Unified Multi-Rate Control: from Low Level Actuation to High Level Planning

Ugo Rosolia, *Member, IEEE,*, Andrew Singletary, *Member, IEEE,* and Aaron D. Ames, *Senior Member, IEEE*

***Abstract*—In this paper we present a hierarchical multi-rate control architecture for nonlinear autonomous systems operating in partially observable environments. Control objectives are expressed using syntactically co-safe Linear Temporal Logic (LTL) specifications and the nonlinear system is subject to state and input constraints. At the highest level of abstraction, we model the system-environment interaction using a discrete Mixed Observable Markov Decision Problem (MOMDP), where the environment states are partially observed. The high level control policy is used to update the constraint sets and cost function of a Model Predictive Controller (MPC) which plans a reference trajectory. Afterwards, the MPC planned trajectory is fed to a low-level high-frequency tracking controller, which leverages Control Barrier Functions (CBFs) to guarantee bounded tracking errors. Our strategy is based on model abstractions of increasing complexity and layers running at different frequencies. We show that the proposed hierarchical multi-rate control architecture maximizes the probability of satisfying the high-level specifications while guaranteeing state and input constraint satisfaction. Finally, we tested the proposed strategy in simulations and experiments on examples inspired by the Mars exploration mission, where only partial environment observations are available.**

***Index Terms*—partially observable, predictive control, control barrier function, multi-rate control, hierarchical control.**

I. Introduction

Control design for complex cyber-physical systems, which are described by continuous and discrete variables, is usually divided into different layers [1]–[9]. Each layer is designed using model of increasing accuracy and complexity, which allow the controller to take high-level decision, e.g. perform an overtaking maneuver, and to compute low-level commands, e.g. the input current to a motor. High-level decisions and low-level control actions are computed at different frequencies and the interaction between layers should be taken into account to guarantee safety of the closed-loop system [1].

Control policies for high-level decision making are usually synthesized using discrete model abstractions and the high-level control objectives are often expressed by Linear Temporal Logic (LTL) formulas [10], as they are a formalism to express high-level system behaviors using logical and temporal operators [10]. Motion planning with LTL and syntactically co-safe LTL (scLTL) specifications has been widely studied in literature [1]–[3], [5], [6], [11]–[20]. For deterministic systems with finite-state spaces several approaches and toolboxes are available for synthesis [1]–[3], [11]–[13]. When the system-environment interaction are uncertain, the high-level abstractions are described by discrete Markov Decision Processes (MDPs) and the high-level decision making problem can be solved exactly using dynamic programming, policy iteration and linear programming strategies [21]. On the other hand, when the system dynamics are uncertain and only partial observations are available, the system-environment interaction can be modeled using discrete Partially Observable Markov Decision Processes (POMPDs). Computing a control policy in POMDPs settings is NP-hard [22], but approximate solutions can be computed using finite state controllers [23] and performing point-based approximations [24].

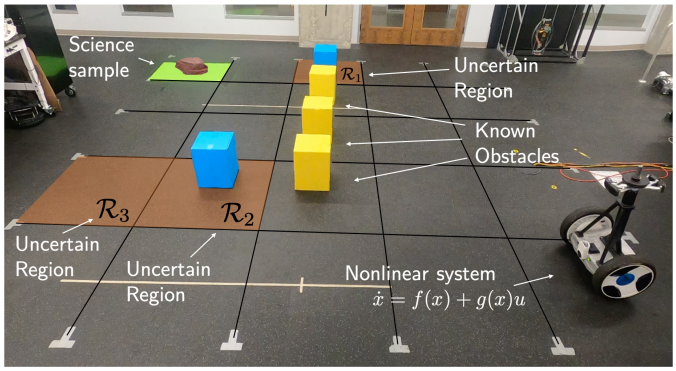

Fig. 1. This figure shows an environment composed of 25 cells, 3 obstacles (yellow boxes) and 3 uncertain regions (light brown). In this example the goal of the controller is to explore the state space in order to find a science sample.

Given a high-level decision, reachability-based techniques [1], [2] or simulation-based abstractions [3], [4] maybe used to compute a goal set for the continuous time system, e.g., a subset of a lane where we would like to drive the vehicle when performing an overtaking maneuver. Therefore, the input to the system's actuators is computed solving mid-level planning and low-level control problems, which have been studies extensively in literature [9], [25]–[34]. The planning problem is usually defined for a simplified model and the resulting reference trajectory is then tracked using low-level controllers, which leverage the nonlinear system dynamics. Tracking controllers may be synthesized using Hamilton-Jacobi (HJ) reachability analysis [9] or sum-of-squares programming [29], [30]. Another strategy to solve mid-level planning and low-level control problems is to use nonlinear tube MPC [31]–[35], where the difference between the planned trajectory and the actual one is over approximated using Lyapunov based analysis or Lipschitz properties of the nonlinear dynamics. When the planned trajectory is computed without taking into account tracking errors, safety can be guaranteed using filters which, given a desired mid-level command and/or planned

U. Rosolia, A. Singletary and A. D. Ames are with the AMBER lab at the California Institute of Technology, Pasadena, CA, USA, e-mail: {urosolia, asinglet, ames}@caltech.edu.

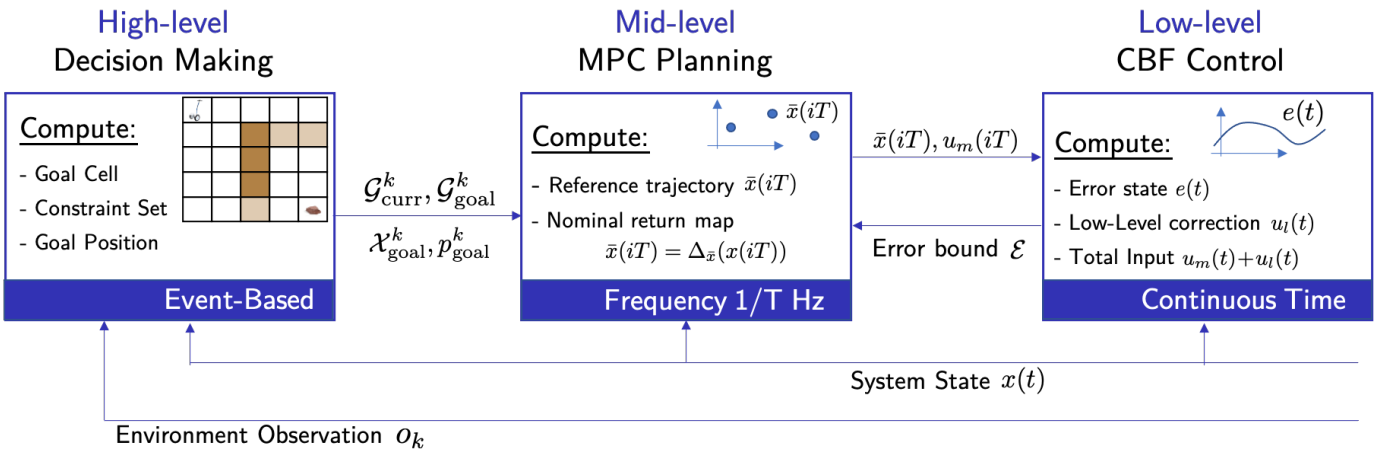

Fig. 2. Multi-rate control architecture. The high-level decision maker leverages the system's state $x(t)$ and partial environment observations o_k to compute a goal cell, the constraint set and the goal positions, which are fed to the mid-level MPC planner. The planner computes a reference trajectory given the tracking error bounds $\mathcal{E}$ from the low-level tracking controller. Finally at the lowest level, the control action is computed summing up the mid-level input $u_m(t)$ and the low-level input $u_l(t)$.

trajectory, compute a safe control action using CBFs [25]–[27], feasibility of an MPC problem [28] or reachability analysis [8].

In this work we present a multi-rate hierarchical control scheme for nonlinear systems operating in partially observable environments. Our architecture, which is composed by three layers running at different frequencies, guarantees constraint satisfaction and maximization of the closed-loop probability of satisfying the high-level specifications. At the lowest level, we leverage Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs), which are based on the continuous time nonlinear system model and guarantee a bounded tracking error. The mid-level planning layer computes a reference trajectory using an MPC, which leverages a simplified prediction model and the low-level tracking error bounds. Finally, at the highest level of abstraction we model the system-environment interaction using Mixed Observable Markov Decision Processes (MOMDPs), which allows us to account for partial environment observations.

Contribution: Our contribution is threefold. First, we show how to integrate a CLF-CBF tracking controller with an MPC planner. Compared to our previous work [36], the MPC planner is based on a fixed-tube robust MPC scheme [37], where the initial state of the planned trajectory is an optimization variable. For this reason, the proposed strategy does not require the online computation of robust reachable sets to formulate the MPC problem and therefore it is computationally more efficient than the approach proposed in [36]. Second, we introduce a mid-level planner, which leverages an MPC with time-varying constraint sets and cost function. These time-varying components are given by the high-level decision maker and they can jeopardize the feasibility of the MPC problem. For this reason, we propose a contingency scheme, which guarantees feasibility of the MPC planner with time-varying components. The feasibility of this contingency plan and the low-level tracking error bounds guarantee safety, when a local reachability assumption on the system dynamics is satisfied. Such reachability assumption, which is tailored to navigation problems, together with the proposed contingency scheme allows us to avoid the construction of finite state abstractions defined over the entire state space. Third, we show how to model the system-environment interaction using Mixed Observable Markov Decision Processes (MOMDPs), where the system state is fully observable and the environment state is partially observable. We use the high-level action from the MOMDP to update the MPC time-varying components and we show that our hierarchical strategy guarantees that the probability of satisfying the high-level specifications is maximized. Finally, we test our strategy on navigation task shown in Figure 1, where a Segway like-robot has to find science samples while navigating a partially observable environment.

This paper is organized as follows. Section II describes the problem formulation. The hierarchical architecture is introduced in Section III and the closed-loop properties are discussed in Section IV. Finally, we illustrate the effectiveness of the proposed strategy with high-fidelity simulations and hardware experiments.

Notation: The Minkowski sum of two sets $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{Y} \subset \mathbb{R}^n$ is denoted as $\mathcal{X} \oplus \mathcal{Y}$, and the Pontryagin difference as $\mathcal{X} \ominus \mathcal{Y}$. $\mathcal{K}^e$ is the set of extended class-$\mathcal{K}^e$ functions β which are strictly increasing and $\beta(0) = 0$. For for a set $\mathcal{A} \subset \mathbb{R}^n$ and a vector $x \in \mathbb{R}^n$, we denote the projection

$$\text{Proj}(x, \mathcal{A}) = \underset{d \in \mathcal{A}}{\text{argmin}} \, ||x - d||_2.$$

and the cardinality of the set $\mathcal{A}$ as $|\mathcal{A}|$. We define $\mathbb{Z}_{0+} = \{0, 1, 2, \ldots\}$ and $\mathbb{R}_{0+} = \{x \in \mathbb{R}^n | x \geq 0\}$ which denote the set of positive integers and real numbers, respectively. Finally, given $t \in \mathbb{R}_{0+}$ and $T \in \mathbb{Z}_{0+}$ we define $\lfloor t/T \rfloor = \text{floor}(t/T)$.

II. Problem Formulation

This section describes the problem formulation. First, we introduce the continuous system dynamics. Afterwards, we present the discrete environment model. Finally, we describe the synthesis goals and we summarize the overall control architecture from Figure 2.

System Model: As discussed in the introduction, our goal is to design a controller for nonlinear dynamical systems. In particular, we consider nonlinear control affine systems of the following form:

$$\dot{x}(t) = f\big(x(t)\big) + g\big(x(t)\big)u(t), \tag{1}$$

where f and g are Lipschitz continuous, the input $u(t) \in \mathbb{R}^{n_u}$ and the state $x(t) = [p^\top(t), q^\top(t)]^\top \in \mathbb{R}^{n_x}$ for the position vector $p(t) \in \mathbb{R}^{n_p}$ and the vector $q(t) \in \mathbb{R}^{n_q}$ collecting the remaining states. Furthermore, the above system is subject to the following state and input constraints:

$$u(t) \in \mathcal{U}, p(t_i) \in \mathcal{X}_p \text{ and } q(t_i) \in \mathcal{X}_q \tag{2}$$

for all $t \in \mathbb{R}_{0+}$ and $t_i = iT$ for all $i \in \mathbb{Z}_{0+}$. The time constant T is specified by the user and, as it will be clear later on, it defines the frequency at which the controller updates the planned trajectory. In the above equation (2), $\mathcal{X}_p$ represents free space and $\mathcal{X}_q$ is a user-defined constraint set.

Remark 1. We consider state constraints which are enforced pointwise in time to streamline the presentation. The proposed control strategy can be extended to account for constraints which must hold for all time $t \in \mathbb{R}_{0+}$. In this case, it is required to modify the low-level controller as discussed in [36].

Environment Model: We consider nonlinear dynamical systems operating in partially observable environments, which are partitioned into $\mathcal{C}_1, \ldots, \mathcal{C}_c$ cells as in the example from Figure 1. We assume that the state of the system is perfectly observable, but we are given only partial observations about the environment state. Thus, at the highest level of abstraction, we model the interaction between the nonlinear system (1) and the environment using a Mixed Observable Markov Decision Process (MOMDP). A MOMDP provides a sequential decision-making formalism for high-level planning under mixed full and partial observations [38] and it is defined as tuple $(\mathcal{S}, \mathcal{Z}, \mathcal{A}, \mathcal{O}, T_s, T_z, O)$, where

- $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$ is a set of fully observable states;
- $\mathcal{Z} = \{1, \ldots, |\mathcal{Z}|\}$ is a set of partially observable states;
- $\mathcal{A} = \{1, \ldots, |\mathcal{A}|\}$ is a set of actions;
- $\mathcal{O} = \{1, \ldots, |\mathcal{O}|\}$ is the set of observations for the partially observable state $z \in \mathcal{Z}$;
- The function $T_s : \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ describes the probability of transitioning to a state s' given the action a and system's state (s, z), i.e.,

$$T_s(s, z, a, s') := P(s_{k+1} = s' | s_k = s, z_k = z, a_k = a);$$

- The function $T_z : \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \times \mathcal{S} \times \mathcal{Z} \to [0, 1]$ describes the probability of transitioning to a state z' given the action a, the successor observable state s' and the system's current state (s, z), i.e.,

$$T_z(s, z, a, s', z') := P(z_{k+1} = z' | s_k = s, z_k = z, a_k = a, s_{k+1} = s');$$

- The function $O : \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$ describes the probability of observing the measurement $o \in \mathcal{O}$, given the current state of the system $(s', z') \in \mathcal{S} \times \mathcal{Z}$ and the action a applied at the previous time step, i.e.,

$$O(s', z', a, o) := P(o_k = o | s_k = s', z_k = z', a_{k-1} = a);$$

MOMDPs were introduced in [38] to model systems where a subspace of the state space is perfectly observable. As we will discuss later on, our hierarchical architecture leverages robust control methodologies to guarantee that the high-level transitions of the observable state are deterministic. Therefore, we consider MOMDPs with the following transitions dyanmics:

$$T_s(s, z, a, s') = \begin{cases} 1 & \text{If } s' = f_s(s, z, a) \\ 0 & \text{Else} \end{cases},$$

for the high-level update function $f_s : \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \to \mathcal{S}$.

Specifications: High-level objectives are expressed using syntactically co-safe Linear Temporal Logic (scLTL) specifications. An scLTL specification is defined as follows:

$$\psi := p \mid \neg p \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \psi_1 U \psi_2 \mid \psi_1 \bigcirc \psi_2,$$

where the atomic proposition $p \in \{\texttt{true}, \texttt{false}\}$ and ψ, ψ_1, ψ_2 are scLTL formulas, which can be defined using the logic operators negation ($\neg$), conjunction ($\wedge$) and disjunction ($\vee$). Furthermore, scLTL formulas can be specified using the temporal operators until (U) and next ($\bigcirc$). Each atomic proposition p is associated with a subset of the high-level state space $\mathcal{P} \subset \mathcal{S} \times \mathcal{Z}$ and, for a high-level state $\omega_k = (s_k, z_k)$, the proposition p is `true` if $\omega_k \in \mathcal{P}$. Finally, we say that a high-level trajectory $\boldsymbol{\omega} = [\omega_0, \omega_1, \ldots]$ satisfies the specification ψ and we write

$$\boldsymbol{\omega} \models \psi \tag{3}$$

when the following holds: $i)$ $\boldsymbol{\omega} \models p \iff \omega_k \in \mathcal{P}, \forall k \geq 0$, $ii)$ $\boldsymbol{\omega} \models \psi_1 \wedge \psi_2 \iff (\boldsymbol{\omega} \models \psi_1) \wedge (\boldsymbol{\omega} \models \psi_1)$, $iii)$ $\boldsymbol{\omega} \models \psi_1 \vee \psi_2 \iff (\boldsymbol{\omega} \models \psi_1) \vee (\boldsymbol{\omega} \models \psi_1)$, $iv)$ $\boldsymbol{\omega} \models \psi_1 U \psi_2 \iff \exists k$ such that $[\omega_0, \ldots, \omega_k] \models \psi_1$ and $[\omega_{k+1}, \omega_{k+2}, \ldots] \models \psi_2$, $v)$ $\boldsymbol{\omega} \models \psi_1 \bigcirc \psi_2 \iff [\omega_0, \ldots, \omega_k] \models \psi_1$ implies that $[\omega_{k+1}, \omega_{k+2}, \ldots] \models \psi_2$.

In the example from Figure 1, the high-level specification is not to collide with an obstacle until the goal is reached. This objective is expressed by the scLTL formula $\psi = \neg\texttt{Collision} U \texttt{Goal}$, where the atomic proposition `Collision` is `true` when the system (1) is in a cell occupied by an obstacle and the atomic `Goal` is `true` when the system (1) reached the goal location.

Synthesis Objectives: Given the system's state $x(t) \in \mathbb{R}^n$ and k observations $\boldsymbol{o}_{k-1} = [o_0, \ldots, o_{k-1}] \in \mathcal{O}^k$ about the environment, our goal is to design a control policy

$$\pi : \mathbb{R}^n \times \mathcal{O}^k \to \mathcal{U}, \tag{4}$$

which maps the state $x(t)$ and the observation vector $\boldsymbol{o}_{k-1}$ to the continuous control action $u \in \mathcal{U}$. Furthermore, the control policy (4) should guarantee that state and input constraints (2) are satisfied and that the probability of satisfying the specification (3) is maximized. Notice that standard control strategies for nonlinear systems can be used to guarantee constraint satisfaction [25]–[27], [31]–[35]. Furthermore, standard decision making methodologies for Partially Observable Markov

Decision Processes (POMDPs) can be used to synthesize a control policy which maximizes the probability of satisfying the specification [5], [6], [15]–[18]. In this paper, we bridge the gap between the two communities and we propose a hierarchical control scheme for nonlinear systems operating in partially observable environments, which guarantees that state and input constraints are satisfied and that the probability of satisfying the specifications is maximized.

Navigation Example: Figure 1 shows our motivating example, where a Segway has to reach a goal cell while avoiding know obstacles and exploring uncertain regions, which may be traversable with some probability. The Segway dynamics are nonlinear and the system is open-loop unstable, for this reason it is required a low-level high frequency controller that stabilizes the system during operations. On the other hand, at the highest level of abstraction we model the system using the discrete state $s_k \in \mathcal{S}$, which denotes the grid cell containing the nonlinear system (1), and the environment state $z_k \in \mathcal{Z}$ representing the traversability of the uncertain regions $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$. For instance in the example from Figure 1, the environment's state $z_k = [0, 0, 1]$ as regions $\mathcal{R}_1$ and $\mathcal{R}_2$ are not traversable and region $\mathcal{R}_3$ is traversable.

Strategy Overview: We summarize the proposed multi-rate control architecture depicted in Figure 2. The key idea is to divide the controller into three layers and compute the control action $u(t)$ as the summation of a high-frequency component $u_l(t)$ and a low-frequency component $u_m(t)$, i.e.,

$$u(t) = u_l(t) + u_m(t).$$

At the lowest level, the control action u_l is updated continuously (at high frequency $\sim 1/10kHz$) and it is computed using Control Barrier Functions (CFBs), which leverage the full-nonlinear model (1) to track a *reference trajectory* $\bar{x}(t)$. The middle layer updates at a constant frequency the reference trajectory $\bar{x}(t)$ and reference input u_m, which is computed using a Model Predictive Controller (MPC). This reference trajectory steers the system from the current state $x(t)$ to a *goal cell* $\mathcal{C}^k_{\text{goal}}$. Finally, the high-level planner computes the goal cell $\mathcal{C}^k_{\text{goal}}$ based on partial observations o_k about the environment.

III. Unified Multi-Rate Architecture

In this section, we describe the multi-rate control architecture. First, we design a low-level CLF-CBF controller, which tracks a reference state-input trajectory and guarantees bounded tracking errors. Afterwards, we show how to update the state-input reference trajectory leveraging an MPC, which is designed using a goal state computed from a discrete high-level decision maker. Finally, we introduce the hierarchical multi-rate architecture, which guarantees that the synthesis objectives from Section II are satisfied.

A. Low-Level Control

We leverage CBFs and CLFs to design a low-level tracking controller for the nonlinear system (1). CBFs guarantee safety for nonlinear system [26], but they are suboptimal as the control action is computed without forecasting the system's trajectory. For this reason, we use CBFs to enforce safety around a reference state-input trajectory that is computed at low frequency by the mid-level planner, as shown in Figure 2.

Error Model: At the lowest layer, the goal of the controller is to track a reference trajectory $\bar{x}(t)$. We assume that the reference trajectory is given by the following Linear Time-Varying (LTV) model:

$$\Sigma_{\bar{x}} : \begin{cases} \dot{\bar{x}}(t) = A_{\lfloor t/T \rfloor}\bar{x}(t) + B_{\lfloor t/T \rfloor}u_m(t), & t \in \mathcal{T} \\ \bar{x}^+(t) = \Delta_{\bar{x}}(x^-(t)), & t \in \mathcal{T}^c \end{cases}, \quad (5)$$

where $\mathcal{T}^c = \cup_{j=0}^{\infty}\{jT\}$, $\mathcal{T} = \cup_{j=0}^{\infty}(jT, (j+1)T)$ and the time T from (2) is specified by the user. Furthermore, we denote $\bar{x}^-(t) = \lim_{\tau \nearrow t}\bar{x}(\tau)$ and $x^+(t) = \lim_{\tau \searrow t}\bar{x}(\tau)$ as the right and left limits of the reference trajectory $\bar{x}(t) \in \mathbb{R}^n$, which is assumed left continuous. In the above system, the reference input $u_m(t) \in \mathbb{R}^d$ and the *reset map* $\Delta_{\bar{x}}$, which depends on the state of the nonlinear system (1), are given by the middle layer as we will discuss in Section III-B. Finally, the time-varying matrices $(A_{\lfloor t/T \rfloor}, B_{\lfloor t/T \rfloor})$ are known and, in practice, may be computed linearizing the system dynamics (1), as discussed in the result section.

Given the nonlinear system (1) and the LTV model (5), we define the error state $e(t) = x(t) - \bar{x}(t)$ and the associated error dynamics:

$$\Sigma_e : \begin{cases} \dot{e}(t) = f_e(x(t), \bar{x}(t), u_l(t) + u_m(t), t), & t \in \mathcal{T} \\ e^+(t) = x^+(t) - \bar{x}^+(t), & t \in \mathcal{T}^c \end{cases} \quad (6)$$

where the time-varying error dynamics are

$$\begin{aligned} &f_e(x,\bar{x}, u_l + u_m, t) \\ &\quad = f(x) + g(x)(u_l + u_m) - (A_{\lfloor t/T \rfloor}\bar{x} + B_{\lfloor t/T \rfloor}u_m). \end{aligned}$$

In the above definition, we dropped the dependence on time for states and inputs to simplify the notation. Furthermore, we introduce the low-level input constraint set $\mathcal{U}_l \subset \mathcal{U}$ and the mid-level input constraint set $\mathcal{U}_m \subset \mathcal{U}$ which partition the input space, i.e.,

$$\mathcal{U}_l \oplus \mathcal{U}_m = \mathcal{U}.$$

Next, we design a low-level controller which guarantees that the reference trajectory $\bar{x}(t)$ from the LTV model (5) is tracked within some error bounds.

Control Barrier and Lyapunov Functions: We show how to design a tracking controller using CBFs and CLFs [26]. First, we define the candidate Lyapunov function

$$V(e) = ||e||_Q, \quad (7)$$

where $||e||_Q = e^\top Qe$. Furthermore, we introduce the following safe set for the error dynamics (6):

$$\mathcal{E} = \{e \in \mathbb{R}^n : h_e(e) \geq 0\} \subset \mathbb{R}^n. \quad (8)$$

The above function h_e is defined by the user and it depends on the application as discussed in the result section.

Finally, the CBF associated with the safe set (8), and the CLF from (7) are used to define the following CLF-CBF Quadratic Program (QP):

$$\begin{aligned}\min_{v_l\in\mathcal{U}_l,\gamma}\ & ||v_l||_2 + c_1\gamma^2 \\ \text{s.t.}\ & \frac{\partial V(e)}{\partial e} f_e(x,\bar{x},v_l+u_m) \le -c_2 V(e) + \gamma \\ & \frac{\partial h_e(e)}{\partial e} f_e(x,\bar{x},v_l+u_m) \ge -\alpha_2(h_e(e)),\end{aligned} \tag{9}$$

where we dropped the time dependence to simplify the notation. In the above QP, the parameters $c_1 \in \mathbb{R}_{0+}$, $c_2 \in \mathbb{R}_{0+}$, $\alpha_1 \in \mathcal{K}^e$ and $\alpha_2 \in \mathcal{K}^e$. Let $v_l^*(t)$ be the optimal input action from the QP (9), then the low-level control policy is defined as follows:

$$u_l(t) = \pi_l\big(x(t), \bar{x}(t), u_m(t)\big) = v_l^*(t). \tag{10}$$

Assumption 1. The CLF-CBF QP (9) is feasible for all $e = x - \bar{x} \in \mathcal{E}$ and for all $u_m \in \mathcal{U}_m$.

The low-level control policy (10) guarantees that the difference between the evolution of the nonlinear system (1) and the LTV model (5) is bounded. Indeed, when Assumption 1 is satisfied, the CLF-CBF QP (9) guarantees invariance of the safe set (8) for all $t \in (iT, (i+1)T)$ and $i \in \mathbb{Z}_{0+}$, as discussed in Section IV. Next, we show how to design a mid-level planner which leverages the safe set $\mathcal{E}$ from (8).

B. Mid-Level Planning

In this section we describe the mid-level planning strategy. At this level of abstraction, we assume that we are given a goal grid cell where we would like to steer the system. Afterwards, we compute a reference state-input trajectory using a Model Predictive Controller (MPC), which leverages a simplified model and the tracking error bounds from the previous section.

Grid Model: Given the state $x(t) = [p^\top(t), q^\top(t)]^\top$, we define the current grid cell $\mathcal{C}^k_{\text{curr}}$, which contains the nonlinear system (1) for time $t \in [t^k, t^{k+1})$, i.e.,

$$p(t) \in \mathcal{C}^k_{\text{curr}} \subset \mathcal{X}_p,\ \forall t \in [t^k, t^{k+1}). \tag{11}$$

Similarly, we define the goal cell $\mathcal{C}^k_{\text{goal}}$, which represents the region where we want to steer the system for time $t \in [t^k, t^{k+1})$. Finally, we introduce the goal equilibrium sets $\mathcal{X}^k_{\text{curr}}$ and $\mathcal{X}^k_{\text{goal}}$, which collect the unforced equilibrium states that are contained into $\mathcal{C}^k_{\text{curr}}$ and $\mathcal{C}^k_{\text{goal}}$, i.e., for $i \in \{\text{curr}, \text{goal}\}$

$$\mathcal{X}^k_i = \{x = [p,q] \in \mathbb{R}^n | p \in \mathcal{C}^k_i, \dot{x} = f(x) = 0\} \subset \mathbb{R}^n. \tag{12}$$

Throughout this section, we assume that t^k, $\mathcal{X}^k_{\text{goal}}$, $\mathcal{C}^k_{\text{curr}}$ and $\mathcal{C}^k_{\text{goal}}$ are given by the high-level planner and we synthesize a controller to drive the system from the current cell $\mathcal{C}^k_{\text{curr}}$ to the goal cell $\mathcal{C}^k_{\text{goal}}$.

Model Predictive Control: We design a Model Predictive Controller (MPC) to compute the mid-level input $u_m(t)$ that defines the evolution of the reference trajectory (5) and to define the return map $\Delta_{\bar{x}}$ for the LTV model (5). The MPC problem is solved at 1/T Hertz and therefore the reference mid-level input is piecewise constant, i.e.,

$$\dot{u}_m(t) = 0\ \forall t \in \mathcal{T} = \cup_{k=0}^{\infty}(kT, (k+1)T).$$

Next, we introduce the following discrete time linear model:

$$\bar{x}^d\big((i+1)T\big) = \bar{A}_i \bar{x}^d\big(iT\big) + \bar{B}_i u\big(iT\big), \tag{13}$$

where the transition matrices are

$$\bar{A}_i = e^{A_i T} \text{ and } \bar{B}_i = \int_0^T e^{A_i(T-\eta)} B_i d\eta.$$

for all $i \in \mathbb{Z}_{0+}$. Now notice that, as the mid-level input u_m is piecewise constant, if at time $t_i = iT$ the state $\bar{x}(iT) = \bar{x}^+(iT) = \bar{x}^d(iT)$, then at time $t_{i+1} = (i+1)T$ we have that

$$\bar{x}^-((i+1)T) = \bar{x}^d((i+1)T). \tag{14}$$

Given the discrete time model (13), at time $t_i = iT \in \mathcal{T}^c$ we solve the following finite time optimal control problem:

$$\begin{aligned} & J(x(iT), N) = \\ & \min_{\boldsymbol{v}_t, x^d_{i|i}}\ ||x^d_{i|i} - x(iT)||_{Q_e} + \sum_{t=i}^{i+N-1} h\big(x^d_{t|i}, v_{t|i}\big) \\ & \qquad\qquad + ||p^d_{i+N|i} - p^k_{\text{goal}}||_{Q_f} \\ & \text{s.t.}\ \ x^d_{t+1|i} = \bar{A}_t x^d_{t|i} + \bar{B}_t v^d_{t|i} \\ & \qquad x^d_{t|i} = \begin{bmatrix} p^d_{t+1|i} \\ q^d_{t+1|i} \end{bmatrix} \in \mathcal{X}^k_{p,q} \ominus \mathcal{E},\ v^d_{t|i} \in \mathcal{U}_m \\ & \qquad x^d_{i|i} - x(iT) \in \mathcal{E} \\ & \qquad x^d_{i+N|i} \in \mathcal{X}^k_{\text{goal}} \ominus \mathcal{E}_p, \forall t = \{i, \dots, i+N-1\} \end{aligned} \tag{15}$$

where $\mathcal{E}$ is defined in (8), $||p||_Q = p^\top Q p$,

$$\mathcal{X}^k_{p,q} = \left\{ x = \begin{bmatrix} p \\ q \end{bmatrix} \in \mathbb{R}^n | p \in \mathcal{C}^k_{\text{curr}} \cup \mathcal{C}^k_{\text{goal}} \text{ and } q \in \mathcal{X}_q \right\} \tag{16}$$

and

$$\mathcal{E}_p = \left\{ e = \begin{bmatrix} e_p \\ 0 \end{bmatrix} \in \mathbb{R}^n | \exists e_q \in \mathbb{R}^{n_p} \text{ and } \begin{bmatrix} e_p \\ e_q \end{bmatrix} \in \mathcal{E} \right\}. \tag{17}$$

Notice that the MPC problem (15) is designed based on the time-varying components $\mathcal{X}^k_{\text{goal}}, \mathcal{C}^k_{\text{curr}}, \mathcal{C}^k_{\text{goal}}, p^k_{\text{goal}}$ which are given by the high-level decision maker, as shown in Figure 2. Problem (15) computes a sequence of open loop actions $\boldsymbol{v}^d_t = [v^d_{t|t}, \dots, v^d_{t+N|t}]$ and an initial condition $x^d_{i|i}$ such that the predicted trajectory steers the system to the terminal set $\mathcal{X}^k_{\text{goal}}$, while minimizing the cost and satisfying state and input constraints. Let $\boldsymbol{v}^{d,*}_t = [v^{d,*}_{t|t}, \dots, v^{d,*}_{t+N|t}]$ be the optimal solution and $[x^{d,*}_{t|t}, \dots, x^{d,*}_{t+N|t}]$ the associated optimal trajectory, then the mid-level policy is

$$\Pi_m : \begin{cases} u_m(t) = \pi_m\big(x(t), N\big) = v^{d,*}_{t|t} & t \in \mathcal{T}^c \\ \dot{u}_m(t) = 0 & t \in \mathcal{T} \end{cases} \tag{18}$$

Finally, we define the return map from the LTV model (5) as follows:

$$\Delta_{\bar{x}}(x(t)) = x^{d,*}_{t|t}. \tag{19}$$

Assumption 2. Consider the equilibrium set $\mathcal{X}^k_{\text{curr}}$ defined in Equation (12). For all states $x(t) \in \mathcal{X}^k_{\text{curr}} \oplus \mathcal{E}$ Problem (15) is feasible with horizon N.

The above assumption is satisfied when any equilibrium state $\bar{x} \in \mathcal{X}^k_{\text{curr}}$ of the discrete time system (13) can be steered to the goal equilibrium set $\mathcal{X}^k_{\text{goal}}$ in at most N time steps. More formally, Assumption 2 holds when, for the discrete time system (13), $\mathcal{X}^k_{\text{goal}}$ is N-step backward reachable from the set $\mathcal{X}^k_{\text{curr}}$.

In Section IV, we will show that when the nonlinear system (1) and the LTV system (5) are in closed-loop with the low-level policy (10) and the mid-level policy (18), then state and input constraints (2) are satisfied for system (1). Furthermore, the nonlinear system (1) is steered from the current cell $\mathcal{C}^k_{\text{curr}}$ to the goal cell $\mathcal{C}^k_{\text{goal}}$ in finite time.

C. High Level Decision Making

In this section, we first describe how to compute a control policy which maximizes the probability of satisfying the specifications. Afterwards, we show how to compute the time-varying components p^k_{goal}, $\mathcal{C}^k_{\text{curr}}$, $\mathcal{C}^k_{\text{goal}}$ and $\mathcal{X}^k_{\text{goal}}$ used in the MPC problem (15).

Belief Model: For the MOMDP from Section II, we have that the environment state z_k is not perfectly observed. Therefore as in [38], we introduce the belief space $\mathcal{B} = \{b \in \mathbb{R}^{|\mathcal{Z}|} : \sum_{z=1}^{|\mathcal{Z}|} b^{(z)} = 1\}$ and the belief state $b_k \in \mathcal{B}$, which represents the posterior probability that the partially observable state z_k equals $z \in \mathcal{Z}$, i.e., $b_k = [b_k^{(1)}, \dots, b_k^{(|\mathcal{Z}|)}]$ with

$$b_k^{(z)} = \mathbb{P}(z_k = z | \boldsymbol{o}_k, \boldsymbol{s}_k, \boldsymbol{a}_{k-1}), \ \forall z \in \{1 \dots, |\mathcal{Z}|\}$$

where at time k the observation vector $\boldsymbol{o}_k = [o_0, \dots, o_k]$, the observable state vector $\boldsymbol{s}_k = [s_0, \dots, s_k]$ and the actions vector $\boldsymbol{a}_{k-1} = [a_0, \dots, a_{k-1}]$. The belief is a sufficient statistic and, for all $z' \in \mathcal{Z}$, it evolves accordingly to the following update equation:

$$b_{k+1}^{(z')} = \eta O(s_{k+1}, z', a_k, o_k) \times \sum_{z \in \mathcal{Z}} T_s(s_k, z, a_k, s_{k+1}) T_z(s_k, z, a_k, s_{k+1}, z_{k+1}) b_k^{(z)}$$

where η is a normalization constant [38]. Notice that the above update equation can be written in a compact form, i.e.,

$$b_{k+1} = f_b(s_{k+1}, s_k, o_k, a_k, b_k), \tag{20}$$

where $f_b : \mathcal{S} \times \mathcal{S} \times \mathcal{O} \times \mathcal{A} \times \mathcal{B} \to \mathcal{B}$. Finally, given the belief b_k, we introduce the maximum likelihood environment state estimate:

$$\hat{z}_k = \underset{z \in \mathcal{Z}}{\text{argmax}} \, \mathbb{P}(z_k = z | \boldsymbol{o}_k, \boldsymbol{s}_k, \boldsymbol{a}_{k-1}) = \underset{z \in \mathcal{Z}}{\text{argmax}} \ b^{(z)}. \tag{21}$$

Quantitative Control Policy: At the highest level of abstraction our goal is to compute a control policy π_h, which maximizes the probability that the high-level trajectory $\boldsymbol{\omega}$ satisfies the specifications ψ. Such control control policy can be computed solving the following quantitative problem:

$$\pi_h = \underset{\pi}{\text{argmax}} \, \mathbb{P}^{\pi}[\boldsymbol{\omega} \models \psi], \tag{22}$$

Algorithm 1: Update High-Level

1 **inputs:** $x(t)$, o_k, s_{k-1}, a_{k-1}, b_{k-1} ;
2 set current high-level state $s_k = \texttt{getState}(x(t))$;
3 compute current set $\mathcal{C}^k_{\text{curr}} = \texttt{getCell}(s_k)$;
4 update belief b_k using (20) ;
5 compute high-level action $a_k = \pi_h(s_k, b_k)$;
6 compute maximum likely estimate $\hat{z}_k$ using (21);
7 update state $s_{k+1} = f_s(s_k, \hat{z}_k, a_k)$;
8 compute goal set $\mathcal{C}^k_{\text{goal}} = \texttt{getCell}(s_{k+1})$;
9 computed the forecasted action $\hat{a} = \pi_h(s_{k+1}, b_k)$;
10 compute the forecasted state $\hat{s}_{k+2} = f_s(s_{k+1}, \hat{z}_k, \hat{a})$;
11 set forecasted set $\mathcal{C}^k_{\text{forc}} = \texttt{getCell}(s_{k+1})$;
12 get forecasted cell center $c^{\text{forc}} = \texttt{getCenter}(\mathcal{C}^k_{\text{forc}})$;
13 compute goal position $p^k_{\text{goal}} = \text{Proj}(c^{\text{forc}}, \mathcal{C}^k_{\text{goal}})$;
14 **return:** a_k, b_k, s_k, $\mathcal{C}^k_{\text{goal}}$, $\mathcal{C}^k_{\text{curr}}$, p^k_{goal}

where $\mathbb{P}^{\pi}[\boldsymbol{\omega} \models \psi]$ represents the probability that the specification ψ is satisfied for the closed-loop trajectory $\boldsymbol{\omega}$ under the policy π. The solution to the above qualitative problem can be approximated using point-based and simulation-based strategies [5], [6], [15]–[17]. The resulting high-level control policy maps the high-level state s_k and the environment belief b_k to the high-level control action a_k, i.e.,

$$a_k = \pi_h(s_k, b_k). \tag{23}$$

The high-level policy (22) is leveraged in Algorithm 1 to compute the goal position p^k_{goal} and the sets $\mathcal{C}^k_{\text{curr}}$ and $\mathcal{C}^k_{\text{goal}}$, which are used in the MPC problem (15). In Algorithm 1, we first use the function `getState`, which maps the current state $x(t)$ to the high-level state s_k representing the cell containing the nonlinear system 1 (line 2). Then, we compute the current cell $\mathcal{C}^k_{\text{curr}}$ associated with the high-level state s_k using the function `getCell` (line 3). Afterwards, we update the belief state b_k and we compute the control action a_k (lines $4-5$). Given the control action a_k and the maximum likelihood estimator of the environment state $\hat{z}_k$, we update the high-level state and we compute the goal cell $\mathcal{C}^k_{\text{goal}}$ (lines $6-8$). Next, given the current belief b_k, we forecast the action $\hat{a}$ and state $\hat{s}_{k+2}$ (lines $9-10$). These quantities are used to compute the forcasted cell center $c^{\text{forc}} \in \mathbb{R}^{n_p}$ and the forecasted cell $\mathcal{C}^k_{\text{forc}}$ associated with the forecasted state $\hat{s}_{k+2}$ (lines $11-12$).

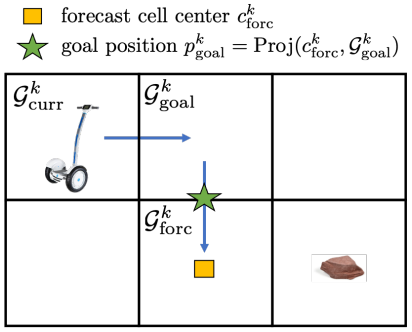

Fig. 3. The above figure illustrated the high-level updated from Algorithm 1.

Finally, the goal cell $\mathcal{C}^k_{\text{goal}}$ and the forecasted center $c^{\text{forc}} \in \mathbb{R}^{n_p}$ are used to compute the goal position p^{goal} (line 13).

Figure 3 illustrates Algorithm 1. In this example, the Segway is located in the top left corner of the grid and the current high-level action a_k is to move east. The figure also shows the forecasted action $\hat{a}$ that the Segway would take from the goal region, if the belief b_k is not updated. Basically, $\hat{a}$ is a high-level open-loop prediction of the future control action and it is used to incorporate forecast into the high-level decision maker. Indeed, the goal position p^k_{goal} is computed projecting the forecasted cell center c^{forc} onto the goal cell $\mathcal{C}^k_{\text{goal}}$.

D. Control Architecture

Finally, we introduce the multi-rate hierarchical control architecture which leverages the low-level, mid-level and high-level control policies from the previous sections. The multi-rate control Algorithm 2 details the architecture depicted in Figure 2. When the nonlinear system (1) reaches the goal cell (i.e., $p(t) \in \mathcal{C}^k_{\text{goal}}$), the high-level decision maker reads the new observations o_{k+1} and updates high-level state, action, goal position p^k_{goal}, goal cell $\mathcal{C}^k_{\text{goal}}$ and current cell $\mathcal{C}^k_{\text{curr}}$ (lines $3-4$). Finally, it updates the high-level time k and it initializes the MPC horizon $N^k_i = N$. Afterwards, the mid-level planner (lines $8-20$) updates the mid-level time counter i and the planned trajectory at a constant frequency of $1/T$ Hertz. First, it solves the MPC problem (15) with $N = N^k_i$ and time-varying components $\mathcal{X}^k_{\text{goal}}$, $\mathcal{C}^k_{\text{goal}}, \mathcal{C}^k_{\text{curr}}$ and p^k_{goal}. If the MPC problem is not feasible, the planner computes a contingency plan (lines $10-14$), otherwise it updates the prediction horizon (lines 15-16). Finally, Algorithm 2 computes the low-level control action solving the CLF-CBF QP (9) and the total control input $u(t) = u_l(t) + u_m(t)$.

Algorithm 2: Multi-Rate Control

1 **inputs:** k, s_k, b_k, a_k, i, $x(t)$, $u_m(t)$, $\bar{x}(t)$, $\mathcal{C}^k_{\text{curr}}$, $\mathcal{C}^k_{\text{goal}}$, p^k_{goal}, N^k_i, $\mathcal{C}^{k-1}_{\text{curr}}$, $\mathcal{C}^{k-1}_{\text{curr}}$, p^{k-1}_{goal}, N^{k-1}_i ;
2 **if** $q(t) \in \mathcal{C}^k_{goal}$ *or* $k = 0$ **then**
`// Update high-level goal`
3 measure o_{k+1} ;
4 update a_{k+1},b_{k+1}, s_{k+1}, $\mathcal{X}^k_{\text{goal}}$, $\mathcal{C}^{k+1}_{\text{goal}}$, $\mathcal{C}^{k+1}_{\text{curr}}$, p^{k+1}_{goal} using Algorithm 1 with $x(t)$, o_{k+1}, s_k, a_k, b_k ;
5 set $N^{k+1}_i = N$;
6 $k = k + 1$;
7 **end**
8 **if** $t \in \mathcal{T}^c = \cup_{j=0}^{\infty}\{jT\}$ **then**
`// Update mid-level plan`
9 solve MPC problem (15) with $N = N^k_i$, and $\mathcal{X}^k_{\text{goal}}$, $\mathcal{C}^k_{\text{curr}}$, $\mathcal{C}^k_{\text{goal}}$, p^k_{goal} ;
10 **if** *the MPC problem* (15) *is not feasible* **then**
11 solve MPC problem (15) with $N = N^{k-1}_i$, and $\mathcal{X}^{\text{goal}}_{k-1}$, $\mathcal{C}^{k-1}_{\text{curr}}$, $\mathcal{C}^{k-1}_{\text{goal}}$, p^{k-1}_{goal} ;
12 set $N^{k-1}_{i+1} = \max(1, N^{k-1}_i - 1)$;
13 set $N^k_{i+1} = N^k_i$;
14 **else**
15 set $N^{k-1}_{i+1} = N^{k-1}_i$;
16 set $N^k_{i+1} = \max(1, N^k_i - 1)$;
17 **end**
18 set $u_m(t) = v^{d,*}_{t|t} + K(x(t) - \bar{x}^{d,*}_{t|t})$;
19 update $\bar{x}(t) = \Delta_{\bar{x}}(x(t)) = \bar{x}^{d,*}_{t|t}$;
20 $i = i + 1$;
21 **end**
`// Compute low-level control`
22 solve the CBF problem (9) ;
23 Compute total input $u(t) = u_l(t) + u_m(t)$;
24 **Return:** $u(t)$, k, s_k, b_k, a_k, i, $x(t)$, $u_m(t)$, $\bar{x}(t)$, $\mathcal{C}^k_{\text{curr}}$, $\mathcal{C}^k_{\text{goal}}$, p^k_{goal}, N^k_i, N^{k-1}_i

IV. Safety and Performance Guarantees

In this section we show the properties of the proposed multi-rate control architecture. We consider the augmented system:

$$\Sigma_{\text{aug}} : \begin{cases} \dot{x}(t) = f\big(x(t)\big) + g\big(x(t)\big)\big(u_l(t) + u_m(t)\big), & t \geq 0 \\ \dot{\bar{x}}(t) = A_{\lfloor t/T \rfloor}\bar{x}(t) + B_{\lfloor t/T \rfloor}u_m(t), & t \in \mathcal{T} \\ \bar{x}^+(t) = \Delta_{\bar{x}}(x^-(t)), & t \in \mathcal{T}^c \end{cases} \tag{24}$$

where the nonlinear dynamics for state $x(t) \in \mathbb{R}^n$ are defined in (1) and the LTV model for the nominal state $\bar{x}(t) \in \mathbb{R}^n$ is defined in (5) for the reset map (19) given by the MPC. In what follows, we analyse the properties of the proposed multi-rate control Algorithm 2 in closed-loop with system (24). We show that the closed-loop system satisfies state and input constraints (2) and that the proposed algorithm maximizes the probability of satisfying the specifications. Notice that in practice the state $x(t)$ is given by the nonlinear system (1), whereas the nominal state $\bar{x}(t)$ is computed by the low-level layer to update the tracking error $e(t)$, as shown in Figure 2.

Proposition 1. *Consider the closed-loop system* (10) *and* (24) *with mid-level input* $u_m(t) \in \mathcal{U}_m$ *and* $\dot{u}_m(t) = 0, \forall t \in \mathcal{T}$. *If Assumption 1 holds and the error* $e(kT) = x(kT) - \bar{x}(kT) \in \mathcal{E}$ *for all* $k \in \mathbb{Z}_{0+}$, *then the control policy* (10) *guarantees that* $e(t) \in \mathcal{E}$ *and* $u_l(t) \in \mathcal{U}_l,\ \forall t \in [kT, (k+1)T)$.

Proof: The proof follows from standard CBF arguments [26]. First, we notice that the error $e(kT) = x(kT) - \bar{x}(kT)$ follows the error dynamics in (6). Furthermore, by construction the time-varying matrices $(A_{\lfloor t/T \rfloor}, B_{\lfloor t/T \rfloor})$ are constant for $t \in [kT, (k+1)T)$. Therefore, for all $k \in \mathbb{Z}_{0+}$ and $t \in [kT, (k+1)T)$, we have that error dynamics in (6) are nonlinear control affine for the low-level input u_l. This fact implies that, if at time $t = kT$ the error $e(kT) = x(kT) - \bar{x}(kT) \in \mathcal{E}$, then from the feasibility of the CLF-CBF QP (9) from Assumption 1 we have that $e(t) = x(t) - \bar{x}(t) \in \mathcal{E},\ \forall t \in [kT, (k+1)T)$. ■

Remark 2. We underline that Assumption 1 is satisfied for some $\alpha_1 \in \mathcal{K}^e$ and $\alpha_2 \in \mathcal{K}^e$ when the set $\mathcal{E}$ is robust control invariant for system (6) with $u_m(t) \in \mathcal{U}_m$ and mild assumptions on the Lie derivative of (6) hold (see [26] for further details). The set $\mathcal{E}$ may be hard to compute and standard techniques are based on HJB reachability analysis [9], SOS programming [30], Lyapunov-based methods [31] and Lipschitz properties of the system dynamics [34], [39].

Lemma 1 shows that between time $t_i = iT$ and $t_{i+1} = (i+1)T$ the difference between the state x and the state $\bar{x}$ of the reference trajectory is bounded. Next, we show that this property allows us to guarantee safety and convergence in finite time to the goal cell $\mathcal{C}^k_{\text{goal}}$ for the nonlinear system (1). In turns, convergence in finite time allows us to show that the high-level specifications are satisfied, when the following assumption holds.

Assumption 3. For the environment state sequence $\boldsymbol{z} = [z_0, z_1, \ldots]$, we have that for all $s \in \mathcal{S}$ there exists a high-level control policy $\kappa : \mathcal{S} \times \mathcal{B} \to \mathcal{A}$ such that the high-level trajectory $\boldsymbol{\omega}$ satisfies the specifications ψ with probability one.

Theorem 1. *Let Assumptions 1-3 hold and consider system* (24) *in closed-loop with Algorithm 2. If at time $t_i = iT$ the MPC problem* (15) *is feasible with $N^k_i = N$ and time-varying components $\mathcal{X}^k_{goal}, \mathcal{C}^k_{curr}, \mathcal{C}^k_{goal}$ and p^k_{goal}, then there exists a $j \in \{i, \ldots, i+N-1\}$ such that the closed-loop system satisfies state and input constraints* (2) *for all $t \in \{iT, \ldots, jT\}$ and the state $x((j+1)T) = [p^\top((j+1)T), q^\top((j+1)T)]^\top$ reaches the goal cell $\mathcal{C}^k_{goal}$, i.e., $p((j+1)T) \in \mathcal{C}^k_{goal}$.*

Proof: First, we show that Algorithm 1 returns a goal cell $\mathcal{C}^k_{\text{goal}}$ which is contained in the feasible set $\mathcal{X}_p$. From Assumption 3 we have that, for all $s \in \mathcal{S}$ and the environment state sequence $\boldsymbol{z} = [z_0, z_1, \ldots]$, there exists a policy which satisfies the specifications with probability one. Consequently the high-level policy (22), which maximizes the probability of satisfying the specifications, takes an high-level action a_k which avoids collision, i.e.,

$$\mathcal{C}^k_{\text{goal}} \subset \mathcal{X}_p. \tag{25}$$

Next, we show that if at time $t_i = iT$ the MPC problem (15) is feasible with $\mathcal{X}^k_{\text{goal}}, \mathcal{C}^k_{\text{goal}}, \mathcal{C}^k_{\text{curr}}, p^k_{\text{goal}}$ and $N^k_i > 1$, then at time $t_{i+1} = (i+1)T$ the MPC problem (15) is feasible with $\mathcal{X}^k_{\text{goal}}, \mathcal{C}^k_{\text{goal}}, \mathcal{C}^k_{\text{curr}}, p^k_{\text{goal}}$ and $N^k_{i+1} = N^k_i - 1$. Let

$$[x^{d,*}_{i|i}, x^{d,*}_{i+1|i}, \ldots, x^{d,*}_{i+N^k_i|i}] \text{ and } [u^{d,*}_{i|i}, \ldots, u^{d,*}_{i+N^k_i-1|i}]$$

be the optimal state input sequence to the MPC problem (15) at time $t_i = iT$. Then, from Lemma 1, equation (14) and the definition of the return map (19), we have that

$$x((i+1)T) - \bar{x}^{d,*}_{i+1|i} = x((i+1)T) - \bar{x}((i+1)T) \in \mathcal{E} \tag{26}$$

and therefore, by standard MPC arguments, the following sequences of $N^k_i - 1$ states and $N^k_i - 2$ inputs

$$[x^{d,*}_{i+1|i}, \ldots, x^{d,*}_{i+N^k_i|i}] \text{ and } [u^{d,*}_{i+1|i}, \ldots, u^{d,*}_{i+N^k_i-1|i}] \tag{27}$$

are feasible at time $t_{i+1} = (i+1)T$ for the MPC problem (15) with $\mathcal{X}^k_{\text{goal}}, \mathcal{C}^k_{\text{goal}}, \mathcal{C}^k_{\text{curr}}, p^k_{\text{goal}}$ and $N^k_{i+1} = N^k_i - 1$.

Now, we show that state and input constraints are satisfied until the system reaches the goal set $\mathcal{C}^k_{\text{goal}}$. Recall that by assumption the MPC problem is feasible at time $t_i = iT$ with $\mathcal{X}^k_{\text{goal}}, \mathcal{C}^k_{\text{goal}}, \mathcal{C}^k_{\text{curr}}, p^k_{\text{goal}}, N_i = N$ and assume that $p(jT) \notin \mathcal{C}^k_{\text{goal}}$ for all $j \in \{i, \ldots, i+N-1\}$. By induction the MPC problem (15) with $\mathcal{X}^k_{\text{goal}}, \mathcal{C}^k_{\text{goal}}, \mathcal{C}^k_{\text{curr}}, p^k_{\text{goal}}$ and $N^k_j = N^k_i - j$ is feasible for all $j \in \{i, \ldots, i+N-1\}$. Consequently, Algorithms 1 returns a feasible mid-level control action[1] $u_m(t) \in \mathcal{U}_m$. Furthermore, from Lemma 1 we have the low-level controller returns a feasible control action $u_l(t) \in \mathcal{U}_l$ and therefore

$$u(t) = u_l(t) + u_m(t) \in \mathcal{U}_l \oplus \mathcal{U}_m = \mathcal{U}, \forall t \in \mathbb{R}_{0+}. \tag{28}$$

The feasibility of the state-input sequences in (27) for the MPC problem solved with $\mathcal{X}^k_{\text{goal}}, \mathcal{C}^k_{\text{goal}}, \mathcal{C}^k_{\text{curr}}, p^k_{\text{goal}}$ implies that

$$\begin{aligned} &x^{d,*}_{j|j} \in \mathcal{X}^k_{p,q} \ominus \mathcal{E} \\ &x(jT) - x^{d,*}_{j|j} \in \mathcal{E}, \end{aligned} \tag{29}$$

$\forall j \in \{i, \ldots, i+N-1\}$. Consequently, from the above equation and definition (16), we have that

$$p(jT) \in \mathcal{X}_p \text{ and } q(jT) \in \mathcal{X}_q, \forall j \in \{i, \ldots, i+N-1\}.$$

Finally, we show that the state $x(t)$ of the augmented system (24) in closed-loop with Algorithm 2 converges to the goal cell $\mathcal{C}^k_{\text{goal}}$ in finite time. We have shown that, if $p(jT) \notin \mathcal{C}^k_{\text{goal}}$ for all $j \in \{i, \ldots, i+N-1\}$, then the MPC problem is feasible for all time $t_k = kT$ and $k \in \{i, \ldots, i+N-1\}$. Now we notice that by feasibility of the MPC problem at time $t_{i+N-1} = (i+N-1)T$ with $N_{i+N-1} = 1$, we have that the optimal planned trajectory satisfies

$$x^{d,*}_{i+N|i+N-1} \in \mathcal{X}^k_{\text{goal}} \ominus \mathcal{E}_p.$$

From Lemma 1, equation (14) and the definition of the return map (19), we have that

$$x((i+N)T) - \bar{x}^{d,*}_{i+N|i} = x((i+N)T) - \bar{x}((i+N)T) \in \mathcal{E}.$$

The above equation together with definition (17) imply that at time $t_{i+N} = (i+N)T$

$$x((i+N)T) = \begin{bmatrix} p((i+N)T) \\ q((i+N)T) \end{bmatrix} \in \mathcal{X}^k_{\text{goal}} \ominus \mathcal{E}_p \oplus \mathcal{E}$$

and therefore $p((i+N)T) \in \mathcal{C}^k_{\text{goal}}$.

Concluding, if for all time $t_j = jT$ and $j \in \{i, \ldots, i+N-1\}$ we have that $p(jT) \notin \mathcal{C}^k_{\text{goal}}$, then $p((i+N)T) \in \mathcal{C}^k_{\text{goal}}$. Thus, the closed-loop system converges to the goal cell $\mathcal{C}^k_{\text{goal}}$ in finite time. ∎

Finally, we leverage Theorem 1 to show that the multi-rate control Algorithm 2 steers the system in finite time to goal cell $\mathcal{C}^k_{\text{goal}}$ for all $k \in \mathbb{Z}_{0+}$ and, consequently, the closed-loop systems satisfies the high-level specifications when Assumption 3 is satisfied. In particular, we show that the contingency plan from lines 10-14 of Algorithm 2 guarantees feasibility of the planner when the time-varying components are updated.

Theorem 2. *Let Assumptions 1-3 hold and consider system* (24) *in closed-loop with Algorithm 2. If $x(0) \in \mathcal{X}^k_{curr} \oplus \mathcal{E}$, then the closed-loop system* (2) *and* (24) *satisfies the high-level specifications.*

Proof: The proof follows by induction. Assume that at time $t_i = iT$ the closed-loop system reaches the goal

[1]Note that as $p(jT) \notin \mathcal{C}^k_{\text{goal}}$ for all $j \in \{i, \ldots, i+N-1\}$ the MPC time-varying components are not updated.

cell $\mathcal{C}^k_{\text{goal}}$, i.e, $p(iT) \in \mathcal{C}^k_{\text{goal}}$. Then, at time $t_i = iT$ we have that the high-level decision maker from Algorithm 2 (lines 2-9) updates the high-level time and the time-varying components $\mathcal{X}^{k+1}_{\text{goal}}, \mathcal{C}^{k+1}_{\text{curr}}, \mathcal{C}^{k+1}_{\text{goal}}, p^{k+1}_{\text{goal}}$ used to design the MPC problem (15). After the high-level update, the MPC problem with $N = N^{k+1}_j$, $\mathcal{X}^{k+1}_{\text{goal}}, \mathcal{C}^{k+1}_{\text{curr}}, \mathcal{C}^{k+1}_{\text{goal}}$, and p^{k+1}_{goal} maybe either feasible or unfeasible[2]. Thus, we analyse the following three cases for $j \geq i$:

Case 1: The MPC problem with $\mathcal{C}^{k+1}_{\text{goal}}, \mathcal{C}^{k+1}_{\text{curr}}, p^{k+1}_{\text{goal}}$ and $N = N^{k+1}_j$ is feasible, therefore from Theorem 1 we have that Algorithm 2 steers the nonlinear system to the goal $\mathcal{C}^{k+1}_{\text{goal}}$.

Case 2: The MPC problem with $\mathcal{C}^{k+1}_{\text{goal}}, \mathcal{C}^{k+1}_{\text{curr}}, p^{k+1}_{\text{goal}}$ and $N = N^{k+1}_j$ is not feasible and $N^k_j = 1$. Then from Theorem 1, we have that the contingency MPC with N^k_j, $\mathcal{C}^k_{\text{goal}}, \mathcal{C}^k_{\text{curr}}$ and p^k_{goal} is feasible and Algorithm 1 returns a feasible control action. Furthermore, as $N^k_j = 1$ the terminal state of the optimal predicted trajectory is

$$x^{d,*}_{j+1|j} \in \mathcal{X}^k_{\text{goal}} \ominus \mathcal{E}_p.$$

The above equation together with equation (26) imply that

$$x((j+1)T) \in \mathcal{X}^k_{\text{goal}} \ominus \mathcal{E}_p \oplus \mathcal{E} \subset \mathcal{X}^k_{\text{goal}} \oplus \mathcal{E},$$

therefore from Assumption 2 we have that at the next time step $t_{j+1} = (j+1)T$ the MPC problem with $N^{k+1}_{j+1} = N$, $\mathcal{X}^{k+1}_{\text{goal}}, \mathcal{C}^{k+1}_{\text{goal}}, \mathcal{C}^{k+1}_{\text{curr}}$ and p^{k+1}_{goal} is feasible and, from Theorem 1, we have that Algorithm 2 steers the nonlinear system to the goal $\mathcal{C}^{k+1}_{\text{goal}}$ in finite time.

Case 3: The MPC problem with $\mathcal{C}^{k+1}_{\text{goal}}, \mathcal{C}^{k+1}_{\text{curr}}, p^{k+1}_{\text{goal}}$ and $N = N^{k+1}_j$ is not feasible and $N^k_j > 1$. Then from Theorem 1, we have that the contingency MPC with N^k_j, $\mathcal{C}^k_{\text{goal}}, \mathcal{C}^k_{\text{curr}}$ and p^k_{goal} is feasible.

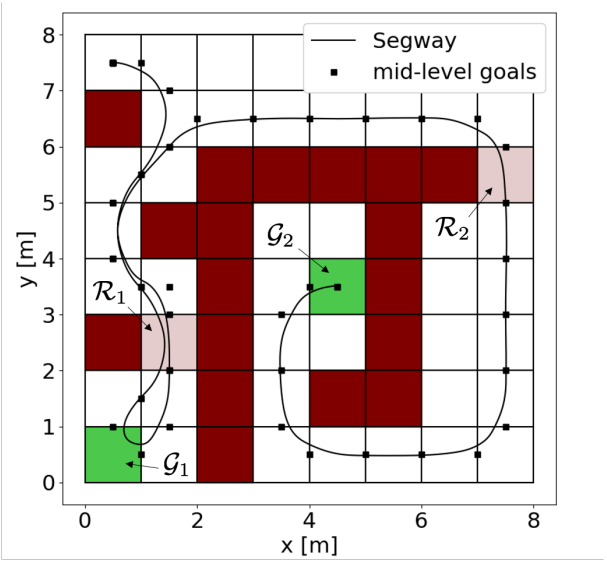

Fig. 4. Closed-loop trajectory. The Segway first explores regions $\mathcal{R}_1$, which is traversable, and $\mathcal{G}_1$ that does not contain the science sample. Afterwards, it explores the traversable region $\mathcal{R}_2$ and it reaches $\mathcal{G}_2$.

[2]Unfeasiblity may be caused by the update of $\mathcal{C}^{k+1}_{\text{goal}}$, $\mathcal{C}^{k+1}_{\text{curr}}$ and $\mathcal{X}^{k+1}_{\text{curr}}$.

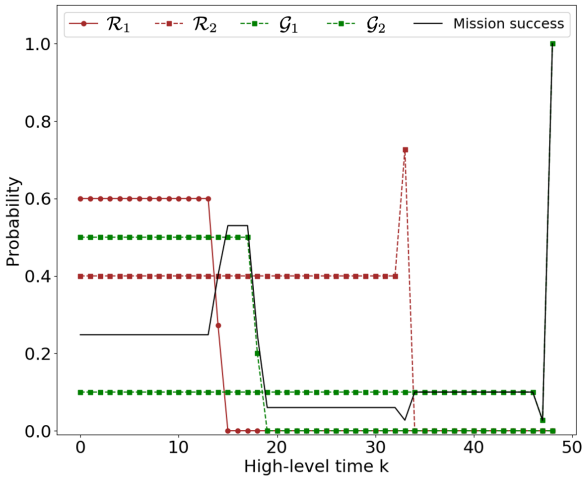

Fig. 5. This figure shows the closed-loop probability of mission success, which equals the probability of satisfying the high-level specifications. Furthermore, we reported also the belief about regions $\mathcal{R}_1$ and $\mathcal{R}_2$ being free about the goal regions $\mathcal{G}_1$ and $\mathcal{G}_2$ containing the science sample.

Concluding, we have that by assumption $x(0) \in \mathcal{X}^k_{\text{curr}} \oplus \mathcal{E}$, which from Assumption 2 implies that at time $t = 0$ the MPC is feasible and therefore by Theorem 1 Algorithm 2 steers system (1) to $\mathcal{G}^0_{\text{goal}}$. Afterwards, we have that as $N^k_{j+1} = N^k_j - 1$ Case 3 occurs at most N times until the conditions from Case 1 or Case 2 hold. Therefore, from Cases 1-2, we have that Algorithm 2 steers system (24) to the goal cell $\mathcal{C}^k_{\text{goal}}$ for all $k \in \mathbb{Z}_{0+}$. Consequently, as the high-level policy (22) maximizes the probability of satisfying the specifications and from Assumption 3 there exists a policy which completes the control task with probability one, we have that the closed-loop system satisfies the specifications. ■

V. Results

We tested the proposed strategy in simulation and experiment on navigation tasks inspired by the Mars exploration mission [5], [6], [14]. We control a Segway-like robot and our goal is to explore the environment to find science samples which may be located in known goal regions $\mathcal{G}_i$ with some probability. The high-level specification is $\psi = \neg\texttt{collision} U \texttt{sample}$, where the atomic proposition `collision` is true when the Segway is in a cell which is not traversable and the atomic proposition `sample` is `true` when the Segway is in a goal cell $\mathcal{G}_i$ which contains a science sample. While performing the search task, we have to collect observations to determine the state of the uncertain region $\mathcal{R}_i$, which may be traversable with some probability. The controller has access to only partial observations about the environment. In particular, the Segway receives a perfect observation about the state of the uncertain region $\mathcal{R}_i$ when one cell away, an observation which is correct with probability 0.8, when the Manhattan distance is smaller than two, and an uninformative observations otherwise. Similarly, the Segway receives a partial observation about the goal region $\mathcal{G}_i$ which is

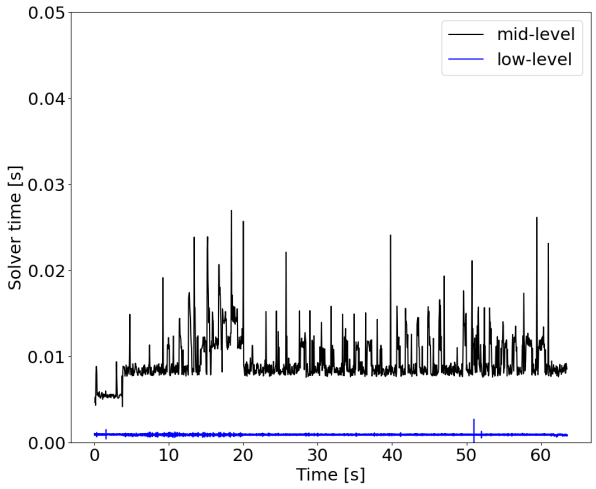

Fig. 6. This figure shows the computational time associated with middle and low layers. It takes on average 12 ms to compute the mid-level control actions and less than 1 ms to compute the low-level commands. In this example the middle layer is discretized at 20 Hz and the lowest level at 1 kHz.

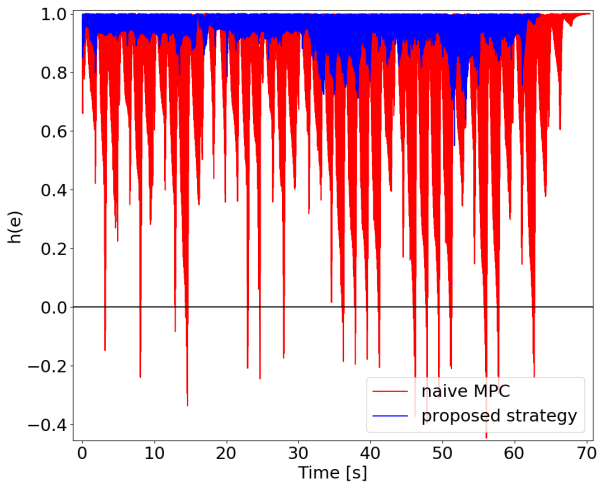

Fig. 7. Comparison between the barrier function associated with the proposed strategy and a naive strategy MPC which is based on the linearized dynamics. As shown in the figure, when the low-level controller is not used the difference between the planner trajectory and the MPC trajectory grows and, as a results, the barrier function (30) becomes negative.

correct with probability 0.7, when one cell away and a perfect observations when the goal cell $\mathcal{G}_i$ is reached.

The state of the Segway is $x = [X, Y, \theta, v, \dot{\theta}, \psi, \dot{\psi}]$, where $(X-Y)$ represents the position of the center of mass, $(\theta, \dot{\theta})$ the heading angle and yaw rate, v the velocity and $(\psi, \dot{\psi})$ the rod's angle and angular velocity. The control input $u = [T_l, T_r]$, where T_l and T_r are the torques to the left and right wheel motors, respectively. In order to implement the low-level CLF-CBF QP we used the following function:

$$h(e) = 1 - ||\text{diag}(v_h)(x - \bar{x})||_2^2 \tag{30}$$

where $v_h = [1/0.02, 1/0.02, 1/0.1, 1/0.1, 1/0.3, 1/0.1, 1/0.3]$ and $\bar{x} = [\bar{X}, \bar{Y}, \bar{\theta}, \bar{v}, \dot{\bar{\theta}}, \bar{\psi}, \dot{\bar{\psi}}]$ represents the state of the nominal system from (5). The candidate control Lyapunov function is

$$V(e) = ||\text{diag}(v_v)(x - \bar{x})||_2^2$$

where $v_v = [100, 100, 100, 100, 10000, 10000, 100]$ and in the CLF-QBF QP (9) we used $c_1 = 1$, $c_2 = 10$ and $\alpha_2(x) = x$. The planning model (5) is computed iteratively linearizing the Segway dynamics around the predicted MPC trajectory. This strategy is standard in MPC, for more details on the linearization strategy please refer to [40]. The stage cost $h(x, u) = x^\top Q u + u^\top R u$ and the tuning matrices are $Q = \text{dial}(0.1, 0.1, 0, 0, 10, 1, 10)$, $R = \text{diag}(0.01, 0.01)$ and $Q_f = \text{diag}(100, 100)$. Furthermore, we added an input rate cost with penalty $Q_{\text{rate}} = 0.1$ and a slack variable for the terminal constraint on the state $q_{t+N|i}$ with weight $Q_{\text{slack}} = \text{diag}(100, 100, 100)$. Finally, we approximated $\mathcal{S} = \{e = x - \bar{x} \in \mathbb{R}^n : h(e) \geq 0\} = \{e = x - \bar{x} \in \mathbb{R}^n : ||\text{diag}(v_v)(x - \bar{x})||_2^2 \leq 1\}$ with $\bar{\mathcal{S}} = \{e = x - \bar{\in}\mathbb{R}^n : ||\text{diag}(v_v)(x - \bar{x})||_\infty \leq 1\}$. This strategy allows us to write the MPC problem (15) as a QP[3], which we solved using OSQP [41].

A. *Simulation*

We implemented the proposed strategy in our high-fidelity Robotic Operating System (ROS) simulator. Figure 4 shows the locations of the uncertain and goal regions. The code can be found at `https://github.com/DrewSingletary/segway_sim`, please check the `REAME.md` to replicate our results. In this example the goal regions $\mathcal{G}_1$ and $\mathcal{G}_2$ may contain a science sample with probability 0.6 and 0.4, respectively. Whereas, regions $\mathcal{R}_1$ and $\mathcal{R}_2$ may be traversable with probability 0.5 and 0.1, as shown in Figure 5.

Figure 4 shows the closed-loop trajectory of the Segway. We notice that the controller explores the uncertain region $\mathcal{R}_1$, which in this example is traversable and afterwards it reaches the goal regions $\mathcal{G}_1$. As shown in Figure 5, at the high-level time $k = 19$ the controller figures out that the goal cell $\mathcal{G}_1$ does not contain a science sample and, consequently, the probability of mission success drops. Afterwards, the controller steers the Segway to the traversable region $\mathcal{R}_2$ and to the goal regions $\mathcal{G}_2$. In this example, the goal regions $\mathcal{G}_2$ contains a science sample and the mission is completed successfully, as shown in Figure 5.

The mid-level is discretized for $T = 50$ ms and the low-level at 1 kHz. Figure 6 shows the computational time associated with mid-level and low-level control actions. It takes on average 12 ms to compute the mid-level control action $u_m(t)$ and less than 1 ms to compute the low-level action $u_l(t)$.

Finally, we analyse the evolution of the barrier function (30), which quantities the difference between the trajectory $x(t)$ of

[3]Note that using $\mathcal{S}$ renders the MPC problem an SOCP, which is convex but computationally more demanding.

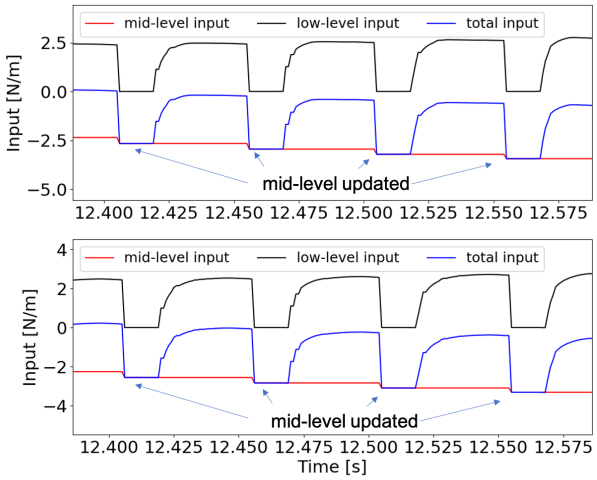

Fig. 8. Input torque sent to the right (top) and left (bottom) motor over a period of 0.2 second. The mid-level input is updated at 20 Hz, whereas the low-level action is updated at 1 kHz. Notice that the total input is the summation of the low and mid-level inputs.

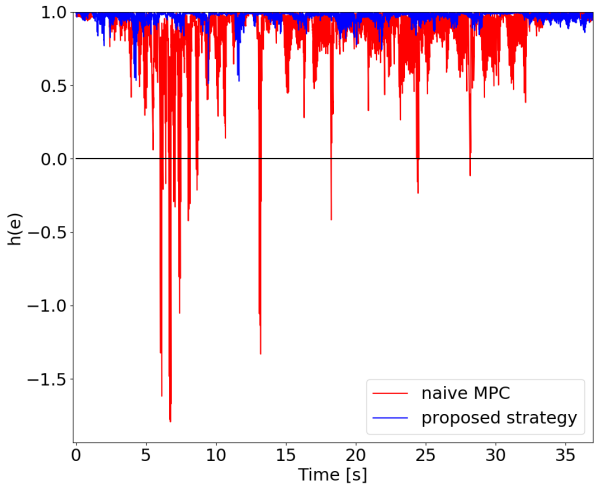

Fig. 9. Experimental comparison between the barrier function associated with the proposed strategy and a naive MPC which is based on the linearized dynamics. Also in this case, when the low-level controller is not used, the difference between the planner trajectory and the MPC trajectory grows and, as a results, the barrier function (30) becomes negative.

system (1) and the reference trajectory $\bar{x}(t)$ associated with nominal model[4] (5). We compared the proposed strategy with a naive MPC which is synthesized as in (15), but without taking into account the effect of the tracking error, i.e., we do not tighten the constraints and we set $x_{i|i} = x(t)$. Figure 7 shows the evolution of the barrier function for the proposed strategy and the naive MPC. We notice that when the low-level controller is not used, the barrier function becomes negative and in general has a lower magnitude. Therefore, this figure shows the advantage of the proposed hierarchical control architecture, where the low-level high-frequency controller is leveraged to track the reference trajectory. Indeed, this high-frequency feedback is used to modify the mid-level control actions, as shown in Figure 8. As discussed, the mid-level control action is updated at 20 Hz and the low-level input at 1 kHz. Notice that after the update of the mid-level input, the contribution of the low-level input towards the total control action is limited. However, as time progresses the linearization used to plan the reference trajectory is less and less accurate and for this reason, the magnitude of low-level controller increases.

B. Experiment

We implemented the proposed multi-rate hierarchical control strategy on the Segway-like robot shown in Figure 1. State estimation is based on wheel encoders and IMU data from a VectorNav VN-100. The state estimate and the low-level control action u_l are computed at 800 Hz on the Segway, which is equipped with an ARM Cortex-A57 (quad-core) @ 2 GHz CPU running the ERIKA3 RTOS. On the other hand, the mid-level planner discretized at 20 Hz and the high-level decision maker run on a desktop with an Intel Core i7-8700 CPU (6-cores) @ 3.7 GHz CPU, which sends the reference trajectory $\bar{x}$ and the reference input u_m via WiFi.

Figure 1 shows the location of the three uncertain regions $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$ which may be traversable with probability 0.9, 0.3 and 0.2, respectively. In this example, we assume that the goal region $\mathcal{G}_1$ contains the science sample with probability 1. Figure 10 shows the closed-loop trajectory. First, the controller explores region $\mathcal{R}_1$, which is not traversable and afterwards it steers the Segway towards regions $\mathcal{R}_2$ and $\mathcal{R}_3$. After collecting observations about the environment, the controller detects that region $\mathcal{R}_2$ is not traversable and that region $\mathcal{R}_3$ is free space that the Segway can navigate through to reach the goal region $\mathcal{G}_1$. A video of the experiment and comparison with a naive MPC can be found at `https://www.youtube.com/watch?v=Q-Mm0ywPh_I`.

Figure 9 shows the evolution of the control barrier function (30). We compare the proposed strategy with a naive

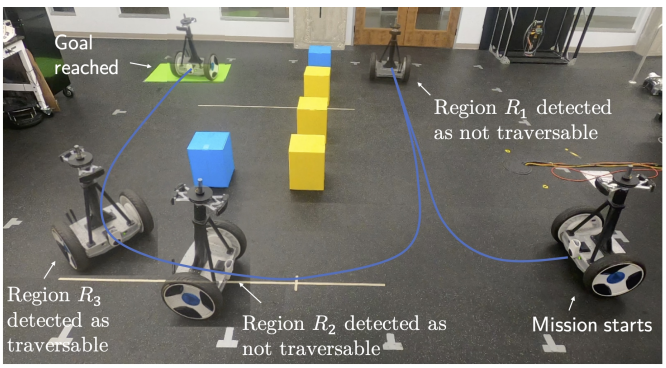

Fig. 10. Closed-loop trajectory during the experiment. The Segway first explores the uncertain regions $\mathcal{R}_1$, $\mathcal{R}_1$ and $\mathcal{R}_1$ and. afterwards it reaches the goal region.

[4]In this example the nominal model is computed iteratively linearizing the nonlinear dynamics

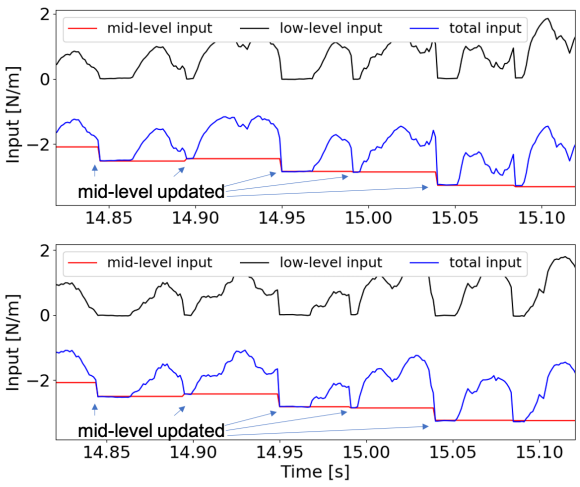

Fig. 11. Experimental results. Input torque sent to the right (top) and left (bottom) motor over a period of 0.3 seconds. The mid-level input is updated at 20 Hz, whereas the low-level action is updated at 800 Hz. Notice that the total input is the summation of the low and mid-level inputs.

MPC which is designed as in (15), but without robustifying the constraint sets and setting $x_{i|i} = x(t)$. Also in this case, when the high-frequency low-level controller is not active, the barrier function becomes negative meaning that the error e does not belong to the safe set $\mathcal{E}$, i.e., $e(t) \notin \mathcal{E}$ for all $t \in \mathbb{R}_{0+}$. This result highlights the importance of the low-level high-frequency feedback from the CLF-CBF QP, which compensates for the model mismatch at the planning layer. Indeed, the MPC planner uses a linearized and discretized model, which is a first order approximation of the true dynamics. This approximation is accurate at the discrete time instances when the MPC input is computed. For this reason, the low-level CLF-CBF QP tracking controller computes the high-frequency component $u_l(t)$ which corrects the mid-level piecewise constant input $u_m(t)$, as shown in Figure 11.

VI. Conclusions

In this paper we presented a multi-rate hierarchical control architecture for navigation tasks in partially observable environments. At the lowest level we leverage a CLF-CBF QP, which is used to track a reference trajectory within some error bounds. The reference trajectory is computed by a mid-level planner which leverages an MPC with time-varying terminal components. The feasibility of the MPC planner is guaranteed via a contingency scheme and a local reachability assumption on the planning model. Finally, at the highest level of abstraction, we showed how to model the system-environment interaction using a MOMDP and we proposed an algorithm to update the MPC time-varying components. The effectiveness of the proposed strategy is shown on navigation examples, where a Segway-like robot has to find science samples, while avoiding partially observable obstacles.

VII. Acknowledgements

The authors would like to thank Geoffroy le Courtois du Manoir for helping with the experiments.

References

[1] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[2] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, 2010, pp. 101–110.

[3] P. Tabuada and G. J. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[4] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000.

[5] S. Haesaert, R. Thakker, R. Nilsson, A. Agha-mohammadi, and R. M. Murray, "Temporal logic planning in uncertain environments with probabilistic roadmaps and belief spaces," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 6282–6287.

[6] S. Haesaert, P. Nilsson, C. I. Vasile, R. Thakker, A.-a. Agha-mohammadi, A. D. Ames, and R. M. Murray, "Temporal logic control of pomdps via label-based stochastic simulation relations," *IFAC-PapersOnLine*, vol. 51, no. 16, pp. 271–276, 2018.

[7] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *arXiv preprint arXiv:1809.06746*, 2018.

[8] Y. S. Shao, C. Chao, S. Kousik, and R. Vasudevan, "Reachability-based trajectory safeguard (rts): A safe and fast reinforcement learning safety layer for continuous control," *arXiv preprint arXiv:2011.08421*, 2020.

[9] 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 *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1517–1522.

[10] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, 1977, pp. 46–57.

[11] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multi-agent motion tasks based on ltl specifications," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 1. IEEE, 2004, pp. 153–158.

[12] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 4885–4890.

[13] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[14] P. Nilsson, S. Haesaert, R. Thakker, K. Otsu, C.-I. Vasile, A.-A. Agha-Mohammadi, R. M. Murray, and A. D. Ames, "Toward specification-guided active mars exploration for cooperative robot teams," *Robotics: Science and Systems (RSS)*, 2018.

[15] M. Bouton, J. Tumova, and M. J. Kochenderfer, "Point-based methods for model checking in partially observable markov decision processes." in *AAAI*, 2020, pp. 10 061–10 068.

[16] C.-I. Vasile, K. Leahy, E. Cristofalo, A. Jones, M. Schwager, and C. Belta, "Control in belief space with temporal logic specifications," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 7419–7424.

[17] Y. Wang, S. Chaudhuri, and L. E. Kavraki, "Bounded policy synthesis for pomdps with safe-reachability objectives," *arXiv preprint arXiv:1801.09780*, 2018.

[18] M. Ahmadi, R. Sharan, and J. W. Burdick, "Stochastic finite state control of pomdps with ltl specifications," *arXiv preprint arXiv:2001.07679*, 2020.

[19] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *International conference on computer aided verification*. Springer, 2011, pp. 585–591.

[20] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A storm is coming: A modern probabilistic model checker," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 592–600.

[21] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[22] E. J. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: Discounted costs," *Operations research*, vol. 26, no. 2, pp. 282–304, 1978.

[23] P. Poupart and C. Boutilier, "Bounded finite state controllers," in *NIPS*, 2003.

[24] J. Pineau, G. Gordon, S. Thrun *et al.*, "Point-based value iteration: An anytime algorithm for pomdps," in *IJCAI*, vol. 3, 2003, pp. 1025–1032.

[25] T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. Ames, "Towards a framework for realizable safety critical control through active set invariance," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 98–106.

[26] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, Aug 2017.

[27] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[28] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 7130–7135.

[29] H. Yin, M. Bujarbaruah, M. Arcak, and A. Packard, "Optimization based planner tracker design for safety guarantees," *arXiv preprint arXiv:1910.00782*, 2019.

[30] S. Singh, M. Chen, S. L. Herbert, C. J. Tomlin, and M. Pavone, "Robust tracking with model mismatch for fast and safe planning: an sos optimization approach," *arXiv preprint arXiv:1808.00649*, 2018.

[31] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5883–5890.

[32] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Vehicle System Dynamics*, vol. 52, no. 6, pp. 802–823, 2014.

[33] M. Kögel and R. Findeisen, "Discrete-time robust model predictive control for continuous-time nonlinear systems," in *2015 American Control Conference (ACC)*. IEEE, 2015, pp. 924–930.

[34] S. Yu, C. Maier, H. Chen, and F. Allgöwer, "Tube mpc scheme based on robust control invariant set with application to lipschitz nonlinear systems," *Systems & Control Letters*, vol. 62, no. 2, pp. 194–200, 2013.

[35] J. Köhler, R. Soloperto, M. A. Muller, and F. Allgower, "A computationally efficient robust model predictive control framework for uncertain nonlinear systems," *IEEE Transactions on Automatic Control*, 2020.

[36] U. Rosolia and A. D. Ames, "Multi-rate control design leveraging control barrier functions and model predictive control policies," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 1007–1012, 2021.

[37] D. Q. Mayne, M. M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[38] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee, "Planning under uncertainty for robotic tasks with mixed observability," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1053–1068, 2010.

[39] Y. Chen, H. Peng, J. Grizzle, and N. Ozay, "Data-driven computation of minimal robust control invariant set," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4052–4058.

[40] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: a predictive control approach," *IEEE Transactions on Control Systems Technology*, 2019.

[41] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, pp. 1–36, 2020.