SPARK: A Toolbox for Safe Humanoid Autonomy and Teleoperation

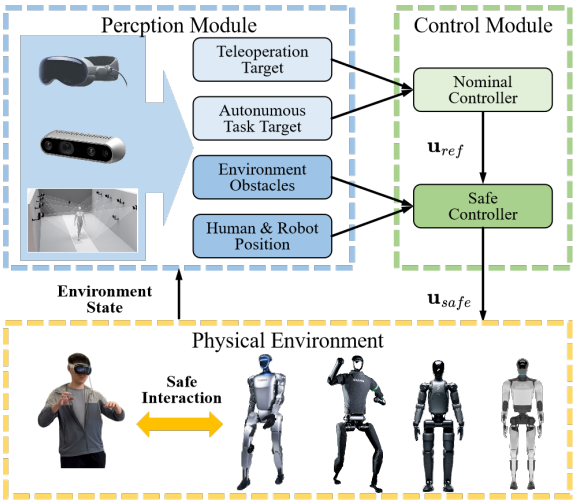
Yifan Sun, Rui Chen, Kai S. Yun, Yikuan Fang, Sebin Jung, Weiye Zhao, and Changliu Liu
Robotics Institute, Carnegie Mellon University
{yifansu2,ruic3,sirkhooy,yikuanf,sebinj,weiyezha,cliu6}@andrew.cmu.edu

***Abstract*—This paper presents the Safe Protective and Assistive Robot Kit (SPARK), a comprehensive toolbox designed to ensure safety in humanoid autonomy and teleoperation. Humanoid robots have gained significant popularity due to their versatility in interacting with complex environments. It is critical to ensure that no harm is caused during those interactions since unsafe behaviors can pose serious risks to the environment and human safety, considering humanoids' physical capabilities. Therefore, robust safety measures are essential in general humanoid robotics research and deployment. The inherently complex physical structures of humanoid robots further complicate the design of task-specific safety solutions. To alleviate this challenge, we introduce SPARK, a modular toolbox that integrates state-of-the-art safe control algorithms into a generic robot control framework. SPARK enables users to configure safety behaviors across multiple dimensions, such as defining safety criteria and sensitivity levels, allowing for optimized trade-offs between safety and performance. The toolbox is natively compatible with the Unitree G1 humanoid robot, utilizing Apple Vision Pro (AVP) for perception, while also offering interfaces for seamless customization with alternative hardware setups. By integrating SPARK as a fail-safe mechanism, users can significantly improve the safety of their existing humanoid systems.**

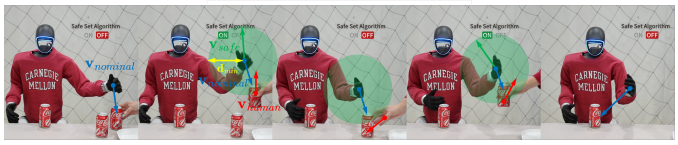
I. Introduction

Humanoid robots, with their human-like form and capabilities, hold the promise of seamless integration into environments designed for humans, performing tasks ranging from industrial automation to personal assistance. To deploy those robots into daily life, ensuring their safe operation, in the sense of guaranteeing humanoids to interact with their environments and humans without causing harm, is critical. This necessitates *provable safety* - mathematical assurances of safe behavior - that allow humanoids to adjust in real-time to dynamic scenarios. Ensuring safe operation for humanoid robots is particularly challenging because they possess both mobility and complex limb articulation, leading to a high-dimensional control problem. Managing safety at the limb level is essential to prevent the implementation of overly conservative constraints that could limit the robot's capabilities. The dynamic and unpredictable nature of real-world environments necessitates a flexible approach to safety.

To this end, we introduce SPARK (Safe Protective and Assistive Robot Kit), a Python toolbox designed to provide robust safety guarantees for humanoid robots during autonomy and teleoperation. SPARK implements a suite of safe controllers and planners, centered around the *Safe Set Algorithm (SSA)* [3]. SSA modifies potentially unsafe control inputs to ensure compliance with safety constraints without hindering performance. As a toolbox, SPARK offers versatility in hardware compatibility and allows users to configure tasks and safety constraints with Python APIs. This adaptability allows users to quickly integrate SPARK with their existing pipelines to safeguard the development process. For example, developers can incorporate SPARK into their stacks to ensure safety guarantees during autonomous operations. In teleoperation tasks, SPARK overrides inappropriate operator commands, preventing collisions with obstacles or humans even in highly dynamic scenarios. SPARK also allows deeper customizations if the default configurations do not suffice their need. For example, researchers can pass their custom system models and controllers to SPARK and deploy experimental code directly on hardware with the confidence that any unsafe actions will be filtered out.



(a) Framework of SPARK system.



(b) Safe teleoperation with SPARK.

Figure 1: SPARK system framework and example usage of SPARK in a teleoperation task. In (b), the teleoperated humanoid robot tries to approach the Coke cans but gets interrupted by the human who aims at the same targets. Sensing potential collision, SPARK overrides the teleoperation command (blue) and generates safe actions (green). Teleoperation commands pass to the robot when no safety risks are present (left and right most frames).

SPARK is implemented for Unitree's G1 humanoid platform and Apple Vision Pro (AVP) for perception and can be quickly customized for alternative humanoid and perception systems. Key features of SPARK include:

- **Safety constraint satisfaction**: safe controller for satisfying user-defined safety criteria.
- **Optimized safety-performance trade-off**: configurable trade-off between task-specific goals and safety.
- **Modularized design**: convenient plug-and-play without the need to alter existing pipelines.
- **Compatibility**: generic framework compatible with the general class of humanoid robots and perception systems.
- **Accessible Python API**: support for both out-of-box deployment and deep customizations.

II. Safe Humanoid Autonomy and Teleoperation

In this section, we describe the safe humanoid autonomy and teleoperation framework, and formulate the core safe control problem to be solved by SPARK.

A. Safe Humanoid Autonomy and Teleoperation Framework

As shown in Figure 1a, the hardware of the proposed framework primarily consists of three components: a human user, a humanoid robot, and perception devices. The human user interacts with the robot, either serving as the teleoperator or being modeled as environmental obstacles. The humanoid robot executes teleoperation commands while avoiding collisions with obstacles in its environment. Various perception methods can be employed, including the Apple Vision Pro, RGB cameras, and Motion Capture Systems, all of which can be integrated into the perception module.

The software framework for safe humanoid autonomy and teleoperation is divided into two main modules:

1) Perception Module: The Perception Module processes the state of the hardware system and converts it into observations required by the controller. By analyzing the relative position of the human user within the robot's reference frame, it forwards the teleoperation target position to the controller. Additionally, it detects and relays the positions of obstacles in the environment for collision avoidance.

2) Controller Module: The Controller Module is the core of the SPARK system. It receives teleoperation targets from the Perception Module and uses a nominal controller to compute reference controls for the robot joints via inverse kinematics. The safe controller then refines these reference controls, integrating information about environmental obstacles to generate safe control commands for the robot.

B. Safe Control Problem

The safe controller of SPARK is aiming at achieving both efficient and safe interation with the environment by solving the following safe control problem:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \|\mathbf{u} - \mathbf{u}_{\text{ref}}\|^2_{\mathbf{Q}_\mathbf{u}} \\ \mathbf{s.t.} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \\ & \mathbf{x} \in \mathbf{X}_\text{s} \end{aligned} \tag{1}$$

Where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{N_\mathbf{x}}$ represents the system state, and $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^{N_\mathbf{u}}$ is the control variable corresponding to the $N_\mathbf{u}$ degrees of freedom. $f(\mathbf{x}, \mathbf{u})$ denotes the system dynamics, and $\mathbf{X}_\text{s}$ is the set of safe system states. At each timestep, the objective of the control problem is to track the reference control $\mathbf{u}_{\text{ref}}$ while satisfying both the system dynamic constraints and the safety constraints.

The challenges of safe control for humanoid robots primarily arise from two aspects:

1) Complex Dynamics: A humanoid robot combines the features of both a mobile robot and a dual-arm manipulator system, resulting in a high-dimensional and nonlinear dynamic system. The coupling between the inherent uncertainty of legged robot locomotion and the high dimensionality of the dual-arm manipulator further increases the complexity of the robot's dynamics. Additionally, the system's complexity is amplified by the partial dependencies among the degrees of freedom (DOFs). For example, the DOFs of one arm do not directly affect the state of the other arm. However, DOFs that influence the robot's localization directly impact the position of the floating base, which is shared by both arms. These interdependencies make it particularly challenging for the robot to precisely track specific targets and perform safe motions in Cartesian space.

2) Dexterous Safety: Beyond the dynamic constraints, safety constraints add another layer of complexity to the safe control problem. A humanoid robot must simultaneously satisfy multiple safety constraints to safely operate in the real world without being conservative. For instance, humanoids may operate in confined spaces where different body parts interact closely with various environment objects, requiring precise pose adjustments to avoid collisions with obstacles. Hence, humanoids must be modeled at a granular limb level rather than a coarse whole-body level, and all limb-obstacle combinations must be considered simultaneously. In addition, humanoids must avoid self-collision, requiring a group of constraints that scales combinatorially with respect to the granularity of modeling. Those diverse constraints combined render the safe state space $\mathbf{X}_\text{s}$ highly non-convex, making it hard to ensure safety in real-time.

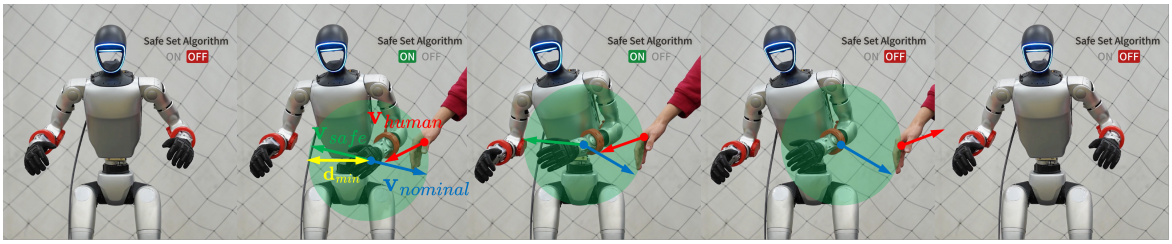
As a core contribution, SPARK implements generic safe controllers compatible with various hardware systems to accelerate the safe deployment of humanoids in real life. In the next section, we describe the safety backbone of SPARK, which algorithmically addresses humanoid safe control.

III. Safe Control via Safe Set Algorithm

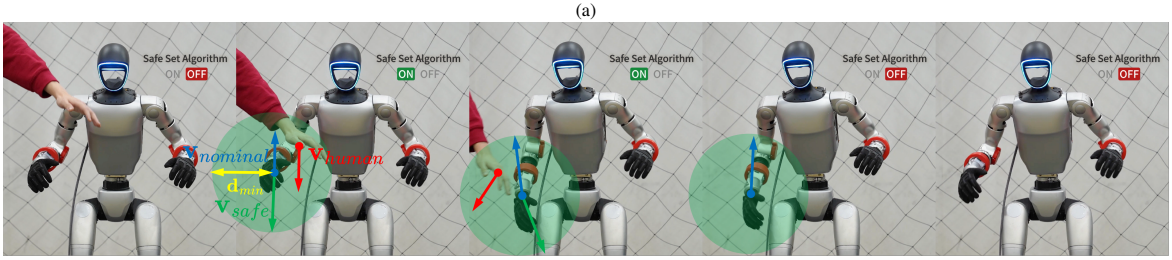
In this section, we first review *Safe Set Algorithm (SSA)* [3], the safety backbone of SPARK. Then, we explain how SSA is leveraged to address optimal control problems with multiple safety constraints, which naturally arise when deploying humanoids in the real world.

A. Safe Control Objectives

SSA assumes that the user-specified set of safe system states $\mathbf{X}_\text{S}$ is the zero sublevel set of some piecewise smooth energy



(a)



(b)

Figure 2: Limb-level collision avoidance with static humanoid reference pose.

function $\phi_0 := \mathcal{X} \mapsto \mathbb{R}$. Namely, $\mathbf{X}_\mathrm{S} := \{\mathbf{x} \in \mathcal{X} \mid \phi_0(\mathbf{x}) \leq 0\}$. Users can define $\mathbf{X}_\mathrm{S}$ and ϕ_0 to fit specific scenarios. For instance, to keep a minimal distance d_min from some obstacle, ϕ_0 can be $\phi_0 = d_\mathrm{min} - d$ where d is the distance to the obstacle. SSA provably ensures safety by establishing a subset $\mathbf{X}_\mathrm{safe}$ of $\mathbf{X}_\mathrm{S}$ such that $\mathbf{x}$ never leaves $\mathbf{X}_\mathrm{safe}$ as long as $\mathbf{x}$ has been in that set. $\mathbf{X}_\mathrm{safe}$ is also referred to as the *safe set*. Formally, if $\mathbf{x}(t_0) \in \mathbf{X}_\mathrm{safe}$, then $\mathbf{x}(t) \in \mathbf{X}_\mathrm{safe}$ for $t \geq t_0$. We refer to such property as *forward invariance (FI)*. Secondly, if the state $\mathbf{x}$ starts outside $\mathbf{X}_\mathrm{safe}$, it should enter $\mathbf{X}_\mathrm{safe}$ in finite time. Formally, for any t_0 such that $\mathbf{x}(t_0) \notin \mathbf{X}_\mathrm{safe}$, there exists $t \in [t_0, \infty)$ such that $\mathbf{x}(t) \in \mathbf{X}_\mathrm{safe}$. We refer to such a property as *finite-time convergence (FTC)*.

B. Safe Set Algorithm

To keep $\mathbf{x}$ within the safe set, SSA enforces a Lyapunov-like condition on the energy function ϕ_0 and constraints its rate of change. In cases of high relative degree, $\mathbf{u}$ may not appear in $\dot{\phi}_0$, preventing one from directly controlling ϕ_0. For example, $\dot{\phi}_0 = -\dot{d}$ depends only on velocities and cannot be affected by the acceleration input of a second-order system. Hence, we need an alternative energy function ϕ tailored to the system dynamics. For instance, for an acceleration-driven system (e.g., second-order), we can employ a velocity-based energy function ϕ (e.g., first-order), such that $\dot{\phi}$ depends on the input and can be used to construct safety constraints on $\mathbf{u}$. The safe set algorithm (SSA) [3] introduces a systematic approach to the design of such ϕ, which is adopted by `SPARK`. SSA introduces a continuous, piecewise smooth energy function $\phi := \mathcal{X} \mapsto \mathbb{R}$, or *safety index*, to quantify safety while considering the system dynamics. An n^th $(n \geq 0)$ order safety index ϕ_n has the following general form:

$$\phi_n = (1 + a_1 s)(1 + a_2 s) \dots (1 + a_n s)\phi_0, \tag{2}$$

where s is the differentiation operator. (2) can also be expanded as

$$\phi_n := \phi_0 + \textstyle\sum_{i=1}^{n} k_i \phi_0^{(i)}. \tag{3}$$

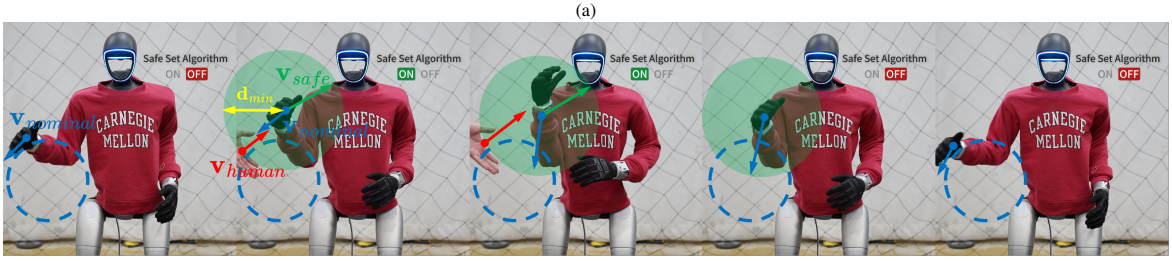
where $\phi_0^{(i)}$ is the i^th time derivative of ϕ_0. ϕ_n should satisfy that (a) the characteristic equation $\prod_{i=1}^{n}(1 + a_i s) = 0$ only have negative real roots to prevent overshooting of ϕ_0 and (b) $\phi_0^{(n)}$ has relative degree one to the control input $\mathbf{u}$. For instance, if $\mathbf{X}_\mathrm{S}$ is defined by $\phi_0 = d_\mathrm{min} - d$, we can use $\phi_1 = d_\mathrm{min} - d - k\dot{d}$ for collision-avoidance for an acceleration-driven system. It can be shown that if $\dot{\phi}_n(\mathbf{x}, \mathbf{u}) \leq -\eta$ when $\phi_n(\mathbf{x}) \geq 0$ for some constant $\eta > 0$, both forward invariance within a safe set $\mathbf{X}_\mathrm{safe}$ and finite-time convergence to that set are guaranteed [1]. Hence, the safe control law c_{ϕ_n} of SSA can be written as the following optimization:

$$\min_{\mathbf{u} \in \mathcal{U}} \mathcal{J}(\mathbf{u}) \;\; \text{s.t.} \;\; \dot{\phi}_n(\mathbf{x}, \mathbf{u}) \leq -\eta \text{ if } \phi_n(\mathbf{x}) \geq 0 \tag{4}$$

where the objective $\mathcal{J}$ is arbitrary. For instance, one can assign $\mathcal{J}(\mathbf{u}) = \|\mathbf{u}\|$ if minimal effort is desired when safety is the sole objective.

SSA has been applied to various robot arm platforms [5, 8] for safe human-robot interaction. The advantage of SSA over other safe control methods (e.g., control barrier functions) in interactive environments are discussed in [11]. The safety index can be synthesized or adapted using rule-based approaches [13], evoluntionary optimization [12], adversarial optimization [7], reinforcement learning [9], or sum-of-squares programing [1, 2, 14]. There are various extensions of SSA to handle uncertainties of human behaviors [4] and efficiently learning human models during interactions [10], which can be modules directly added to our toolbox. A comprehensive discussion for achieving safety and efficiency during human-robot interactions can be found in the book [6].

(a)



(b)

Figure 3: Limb-level collision avoidance with dynamic humanoid reference poses.

C. SSA for Humanoids

As mentioned in section II-B, humanoid safety poses optimal control problems with multiple safety constraints due to limb-level modeling of the robot. To handle that problem, SPARK extends (4) to multi-constraint cases by allowing the set of safe system states $\mathbf{X}_\mathrm{S}$ to be described by $M \geq 1$ energy functions ϕ_0. Formally, we write $\mathbf{X}_\mathrm{S} := \{\mathbf{x} \in \mathcal{X} \mid \phi_0[i](\mathbf{x}) \leq 0,\ \forall i \in [M]\}$. Taking system dynamics into account, we can similarly acquire M safety indices $\{\phi_n[i]\}_{i\in[M]}$. Based on those, SPARK places one safe control constraint in the form of (4) for each $i \in [M]$. Replacing the original safety constraint $\mathbf{x} \in \mathbf{X}_\mathrm{S}$ in (1), we write the generalized safe control problem solved by SPARK as

$$\begin{aligned} \min_{\mathbf{u}} \quad & \|\mathbf{u} - \mathbf{u}_\mathrm{ref}\|^2_{\mathbf{Q}_\mathbf{u}} \\ \mathbf{s.t.} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \\ & \forall i \in [M],\ \dot{\phi}_n[i](\mathbf{x}, \mathbf{u}) \leq -\eta \text{ if } \phi_n[i](\mathbf{x}) \geq 0 \end{aligned} \tag{5}$$

It can be shown that under the control constraints in (5), a safe set $\mathbf{X}_\mathrm{safe} \subseteq \mathbf{X}_\mathrm{S}$ can still be found in which both FI and FTC properties are satisfied to provide theoretical safety guarantees. The detailed proof is omitted here due to the scope of this paper but can be easily derived following [1].

IV. Experiments and Case Studies

In this section, we present several examples of SPARK deployment on a physical humanoid robot. The configuration of both the nominal control objectives and the safety constraints are described for each example.

A. Experimental Setup

For our experiments, we utilized a Unitree G1 humanoid robot, which features a total of 29 degrees of freedom (DOFs). The perception module employed an Apple Vision Pro to capture the human user's gestures and determine the position of the humanoid robot. The robot's dynamic system was modeled as a mobile dual-manipulator system.

For the locomotion task, three DOFs were considered: x velocity, y velocity, and yaw angular velocity, which describes the rotational velocity around the z axis. Regarding the upper-body dynamics, each arm was treated as a general manipulator with seven DOFs, while the waist was equipped with three rotational joints. This results in a total of 20 DOFs for the system used in our experiment.

To evaluate the safety and performance of SPARK, we designed three distinct test cases. In each case, the humanoid robot was tasked with avoiding collisions with dynamic obstacles while tracking various nominal targets with its hands. Here the human user is treated as the obstacle which needs to be avoided by the robot. In order to model the collision volumes, we wrap each joint of the robot arms and the human user's hands with a sphere. The following sections provide a detailed description of each test case and an analysis of the robot's performance.

B. Limb-level Collision Avoidance

To evaluate the safety performance of SPARK for limb-level motion, we begin with the simplest task, in which the robot remains in a fixed position while avoiding potential collisions with the human user. The nominal control is defined as:

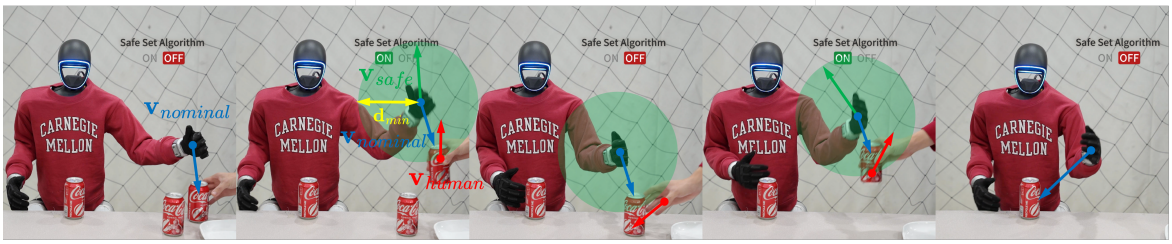
$$\mathbf{u}^r_\mathrm{ref} = K_p(\mathbf{x}^R_\mathrm{target} - \mathbf{x}^R), \tag{6}$$

where $\mathbf{x}^R$ represents the current joint state of the robot, and $\mathbf{x}^R_\mathrm{target}$ is the constant target position for each degree of freedom. The proportional gain K_p is used to track the target position via the nominal control $\mathbf{u}^r_\mathrm{ref}$.

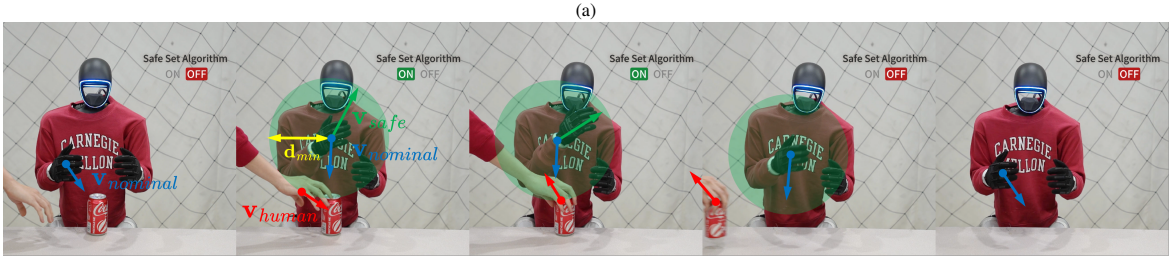
The safety index for this task is defined as:

$$\phi_n[i](\mathbf{x}^R) = d_\mathrm{min}[i] - D[i](\mathbf{x}^R, \mathbf{x}^H) \tag{7}$$

where $\phi_n[i]$ is the safety index for the ith collision volume pair between the human and the robot. $d_\mathrm{min}[i]$ is the minimum



(a)



(b)

Figure 4: Limb-level collision avoidance with teleoperation commands.

required distance to be maintained between the ith collision volume pair. The function $D[i](\mathbf{x}^R, \mathbf{x}^H)$ calculates the minimum distance between the ith collision volume pair given the state of the robot, $\mathbf{x}^R$, and the human user, $\mathbf{x}^H$.

If the minimum distance between the ith collision volume pair becomes smaller than $d_{\min}[i]$, the safety index $\phi_n[i](\mathbf{x}^R)$ will turn positive, activating the safe controller to prevent collisions.

As shown in Figure 2, when the user attempts to approach the robot arm from various angles, the robot reacts by moving away from the human's hand if the minimum distance between them becomes smaller than $d_{\min}$. Once the robot detects that the human hand has moved away and the surrounding environment is safe, it resumes following the nominal controller's commands and returns to the target position.

To evaluate the performance of the SPARK safe controller in tracking a dynamic target position, we designed a dynamic limb-level collision avoidance test case. Unlike the static test, where the nominal controller tracks a fixed target, the nominal controller in this test is tasked with following a dynamic target $\mathbf{x}^R_{\text{target}}$. Specifically, we defined a circular trajectory for the right hand of the humanoid robot as the dynamic target trajectory. In this scenario, the robot must track the target trajectory while simultaneously avoiding collisions with the human user with the same safety index ϕ as defined before.

From Figure 3, we can observe that when the human hand remains outside the $d_{\min}$ region, the robot follows the reference trajectory and moves along the circular path. However, if the human hand approaches too closely to the robot's hands, the humanoid robot utilizes both its waist and arm degrees of freedom (DOFs) to avoid potential collisions, regardless of changes in the target position. Once the human hand moves away, the robot returns to the reference trajectory while remaining prepared for future collision avoidance.

C. Safe Teleoperation

To evaluate the capability of the safe controller in more general scenarios, we removed the predefined dynamic target and tasked the robot with following another human user's teleoperation commands while maintaining collision avoidance. This setup introduces the concept of "Safe Teleoperation." In this test, the target of the humanoid nominal controller, $\mathbf{x}^R_{\text{target}}$, is not predesigned but instead generated in real-time by a human teleoperator. This significantly increases the challenge for the robot to generate safe motions in an unpredictable and dynamic environment.

In our experiment, we created a realistic scenario where the robot attempts to retrieve objects from a table. During the task, if the human user reaches for the same object as the robot, the safe controller is triggered. The robot prioritizes collision avoidance over executing the teleoperation commands, ensuring safe interaction and protecting both the humanoid robot and the human from potential hazards caused by the limited perception of a remote teleoperator.

V. Discussion and Conclusion

In this paper, we introduce SPARK, a toolbox for safeguarding the development of humanoid autonomy and teleoperation. We introduce the safe humanoid control framework as well as the core safe control algorithms SPARK was built upon. SPARK provides adjustable trade-offs between safety and performance, fitting various user requirements. With a modular design and accessible APIs, SPARK provides compatibility with a wide range of tasks with different hardware systems and levels of customizations. With SPARK, humanoid research and deployment can be significantly accelerated with enhanced hardware and environment safety. In future work, we plan to keep monitoring the development of humanoid hardware and research and updating SPARK with support for emerging safe control strategies and hardware systems.

VI. Acknowledgement

This work is supported by the National Science Foundation under grant No. 2144489.

References

[1] Rui Chen, Weiye Zhao, and Changliu Liu. Safety index synthesis with state-dependent control space. In *American Controls Conference*, 2024.

[2] Rui Chen, Weiye Zhao, Ruixuan Liu, Weiyang Zhang, and Changliu Liu. Real-time safety index adaptation for parameter-varying systems via determinant gradient ascend. In *American Control Conference*, 2024.

[3] Changliu Liu and Masayoshi Tomizuka. Control in a safe set: Addressing safety in human-robot interactions. In *2014 ASME Dynamic Systems and Control Conference*. ASME, 2014.

[4] Changliu Liu and Masayoshi Tomizuka. Safe exploration: Addressing various uncertainty levels in human robot interactions. In *2015 American Control Conference (ACC)*, pages 465–470. IEEE, 2015.

[5] Changliu Liu and Masayoshi Tomizuka. Algorithmic safety measures for intelligent industrial co-robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3095–3102. IEEE, 2016.

[6] Changliu Liu, Te Tang, Hsien-Chung Lin, and Masayoshi Tomizuka. *Designing robot behavior in human-robot interactions*. CRC Press, 2019.

[7] Simin Liu, John Dolan, and Changliu Liu. Safe control under input saturation with neural control barrier functions. In *Conference on Robot Learning*, 2022.

[8] Simin Liu, John Dolan, and Changliu Liu. Safe control under input saturation with neural control barrier functions. In *6th Annual Conference on Robot Learning*, 2022.

[9] Haitong Ma, Changliu Liu, Shengbo Eben Li, Sifa Zheng, and Jianyu Chen. Joint synthesis of safety certificate and safe control policy using constrained reinforcement learning. In *Learning for Dynamics and Control Conference*, pages 97–109. PMLR, 2022.

[10] Ravi Pandya and Changliu Liu. Safe and efficient exploration of human models during human-robot interaction. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6708–6715. IEEE, 2022.

[11] Tianhao Wei and Changliu Liu. Safe control algorithms using energy functions: A unified framework, benchmark, and new directions. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 238–243, 2019.

[12] Tianhao Wei and Changliu Liu. Safe control with neural network dynamic models. In *Learning for Dynamics and Control Conference*, pages 739–750. PMLR, 2022.

[13] Weiye Zhao, Tairan He, and Changliu Liu. Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

[14] Weiye Zhao, Tairan He, Tianhao Wei, Simin Liu, and Changliu Liu. Safety index synthesis via sum-of-squares programming. In *2023 American Control Conference (ACC)*, pages 732–737. IEEE, 2023.