

# Whole-Body Safe Control of Robotic Systems with Koopman Neural Dynamics

Sebin Jung<sup>1</sup>, Abulikemu Abuduweili<sup>1</sup>, Jiaying Li<sup>1</sup>, Changliu Liu<sup>1</sup>

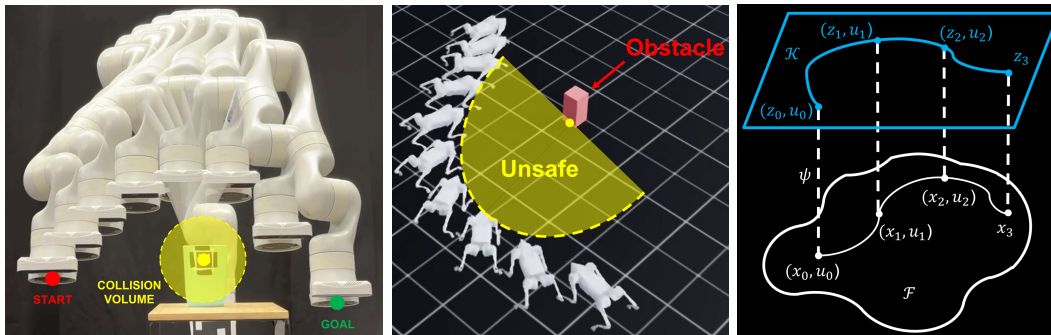


Fig. 1: Left: Kinova Gen3 safely moving from START (red) to GOAL (green) while avoiding the collision volume (yellow); collision avoidance is enforced using the learned Koopman dynamics within our unified Koopman–SSA MPC. Middle: Floating-base safe control in Isaac simulation (Unitree Go2). The learned Koopman-MPC tracks the reference while actively avoiding the obstacle-induced unsafe region; collision avoidance is enforced using the learned Koopman dynamics within the MPC rollout. Right: We use the learned Koopman lifted space and its linear dynamics (blue) to predict motion in MPC and evaluate the safety constraints, which enables the collision avoidance shown in the left and middle panels.

**Abstract**—Controlling robots with strongly nonlinear, high-dimensional dynamics remains challenging, as direct nonlinear optimization with safety constraints is often intractable in real time. The Koopman operator offers a way to represent nonlinear systems linearly in a lifted space, enabling the use of efficient linear control. We propose a data-driven framework that learns a Koopman embedding and operator from data, and integrates the resulting linear model with the Safe Set Algorithm (SSA). This allows the tracking and safety constraints to be solved in a single quadratic program (QP), ensuring feasibility and optimality without a separate safety filter. We validate the method on a Kinova Gen3 manipulator and a Go2 quadruped, showing accurate tracking and obstacle avoidance.

## I. INTRODUCTION

Ensuring safe control of robotic systems is fundamentally difficult due to the interplay of complex dynamics, high dimensionality, and safety-critical constraints. A first challenge arises from the strong nonlinearity of the dynamics: joint couplings, contact interactions, and nonlinear actuation make the direct embedding of such models into optimization-based controllers computationally prohibitive. Even when accurate models are available, nonlinear formulations with safety constraints often lead to nonconvex programs that cannot be solved in real time [1]. A second challenge is on feasibility, which emerges at the boundary of the safe set, where a controller may fail to generate feasible inputs to steer the system back into safety. Finally, when using learned dynamics, approximation errors can propagate into safety constraints, rendering the assumed safe control unsafe.

A range of data-driven methods learn predictive models of nonlinear dynamics, but they rarely translate into real-time

safe control. Dynamic Mode Decomposition and its extensions approximate nonlinear systems with linear operators in lifted spaces [2], [3], [4], while sparse-regression methods such as SINDy identify compact governing equations [5]. Deep recurrent and latent-state models extend predictive power [6], and Koopman operator theory provides a principled framework for globally linearizing nonlinear dynamics [7], [8], [9]. Yet these approaches focus on prediction, and do not ensure that modeling errors or online computation limits can meet hard safety constraints during control.

On the control side, no existing family of methods simultaneously handles high dimensionality, model uncertainty, and strict safety guarantees. Classical nonlinear controllers such as feedback linearization require exact models [10], and trajectory-optimization techniques like Iterative Linear Quadratic Regulator (iLQR) or Nonlinear Model Predictive Control (NMPC) scale poorly as degrees of freedom grow [11]. Reinforcement learning has shown striking successes in robotics [12], but it demands large data and provides only probabilistic safety at best [13]. Control Barrier Functions (CBFs) and Safe-Set Algorithms provide convex safety-filtering mechanisms [14], [15] yet remain difficult to formally synthesize with learned or highly nonlinear models in real time. Even when synthesis is learned from data, optimization-based coupling with hard safety constraints is challenging and often exhibits feasibility issues in practice. Moreover, under strongly nonlinear dynamics, safety is frequently enforced via a separate filtering layer on top of a nominal controller, which can induce performance–safety trade-offs such as overly conservative behavior or deadlock. Recent efforts to merge Koopman embeddings with CBFs [16] still separate nominal control from safety filtering, creating feasibility breakdowns near the boundary.

<sup>1</sup>Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. Contact: {sebinj, abulikea, jiayingl, cliu6}@andrew.cmu.edu

We propose a data-driven safe control framework that combines Koopman operator theory with whole-body robot dynamics. This framework (Fig. 3) tackles nonlinearity, safety-boundary infeasibility, and learned-model approximation errors, enabling scalable, real-time safe control of nonlinear robotic systems. The main contributions are:

**Synthesis of Safe Control via Koopman Linearization.**

We formulate whole-body safe control of robotic systems by globally linearizing their nonlinear dynamics through the Koopman operator. Unlike filtering- or projection-based two-stage architectures [17], which apply safety constraints as a post-processing step on nominal controls, our approach synthesizes safety directly into the nominal controller. This unified formulation ensures consistent reasoning about performance and safety while avoiding the conservatism and inefficiencies of safety filtering. As evidenced by Fig. 1, the proposed Koopman-based formulation enables safe control in both fixed-base and floating-base settings, successfully enforcing collision avoidance while maintaining task execution.

**Safety Index Synthesis for Learned Dynamics.** To mitigate infeasible control issues that arise when coupling learned Koopman dynamics with hard safety constraints, we introduce an adversarial fine tuning scheme for the safety index. This procedure adapts the safety specification to the learned dynamics, improving forward invariance inside the safe set by significantly reducing the risk of unsolvable constraints.

**Adaptation to Real-World Hardware.** We demonstrate the effectiveness of the proposed framework through sim-to-real deployment on a Kinova Gen3 manipulator with minimal retraining. As illustrated in Fig. 1, the Kinova safely tracks a given trajectory while avoiding the designated collision volume (yellow). A video demonstration of this experiment is provided in the supplementary material. We defer additional floating-base results on the Unitree Go2, including extended qualitative rollouts and supplementary analysis, to our extended version, available at [18].

## II. PRELIMINARIES

### A. Koopman Operator Theory

Given the nonlinear system with discrete dynamics:

$$x_{k+1} = f(x_k) \quad (1)$$

Koopman Operator  $\mathcal{K}$  is an infinite dimensional linear operator that maps the nonlinear dynamics to linear dynamics [19] and evolves the embedding functions  $\psi$  of the state:

$$\mathcal{K}\psi(x) = \psi \circ f(x) \quad (2)$$

Considering a nonlinear system with the control input  $u_k \in \mathcal{U} \subseteq \mathbb{R}^m$  and system state  $x_k \in \mathcal{X} \subset \mathbb{R}^n$ , given by  $x_{k+1} = f(x_k, u_k)$ , the evolution flow follows as:

$$\mathcal{K}\psi(x_k, u_k) = \psi(f(x_k, u_k)) = \psi(x_{k+1}) \quad (3)$$

where  $\psi : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^\infty$ . In practice, we constrain the latent space to a finite-dimensional vector space and approximate the Koopman operator, making  $\psi : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^d$  where  $d \in [0, \infty) \subset \mathbb{R}^+$ .

The embedding function  $\psi(x, u)$  can be separated as  $\psi(x, u) = [\psi_x(x, u); \psi_u(x, u)]$ , and we focus on the case where  $\psi_x(x, u) = \psi_x(x)$  and  $\psi_u(x, u) = \hat{u} = u$ . Here  $\psi_x : \mathbb{R}^n \rightarrow \mathbb{R}^{n'}$  denotes state embedding and  $\hat{u}$  is control in latent space. Under this formulation, the system dynamics can be expressed as:

$$\psi_x(x_{k+1}) = A\psi_x(x_k) + Bu_k \quad (4)$$

where  $A, B$  comes from:

$$\mathcal{K} = \begin{bmatrix} A \in \mathbb{R}^{n' \times n'} & B \in \mathbb{R}^{n' \times m} \\ C \in \mathbb{R}^{m \times n'} & D \in \mathbb{R}^{m \times m} \end{bmatrix} \quad (5)$$

We parameterize state embedding  $\psi_x$  as a neural network  $\psi_\omega$  and denote the lifted state  $z_k = \psi_x(x_k) = [x_k; \psi_\omega(x_k)]^\top$ . This preserves the original state data along with the neural network embedded state in order to prevent the information loss that may occur through nonlinear mapping [20]. Specifically, we can retrieve the original state via:

$$x_k = Pz_k \quad (6)$$

where  $P = [I_n; 0] \in \mathbb{R}^{n \times (n+n')}$ . Then, the lifted state evolution takes the form:

$$z_{k+1} = Az_k + Bu_k \quad (7)$$

This formulation allows us to design linear controllers for state-constrained control problems (e.g., collision avoidance) by leveraging the linearity of lifted dynamics. In addition, we use end-to-end training framework for the acquisition of embedding function as well as the Koopman operator  $(\psi_\omega, A, B)$ . Readers can refer to [19] for further understanding of Koopman Operator Theory.

### B. Safe Control Synthesis through Adversarial Fine Tuning

The goal of safe control is to design a controller that tracks a reference while ensuring forward invariance inside the allowable set  $\mathcal{A}$  [21]. For a function  $s : \mathbb{R}^n \rightarrow \mathbb{R}$  let  $\mathcal{S} := \{s\}_{\leq 0}$  be its zero-sublevel set,  $\partial\mathcal{S} := \{s\}_{=0}$  the boundary of this set. Assuming we are given: (1) a control-affine system  $\dot{x} = f(x) + g(x)u$  where  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$  and  $g : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$  are locally Lipschitz continuous on  $\mathbb{R}^n$ , (2) the control set  $\mathcal{U}$  is a bounded convex polytope, (3) safety specification  $\phi_0 : \mathbb{R}^n \rightarrow \mathbb{R}$  implicitly defines the allowable set  $\mathcal{A} := \{\phi_0\}_{\leq 0}$ .

If we directly impose the constraint  $\dot{\phi}_0(x) \leq 0$  at the boundary of allowable set  $\partial\mathcal{A}$  for a limit-agnostic index  $\phi_0$  and bounded  $\mathcal{U}$ , the feasible set can collapse, causing controller saturation or an unsolvable constraint. We therefore *reshape* the safety specification by introducing a parameterized index:

$$\phi_\rho = h(\rho, \phi_0) \quad (8)$$

where  $h : \mathbb{R} \rightarrow \mathbb{R}$  with learnable parameters represented as  $\rho$ . The zero-sublevel set of  $\phi_\rho$  defines a more conservative safe set  $\mathcal{S} := \{x | \phi_\rho(x) \leq 0\}$ . At its boundary, the safe control constraint is defined as:

$$\dot{\phi}_\rho(x, u) = \nabla\phi_\rho(x)^\top f(x) + \nabla\phi_\rho(x)^\top g(x)u \leq 0 \quad \forall x \in \partial\mathcal{S} \quad (9)$$

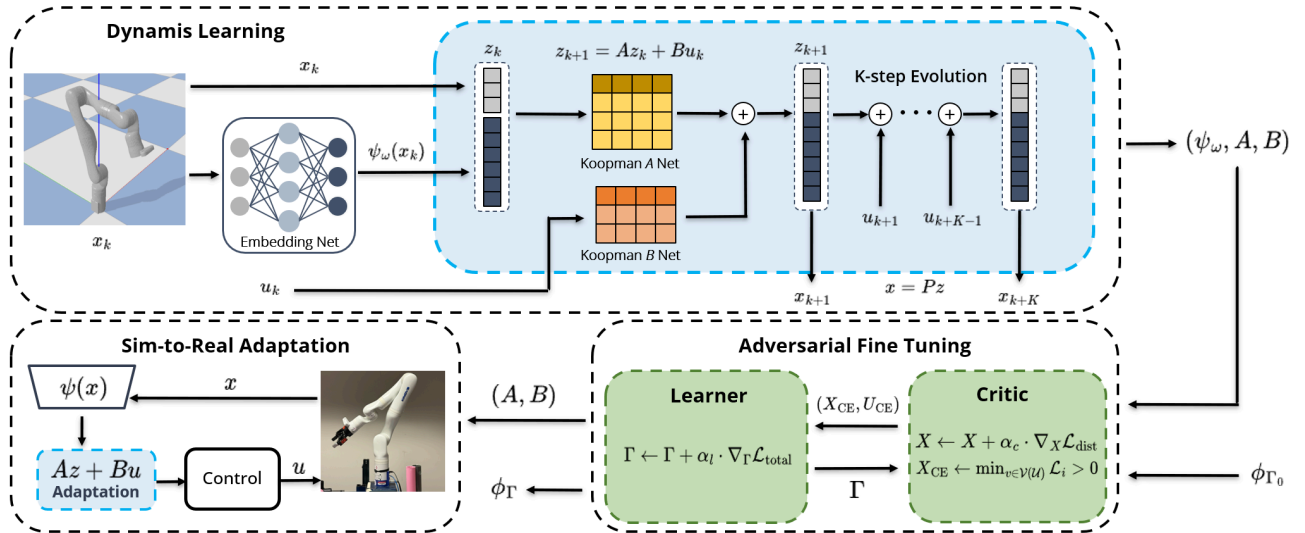


Fig. 3: Overview of training framework for Koopman Safe Control. Neural embedding function and Koopman Operators are first trained, safety index is adapted to the learned dynamics via adversarial fine tuning, and the model is migrated to real environment.

in order to guarantee forward invariance inside  $\mathcal{S}$ . Readers can refer to [22] for further details.

To ensure feasibility under input limits and model approximation, we adapt  $\rho$  via the min-max program:

$$\min_{\rho} \max_{x \in \partial \mathcal{S}} \inf_{u \in \mathcal{U}} \dot{\phi}_{\rho}(x, u) \quad (10)$$

which can be evaluated over the vertex set of control  $v \in \mathcal{V}(\mathcal{U})$  when  $\mathcal{U}$  is a polytope [22]. Thus, the objective can be computed as a discrete minimization:

$$\mathcal{L}(\rho, x) := \min_{v \in \mathcal{V}(\mathcal{U})} \dot{\phi}_{\rho}(x, v) \quad (11)$$

This procedure *learns* a safety index that avoids infeasible constraints while preserving forward invariance.

### III. PROBLEM FORMULATION

In this paper, we consider whole-body safe control of an  $n$ -DoF articulated rigid-body system (including fixed-base manipulators and floating-base legged or wheeled robots) operating amid static and dynamic obstacles. The target system is the dynamics in generalized coordinates:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau + J^{\top}(q)f \quad (12)$$

where  $q \in \mathbb{R}^n$  denotes a vector of generalized coordinates (e.g., joint positions for fixed-based robots; joint and base positions for floating-based robots), and  $\dot{q}$  the corresponding generalized velocities. Here  $M(q)$  is the inertia matrix,  $C(q, \dot{q})$  the Coriolis and centrifugal terms,  $G(q)$  the gravity vector,  $\tau$  the applied joint torques,  $J(q)$  the contact Jacobian, and  $f$  the contact forces. Equation (12) represents the low-level dynamics that must ultimately be respected, but which are highly nonlinear and high-dimensional to directly embed into real-time safety-critical controllers.

Our control objective is trajectory tracking in the presence of obstacles, under uncertainty in both robot motion and obstacle motion. We consider all obstacles and robot link

collision volumes to be spheres. This choice allows (i) distance functions and gradients admit closed-form expressions, enabling efficient evaluation of safety constraints, and (ii) spherical volumes approximate local link geometry while simplifying multi-link safety verification into a minimum-distance calculation between spheres.

We define a signed distance from the ego towards the obstacle as safety specification. Let the state be  $x := q$ . For a target link's center of mass  $p_{\text{ego}}(x) \in \mathbb{R}^3$  and obstacle center  $p_{\text{obs}} \in \mathbb{R}^3$ :

$$\phi_0(x) = d_{\min} - \|p_{\text{ego}}(x) - p_{\text{obs}}\|_2 \quad (13)$$

where  $d_{\min} \in \mathbb{R}^+$  is the minimum distance to be maintained between ego and obstacle. The nonlinear state-based safety specification  $\phi_0(x) \leq 0$  can be posed as state-dependent constraints on the control space:

$$\dot{\phi}_0(x_k, u_k) \leq b_{\phi}(x_k) \quad (14)$$

where the safety bound  $b_{\phi}(x)$  is defined as

$$b_{\phi}(x) = \begin{cases} 0 & \text{if } x \in \partial \mathcal{A}, \\ -\lambda & \text{if } x \notin \mathcal{A}, \\ \infty & \text{otherwise.} \end{cases} \quad (15)$$

This safe control constraint can be directly embedded into NMPC. However, solving with full nonlinear dynamics is computationally prohibitive and often infeasible in real time, while local linearizations quickly lose validity away from the expansion point. Synthesizing safety-filtering methods is nontrivial especially at the safe set boundary, and black-box learned models risk false safety guarantees due to approximation error. These drawbacks highlight the need for an approach that embeds safety directly while remaining tractable.

In summary, the problem we address is to synthesize safe, real-time feasible whole-body controllers for articulated rigid-body systems by (i) globally linearizing nonlinear

dynamics via Koopman operators, (ii) embedding safety constraints directly within a single MPC optimization, and (iii) adversarially adapting the safety index to ensure non-emptiness of safe control set as well as constraint solvability under learned dynamics.

#### IV. METHODOLOGY

In this section we present a single linear MPC formulation for whole-body safe control of articulated robots. The approach (1) learns a globally linear lifted model via Koopman operators, (2) embeds and redesigns the safe control constraint so it remains feasible at the safe state boundary, (3) adapts the model for hardware.

##### A. Safe MPC Formulation via Koopman Dynamics

Although naive models such as single-integrator dynamics [14] have been widely used for safety-critical control, they are ill-suited for articulated robots: they ignore actuation limits, joint coupling, and dynamic feasibility, leading to controllers that may certify safety in theory but fail in practice on high-DOF systems. To enable tractable safe control, we approximate (12) with a globally linear lifted model using Koopman operator. Following (2) and (3), we define an embedding  $\psi : \mathbb{R}^n \rightarrow \mathbb{R}^{n+d}$  and lifted state  $z_k = [x_k; \psi_\omega(x_k)]^\top$  with  $x_k = [p_k; q_k]^\top$ , where  $p_k \in \mathbb{R}^3$  denotes the Cartesian position of the target point on the robot for linearization (e.g., end-effector or base location for tracking, center of the collision volume for collision avoidance). The dynamics are approximated as

$$z_{k+1} = Az_k + Bu_k, \quad x_k = Pz_k \quad (16)$$

where  $u_k$  denotes joint velocity commands,  $A, B$  are Koopman matrices, and  $P$  projects the lifted state back to the original coordinates. By wrapping the low-level torque dynamics (12) into the Koopman approximation (16), our controller operates at the velocity-control level. This design choice is crucial: it allows the use of first-order safety indices, simplifies safety constraint construction, and improves computational efficiency.

1) *Feedforward Prediction*: Given the current state  $x_t$ , we predict  $K$  steps forward using the learned Koopman operators  $A$  and  $B$ , which are represented as linear layers. We roll out the dynamics:

$$\hat{z}_{t+k+1} = A\hat{z}_{t+k} + B\hat{u}_{t+k}, \quad k = [0, \dots, K-1] \quad (17)$$

$$z_t = \psi_x(x_t) = [x_t; \psi_\omega(x_t)]^\top \quad (18)$$

$$x_{t+k} = Pz_{t+k} \quad (19)$$

$$u_{t+k} = \hat{u}_{t+k} \quad (20)$$

2) *K-steps Prediction Loss*: Since we don't have a separate network for control embedding  $\psi_u$ , we only train the state embedding network  $\psi_\omega$  and Koopman matrices  $A, B$  end-to-end using a  $K$ -step prediction loss. Given dataset  $\{x_i, u_i\}$  for  $i = 0, \dots, K$ , we compute embedded states  $z_i = \psi_x(x_i)$  and predicted lifted states  $\hat{z}_i$  via forward rollout.

The loss is:

$$\mathcal{L}_{\text{pred}}(\omega) = \sum_{i=1}^K \gamma^{i-1} \text{MSE}(z_i, \hat{z}_i) \quad (21)$$

where  $\gamma$  is a decay factor and MSE is the mean squared error [20].

3) *Safe Linear MPC Formulation*: Using the Koopman dynamics (16), the time derivative of safety constraint (14) and (15) can be expressed as:

$$\begin{aligned} \dot{\phi}_0 &= \frac{\partial \phi_0(\mathbf{x})}{\partial \mathbf{x}} \dot{\mathbf{x}} \\ &\approx \frac{\partial \phi_0(\mathbf{x})}{\partial \mathbf{x}} \frac{x_{k+1} - x_k}{\Delta t} \\ &= \nabla_x \phi_0(x_k)^\top \frac{(PA - P)z_k + PBu_k}{\Delta t} \end{aligned} \quad (22)$$

This expression is linear in  $u$  and can be directly inserted as a linear constraint in MPC. Considering quadratic cost with desired state  $x^{\text{des}}$ , the system becomes linear in lifted space and we can formulate a quadratic programming (QP) over horizon  $N$  as:

$$\begin{aligned} \min_{u_{0:N-1}} \quad & \sum_{k=0}^{N-1} \|Pz_k - x_k^{\text{des}}\|_Q^2 + \|u_k\|_R^2 + \|Pz_N - x_N^{\text{des}}\|_{Q_N}^2 \\ \text{s.t.} \quad & z_{k+1} = Az_k + Bu_k, \quad \forall k = 0, \dots, N-1 \\ & \nabla_x \phi_0(x_k)^\top \frac{(PA - P)z_k + PBu_k}{\Delta t} \leq b_\phi(x_k), \forall k \\ & z_0 = \psi_x(x_0), \quad u_k \in [u_{\min}, u_{\max}] \end{aligned} \quad (23)$$

##### B. Redesign the Safety Constraint for Persistent Feasibility

Although Koopman dynamics allows us to formulate a single QP for nonlinear systems, extra dimensions introduced by lifted space can cause feasibility issue. Let us take a more rigorous look at  $\dot{\phi}_0$ . First, matrices  $A, B$  can be decomposed as:

$$A = \begin{bmatrix} A_{xx} & A_{x\psi} \\ A_{\psi x} & A_{\psi\psi} \end{bmatrix}, B = \begin{bmatrix} B_x \\ B_\psi \end{bmatrix} \quad (24)$$

Given  $z_k = [x_k; \psi_\omega(x_k)]^\top$ , the lifted dynamics formulation for original state  $x$  becomes:

$$x_{k+1} = A_{xx}x_k + A_{x\psi}\psi_\omega(x_k) + B_x u_k \quad (25)$$

Extra term  $A_{x\psi}\psi_\omega(x_k)$  propagates into the time derivative computation of (14) and  $\dot{\phi}_0 = \nabla_x \phi_0^\top \dot{x} \leq b_\phi$  in (23) becomes:

$$\begin{aligned} & \frac{(p_{\text{ego}} - p_{\text{obs}})}{\|p_{\text{ego}} - p_{\text{obs}}\|} \frac{\partial p_{\text{ego}}}{\partial x} B_x u_k \leq b_\phi - \\ & \frac{(p_{\text{ego}} - p_{\text{obs}})}{\|p_{\text{ego}} - p_{\text{obs}}\|} \frac{\partial p_{\text{ego}}}{\partial x} (A_{xx}x_k + A_{x\psi}\psi_\omega(x_k) - x_k) \end{aligned} \quad (26)$$

These extra lifted contributions  $A_{x\psi}\psi_\omega(x_k)$  can shift the half-space constraint on  $u_k$  so that no bounded input  $u_k \in [u_{\min}, u_{\max}]$  can satisfy  $\dot{\phi}_0 \leq b_\phi$  for all active links simultaneously.

To adapt our safety specification to learned dynamics, we introduce Learner-Critic architecture for adversarial fine tuning. Here, we focus on the specific setting of a 7-DoF Kinova Gen3 manipulator for clarity and conciseness. Let

$d = \|p_{\text{ego}}(x) - p_{\text{obs}}\|_2$  and define a nonlinear safety index  $\phi(x) : \mathbb{R}^n \rightarrow \mathbb{R}$  in Cartesian coordinate as:

$$\phi_{n,\beta}(x) = d_{\min}^n - d^n + \beta d \quad (27)$$

where  $n, \beta \in \mathbb{R}$  are learnable parameters. This parameterization reduces the second order formulation in [21] to first order. Nevertheless, this updated formula will reshape the gradients in (26) to make it easier to find the corresponding  $u_k$  within the control limit and potentially conflicting constraints from multiple collision volumes. Let  $\Gamma \doteq (n, \beta)$  and learning is formulated as a min-max optimization problem:

$$\min_{\Gamma} \max_{x \in \partial \mathcal{S}} \mathcal{L}_{\text{infeas}}(\Gamma, x) \quad (28)$$

where the *infeasibility risk* is:

$$\mathcal{L}_{\text{infeas}}(\Gamma, x) := \min_{v \in \mathcal{V}(\mathcal{U})} \dot{\phi}_{\Gamma}(x, v) \quad (29)$$

Unlike the original formulation in [22], where the safety index  $\phi(x)$  is scalar and defined over the entire system, the application on the 7-DOF Kinova arm requires each link to be evaluated for safety independently. We define a linkwise safety index  $\phi^i(x)$  for each link  $i = 1, \dots, 7$ , and consider all of them as safety constraints. This linkwise definition introduces substantial complexity into both the constraint evaluation and the process of counterexample collection, compared to [22], where only a single constraint is considered.

To enable adversarial fine tuning under this setup, we modify a Learner–Critic architecture. The Critic attempts to collect boundary states and associated control counterexamples that violate the constraint  $\dot{\phi}^i(x, u) \leq 0$  for all active links at the safety boundary. However, instead of performing explicit Projected Gradient Descent (PGD) on *infeasibility risk* [23], our implementation uses a two-stage process that achieves an equivalent effect, adapted to the multi-link robotic setting.

First, boundary states are sampled using a differentiable optimization procedure. Specifically, in the presence of an obstacle in 3D space we minimize the distance between any of the robot link’s center-of-mass positions and the obstacle center:

$$\mathcal{L}_{\text{dist}}(x) = \sum_{i=1}^7 \max\{0, d_{\min} - \|p_i(x) - p_{\text{obs}}\|_2\} \quad (30)$$

We emphasize that this projection step is defined using the *geometric* safety specification  $d = d_{\min}$  (a 0.2 m clearance shell) rather than the learned level set  $\{\phi_{n,\beta} = 0\}$  as done in [22]. Anchoring counterexample collection to the physical clearance requirement ensures that the Critic explicitly probes where infeasibility first emerges for the original safety margin, while  $\phi_{n,\beta}$  is subsequently tuned to restore feasibility under Koopman dynamics. This is also empirically plausible as we are working in a MPC setting instead reactive safety filtering setting. Gradient descent is used to optimize  $x$  until this projection loss reaches a small threshold (empirically set to 0.0001), ensuring that the robot is brought close to the safety boundary. This serves as a

Scenario	# Infeasible / $T_{\text{exp}}$
Single Obstacle (Untrained)	108/4000
Multi Obstacle (Untrained)	632/4000
Single Obstacle ( <b>Trained</b> )	42/4000
Multi Obstacle ( <b>Trained</b> )	113/4000

TABLE I: QP infeasibility counts comparison. Single Obstacle includes one dynamic obstacle chasing one of Kinova’s links, whereas Multi Obstacle includes seven extra static obstacles placed at random positions. This experiment is conducted without slack relaxation. Each entry reports the number of infeasible QP solves among the total control steps in the experiment ( $T_{\text{exp}} = 4000$ ).

practical projection step, and thus effectively making this process a PGD.

Next, for each near-boundary state, the Critic samples control candidates from a set of saturated control vertices  $\mathcal{V}(\mathcal{U})$ . For each control  $u$ , it computes *infeasibility risk*. A counterexample is recorded only if no control in the admissible set satisfies:

$$\mathcal{L}_{\text{infeas}}^i(\Gamma, x) \leq 0 \quad \forall i \in \partial \mathcal{S} \quad (31)$$

for all links  $i$  in contact with the boundary.

To prevent the safety index from becoming overly conservative, which could increase infeasibility, we bound the Critic’s influence by allowing 10 trials to collect the desired number of counterexamples (e.g., 50). If this quota is not reached, we consider it sufficiently tuned. This guards against overfitting to pathological states, which would otherwise cause the Learner to reshape  $\phi$  in an attempt to satisfy constraints that cannot be satisfied, leading to an overly conservative and distorted safety index.

The Learner then performs a gradient update on the safety index parameters  $\Gamma$  by minimizing a combined loss:

$$\mathcal{L}_{\text{total}} = \frac{1}{M} \sum_{x \in \mathcal{B}} \mathbb{E}_{i \in \partial \mathcal{S}} [\mathcal{L}_{\text{infeas}}^i(\Gamma, x)] + \mathcal{L}_{\text{reg}} \quad (32)$$

where  $\mathcal{B}$  is the batch of counterexamples,  $M$  is the size of a batch,  $\mathcal{L}_{\text{reg}} = \mu \|\Gamma - \Gamma_0\|_2^2$  is a regularization loss penalizing deviation from heuristically chosen good initial values  $\Gamma_0 = (n_0, \beta_0)$ . The overall procedure is summarized in Algorithm 1.

The final MPC problem then becomes:

$$\begin{aligned} \min_{u_0:N-1} \quad & \sum_{k=0}^{N-1} \|Pz_k - x_k^{\text{des}}\|_Q^2 + \|u_k\|_R^2 + \|Pz_N - x_N^{\text{des}}\|_{Q_N}^2 \\ \text{s.t.} \quad & z_{k+1} = Az_k + Bu_k, \quad \forall k = 0, \dots, N-1 \\ & \nabla_x \phi_{n,\beta}^i(x_k)^\top \frac{(PA - P)z_k + PBu_k}{\Delta t} \leq b_\phi(x_k), \forall i, k \\ & z_0 = \psi_x(x_0), \quad u_k \in [u_{\min}, u_{\max}] \end{aligned} \quad (33)$$

The effectiveness of adversarial fine tuning in reducing infeasible cases is quantified in Table I, which compares QP infeasibility counts across different obstacle settings before and after training.

### C. Sim to Real Adaptation

To bridge the gap between simulation and hardware, we first quantify the sim-to-real mismatch by executing identical reference trajectory tracking experiments in PyBullet

---

**Algorithm 1** Adversarial Fine Tuning
 

---

**Require:**  $\Gamma_0 = (n_0, \beta_0)$ ,  $p_{\text{obs}}$ ,  $d_{\text{min}}$  regularization weight  $\mu$ , batch size  $N_{\text{batch}}$ , trials for Critic  $N_{\text{trials}}$

```

1: function COLLECTCE( $\Gamma$ ,  $p_{\text{obs}}$ ,  $d_{\text{min}}$ ,  $N_{\text{batch}}$ ,  $N_{\text{trials}}$ )
2:   Set learning rate  $\alpha_c$ ,  $N_{\text{collected}} = 0$ 
3:   for  $N_{\text{trials}}$  do
4:      $X \leftarrow$  uniformly sample within operating region
5:      $X \leftarrow X + \alpha_c \cdot \nabla_X \mathcal{L}_{\text{dist}}$ 
6:      $X_{\text{CE}} \leftarrow \min_{v \in \mathcal{V}(\mathcal{U})} \mathcal{L}_i > 0 \forall i = 1, \dots, 7$ 
7:     if  $N_{\text{collected}} \geq N_{\text{batch}}$  then
8:       return  $(X_{\text{CE}}, U_{\text{CE}})$ 
9:     end if
10:  end for
11:  Training complete
12: end function
13: function UPDATE( $\Gamma$ ,  $X_{\text{CE}}$ ,  $U_{\text{CE}}$ )
14:  Set learning rate  $\alpha_l$ 
15:   $\Gamma \leftarrow \Gamma + \alpha_l \cdot \nabla_{\Gamma} \mathcal{L}_{\text{total}}$  (from Eq. (32))
16:  return  $\Gamma$ 
17: end function
18: function MAIN
19:   $\Gamma = (n_0, \beta_0)$ 
20:  while not done do
21:     $(X_{\text{CE}}, U_{\text{CE}}) \leftarrow$  CollectCE( $n$ ,  $\beta$ )
22:     $\Gamma \leftarrow$  Learn( $\Gamma$ ,  $X_{\text{CE}}$ ,  $U_{\text{CE}}$ )
23:  end while
24:  return  $\Gamma$ 
24: end function

```

---

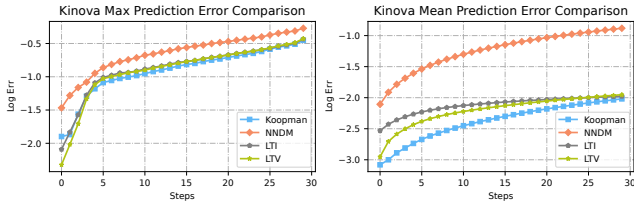


Fig. 4: Prediction Error Comparison. The analytic baselines (LTI and LTV), KDM, and NNNDM are compared against PyBullet ground truth. KDM maintains the lowest long-horizon error growth, while the analytic model is competitive in short horizons but diverges.

simulation and on the real robot. The deviation in tracking performance serves as a measure of model discrepancy. Instead of retraining the entire Koopman embedding and operator matrices, we collect hardware data and fine-tune only the  $A$  and  $B$  matrices of the lifted linear dynamics (16). This lightweight adaptation step efficiently accounts for unmodeled dynamics differences between simulation and hardware, enabling migration with minimal retraining.

## V. EXPERIMENTS

In this section, we conduct experiments using a Kinova Gen3 manipulator and a Unitree Go2 to evaluate our approach.

### A. Experiment Setup

For all dynamics models considered—Koopman Dynamics Model (KDM), Neural Network Dynamics Model (NNNDM) [15], and the analytic baselines—the state is defined as the concatenation of target link position  $p_k$  and joint angles  $q_k$ :  $x_k = [p_k; q_k]^\top$ .

1) *Analytic Baselines:* We compare two analytic formulations: a time-invariant (LTI) model and a time-varying (LTV) variant. Both share the same state update for the joint angles and end-effector:

$$p_{k+1} = p_k + \Delta t J(\bar{q}_k) u_k \quad (34)$$

$$q_{k+1} = q_k + \Delta t u_k \quad (35)$$

where  $u_k$  is the joint velocity command and  $J(\bar{q}_k)$  denotes the manipulator Jacobian evaluated at a reference configuration  $\bar{q}_k$ . For the LTI baseline,  $\bar{q}_k = q^*$  is fixed, giving a globally linear but coarse approximation. For the LTV system,  $\bar{q}_k = q_k$  is updated at each step, yielding a locally valid linearization that improves tracking while keeping  $q_k$  integrated directly. This LTV formulation supports an MPC controller (LTV MPC) that bridges the gap between a static linear model and full nonlinear MPC.

2) *Learned Models:* We train NNNDM and KDM within PyBullet simulation. Both employ fully connected feedforward networks with hidden layers [256, 256, 256] for either the lifting function  $\psi_\omega$  (KDM) or direct dynamics mapping (NNNDM). Fig. 4 compares the prediction errors across models, showing that the Koopman-based network achieves the lowest prediction error. Once trained, we formulate linear MPC for KDM (KMPC), LTI, and LTV baselines, and shooting-based NMPC for the other models. For all non-Koopman models, a safety filter is additionally applied to enforce state constraints.

3) *Controller and solver setup:* All QP-based controllers (KMPC, LTIMPC, LTV MPC, and the safety filters) are solved using OSQP. For a fair comparison, we use the same MPC horizon  $H = 9$  for all methods with control input a 7D joint velocity command  $u_k \in \mathbb{R}^7$  along with box limits matching the Kinova joint velocity saturation. We use a first-order safety bound with  $\lambda = 0.05$  in all rollouts. For numerical stability, slack relaxation is used whenever we enforce safety constraints via a QP: KMPC includes slack directly in its unified QP, while the analytic baselines and NNNDM apply slack in the separate safety-filter QP.

### B. Safe Control in 2D Space

This experiment incorporates collision avoidance of the end-effector as a safety constraint. We use the first-order safety index from (13) with  $d_{\text{min}} = 0.2$  for NNNDM and the analytic baseline, and the adapted safety index from (27) for KDM. The KDM control law solves the QP in (23), whereas MPC for the other models solve equivalent tracking problem in original state space with the safety filter formulated as:

$$\begin{aligned}
\min_{u_k} \quad & \|u_k - u_k^{\text{ref}}\|^2 \\
\text{s.t.} \quad & \nabla_x \phi_0(x_k)^\top \frac{x_{k+1} - x_k}{\Delta t} \leq b_\phi(x). \quad (36)
\end{aligned}$$

We employ a 9-step horizon for all models to ensure consistent comparison. Results depicted in Fig. 5 demonstrate effective obstacle avoidance and accurate tracking of KMPC.

	Single Obstacle	Multi Obstacle (6 total)	Avg. Distance to Target [m]	Avg. Max $\phi$ Over Links	Avg. Mean $\phi$ Over Links	Avg. Min Dist. to Obstacle [m]	Cumulative Cost
	Avg. Comp. Time [s]	Avg. Comp. Time [s]					
<b>KMPC (Ours)</b>	9.66e-3 $\pm$ 0.15e-3	1.496e-2 $\pm$ 0.23e-3	0.071860	-0.03828	-0.21492	0.21913	13154
LTIMPC	5.19e-3 $\pm$ 0.43e-3	0.629e-2 $\pm$ 0.79e-3	0.572857	-0.15237	-0.33137	0.32237	307378578
LTVMPC	14.98e-3 $\pm$ 0.12e-3	0.629e-2 $\pm$ 0.79e-3	0.144116	-0.09723	-0.18294	0.26745	93292
NMPC-10	40.49e-3 $\pm$ 0.96e-3	5.299e-2 $\pm$ 3.06e-3	0.116499	-0.16023	-0.25239	-	81261
NMPC-100	412.59e-3 $\pm$ 227.05e-3	1.04704 $\pm$ 0.03220	0.009787	-0.12234	-0.25224	-	160

TABLE II: Computation Time and Performance Comparison for an episode length of 4000 control steps. The two left columns report average computation time for safe control, while the remaining columns summarize tracking performance and safety in the 3D space. Cumulative cost reflects tracking performance. For safety, **Avg. Max  $\phi$**  denotes the maximum  $\phi$  over all links at each time step, averaged over the episode, and **Avg. Mean  $\phi$**  denotes the mean  $\phi$  over links at each time step, averaged over the episode; in both cases, lower values indicate safer behavior. **Avg. Min Dist. to Obstacle** reports the minimum link-to-obstacle distance averaged over the episode, where larger values indicate safer separation. The symbol “-” indicates that the robot collided with the obstacle during the experiment. Across these complementary metrics, KMPC achieves the *best performance-safety trade-off*, attaining low tracking cost while making only necessary evasion, thereby maintaining moderately low  $\phi$  (both Avg. Max and Avg. Mean).

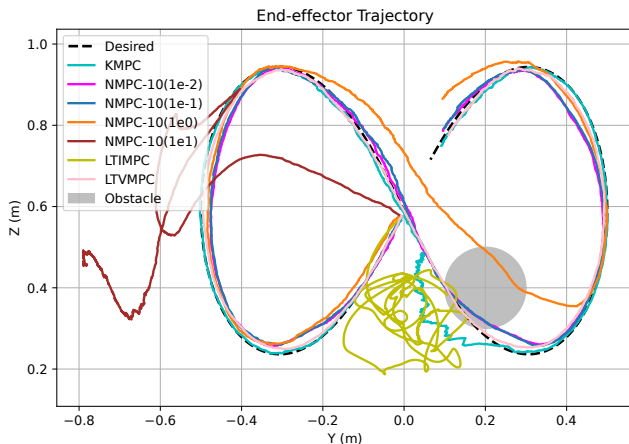


Fig. 5: Safe Tracking Comparison. NMPC-10 is nonlinear MPC with 10 times shooting, values in parentheses represent slack weight for safety filter relaxation. Linear counterparts, including KMPC did not require slack relaxation for this experiment. KMPC outperforms NMPC-10 both in tracking and safety constraint satisfaction.

### C. Safe Control in 3D Space

In this scenario, the robot is tasked with tracking a reference end-effector trajectory in full 3D space while avoiding static obstacles placed along the path. This setting requires simultaneous trajectory tracking and obstacle avoidance across all seven links. The MPC optimization problem is identical except for the linkwise-extended safety constraint:

$$\dot{\phi}^i(x, u) \leq b_\phi(x), \quad \forall i = 1, \dots, 7. \quad (37)$$

The performance results in Table II illustrate robust trajectory tracking with obstacle evasion, validating the practicality of KMPC frameworks for safe robot control. While KMPC does not attain the numerically best value for every individual safety metric ( $\phi$ ) or cumulative cost, the aggregate results indicate that KMPC selects the most favorable operating point between aggressive tracking and conservative safety among all baselines. LTI and LTV baselines show poor tracking cost (over  $7.1\times$  higher than KMPC) due to model mismatch and increased penalty from constraint handling, while NMPCs risks colliding into the obstacle. The proposed KMPC also achieves faster computation compared to NMPC (over  $4.2\times$  faster) by exploiting linear Koopman dynamics, whereas

shooting-based MPC becomes computationally impractical in these scenarios.

### D. Sim-to-Real Adaptation

To evaluate the robustness of our approach beyond simulation, we deployed the fine-tuned Koopman-based safe control pipeline directly on the Kinova Gen3 manipulator. Figure 6 compares state prediction errors before and after fine-tuning, showing substantial reduction in both joint angle and end-effector position errors. Importantly, this improvement is achieved by adapting only the linear Koopman operators while keeping the embedding fixed, suggesting that the learned latent coordinates remain transferable and that the dominant sim-to-real mismatch is captured by linear dynamics in lifted space. These results confirm that the proposed methodology not only improves model accuracy in simulation but also transfers effectively to real hardware without retraining the embedding. Our experiments demonstrate that the unified pipeline, combining Koopman lifting and adversarial safety index tuning, achieves reliable performance on the physical system. A supplementary demo video is provided, illustrating safe control execution on hardware and validating the practical deployability of the proposed framework.

Collectively, these experiments demonstrate the versatility and effectiveness of our proposed Koopman-based control methodology, from simple trajectory tracking to dynamic safety-critical scenarios. It is worth noting that since safety specification is derived from Koopman space, successful evasion also depicts the accuracy in model approximation.

### E. Extension to Floating-Base Systems

Unlike fixed-base manipulators, floating-base robots introduce additional unactuated base degrees of freedom whose motion is coupled with the actuated joints, so the state must explicitly include the base pose/velocity and safety constraints must be enforced over the full-body (base + joints) kinematics. Due to limited space, we provide the complete formulation details and additional Unitree Go2 results in the appendix of our extended version at [18].

## VI. FUTURE WORK AND CONCLUSION

While the proposed Koopman-based control framework shows promising results, several limitations remain and mo-

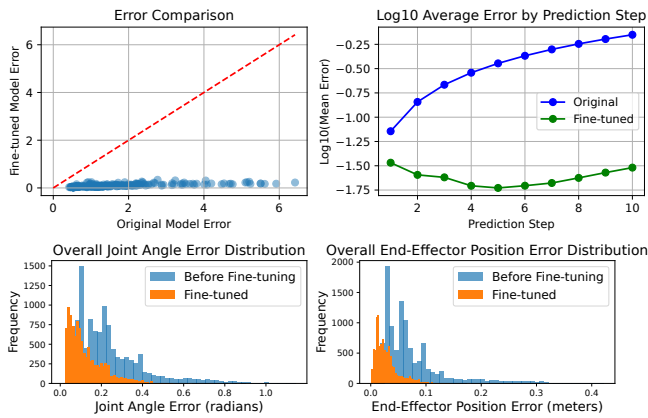


Fig. 6: Prediction error comparison for Kinova-Gen3 before and after fine-tuning. Top: (Left) Scatter plot of original versus fine-tuned model errors, where nearly all points lie below the  $y = x$  line (red), indicating consistent improvement. (Right) Average prediction error across horizons ( $\log_{10}$  scale), showing that fine-tuned Koopman dynamics maintain significantly lower error over multiple steps. Bottom: Error distribution over all samples, comparing before and after fine-tuning for (Left) joint angles and (Right) end-effector positions, both of which show markedly reduced errors after fine-tuning.

tivate future directions. Our current formulation relies on a first-order safety index based on positional information, which may be insufficient for dynamic or fast-evolving environments. Future work will explore higher-order safety indices that incorporate velocity and curvature, likely requiring torque- or acceleration-level control. These extensions will further improve the scalability, safety, and real-world viability of Koopman-based safe control.

This project presents a data-driven safe control framework that combines Koopman operator theory, SSA, and adversarial fine tuning to enable efficient, verifiable control of nonlinear robotic systems. By lifting nonlinear dynamics into a latent linear space, we exploit linear control techniques while enforcing safety constraints through a single QP. We validate the approach on the Kinova Gen3 manipulator and Unitree Go2 in tasks ranging from trajectory tracking to dynamic obstacle avoidance with multi-step MPC, demonstrating reliable tracking and real-time safety. The framework’s linear structure also facilitates adaptive extensions and hardware deployment. Looking forward, we aim to incorporate higher-order safety indices. Overall, Koopman-based safe control offers a scalable and interpretable alternative to conventional model-free methods, particularly in safety-critical robotics.

#### ACKNOWLEDGMENT

This work was in part supported by the National Science Foundation (NSF) under Grant No. 2144489.

#### REFERENCES

- [1] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, “From linear to nonlinear mpc: bridging the gap via the real-time iteration,” *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.
- [2] P. J. Schmid, “Dynamic mode decomposition of numerical and experimental data,” *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.
- [3] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, “A data-driven approximation of the koopman operator: Extending dynamic mode decomposition,” *Journal of Nonlinear Science*, vol. 25, pp. 1307–1346, 2015.

- [4] M. Korda and I. Mezić, “Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control,” *Automatica*, vol. 93, pp. 149–160, 2018.
- [5] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.
- [6] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, “Embed to control: A locally linear latent dynamics model for control from raw images,” *Advances in neural information processing systems*, vol. 28, 2015.
- [7] F. Li, A. Abuduweili, Y. Sun, R. Chen, W. Zhao, and C. Liu, “Continual learning and lifting of koopman dynamics for linear control of legged robots,” in *Proceedings of the 7th Annual Learning for Dynamics & Control Conference*, ser. Proceedings of Machine Learning Research, N. Ozay, L. Balzano, D. Panagou, and A. Abate, Eds., vol. 283. PMLR, 04–06 Jun 2025, pp. 136–148.
- [8] B. Lusch, J. N. Kutz, and S. L. Brunton, “Deep learning for universal linear embeddings of nonlinear dynamics,” *Nature communications*, vol. 9, no. 1, p. 4950, 2018.
- [9] H. Chen, A. ABUDUWEILI, A. Agrawal, Y. Han, H. Ravichandar, C. Liu, and J. Ichnowski, “Korol: Learning visualizable object feature with koopman operator rollout for manipulation,” in *Conference on Robot Learning*. PMLR, 2025, pp. 4509–4524.
- [10] H. K. Khalil and J. W. Grizzle, *Nonlinear systems*. Prentice hall Upper Saddle River, NJ, 2002, vol. 3.
- [11] Y. Tassa, T. Erez, and E. Todorov, “Synthesis and stabilization of complex behaviors through online trajectory optimization,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.
- [12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [13] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, “End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks,” 2019. [Online]. Available: <https://arxiv.org/abs/1903.08792>
- [14] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs for safety critical systems,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.
- [15] T. Wei and C. Liu, “Safe control with neural network dynamic models,” in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 739–750.
- [16] V. Zinage and E. Bakolas, “Neural koopman control barrier functions for safety-critical control of unknown nonlinear systems,” 2022. [Online]. Available: <https://arxiv.org/abs/2209.07685>
- [17] K. P. Wabersich and M. N. Zeilinger, “A predictive safety filter for learning-based control of constrained nonlinear dynamical systems,” *Automatica*, vol. 129, p. 109597, 2021.
- [18] S. Jung, A. Abuduweili, J. Li, and C. Liu, “Whole-body safe control of robotic systems with koopman neural dynamics,” 2026. [Online]. Available: <https://arxiv.org/abs/2603.03740>
- [19] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, “Modern koopman theory for dynamical systems,” *arXiv preprint arXiv:2102.12086*, 2021.
- [20] H. Shi and M. Q.-H. Meng, “Deep koopman operator with control for nonlinear systems,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7700–7707, 2022.
- [21] C. Liu and M. Tomizuka, “Control in a safe set: Addressing safety in human-robot interactions,” in *Dynamic Systems and Control Conference*, vol. 46209. American Society of Mechanical Engineers, 2014, p. V003T42A003.
- [22] S. Liu, C. Liu, and J. Dolan, “Safe control under input limits with neural control barrier functions,” in *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 1970–1980. [Online]. Available: <https://proceedings.mlr.press/v205/liu23e.html>
- [23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” *arXiv preprint arXiv:1706.06083*, 2017.