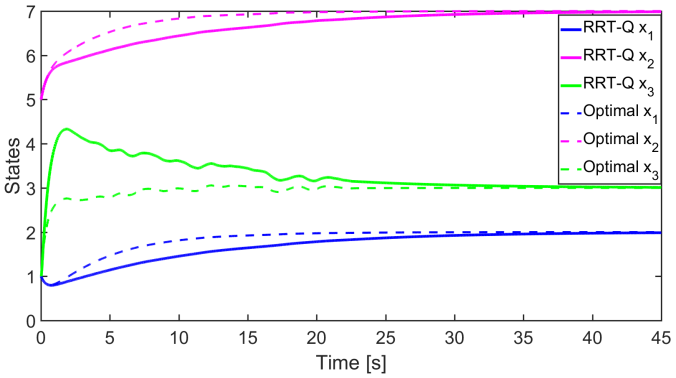

Fig. 5. The evolution of the states for a TPBVP of an F16 aircraft system using RRT-Q*. The states asymptotically converge to the optimal solution as described in Theorem 2.

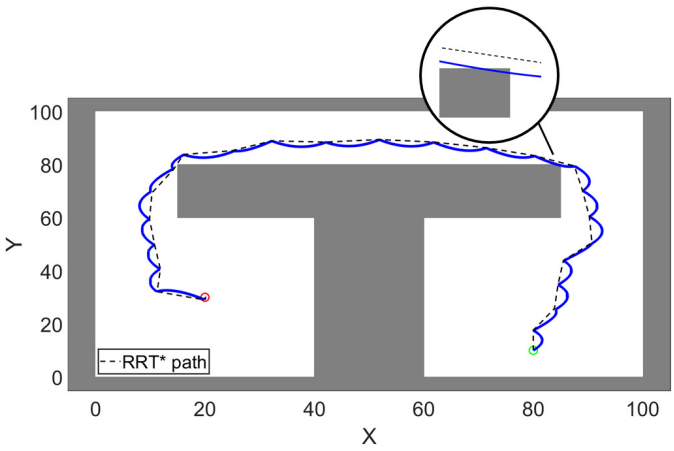

Fig. 6. The partially RRT-Q* framework is equipped with only the online policy estimation and the terminal state evaluation and not the static obstacle augmentation, and the local re-planning phases.

defined matrix R is symmetric and we are also employing the half-vectorization in (16). The initial and the final states $\mathcal{X}_0$, $\mathcal{X}_\text{G}$ are given by the RRT*. The stiffness, and the damping coefficients are $k_\text{x} = k_\text{y} = 20$ N/m, and $c_\text{x} = c_\text{y} = 45$ kg/s respectively. The net mass of the robot is $m_\text{n} = 10$ kg, the fuel mass at the beginning of the problem $m_\text{f} = 30$ kg, and the consumption rate $\alpha = 0.05$. The state space is described by the Cartesian space $\mathcal{X} \in [0, 100] \times [0, 100] \subset \mathbb{R}^2$. We consider exact knowledge of the obstacle space, we require full state feedback, and we compute offline the global path with RRT*.

We perform three sets of simulations. First, we evaluate the efficacy of the proposed methodology, yet without performing the static obstacle augmentation, and the local re-planning steps. Next, we assess the full framework of the RRT-Q* for the linear-time varying case, by simulating the dynamics given in (38) and the mass drift model in (39). Lastly, we demonstrate the ability of the algorithm to perform in challenging obstacle environments.

For the first case, the proposed framework is depicted in Figure 6, without the static obstacle augmentation and the local re-planning phases. The motion of the robot is illustrated with a blue solid line, the start state x_start with a green circle, the goal state x_goal with a red circle, and the global path π with a dashed black line. The motion of the robot equipped with the partially proposed framework efficiently performs waypoint tracking, yet a collision with the obstacle in occurred. This reveals that only in an obstacle-free environment the RRT-Q* can operate without the need of implementing the static obstacle augmentation and the local re-planning framework.

Next, we demonstrate the ability of the Algorithm 1 to perform in obstacle environments, even when a variation in the system occurs. We gather full state feedback from the system described in (38), yet with a drift mass at every i-TPBVP as in (39). The total mass reduction according to the selected parameters is 75% and the robot motion is depicted in Figure 7. The red crosses represent the discarded nodes V_aug that are located in the augmented obstacle space $\mathcal{X}_\text{obs}^\text{aug}$. The inflated space is drawn with light purple. The local start state $x_\text{start}^\text{loc}$ and the local goal state x_goal^loc are presented with red rectangles, while the local graph $\mathcal{G}_\text{new}$ is shown with solid light orange lines. The feasible local path π^loc is illustrated with a red dashed line. Moreover, we demonstrate the evolution of all 25 TPBVP in Figure 9. The solid blue line and the solid pink line illustrate the propagation of states on x-axis and y-axis respectively. The dashed red line and the dashed green line represent the reference coordinates x_r and y_r respectively. The vertical dashed line depicts the final time according to the terminal state evaluation framework for admissible window factor $\beta = 5\%$. The motion of the robot, equipped with the proposed algorithmic framework RRT-Q*, safely navigates from the start state x_start to the goal state x_goal by avoiding collision with the obstacle, even when online re-planning in two areas is required. Also, the RRT-Q* efficiently handles the system variations in mass. This reveals that the governing dynamics do not affect the performance of our proposed motion planning technique. We select such variations (i.e. 75%) to demonstrate the efficacy in extreme case scenarios, yet this basically means that we drop 3% of the mass fuel after every i-step and not during each i-TPBVP. Such applications may include robotic manipulators after picking or dropping objects. As a result the proposed framework is a uniform model-free approach and can be applied to systems that even alter their dynamics after every i-step. Figure 8 presents an agent equipped with our proposed algorithmic framework that avoids collision in a challenging obstacle environment.

TABLE I
TERMINAL STATE EVALUATION

Factor β (%)	Iterations	Reduction (%)
10	$2,010$	59.8
5	$2,420$	51.6
1	$3,380$	32.4
0	$5,000$	0

In Table I a comparison study of the terminal state evaluation for the problem depicted in Figures 7 and 9 is presented. The terminal state evaluation employs an admissible window factor β and the initial state distance D_0. For various factors $\beta = 1\%$, $\beta = 5\%$, and $\beta = 10\%$ the iteration number is being reduced by 59.8%, 51.6%, and 32.4% respectively. The total number of iterations $(5,000)$ follows from the $K = 25$

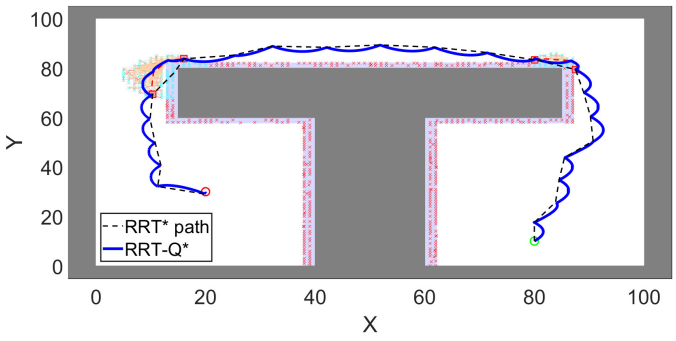

Fig. 7. The motion of a robot with the proposed algorithmic framework RRT-Q* that performs efficiently online re-planning in a small free space and avoids the collision with the obstacle. The change in dynamics is due to a sequential 75% reduction of the mass. The proposed RRT-Q* can efficiently address such variations in dynamics, while being optimal.

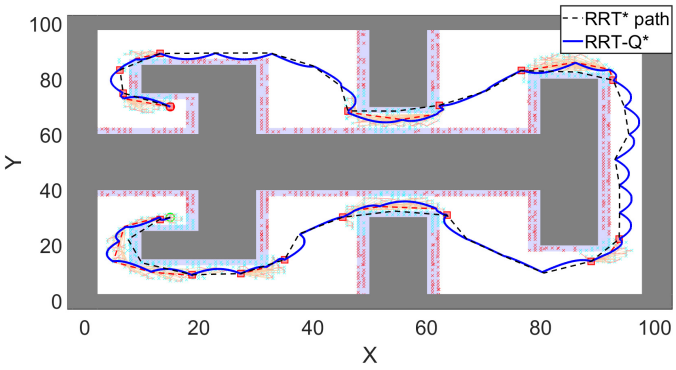

Fig. 8. The online kinodynamic motion planning RRT-Q* with completely unknown dynamics in a complex obstacle environment. The algorithm performs local re-planning at eight areas according to the static obstacle augmentation.

TPBVP, the finite horizon $T = 10$ s, and the internal dynamics time $\Delta t = 0.05$ s. The terminal state evaluation facilitates the online implementation, contributing to an important reduction of the computational effort.

C. Qualitative Comparison

In Table II we provide a qualitative comparison of the proposed technique with other kinodynamic motion planning techniques. We consider four specifications, the optimality; the online implementation; the robustness; and knowledge about the system dynamics.

We select optimality as a basis of this comparison. Although, some approaches evaluate different performance criteria and other time constraints. More specifically, LQR-Trees and our approach solve the minimum-energy problem in a finite horizon as given in (2). LQR-RRT* assesses a minimum energy performance, yet in an infinite horizon. Kinodynamic RRT* evaluates a minimum time-fuel performance in a finite horizon. Regarding the optimization performance, minimum energy problems penalize the control and the states simultaneously, while minimum time-fuel problems penalize only the control. Thus, minimizing the energy provides a variety of control design choices that corresponds to better performance [9]. Indeed, time horizon constraints are also important. More specifically, finite horizon ensures optimal performance at a specific time, while infinite horizon does not consider any time constraints for optimality. Technically, finite horizon induces the differential Riccati equation, rather than the algebraic Riccati equation of infinite horizon problems. This time dependence makes the solution of the problem more challenging. But in practice, most motion planning applications require the completion of a mission at a specific time. Hence, for the motion planning paradigm, the finite horizon approach is crucial. It is to be noted that our analysis provides Theorem 2, which guarantees closed-loop stability of the equilibrium point, and thus optimality is guaranteed with asymptotic convergence properties.

Online implementation can be only achieved with the proposed framework, as it requires the computation of two simple gradient descent laws given by (22), (23), and the local

TABLE II
KINODYNAMIC MOTION PLANNING COMPARISON

	LQR-Trees [3]	**LQR-RRT*** [14]	**Kinodynamic RRT*** [16]	**RRT-Q***
Optimality	✓	✓	✓	✓
Horizon	Finite	Infinite	Finite	Finite
Performance	Energy	Energy	Fuel-Time	Energy
Online	✗	✗	✗	✓
Feedback	✓	✓	✗	✓
Model-Free	✗	✗	✗	✓
Robustness	Bounded	Bounded	✗	✓

re-planning at a relatively small free space without any re-sampling. The other works need to solve the Riccati equation that inherits extensive offline computation and comprises the model of the system. The proposed framework is model-free, as we approximate the optimal policy in (16) without any information of the system dynamics. The other works require the system's model for their calculations. To this end, our technique is suitable for any unmanned vehicle with linear dynamics that satisfy the Assumtpion 1. We compare robustness of controllers in model uncertainties. The structure of LQR-Trees, and LQR-RRT* provide some level of robustness, i.e. structured uncertainties with certain limits [40]. Yet, if these limits are exceeded, the system becomes unstable. Kinodynamic RRT* employs an open-loop controller which has precomputed offline the policies, and hence uncertainties cannot be transcended. Our methodology does not employ the dynamics of the system, and thus no model uncertainties appear.

VII. DISCUSSION

In this section we discuss the computational complexity of the proposed algorithm. Furthermore, we provide instructions for the implementation of the algorithmic framework in real world scenarios and we list its limitations.

A. Computational Complexity

The computational complexity of the offline process depends purely on the RRT*. The required time to build the graph $\mathcal{G} = (V, E)$, with $V_\mathcal{G} = |V|$, is $\Theta(V_\mathcal{G} \log V_\mathcal{G})$ [4], where $\Theta(\cdot)$ is the tight bound.

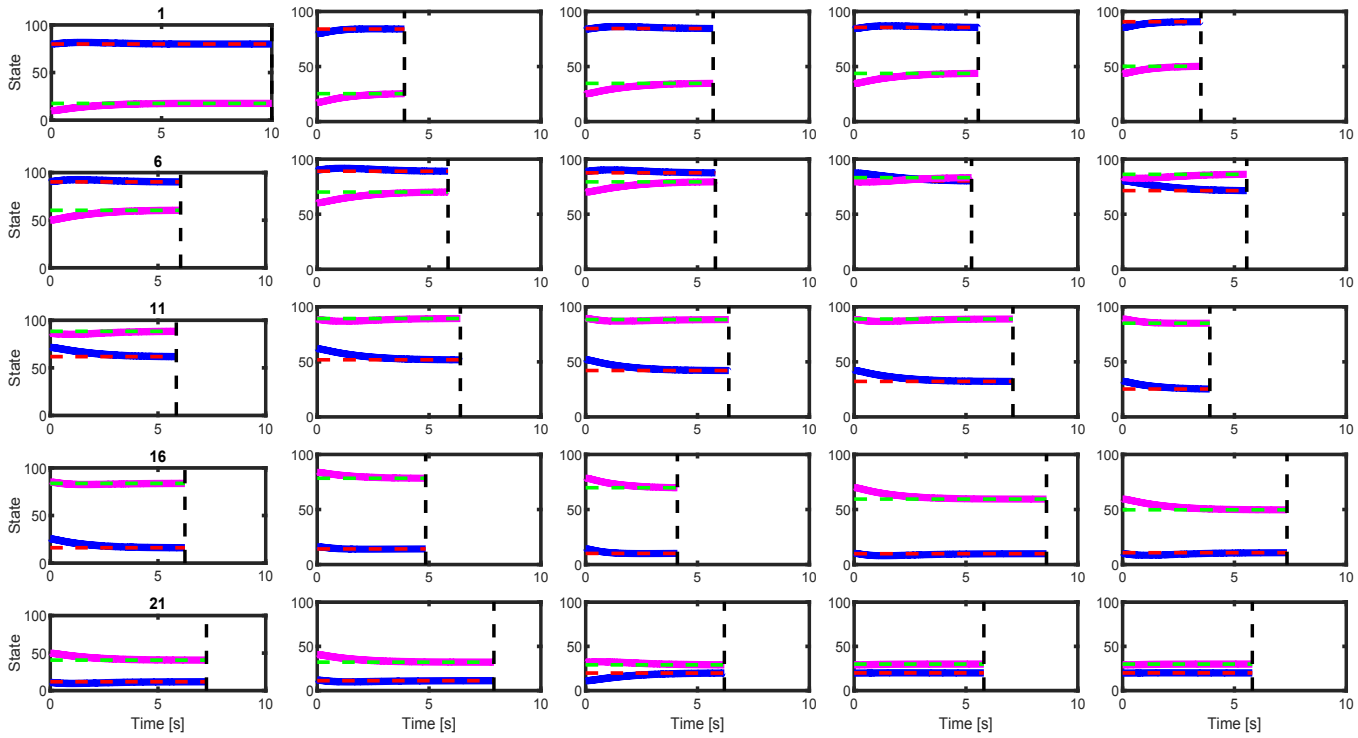

Fig. 9. Evolution of the states for 25 problems. That is equivalent with the problem depicted in Figure 7, but shows the evolution of each TPBVP and the effect of the terminal state evaluation framework.

The time complexity of the rest algorithm has to be significantly lower to allow for the online implementation of the RRT-Q*. For the Q-function estimation the complexity is determined by (16), (22). More specifically, the critic network grows quadratic with the size of the augmented state $\hat{U} \in \mathbb{R}^{n+m}$, that is $\mathcal{O}((n+m)^2)$. Similarly, the control estimation is determined by (17), (23), where the time complexity yields $\mathcal{O}(n^2)$. Essentially, the computational requirements rely on the gradient descent estimation algorithms as described in [41]. The re-planning phase is based on the RRT* and thus the time complexity is $\Theta(V_{\text{free}} \log V_{\text{free}})$. For the connectedness assessment of the local space, the time complexity is given as $\mathcal{O}(l)$, with l the number of checking points on the circle circumference.

The overall time complexity of the online phase is $\mathcal{O}(V_{\text{free}} \log V_{\text{free}} + (n+m)^2)$. However, in the proposed methodology the local free space is significantly reduced $\mathcal{X}_{\text{free}}^{\text{loc}}$, so that the impact of the free re-planning nodes V_{free} is negligible. Also, the local re-planning phase is activated only when a collision of the global is occurred. To this end, during most of the navigation time the overall time complexity of the online phase further reduces to $\mathcal{O}((n+m)^2)$.

B. Implementation Details

The proposed framework requires standard treatment for its implementation. First, the exact map of the environment need to be given a priori. The exact knowledge of the map is mandatory for the RRT*, so this requirement applies to every methodology that employs such algorithms. For the online, model-free control of the robot, an accurate full state and input feedback need to be given to the system as described in Figure 1. That is also common to any control strategy that employs linear dynamics with full state feedback. Moreover, according to Assumption 1 the system is detectable, which ensures that we can compute the state from knowledge of the output. Next, the actuation scheme need to produce rich enough signals to satisfy the persistency of excitation condition as described in Lemma 2. Likewise, most system identification techniques or adaptive control schemes need to satisfy such a condition.

C. Limitations of RRT-Q*

The limitations associated with the proposed methodology are important for the applicability to various cases. The RRT-Q* can be employed for unknown continuous-time linear systems with the Assumption 1. Now, if a nonlinear system can be linearized about an equilibrium point, and as long as the unknown linearized plant and input matrix satisfy the Assumption 1, then the methodology is valid. Yet, it is to be noted that the proposed scheme, as well as the stability analysis does not concern the nonlinear case. Moreover, the proposed framework is robust to model uncertainties, but not to any kind of disturbances, such as external disturbances or measurement noise. Also, the persistency of excitation condition is often challenging to be satisfied, and thus, for these cases, relaxed persistency of excitation approaches [42], [43] may be more applicable. Another limitation of RRT-Q* lies in the structure of the environment. It is to be noted that our approach employs static obstacles, but this does not necessary dictates a static environment. More specifically, the

obstacles cannot rotate and translate, yet they are augmented for the re-planning phase. Thus, the environment is dynamic, as the shape of the obstacles increases through time depending on the kinodynamic distance. That is the inauguration of our work towards online, model-free autonomous navigation with continuous-time Q-learning in dynamic environments with moving obstacles.

VIII. Conclusion

This paper proposed an online motion planning algorithmic framework RRT-Q*. More precisely, we employed Q-learning to approximate the optimal policy of a continuous linear time-invariant system and navigate in the free space given TPBVP from the RRT*. We discussed the mathematical formulation that guarantees asymptotic stability and optimality of kinodynamic motion planning of systems with completely unknown/uncertain dynamics. We presented the algorithmic framework of the RRT-Q* and we proposed a terminal state evaluation that reduces significantly the computational effort and facilitates online implementation. We also discussed a static obstacle augmentation along with a local re-planning framework that facilitates the online and collision-free implementation. We provided simulation examples that validate the efficacy of the proposed RRT-Q*.

References

[1] 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.

[2] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.

[3] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-Trees: Feedback motion planning via sums-of-squares verification," *International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[5] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.

[6] R. Allen and M. Pavone, "A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance," in *AIAA Guidance, Navigation, and Control Conference*, 2016, p. 1374.

[7] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[8] W. Wolfslag, M. Bharatheesha, T. M. Moerland, and M. Wisse, "RRT-Colearn: Towards kinodynamic planning without numerical trajectory optimization," *IEEE Robotics and Automation Letters*, 2018.

[9] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos, *Optimal Control*, 3rd ed. John Wiley & Sons,, 2012.

[10] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with Gaussian processes," in *European Control Conference*, 2015, pp. 2496–2501.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[12] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annual Review of Control, Robotics, and Autonomous Systems*.

[13] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[14] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 2537–2542.

[15] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, "Optimal sampling-based planning for linear-quadratic kinodynamic systems," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 2429–2436.

[16] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 5054–5061.

[17] Y. Li, R. Cui, Z. Li, and D. Xu, "Neural network approximation-based near-optimal motion planning with kinodynamic constraints using RRT," *IEEE Transactions on Industrial Electronics*, 2018.

[18] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

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

[20] M. Otte and E. Frazzoli, "RRT$^{\text{X}}$: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[21] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Courier Corporation, 2012.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[23] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2042–2062, 2018.

[24] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[25] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2010, vol. 39.

[26] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Systems*, vol. 32, no. 6, pp. 76–105, 2012.

[27] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal adaptive control and differential games by reinforcement learning principles*. IET, 2013, vol. 2.

[28] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[29] A. Faust, O. Ramirez, M. Fiser, K. Oslund, A. Francis, J. Davidson, and L. Tapia, "PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *IEEE International Conference on Robotics and Automation*, 2018.

[30] P. Mehta and S. Meyn, "Q-learning and pontryagin's minimum principle," in *IEEE Conf. on Decision and Control*, 2009, pp. 3598–3605.

[31] Y. Jiang and Z.-P. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics," *Automatica*, vol. 48, no. 10, pp. 2699–2704, 2012.

[32] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.

[33] K. G. Vamvoudakis, "Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach," *Systems & Control Letters*, vol. 100, pp. 14–20, 2017.

[34] W. Rudin, *Principles of mathematical analysis*. McGraw-hill New York, 1964, vol. 3.

[35] A. Bryson and Y.-C. Ho, "Applied optimal control: Optimization, estimation, and control (revised edition)," *Levittown, Pennsylvania: Taylor & Francis*, 1975.

[36] H. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

[37] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft control and simulation: Dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[38] F. Al-Bender, V. Lampaert, and J. Swevers, "The generalized Maxwell-slip model: A novel model for friction simulation and compensation," *IEEE Transactions on Automatic Control*, vol. 50, no. 11, pp. 1883–1887, 2005.

[39] J. R. Forbes and C. J. Damaren, "Passive linear time-varying systems: State-space realizations, stability in feedback, and controller synthesis," in *American Control Conference*, 2010, pp. 1097–1104.

[40] J. Doyle and G. Stein, "Multivariable feedback design: Concepts for a classical/modern synthesis," *IEEE Transactions on Automatic Control*, vol. 26, no. 1, pp. 4–16, 1981.

[41] P. Baldi, "Gradient descent learning algorithm overview: A general dynamical systems perspective," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 182–195, 1995.
[42] K. G. Vamvoudakis, M. F. Miranda, and J. P. Hespanha, "Asymptotically stable adaptive-optimal control algorithm with saturating actuators and relaxed persistence of excitation." *IEEE Transactions Neural Networks and Learning Systems*, vol. 27, no. 11, pp. 2386–2398, 2016.
[43] G. Chowdhary and E. Johnson, "Concurrent learning for convergence in adaptive control without persistency of excitation," in *IEEE Conference on Decision and Control*, 2010, pp. 3674–3679.