CPI: Conservativeness, Permissiveness and Intervention Metrics for Shared Control Evaluation

Yu Zhou[1] and Kris Hauser[2]

***Abstract*— This paper presents an approach to measure the ability of a shared control system to track user input while simultaneously ensuring safety. These CPI metrics, based on reachability theory, are used to measure the Conservativeness (C), Permissiveness (P) and the amount of Intervention (I) applied to a user nominal control. The metrics apply to arbitrary dynamic systems, state and control constraints, and unlike other existing metrics, they apply to non-differentiable shared controllers including controllers implemented in procedural code. Moreover, we propose a parallel algorithm based on Rapidly-exploring Random Trees (RRTs) for conducting the reachability analysis necessary for computing conservativeness and permissiveness metrics efficiently. We demonstrate how CPI metrics may be used to evaluate a Linear-Quadratic Regulator (LQR) and two different Model Predictive Controller (MPC) based safe shared controllers applied to the cartpole system, for different control parameters.**

I. Introduction

Shared control blends human and autonomous control, wherein the controller filters a human user input to generate the system control to provide additional safety or task assistance. It is an important component in robot teleoperation [1], assistive robotics (e.g. robotic wheelchairs [2]), and driver assistance in automobiles [3].

Intervention is the amount by which the system controls differ from the nominal user controls. A controller that intervenes affects the system's global *reachability*. There is an inherent trade-off between safety and performance on the one hand, and intervention and reachability on the other. On the one hand, a shared controller can be overly cautious and restrictive, preventing the user from reaching all unsafe states but also limiting the operator's control over the system. For example, a collision avoidance braking system that limits a vehicle to 10 kph is quite safe, but prevents the vehicle from reaching its operational limits. On the other hand, a controller may be too permissive and allow a careless, negligent or adversarial user to reach unsafe states. For example, a collision avoidance system that only partially slows the vehicle down before a collision is reliant on the human to provide safety. However, existing shared control evaluation methods have focused primarily on task performance, e.g. task success rate, completion time or efficiency in terms of control effort applied, or user preference [4], [5], [6], [7]. These metrics are system-specific, may require subjective surveys, and thus cannot serve as a common language between systems and controllers. The goal of this work is therefore to provide quantitative and system-independent metrics to evaluate the safety-performance trade-offs in designing shared controllers.

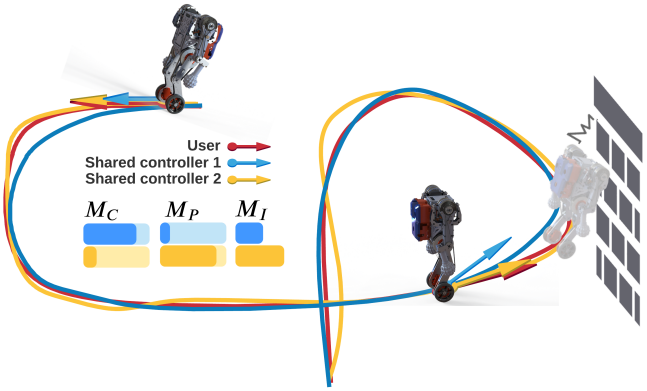


Fig. 1. Qualitative illustration of the CPI metrics (M_C, M_P, M_I). The user velocity command and trajectory are shown as red arrows and curve, respectively. The outputs of two hypotethical shared controllers are drawn in blue and yellow. At the start (top left), the user issues a safe but aggressive command. Controller 1 is more conservative and does not follow the command closely, leading to higher M_C. Then, the user mistakenly drives the robot toward wall obstacle (right). Controller 2 provides the operator freedom to let the robot collide, resulting in higher M_P, while Controller 1 avoids the collision. Based on the overall trajectories, Controller 1 stays slightly closer to the user command, leading to lower M_I. [Best viewed in color.]

We propose the metrics of *conservativeness* and *permissiveness* using concepts from reachability theory. These are dimensionless quantities in the range [0,1] and are independent of the user behavior. Qualitatively, a shared controller can be described as conservative if it leverages only a small portion of the system's reachable viable set and as permissive if it allows driving the system into states that would not be able to reach the safe set. The *intervention* metric is inversely proportional to how much control authority the user has over the system, and low values are preferable if the system designer trusts the operator's expertise and attentiveness, and wants to minimize surprise to the operator. These metrics generalize to any system and serve as a basis upon which fair comparison and understanding between shared controllers can be made. They may then be used to design and select the shared controller most appropriate to given requirements (Fig. 1).

The C and P metrics require calculating reachable and viable sets under the given shared controller, which are

This work is partially supported by NSF NRI Grant #2024775.

[1]Yu Zhou is with the Department of Electrical Computer Engineering, University of Illinois at Urbana-Champaign, USA yuzhou7@illinois.edu

[2]Kris Hauser is with the Department of Computer Science, University of Illinois at Urbana-Champaign, USA kkhauser@illinois.edu

called the *controller-dependent reachable and viable sets*. Although past work has addressed reachable and viable set computation using Hamilton-Jacobi (HJ) methods, applying these methods in our setting requires the controller to be differentiable. To calculate reachable and viable sets of more complex controllers, we introduce a reachability estimation approach based on a parallel RRT-based algorithm.

We evaluate the CPI metrics on an LQR and MPC-based safe shared controllers applied to the cartpole system for different control parameters. In particular, our approach can be applied to a minimum intervention MPC, which performs a finite-horizon trajectory optimization to minimize intervention while maintaining safety, whereas prior HJ methods do not apply.

II. Related work

1) Safe Shared Control: The safe shared controller can be decomposed into two parts: potentially unsafe human input and a shared controller which tracks user input in a minimum intervention manner and only modifies it when considered unsafe. The safety to be verified is stability certification and constraint set certification [8].

Stability verification concerns closed-loop stability. Methods such as Sums-Of-Squares (SOS) to search for Lyapunov functions can be used to verify safety by constructing funnels [9] with Lyapunov properties. However, finding a Lyapunov function for a system is nontrivial. Constraint set certification tries to find a policy that keeps the system inside a control invariant safe set (CIS), the set of initial states for which there exists a controller such that the system constraints are never violated [10], when the human input is considered unsafe. This can be achieved through a safety filter style controller [11]. One approach under this framework is an Active Set Invariance Filter (ASIF) based on control barrier functions (CBFs) [12], which puts the nominal control input through a quadratic program to ensure it obeys certain constraints that define the safe set of the system. However, it is only pointwise optimal, and finding valid CBFs for a general dynamical system is generally challenging. Another predictive safety filter approach is given in [13], which guarantees the safety of a learned controller by using a predictive controller to find the closest control that is safe.

2) Shared Control Evaluation: Various metrics have been applied to evaluate shared control. Carlson et al. evaluate a robotic wheelchair in terms of performance, attention and workload with emphasis on the human factor [4]. Tee et al. introduce metrics for teleoperation task performance on a curved object surface [5] such as task duration, normalized error, jerk, and user experience using the NASA TLX questionnaire. Broad et al. use the average observed deviation between the user input and the closest safe signal as well as the average percentage of sampled rollouts that are safe at each timestep as safety metrics to evaluate a shared controller [7]. Oh et al. propose four quantitative metrics for obstacle avoidance tasks: task duration, travelled distance, minimum proximity to the obstacle and the cosine distance between controls [6]. Although some existing metrics capture user behavior and objective measures of safety, our work provides a quantitative, control-theoretic and system-independent framework upon which to evaluate shared control.

3) Safety verification and viability: The control literature has studied safety verification and viability checking extensively. The standard safety verification problem focuses on proving whether there exist trajectories entering a set of forbidden or unsafe states through *forward* reachability analysis [14]. Viability checking, on the other hand, is a *backward* reachability problem that involves finding all the states from which a safe set can be reached [15].

The conservativeness and permissiveness metrics require computing forward and backward reachable sets. There are multiple ways to do so. *Hamilton-Jacobi* (HJ) methods solve a partial differential equation to give an over-approximation of the reachable sets [16]. This technique however requires differentiable dynamics and suffers from a time complexity which increases exponentially with the state dimension. Recent work decomposes the computation of a reachable set into several smaller dimensions [17], but suffers from an over-approximation that worsens in higher dimensions and requires knowledge of how to decompose the system appropriately. *Set propagation* over-approximates the reachable set using polygonal approximations, and benefits from existing software toolboxes [18]. However, accurate set propagation for nonlinear systems is still a challenging problem and an active area of research [19]. Finally, reachable sets can also be computed with sampling-based methods, which require intelligent sampling strategies to obtain better coverage. For example, Lew et al. use an adversarial strategy to sample states that can generate a larger convex set [20]. This comes at the expense of an over-approximation of reachable sets that becomes non-negligible when those are highly non-convex.

We opt for a sampling-based method based on the RRTs framework [21], [22]. Prior work has used RRT as a falsification method focusing on generating a set of test scenarios that cause the system to fail [21]. A similar algorithm, R3T [22], uses reachable set approximations to improve the RRT distance metric and achieve faster convergence speeds. We build on these approaches for shared controller-dependent reachable and viable set computation in this work.

III. Safe Shared Control Problem formulation

Let S be the system upon which we design a shared controller. The dynamics of S are modeled by $\dot{x} = f(x, u)$ with states $x \in \mathbb{R}^n$ and controls $u \in \mathbb{R}^m$. Let $\mathcal{X}$ be the set of feasible states, and $\mathcal{U}$ the set of admissible controls, i.e. the set of states the system is allowed to be in and the set of controls it is allowed to execute, respectively.

A shared controller takes the system's current state x and a human input π_h to generate a safe control policy π_s:

$$\pi_s \equiv \pi_s(x, \pi_h) : \mathcal{X} \times \mathcal{U}_h \to \mathcal{U} \tag{1}$$

where $\mathcal{U}_h$ is the space of human inputs. The human input is interpreted as being generated by a controller $\pi_h \equiv \pi_h(x, t)$ which is usually unknown to the shared controller π_s.

1) Safety: Let $\mathcal{X}_0 \subseteq \mathcal{X}$ be the initial set, i.e. the set of feasible states the system may start in, and $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$ the *safe set* , which refers to a set of feasible states in which a known auxiliary controller (e.g. LQR) is guaranteed to maintain the system: $\mathcal{X}_{\text{safe}}$ is chosen to be a system-specific CIS, and may be conservatively set as a small region of feasible states near the equilibrium, e.g. the set of feasible states with zero velocity.

2) Human input: The space of user controls $\mathcal{U}_h$ is typically a user interface design choice. For instance, if the user can drive the system's controls directly, where $\pi_h := u^h$, then $\mathcal{U}_h = \mathcal{U}$. In addition to direct control, other more intuitive and practical interfaces may designate position, velocity, higher-order targets or a combination thereof as the user command: $\pi_h := x^h$. For example, if the system state $x = (q, \dot{q})$ consists of a configuration q and its derivative $\dot{q}$, then a position control scheme sets $\mathcal{U}_h$ to a $(n/2)$-D subspace of $\mathcal{X}$.

3) Objectives: The goal of a safe shared controller is to design a policy which follows the human operator's command π_h as closely as possible and guarantees safety. To that end, we define an *intervention objective* $\ell_{\text{int}}(\cdot)$. Assuming direct control u^h, $\ell_{\text{int}}(\cdot)$ can for example be defined as: $\ell_{\text{int}}(x, u, \pi_h) = \|u - u^h\|_R^2 := (u - u^h)^\top R(u - u^h)$, where R is a cost matrix. If the user control is interpreted as defining a target state x^h, then ℓ_{int} can be defined as: $\ell_{\text{int}}(x, u, \pi_h) = \|x - x^h\|_Q^2$, where the cost matrix Q weighs the importance of different state dimensions.

The *minimum intervention shared control* (MISC) [7] problem can be therefore formulated as the following optimization problem, assuming discrete time:

$$\begin{aligned} \underset{\mathbf{x}[\cdot],\mathbf{u}[\cdot]}{\text{minimize}} \quad & \sum_{t=0}^{T-1} \ell_{\text{int}}(x_t, u_t, \pi_h(x_t, t)) \\ \text{subject to} \quad & \forall t, x_{t+1} = f(x_t, u_t), \\ & x_0 = x(0), x_t \in \mathcal{X}, u_t \in \mathcal{U}, \\ & \text{Additional constraints.} \end{aligned} \tag{2}$$

with the time horizon $T \in \mathbb{R}_+ \cup \{+\infty\}$, $\mathbf{x}[\cdot] = (x_0, x_1, \dots)$ and $\mathbf{u}[\cdot] = (u_0, u_1, \dots)$. Safety is encoded through state and control constraints with the sets $\mathcal{X}$ and $\mathcal{U}$. Additional constraints may include other artificial state and control constraints, but they are not required in this definition.

MISC is an idealized goal; Eq. 2 is not actually solvable in practice since *we do not have access to the user's control policy* π_h. Instead, the system can only approximate the MISC using the current user command $u_0^h = \pi_h(x_0, 0)$ and past observation. MISC is often defined using a 1-step loss [23] (i.e., minimizing only $\ell_{\text{int}}(x_0, u_0, u_0^h)$), which works fairly well if the human provides direct control, but works poorly for target tracking, particularly in underactuated systems. For tracking, past approaches include discounting future target deviation cost to the currently commanded target [23], and using intention prediction to obtain future human command trajectories [24]. In our experiments in Sec. VI-A.2, we introduce two MPC-based controllers that vary in their approach to approximating MISC: MPC Safety Filter (MPC-SF) [13] and MPC Target Tracking (MPC-TT). MPC-SF obtains the system control from an unsafe controller, and adopts a 1-step loss. MPC-TT is an undiscounted formulation of (2).

IV. Reachability analysis

Here we lay the groundwork, based on reachable sets, needed to define the CPI metrics in Section V. We define reachable and viable sets, distinguishing between controller-dependent and controller-independent quantities. Let $\mathbb{U} = \mathbb{R}_+ \to \mathcal{U}$ and $\mathbb{U}_h = \mathbb{R}_+ \to \mathcal{U}_h$.

A. Controller-independent sets

We define the controller-independent forward reachable set as the set of states that can be reached from $\mathcal{X}_0$ via a feasible state trajectory obeying some feasible control:

$$\begin{aligned} \mathcal{R}(\mathcal{X}_0) = \{\bar{x} \mid \exists T > 0,\ \exists u \in \mathbb{U},\ x(0) \in \mathcal{X}_0, \\ \dot{x} = f(x, u),\ x(T) = \bar{x}\} \end{aligned} \tag{3}$$

Similarly, we define the controller-independent viable set as the set of states that can reach some $\mathcal{X}_{\text{safe}}$ via a feasible state trajectory obeying some feasible control:

$$\begin{aligned} \mathcal{V}(\mathcal{X}_{\text{safe}}) = \{\bar{x} \mid \exists T > 0,\ \exists u \in \mathbb{U},\ x(0) = \bar{x}, \\ \dot{x} = f(x, u),\ x(T) \in \mathcal{X}_{\text{safe}}\} \end{aligned} \tag{4}$$

Note that $\mathcal{R}(\mathcal{X}_0)$ and $\mathcal{V}(\mathcal{X}_0)$ depend only on the system controls and dynamics, not the shared controller.

B. Controller-dependent sets

The controller-dependent forward reachable set is defined as the set of states reachable under the shared controller π_s under any user control inputs $u^h \in \mathcal{U}_h$:

$$\begin{aligned} \mathcal{R}(\pi_s, \mathcal{X}_0) = \{\bar{x} \mid \exists T > 0,\ \exists u^h \in \mathbb{U}_h,\ x(0) \in \mathcal{X}_0, \\ \dot{x} = f(x, \pi_s(x, u^h)),\ x(T) = \bar{x}\} \end{aligned} \tag{5}$$

Similarly, we define the controller-dependent viable set:

$$\begin{aligned} \mathcal{V}(\pi_s, \mathcal{X}_{\text{safe}}) = \{\bar{x} \mid \exists T > 0,\ \exists u^h \in \mathbb{U}_h, \\ \dot{x} = f(x, \pi_s(x, u^h)),\ x(T) \in \mathcal{X}_{\text{safe}}\} \end{aligned} \tag{6}$$

which is the set of states that can reach $\mathcal{X}_{\text{safe}}$ via a feasible state trajectory obeying a shared controller policy π_s as a response to user input from $\mathbb{U}_h$.

These sets are illustrated in Fig. 2. Note that $\mathcal{R}(\pi_s, \mathcal{X}_0) \subseteq \mathcal{R}(\mathcal{X}_0)$ and $\mathcal{V}(\pi_s, \mathcal{X}_{\text{safe}}) \subseteq \mathcal{V}(\mathcal{X}_{\text{safe}})$. We confine $x(t) \in \mathcal{X}$ at all times.

C. Connections with other sets and functions

We point out here connections between our set definitions and some other key sets defined in control theory related to stability guarantees and safety constraints. The difference often depends on whether it's forward or backward propagation of the dynamics, what is the target set considered, if a controller is applied, whether the disturbance is considered as well as time horizon used in set computing, whether it's discrete or continuous representation, etc. Given a target set $\mathcal{X}_N$, the maximum controllable set [10] is the same as our

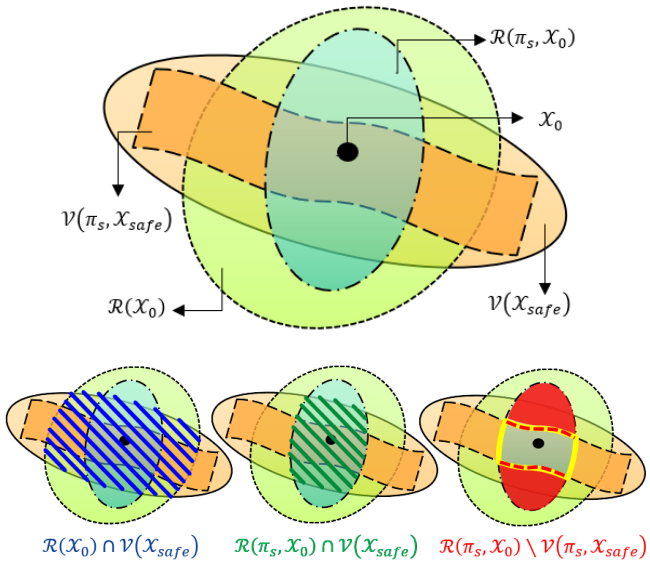


Fig. 2. Illustration of a controller-independent reachable set $\mathcal{R}(\mathcal{X}_0)$ (green area with the densely dashed boundary), a viable set $\mathcal{V}(\mathcal{X}_{\text{safe}})$ (light orange with the solid boundary), a controller-dependent reachable set $\mathcal{R}(\pi_s, \mathcal{X}_0)$ (blue with the dashed and dotted boundary) and a viable set $\mathcal{V}(\pi_s, \mathcal{X}_{\text{safe}})$ (orange with the sparsely dashed boundary) used in M_C and M_I. [Best viewed in color.]

controller-independent viable set $\mathcal{V}(\mathcal{X}_N)$, and both of them are valid CIS. Our controller-dependent viable set $\mathcal{V}(\pi_s, \mathcal{X}_N)$ is a valid positive invariant set [10]. The region of attraction [25] is a subset of $\mathcal{V}(\pi_s, \mathcal{X}_N)$. Finally, the control Lyapunov functions (CLFs) [26] is a continuous-time representation of a subset of $\mathcal{V}(\mathcal{X}_N)$ and the CBFs [26] can be seen as a continuous-time version of a subset of our reachable set $\mathcal{R}(\mathcal{X}_0)$. If systems are subject to disturbance $w(k) \in \mathcal{W}$ then *robust* will be used to indicate the sets, usually added in front of the set definitions

V. CPI METRICS

The CPI metrics map a safety controller π_s and a user behavior π_h to a three-tuple (M_C, M_P, M_I), measuring conservativeness in terms of how much we are limiting the capabilities of the system artificially through adding intervening controls, permissiveness in terms of the portion of the forward reachable set that is outside of the viable set, and the intervention in terms of the actual amount of intervention applied on a user control, respectively. We note that M_C and M_P have the favorable properties of being dimensionless and taking the range $[0, 1]$, with 0 being better. M_I is in $\mathbb{R}_+$, with 0 corresponding to no intervention from the shared controller.

A. Conservativeness Metric

M_C is defined as one minus the fraction of the intersection of the controller-independent reachable and viable sets that is reachable under the safety controller, i.e.

$$M_C = 1 - \frac{\mathbf{vol}(\mathcal{R}(\pi_s, \mathcal{X}_0) \cap \mathcal{V}(\mathcal{X}_{\text{safe}}))}{\mathbf{vol}(\mathcal{R}(\mathcal{X}_0) \cap \mathcal{V}(\mathcal{X}_{\text{safe}}))}, \tag{7}$$

where **vol** stands for a volume measure.

M_C captures whether a controller is conservative, with more conservative controllers having higher values of M_C, and less conservative controllers having M_C closer to 0. As an extreme, the pass-through controller $\pi_s(x, \pi_h) = \pi_h$ that simply replicates the user control will exhibit $M_C = 0$.

B. Permissiveness Metric

M_P is defined as the fraction of the boundary of the controller-dependent reachable set that is not viable.

$$M_P = \frac{\mathbf{vol}\left(\partial\left\{\mathcal{R}(\pi_s, \mathcal{X}_0) \setminus \mathcal{V}(\pi_s, \mathcal{X}_{\text{safe}})\right\} \cap \partial\mathcal{V}(\pi_s, \mathcal{X}_{\text{safe}})\right)}{\mathbf{vol}(\partial\left\{\mathcal{R}(\pi_s, \mathcal{X}_0) \cap \mathcal{V}(\pi_s, \mathcal{X}_{\text{safe}})\right\})} \tag{8}$$

where ∂ denotes the boundary, and the volume measure here operates on sets of dimension $n - 1$. The numerator is shown as the dashed red boundary in the bottom right of Fig. 2, and the denominator is the bold yellow boundary. This ratio estimates of the likelihood that the user would reach an unsafe boundary of the viable set, with more permissive controllers having higher values of M_P and safer controllers having M_P closer to 0. As an extreme, a controller that enforces staying at a safe state at all times (e.g. for the cartpole system, an LQR which does not take any user input and instead simply tracks the upright position) will exhibit $M_P = 0$. Less safe controllers will have $0 < M_P < 1$ with larger values indicating more "dangerous" controllers.

C. Intervention Metric

M_I is the expected value of the amount of intervention applied to the user nominal control over a distribution of user behaviors π_h. Define a trajectory τ as a sequence of states and controls $\tau = (x_0, u_0, x_1, u_1, ...)$. We collect a set of trajectories $\mathcal{D} = \tau_{i=1,...,K}$, where each trajectory is obtained by letting the user control the system while being assisted by the shared controller. We then define the M_I metric as the sample estimate:

$$M_I = \frac{1}{|\mathcal{D}| \cdot N} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T-1} \hat{\ell}_{\text{int}}(x, u, \pi_h), \tag{9}$$

where $N > 0$ is the length of each trajectory for which we estimate M_I, and $\hat{\ell}_{\text{int}}(x, u, \pi_h)$ is an intervention objective as defined in Sec. III-.3 but with a fixed cost matrix to make it possible to compare different controllers on a comparable score. As stated, this metric assumes that all user commands are meaningful and should be followed if possible. If there exists a way to measure how "meaningful" a command is at a given time step, it should be used to weight the intervention score. M_I reaches its minimum at 0 by a controller that always replicates the user's desired control.

D. Reachable and viable set computation

To calculate volumes, state space are discretized onto grid cells with resolution r, which dictates the accuracy of the sets: finer resolution yields a closer and smoother approximation of the true sets, at the expense of longer computation time. We are concerned with the *volumetric ratio*, which is the number of occupied grid cells over the total number of cells in the grid.

Algorithm 1 Reachability Estimation with RRT

Input
  N maximum number of samples
  n number of trajectories sampled per extension
$\{x_{\text{init}}\} \leftarrow$ SampleStates($\mathcal{X}_{\text{init}}$)
$T \leftarrow \{x_{\text{init}}\}$
$\mathcal{R} \leftarrow \emptyset$
for $i = 1, ..., N$ **do**
  $x_{\text{rand}} \leftarrow$ SampleState($\mathcal{X}$)
  $x_{\text{nearest}} \leftarrow$ FindNearestTreeNode(T, x_{rand})
  $\Xi \leftarrow$ SampleTrajectories(x_{nearest}, n)
  $\mathcal{R} \leftarrow \mathcal{R} \cup \Xi$
  $x_{\text{new}} \leftarrow$ FindNearestState(Ξ, x_{rand})
  Add x_{new} to T as a child of x_{nearest}
return $\mathcal{R}$

We adapted the RRT algorithm to compute reachable sets as outlined in Alg. 1. First, we sample initial states from $\mathcal{X}_{\text{init}}$, and then start building a tree in RRT fashion by sampling a random state $x_{\text{rand}} \in \mathcal{X}$ and finding the nearest node x_{nearest} in the tree. A weighted Euclidean distance metric is used, and the nodes in tree are stored in a k-d tree data structure to accelerate nearest neighbors computation. From x_{nearest}, we generate n trajectories with different controls. In the case of shared controllers, controls are sampled from the space of user inputs. The generated trajectories are added into the reachable set, and the closest terminal state to x_{rand} is denoted x_{new}. x_{new} is added into the tree as a child of x_{nearest}. We then keep sampling until convergence of the reachable set, which is defined as the rate of change of the volumetric ratio falling below a threshold for certain number of iterations.

In order to speed up the computation and convergence rate, instead of building one single tree of N nodes, we build k trees of N/k nodes in parallel and take the aggregation of the reachable sets resulting from each tree as the final result. The aggregation is done by taking the union of the occupied grids in each reachable set. In this way, the algorithm takes $\mathcal{O}(\frac{N}{k}\log(N/K))$ time – in other words, more than k times faster than with one single tree, and in practice we have found that it achieves a same or better coverage, likely due to the history-dependent nature of RRT construction. We can also choose a larger grid resolution and interpolate in trajectories to speed up the convergence of the reachable set.

The method to compute viable sets is similar to the way forward reachable sets are computed. The only difference is that we need to reverse the direction of time in the dynamical system, i.e. we apply backward instead of forward differencing.

Note that our method does not require any derivatives of the system, unlike HJ reachability which requires a differentiable expression of the shared controller. This makes our approach suitable to non-differentiable shared controllers, e.g. those that result from solving an optimization problem. Moreover, we only record the index of the reachable or viable grid during computation instead of storing the whole grid space, making it amenable to some extent to high-dimensional systems.

VI. Experimental results

To evaluate our work, we compute the CPI metrics for LQR and MPC-based safe controllers to cartpole via numerical simulation, and illustrate how they correspond to qualitative system behavior. Note that the metrics are computed offline given access to a shared controller and system simulator as stated in Section III and a sample of user trajectories for the M_I metric as stated in Section V-C.

A. Setup

1) Dynamical systems: We consider a cartpole system with configuration $\mathbf{q} = [d \quad \theta]^\top$, where d is the horizontal position of the cart, θ is the angle of the pole (defined to be 0 at the upright position). The standard cartpole dynamics operate on the state variable $\mathbf{x} = [\mathbf{q} \quad \dot{\mathbf{q}}]^\top$ and the control variable which is the force applied on the cartpole. The state and control spaces are chosen to be $\mathcal{X} = [-\infty, \infty] \times [-\frac{\pi}{2}, \frac{\pi}{2}] \times [-6, 6] \times [-2\pi, 2\pi]$ and control $\mathcal{U} = [-50, 50]$. The initial set $\mathcal{X}_0 = \{0\}^4$ and the safe set $\mathcal{X}_{\text{safe}} = \left\{[-\infty, \infty] \times \{0\}^3\right\} \cap \mathcal{X}$. The partition resolution of the state space is $r = 0.1$. The user input considered is a state command x^h generated randomly from $[-10, 10] \times \{0\} \times \{0\} \times \{0\}$, and is assumed to change at every time step, set to $dt = 0.02$s.

2) Shared controllers: The first controller we consider is an infinite-horizon LQR of weight matrices Q_{lqr} and R, with the dynamics linearized at equilibrium point. Q_{lqr} is varied in the experiments as shown in Table I while R is fixed to 1. User input is saturated if the generated policy is beyond the range of admissible controls.

We also consider the two safe shared controllers MPC-SF and MPC-TT. The MPC-SF [13] is formulated as a one-step penalization of the control deviation:

$$\begin{aligned} \underset{\mathbf{u}[\cdot]}{\text{minimize}} \quad & ||u_0 - u^h||_R^2 \\ \text{subject to} \quad & x_0 = x(0),\ x_N \in \mathcal{X}_N \\ & \forall k \in \{0, \ldots, N-1\}, \\ & x_{k+1} = f(x_k, u_k), x_k \in \mathcal{X}, u_k \in \mathcal{U}. \end{aligned} \tag{10}$$

where N is the control horizon and $\mathcal{X}_N$ the terminal constraint set. We formulate MPC-TT as the aggregate penalization of the target deviation over a given horizon N:

$$\begin{aligned} \underset{\mathbf{x}[\cdot],\mathbf{u}[\cdot]}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \|x_k - x^h\|_Q^2 \\ \text{subject to} \quad & \text{same constraints as in Eq. 10} \end{aligned} \tag{11}$$

A terminal cost term $V(x_N) = \|x_N - x^h\|_{Q_N}^2$ can be added in Eq. 11, but not required. This formulation permits us to replace x^h with a predicted trajectory coming from our testing dataset.

The cost matrices used are $R = [1]$ and $Q = I_4$. The terminal constraint set $\mathcal{X}_N$ is set to $\mathcal{X}_{\text{safe}}$. The optimization time step is $h = 0.1$ and the problem is solved via direct

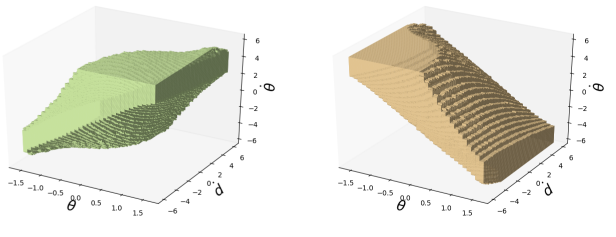


Fig. 3. Controller-independent reachable set $\mathcal{R}(\mathcal{X}_0)$ (left) and viable set $\mathcal{V}(\mathcal{X}_{\text{safe}})$ (right) of a cartpole system using RRT-based reachability analysis. The volumetric ratio is 46.92% and 47.23% for each.

collocation using the open-source solver CasADi [27]. We treat the MPC horizon as a parameter, N =5, 10, or 20.

B. Reachability analysis

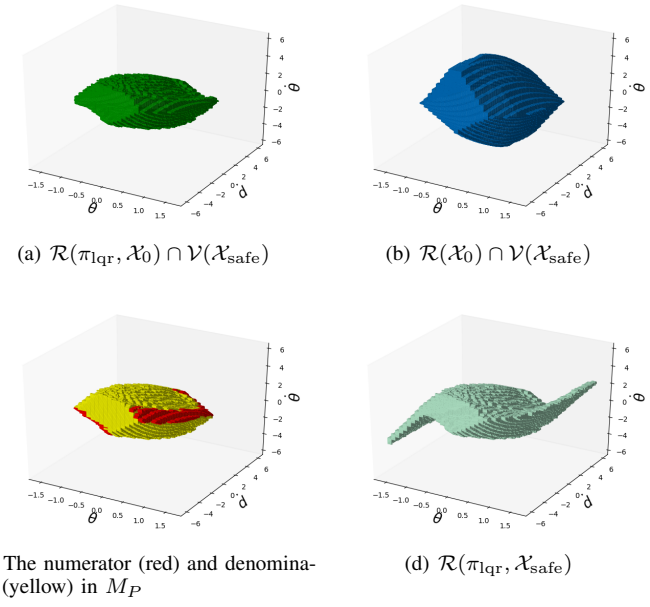
We remove the position dimension of the state from our state space grid representation since it is a dynamically invariant dimension of the system. The controller-independent reachable and viable sets, $\mathcal{R}(\mathcal{X}_0)$ and $\mathcal{V}(\mathcal{X}_{\text{safe}})$, as computed per Alg. 1 are shown in Fig. 3. The volumetric ratios are 46.92% and 47.23%, respectively. We use the following parameters: tree count $k = 10$, number of sampled nodes $N = 100000$, number of trajectories sampled for each node $n = 15$, trajectory length 10, and weight matrix for the distance metric $w = \text{diag}(0, 4, 1, 1)$ (θ is more safety-critical than the other dimensions). The total time for computing the reachable set is around 30 min. The parameters listed here could be optimized with further investigation.

In order to evaluate whether the sets are reasonable in terms of space coverage, we use HJ reachability (via the level set toolbox [28]) as a baseline and compare the volumetric ratios of the two methods. The volumetric ratios for $\mathcal{R}(\mathcal{X}_0)$ and $\mathcal{V}(\mathcal{X}_{\text{safe}})$ are 47.47% and 47.36%, respectively, with a computing time of around 2 hours. We can thus bring the coverage of the sets from our sampling-based method within 2% of the baseline HJ reachability method while requiring approximately one fourth of the computing time.

C. CPI metrics evaluation

Fig. 4 shows some sets computed to evaluate M_C and M_P for cartpole model with an LQR controller ($Q_{\text{lqr}} = 10 \cdot I_4$). Metric values for other weights are given in Table I. We observe larger weights result in smaller values of M_C, because the controller commands more aggressive movements, thereby resulting in a larger coverage of reachable states that are viable. This also lets the user drive the system into more non-viable states, thereby resulting in larger values of M_P.

CPI metrics for cartpole system with MPC-based safe shared controller are shown in Table II, We observe that in both formulations M_C decreases with the horizon, which is also aligned with the results in Fig. 5. This is due to larger horizons allowing the system to reach more states, which can further recover the maximum reachable set when $N \to \infty$. In the extreme case, an infinite horizon MPC with an unbounded $\mathcal{U}$ would make conservativeness tend to zero, i.e. $M_C = 0$. M_P, on the other hand, is always 0 because



(a) $\mathcal{R}(\pi_{\text{lqr}}, \mathcal{X}_0) \cap \mathcal{V}(\mathcal{X}_{\text{safe}})$

(b) $\mathcal{R}(\mathcal{X}_0) \cap \mathcal{V}(\mathcal{X}_{\text{safe}})$

(c) The numerator (red) and denominator (yellow) in M_P

(d) $\mathcal{R}(\pi_{\text{lqr}}, \mathcal{X}_{\text{safe}})$

Fig. 4. Sets used to calculate M_C and M_P for cartpole with LQR.

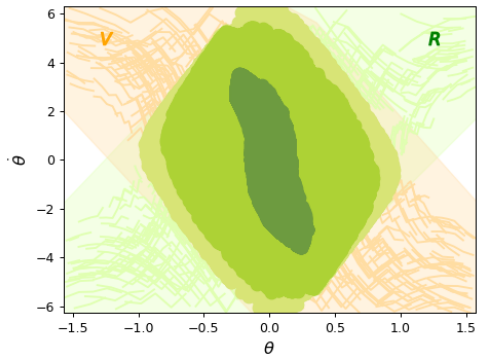


Fig. 5. $\mathcal{R}(\pi_s, \mathcal{X}_0)$ for MPC-TT with horizons N=5 (dark green), N=10 (green) and N=20 (light green). Set labeled with "R" is $\mathcal{R}(\mathcal{X}_0)$ and labeled with "V" is $\mathcal{V}(\mathcal{X}_{\text{safe}})$. Curves in the two sets are trajectory samples generated when computing the sets. All the sets are in $[\theta \quad \dot{\theta}]$ dimensions. [Best viewed in color.]

the terminal constraint guarantees that reachable states are always viable. Indeed, if MPC finds a solution that fulfills the terminal constraint, since $\mathcal{X}_N \subset \mathcal{V}(\pi_s, \mathcal{X}_{\text{safe}})$, then all the intermediate states will be in the viable set.

Finally, for the M_I metric, both LQR and MPC controllers are evaluated over a user input dataset $\mathcal{D}$ that is mixed with user input trajectories generated through sinusoidal, step, and linear functions. The dataset is designed such that we have user input with different challenges to experiment on. Sinusoidal functions take the form $A\sin(2\pi f t)$ where A and f are drawn uniformly at random from $[1, 10]$ and $(0, \frac{1}{2\pi}]$, respectively. Step functions take the form $\sum_{i=1}^{m} A_i \cdot \mathbb{1}\{t_{i-1} \leq t \leq t_i\}$ with the number of steps m drawn uniformly at random from $\{1, 2, 3, 4\}$ and A_i and t_i from $[-10, 10]$ (and $t_0 = 0$) and $[0, 10]$, respectively. Finally, linear functions take the form $k \cdot t$ with k drawn uniformly at random from

TABLE I

CPI METRICS FOR LQR, VARYING WEIGHT MATRIX

Q_{lqr}	I_4	$10 \cdot I_4$	$100 \cdot I_4$
M_C	0.586	0.381	0.302
M_P	0.061	0.153	0.190
M_I	0.36	0.32	0.31

TABLE II

CPI METRICS FOR MPC-SF AND MPC-TT VARIANTS.

	MPC-SF			MPC-TT		
Horizon	5	10	20	5	10	20
M_C	0.988	0.539	0.446	0.994	0.392	0.071
M_P	0.0	0.0	0.0	0.0	0.0	0.0
M_I	0.38	0.32	0.32	0.40	0.31	0.32
M_I(Pred.†)				0.40	0.24	0.24

†: Clairvoyant prediction of future user command

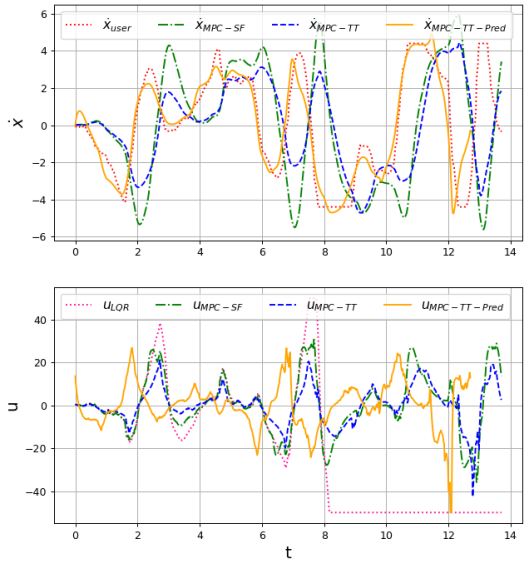


Fig. 6. Forward velocity $\dot{x}_\bullet$ and control output $u_\bullet$ from different controllers given real human velocity input $\dot{x}_{\text{user}}$. The command is aggressive and LQR violates feasibility at approximately 7 s. Each MPC controller is able to filter out unsafe controls and the system remains feasible. MPC-TT-Pred tracks the user command the best out of the four controllers because it is able to anticipate future changes of the command.

$[-1, 1]$. Each function category contains 25 samples, with the duration set to 10 s. Trajectories that leave the feasible space are truncated in the calculation of M_I.

From Table I we observe that larger Q_{lqr} values lead to smaller M_I. This is due to larger cost matrices helping the tracking converge faster, resulting in a smaller intervention. Results in Table II show that the intervention decreases at $N = 10$ for both MPC-SF and MPC-TT in general because the controller a larger horizon gives more time to plan and track. However, increasing the horizon further does not help because the constant user input assumption becomes less valid as the prediction horizon increases. This can be alleviated by including a human intention prediction module. The result of MPC-TT with human intention prediction (MPC-TT-Pred) is also shown in Table II, where we feed in the clairvoyant human trajectory as the trajectory to be tracked. The results show that accurate intent prediction has a strong impact on reducing the intervention metric.

Although the prior tests used synthetic user inputs to estimate M_I, we show that the metric is indeed predictive of real-world performance. We collected data from a human operator through a human-machine interface for full-body control of a wheeled robot and used it as a velocity command [29]. We scaled the user input signal by 40 in order to trigger LQR failure cases and then fed that LQR control output to MPC-SF (horizon $N = 10$). We also send the scaled user input to be tracked by MPC-TT and the clairvoyant human velocity trajectory to MPC-TT-Pred. The output is shown in Fig. 6. Each MPC variant produces a safe control, but MPC-TT-Pred fits the user velocity command the best. This result is consistent with the M_I estimations of Table II in which MPC-TT-Pred outperforms the other techniques.

D. Discussion

1) Testing on high-dimensional system: We applied our reachability analysis approach on a 7 degrees-of-freedom double-wheeled inverted pendulum (DWIP) system [30] to extract the sets required for metric evaluation for LQR controllers. The state variables $\mathbf{x} = [x \;\; y \;\; \theta \;\; \psi \;\; \dot{d} \;\; \dot{\theta} \;\; \dot{\psi}]$ include xy-plane coordinates, tilt angle θ, steering angle ψ and their velocities, as well as the forward velocity $\dot{d}$. The controls are wheel torques. The state and control spaces are chosen to be $\mathcal{X} = [-4, 4] \times [-4, 4] \times [-\frac{\pi}{2}, \frac{\pi}{2}] \times [\pi, \pi] \times [-3, 3] \times [-3, 3] \times [-3, 3]$ and $\mathcal{U} = [-5, 5] \times [-5, 5]$. Initial set is $\mathcal{X}_0 = \{0\}^7$ and safe set is $\mathcal{X}_{safe} = \{[-\infty, \infty] \times [-\infty, \infty] \times 0 \times [-\infty, \infty] \times \{0\}^3\} \bigcap \mathcal{X}$. We use a coarser partition resolution $r = [1.0, 1.0, 0.2, 1.0, 0.5, 0.5, 0.5]$ so as to reduce computation time. The user input are forward movement $\delta d \in [-5, 5]$ and steering change $\delta\psi \in [-2, 2]$. The cost matrices used in LQR are $R = I_2$ and Q is set to I_6, $10 \cdot I_6$, and $100 \cdot I_6$. Note we reduce state from 7D to 6D by substituting x, y with d when computing LQR gain to remove the correlated dimension. We used 18 trees each with 50k nodes for reachability analysis and the volumetric ratios for $\mathcal{R}(\mathcal{X}_0)$ and $\mathcal{V}(\mathcal{X}_{\text{safe}})$ are 18.25% and 18.21%, respectively. The resulting M_C are 0.373, 0.098, 0.024 for each Q and M_P are 0.515, 0.537, 0.539. Therefore, we obtain the same conclusion for both metrics: larger weights result in less conservativeness and increasing permissiveness.

The results are sensitive to the number of iterations and nodes sampled. Since our approach is an approximation of the true reachable set, when HJ reachability computation is tractable, we can use it as a reference to guide parameter

tuning. However, we lose this sanity checking benefit for higher dimensional systems as HJ reachability scales poorly. Although we could alleviate the computational burden of high dimensional reachability analysis via coarser resolutions and looser convergence criteria, a more principled approach would be to devise a more efficient representation of reachable sets that have lower memory consumption and approximate better in higher dimensions. This is a future direction to investigate.

2) User behavior model: M_I depends on the user behavior distribution model. In particular, humans can learn to predict the behavior of the shared controller, and early partial intervention before violating a safety constraint can teach the user about the limits of the system. At the same time, π_s can have better tracking performance in terms of intervention when it can predict the human policy π_h closely. This can be our future work to solve.

VII. Conclusion

We introduced CPI metrics to evaluate shared controllers in a principled manner by quantifying their conservativeness, permissiveness and the amount of intervention, given some user behavior. We proposed an RRT-based framework for efficiently computing the reachable sets required for said metrics. Case study on the cartpole comparing LQR with different MPC-based safe shared controllers has shown that these metrics are useful to evaluate a shared controller and to choose the most appropriate one for given requirements. We envision several promising directions for future work: considering different user behavior models when generating the CPI metrics, varying user input and the shared control task with more complex environments, and empirically confirming the metrics on a real robot.

References

[1] Z. Li, P. Moran, Q. Dong, R. J. Shaw, and K. Hauser, "Development of a tele-nursing mobile manipulator for remote care-giving in quarantine areas," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3581–3586.

[2] A. Erdogan and B. D. Argall, "The effect of robotic wheelchair control paradigm and interface on user performance, effort and preference: an experimental assessment," *Robotics and Autonomous Systems*, vol. 94, pp. 282–297, 2017.

[3] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Ph.D. dissertation, Technische Universität München, 2010.

[4] T. Carlson and Y. Demiris, "Collaborative control for a robotic wheelchair: evaluation of performance, attention, and workload," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 876–888, 2012.

[5] K. P. Tee and Y. Wu, "Experimental evaluation of divisible human-robot shared control for teleoperation assistance," in *TENCON 2018-2018 IEEE Region 10 Conference*. IEEE, 2018, pp. 0182–0187.

[6] Y. Oh, S.-W. Wu, M. Toussaint, and J. Mainprice, "Natural gradient shared control," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 1223–1229.

[7] A. Broad, T. Murphey, and B. Argall, "Highly parallelized data-driven mpc for minimal intervention shared control," *arXiv preprint arXiv:1906.02318*, 2019.

[8] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *arXiv preprint arXiv:2108.06266*, 2021.

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

[10] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[11] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The simplex architecture for safe online control system upgrades," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207)*, vol. 6. IEEE, 1998, pp. 3504–3508.

[12] 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.

[13] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger, "A predictive safety filter for learning-based racing control," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7635–7642, 2021.

[14] I. M. Mitchell, "Comparing forward and backward reachability as tools for safety analysis," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2007, pp. 428–443.

[15] A. D. Wang, "A methodology to quantify risk of failure for dynamic robots," Ph.D. dissertation, Massachusetts Institute of Technology, 2019.

[16] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2242–2253.

[17] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, "Decomposition of reachable sets and tubes for a class of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3675–3688, 2018.

[18] M. Althoff, D. Grebenyuk, and N. Kochdumper, "Implementation of taylor models in cora 2018," in *Proc. of the 5th International Workshop on Applied Verification for Continuous and Hybrid Systems*, 2018.

[19] M. Althoff, G. Frehse, and A. Girard, "Set propagation techniques for reachability analysis," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 369–395, 2021.

[20] T. Lew and M. Pavone, "Sampling-based reachability analysis: A random set theory approach with adversarial sampling," *arXiv preprint arXiv:2008.10180*, 2020.

[21] J. M. Esposito, "Randomized test case generation for hybrid systems: metric selection," in *Thirty-Sixth Southeastern Symposium on System Theory, 2004. Proceedings of the*. IEEE, 2004, pp. 236–240.

[22] A. Wu, S. Sadraddini, and R. Tedrake, "R3t: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4245–4251.

[23] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2994–3008, 2017.

[24] K. Hauser, "Recognition, prediction, and planning for assisted teleoperation of freeform tasks," *Autonomous Robots*, vol. 35, no. 4, pp. 241–254, 2013.

[25] A. El-Guindy, D. Han, and M. Althoff, "Estimating the region of attraction via forward reachable sets," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 1263–1270.

[26] 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.

[27] 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, 2019.

[28] I. M. Mitchell, "A toolbox of level set methods," *UBC Department of Computer Science Technical Report TR-2007-11*, 2007.

[29] S. Wang and J. Ramos, "Dynamic locomotion teleoperation of a wheeled humanoid robot reduced model with a whole-body human-machine interface," *arXiv preprint arXiv:2109.03906*, 2021.

[30] S. Kim and S. Kwon, "Dynamic modeling of a two-wheeled inverted pendulum balancing mobile robot," *International Journal of Control, Automation and Systems*, vol. 13, no. 4, pp. 926–933, 2015.