MPC-CBF with Adaptive Safety Margins for Safety-critical Teleoperation over Imperfect Network Connections

Riccardo Periotto[†], Mina Ferizbegovic[‡], Fernando S. Barbosa[‡], and Roberto C. Sundin[‡]

Abstract— The paper focuses on the design of a control strategy for safety-critical remote teleoperation. The main goal is to make the controlled system track the desired velocity specified by an operator while avoiding obstacles despite communication delays. Control Barrier Functions (CBFs) are used to define the safety constraints that the system has to respect to avoid obstacles, while Model Predictive Control (MPC) provides the framework for adjusting the desired input, taking the constraints into account. The resulting input is sent to the remote system, where appropriate low-level velocity controllers translate it into system-specific commands. The main novelty of the paper is a method to make the CBFs robust against the uncertainties caused by the network delays affecting the system's state and do so in a less conservative manner. The results show how the proposed method successfully solves the safety-critical teleoperation problem, making the controlled systems avoid obstacles with different types of network delay. The controller has also been tested in simulation and on a real manipulator, demonstrating its general applicability when reliable low-level velocity controllers are available.

I. Introduction

Teleoperation is the technical term indicating the remote control of systems over long distances. Over the last decades, it has been successfully applied in various scenarios, including nuclear power plants, submarine and space missions, medical operations, and industrial procedures [1]. The number of applications in which it acts as a baseline is continuously growing. In particular, its integration with robotics has led to the development of a specialized field named *telerobotics*, which aims to merge the objectives of both disciplines [2].

In telerobotics, two fundamental elements to consider to ensure correct and reliable behavior of the system are *control* and *safety*, where the former is the one in charge of making the system achieve certain tasks, while the latter of respecting specific constraints [1], [3]. In recent years, the relevance of safety has grown considerably, particularly due to the increasing adoption of *autonomous systems*. By definition, autonomous systems are machines capable of performing tasks without human intervention. In this type of application, safety is no longer solely an operator's responsibility but becomes a requirement that the system has to fulfill by itself. Typical examples of safety requirements are those needed to comply with regulations, guarantee the protection of people working nearby, and avoid damaging the environment.

---

† Riccardo Periotto is with Agile Robots AG, Munich, Germany. ‡Mina Ferizbegovic, Fernando S. Barbosa, and Roberto C. Sundin are with Ericsson Research, Stockholm, Sweden. E-Mail: {mina.ferizbegovic, fernando.dos.santos.barbosa, roberto.castro.sundin}@ericsson.com
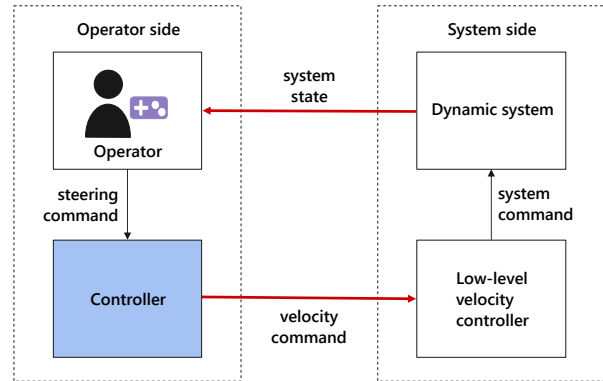
Fig. 1: **High-level problem architecture.** The two larger dashed boxes represent the two sides of the network. The controller executes on the same side as the operator controlling the system, while the system and the low-level controller are on the opposite side. The thicker red arrows represent the information shared between the two sides through the remote connection, while the thinner ones represent the information exchanged between elements operating in the same location. The highlighted block symbolizes the main focus of the paper, which is an optimization-based controller in charge of modifying the desired velocity specified by the operator to enhance the safety guarantees of the system.

The problem addressed in this work is the safety-critical teleoperation of dynamic systems. The high-level architecture showing the entities involved in this problem and their interactions is presented in Fig. 1, where an operator uses a steering device (e.g., a gamepad) to control the system. The signal of the steering device is encoded as a desired velocity and sent to a controller module in charge of refining it considering the safety constraints and sending it through a network connection. This controller is the part of the architecture on which this paper primarily focuses. Once the velocity command reaches the system's side, low-level controllers translate it to specific system's inputs. The operator guides the system based on the received representation of it. This can be, for example, a video stream captured with one or several cameras or a real-time rendering in dedicated software.

This paper concentrates on the challenges related to the corruption of information due to network delay over remote connections and proposes a method to account for the resulting system uncertainties. More precisely, we explore the effectiveness of approaches combining Model Predictive Control (MPC) and Control Barrier Functions (CBFs) for obstacle avoidance when the teleoperation is affected by communication delays. Our contributions are twofold: i) we provide a real-time capable controller based on MPC

and CBF that allows for online obstacle avoidance and; ii) propose a dynamic safety margin that adapts to network performance such that the system becomes resilient towards network disturbances. The adaptable safety margin makes the method less conservative than approaches based on upper bounds and worst-case scenarios. The paper is structured as follows: Section II contextualizes the content of the paper within the existing literature, while Section III presents the theory underlying our method. Section IV explains the method and Section V illustrates the results, together with how we collected them. Section VI concludes the paper.

II. RELATED WORK

The problem of controlling robots while avoiding obstacles is historically known. Offline trajectory planning is a possible method to address this problem, but it is only effective when both the environment and the path the system has to follow are known [2], [4]. In many real-world applications, the environment where the system operates and the desired trajectory are unknown in advance, making online solutions essential. Among the relevant and traditionally known online approaches, those based on haptic feedback and Artificial Potential Fields (APFs) play a significant role. Nonetheless, they do not provide the safety guarantees given by techniques such as CBFs [5], [6].

First introduced in [7] and later refined and popularized in [8]–[10], CBFs have now become a popular tool in the control literature. They are an extension of Barrier Functions (BFs) to systems with control inputs and can transform constraints defined on the system state to constraints on its inputs [11]. Multiple sources attribute their success to their ability to ensure safety, obstacle avoidance, and invariance properties, as well as their natural relationship with Lyapunov functions [9], [10], [12]. Indeed, just as Control Lyapunov Functions (CLFs) provide a mathematical framework for stability analysis, CBFs provide a complementary framework for safety. In many works, they act as a safety regulator by minimally modifying the input calculated by a stabilizing or tracking controller.

The main limitations of a standard application of CBFs such as the one presented in [9], [10], and [13], are that: they are defined in continuous time, they modify the input by only considering the information available without any predictive approach, and they do not take into account the uncertainties of the system's state. Recent approaches attempt to overcome these shortcomings by adopting mechanisms to: (i) perform in discrete time, (ii) consider the future behaviors of the system, and (iii) increase the tolerance against disturbances.

Of these mechanisms, the first has been adopted widely in the field, with several works also demonstrating its properties mathematically [14], [15].

The second mechanism can be implemented by considering a model of the system's dynamics within the optimization problem used to compute the input. In this regard, solutions based on MPC and their variants are increasingly being proposed, rather than more standard approaches based on Quadratic Programming (QP) [4], [16], [17]. However, a significant limitation that the proposed work share is the adoption of Euclidean norms to define the obstacle avoidance constraints, for example by requiring the distance between the robot and the obstacles to be larger than a given value [18]. The drawback of this approach is that it restricts the system's movement only when it is relatively close to the obstacles, technically, when the reachable set along the horizon intersects with the obstacles [18]. Consequently, to enable the agent to take proactive actions to avoid obstacles, a longer prediction horizon is required in these cases, increasing the complexity of the optimization problem [3], [11], [18]. The definition of constraints through CBFs partially mitigates the problem, but undermines the invariance and feasibility properties provided by other MPC implementations.

The third mechanism relates more closely to the definition of safety constraints through CBFs and how to make them robust in the presence of uncertainties or external disturbances, which may degrade or compromise the safety guarantees [19]. To address this challenge, researchers have proposed extensions of CBFs defined considering the uncertainties and disturbances affecting the system using online parameter adaptation [20], data-driven approaches [21], and disturbance-observer-based techniques [19]. The research in this field of robustifying CBFs is still active and new approaches are continuously being proposed depending on the scenario considered [22].

The method proposed in this paper originates from the combination of these three mechanisms, with CBF constraints robustified with a time-varying safety margin proportional to the network disturbance, modifying the commanded control input in a discrete and predictive fashion.

III. PRELIMINARIES

A. Robust CBFs

Let us consider the nominal dynamics of a (possibly) nonlinear, control-affine system

$$\dot{\mathbf{x}}(t) = f_c(\mathbf{x}(t)) + g_c(\mathbf{x}(t))\mathbf{u}(t), \qquad (1)$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{u}(t) \in \mathcal{U}_{\mathrm{adm}} \subset \mathbb{R}^m$ denote the state and the input of the system, and $\mathcal{U}_{\mathrm{adm}}$ the admissible set of inputs. Functions $f_c : \mathbb{R}^n \to \mathbb{R}^n$ and $g_c : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ are assumed to be locally Lipschitz continuous functions, while the input corresponds to a feedback controller of the form $\mathbf{u}(t) = k(\mathbf{x}(t))$, such that the resulting dynamical system in Eq. (1) also is locally Lipschitz.

A CBF is then commonly defined as any continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ for which there exists an extended class $\mathcal{K}_\infty$ function α such that for all $\mathbf{x} \in \mathbb{R}^n$:

$$\sup_{\mathbf{u} \in \mathcal{U}_{\mathrm{adm}}} [L_{f_c} h(\mathbf{x}) + L_{g_c} h(\mathbf{x})\mathbf{u}] \geq -\alpha(h(\mathbf{x})), \qquad (2)$$

where $L_{f_c} h(\mathbf{x}) = \nabla h(\mathbf{x}) f_c(\mathbf{x})$ and $L_{g_c} h(\mathbf{x}) = \nabla h(\mathbf{x}) g_c(\mathbf{x})$ are Lie derivatives of h along f_c and g_c, see [8], [21]–[23].

Let us now consider an extension of the nominal dynamics

$$\dot{\mathbf{x}}(t) = f_c(\mathbf{x}(t)) + g_c(\mathbf{x}(t))\mathbf{u}(t) + \tilde{f}(\mathbf{x}(t)) + \tilde{g}(\mathbf{x}(t))\mathbf{u}(t), \quad (3)$$

where the new functions $\tilde{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $\tilde{g} : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ are locally Lipschitz continuous and represent the differences between the real and nominal system models [22], [24]. A possible approach to ensure safety despite model mismatch is to introduce a robustifying term $\sigma(\mathbf{x}, \mathbf{u})$ in the safety constraint. Taking inspiration from [24], a continuously differentiable function h is defined in [22] as a Robust CBF (RCBF) for Eq. (3) on the 0-superlevel set of h, if 0 is a regular value of h[1] and there exist functions $\sigma : \mathbb{R}_{\geq 0} \times \mathbb{R} \rightarrow \mathbb{R}$ and $\alpha \in \mathcal{K}_\infty$ such that

$$\sup_{\mathbf{u} \in \mathcal{U}_{\mathrm{adm}}} [\dot{h}_n(\mathbf{x}, \mathbf{u}) - \sigma(\mathbf{x}, \mathbf{u})] \geq -\alpha(h(\mathbf{x})), \qquad (4)$$

with h_n being the CBF for the nominal dynamics as in (2).

The idea behind introducing σ is to make the controller take a more conservative action against the undesired effects on safety caused by uncertainties. In [22], the authors illustrate an example of robust safety in a case with only bounded additive uncertainty (i.e., $||\tilde{f}(\cdot)|| \leq \bar{w}$ and $\tilde{g}(\cdot) = 0$). As we will present, our method follows the same setting, assuming that the uncertainties affecting the system result in a state shift representable as an additive term.

B. CBF discretization

When defining CBFs, a popular choice is to choose $\alpha(\cdot)$ as a linear function with constant coefficient $\beta \in \mathbb{R}_{\geq 0}$, i.e. $\alpha(h(\mathbf{x})) = \beta h(\mathbf{x})$. Assuming this choice and following the reasoning in [11], we can approximate $\dot{h}$ of Eq. (2) with

$$\dot{h}(\mathbf{x}, \mathbf{u}) \approx \frac{\Delta h}{\Delta t} = \frac{h_{t+\Delta t} - h_t}{\Delta t} \geq -\beta h_t, \qquad (5)$$

where $h_{t+\Delta t}$ and h_t represent the value of $h(\mathbf{x})$ in the next and in the current time step, while Δt is the control time interval. Following [14], Eq. (5) can be reformulated and extended to the discrete-time domain as

$$\Delta h(\mathbf{x}_k, \mathbf{u}_k) \geq \gamma h(\mathbf{x}_k) \qquad (6)$$

where $\gamma \in (0, 1]$ is a new variable incorporating Δt and β, while $\Delta h(\mathbf{x}_k, \mathbf{u}_k) = h(\mathbf{x}_{k+1}) - h(\mathbf{x}_k)$. With this, Eq. (6) is also equivalent to

$$h(\mathbf{x}_{k+1}) \geq (1 - \gamma)h(\mathbf{x}_k), \qquad (7)$$

which is the condition playing the role of Eq. (2) in the case of discrete CBFs (DCBFs).

IV. METHODS

Our method aims at unifying MPC and CBF for the purpose of increasing safety over a networked control architecture. For this purpose, there are a few things to consider to form our MPC problem: (i) the continuous-time nature of standard CBFs, (ii) the lack of precise real-time information of the system state due to network delays, and (iii) the lack of knowledge of the operator's behavior throughout the entire MPC prediction horizon.

For (i), we follow the discretization steps of Section III-B to arrive at

$$h(\mathbf{x}_{k+1}) \geq \omega(1 - \gamma)h(\mathbf{x}_k), \qquad (8)$$

where we have added an optimizable slack variable $\omega \in \mathbb{R}_{\geq 0}$ to the CBF in Eq. (7). The slack variable ω allows relaxing the decay rate imposed by the CBF, thereby enhancing feasibility. However, in order to minimize the deviation from the nominal decay rate $1 - \gamma$, a regularization term, such as $\Phi(\omega) := P_\omega(\omega - 1)^2$ for a $P_\omega \in \mathbb{R}_{>0}$, should be added to the cost function [25] (or see Eq. (10a)).

For (ii), we let our MPC rely on a predicted value of the state obtained by integrating the system dynamic model over an estimation of the network delay, such as seen in, e.g., [17]. Predicting the behavior of the operator, (iii), would depend upon the operator we are considering. If the operator is a high-level controller or trajectory planner, it is possible to know a priori the desired trajectory. In the case of a human operator, this is, however, not the case. For this work, we opted for a method that would be agnostic to the type of operator, and thus followed the approach proposed in [26], according to which *the simplest prediction methods outperform the more complex ones*. In practical terms, we let the MPC adopt the last velocity input received as the desired velocity throughout the entire horizon.

To deal with the uncertainties due to delays, we propose the addition of a safety margin σ_k, analogous to Eq. (4), to the discrete time CBF in Eq. (8), obtaining

$$h(\mathbf{x}_{k+1}) - \sigma_k \geq \omega_k(1 - \gamma)h(\mathbf{x}_k). \qquad (9)$$

Inspired by [3], [11], [22], we call the function satisfying this condition Optimal Decay RCBF (ODRCBF). However, unlike $\sigma(\mathbf{x}, \mathbf{u})$ in Eq. (4), we further propose that σ_k should also be a function of a set of network quality measures, such as round-trip time (RTT), packet loss frequency, etc.

After the above derivations, we arrive at an MPC optimization problem of the form[2]:

$$J_k^* = \min_{\mathbf{u}_{k:k+N-1}, \mathbf{\Omega}_{1:n_{\mathrm{obs}}}} p(\mathbf{u}_{k+N}, \hat{\mathbf{u}}_k^{\mathrm{des}})$$

$$+ \sum_{i=0}^{N-1} \left[q(\mathbf{u}_{k+i}, \hat{\mathbf{u}}_k^{\mathrm{des}}) + \sum_{j=1}^{n_{\mathrm{obs}}} \Phi(\omega_{k+i}^{(j)}) \right] \qquad (10a)$$

$$\text{s.t.: } \mathbf{x}_{k+i+1} = f_{\mathrm{kin}}(\mathbf{x}_{k+i}, \mathbf{u}_{k+i})$$

$$\mathbf{x}_{k+i} \in \mathcal{X}$$

$$\mathbf{u}_{k+i} \in \mathcal{U}_{\mathrm{adm}}$$

$$\mathbf{x}_k = \hat{\mathbf{x}}_k$$

$$h^{(j)}(\mathbf{x}_{k+i+1}) - \sigma_k \geq \omega_{k+i}^{(j)}(1 - \gamma)h^{(j)}(\mathbf{x}_{k+i}) \quad (10b)$$

$$\mathbf{\Omega}_j = \omega_{k:k+N-1}^{(j)}$$

$$\omega_{k+i}^{(j)} \geq 0$$

where $p(\mathbf{u}_{k+N}, \hat{\mathbf{u}}_k^{\mathrm{des}}) = \left\| \mathbf{u}_{k+N} - \hat{\mathbf{u}}_k^{\mathrm{des}} \right\|_P^2$, $q(\mathbf{u}_{k+i}, \hat{\mathbf{u}}_k^{\mathrm{des}}) = \left\| \mathbf{u}_{k+i} - \hat{\mathbf{u}}_k^{\mathrm{des}} \right\|_Q^2 + \left\| \mathbf{u}_{k+i} \right\|_R^2 + \left\| \Delta \mathbf{u}_{k+i+1} \right\|_S^2$, and Φ is the previously introduced regularization function. N is the horizon length and $i \in \{0, \ldots, N-1\}$, while $j \in \{1, \ldots, n_{\mathrm{obs}}\}$, with n_{obs} being the number of obstacles. $\mathbf{\Omega}_j$ denotes the

---

[1] A point q is a *regular value* of h if $h(x) = q$ implies $\nabla h(x) \neq 0$.

[2] $\|v\|_M$ represents the magnitude of vector v weighted with the positive matrix M; mathematically $\|v\|_M = \sqrt{v^\top M v}$.

slack variables associated with obstacle j. P, Q, R, and S are positive definite weight matrices. $f_{\text{kin}} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ denotes the system kinematics; $\hat{\mathbf{u}}_k^{\text{des}}$ is the predicted desired input, while $\hat{\mathbf{x}}_k$ the predicted state (both at timestep k); $\mathcal{X}$ is the state domain, which corresponds to the 3D space in case there are no constraints, otherwise it is the working space the controlled system is subject to. Note that the state domain does not need to take into account the obstacles as those are considered directly by the ODRCBFs. As usual for MPC, only the first entry of the optimal solution $\mathbf{u}_k^* = \arg \min J_k^*$ is applied to the system.

A major difference from the standard formulation is that the safety constraints in Eq. (10b) are defined through ODR-CBFs, rather than through Euclidean distances or other types of CBFs as done in previous works. Note that a constraint like the one referenced has to be integrated for each obstacle and timestep related to the problem definition.

Another difference comes from our assumption on the existence of a low-level controller (see Fig. 1) so that the objective is to track a desired velocity, rather than a path.

Now that we have described the main parts of the formulation, we can dig into the details and explain how we define the h function and the safety margin σ in our specific case.

In this paper, we approximate both the robot and the obstacles as spheres. In addition, we assume we want to have a minimum distance margin d_{min} between the robot and the obstacle. The rationale behind this margin is that, even if the system and the obstacle do not touch each other, conditions become unsafe when the distance between the two is below a certain threshold. The margin can also be applied to cases when the true state of the system is subject to uncertainties. Given $\mathbf{x}$ and $\mathbf{p}^{(j)}$ as the centers of the robot and of obstacle j, r_{rob} and $r_{\text{obs}}^{(j)}$ their spherical radii, and d_{min} the minimum distance margin, we define the function $h^{(j)}$ as

$$h^{(j)}(\mathbf{x}) = \left\| \mathbf{x} - \mathbf{p}_{\text{obs}}^{(j)} \right\|^2 - \left(r_{\text{rob}} + r_{\text{obs}}^{(j)} + d_{\text{min}} \right)^2. \quad (11)$$

Concerning σ, we already stated that its objective is to make the controller filter the velocity command provided and compensate for the uncertainties affecting the system's information. As mentioned, we only assume additive uncertainty caused by delays, which translates to a shift between the system's position used by the controller at a given time and the real one. This shift is proportional to both the delays and the system's velocity. There are multiple ways to combine these variables to design a safety margin. The approach we propose adopts the following equation

$$\sigma_k = \|\mathbf{v}_k\| \cdot \bar{T}_k^{\text{RTT}} + v_{\text{max}} \cdot k_\sigma \cdot \bar{\sigma}_k^{\text{RTT}}, \quad (12)$$

where $\|\mathbf{v}_k\|$ and v_{max} are the system's velocity and maximum velocity magnitudes, while $\bar{T}_k^{\text{RTT}}$ and $\bar{\sigma}_k^{\text{RTT}}$ the estimated values of the round trip time (RTT) and its standard deviation. The idea is to make the safety margin directly proportional to the delay affecting the information and its oscillation, with the possibility of adjusting it by changing the system's velocity or the factor $k_\sigma \in \mathbb{R}_{\geq 0}$. Note that, in a scenario such as the one we are considering, there are
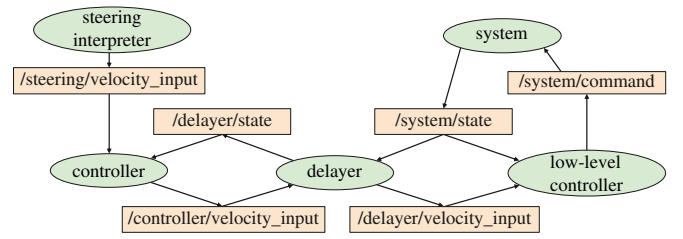

Fig. 2: **Node architecture.** Graph showing the connections between the different entities involved in the project. Each entity corresponds to a ROS 2 node and is represented with an ellipse, while the topics are in rectangles. The graph has been redrawn from that obtained using the rqt_graph tool.
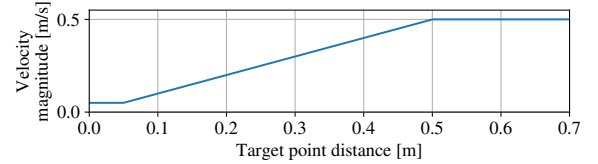

Fig. 3: **Desired velocity magnitude law.** Relationship between the desired velocity magnitude specified by the tester and the system's distance from the target position. When the distance is smaller than 0.15 m, its magnitude is at the lower limit 0.05 m/s. On the other hand, when the distance is greater than 0.50 m, the magnitude is at its upper limit 0.50 m/s.

multiple sources of delay: the delay of transmitting the input, the computation time for solving the optimization problem, and the delay of receiving the system state information. Despite this, their effect on the shift described above is similar and we can thus consider them all together by using the RTT.

Given the above, we can see our method based on the σ_k defined in Eq. (12) as another data-driven approach to calculate a time-varying safety margin for RCBFs similar to those reported in [22, Table 1], but in the case of a system disturbed by delays.

V. RESULTS

In this section, we aim to validate the controller presented in Section IV in simulation and a hardware setup. In simulation, we evaluate the performance difference in a few metrics when the controller utilizes the proposed dynamic safety margin σ_k against the case when it does not ($\sigma_k = 0$). In hardware, we validate that the controller with safety margin σ_k can run in real time and that the safety constraints are not violated when the system operates under realistic network delay.

For both the simulated and real case, we use ROS 2 Galactic [27] as the underlying communication infrastructure with a setup as illustrated in Fig. 2, displaying the rqt_graph[3] with the connections between the different ROS nodes and topics. The nodes at its core are: (i) the steering interpreter translating the steering commands sent by the operator into 3D Cartesian velocity input, (ii) the low-level velocity controller, (iii) a delayer node capable of simulating network delays in packet transmission, (iv) a system node which

---

[3]http://wiki.ros.org/rqt_graph

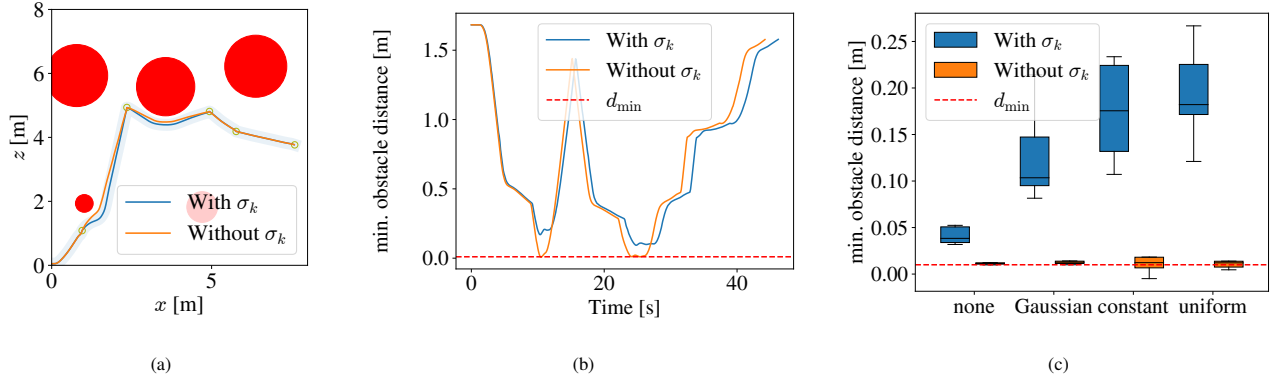(a)   (b)   (c)

Fig. 4: **Simulation results.** (a) xz-plane projections of the paths followed by the system for a task in simulation with obstacles in red and the target points as yellow circles. The communication is affected by a Gaussian delay with $50\,\text{ms}$ mean and $20\,\text{ms}$ standard deviation. (b) A minimum distance to obstacle for both methods and the same task. (c) Statistical results for all tasks showing minimum obstacle distance. The whiskers denote the most extreme values.
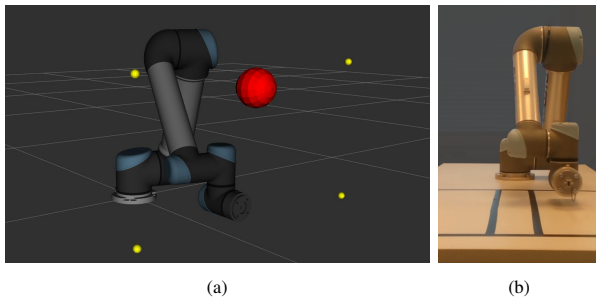

(a)   (b)

Fig. 5: **Hardware setup.** (a) 3D rendering of the UR5 manipulator and the target task in rviz. The red sphere represents the obstacle the end effector has to avoid, while the yellow ones target points the end effector has to reach. (b) The real UR5 manipulator used in the experiment.

Fig. 6: **Hardware results.** Results from the hardware experiment using the UR5 robot with a recorded set of network delays of an average RTT of $11.61\,\text{ms}$. The obstacle is marked in red, and the target points are yellow circles. The minimum required margin $r_{\text{obs}+d_{\min}}$ has been highlighted in light blue along the UR5 trajectory.

receives low-level control commands and sends its state, and (v) our controller. The delayer node essentially acts as an intermediary between the two sides of the communication and delays the transmission of messages by holding them for a given time. The delay values can be either generated from a probability distribution or taken from a dataset of delay samples.

To ensure an unbiased comparison of various controller instances, we conducted the experiments using what we refer to as the *tester* module. This module sends the velocity commands to make the system complete the task, both in the simulation and hardware scenarios. Its role is to mimic a human operator, directing the system toward its next target position, but without considering the obstacles. The relationship between the velocity command sent by the tester module and the system's distance from the target position is depicted in Fig. 3. Note that, it is not crucial that the behavior just described represents a real operator's intention; the crucial part for our considerations is to maintain a consistent method for imposing identical velocities under the same conditions, allowing for a fair comparison of the different formulations.

The tasks assigned to the two systems in the two scenarios were different, but based on the same idea of reaching a set of target points in the working space while avoiding obstacles.
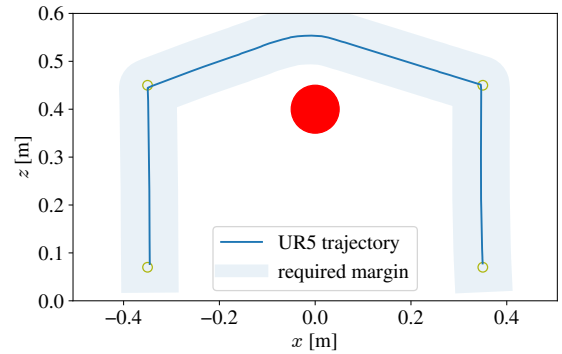
The location of target points and obstacles were generated depending on the experiment.

The proposed controller Eq. (10) was implemented as a CasADi [28] model with Ipopt [29] as a solver, and initially tested in a simulated environment using Gazebo[4] and later on a Universal Robots UR5[5] robotic manipulator operating in the laboratory. We used the following values for matrices in Eq. (10):

$$Q = I_3 \cdot 10, \; R = I_3, \; P = I_3 \cdot 10, \; S = I_3 \cdot 10, \; P_\omega = 100,$$

and let $d_{\min} = 0.01\,\text{m}$ and $\gamma = 0.5$. The system kinematics were set using the Euler forward method such that $\hat{\mathbf{x}}_{k+i+1} = \hat{\mathbf{x}}_{k+i} + \mathrm{d}t\,\mathbf{u}_{k+i}$, where $\mathrm{d}t$ was set to $0.1\,\text{s}$. The MPC was running at a frequency of $10\,\text{Hz}$ in both cases and with a horizon $N = 10$. The values $\bar{T}^{\text{RTT}}$ and $\bar{\sigma}_k^{\text{RTT}}$ were estimated using the moving average approach, with a window length of 30 samples. We let $k_\sigma = 1$ for both cases, while $\|v_{\max}\|$ is set to $0.87\,\text{m/s}$ and $0.52\,\text{m/s}$ for the simulation and hardware experiment, respectively. While no constraints were set on

---

[4] https://gazebosim.org/home/
[5] https://universal-robots.com/products/

TABLE I: **Aggregated simulation results.** Success rate of the proposed method with and without the safety margin σ_k aggregated by delay type. The results refer to the execution of 10 different tasks in the simulation.

Delay type	None	Gaussian	Constant	Uniform
Params [ms]	-	$\mu = 50$, $\sigma = 20$	200	$[50 - 20]$
With σ_k	100%	100%	100%	100%
Without σ_k	90%	90%	60%	70%

the state, we set input constraints such that for any element u in $\mathbf{u}_{k:k+N-1}$, $|u| \leq u_b$, where u_b was set to 0.5 m/s in simulation and 0.3 m/s in the hardware experiment.

A. Simulation results

In the simulation, the system controlled is a free-flyer box capable of moving in 3D space. The box is 0.25 m wide, 0.25 m long, and 0.1 m high and has a mass of 1 kg. We thus let r_{rob} be defined as the radius of the circumscribed sphere of this cuboid. The simulation framework used was Gazebo 11, and velocity control of the system was achieved by coupling the Gazebo Force plugin with a PID controller, and the system position and velocity was given through the Gazebo P3D plugin.

In our evaluations, 10 different tasks were constructed by placing out a random number of obstacles and target points in the xz-plane in the domain $[0.45 \text{ m}, 7.6 \text{ m}]^2 \times [0, 0]$. The position of the obstacles and target points, and the radius of the obstacles were random, but to ensure feasibility, we resampled if an obstacle or target point was too close to another obstacle. For the delay, we considered four cases: (i) no delay, (ii) delay from a Gaussian distribution with 50 ms mean and 20 ms standard deviation, (iii) constant delay of 200 ms, and (iv) delay from a uniform distribution over the interval 50-200 ms.

Table I shows the success rate obtained by the proposed method in simulation, when guiding the system to complete the 10 tasks with and without the safety margin σ_k, and for the different delay distributions. The 90 % success rate for the case of no delay and without σ_k, is likely to be attributed to the fact that, even if there are no network delays affecting the communication, the information is always subject to the delay caused by the time needed to solve the optimization problem, which can be mitigated through the safety margin.

To better demonstrate the effect of the safety margin, one of the tasks subject to Gaussian delay is shown in Figs. 4(a) and 4(b). Fig. 4(a) shows a xz-plane projection of the results, while Fig. 4(b) shows the minimum distance to an obstacle over time. Clearly, the execution becomes unsafe at times when the safety margin σ_k is not used, while the controller with σ_k maintains the distance well above $d_{\min}$ at all times, even though the overall trajectory is minimally modified. The safety margin is thus not leading to an overly conservative behavior, but rather, it makes the controller intervene only when the system is close to reaching an unsafe region. Fig. 4(c) shows the statistics for all ten tasks and indicates that the increased ability to maintain a safe distance when using σ_k is a general trend. Further, we notice that this also

applies for the case of no delay, which we again attribute to the increased safety added by taking the computation time into account.

B. Hardware results

The hardware system we used to test the controller is a Universal Robots UR5 manipulator operating in the Kista laboratory as seen in Fig. 5(b). Unlike the previous case, the velocity commands obtained by the optimizer and sent through the network to the low level controller are not directly applicable to this type of system, but have to be translated into joint motor commands. We utilized Moveit Servo to handle this translation, allowing seamless execution of the desired motion on the UR5 manipulator end effector while also providing end effector position estimates. The task tested on this system is similar to that of the previous case, although the distances between the points are generally shorter because of the limitation of the working space reachable by the robot. For this experiment we let $r_{\text{rob}} = 5$ cm, always positioned at the origin of the end effector reference frame.

Unlike in the simulation, we tested the controller on a single task with different delay types. Fig. 5(a) depicts the 3D rendering of this task in rviz[6]. As a final result, our method successfully maintained safety during the teleoperation of the UR5 robot, even when faced with various types of delays. Fig. 6 shows the result when using recorded network delays between Lund and Stockholm with an average RTT of 11.61 ms and standard deviation 3.29 ms, indicating that the controller is capable of maintaining safe system behavior also in real-time applications while under the influence of realistic delay.

VI. CONCLUSIONS

Our results indicate that the proposed method effectively aligns with the objective of ensuring the safety of the controlled system by minimally modifying the input provided by a remote operator despite the network delays in a less conservative manner. In the case analyzed, safety refers to the ability of the system to avoid obstacles placed in the same environment where it has to operate. The versatility of the method and the soundness of the implementation choices were validated through testing in both the Gazebo simulator and on a real UR5 industrial robot. Future work may focus on the system's ability to maintain safety in scenarios involving packet loss, consider moving and differently shaped obstacles, and refine the accuracy of the system state prediction.

---

[6]https://index.ros.org/p/rviz2/

REFERENCES

[1] J. Vertut and P. Coiffet, *Teleoperation and Robotics*. Springer Netherlands, 1985.

[2] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008.

[3] J. Zeng, "Challenges of control barrier functions: Optimization, control, planning, and navigation," UC Berkeley, 2022.

[4] S. Hu, E. Babaians, M. Karimi, and E. Steinbach, "NMPC-MP: Real-time nonlinear model predictive control for safe motion planning in manipulator teleoperation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8309–8316.

[5] A. Singletary, K. Klingebiel, J. Bourne, A. Browning, P. Tokumaru, and A. Ames, "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sep. 2021.

[6] F. Ferraguti, C. T. Landi, A. Singletary, H.-C. Lin, A. Ames, C. Secchi, and M. Bonfe, "Safety and efficiency in robotics: The control barrier functions approach," vol. 29, no. 3, pp. 139–151, 2022.

[7] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[8] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431.

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

[10] 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.

[11] S. Li, Z. Yuan, Y. Chen, F. Luo, Z. Yang, Q. Ye, W. Fu, and Y. Fu, "Optimizable control barrier functions to improve feasibility and add behavior diversity while ensuring safety," *Electronics*, vol. 11, no. 22, p. 3657, Nov. 2022.

[12] G. Wu and K. Sreenath, "Safety-critical and constrained geometric control synthesis using control lyapunov and control barrier functions for systems evolving on manifolds," in *2015 American Control Conference (ACC)*. IEEE, 2015, pp. 2038–2044.

[13] B. Xu and K. Sreenath, "Safe teleoperation of dynamic UAVs through control barrier functions," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7848–7855.

[14] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," in *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation, 2017.

[15] A. Singletary, Y. Chen, and A. D. Ames, "Control barrier functions for sampled-data systems with input delays," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 804–809.

[16] M. Rubagotti, T. Taunyazov, B. Omarali, and A. Shintemirov, "Semi-autonomous robot teleoperation with obstacle avoidance via model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2746–2753, Jul. 2019.

[17] V. N. Sankaranarayanan, G. Damigos, A. S. Seisa, S. G. Satpute, T. Lindgren, and G. Nikolakopoulos, "PACED-5G: Predictive autonomous control using edge for drones over 5G," in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, Jun. 2023.

[18] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *2021 American Control Conference (ACC)*. IEEE, May 2021.

[19] A. Alan, T. G. Molnar, E. Das, A. D. Ames, and G. Orosz, "Disturbance observers for robust safety-critical control with control barrier functions," *IEEE Control Systems Letters*, vol. 7, pp. 1123–1128, 2023.

[20] A. J. Taylor and A. D. Ames, "Adaptive safety with control barrier functions," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1399–1405.

[21] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Robust adaptive control barrier functions: An adaptive and data-driven approach to safety," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 1031–1036, Jul. 2021.

[22] A. Alan, T. G. Molnar, A. D. Ames, and G. Orosz, "Parameterized barrier functions to guarantee safety under uncertainty," *IEEE Control Systems Letters*, vol. 7, pp. 2077–2082, 2023.

[23] L. Perko, *Differential equations and dynamical systems*. Springer Science & Business Media, 2013, vol. 7.

[24] M. Jankovic, "Robust control barrier functions for constrained stabilization of nonlinear systems," *Automatica*, vol. 96, pp. 359–367, Oct. 2018.

[25] J. Zeng, Z. Li, and K. Sreenath, "Enhancing feasibility and safety of nonlinear model predictive control with discrete-time control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, Dec. 2021.

[26] R. Chipalkatty, G. Droge, and M. B. Egerstedt, "Less is more: Mixed-initiative model-predictive control with human inputs," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 695–703, Jun. 2013.

[27] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022.

[28] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, Jul. 2018.

[29] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Apr. 2005.