

# Viability-Preserving Passive Torque Control

Anonymous Author(s)

**Abstract**—Conventional passivity-based torque controllers for manipulators are typically unconstrained, which can lead to safety violations under external perturbations. In this paper, we employ viability theory to pre-compute safe sets in the state-space of joint positions and velocities. These viable sets, constructed via data-driven and analytical methods for self-collision avoidance, external object collision avoidance and joint-position and joint-velocity limits, provide constraints on joint accelerations and thus joint torques via the robot dynamics. A quadratic programming-based control framework enforces these constraints on a passive controller tracking a dynamical system, ensuring the robot states remain within the safe set in an infinite time horizon. We validate the proposed approach through simulations and hardware experiments on a 7-DoF Franka Emika manipulator. In comparison to a baseline constrained passive controller, our method operates at higher control-loop rates and yields smoother trajectories. Project Website: [vpp-tc.github.io](https://vpp-tc.github.io)

## I. INTRODUCTION

Physical human-robot interaction (pHRI) is ubiquitous in industrial manufacturing and domestic environments, where effective operation hinges on close coordination between humans and robots. To operate safely under physical contact, robots are required to admit external perturbations while preserving closed-loop stability. This necessitates the design of passive torque controllers. Passivity is characterized by absorbing at least as much energy as is released. This property endows robots with robustness and stability when subjected to external disturbances [1]. The classical impedance/admittance framework ensures passivity under appropriate damping [2], [3], and port-Hamiltonian methods [4] achieve it via energy shaping. The energy-tank framework monitors next exchanged energy, accumulates dissipation in a virtual reservoir, and authorized controller adaptations only with a nonnegative energy budget, making it suitable for passivity-preserving variable-impedance control [5]. In scenarios where trajectories are specified via a dynamical-systems (DS) representation [6], [7], [8] proposed a passive torque controller for trajectory tracking.

Nevertheless, such passivity-based controllers generally lack explicit constraint handling, so collision avoidance and joint-limit satisfaction cannot be guaranteed throughout the entire mission execution. To address this problem, [9] (CPIC) leverages the Control Barrier function (CBF) framework [10] and formulates a hierarchically prioritized quadratic program (QP) with hard and soft constraints that constrains a passive torque controller. Within this formulation, joint-limit, self-collision avoidance, external object collision avoidance, and singularity avoidance are treated as safety-critical constraints by defining joint-space analytic or data-driven barrier functions and encoding them as torque-

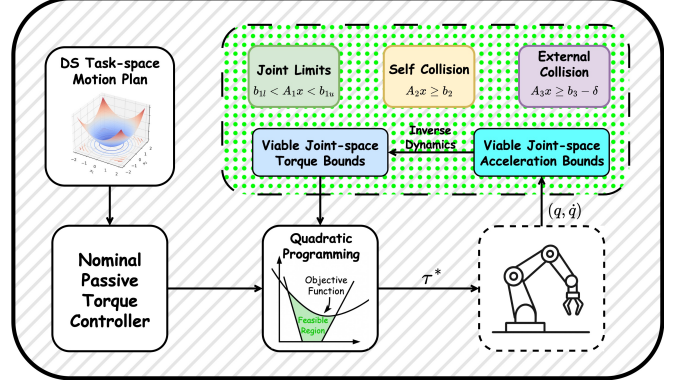


Fig. 1. Schematic of the proposed viability-preserving, passivity-based torque controller. Joint-limit torque bounds are derived analytically, whereas bounds for self-collision and external-object collisions are data-driven.

level constraints via Exponential Control Barrier Functions (ECBF) [11]. Nonetheless, although this control framework explicitly constrained the torques generated by the passive controller, it has several drawbacks. Although the constraints are partitioned into soft and hard categories, mutual conflicts among the hard constraints can still occur, rendering the QP infeasible. In addition, the QP framework is susceptible to deadlock and entrapment in local minima in the joint space. A further computational challenge arises when determining the parameters of the ECBF: because the safety boundary is learned from data and defined at the joint level while the constraints are imposed at the torque level, one must compute the Hessian of the boundary function with respect to the joint variables. When this boundary is represented by neural networks, as in [9], obtaining real-time second-order information becomes challenging.

**Contributions** In this work, inspired by [9], [12], [13], we introduce safety bounds in an augmented state space comprising joint positions and joint velocities. Operating in this space allows us to derive bounds on joint accelerations, and thus on joint torques via the robot dynamics, without requiring second-order derivatives of the boundary functions. We formalize these bounds using the control-theoretic notion of viability, which guarantees controlled forward invariance over an infinite time horizon for some control input sequences. We construct viable set for self-collision and external-object collision avoidance through data-driven methodologies, and we obtain viable sets for joint-position and joint-velocity limits through analytical derivations (in the spirit of [13]). Building on these sets, we develop a passivity-preserving torque-control framework (Fig. 1) that maintains state viability by solving a QP with torque-level constraints induced by the viable sets; which activate adaptively as the

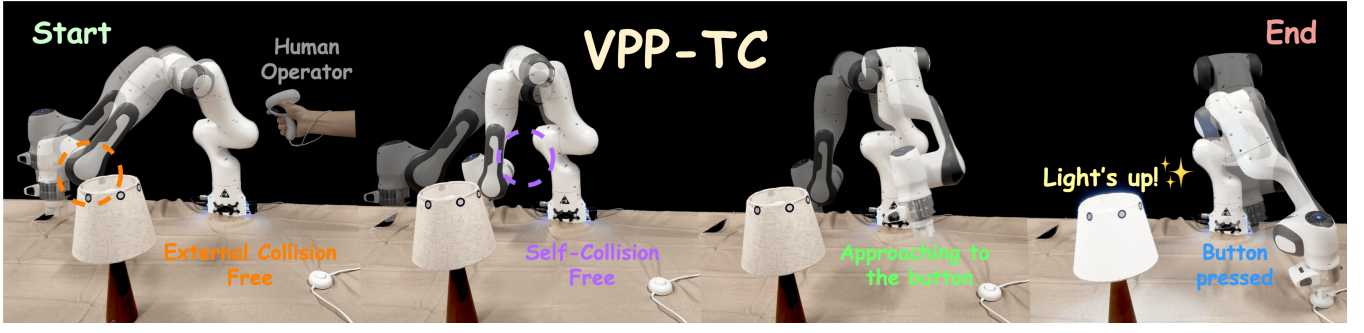


Fig. 2. Real-world teleoperation task - pressing a desk lamp button to turn it on. The human operator provides only coarse target positions (no precomputed collision-free trajectory). Our proposed torque controller **VPP-TC** ensures safety near the lamp and prevents self-collision while accomplishing the task.

state approaches the corresponding viability boundaries.

## II. PRELIMINARIES

### A. Dynamical System Motion Planning

We adopt a Dynamical System (DS) formulation for task-space planning [6]. Let  $x \in \mathbb{R}^d$  denote the task-space state, whose nominal evolution is governed by the autonomous ODE  $\dot{x} = f(x)$ ,  $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ , where  $f$  is continuously differentiable and stable attractor  $x^* \in \mathbb{R}^d$ , i.e.,

$$\exists \delta_f, \text{ s.t. } \lim_{t \rightarrow \infty} \|x(t) - x^*\| \leq \delta_f, \quad (1)$$

where  $\delta_f$  is a positive real number. Accordingly, we set the desired task-space velocity to  $\dot{x}_d = f(x)$ .

### B. Passivity-Based Control with Dynamical Systems [8]

To enforce task-space passivity while following a desired Dynamical System (DS) motion plan, we use a velocity-based impedance controller:

$$F_c = G_x(x) - D(x)(\dot{x} - f(x)), \quad (2)$$

where  $G_x(x) = J(q)^{-\top} G(q)$  is the gravity term expressed in task space,  $D(x) \in \mathbb{R}^{d \times d}$  is a state-dependent damping matrix,  $\dot{x}$  is measured task-space velocity, and  $f(x)$  is the desired DS velocity. Intuitively, the gravity term compensates potential energy due to gravity, and the damping term applies anisotropic dissipation to the velocity error. With a suitable  $D(x)$ , the controller injects power only along the desired motion  $f(x)$  and dissipates power in directions orthogonal to it, which yields passive behavior while tracking the DS.

**Assumption 1:** The vector field  $f(x)$  is the gradient flow of a scalar potential  $\mathcal{P}(x)$ , that is,  $f(x) = -\nabla_x \mathcal{P}(x)$ . The damping matrix is positive semidefinite and factorizes as  $D(x) = V(x)\Lambda V(x)^\top$ , with  $\Lambda$  a diagonal matrix of non-negative entries. The columns of  $V(x)$  form an orthonormal frame chosen so that  $v_1 = f(x)/\|f(x)\|$  aligns with the desired direction of motion, while the remaining columns span its orthogonal complement.

Under these conditions, the negative velocity-error feedback steers  $\dot{x}$  towards  $f(x)$ , producing kinetic energy only along the DS direction and dissipating it elsewhere, thereby achieving task-space passivity during tracking.

### C. Safety Guaranteed by Viability

We define the set of viable states for an  $n$ -DoF manipulator as the Cartesian product of a set of  $q \in \mathbb{R}^n$  (joint positions) and  $\dot{q} \in \mathbb{R}^n$  (joint velocities),  $(q, \dot{q}) \in \mathcal{V}$ . A state  $(q, \dot{q}) \in \mathbb{R}^{2n}$  is viable if from any initial state within  $\mathcal{V}$ , there exists a control sequence  $\ddot{q}$  (acceleration) that generates an infinite state sequence which is inside the viable set, whereas a merely feasible state may leave the feasible set under future control sequences (Fig. 3) [12], [13]. We formally define the viable states and viable set as:

**Definition 1 (Viability):**

$$(q(0), \dot{q}(0)) \in \mathcal{V} \Leftrightarrow \exists (\ddot{q}_i)_{i=0}^{\infty} : (q(t), \dot{q}(t)) \in \mathcal{F}, \forall t \geq 0; \\ |\ddot{q}_k| \leq \ddot{q}_k^{\max}, k = 1, 2, \dots, n, \quad (3)$$

where  $\ddot{q}_k^{\max}$  represents hardware joint acceleration limits of the  $k$ -th joint.  $\mathcal{F} \in \mathbb{R}^{2n}$  denotes the feasible set of state pairs  $(q, \dot{q})$  that satisfy all constraints.

We define the safe set as the viable state set  $\mathcal{V}$ .

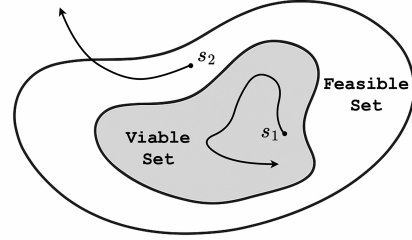


Fig. 3. Viability concept:  $s_1$  is feasible and viable, whereas  $s_2$  is feasible but non-viable.

**Assumption 2:** The initial state  $(q_0, \dot{q}_0)$  is assumed to be within the viable state set  $\mathcal{V}$ .

## III. PROBLEM FORMULATION

We seek to control an  $n$ -DoF manipulator with rigid-body dynamics derived by the Euler-Lagrange equation:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau_c + \tau_{ext}, \quad (4)$$

where  $\ddot{q} \in \mathbb{R}^n$  denote joint accelerations.  $M(q) \in \mathbb{R}^{n \times n}$ ,  $C(q, \dot{q}) \in \mathbb{R}^{n \times n}$ ,  $G(q) \in \mathbb{R}^n$  denote the inertia matrix, Coriolis matrix, and gravity respectively.  $\tau_c \in \mathbb{R}^n$  and  $\tau_{ext} \in \mathbb{R}^n$  denote the control input and external torque applied to the joints.

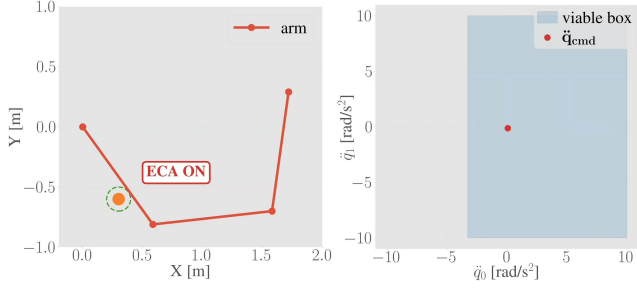


Fig. 4. Planar 3-DoF demo. Left: workspace trajectory. Right: viable box of  $(\dot{q}_0, \dot{q}_1)$  under hardware limits  $[-10, 10]$  rad/s<sup>2</sup>. With ECA enabled, the admissible box contracts as the obstacle constraint activates, illustrating viability-induced acceleration bounds.

The goal of this paper is to provide torque bounds on nominal passive torque controllers [8]. The torque bounds are intended to guarantee viability [12], [13] for joint space constraints, whose feasible sets are formulated as

- Joint Position and Velocity Limits:

$$\mathcal{F}_{\text{jnt}} := \{(q, \dot{q}) \mid q^- \leq q \leq q^+, |\dot{q}| \leq \dot{q}_{\text{max}}\}.$$

where  $q^-, q^+ \in \mathbb{R}^n$  denote the per-joint lower/upper position bounds,  $\dot{q}_{\text{max}} \in \mathbb{R}_+^n$  denotes the per-joint maximum velocity, and the inequalities are interpreted elementwise.

- Self-Collision Avoidance:

$$\mathcal{F}_{\text{sca}} := \{(q, \dot{q}) \mid q \in \mathcal{C}_{\text{sca}}\},$$

where  $\mathcal{C}_{\text{sca}} \subset \mathbb{R}^n$  is the set of self-collision-free configurations.

- External Obstacle Collision Avoidance:

$$\mathcal{F}_{\text{eca}} := \{(q, \dot{q}) \mid q \in \mathcal{C}_{\text{eca}}\},$$

where  $\mathcal{C}_{\text{eca}} \subset \mathbb{R}^n$  is the set of external-collision-free configurations.

According to (3), we can then obtain the separated viable state set  $\mathcal{V}_{\text{jnt}}$ ,  $\mathcal{V}_{\text{sca}}$  and  $\mathcal{V}_{\text{eca}}$ . The final viable state set is given by the intersection  $\mathcal{V} = \mathcal{V}_{\text{jnt}} \cap \mathcal{V}_{\text{sca}} \cap \mathcal{V}_{\text{eca}}$  which is defined as the safe state set.

**Problem 1:** Consider a torque controlled manipulator with dynamics (4). Design a passive torque controller that guarantees safety by preserving the state  $(q, \dot{q})$  within the viable state set  $\mathcal{V}$ .

#### IV. PROPOSED APPROACH

Our proposed approach handles the three aforementioned forms of collision avoidance in a unified manner by testing whether the current state belongs to the viable set. Whenever the state is outside the viable set, we formulate and solve an optimization problem to return it to the viable set. We refer to this method as the *Viability-Preserving Passive Torque Controller (VPP-TC)*, which we will use for brevity throughout the remainder of the paper. A planar 3-DoF example illustrating viability-induced acceleration bounds is shown in Fig. 4. The viable box for  $(\dot{q}_0, \dot{q}_1)$  shrinks from the hardware limits  $[-10, 10]$  rad/s<sup>2</sup> as the ECA constraint activates.

#### A. Self-Collision Avoidance Constraints

We conservatively define a joint state as self-collision-free viable if, under maximum deceleration opposite to the current joint velocity, the robot can come to a complete stop without any self-collision throughout the braking trajectory.

**Definition 2 (Self-collision-free viability):** Let  $\mathcal{C}_{\text{sca}} \subset \mathbb{R}^n$  be the set of self-collision-free configurations. Given an initial state  $(q_0, \dot{q}_0) \in \mathbb{R}^{2n}$ , define the *braking control*  $u^{\text{br}}(t)$  that applies maximum deceleration opposite to the current velocity, element-wise, until stop, and let  $(q(t), \dot{q}(t))$  be the corresponding trajectory. The *braking time*  $T_{\text{br}}(q_0, \dot{q}_0)$  is the first time such that  $\|\dot{q}(T_{\text{br}})\| = 0$ , and the *braking trajectory set* is

$$\mathcal{C}_{\text{br}}(q_0, \dot{q}_0) := \{(q(t), \dot{q}(t)) \mid \forall t \in [0, T_{\text{br}}]\}. \quad (5)$$

We then define the conservative self-collision viable set as

$$\mathcal{V}_{\text{sca}} := \{(q_0, \dot{q}_0) \mid q(t) \in \mathcal{C}_{\text{sca}}, \forall t \in [0, T_{\text{br}}]\}.$$

**Learning  $\mathcal{V}_{\text{sca}}$ :** To learn such conservative viability from data, we employ a transformer-based neural network to map states  $(q, \dot{q})$  to a real value  $\Gamma(q, \dot{q}) : \mathbb{R}^{2n} \rightarrow \mathbb{R}$ . The network first applies a linear projection and learned positional encodings to the input  $(q, \dot{q})$ , followed by a multi-layer transformer encoder that processes the sequence of state embeddings. The output is aggregated via global average pooling and passed through an MLP classifier with GeLU activation to produce the logits  $(\ell_1, \ell_2)$ , from which  $\Gamma(q, \dot{q})$  is defined as  $\Gamma(q, \dot{q}) = \ell_1 - \ell_2$ .

This mapping is used to define an *empirical surrogate* of the viable state set

$$\hat{\mathcal{V}}_{\text{sca}} := \{(q, \dot{q}) \mid \Gamma(q, \dot{q}) > 0\}.$$

For scenarios where  $\Gamma(q, \dot{q}) \in (0, \epsilon_{\text{sca}}]$ , with  $\epsilon_{\text{sca}} \in \mathbb{R}_{++}$  denoting a small positive scalar, we impose a constraint enforcing  $\Delta\Gamma(q, \dot{q}) \geq 0$ , thereby ensuring that the state remains within the viable state set. By Taylor expansion and omitting high-order terms, we have

$$\begin{aligned} \Delta\Gamma(q, \dot{q}) &= \nabla_q \Gamma^\top \cdot \Delta q + \nabla_{\dot{q}} \Gamma^\top \cdot \Delta \dot{q} \\ &= \nabla_q \Gamma^\top \cdot (\dot{q}\delta t + \frac{1}{2}\ddot{q}\delta t^2) + \nabla_{\dot{q}} \Gamma^\top \cdot (\ddot{q}\delta t) \\ &= (\frac{1}{2}\nabla_q \Gamma^\top \cdot \delta t^2 + \nabla_{\dot{q}} \Gamma^\top \cdot \delta t)\ddot{q} + \nabla_q \Gamma^\top \cdot \dot{q}\delta t, \end{aligned} \quad (6)$$

which is an affine function of  $\ddot{q}$ . For simplicity, we define  $g_{\text{se}} = \frac{1}{2}\nabla_q \Gamma^\top \cdot \delta t^2 + \nabla_{\dot{q}} \Gamma^\top \cdot \delta t$  and  $b_s = \nabla_q \Gamma^\top \cdot (\dot{q}\delta t)$ . Then we have the constraint of  $\ddot{q}$  as  $g_{\text{se}}\ddot{q} + b_s \geq 0$ . Plugging this acceleration constraint into the robot dynamics (4), we have the torque constraint as:

$$g_{\text{se}} M^{-1} \tau \geq g_{\text{se}} M^{-1} (C\dot{q} + G) - b_s \quad (7)$$

#### B. External Obstacle Collision Avoidance Constraints

We employ the Bernstein Polynomial representation [14] to encode the SDF of every robot link. Let  $\Omega_n$  denote the  $n$ -th link of a robot, for a query point  $p \in \mathbb{R}^3$  expressed in the base frame, the distance from  $p$  to the entire robot

surface at configuration  $q$  is the pointwise minimum over all links:

$$S(p, q) = \min_{n=1, \dots, N} S_{\Omega_n}^b(p, q). \quad (8)$$

Then we define the viability-preserving SDF as the minimum instantaneous SDF encountered along the braking trajectory  $\mathcal{C}_{br}(q_0, \dot{q}_0)$  defined in subsection IV-A:

$$S_v(p, q_0, \dot{q}_0) = \min_{t \in [0, T_{br}]} S(p, q(t)). \quad (9)$$

Following [14], we replace the nondifferentiable min in Eqs. 8-9 with a differentiable smooth approximation; in our implementation we use a soft-min (log-sum-exp) surrogate to ensure differentiability and avoid gradient discontinuities.

**Definition 3 (External-collision-free viability):** Similar to subsection IV-A, we define the conservative external-collision viable set as

$$\mathcal{V}_{eca} := \{ (q_0, \dot{q}_0) \mid q(t) \in \mathcal{C}_{eca}, \forall t \in [0, T_{br}] \}.$$

Also, we use the viability-preserving SDF  $S_v(p, q, \dot{q})$  to define an *empirical surrogate* of the viable state set

$$\hat{\mathcal{V}}_{eca} := \{ (q, \dot{q}) \mid S_v(p, q, \dot{q}) > 0 \},$$

For scenarios where  $S_v(p, q, \dot{q}) \in (0, \epsilon_{eca}]$ , with  $\epsilon_{eca} \in \mathbb{R}_{++}$  denoting a small positive scalar, we impose a constraint enforcing  $\Delta S_v(p, q, \dot{q}) \geq 0$ , thereby ensuring that the state remains within the viable state set. By taking the Taylor expansion of  $S_v(p, q, \dot{q})$  and omitting high-order terms, we have

$$\begin{aligned} \Delta S_v(p, q, \dot{q}) &= \nabla_q S_v^\top \cdot \Delta q + \nabla_{\dot{q}} S_v^\top \cdot \Delta \dot{q} + \nabla_p S_v^\top \cdot \Delta p \\ &= \nabla_q S_v^\top \cdot (\dot{q} \delta t + \frac{1}{2} \ddot{q} \delta t^2) + \nabla_{\dot{q}} S_v^\top \cdot (\ddot{q} \delta t) \\ &\quad + \nabla_p S_v^\top \cdot \Delta p \\ &= (\frac{1}{2} \nabla_q S_v^\top \cdot \delta t^2 + \nabla_{\dot{q}} S_v^\top \cdot \delta t) \ddot{q} \\ &\quad + \nabla_q S_v^\top \cdot (\dot{q} \delta t) + \nabla_p S_v^\top \cdot \Delta p. \end{aligned} \quad (10)$$

For simplicity, we define  $g_{ee} = \frac{1}{2} \nabla_q f^\top \cdot \delta t^2 + \nabla_{\dot{q}} f^\top \cdot \delta t$  and  $b_e = \nabla_q f^\top \cdot (\dot{q} \delta t) + \nabla_p f^\top \cdot \Delta p$ . The constraint is then expressed as  $g_{ee}^\top \ddot{q} + b_e \geq 0$ , which is also affine to  $\ddot{q}$ . Plugging this acceleration constraint into dynamics (4), we have the torque constraint as

$$g_{ee} M^{-1} \tau \geq g_{ee} M^{-1} (C \dot{q} + G) - b_e \quad (11)$$

### C. Joint Position and Velocity Limit Constraints

In this subsection, inspired by [13], we describe the formulation of joint position and velocity limits constraints using the definition of viability. We aim to find the bounds of torques to preserve the state  $(q, \dot{q})$  within  $\mathcal{V}_{int}$ . The joint acceleration bound is obtained by Algorithm 1 [13] for position inequalities,

$$\frac{1}{\delta t} (-\dot{q}_{\max} - \dot{q}) \leq \ddot{q} \leq \frac{1}{\delta t} (\dot{q}_{\max} - \dot{q}) \quad (12)$$

for velocity inequalities, and the intersection of

$$\begin{aligned} \dot{q} + \delta t \ddot{q} &\leq \sqrt{2 \ddot{q}_{\max} (q^+ - q - \delta t \dot{q} - 0.5 \delta t^2 \ddot{q})} \\ \dot{q} + \delta t \ddot{q} &\geq -\sqrt{2 \ddot{q}_{\max} (q + \dot{q} \delta t + 0.5 \ddot{q} \delta t^2) - q^-} \end{aligned} \quad (13)$$

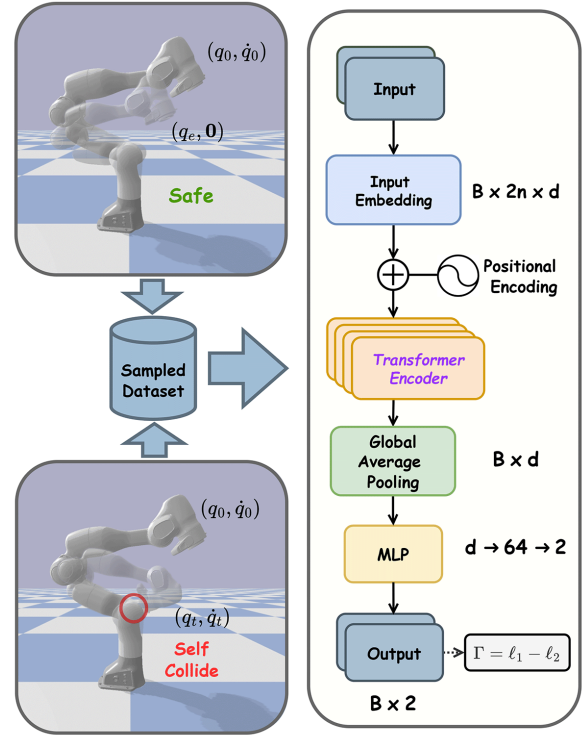


Fig. 5. Braking rollout from  $(q_0, \dot{q}_0)$  to rest  $(q_e, \mathbf{0})$ : label **Safe** iff every state along the trajectory to  $(q_e, \mathbf{0})$  is self-collision-free; otherwise **Self-Collide** at the first contact  $(q_t, \dot{q}_t)$ . The resulting dataset trains a network that maps  $(q, \dot{q})$  to a viability score  $\Gamma$ .

for viability inequalities. By taking the intersection of these three sets of inequalities with the hardware acceleration limits  $\ddot{q}_{max}$ , we obtain a maximal lower bound  $\ddot{q}_{lb}$  and a minimal upper bound  $\ddot{q}_{ub}$  such that

$$\ddot{q}_{lb} \leq \ddot{q} \leq \ddot{q}_{ub} \quad (14)$$

For further details, please refer to [13].

We can then obtain the torque constraints by plugging (14) into the dynamics, which yields

$$\ddot{q}_{lb} + M^{-1} (C \dot{q} + G) \leq M^{-1} \tau \leq \ddot{q}_{ub} + M^{-1} (C \dot{q} + G) \quad (15)$$

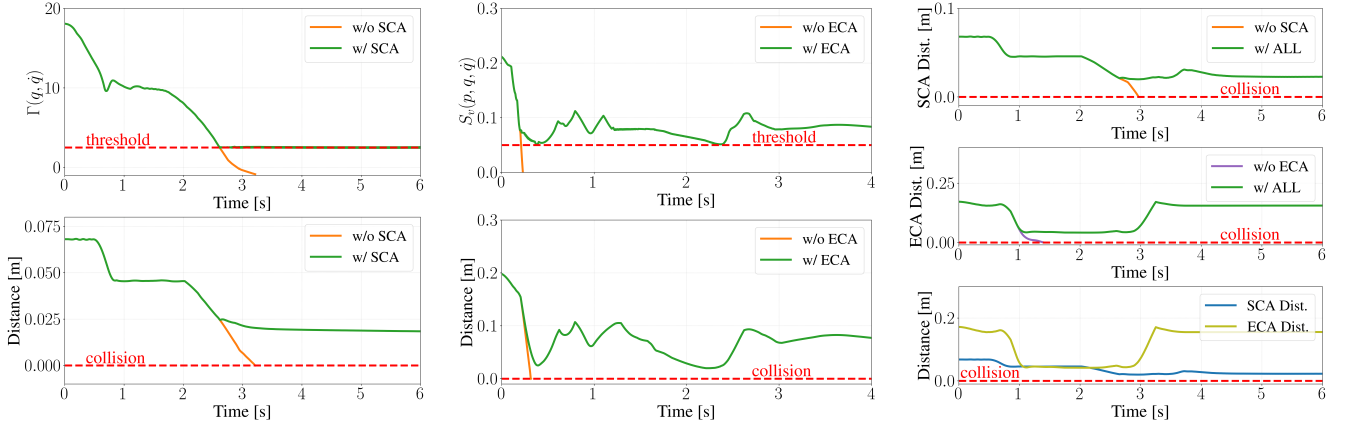
### D. Passive Torque Control with All Constraints

In our control framework, the joint position and velocity limit constraints are always active, the self-collision avoidance constraints and external obstacle collision avoidance constraints are active when the state is close to the boundary of the viable set. The optimization problem is formulated as:

$$\begin{aligned} \min_{\tau} \quad & \frac{1}{2} \|\hat{J}(q)^{-\top} \tau - F_c(x)\|_2^2 + \alpha_1 \|\tau\|_2^2 + \alpha_2 \|\delta\|_2^2 \\ \text{s.t.} \quad & \ddot{q}_{lb} + M^{-1} (C \dot{q} + G) \leq M^{-1} \tau \leq \ddot{q}_{ub} + M^{-1} (C \dot{q} + G) \\ & g_{se} M^{-1} \tau \geq g_{se} M^{-1} (C \dot{q} + G) - b_s \wedge (\Gamma \in (0, \epsilon_{sca}]) \\ & g_{ee} M^{-1} \tau \geq g_{ee} M^{-1} (C \dot{q} + G) - b_e - \delta \wedge (S_v \in (0, \epsilon_{eca}]) \\ & \tau \in [\tau_{\min}, \tau_{\max}] \\ & \delta \geq 0 \end{aligned} \quad (16)$$

We define  $\hat{J}(q)^\top$  as  $((J(q)J(q)^\top)^{-1} + \sigma^2 \mathbb{I}_n)J(q)$  to avoid singular configurations,  $\sigma$  denotes a small real number and  $\mathbb{I}_n$  denotes an identity matrix.  $\delta$  is a non-negative slack variable. Passivity analysis see in Appendix A.





(a) With the SCA, the learned score  $\Gamma(q, \dot{q})$  and the (b) With the ECA, the viability-preserving distance (c) With both SCA and ECA active, the controller min link-link distance remain above the threshold;  $S_v(p, q, \dot{q})$  and the clearance to the obstacle stay maintains positive distances to self-collision and without SCA the robot enters self-collision. positive; without ECA they collide. external obstacles throughout the trajectory.

Fig. 6. Simulation results of VPP-TC under different safety constraints.

## V. EXPERIMENTAL RESULTS

All experiments were conducted using a 7-DoF Franka Panda manipulator. For simulations, we employed PyBullet and solved the QP optimization problem using CVXPY [15], while CVXGEN [16] was utilized for real-robot experiments. Videos of both PyBullet simulations and hardware experiments are included in the supplementary material and available on our project website.

### A. Network Training

We train two learning models used by the controller: (i) a transformer-based classifier  $\Gamma(q, \dot{q})$  that predicts conservative self-collision viability, described in Sec. IV-A, and (ii) per-link SDFs represented with Bernstein polynomials [14] that enable fast and differentiable distance queries for external-collision reasoning, used in the computation of the viability-preserving distance  $S_v$  in (9).

1) *Self-collision network*: We generate 3 million labeled pairs  $(q, \dot{q})$  in simulation by rolling out the braking control until the system comes to a full stop. A sample is marked viable if the entire rollout is self-collision-free and non-viable otherwise. This dataset is used to train the transformer-based classifier with a standard Transformer-Encoder backbone [17]: with feed-forward size 128, 2 attention heads per layer, and 4 stacked transformer encoder layers, trained for 30 epochs. On the validation set, we select  $\gamma_{\text{thr}}$  via a threshold sweep over  $\Gamma$ , choosing the value that leads to better recall for the viable class. On the test set at this fixed threshold, the classifier attains 99.27% accuracy and 99.74% recall.

2) *External-collision network*: For each robot link  $\Omega_n$ , we represent the geometry by an offline Bernstein-polynomial SDF  $S_{\Omega_n}^b$  learned from its mesh model [14]. We define a cubic volume around the link; points inside the volume are normalized into  $[0, 1]$ , while points outside are projected onto the boundary. The SDF of a link is parameterized by a trivariate Bernstein basis with 24 basis functions per axis (total  $24^3$ ), and the coefficient vector is optimized by recursive least squares with ridge regularization, updating the

weights incrementally on mini-batches. During inference, all link-level SDFs are transformed to the base frame using the kinematic chain, and the whole-body SDF  $S(p, q)$  is computed as in Eq. (8). We evaluate external-collision distances along the entire braking trajectory. At each discretized step  $t \in [0, T_{\text{br}}]$ , we query the whole-body SDF  $S(p, q(t))$  to measure clearance and take the min across all time steps. The resulting *viability-preserving distance* is defined as Eq. (9).

### B. Simulation Experiments

We designed three simulation experiments on the 7-DoF Franka Panda in PyBullet to evaluate the effectiveness of the proposed viability-preserving passive torque controller under different safety constraints: 1) joint limits with self-collision avoidance, 2) joint limits with external collision avoidance, and 3) joint limits with both self-collision and external collision avoidance. The robot is initialized at joint position  $[0.669, -0.346, -0.742, -1.66, -0.367, 2.3, 1.99]^\top$  with all joint velocities set to zero. Additionally, we compared our method against the baseline *Constrained Passive Interaction Control (CPIC)* [9], highlighting improvements in computational efficiency, path length, and trajectory smoothness. All simulation experiments were performed on a workstation running Ubuntu 20.04, equipped with an AMD Ryzen 7 9800X3D CPU and an NVIDIA GeForce RTX 4060 GPU.

1) *Joint Position and Velocity Limits & Self-Collision Avoidance*: In this scenario, the desired task-space DS  $f(x)$  has a convergence point located within the manipulator's own body. Without enforcing the self-collision constraint, the robot would inevitably collide with itself.

Thus, we simply define the potential function as

$$V(x) = (x - x^*)^\top P(x - x^*), \quad (17)$$

where  $x^* = [0, 0, 0.3]^\top$  and  $P = -25 \mathbb{I}_n$ . The associated vector field is then given by

$$\dot{x} = f(x) = \nabla_x V(x) = 2P(x - x^*). \quad (18)$$

In this test, we set a threshold  $\gamma_{\text{thr}} = 2.5$ , which the manipulator is required to remain above. This threshold was

TABLE I  
COMPARISON OF CPIC AND VPP-TC ACROSS DIFFERENT CONSTRAINT SETTINGS (AVERAGE OVER 5 TRIALS).

	Control Loop Rate [Hz] $\uparrow$			Trajectory Length [m] $\downarrow$			Trajectory Jerkness $\downarrow$		
	SCA	ECA	ALL	SCA	ECA	ALL	SCA	ECA	ALL
CPIC	126.1 $\pm$ 1.4	72.6 $\pm$ 0.3	65.8 $\pm$ 0.2	1.12 $\pm$ 0.29	1.54 $\pm$ 0.23	1.59 $\pm$ 0.23	8.8 $\pm$ 2.3	1913.7 $\pm$ 726.0	2173.9 $\pm$ 788.0
VPP-TC	<b>171.6<math>\pm</math>9.1</b>	<b>153.0<math>\pm</math>4.9</b>	<b>155.4<math>\pm</math>10.3</b>	<b>0.75<math>\pm</math>0.18</b>	<b>1.23<math>\pm</math>0.19</b>	<b>1.19<math>\pm</math>0.21</b>	<b>1.2<math>\pm</math>0.4</b>	<b>216.2<math>\pm</math>262.8</b>	<b>186.7<math>\pm</math>116.8</b>

selected based on the recall performance of the transformer-based neural network classifier, ensuring a high likelihood of correctly identifying viable states. Fig. 6(a) shows that, without the SCA constraint, the manipulator enters self-collision configurations where link1 and link6 intersect. With the self-collision constraint enforced, all non-adjacent links maintain a minimum self-collision distance, implicitly guaranteed through  $\Gamma(q, \dot{q})$ . The simulation runs for 6 seconds.

2) *Joint Position and Velocity Limits & External Obstacle Collision Avoidance*: We designed a simple scenario in which the robot is tasked with reaching the convergence point  $x^* = [0, -0.6, 0.3]^\top$  while avoiding both a static and a dynamic obstacle along its path. The static obstacle is located at  $[0.4, -0.3, 0.4]^\top$ , whereas the dynamic obstacle follows a trajectory  $[0, -0.4, 0.5 + 0.1 \sin(2t_{\text{run}})]^\top$  as a function of runtime  $t_{\text{run}}$ . Both obstacles are modeled as spheres with a radius of 5 cm. The simulation runs for 4 seconds, during which the manipulator is required to maintain a clearance of at least 5 cm from the obstacle surfaces. As shown in Fig. 6(b), the incorporation of the external collision avoidance constraints ensures that collisions with the obstacles are prevented. The simulation runs for 4 seconds.

3) *ALL Constraints*: We designed a scenario where the robot would collide with a static spherical obstacle of radius 5 cm, located at  $[0, 0.3, 0.2]^\top$ , on its way to the target position  $[-0.1, 0, 0.3]^\top$ . In addition, self-collision configurations arise if the SCA constraint is not enforced, as shown in the top two subfigures of Fig. 6(c). By simultaneously applying joint-limit, self-collision, and external-collision constraints, our controller ensures safe and feasible motion throughout the trajectory, as illustrated in the bottom subfigure of Fig. 6(c). The simulation runs for 6 seconds.

4) *Comparison with CPIC*: We compare our method against the baseline CPIC controller with **three** sets of experiments. First, we conducted five experiments under the SCA setting, where the robot's initial joint configuration was randomly sampled and the target was fixed at  $[0, 0, 0.3]^\top$ . Similarly, five experiments were conducted under the ECA setting with one static obstacle; both the obstacle position and the robot's initial configuration were randomly sampled. It is worth noting that our VPP-TC enforces external-collision avoidance by considering the full geometry of all robot links, whereas in the version of CPIC available to us, only the last three links were taken into account. Finally, five experiments were performed under the ALL-constraints setting, with the robot initialized at  $[0, 0, 0, -1.5708, 0, 1.8675, 0]^\top$ , a static obstacle located randomly, and the target was fixed at  $[-0.25, -0.35, 0.5]^\top$ . Each ALL trial was run for  $t$  seconds. For all experiments, we use three quantitative

metrics to evaluate performance. (i) *Control Loop Rate*: is defined as the average number of control steps executed per second, with higher values indicating better performance. In simulation, it is computed as the total number of simulated control steps divided by the actual wall-clock runtime on the computer. (ii) *Path Length*: is the total distance traveled by the end-effector in simulation to accomplish the task. A shorter path length indicates a more efficient trajectory. (iii) *Trajectory Jerkness*: reflects the smoothness of the executed trajectory. Let  $x(t) \in \mathbb{R}^3$  be the end-effector trajectory over  $t \in [0, T]$ , with jerk  $j(t) = \ddot{x}(t)$  and path length  $L = \int_0^T \|\dot{x}(t)\| dt$ . The normalized jerk is then defined as

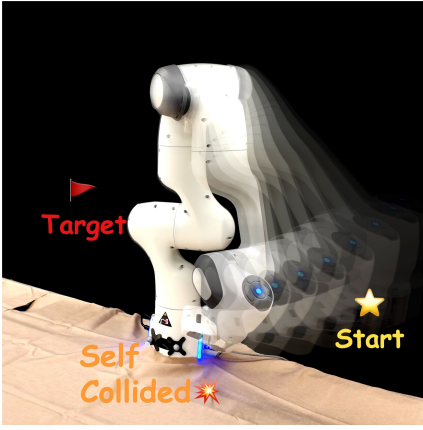
$$\text{NJ} = \frac{1}{L^2 T^5} \int_0^T \|\ddot{x}(t)\|^2 dt, \quad (19)$$

where smaller values indicate smoother trajectories. As summarized in Table I, VPP-TC consistently achieves the lowest computation time, shortest path length to the task position, and significantly reduced trajectory jerkness vs. CPIC.

### C. Real-Robot Experiments

In addition to simulation, we conducted three experimental scenarios on the 7-DoF Franka Panda to examine the controller's effectiveness under real-world conditions: 1) joint limits with self-collision avoidance, 2) joint limits with external collision avoidance, and 3) joint limits with both self-collision and external collision avoidance. Experiment 1) was performed on an Intel NUC 13 Pro running Ubuntu 20.04. Experiments 2) and 3) were conducted using both an Intel NUC 13 Pro and a workstation running Ubuntu 22.04, equipped with an Intel i7-11700K CPU and an NVIDIA RTX 3070 GPU. All real-robot experiments employed ROS Noetic for communication, where the workstation setup used *RoboStack* [18] to build the Noetic environment.

1) *Joint Position and Velocity Limits & Self-Collision Avoidance*: We performed three experiments in this setting. In the first two, the robot started from the joint configuration  $[0.483, 0.200, 0.712, -2.240, -0.190, 2.384, 1.325]^\top$  at rest and was commanded to reach the target position  $[0, -0.3, 0.4]^\top$ . Without the self-collision avoidance (SCA) constraint, the arm quickly entered Reflex Mode due to a self-collision warning and halted (Fig. 7(a)). With the SCA constraint enabled, the robot safely avoided self-collision and reached the target (Fig. 7(b)). In the third experiment, we attached a UMI gripper as the end-effector and retrained the neural network  $\Gamma_{\text{umi}}(q, \dot{q})$  to incorporate the new geometry, showing that the learned viability boundary generalizes across different robot geometries and continues to prevent unsafe states (Fig. 7(c)). Moreover, the fact that the arm cannot be pushed beyond the boundary highlights



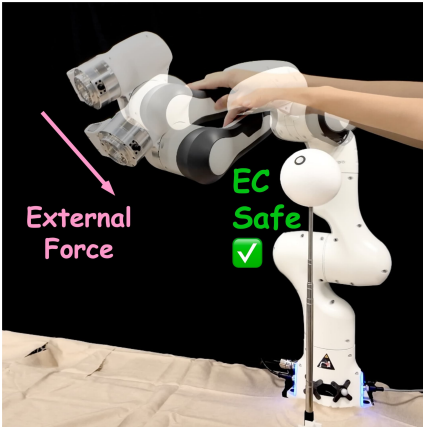
(a) Without SCA, the arm quickly triggers Reflex Mode due to self-collision and halts.



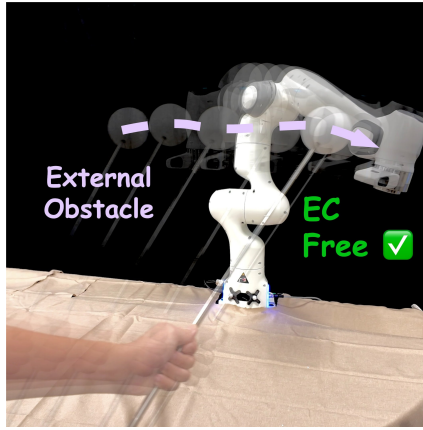
(b) With SCA enabled, the arm safely avoids self collision and reaches the target.



(c) With a UMI gripper, the arm cannot be pushed beyond the viability boundary.



(d) Static obstacle with external push: the arm avoids obstacle and stops with a positive clearance.



(e) Moving obstacle approaching: the arm yields and maintains a safe clearance from obstacle.



(f) Teleoperating near a lamp: positive clearance from obstacle preserved.

Fig. 7. Real-robot demonstrations of the proposed controller. Top row: self-collision avoidance (SCA). Bottom row: external-collision avoidance (ECA). Trajectory overlays show that activating the corresponding constraint prevents contact and maintains positive clearance while allowing the task to proceed.

that the controller enforces a normal constraint reaction and clamps the normal velocity to zero, while tangential damping dissipates motion. In this way, external input energy is either dissipated through damping or absorbed by the constraint reaction, and no net energy is injected into the environment, reflecting the task-space passivity of the controller.

2) *Joint Position and Velocity Limits & External Obstacle Avoidance*: The positions of external obstacles were provided by an OptiTrack motion capture system. We first tested the case of a static spherical obstacle placed next to the robot. When external forces were applied to push the arm toward the obstacle from different joint configurations, the arm resisted the motion and stopped while maintaining a safe clearance before contact (Fig. 7(d)). Similar to the self-collision scenario, as the arm approached the obstacle, it was unable to be pushed any further, indicating that the controller generated a reaction force along the obstacle normal while suppressing motion in that direction. At the same time, damping in the tangential directions dissipated the residual motion. Consequently, the external work was either absorbed by the constraint reaction or dissipated, and the system remained passive in task space without injecting net energy

into the environment. In a second experiment, we tested a moving spherical obstacle approaching the arm. The robot responded by actively moving away and maintained a safe clearance without collision (Fig. 7(e)). More dynamic and extreme scenarios with either multiple or high-speed moving obstacles are demonstrated in the accompanying video and on our project website. Unlike the previous CPIC approach, which struggles with multiple or dynamic obstacles due to computational bottlenecks, our VPP-TC handles such cases effectively. The high update rate ( $\sim 200$  Hz with two dynamic obstacles) enables true whole-body control with multiple obstacles, whereas CPIC in practice only considered the distance between the obstacle and the 7th link. Moreover, since the external-collision avoidance constraint (10) implicitly incorporates obstacle velocity through the  $\Delta p$  term, VPP-TC can also robustly handle fast-moving obstacles.

3) *ALL Constraints*: In this experiment, the robot was teleoperated by specifying only coarse target positions rather than precise collision-free trajectories, and tasked with pressing the switch of a desk lamp to turn it on (Fig. 7(f)). Along the way to the switch, the planned path intersected with the lamp body (external obstacle) and also brought the arm

close to the base link, creating a risk of self-collision. With VPP-TC, however, the robot avoided both external and self-collisions without any explicit motion planning, and safely reached the switch to accomplish the task (Fig. 2). This result highlights several key advantages of VPP-TC:

- It provides unified handling of joint limits, self-collision, and external collision constraints within the same framework.
- It enables safe execution of high-level teleoperation commands without requiring handcrafted obstacle-avoiding trajectories.
- It ensures robust task-space passivity, so that safety is guaranteed under coarse or uncertain human inputs.

These features underline the potential of VPP-TC as a versatile controller for safe human-in-the-loop manipulation.

## VI. CONCLUSIONS AND FUTURE WORKS

This paper proposed a novel viability-based control framework that enforces safety constraints for passive torque-controlled robots. We construct viable sets in the state space via a combination of data-driven learning and analytical derivation. A quadratic program then enforces the resulting torque bounds. We demonstrate the effectiveness and performance advantages of the proposed approach over a baseline through both simulation and hardware experiments. Our current control framework presumes an accurate dynamics model. In future work, we aim to accommodate bounded mismatches between the true and estimated dynamics and design a robust controller that guarantees viability in the presence of modeling errors.

## REFERENCES

- [1] A. van der Schaft, *Passive State Space Systems*. Cham: Springer International Publishing, 2017, pp. 59–99.
- [2] J. Colgate, “Robust control of dynamically interacting systems,” *International Journal of Control*, vol. 48, no. 1, pp. 65–88, Jul. 1988.
- [3] N. Hogan, “Impedance control: An approach to manipulation,” in *1984 American Control Conference*, 1984, pp. 304–313.
- [4] A. S. Huaman, O. B. Cieza, and J. Reger, “Passivity-based control for the cart-pole in implicit port-hamiltonian representation: An experimental validation,” in *2021 European Control Conference (ECC)*, 2021, pp. 2080–2085.
- [5] A. Dietrich, X. Wu, K. Bussmann, C. Ott, A. Albu-Schäffer, and S. Stramigioli, “Passive hierarchical impedance control via energy tanks,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 522–529, 2017.
- [6] A. Billard, S. S. Mirrazavi Salehian, and N. Figueroa, *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. Cambridge, USA: MIT Press, 2022.
- [7] N. Figueroa and A. Billard, “Locally active globally stable dynamical systems: Theory, learning, and experiments,” *The International Journal of Robotics Research*, vol. 41, no. 3, pp. 312–347, 2022.
- [8] K. Kronander and A. Billard, “Passive interaction control with dynamical systems,” *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 106–113, 2015.
- [9] Z. Zhang, T. Li, and N. Figueroa, “Constrained passive interaction control: Leveraging passivity and safety for robot manipulators,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 13 418–13 424.
- [10] 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)*, 2019.
- [11] Q. Nguyen and K. Sreenath, “Exponential control barrier functions for enforcing high relative-degree safety-critical constraints,” in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 322–328.

- [12] M. A. Bouguerra, T. Fraichard, and M. Fezari, “Viability-based guaranteed safe robot navigation,” *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, pp. 459–471, 2019.
- [13] A. D. Prete, “Joint position and velocity bounds in discrete-time acceleration/torque control of robot manipulators,” *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 281–288, 2018.
- [14] Y. Li, Y. Zhang, A. Razmjoo, and S. Calinon, “Learning robot geometry as distance fields: Applications to whole-body manipulation,” 2023.
- [15] S. Diamond and S. Boyd, “CVXPY: A Python-embedded modeling language for convex optimization,” *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.
- [16] J. Mattingley and S. Boyd, “CVXGEN: A code generator for embedded convex optimization,” *Optimization and Engineering*, vol. 12, no. 1, pp. 1–27, 2012.
- [17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017.
- [18] T. Fischer, W. Vollprecht, S. Traversaro, S. Yen, C. Herrero, and M. Milford, “A robotstack tutorial: Using the robot operating system alongside the conda and jupyter data science ecosystems,” *IEEE Robotics and Automation Magazine*, 2021.

## APPENDIX

### A. VPP-TC passivity analysis

We analyze the passivity of the controller generated by the simplest version of (16) with just one constraint active,

$$\min_{\tau_c} \frac{1}{2} \|J(q)^{-\top} \tau_c - F_c(x)\|_2^2, \text{ s.t. } a^\top \tau_c \leq b, \quad (20)$$

where  $a \in \mathbb{R}^n$  and  $b \in \mathbb{R}$  are the parameters of the active affine constraint. We have the closed-form solution of the above QP as:

$$\tau_c^* = J(q)^\top F_c - \frac{a^\top J(q)^\top F_c - b}{a^\top J(q)^\top J(q)a} J(q)^\top J(q)a. \quad (21)$$

We analyze the passivity in task-space, the equivalent task space force is expressed as:

$$\begin{aligned} F_c^* &= J(q)^{-\top} \tau_c \\ &= F_c - J(q)^{-\top} \frac{a^\top J(q)^\top F_c - b}{a^\top J(q)^\top J(q)a} J(q)^\top J(q)a \\ &= F_c - \frac{a^\top J(q)^\top F_c - b}{a^\top J(q)^\top J(q)a} J(q)a \\ &= (I - \frac{J(q)a(J(q)a)^\top}{(J(q)a)^\top J(q)a}) F_c - \frac{bJ(q)a}{a^\top J(q)^\top J(q)a} \\ &= \bar{A}F_c - \bar{B}, \end{aligned} \quad (22)$$

where  $\bar{A} = I - \frac{J(q)a(J(q)a)^\top}{(J(q)a)^\top J(q)a}$ ,  $\bar{B} = \frac{bJ(q)a}{a^\top J(q)^\top J(q)a}$ . We define the storage function as  $S = \frac{1}{2} \dot{x}^\top M \dot{x} + \lambda_1 P(x)$ , where  $P(x)$  is the potential energy. We can make the following derivation:

$$\begin{aligned} \dot{S} - \dot{x}^\top f_{ext} &= \dot{x}^\top M \ddot{x} + \frac{1}{2} \dot{x}^\top \dot{M} \dot{x} + \lambda_1 \dot{x}^\top \nabla_x P \\ &= \dot{x}(-G(x) + F_c^*) + \lambda_1 \dot{x}^\top \nabla_x P \\ &= \dot{x}(\bar{A} - I)G - \dot{x}\bar{A}Dx - \dot{x}\bar{B} \\ &\quad + \lambda_1 \dot{x}^\top (\nabla_x P + \bar{A}f(x)). \end{aligned} \quad (23)$$

We can conclude that for task-space state  $(x, \dot{x})$  satisfies  $\dot{x}(\bar{A} - I)G - \dot{x}\bar{A}Dx - \dot{x}\bar{B} + \lambda_1 \dot{x}^\top (\nabla_x P + \bar{A}f(x)) \leq 0$ , the robot is passive. Otherwise, passivity is lost and the robot becomes stiff and enforce the constraints.