Towards Safe Human-Quadrotor Interaction: Mixed-Initiative Control via Real-Time NMPC

Bárbara Barros Carlos [ID], Antonio Franchi [ID], and Giuseppe Oriolo [ID]

Abstract—This article presents a novel algorithm for blending human inputs and automatic controller commands, guaranteeing safety in mixed-initiative interactions between humans and quadrotors. The algorithm is based on nonlinear model predictive control (NMPC) and involves using the state solution to assess whether safety- and/or task-related rules are met to mix control authority. The mixing is attained through the convex combination of human and actual robot costs and is driven by a continuous function that measures the rules' violation. To achieve real-time feasibility, we rely on an efficient real-time iteration (RTI) variant of a sequential quadratic programming (SQP) scheme to cast the mixed-initiative controller. We demonstrate the effectiveness of our algorithm through numerical simulations, where a second autonomous algorithm is used to emulate the behavior of pilots with different skill levels. Simulations show that our scheme provides suitable assistance to pilots, especially novices, in a workspace with obstacles while bolstering computational efficiency.

Index Terms—Safety in HRI, optimization and optimal control, aerial systems: mechanics and control.

I. INTRODUCTION

QUADROTORS have been widely used across various applications, resulting in a significant market expected to grow exponentially. Due to this increasing demand, one can envisage that, in the future, learning to fly a quadrotor will become a taken-for-granted skill, just as learning to drive a car nowadays. A first step towards making this idea a realistic prospect could be a mixed-initiative control approach that guarantees the aerial system's "working conditions" while the novice operator learns to fly it. In other words, as long as some safety- and/or task-related rules are met, the operator commands are obeyed. Once novices start working in the context of complex tasks, it is essential to provide them with assistance that gradually decreases as their skill increases – here, skill refers to the knowledge and ability that allow the use of the robot to accomplish a task. Conversely, one could think of an experienced operator who already has the skill level that enables him/her to fly a quadrotor. But as his/her cognitive load is primarily focused on the short-term aspect of the task, he/she cannot consider other underlying factors in the long run (e.g., safety). It is then essential to have a control approach that supervises the long-term task to prevent the operator from being overwhelmed by (too) high engagement.

To fill these gaps, this paper proposes an efficient mixed-initiative controller based on nonlinear model predictive control (NMPC) to enforce safety in human-quadrotor interactions. The primary objective of this study is to provide a "safety-layer" controller over which one may define metrics to assess pilots' learning curves. The technical difficulties associated with the proposed controller are twofold: first, satisfy both the existing constraints and the human intentions; second, solve the underlying optimization problems that include a high-dimensional quadrotor system under the available computation time. We demonstrate how to handle these difficulties using concepts of zone MPC [1] and online weight adaptation within the real-time iteration (RTI) scheme – an efficient Newton-type algorithm for real-time NMPC applications. We exploit high-performance numerical optimization algorithms to further speed up solution times, including a structure-exploiting convex solver and linear algebra kernels for small-to-midsize matrices. The mixed-initiative controller is validated in simulation against a second autonomous algorithm that emulates pilots with different skill levels.

A. Related Works

One way robots and humans can interact is to put the human in the loop with some blending scheme [2]: the human and the automatic controller have control over the robot, and both systems must adapt to ensure task completion. For example, in exoskeleton systems, the interplay between robot and human may be characterized as an interaction between teacher and student in a learning process. The teacher (robot) tries to minimize the student (human) error, applying a minimal effort [3]. In the context of autonomous cars, the driver interacts with the automatic controller that aims at reducing the driver's workload and, at the same time, taking prompt actions in case of human failure [4], [5]. For human-swarm systems, the interaction paradigm is less obvious. As the size of the swarm increases, control should become more focused on the swarm as a whole rather than on the individuals, given the human's limited capacity to multitask. In this case, the interaction is more concerned with the human-swarm ability to accomplish a particular task than with the swarm spatial positioning [6].

Other interactive approaches rely on methods designed from the human perspective, where, in general, the presence of haptic cues (e.g., force feedback) increases situational awareness. These methods lean heavily in favor of perceiving the human as the source of action and the robot as the passive collaborator. For instance, virtual fixtures have been used to inform the operator of the highest comfort position, distance from the target position, proximity to unsafe kinematic configurations, or misalignment in contact-driven surface conditioning tasks [7], [8]. Obstacle avoidance is one of the most critical requirements to be met in many robotics scenarios. In this matter, researchers have been using haptic feedback to warn the operator about instantaneous collisions [9], [10]. Several human-collaborative schemes are compared in [11] where it is shown that haptic feedback is one of the main aspects. Among other considerations, it should be noted that the method used to generate assistive haptic cues strongly determines the usefulness of kinesthetic guidance to a large extent. For this reason, considerable effort has been put into introducing new assistive methods to improve task performance explicitly. The authors in [12] propose a force-feedback telepresence method based on an external wrench estimation that enables the operator to feel, e.g., wind and contact forces. The work in [13] uses the gradient of an artificial potential field to generate repulse forces that help humans to avoid touching adjacent vessels while operating surgical robots. Interested readers are referred to [14], [15] for the introduction of other noteworthy methods.

B. Contributions and Paper Structure

The main contributions of this work are twofold:
- A mixed-initiative control scheme for human-quadrotor systems based on NMPC with safety guarantees.
- Numerical simulations that show the effectiveness and computational efficiency of the proposed algorithm.

The proposed mixed-initiative (MI) control approach for human-quadrotor interaction leads to major advantages compared to existing methods if applied to our scenario. Unlike [2]–[5], our approach offers a rigorous predictive diagnosis of interaction degradation so that most of the control authority is allocated to the most capable agent, ultimately yielding a time-varying synergy between the human and the automatic controller. While the predictive behavior is relevant in a highly nonlinear context (disregarded in previous works), the dynamic synergy is particularly important to manage the control authority in conflict situations, i.e., when human and automatic controller commands are rather different. To the best of our knowledge, this is the first work that subsumes the advantages of the RTI scheme, zone MPC, and high-performance algorithms to synthesize a nonlinear MI controller cognizant of how to mix control authority to enforce safety and with real-time capabilities. For this reason, this work makes a significant contribution to human-robot interaction literature.

The paper is structured as follows. Section II states the problem we want to solve. Section III provides the details of the proposed algorithm. The numerical testbeds and the obtained results are discussed in Section IV. Section V offers some hints on future work.

C. Notation

The $n \times n$ identity matrix is denoted by $\boldsymbol{I}_n$. An all-ones vector of dimension n is denoted by $\boldsymbol{1}_n$, whereas $\boldsymbol{0}_n$ denotes a n-vector of zeros. When dealing with norms of vectors $\boldsymbol{x} \in \mathbb{R}^n$, we denote by $\|\boldsymbol{x}\|$ the Euclidean norm. The set of positive real-valued numbers is denoted by $\mathbb{R}_{>0}$.

II. PROBLEM STATEMENT

The fundamental problem in mixed-initiative control is how to blend human inputs and automatic controller commands to realize the former as closely as possible while enforcing safety. In this context, the formulation of our problem hinges upon the following components:

- A *robot* represented by a nonlinear system

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}, \boldsymbol{u}), \qquad (1)$$

  with state $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$ and control inputs $\boldsymbol{u} \in \mathbb{R}^{n_u}$, subject to state and input constraints.
- A *motion generator* providing a *reference trajectory* $\boldsymbol{\eta}^r$ for the *task variables* $\boldsymbol{\eta} = \boldsymbol{c}(\boldsymbol{\xi})$, which take values in $\mathbb{R}^{n_\eta}$. Typical examples of task variables are a subset of the state variables or the position of some relevant point of the robot. As for the motion generator, it can be either an offline planner (such as a motion planner among obstacles) or an online controller.
- A *human* assigning reference values $\boldsymbol{\nu}^r$ to some subset $\boldsymbol{\nu} \in \mathbb{R}^{n_\nu}$ of the state variables through a control interface. These signals are henceforth called *human inputs* and convey human intentions.
- A set of *working conditions* defining requirements on the state for the robot to operate healthily. These can include both safety rules (e.g., maximum kinetic energy, maximum dissipation) and task-related rules (e.g., the accuracy of end-effector positioning). We assume that working conditions are such that the associated feasible set is convex.

For this setting, we want to devise a *mixed-initiative controller* that will attempt to execute human inputs as much as possible without compromising both working conditions and constraints inherent to the robot's physical limitations. As we will see, in addition to the mixed-initiative controller itself, the control algorithm also includes a *blending mechanism* that predicts the violation of working conditions and gives most of the control authority to the most capable agent at any time.

III. MIXED-INITIATIVE CONTROL

The mixed-initiative controller considers the human operator and the motion generator as two different agents. The blending mechanism assesses the working conditions' violation and distributes control authority accordingly. For example, working conditions may include a requirement of collision-free motion. If human inputs ignore this, a violation will be predicted, and the blending mechanism will shift more control authority to the motion generator. In the following, we describe in detail the proposed algorithm.

A. Algorithm Overview

Mixed control is accomplished through a *mixed-initiative NMPC controller* whose cost function – the *mixed-initiative cost* – is a convex combination between the *actual robot cost* and the *human input cost*. The latter is a cost term that penalizes the deviation from the human inputs. The proposed blending mechanism is a continuous function between 0 and 1, named *predicted healthiness index*, that drives the convex combination.
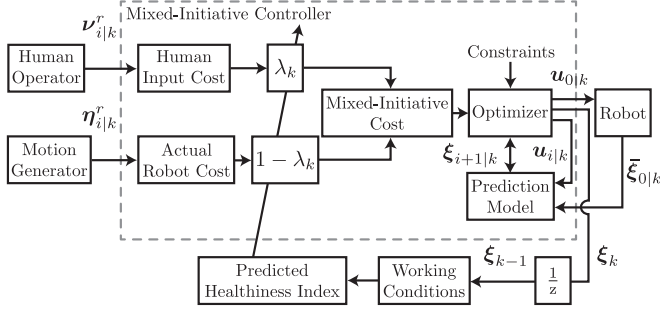
Fig. 1. A block diagram of the proposed mixed-initiative control algorithm.

It is computed in this way: when the index tends to 0, the working conditions' violation is too close and, hence, the mixed-initiative cost tends to the actual robot cost. When the index tends to 1, the violation is far enough and, therefore, the mixed-initiative cost tends to the human input cost.

Based on the state solution of the NMPC controller, at each time instant, the predicted healthiness index computes how far the robot is from violating these conditions within a *virtual horizon* whose length is defined by the designer. It then selects a value between 0 and 1 and blends the individual costs. The idea is that the closer the index to 0, the more troublesome it will be to keep the system within the working conditions in the next virtual horizon, even if full control is given to the motion generator (and human inputs are heavily ignored from that moment on). Vice-versa, the closer the index to 1, the easier the motion generator's job will be if it takes full control of the system. Fig. 1 shows the algorithm's block diagram.

B. Mixed-Initiative Controller

When using NMPC to control a system, a nonlinear non-convex program is solved using the current state as the initial value at each sampling instant. However, as the computational burden associated with the solution can be rather long, the employment of NMPC has only recently been extended to applications where shorter sampling times are required [16]. Typically, a continuous-time, infinite-dimensional optimal control problem (OCP) is tailored according to the problem at hand, discretized into N intervals using a time step Δt over a fixed time horizon t_N, and then solved. In doing so, the tailored OCP is transcribed into a discrete-time, finite-dimensional nonlinear program (NLP), now defined over a refined coarse grid $[t_0, t_N]$, for which the Karush-Kuhn-Tucker conditions are set up and solved at each sampling interval $[t_k, t_{k+1}]$. In our approach, we cast the MI controller as a constrained NLP with linear least-squares tracking cost, formulated as follows:

Problem 1 (Mixed-Initiative Controller):

$$\min_{\boldsymbol{\xi}_k, \boldsymbol{u}_k} \sum_{i=0}^{N-1} L(\boldsymbol{\eta}_{i|k}, \boldsymbol{\nu}_{i|k}, \boldsymbol{h}(\boldsymbol{\xi}_{i|k}), \boldsymbol{u}_{i|k}) + \quad (2a)$$

$$M(\boldsymbol{\eta}_{k+N|k}, \boldsymbol{\nu}_{k+N|k}, \boldsymbol{h}_N(\boldsymbol{\xi}_{k+N|k}))$$

$$\text{s.t. } \boldsymbol{\xi}_{0|k} - \bar{\boldsymbol{\xi}}_{0|k} = 0, \quad (2b)$$

$$\boldsymbol{\xi}_{i+1|k} - \boldsymbol{F}(\boldsymbol{\xi}_{i|k}, \boldsymbol{u}_{i|k}) = 0, \quad i = 0, \dots, N-1, \quad (2c)$$

$$\boldsymbol{\xi}_{i|k} \in \mathcal{X}, \quad i = 1, \dots, N-1, \quad (2d)$$

$$\boldsymbol{u}_{i|k} \in \mathcal{U}, \quad i = 0, \dots, N-1, \quad (2e)$$

where

$$L(\cdot) = \frac{1}{2}((1-\lambda_k)\Delta\boldsymbol{\eta}_{i|k}^T \boldsymbol{Q}_\eta \Delta\boldsymbol{\eta}_{i|k} + \lambda_k \Delta\boldsymbol{\nu}_{i|k}^T \boldsymbol{Q}_\nu \Delta\boldsymbol{\nu}_{i|k}$$

$$+ (1-\lambda_k)\Delta\boldsymbol{e}_{i|k}^T \boldsymbol{Q}_e \Delta\boldsymbol{e}_{i|k} + (1-\lambda_k)\boldsymbol{u}_{i|k}^T \boldsymbol{R}\boldsymbol{u}_{i|k})$$

$$M(\cdot) = \frac{1}{2}((1-\lambda_k)\Delta\boldsymbol{\eta}_{k+N|k}^T \boldsymbol{Q}_{\eta_N} \Delta\boldsymbol{\eta}_{k+N|k}$$

$$+ \lambda_k \Delta\boldsymbol{\nu}_{k+N|k}^T \boldsymbol{Q}_{\nu_N} \Delta\boldsymbol{\nu}_{k+N|k} +$$

$$(1-\lambda_k)\Delta\boldsymbol{e}_{k+N|k}^T \boldsymbol{Q}_{e_N} \Delta\boldsymbol{e}_{k+N|k}).$$

Here, $\boldsymbol{\xi}_k = (\boldsymbol{\xi}_{0|k}, \dots, \boldsymbol{\xi}_{N|k})$ and $\boldsymbol{u}_k = (\boldsymbol{u}_{0|k}, \dots, \boldsymbol{u}_{N-1|k})$ represent the state and input trajectories at time t_k of the discrete-time system whose dynamics are described by $\boldsymbol{F} : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\xi}$. The horizon length is denoted by N while $\bar{\boldsymbol{\xi}}_{0|k}$ denotes the current state estimate. The functions L and M represent the stage and terminal cost terms, respectively, which are weighted by the positive-definite matrices $\boldsymbol{Q}_\eta$, $\boldsymbol{Q}_{\eta_N} \in \mathbb{R}^{n_\eta \times n_\eta}$, $\boldsymbol{Q}_\nu$, $\boldsymbol{Q}_{\nu_N} \in \mathbb{R}^{n_\nu \times n_\nu}$, $\boldsymbol{Q}_e$, $\boldsymbol{Q}_{e_N} \in \mathbb{R}^{n_h \times n_h}$, and $\boldsymbol{R} \in \mathbb{R}^{n_u \times n_u}$. The convex polytopic sets $\mathcal{X}$ and $\mathcal{U}$ implement the state and input constraints associated with the physical limitations of the robot.

Moreover, we denote the task variables tracking error as $\Delta\boldsymbol{\eta}_{i|k} = \boldsymbol{\eta}_{i|k} - \boldsymbol{\eta}_{i|k}^r$, $\Delta\boldsymbol{\eta}_{k+N|k} = \boldsymbol{\eta}_{k+N|k} - \boldsymbol{\eta}_{k+N|k}^r$, and the human inputs' tracking error as $\Delta\boldsymbol{\nu}_{i|k} = \boldsymbol{\nu}_{i|k} - \boldsymbol{\nu}_{i|k}^r$, $\Delta\boldsymbol{\nu}_{k+N|k} = \boldsymbol{\nu}_{k+N|k} - \boldsymbol{\nu}_{k+N|k}^r$. The cost function also includes other penalty terms that may be relevant to the specific application. Their tracking errors are denoted as $\Delta\boldsymbol{e}_{i|k} = \boldsymbol{h}(\boldsymbol{\xi}_{i|k}) - \boldsymbol{e}_{i|k}^r$, and $\Delta\boldsymbol{e}_{k+N|k} = \boldsymbol{h}_N(\boldsymbol{\xi}_{k+N|k}) - \boldsymbol{e}_{k+N|k}^r$, where the output functions $\boldsymbol{h}(\boldsymbol{\xi}_{i|k})$, $\boldsymbol{h}_N(\boldsymbol{\xi}_{k+N|k}) \in \mathbb{R}^{n_h}$ and their respective references $\boldsymbol{e}_{i|k}^r$, $\boldsymbol{e}_{k+N|k}^r \in \mathbb{R}^{n_h}$ are defined by the relative complement $\mathbb{R}^{n_\xi} \setminus (\mathbb{R}^{n_\eta} \cup \mathbb{R}^{n_\nu})$.

The scaling factor $\lambda_k \in (0, 1)$ is the output of the blending mechanism and is used to determine how the control authority should be mixed. Looking at the cost function, one can observe that increasing the value of λ_k will give more weighting to human inputs than motion generator commands. We consider an open interval as we are interested in strictly convex problems. Finally, once the solution for interval $[t_k, t_{k+1}]$ is computed, the first element of the input trajectory $\boldsymbol{u}_{0|k}$ is applied to the system before shifting the horizon forward in time, see Fig. 1.

C. Working Conditions

While the particular requirements that a robot has to obey may differ from one system to another, a generic approach is to express these requirements through suitable working conditions. Here, these conditions are designed to help pilots fly a quadrotor along a collision-free trajectory in an arena with obstacles. To this end, we impose a maximum deviation $r \in \mathbb{R}_{>0}$ from the reference trajectory to enforce safety. In terms of Euclidean norm, this condition reads as follows:

$$d(\boldsymbol{\eta}) = \|\boldsymbol{\eta} - \boldsymbol{\eta}^r\| \le r. \quad (3)$$

Any point $\boldsymbol{\eta}$ satisfying (3) describes a convex free region $\mathcal{B}$, which we defined as

$$\mathcal{B} = \{\boldsymbol{\eta} \in \mathbb{R}^{n_\eta} : \|\boldsymbol{\eta} - \boldsymbol{\eta}^r\| \le r\}. \quad (4)$$

Note that $\mathcal{B}$ is a norm ball fully described by its center point $\boldsymbol{\eta}^r$. This observation implies that the robot is softly constrained to

move inside of a ball, where its actual trajectory is just one of the possible trajectories that avoid collisions, and at best, it is the reference trajectory itself. Although we rely on a map and a collision-free trajectory, one can tailor the method in two ways to deal with unstructured environments: (a) the motion generator can plan collision-free trajectories by taking into account the nearby obstacles detected by the robot sensors; (b) the MPC itself can do the same by incorporating an explicit collision avoidance constraint in the formulation and using the robot sensors to set up this constraint, e.g., see [17].

Unlike the constraints in the MI controller, i.e., Eq. (2b)–(2e), the working conditions may be seen as "soft constraints" that facilitate the decoupled design of the safety- and/or task-related rules through the task variables. By defining them this way, we dovetail zone MPC ideas into our MI controller. Zone MPC is used when the specific setpoint of an output variable is of low relevance compared to a zone delimited by upper and lower bounds, typically read as a soft constraint [1]. This idea can be easily accommodated by choosing a suitable weighting matrix to penalize the output deviation in the cost function. For a linear least-squares tracking cost, an output deviation, which depends linearly on the states, causes the cost to strictly increase as one moves away from the reference. This increase is acceptable as long as the deviation rests inside the zone. As the deviation crosses or approaches the zone's boundary, large penalty weights are set. We revisit this concept by making the weighting matrices of human and actual robot costs antagonist: when the task variables' prediction lies outside $\mathcal{B}$, the human cost weighting gets closer and closer to zero, while the actual robot cost weighting tends to a high value, which brings back the prediction to the interior of $\mathcal{B}$. The self-optimizing output variables allow safety enforcement to the best possible degree. However, minor violations may still occur due to the soft constraint nature of this approach. In principle, one can enforce $\eta \in \mathcal{B}$ as a hard constraint (if deemed necessary) by defining a task-related working condition. Nevertheless, such a constraint will inevitably be made soft in a practical implementation.

D. Predicted Healthiness Index

As previously established, the robot must remain inside $\mathcal{B}$ to enforce safety. Based on that, at each sampling instant t_k, we use the state solution of Problem 1 at time t_{k-1} to compute the predicted healthiness (PH) index $\lambda_k : (0, r) \mapsto (0, 1)$, i.e., the blending mechanism. More precisely, we measure how far the robot is from violating the ball's boundary within a virtual horizon whose length is defined by $N_b \leq N$. Then, we select an index λ_k between 0 and 1 that is used to determine the weighting in Problem 1. The fact that N_b may differ from N allows the designer to choose the level of insight that the blending mechanism should have into the interaction. In what follows, we formally define the PH index.

Given a virtual horizon with N_b intervals and the predicted $\eta_{i|k-1}$ from the state solution of Problem 1 at time t_{k-1}, let us first select the largest residual within $\mathcal{B}$

$$d_{\text{res}|k} = \max_{i=0,\dots,N_b} \|\eta_{i|k-1} - \eta^r_{i|k-1}\| \quad (5)$$

and ensure that it does not outstrip the upper limit allowed

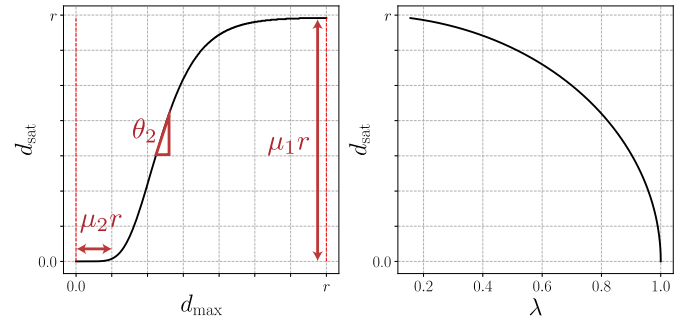$$d_{\text{max}|k} = \min(d_{\text{res}|k}, r). \quad (6)$$

Fig. 2. Left: pictorial description of the function (7). Dashed red lines indicate the bounds on $d_{\text{max}|k}$. Particularly, when $d_{\text{max}|k} = r$ then $d_{\text{sat}|k} = \mu_1 r$. Analogously, when $d_{\text{max}|k} = 0$ then $d_{\text{sat}|k} = \mu_2 r$. Right: an illustration of the function defined in (8) showing a profile relatively quadratic for penalizing residuals inside $\mathcal{B}$.

Then, to make sure the PH index is a strictly positive function in the range of interest, let us consider a saturation function $d_{\text{sat}} : [0, r] \mapsto (0, r)$ defined by the following Richards curve:

$$d_{\text{sat}|k}(d_{\text{max}|k}, \boldsymbol{\Lambda}) =$$

$$\frac{\mu_1 r}{(1 + \zeta \cdot \exp\{-\theta_2(d_{\text{max}|k} - \mu_2 r)\})^{1/\zeta}} \in (0, r), \quad (7)$$

where $\boldsymbol{\Lambda} = (\mu_1, \mu_2, \theta_2, \zeta)$ is a quadruple that parameterizes the curve (see Figure 2-left). In particular, μ_1 is a constant related to the the upper asymptote and whose value is close to 1 but strictly less than 1, μ_2 is a positive real number related to the lag phase, θ_2 is the growth rate, and ζ is a positive real number known as the shape parameter[1].

Definition 1 (PH index): The PH index is defined as

$$\lambda_k(d_{\text{sat}|k}) = \frac{\sqrt{r^2 - d^2_{\text{sat}|k}}}{r} \in (0, 1). \quad (8)$$

An illustration of function (8) is provided in Figure 2-right. Its profile indicates how the MI controller shapes human control authority. The outcome strongly depends on the combination of the r value and $\boldsymbol{\Lambda}$ tuning. For instance, one shall choose μ_1 as close to 1 as possible so that the extrema of λ get as close as possible to 0 and 1, expanding the range of human control authority. Factor μ_2 shall be selected relatively close to 0 as it influences the offset within $\mathcal{B}$ at which human control authority starts to decrease. The rate of decrease θ_2 is defined by the slope at the inflection point. The parameter $0 < \zeta \leq 1$ controls the sigmoid curvature. An in-depth analysis has shown that this parameter has little effect on the blending mechanism.

IV. NUMERICAL TESTBEDS AND DISCUSSION

A. Human-Quadrotor Benchmark

1) Robot: Let us denote with $\{\mathcal{I}\}$ the inertial frame, and with $\{\mathcal{B}\}$ the body frame located at the center of mass (CoM) of a quadrotor. The quadrotor state is composed by its position $\boldsymbol{p} = (x, y, z) \in \mathbb{R}^3$ expressed in $\{\mathcal{I}\}$, orientation $\boldsymbol{\gamma} = (\phi, \theta, \psi) \in \mathbb{R}^3$, linear velocity $\boldsymbol{v}_b = (v_x, v_y, v_z) \in \mathbb{R}^3$ expressed in $\{\mathcal{B}\}$,

---

[1]The shape of the d_{sat} curve depends on ζ. If $\zeta = 1$, one has the *logistic function*. If ζ tends to zero, the curve converges to the *Gompertz function*.

TABLE I
MikroKopter Physical Parameters

m	1.04 kg	l	0.23 m	C_D	10 Nm/kHz^2
C_T	595 N/kHz^2	d	0.5 mN·m·s	J_m	0.08 g·m^2
$\boldsymbol{J} = \mathrm{diag}(0.01, 0.01, 0.07)$ kg·m^2					

angular rate $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ expressed in $\{\mathcal{B}\}$ and rotational speed of the propellers $\boldsymbol{\Omega} = (\Omega_1, \Omega_2, \Omega_3, \Omega_4) \in \mathbb{R}^4$, bounded by $\mathcal{X} = \{\boldsymbol{\Omega} \in \mathbb{R}^4 : \underline{\boldsymbol{\Omega}} \leq \boldsymbol{\Omega} \leq \bar{\boldsymbol{\Omega}}\}$ (vector inequalities are intended component-wise). The control inputs are the rotor torques, defined by $\boldsymbol{u} = (\tau_1, \tau_2, \tau_3, \tau_4) \in \mathbb{R}^4$ and constrained in magnitude by $\mathcal{U} = \{\boldsymbol{u} \in \mathbb{R}^4 : \underline{\boldsymbol{u}} \leq \boldsymbol{u} \leq \bar{\boldsymbol{u}}\}$. The nonlinear dynamics is then given by the first-order ordinary differential equations:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}, \boldsymbol{u}) = \begin{pmatrix} \boldsymbol{S}^T \boldsymbol{v}_b \\ \boldsymbol{E}\boldsymbol{\omega} \\ \frac{1}{m}\boldsymbol{F}_b - \boldsymbol{S}\boldsymbol{G} - \boldsymbol{\omega} \times \boldsymbol{v}_b \\ \boldsymbol{J}^{-1}(\boldsymbol{M}_b - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}) \\ \frac{1}{J_m}(\boldsymbol{u} - C_D\boldsymbol{\Omega} \odot \boldsymbol{\Omega} - d\boldsymbol{\Omega}) \end{pmatrix} \quad (9)$$

with state $\boldsymbol{\xi} = (\boldsymbol{p}, \boldsymbol{\gamma}, \boldsymbol{v}_b, \boldsymbol{\omega}, \boldsymbol{\Omega}) \in \mathbb{R}^{16}$. The quadrotor's mass is denoted by m, vector $\boldsymbol{G} = (0, 0, g) \in \mathbb{R}^3$ with g being the gravitational acceleration; d represents the drag coefficient of the rotor whose scalar inertia is J_m. The element-wise product is denoted with $\odot$. The positive-definite matrix $\boldsymbol{J} \in \mathbb{R}^{3 \times 3}$ denotes the vehicle's inertia matrix. The rotation matrix from $\{\mathcal{I}\}$ to $\{\mathcal{B}\}$ is represented by $\boldsymbol{S} \in SO(3)$. Matrix $\boldsymbol{E} : \mathbb{R}^3 \to \mathbb{R}^{3 \times 3}$ expresses the relation between the instantaneous rates of change of $\boldsymbol{\gamma}$ and the instantaneous components of $\boldsymbol{\omega}$. The total external forces and moments applied to the CoM of quadrotor and expressed in $\{\mathcal{B}\}$ are defined, respectively, as

$$\boldsymbol{F}_b := (0, 0, F_z), \quad \boldsymbol{M}_b := (M_x, M_y, M_z) \quad (10)$$

with
$$F_z = C_T(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \quad (11a)$$

$$M_x = C_T \cdot l(\Omega_2^2 - \Omega_4^2), \quad (11b)$$

$$M_y = C_T \cdot l(\Omega_3^2 - \Omega_1^2), \quad (11c)$$

$$M_z = C_D(\Omega_1^2 + \Omega_3^2 - \Omega_2^2 - \Omega_4^2), \quad (11d)$$

where C_T is the thrust coefficient, C_D is the drag coefficient, and l is the distance between the quadrotor's CoM and the rotor's center.

As an example, we report in Table I the values of the dynamic parameters appearing in (9) corresponding to the MikroKopter[2] quadrotor platform. Note that we perform model scaling as MPC uses normalized input/output variables. Other than equaling/approximating the variables' range, scaling provides further benefits. Among them, a more intuitive tuning of the weights associated with the human and actual robot costs, allowing the designer to emphasize the relative priority of each term rather than the combination of priority and signal scale. It also offers improved numerical conditioning as optimization routines are less prone to round-off errors.

---

[2]https://www.mikrokopter.de/en/home

2) Reference Trajectory: An offline planner provides a Cartesian helical trajectory $\boldsymbol{\eta}^r$ that avoids collisions with existing obstacles. This means that task variables are defined as $\boldsymbol{\eta} := \boldsymbol{p} \in \mathbb{R}^3$. Note that the reference trajectory implicitly enforces safety.

3) Pilot Inputs: Usually, pilots provide the quadrotor with the desired roll, pitch, z angular rate, and total thrust, i.e., $\boldsymbol{g} = (\phi_h, \theta_h, \omega_{z,h}, F_{z,h}) \in \mathbb{R}^4$. For simplicity, we assume that the desired z angular rate is zero and the total thrust is mapped into the speed of the propellers, i.e., $\boldsymbol{\nu}^r : \boldsymbol{g} \mapsto \mathbb{R}^6$.

Literature suggests that MPC provides a favorable basis for shaping the human decision-making process (see [18] and references therein). Supported by these findings, we use a second NMPC controller to simulate the inputs of a human pilot. Its finite-time OCP reads as follows:

$$\min_{\boldsymbol{\xi}_k, \boldsymbol{u}_k} \sum_{i=0}^{N-1} \frac{1}{2}(\Delta\boldsymbol{\eta}_{i|k}^T \boldsymbol{Q}_1 \Delta\boldsymbol{\eta}_{i|k} + \boldsymbol{\gamma}_{i|k}^T \boldsymbol{Q}_2 \boldsymbol{\gamma}_{i|k} + \boldsymbol{v}_{b,i|k}^T \boldsymbol{Q}_3 \boldsymbol{v}_{b,i|k} +$$
$$\boldsymbol{\omega}_{i|k}^T \boldsymbol{Q}_4 \boldsymbol{\omega}_{i|k} + \Delta\boldsymbol{\Omega}_{i|k}^T \boldsymbol{Q}_5 \Delta\boldsymbol{\Omega}_{i|k} + \boldsymbol{u}_{i|k}^T \boldsymbol{R}_1 \boldsymbol{u}_{i|k}) +$$
$$\frac{1}{2}(\Delta\boldsymbol{\eta}_{k+N|k}^T \boldsymbol{Q}_{1N} \Delta\boldsymbol{\eta}_{k+N|k} + \boldsymbol{\gamma}_{k+N|k}^T \boldsymbol{Q}_{2N} \boldsymbol{\gamma}_{k+N|k} +$$
$$\boldsymbol{v}_{b,k+N|k}^T \boldsymbol{Q}_{3N} \boldsymbol{v}_{b,k+N|k} + \boldsymbol{\omega}_{k+N|k}^T \boldsymbol{Q}_{4N} \boldsymbol{\omega}_{k+N|k} +$$
$$\Delta\boldsymbol{\Omega}_{k+N|k}^T \boldsymbol{Q}_{5N} \Delta\boldsymbol{\Omega}_{k+N|k}) \quad (12)$$

$$\text{s.t. } (2b) - (2e).$$

Therein, $\Delta\boldsymbol{\Omega}_{i|k} = \boldsymbol{\Omega}_{i|k} - \boldsymbol{\Omega}_{\text{hov}}$ for $i = 0, \ldots, N$ represents the propeller speed error, with $\boldsymbol{\Omega}_{\text{hov}} = \sqrt{mg/4C_T}$. The weighting matrices are denoted by $\boldsymbol{Q}_j$, $\boldsymbol{Q}_{jN} \succ 0$ for $j = 1, \ldots, 5$ and $\boldsymbol{R}_1 \succ 0$. In particular, constraint (2d) represents the control interface's real limitation.

At each sampling interval $[t_k, t_{k+1}]$, the solution of (12) is computed and the predicted state at stage $i = 1$ is used as pilot inputs, namely, $\phi_h = \phi_{1|k}$, $\theta_h = \theta_{1|k}$, $\boldsymbol{\Omega}_h = \boldsymbol{\Omega}_{1|k}$.

Remark 1: In practice, the total thrust coming from the joystick can be easily mapped into the propellers' desired speed and passed to the MI controller if one uses Eq. (11a) assuming hovering condition, i.e., $\Omega_1 = \Omega_2 = \Omega_3 = \Omega_4 = \Omega_{\text{ss}}$, yielding

$$\Omega_{\text{ss}} = \sqrt{F_{z,h}/4C_T}, \quad \boldsymbol{\Omega}_h = \Omega_{\text{ss}} \cdot \mathbf{1}_4.$$

Different skill levels are obtained by shaping the weighting matrices of the NLP (12). Flying a quadrotor is less automated for novice pilots and, for this reason, requires more of their attention span than experienced ones. Due to their limited self-regulatory ability, their inputs tend to be oscillatory. On the contrary, experienced pilots' inputs tend to be more precise, presumably reflecting their ability to use sensory cues to support their actions. These behaviors were emulated considering distinct tuning for matrices $\boldsymbol{Q}_j$, $\boldsymbol{Q}_{jN}$. Experienced pilots are sketched by an improved tracking performance with a heavily weighted sum of squares. Oppositely, novice pilots underperform using relatively small weights that allow for larger reference deviations.

Assumption 1: Since, in practice, we need to predict human inputs to solve Problem 1, we chose to use a zero-order hold (ZOH) method. This prediction method assumes that future

human inputs will all be the same as current ones, an assumption that has proved effective in experimentation, [19].

4) Additional Penalty Terms: As previously hypothesized, pilots would typically perform continuous maneuvers by changing their inputs all the time. To preserve smoothness in the generated trajectory, we consider additional penalty terms in the MI controller cost function. These terms rely on the following output functions

$$h(\boldsymbol{\xi}_{i|k}) = h_N(\boldsymbol{\xi}_{k+N|k}) = (\psi; \boldsymbol{v}_b; \boldsymbol{\omega}) \in \mathbb{R}^7 \qquad (13)$$

(where the semicolon denotes vector concatenation) and their corresponding references

$$\boldsymbol{e}_{i|k}^r = \boldsymbol{e}_{k+N|k}^r = \boldsymbol{0}_7. \qquad (14)$$

The vector of zeros implies two underlying assumptions: first, we are not dealing with aerobatic maneuvers; second, we assume it is an educated guess to initialize and, thereby, speed up the subsequent optimization algorithm.

5) NLPs Parameterization: Among several approaches available to discretize continuous-time OCPs, this work will use direct multiple shooting [20] due to its convergence and initialization properties. Assuming an equidistant grid and piecewise constant control parameterization, the following initial value problem defines the state trajectory $\boldsymbol{\xi}(\pi)$ at each shooting interval:

$$\dot{\boldsymbol{\xi}}(\pi) = \boldsymbol{f}(\boldsymbol{\xi}(\pi), \boldsymbol{u}_{i|k}), \quad \boldsymbol{\xi}(t_i) = \boldsymbol{\xi}_{i|k}, \quad \pi \in [t_i, t_{i+1}]. \quad (15)$$

Eq. (15) is evaluated numerically, $\boldsymbol{\xi}(t_{i+1}) \approx \boldsymbol{F}(\boldsymbol{\xi}_{i|k}, \boldsymbol{u}_{i|k})$, using a single-step explicit Runge Kutta of 4th order per Δt. Additionally, we approximate the Hessian of the Lagrangian using the Generalized Gauss-Newton method so that structure-exploiting convex solvers can make the most of the block structure of the resulting quadratic program (QP).

B. Implementation Details

1) NMPC Via RTI Scheme: In real-time NMPC applications, Problem 1 needs to be solved at each sampling instant under the available computation time. To that end, we use a numerical strategy based on sequential quadratic programming (SQP) that relies on the solution of a limited number of QP subproblems, the so-called RTI scheme [21]. More precisely, only one linearization and QP solve are carried out per sampling instant, leading to an approximate feedback control policy. An essential element in the RTI scheme is to keep the initial state $\boldsymbol{\xi}_{0|k}$ as a constrained decision variable, often referred to as *initial value embedding*. This trait allows to divide computations into preparation and feedback phase, where the former is typically more expensive. In this work we use the RTI method's implementation through the high-performance software package `acados` [22]. Through its Python template-based interface, we generate the library that implements Problem 1, which is then wrapped by our framework written in Python 3.7.

2) Structure-Exploiting QP Solver and Condensing Approach: The QP subproblems arising in the SQP algorithm in `acados` are addressed using the high-performance `HPIPM` [23] solver, which implements a primal-dual interior-point method. It is built on top of the linear algebra package `BLASFEO` [24], finely tuned for multiple CPU architectures. In particular, `BLASFEO` is hardware-optimized for the moderately sized matrices present in our mixed-initiative NMPC. We use its `X64_INTEL_HASWELL` implementation, which exploits a set

TABLE II
PARAMETERS USED BY THE BLENDING MECHANISM AND THE MIXED-INITIATIVE NMPC

$\boldsymbol{\Omega}$	$\boldsymbol{0}_4$	$\bar{\boldsymbol{\Omega}}$	$0.09 \cdot \boldsymbol{1}_4$ kHz	μ_1	0.99
$\underline{\boldsymbol{u}}$	$-0.1285 \cdot \boldsymbol{1}_4$ Nm	$\bar{\boldsymbol{u}}$	$0.1285 \cdot \boldsymbol{1}_4$ Nm	μ_2	0.3
t_N	0.75 s	N	50	θ_2	55
r	0.175 m	N_b	10	ζ	$1 \cdot 10^{-13}$

$$\boldsymbol{Q}_\eta = \boldsymbol{Q}_{\eta_N} = \text{blkdiag}(100 \cdot \boldsymbol{I}_2, 200), \; \boldsymbol{Q}_\nu = \boldsymbol{Q}_{\nu_N} = 800 \cdot \boldsymbol{I}_6,$$
$$\boldsymbol{Q}_e = \boldsymbol{Q}_{e_N} = \text{blkdiag}(10, 3 \cdot \boldsymbol{I}_5, 10), \; \boldsymbol{R} = 70 \cdot \boldsymbol{I}_4$$

of vectorized instructions for the target CPU. Solution times are further reduced by reformulating the QPs resulting from our NMPC using the efficient partial condensing algorithm implemented in `HPIPM`. Specifically, the algorithm finds the optimal sparsity level for the solver, trading off horizon length for input vector size.

C. Performance Analysis

The parameters used for all the testbeds in the blending mechanism and the mixed-initiative NMPC are in Table II. In particular, we adopt a short virtual horizon N_b to compensate for the fact that with a ZOH method, pilot inputs would have a natural tendency to violate the norm ball's boundary in the long run. The chosen value is a compromise between the base level of "trust" in the human inputs and the switching frequency of control authority. Another assumption is that $\lambda = 0.5$ during the first iterations, i.e., control authority is equally distributed. As previously mentioned, the testbeds incorporate a second autonomous algorithm that emulates a novice and an experienced pilot. This will enable a comparative analysis of our approach.

Figure 3 displays the trajectories generated by the MI control algorithm for both pilots. The results show that the controller maintained the quadrotor inside $\mathcal{B}$, successfully avoiding all obstacles in all cases (see also Fig. 5-left). In Figure 4, we observe that the mixed-initiative NMPC commands closely follow the pilot inputs when the safety constraint is unlikely to be violated. This outcome is more evident for the experienced pilot, where the lines overlap almost completely, implying that the inputs issued kept the quadrotor closer to the reference trajectory. From Figure 5-right, it is obvious that the blending mechanism begins to drop off rapidly the level of control authority for the novice. At the same time, it maintains a relatively high level for the experienced pilot. Overall, note that the control algorithm assists the novice through what is often a very challenging part of their learning journey, the damping in pitching-rolling motions. The algorithm can keep up considerably with the signals' pattern (preserving the pilot's primary intentions), but attenuates their magnitude to cope with the safety constraint. Also, as the physical constraints herein regarded are linear, they can be enforced very efficiently by the optimization algorithm (see Figure 6).

Including maximum deviations provides versatility in the quadrotor motion generation. First, it gives a certain level of spatial freedom so that the quadrotor can leverage the motion by exploiting its dynamics. Second, it similarly adds new degrees of freedom to the controller. From a theoretical point of view, the control objective of the mixed-initiative NMPC can be seen as a target set (in the space of the task variables) instead of a target point, since inside $\mathcal{B}$ there are no preferences between one point and another.
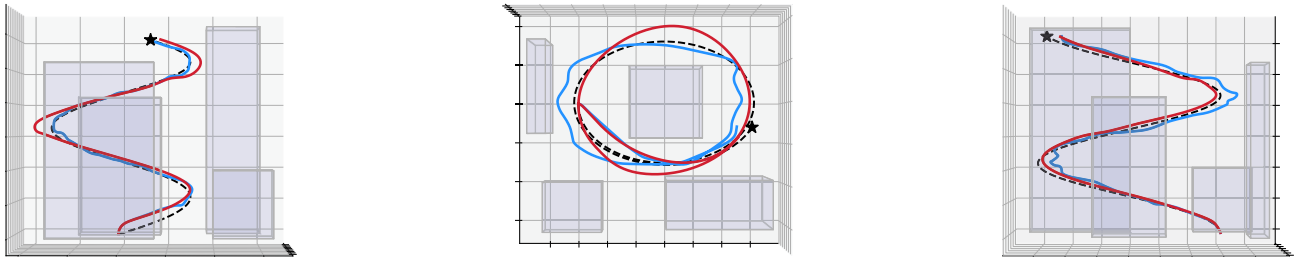
Fig. 3. Generated trajectories among obstacles (purple) using our mixed-initiative NMPC: novice pilot (blue); experienced pilot (red); reference trajectory is dashed; the ⋆ marks the final goal.
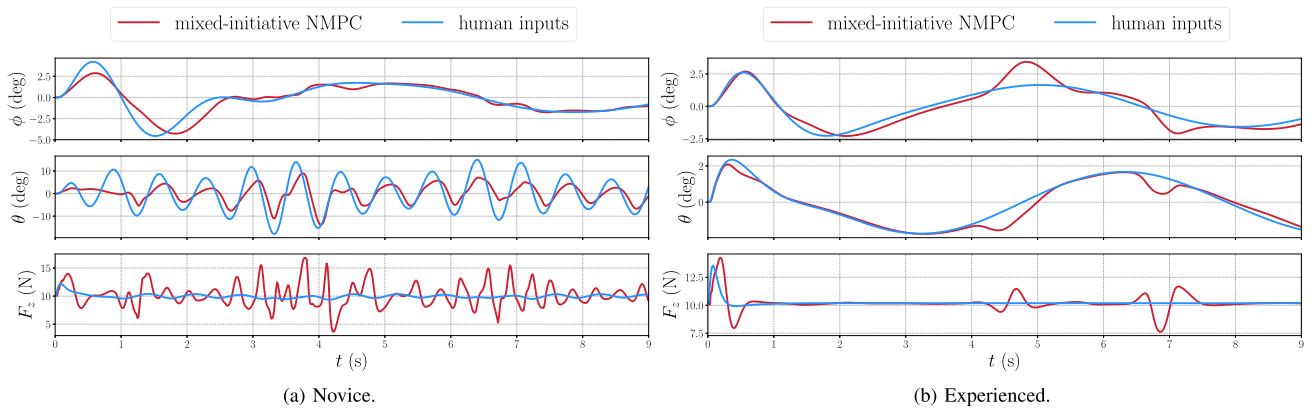

(a) Novice.

(b) Experienced.

Fig. 4. Pilot inputs generated by an autonomous algorithm and mixed-initiative NMPC commands as a result of the blending mechanism.
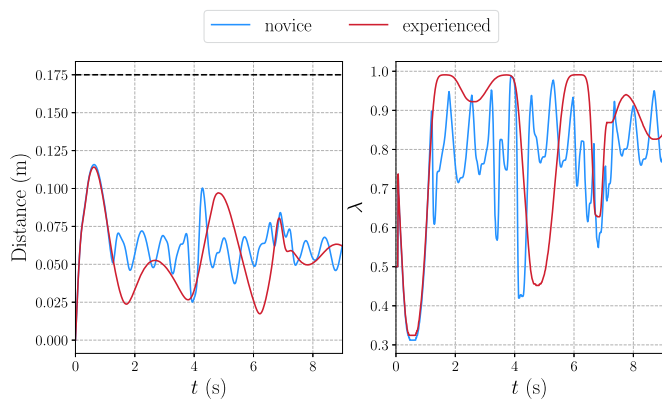

Fig. 5. Left: distance between generated trajectories and reference; r is dashed. Right: predicted healthiness index profiles indicating the level of human control authority.

(a) Novice.

(b) Experienced.

Fig. 6. Generated propeller speeds; the upper bound $\bar{\boldsymbol{\Omega}}$ is dashed.

The findings here indicate that the proposed algorithm provides pilots with more control authority without sacrificing safety or even exceeding the robot's physical limits. Therefore, it has the potential to pave the way to the definition of a standard assistive scheme that can lead to a future improvement in pilot decision-making performance.

D. Computational Burden

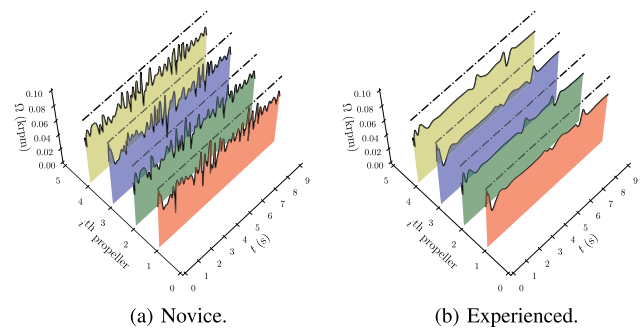Solution times reported are from an Intel Core i5-4288U @ 2.6 GHz running macOS Catalina. For a horizon of $N = 50$, the resulting QP has 1016 optimization variables. The testbeds showed that HPIPM is significantly faster in solving the partially condensed QP than the corresponding dense QP. In this context, computational efficiency is granted through partial condensing as it allows us to exploit hardware throughput better. The average times per SQP-iteration in acados were 4.27 ms for the novice and 3.87 ms for the experienced pilot. Interestingly, one can modify N_b and potentially improve the blending mechanism performance without affecting solution times significantly. Figure 7 shows that changes in N_b affect marginally the solution times. Note that cases in which solution time is larger than Δt or the solver fails are handled using the previous feasible solution at stage $i = 1$.
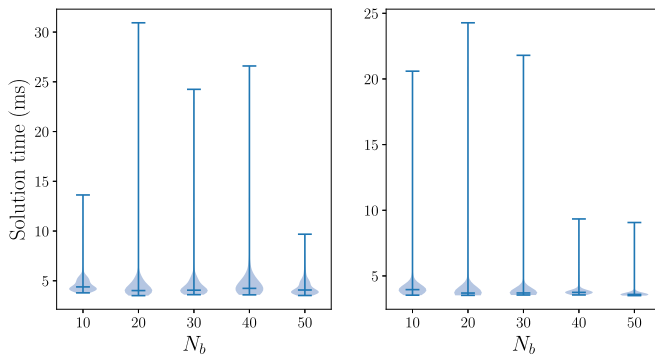
Fig. 7. Violin plot representations of solution times associated with the mixed-initiative controller in simulations with a novice (left) and an experienced pilot (right). Central bars represent the mean values for each virtual horizon N_b.

V. Conclusion

This letter presented a novel mixed-initiative control algorithm based on NMPC to enforce safety in human-quadrotor interactions. The algorithm adopts a predict-then-blend approach to mix human inputs and motion generator commands during operation. It assesses whether safety rules are met to perform the control authority blending through the quadrotor's predicted positions. Then it uses an efficient RTI-based scheme that iteratively refines the combined costs so that all NMPC constraints are fulfilled within the available computation time. Simulations showed that the approach allows safety enforcement with specific benefits for useful assistance to pilots, especially novices, and low computational effort. Thanks to high-performance software implementations, we could further speed up solution times and achieve computational efficiency.

Future work will focus on the inclusion of an actual human in the loop by integrating a joystick and a graphical user interface to the testbed environment. The subsequent phase will encompass the integration of a real quadrotor. For this setting, we want to develop a set of metrics that will help us assess pilots' learning curves and provide us with real-world evidence of improved proficiency. Besides, it is desirable to investigate the stability of the MI controller.

References

[1] J. M. Maciejowski, *Predictive Control: With Constraints*. Harlow, Essex, England: Pearson Education Limited, 2002, pp. 145–163.

[2] N. Amirshirzad, A. Kumru, and E. Oztop, "Human adaptation to human-robot shared control," *IEEE Trans. Human-Mach. Syst.*, vol. 49, no. 2, pp. 126–136, Apr. 2019.

[3] X. Bao, V. Molazadeh, A. Dodson, B. E. Dicianno, and N. Sharma, "Using person-specific muscle fatigue characteristics to optimally allocate control in a hybrid exoskeleton-preliminary results," *IEEE Trans. Med. Robot. Bionics*, vol. 2, no. 2, pp. 226–235, May 2020.

[4] C. Huang, F. Naghdy, H. Du, and H. Huang, "Shared control of highly automated vehicles using steer-by-wire systems," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 2, pp. 410–423, Mar. 2019.

[5] Y. Lu, L. Bi, and H. Li, "Model predictive-based shared control for brain-controlled driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 2, pp. 630–640, Feb. 2020.

[6] C. Nam, P. Walker, H. Li, M. Lewis, and K. Sycara, "Models of trust in human control of swarms with varied levels of autonomy," *IEEE Trans. Human-Mach. Syst.*, vol. 50, no. 3, pp. 194–204, Jun. 2020.

[7] R. Rahal *et al.*, "Caring about the human operator: Haptic shared control for enhanced user comfort in robotic telemanipulation," *IEEE Trans. Haptics*, vol. 13, no. 1, pp. 197–203, Jan.–Mar. 2020.

[8] V. Girbes-Juan, V. Schettino, Y. Demiris, and J. Tornero, "Haptic and visual feedback assistance for dual-arm robot teleoperation in surface conditioning tasks," *IEEE Trans. Haptics*, vol. 14, no. 1, pp. 44–56, Jan.–Mar. 2021.

[9] R. Moccia, C. Iacono, B. Siciliano, and F. Ficuciello, "Vision-based dynamic virtual fixtures for tools collision avoidance in robotic surgery," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1650–1655, Apr. 2020.

[10] C. Masone, M. Mohammadi, P. Robuffo Giordano, and A. Franchi, "Shared planning and control for mobile robots with integral haptic feedback," *Int. J. Robot. Res.*, vol. 37, no. 11, pp. 1395–1420, 2018.

[11] A. Franchi, "Human-collaborative schemes in the motion control of single and multiple mobile robots," in *Trends in Control and Decision-Making for Human-Robot Collaboration Systems*. Switzerland: Springer, 2017, pp. 301–324.

[12] A. Moortgat-Pick, A. Adamczyk, T. Tomić, and S. Haddadin, "Feeling the true force in haptic telepresence for flying robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 9789–9796.

[13] T. Kastritsi, D. Papageorgiou, I. Sarantopoulos, Z. Doulgeri, and G. A. Rovithakis, "Stability of active constraints enforcement in sensitive regions defined by point-clouds for robotic surgical procedures," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 1604–1609.

[14] B. Weber, R. Balachandran, C. Riecke, F. Stulp, and M. Stelzer, "Teleoperating robots from the international space station: Microgravity effects on performance with force feedback," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 8144–8150.

[15] S. Fani *et al.*, "Simplifying telerobotics: Wearability and teleimpedance improves human-robot interactions in teleoperation," *IEEE Robot. Automat. Mag.*, vol. 25, no. 1, pp. 77–88, Mar. 2018.

[16] B. Barros Carlos *et al.*, "An efficient real-time NMPC for quadrotor position control under communication time-delay," in *Proc. 16th Int. Conf. Control, Automat., Robot. Vis.*, 2020, pp. 982–989.

[17] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4459–4466, Oct. 2019.

[18] J. R. Anderson and B. Ayalew, "A cascaded optimization approach for modeling a professional driver's driving style," *J. Dyn. Syst., Meas., Control*, vol. 142, no. 9, pp. 091004:1–091004:10, 2020.

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

[20] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," in *Proc. 9th IFAC World Congr.*, 1984, vol. 17, pp. 1603–1608.

[21] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, 2005.

[22] R. Verschueren *et al.*, "Acados: A modular open-source framework for fast embedded optimal control," pp. 1–37, 2019. [Online]. Available: https://arxiv.org/abs/1910.13753

[23] G. Frison and M. Diehl, "HPIPM: A high-performance quadratic programming framework for model predictive control," pp. 1–7, 2020. [Online]. Available: https://arxiv.org/abs/2003.02547

[24] G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, and M. Diehl, "BLASFEO: Basic linear algebra subroutines for embedded optimization," *ACM Trans. Math. Softw.*, vol. 44, no. 4, pp. 1–30, 2018.