A Shared Autonomy Framework for Assisted Teleoperation Using Control Barrier Functions

Master thesis by Yuanzheng Sun
Date of submission: December 2, 2025

1. Review: Prof. Dr.-Ing. Rolf Findeisen
2. Review: Prof. Jan Peters, Ph.D.
3. Review: Berk Gueler
4. Review: Kay Pompetzki
5. Review: Sebastian Hirt

Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Electrical Engineering and Information Technology
Computer Science

Control and Cyber-Physical Systems
Intelligent Autonomous Systems Group

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 APB TU Darmstadt

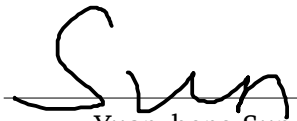
Hiermit erkläre ich, Yuanzheng Sun, dass ich die vorliegende Arbeit gemäß § 22 Abs. 7 APB der TU Darmstadt selbstständig, ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Ich habe mit Ausnahme der zitierten Literatur und anderer in der Arbeit genannter Quellen keine fremden Hilfsmittel benutzt. Die von mir bei der Anfertigung dieser wissenschaftlichen Arbeit wörtlich oder inhaltlich benutzte Literatur und alle anderen Quellen habe ich im Text deutlich gekennzeichnet und gesondert aufgeführt. Dies gilt auch für Quellen oder Hilfsmittel aus dem Internet.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 2. Dezember 2025



Yuanzheng Sun

Abstract

Teleoperation allows a human operator to control a robot arm remotely and is important in situations where direct interaction is not possible or not safe. However, teleoperation becomes difficult in cluttered environments. Limited perception, communication delays, and the mismatch between the robot's many degrees of freedom and the low-dimensional user input make the task demanding, and operators often struggle to move the robot accurately. Several shared-autonomy methods have been developed to reduce this workload by combining human input with autonomous assistance. While these approaches can improve task performance, most of them do not guarantee safety and may still lead to collisions in narrow or highly constrained spaces.

This thesis proposes a shared-autonomy controller that combines a Control Lyapunov Function (CLF) with Control Barrier Functions (CBFs) in a single optimization framework. The method fuses user input, an autonomously generated helper target, and formal safety constraints into one quadratic program. A state-dependent arbitration mechanism adjusts how strongly the CLF influences the motion: the robot provides more assistance when precise positioning is needed, while the user retains full control during larger, unconstrained motions.

We evaluate the controller in three simulated teleoperation environments and compare it with BarrierIK and two shared-autonomy baselines. In both predefined-trajectory experiments and a user study, the proposed method maintains positive clearance, avoids collisions, and produces smoother and more reliable motion than the other approaches. These results show that combining shared autonomy with formal safety tools can make teleoperation safer and easier to use in cluttered environments.

Contents

List of Figures

1 Introduction

1.1 Motivation

Teleoperation enables a human operator to control a robot arm from a distance and remains essential in environments where direct manual operation is infeasible or unsafe, such as nuclear waste handling, micro-scale manipulation, and underwater or space missions [1, 2].

Despite its advantages, teleoperation is fundamentally limited by restricted field of view, insufficient perceptual feedback, and communication delays [3, 4]. These factors increase the cognitive demands placed on the operator, who must interpret incomplete sensory information, coordinate multiple degrees of freedom, and react to uncertainty in real time. Prior work in human–robot interaction has shown that such elevated workload not only degrades task efficiency but also increases the likelihood of operator error [5, 6]. Studies on motion-mapping interfaces report that teleoperators experience physical and cognitive fatigue during long or precise tasks, which reduces accuracy and increases error rates even when the task appears simple [7]. Consequently, many teleoperation scenarios exceed what a human can reliably accomplish unaided. This has motivated the development of shared-autonomy frameworks, in which autonomous assistance is integrated with user inputs to reduce workload and improve the stability and overall effectiveness of remote manipulation [8, 9].

When developing teleoperated robotic systems, especially in tasks where the robot must physically interact with the environment, collision avoidance becomes a central concern. In many practical settings, the workspace is narrow, cluttered, or partially occluded, making it difficult for the operator to judge distances or anticipate contacts based on limited visual information [10]. Studies have established that teleoperation becomes especially unreliable when the environment is cluttered or when the operator must react quickly to changing conditions [11]. In teleoperation and multi-robot studies, researchers found

that operators easily become overwhelmed when they must simultaneously control motion and avoid nearby obstacles, which often leads to inconsistent or unsafe behavior [12]. Even small errors in motion can lead to unintended impacts with surrounding objects, which may compromise both task performance and safety [13]. These findings suggest that obstacle avoidance can be as demanding as the primary manipulation task itself and highlight the need for assistance mechanisms that help maintain safety margins during remote operation.

Taken together, these difficulties show that teleoperation requires not only assistance for task execution but also mechanisms that can maintain safety during motion. Developing a framework that integrates user guidance with real-time collision avoidance is therefore essential for reliable operation in practical environments.

While these challenges highlight the need for assistance during teleoperation, many different approaches have been proposed to reduce operator workload and improve safety in complex environments. Existing work explores a wide range of strategies. Each of these methods addresses part of the problem, but they differ significantly in how they combine user input with autonomous support and in how they maintain safety near obstacles. The following section therefore summarizes related work on shared autonomy, safety-aware teleoperation, and assistance mechanisms for reliable teleoperation.

1.2 Related Work

1.2.1 Shared Autonomy

Shared autonomy has been widely explored as a way to reduce the burden of direct teleoperation by blending human commands with robot assistance. Early work established the idea of combining user input with autonomous control to make teleoperation more reliable, especially when perception or actuation channels are limited [8, 9]. A common strategy is to infer the operator's intended goal and adjust the level of assistance accordingly. Javdani et al. formalize this idea using a POMDP with uncertainty over the user's goal and propose a hindsight-optimization approximation that assists over a distribution of possible targets rather than committing to one [14]. Such methods have shown that shared autonomy can reduce input effort and speed up task execution, although users often report a trade-off between efficiency and their sense of control.

More recent work focuses on adapting the level and structure of assistance to the user. Atan et al. introduce an adaptive shared-autonomy controller that adjusts assistance based on the user's performance, enabling inexperienced operators to complete manipulation tasks more efficiently than with direct teleoperation [15]. Bowman and Zhang argue that using a single arbitration weight across all degrees of freedom can lead to over-dominance of the robot; they propose a dimension-specific arbitration policy that independently regulates assistance along each motion dimension, improving agreement between human and robot strategies during telemanipulation [16].

Assistance mechanisms have also been integrated into shared-autonomy frameworks to support obstacle avoidance and improve robustness. Lima et al. develop an MPC-based controller that augments teleoperation with predictive motion planning, improving task performance during object-picking tasks without removing user control authority [17]. Haptic-assistance systems follow a similar principle: Coffey and Pierson show that force cues can help operators avoid unsafe motions without removing their control authority [18].

Shared autonomy has also been applied to more complex telemanipulation settings. Ozdamar et al. propose a reconfigurable control framework that allows operators to control multiple robot arms individually or cooperatively, showing that shared autonomy can scale to higher-dimensional manipulation tasks with multi-arm systems [19]. These works highlight the potential of shared-autonomy techniques across a variety of teleoperation scenarios.

However, despite these advances, most shared-autonomy methods focus on predicting user intent, blending commands, or offering haptic or motion-level guidance. They typically improve behavior but do not provide *formal safety guarantees*, especially in narrow or cluttered workspaces where small errors can lead to collisions. Potential-field methods can become unreliable near local minima, and learning-based strategies depend heavily on the training distribution. Even MPC-based approaches may fail to guarantee constraint satisfaction under model mismatch or fast user inputs. These limitations suggest that shared autonomy alone is often not enough for ensuring reliable operation in safety-critical telemanipulation. This motivates integrating shared autonomy with *structured safety mechanisms*, such as Control Barrier Functions, to ensure that the robot remains safe even when user inputs are aggressive or misaligned with safe directions.

1.2.2 Obstacle Avoidance in Teleoperation

Obstacle avoidance has long been a central topic in robotics, and many teleoperation assistance methods build on early artificial potential field (APF) formulations. Khatib's real-time framework [20] models obstacles as repulsive potentials that produce continuous forces steering the robot away from collisions while preserving goal-directed motion. These methods are intuitive and computationally efficient but are sensitive to parameter choices and prone to local minima. To address these limitations, Rimon and Koditschek introduced navigation functions that construct global, smooth potentials guaranteeing collision-free convergence in structured environments [21]. While theoretically well-founded, such field-based approaches remain difficult to deploy in highly cluttered or narrow workspaces, which frequently arise in teleoperation.

Subsequent research shifted toward optimization-based collision avoidance. Trajectory-optimization frameworks [22] use signed-distance fields and gradient costs to compute smooth, collision-free motions, whereas real-time inverse-kinematics methods such as CollisionIK embed differentiable collision terms into per-frame nonlinear optimization [23]. These approaches achieve higher responsiveness than global planners, making them suitable for interactive control, but still do not provide formal guarantees of safety under aggressive user inputs or in tight environments.

Beyond explicit collision-avoidance strategies, safety in manipulation and teleoperation has also been studied through collision detection and reactive control. Haddadin et al. [24] demonstrated that fast torque-based detection and compliant reaction strategies can significantly reduce impact forces in physical human–robot interaction, complementing avoidance mechanisms rather than replacing them. More recently, safety-critical control formulations based on Control Barrier Functions (CBFs) have been explored to guarantee constraint satisfaction under uncertainty. CBF theory has been formalized and applied across a range of robotic systems [25]. While these methods provide strong safety guarantees, their integration into high-DoF teleoperation remains limited and risk-tunable CBF controllers have been proposed for human–robot collaboration [26].

1.3 Contributions and Overview

This thesis makes three main contributions toward safe and reliable assisted teleoperation in cluttered environments. First, we introduce a unified CLF–CBF shared-autonomy controller that combines a Control Lyapunov Function for task regulation with Control Barrier

Functions for real-time safety filtering. Unlike previous shared-autonomy approaches that rely on heuristic blending or soft potential-based costs, the proposed method embeds user commands, helper guidance, and safety constraints directly into a single quadratic program, enabling the robot to reason jointly about stability, intent alignment, and safety.

Second, we propose a state-dependent arbitration mechanism that adjusts the influence of the CLF objective according to the discrepancy between the user's command and the assistance target. Instead of overriding the operator when the input is unsafe, the controller gradually increases CLF assistance as the robot approaches the target, enabling more precise and stable convergence while still allowing the user to retain control during larger, exploratory motions.

Third, we evaluate the controller across three cluttered environments and compare it against BarrierIK and two shared-autonomy baselines: one that blends user with autonomous trajectories through a hierarchical switching scheme, and another that combines them within an optimization-based formulation. The results from reference trajectories and a user study show that the proposed CLF–CBF controller maintains positive clearance, prevents collisions, and provides more stable and predictable behavior than the alternative approaches.

The remainder of this thesis is organized as follows. Chapter 2 reviews the necessary background in robot kinematics and the fundamentals of Control Lyapunov and Control Barrier Functions. Chapter 3 introduces the overall methodology used in this work, including the formulation of the teleoperation framework, the proposed CLF–CBF shared-autonomy controller, the associated optimization formulation, and the state-dependent arbitration mechanism. Chapter 4 describes the experimental setup, the simulated environments, the baseline controllers, and the evaluation metrics, and reports the results of both the predefined-trajectory experiments and the user study. Finally, Chapter 5 summarizes the findings and discusses limitations and directions for future research.

2 Foundations

This chapter introduces the fundamental concepts that form the basis of the controller developed in this work. We begin by reviewing the kinematic representation of a robotic system, which describes how the state of the robot evolves through its underlying differential equations. Building on this, Lyapunov stability theory provides a systematic way to analyze whether the system converges to a desired equilibrium, and it offers the mathematical foundation for designing stabilizing controllers. Control Lyapunov Functions (CLFs) extend this classical method to controlled nonlinear systems by turning the qualitative notion of stability into a constructive condition on the control input. Together with Control Barrier Functions (CBFs), which impose safety constraints on the system state, these tools support the optimization-based controller architecture used later in this thesis. Introducing these concepts here clarifies how the theoretical foundations in this chapter directly connect to the safe shared-autonomy framework developed in the subsequent chapters.

2.1 Robot Kinematic Modeling

Kinematics describes the motion of a robot without considering the forces or torques that generate it[27]. It specifies the geometric relationship between the robot's joint variables and the position and orientation of its end-effector, which is important for motion planning and control [28]. For an n-joint robot, the joint configuration is written as

$$\mathbf{q} = \begin{bmatrix} q_1 & q_2 & \dots & q_n \end{bmatrix}^\mathsf{T} \in \mathbb{R}^n. \tag{2.1}$$

Each variable q_i represents the generalized coordinate associated with the i-th joint. The physical meaning of q_i depends on the mechanical structure of the joint. For a revolute joint, q_i corresponds to its rotation angle, whereas for a prismatic joint, q_i corresponds to its linear displacement. In all cases, the value of q_i specifies the current position of

the joint along its allowable degree of freedom and serves as one of the coordinates that define the robot's configuration in joint space.

The pose of the end-effector in Cartesian space is described by its position and orientation. In this work, the position is represented by a three-dimensional vector $\mathbf{p}$ and the orientation $\mathbf{r}$ is represented by a unit quaternion. With this choice, the end-effector pose is written as

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{r} \end{bmatrix} \in \mathbb{R}^7, \tag{2.2}$$

which completely specifies the position and orientation of the end-effector in three-dimensional space. Although the pose is written as a 7D vector, its time derivative can be expressed using a 6D spatial velocity (linear and angular). This quantity, known as the *twist*, is composed of the linear velocity and angular velocity of the end-effector,

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} \in \mathbb{R}^6, \tag{2.3}$$

which together characterize the full instantaneous motion of a rigid body in Cartesian space.

The forward kinematics defines a nonlinear differentiable mapping from the joint space to the Cartesian pose of the end-effector,

$$\mathbf{x} = \mathbf{f}(\mathbf{q}), \tag{2.4}$$

where $\mathbf{q} \in \mathbb{R}^n$ denotes the joint configuration and $\mathbf{x} \in \mathbb{R}^7$ is the corresponding end-effector pose expressed by its position and orientation.

The differential of the forward kinematics defines the Jacobian,

$$\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}(\mathbf{q}) \in \mathbb{R}^{6\times n}, \tag{2.5}$$

which is the first-order linearization of the mapping $\mathbf{f}$ evaluated at the current joint configuration $\mathbf{q}$. The reduction from seven to six rows arises because the time derivative of the pose is expressed as a spatial velocity (the twist), consisting of linear and angular velocity [29, 30, 31]. Evaluating this mapping at the configuration $\mathbf{q}$ yields the familiar relationship between joint and end-effector velocities,

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}. \tag{2.6}$$

The inverse kinematics problem seeks a joint configuration $\mathbf{q}$ that satisfies

$$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x}_d). \tag{2.7}$$

Here $\mathbf{x}_d$ is a desired end-effector pose, expressed by its position and quaternion orientation. To compute joint velocities that achieve a desired end-effector velocity $\dot{\mathbf{x}}_d$, the inverse differential kinematics problem must be solved. When the Jacobian is square and nonsingular, this mapping can be inverted directly,

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{-1}\,\dot{\mathbf{x}}_d. \tag{2.8}$$

For redundant or kinematically singular configurations, the Moore–Penrose pseudoinverse provides a least-squares solution,

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}(\mathbf{q})\,\dot{\mathbf{x}}_d, \tag{2.9}$$

where $\mathbf{J}^{\dagger}$ denotes the pseudoinverse of the Jacobian [29, 31]. This expression is widely used in robotics for real-time motion generation, redundancy resolution, and the enforcement of kinematic constraints.

The Jacobian provides a local linear approximation of the nonlinear kinematic mapping and links small changes in joint coordinates to instantaneous end-effector motion. It is widely used in velocity control, singularity analysis, and the enforcement of motion constraints, forming the basis for the control design used in later sections.

2.2 Operational-Space Control Fundamentals

Operational-Space Control (OSC) [32, 33] provides a task-level representation of manipulator motion. It describes how joint velocities influence the end-effector twist and enables control laws that act directly in task space. In this work, we focus on the kinematic form of OSC, which regulates task-space velocities rather than full robot dynamics. The end-effector motion is expressed as Cartesian velocity defined in (2.3).

A task-space controller regulates the end-effector pose $\mathbf{x}(t)$ toward a desired pose $\mathbf{x}_{\text{des}}(t)$

by constructing a desired Cartesian velocity:

$$\begin{aligned} \boldsymbol{\nu}_{\text{des}}(t) &= -K_{p,o} \begin{bmatrix} \mathbf{p}(t) - \mathbf{p}_{\text{des}}(t) \\ \delta\boldsymbol{\phi}(t) \end{bmatrix}, \\ \delta\boldsymbol{\phi}(t) &= 2\,\text{vec}(\mathbf{r}_{\text{err}}(t))\,, \\ \mathbf{r}_{\text{err}}(t) &= \mathbf{r}(t) \otimes \mathbf{r}_{\text{des}}^{-1}(t) \end{aligned} \tag{2.10}$$

Here $K_{p,o} \in \mathbb{R}^{6\times6}$ is a positive-definite gain that weights the position and orientation feedback terms. The variables $\mathbf{p}(t)$ and $\mathbf{r}(t)$ denote the current end-effector position and quaternion orientation, while $\mathbf{p}_{\text{des}}(t)$ and $\mathbf{r}_{\text{des}}(t)$ denote the desired position and quaternion orientation extracted from the target pose. $\delta\boldsymbol{\phi}(t)$ denotes the orientation error in a minimal 3D representation. The quaternion error encodes the relative rotation between the desired and current orientations, and the mapping $\delta\boldsymbol{\phi}(t) = 2\,\text{vec}(\mathbf{r}_{\text{err}}(t))$ extracts its vector part to obtain the smallest 3D rotation that aligns the two orientations.

The corresponding joint velocity is obtained from

$$\dot{\mathbf{q}}_{\text{task}} = \mathbf{J}^{\dagger}(\mathbf{q})\,\boldsymbol{\nu}_{\text{des}}, \tag{2.11}$$

where $\mathbf{J}^{\dagger}$ is the Moore–Penrose pseudoinverse of the end-effector Jacobian. This gives the minimum-norm joint motion that achieves the desired task-space behavior [34, 35, 36].

For redundant robots ($n > 6$), additional objectives can be introduced through the Jacobian null space

$$\begin{aligned} N(\mathbf{q}) &= \mathbf{I}_n - \mathbf{J}^{\dagger}(\mathbf{q})\,\mathbf{J}(\mathbf{q}), \\ \dot{\mathbf{q}}_N &= N(\mathbf{q})\,K_{p,j}\big(\mathbf{q}_{\text{des}} - \mathbf{q}(t)\big), \end{aligned} \tag{2.12}$$

where $N(\mathbf{q})$ projects joint velocities into the subspace that does not affect the end-effector motion and $\mathbf{I_n}$ denotes the $n \times n$ identity matrix, $\mathbf{q}_{\text{des}}$ is a preferred joint configuration, and $K_{p,j} > 0$ is a proportional gain that governs how strongly the posture error is corrected within the null space.

Combining task-space and null-space components yields the nominal joint velocity

$$\dot{\mathbf{q}}_{\text{nom}} = \dot{\mathbf{q}}_{\text{task}} + \dot{\mathbf{q}}_N. \tag{2.13}$$

This decomposition plays an important role in the safety-critical control framework developed in this thesis. It defines the nominal task behavior, separates task-relevant and redundant motions, and provides the structure through which safety constraints can be formulated in operational space.

2.3 Control Lyapunov Functions (CLF)

The idea of a Control Lyapunov Function (CLF) provides a systematic way to design stabilizing controllers for nonlinear systems. Unlike heuristic approaches such as PID tuning or feedback linearization, a CLF gives a general condition that guarantees asymptotic stability under a suitable feedback law. It connects classical Lyapunov stability analysis with the synthesis of nonlinear feedback controllers [37, 38].

Consider first an autonomous nonlinear system

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}), \tag{2.14}$$

where $\mathbf{z} \in \mathbb{R}^n$ is the state and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is the locally Lipschitz dynamics mapping, ensuring existence and uniqueness of solutions.

For a nonlinear robot model, the closed-form solution of the system trajectories is generally unavailable, and it is therefore difficult to reason directly about whether the state will converge to a desired equilibrium or remain bounded near it. What we ultimately want is a principled way to evaluate the stability properties of the dynamics and, later, to impose such properties through feedback in the controller design. Lyapunov functions provide exactly this mechanism: they act as an energy-like measure whose evolution along the system trajectories reveals whether the equilibrium is stable or attractive. Instead of relying on explicit solutions of Equation (2.14), stability can be inferred from the sign of the Lyapunov derivative. This viewpoint is foundational for the Control Lyapunov Function framework introduced later, in which stability requirements are converted into algebraic inequalities that can be enforced within an optimization-based controller.

A continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is called a Lyapunov function candidate if $V(\mathbf{0}) = 0$ and $V(\mathbf{z}) > 0$ for all $\mathbf{z} \neq \mathbf{0}$. If, in addition, $V(\mathbf{z}) \to \infty$ as $\|\mathbf{z}\| \to \infty$, the function is radially unbounded. From classical Lyapunov theory [39], the equilibrium $\mathbf{z} = 0$ of (2.14) is

- *Lyapunov stable* if $\dot{V}(\mathbf{z}) \leq 0$ in a neighborhood of the origin, and
- *asymptotically stable* if $\dot{V}(\mathbf{z}) < 0$ for all $\mathbf{z} \neq 0$ and $\dot{V}(0) = 0$.

These conditions verify stability but do not explain how to choose a control law that enforces them. This limitation motivates the extension of Lyapunov's method to the Control Lyapunov Function framework.

Most robotic systems can be written in the control-affine form

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}) + \mathbf{g}(\mathbf{z})\,\mathbf{u}, \tag{2.15}$$

where $\mathbf{u} \in \mathbb{R}^m$ is the control input. The vector field $\mathbf{f}$ describes the drift dynamics, and $\mathbf{g}$ determines how the control inputs influence the evolution of the state. The dimension n corresponds to the number of system states as defined in Eq. (2.1), and m represents the number of independent control inputs.

For a differentiable function $V(\mathbf{z})$, its time derivative along solutions of (2.15) follows from the chain rule,

$$\dot{V}(\mathbf{z}, \mathbf{u}) = \nabla V(\mathbf{z})^{\mathsf{T}} \dot{\mathbf{z}}, \tag{2.16}$$

and substitution of (2.15) yields

$$\dot{V}(\mathbf{z}, \mathbf{u}) = \nabla V(\mathbf{z})^{\mathsf{T}} \big(\mathbf{f}(\mathbf{z}) + \mathbf{g}(\mathbf{z})\,\mathbf{u}\big). \tag{2.17}$$

The first term defines the Lie derivative of V along $\mathbf{f}$, and the second term defines the Lie derivative of V along $\mathbf{g}$,

$$\begin{aligned} L_{\mathbf{f}} V(\mathbf{z}) &= \nabla V(\mathbf{z})^{\mathsf{T}} \mathbf{f}(\mathbf{z}), \\ L_{\mathbf{g}} V(\mathbf{z}) &= \nabla V(\mathbf{z})^{\mathsf{T}} \mathbf{g}(\mathbf{z}). \end{aligned} \tag{2.18}$$

The expressions $L_{\mathbf{f}} V$ and $L_{\mathbf{g}} V$ quantify how the Lyapunov function $V(\mathbf{z})$ changes when the state evolves along the vector fields $\mathbf{f}$ and $\mathbf{g}$, respectively. The Lie derivative $L_{\mathbf{f}} V(\mathbf{z})$ evaluates the directional change of V in the direction of the autonomous dynamics $\mathbf{f}$ at the state $\mathbf{z}$, while $L_{\mathbf{g}} V(\mathbf{z})$ measures how V changes due to the influence of the control input through the vector field $\mathbf{g}(\mathbf{z})$. In other words, $L_{\mathbf{f}} V$ captures the natural evolution of V under the system dynamics, and $L_{\mathbf{g}} V$ characterizes how the control input can increase or decrease V [37, 38].

Using these definitions, the time derivative of V can be written compactly as

$$\dot{V}(\mathbf{z}, \mathbf{u}) = L_{\mathbf{f}} V(\mathbf{z}) + L_{\mathbf{g}} V(\mathbf{z})\,\mathbf{u}, \tag{2.19}$$

which expresses how V evolves under the system's natural dynamics and the control input.

A positive-definite and continuously differentiable function V is called a Control Lyapunov Function if, for every nonzero state $\mathbf{z}$, there exists an input $\mathbf{u}$ such that

$$L_{\mathbf{f}} V(\mathbf{z}) + L_{\mathbf{g}} V(\mathbf{z})\,\mathbf{u} + \gamma(V(\mathbf{z})) \leq 0, \tag{2.20}$$

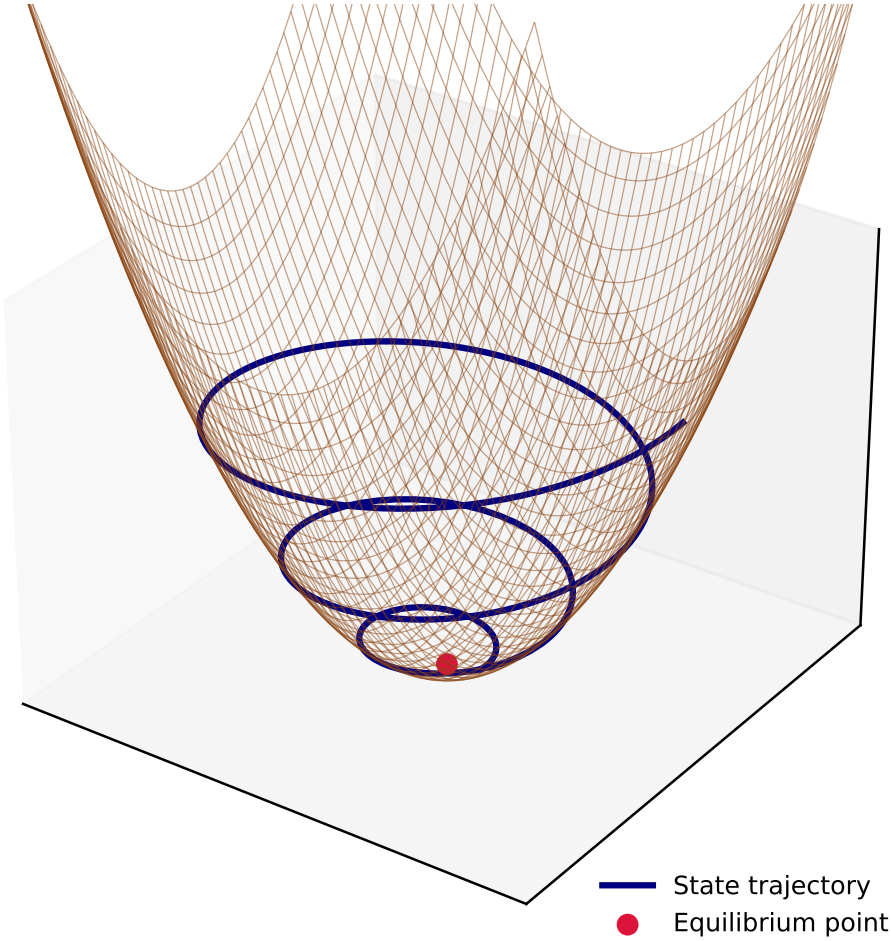


Figure 2.1: Example of a Control Lyapunov Function (CLF) surface for a two-dimensional nonlinear system. The contour lines represent level sets of $V(\mathbf{z})$, and the plotted trajectory illustrates how the state converges toward the equilibrium point where the Lyapunov function attains its minimum.

where γ is a class-$\mathcal{K}$ function [39]. Equivalently, this condition can be written as

$$\inf_{\mathbf{u}\in\mathbb{R}^m} \left[L_f V(\mathbf{z}) + L_g V(\mathbf{z})\,\mathbf{u} + \gamma(V(\mathbf{z}))\right] \leq 0, \qquad \forall\, \mathbf{z} \neq 0. \tag{2.21}$$

$$\inf_{\mathbf{u}\in\mathbb{R}^m} \left[L_{\mathbf{f}} V(\mathbf{z}) + L_{\mathbf{g}} V(\mathbf{z})\,\mathbf{u} + \gamma(V(\mathbf{z}))\right] \leq 0 \qquad \forall\, \mathbf{z} \neq 0. \tag{2.22}$$

If this holds, then for every state one can choose a control input that ensures $\dot{V}(\mathbf{z}, \mathbf{u}) \leq 0$, meaning the system is stabilizable.

For single-input systems ($m = 1$), Sontag [38] derived an explicit continuous stabilizing feedback law. In this special case, the control input $\mathbf{u}$ is scalar and can be written as

$$\mathbf{u}(\mathbf{z}) = -\frac{L_{\mathbf{f}} V(\mathbf{z}) + \sqrt{\left(L_{\mathbf{f}} V(\mathbf{z})\right)^2 + \left(L_{\mathbf{g}} V(\mathbf{z})\right)^4}}{L_{\mathbf{g}} V(\mathbf{z})},$$

provided $L_{\mathbf{g}} V(\mathbf{z}) \neq 0$. Generalizations for multi-input systems are available in the literature.

Geometrically, a CLF can be viewed as an energy landscape over the state space. Each level set of $V(\mathbf{z})$ represents a surface of constant energy, and enforcing $\dot{V} < 0$ moves the state toward regions of lower energy until the equilibrium is reached. The function $\gamma(V)$ controls how quickly V decreases and therefore influences the system's convergence rate. An example of such a CLF surface is shown in Fig. 2.1.

In summary, the CLF framework extends Lyapunov's classical method to controlled nonlinear systems. It converts the qualitative notion of stability into a constructive and control-affine inequality, which can be incorporated directly into quadratic programs or other optimization-based controllers. By ensuring the existence of inputs that always decrease a positive-definite function, CLFs provide a general characterization of stabilizability and form the basis of the CLF–CBF controllers widely used in modern safe robotic control [40, 41].

2.4 Control Barrier Functions

Control Barrier Functions (CBFs) provide a systematic approach for guaranteeing safety in nonlinear control systems. While Control Lyapunov Functions (CLFs) encode convergence toward a goal, CBFs ensure that the system state remains within a predefined admissible

region for all time. This duality of stability and invariance enables a unified optimization-based treatment of performance and safety in modern robotic control [41, 42, 43].

This admissible region is formally represented by the *safe set*. The safe set is defined as the superlevel set of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$,

$$\begin{aligned} \mathcal{C} &= \{\mathbf{z} \in \mathbb{R}^n \mid h(\mathbf{z}) \geq 0\}, \\ \partial\mathcal{C} &= \{\mathbf{z} \in \mathbb{R}^n \mid h(\mathbf{z}) = 0\}. \end{aligned} \tag{2.23}$$

In this formulation, states $\mathbf{z}$ satisfying $h(\mathbf{z}) > 0$ are considered safe, while the condition $h(\mathbf{z}) = 0$ characterizes the boundary of the safe set. The objective of a CBF-based controller is to guarantee that the set $\mathcal{C}$ remains forward invariant under the closed-loop dynamics, ensuring that once the system starts in the safe region it will never leave it [42].

Formally, forward invariance requires that any trajectory starting inside the set stays inside for all future time [43],

$$\mathbf{z}(0) \in \mathcal{C} \quad \Longrightarrow \quad \mathbf{z}(t) \in \mathcal{C},\ \forall\, t \geq 0. \tag{2.24}$$

For the control-affine system

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}) + \mathbf{g}(\mathbf{z})\,\mathbf{u}, \tag{2.25}$$

this can be expressed as a constraint on the time derivative of h along trajectories:

$$\dot{h}(\mathbf{z}, \mathbf{u}) = L_{\mathbf{f}} h(\mathbf{z}) + L_{\mathbf{g}} h(\mathbf{z})\,\mathbf{u}. \tag{2.26}$$

The control input $\mathbf{u} \in \mathbb{R}^m$ represents the actionable command applied to the system. The functions $\mathbf{f}(\mathbf{z})$ and $\mathbf{g}(\mathbf{z})$ describe the drift and control vector fields of the control-affine dynamics in (2.25).

A continuously differentiable h is called a Control Barrier Function if there exists at least one control input $\mathbf{u}$ such that

$$L_{\mathbf{f}} h(\mathbf{z}) + L_{\mathbf{g}} h(\mathbf{z})\,\mathbf{u} + \alpha\big(h(\mathbf{z})\big) \geq 0, \tag{2.27}$$

for all $\mathbf{z} \in \mathcal{C}$ and for a class-$\mathcal{K}$ function $\alpha(\cdot)$. This inequality ensures that once $h(\mathbf{z})$ becomes nonnegative, it cannot decrease below zero again, and hence the safe set $\mathcal{C}$ remains invariant. This variant is referred to as the Zeroing Control Barrier Function (ZCBF) [41]. The ZCBF condition is closely connected to the classical Nagumo theorem for set invariance, which characterizes when the vector field along the boundary of a set prevents trajectories from leaving it. This connection provides a formal interpretation of how ZCBFs guarantee forward invariance of the safe set without imposing unnecessary restrictions on

the motion [44]. An important property of Zeroing Control Barrier Functions is that they are minimally restrictive. Nagumo's theorem states that a closed set $\mathcal{C}$ is forward invariant if, at every boundary point $\mathbf{z} \in \partial\mathcal{C}$, the system vector field does not point outside the set.

Among all control laws that guarantee forward invariance of the safe set $\mathcal{C}$, the ZCBF condition in (2.27) is both necessary and sufficient on compact sets. Intuitively, the barrier constraint only excludes those control inputs that would decrease $h(\mathbf{z})$ in a way that violates safety, while allowing the controller to freely optimize performance everywhere else. Thus, CBFs encode safety with the least possible conservativeness.

In practical robotic applications, several independent safety conditions must hold simultaneously, such as joint limits, obstacle avoidance, and self-collision constraints. Each condition is represented by $h_i(\mathbf{z}) \geq 0$, $i = 1, \ldots, N$, where N denotes the number of independent safety conditions, yielding affine inequalities

$$L_{\mathbf{f}} h_i(\mathbf{z}) + L_{\mathbf{g}} h_i(\mathbf{z})\, \mathbf{u} + \alpha_i\big(h_i(\mathbf{z})\big) \geq 0, \quad \forall i. \tag{2.28}$$

The intersection $\mathcal{C} = \bigcap_i \mathcal{C}_i$ defines the overall safe set. Since all constraints are affine in $\mathbf{u}$, the feasible input set is convex, which facilitates efficient optimization-based control synthesis [41, 43]. A commonly used control synthesis method is the CLF–CBF Quadratic Program (QP) [25, 45]. It computes the minimally modified control input $\mathbf{u}$ close to a nominal command $\mathbf{u}_{\text{des}}$, while enforcing safety and stability conditions and is given by:

$$\min_{\mathbf{u}} \quad \|\mathbf{u} - \mathbf{u}_{\text{des}}\|^2 \tag{2.29a}$$

s.t.

$$L_{\mathbf{f}} V(\mathbf{z}) + L_{\mathbf{g}} V(\mathbf{z})\, \mathbf{u} + \gamma(V(\mathbf{z})) \leq 0, \tag{2.29b}$$

$$L_{\mathbf{f}} h_i(\mathbf{z}) + L_{\mathbf{g}} h_i(\mathbf{z})\, \mathbf{u} + \alpha_i(h_i(\mathbf{z})) \geq 0, \tag{2.29c}$$

$$\mathbf{u}_{\text{min}} \leq \mathbf{u} \leq \mathbf{u}_{\text{max}}. \tag{2.29d}$$

Here, the term $V(\mathbf{z})$ is a Control Lyapunov Function, enforcing convergence toward the task objective through this condition. Each safety constraint is encoded by a Control Barrier Function h_i. The bounds $\mathbf{u}_{\text{min}}$ and $\mathbf{u}_{\text{max}}$ enforce actuator limits and guarantee that the computed control input remains feasible for the robot.

Under mild regularity assumptions, the resulting optimal control $\mathbf{u}^\star(\mathbf{z})$ is unique and locally Lipschitz in $\mathbf{z}$ [42, 41]. The function $\alpha(\cdot)$ governs how aggressively the controller reacts near the boundary. Linear functions such as $\alpha(s) = \lambda s$ yield exponential-like buffers, while higher-order or saturating forms can mitigate stiffness and chattering. In practice,

normalization of constraint gradients and small margin inflations ($h \leftarrow h - \varepsilon$) improve numerical stability.

In summary, Control Barrier Functions provide a constructive mechanism to encode safety as affine constraints on the control input. Since safety is commonly defined as keeping the system state inside a prescribed admissible set, guaranteeing invariance of this safe set is a natural and essential requirement. When combined with CLFs in the QP framework, they yield a least-intrusive safety filter that minimally modifies the nominal control while guaranteeing invariance of the safe set. This framework has been validated in diverse robotic domains such as adaptive cruise control, obstacle avoidance, joint-limit enforcement, and legged locomotion [41, 43].

3 Methodology

This chapter introduces the control methodology implemented in this work. The goal is to combine human intention with autonomous safety guarantees so that the robot can assist the operator without restricting natural motion or causing collisions. To this end, the chapter builds up the components that form our final CLF–CBF controller.

Section 3.1 reviews the RelaxedIK formulation [46], which provides an optimization-based inverse kinematics baseline and defines the task-space objectives for manipulation. Section 3.2 then presents the shared-autonomy arbitration module that blends the operator's input with an autonomous target to generate nominal end-effector commands. Section 3.3 develops the Operational-Space Control Barrier Function (OSCBF) framework [47], which imposes geometric safety constraints while preserving task-consistent robot behavior. Finally, Section 3.4 integrates these elements into a unified CLF–CBF quadratic program that computes safe joint-velocity commands with formal safety guarantees under shared autonomy.

3.1 RelaxedIK

RelaxedIK [46] formulates inverse kinematics as a general constrained optimization problem, rather than relying on analytic kinematic inversion. At each control cycle, the solver constructs a set of differentiable task features $F_m(\mathbf{q})$ that quantify end-effector pose error, joint-limit proximity, motion smoothness, manipulability, and self-collision risk. Instead of enforcing end-effector pose as a hard constraint, RelaxedIK computes the next

joint configuration by minimizing a weighted sum of groove-shaped task costs:

$$\min_{\mathbf{q}} \quad \sum_{i=1}^{K} w_i \, G_i(F_i(\mathbf{q})) \tag{3.1a}$$

$$\text{s.t.} \quad \mathbf{q}_{\text{min}} \leq \mathbf{q} \leq \mathbf{q}_{\text{max}}, \tag{3.1b}$$

$$\sigma_{\text{min}}(\mathbf{J}(\mathbf{q})) \geq \varepsilon \tag{3.1c}$$

where $\mathbf{q}_{\text{min}}$ and $\mathbf{q}_{\text{max}}$ denote the joint-limit bound vectors. All other motion objectives, such as end-effector pose tracking, joint motion smoothness, self-collision avoidance, and manipulability, are incorporated as soft penalty terms through the feature functions $F_i(\mathbf{q})$, each corresponding to the i-th task objective function. They are associated with groove normalization functions $G_i(\cdot)$, rather than being imposed as explicit inequality constraints. In this constraint, the quantity $\sigma_{\text{min}}(\mathbf{J}(\mathbf{q}))$ denotes the smallest singular value of the Jacobian $\mathbf{J}(\mathbf{q})$. The singular value decomposition (SVD) of the Jacobian is given by

$$\begin{aligned} \mathbf{J}(\mathbf{q}) &= \mathbf{U} \, \text{diag}(\sigma_1, \ldots, \sigma_r) \, \mathbf{V}^{\mathsf{T}}, \\ \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r &> 0, \\ \sigma_{\text{min}}(\mathbf{J}(\mathbf{q})) &= \sigma_r, \\ \sigma_{\text{min}}(\mathbf{J}(\mathbf{q})) &\geq \varepsilon. \end{aligned} \tag{3.2}$$

Here, $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{r \times r}$ are orthogonal matrices containing the left and right singular vectors of the Jacobian $J(q)$, respectively. Equation (3.2) summarizes the manipulability constraint. The Jacobian $\mathbf{J}(\mathbf{q})$ is factorized via singular value decomposition (SVD), where the singular values $\sigma_1, \ldots, \sigma_r$ quantify the robot's ability to generate Cartesian velocities from joint velocities. The smallest singular value $\sigma_{\text{min}} = \sigma_r$ measures the distance to kinematic singularity: as it approaches zero, certain Cartesian directions require unbounded joint velocities. Enforcing the inequality $\sigma_{\text{min}}(\mathbf{J}(\mathbf{q})) \geq \varepsilon$ thus guarantees a minimum manipulability margin and prevents the robot from entering ill-conditioned or singular configurations.

A characteristic element of RelaxedIK is the use of a parametric Groove function to map each raw feature $F_i(\mathbf{q})$ into a numerically cost [46], as illustrated in Fig. 3.1.

$$G_i(F_i(\mathbf{q})) = (-1)^{n_i} \exp\left(-\frac{(F_i(\mathbf{q}) - s_i)^2}{2c_i^2}\right) + r_i \, (F_i(\mathbf{q}) - s_i)^4, \tag{3.3}$$

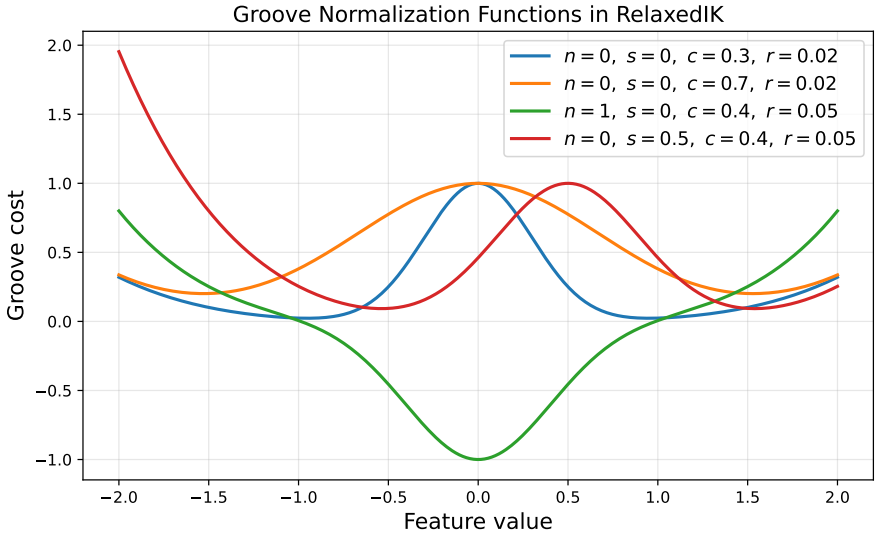


Figure 3.1: Examples of Groove-shaped normalization functions used in RelaxedIK. Each curve corresponds to a different parameter set $\{n_i, s_i, c_i, r_i\}$, demonstrating how the Gaussian "groove" around the target value s_i combines with a quartic term to produce a smooth, numerically stable cost landscape. These functions normalize heterogeneous task features into comparable ranges and yield well-behaved gradients for real-time IK optimization.

where $\{n_i, s_i, c_i, r_i\}$ are term-specific shape parameters. The Gaussian component forms a narrow "groove" around the desired target value s_i, while the surrounding quartic term yields stable growth away from this region. This normalization places heterogeneous task features into comparable numeric ranges, provides smooth gradients for real-time optimization, and allows the static weights w_i to function as intuitive importance multipliers.

With this structure, RelaxedIK can jointly encode end-effector position and orientation tracking within a single differentiable objective function. The problem (3.1) is solved in real time using a constrained nonlinear optimizer such as SLSQP. Because all motion goals enter as soft costs, exact satisfaction of individual objectives is not guaranteed. Instead, the solver automatically relaxes lower-priority features in favor of higher-priority ones when conflicts arise. In environments containing external obstacles or potential robot–environment contact, this motivates augmenting the objective with explicit environment-distance terms and, in our work, further extending the formulation toward constraint-based methods

such as CollisionIK and BarrierIK.

CollisionIK [23] extends the RelaxedIK optimization framework by explicitly incorporating environment–robot proximity information as an additional objective term. The method follows the same instantaneous IK optimization used in RelaxedIK, while introducing an additional distance term that increases the cost when the robot approaches obstacles.

Let $D_o(\mathbf{q})$ denote the minimum distance between a robot link and the o-th obstacle. CollisionIK defines a differentiable collision cost $F_{\text{env}}(\mathbf{q})$ by aggregating distances over a set of obstacles:

$$F_{\text{env}}(\mathbf{q}) = \sum_{o \in \mathcal{O}(t)} \frac{(5\varepsilon)^2}{D_o(\mathbf{q})^2}, \tag{3.4}$$

where $\varepsilon > 0$ is a user-selected safety margin and $\mathcal{O}(t)$ is a set of obstacles. The inverse-square structure ensures rapidly increasing cost as the robot approaches an obstacle, while remaining smooth and differentiable for use in gradient-based optimization.

CollisionIK augments the RelaxedIK objective by adding an environment–robot proximity term while keeping all original features inside the objective, such as pose tracking, joint-limit proximity, motion smoothness, manipulability, and self-collision avoidance. The instantaneous IK update is obtained by solving

$$\min_{\mathbf{q}} \quad \sum_{i=1}^{K} w_i \, G_i\big(F_i(\mathbf{q})\big) \; + \; \lambda_{\text{env}} \, G_{\text{env}}(F_{\text{env}}(\mathbf{q})) \tag{3.5a}$$

$$\text{s.t.} \quad \mathbf{q}_{\min} \le \mathbf{q} \le \mathbf{q}_{\max}, \tag{3.5b}$$

$$\sigma_{\min}(\mathbf{J}(\mathbf{q})) \ge \varepsilon \tag{3.5c}$$

where $F_{\text{env}}(\mathbf{q})$ penalizes proximity to obstacles and $G_{\text{env}}(\cdot)$ is the same Groove-shaped normalization function used in RelaxedIK. Since obstacle avoidance appears only as a soft penalty term, CollisionIK improves practical collision behavior but does not guarantee forward invariance or strict safety under competing objectives.

BarrierIK [48] advances this work by replacing CollisionIK's soft obstacle-avoidance penalties with hard safety constraints expressed through Control Barrier Functions (CBFs). Instead of discouraging proximity to obstacles via a cost term, BarrierIK explicitly enforces forward invariance of a safety set, ensuring that the IK update remains within the safety set under the CBF constraints.

In place of a separate signed-distance function, BarrierIK reuses the distance metric $D_o(\mathbf{q})$ already defined in CollisionIK, where $D_o(\mathbf{q})$ denotes the minimum distance between the

robot and the o-th obstacle. A control barrier function is constructed as

$$h_o(\mathbf{q}) = D_o(\mathbf{q}) - d_{\text{safe}},$$

with $d_{\text{safe}} > 0$ a user-defined safety margin. Ensuring $h_o(\mathbf{q}) \geq 0$ keeps the robot outside the inflated obstacle boundary.

To maintain this condition during each IK update, the joint position $\mathbf{q}$ is required to satisfy the CBF constraint,

$$\nabla h_o(\mathbf{q})^\top \dot{\mathbf{q}} + \alpha\big(h_o(\mathbf{q})\big) \;\geq\; 0, \tag{3.6}$$

where $\alpha(\cdot)$ is an extended class-$\mathcal{K}$ function determining how aggressively the method reacts as the boundary of the safe set is approached. When multiple obstacles are present, enforcing each CBF constraint individually can lead to abrupt switching between active constraints. To avoid such nonsmooth behavior, BarrierIK constructs a smooth approximation of the worst-case CBF constraint.

For each obstacle $o \in \mathcal{O}(t)$, define the corresponding CBF safety margin as

$$B_o(\mathbf{q}) = \nabla h_o(\mathbf{q})^\top \dot{\mathbf{q}} + \alpha\big(h_o(\mathbf{q})\big), \tag{3.7}$$

which quantifies how safely the current velocity command $\dot{\mathbf{q}}$ moves the robot relative to obstacle o. Positive values indicate satisfaction of the CBF condition, whereas negative values indicate an imminent violation of the safety boundary.

Rather than enforcing $B_o(\mathbf{q}) \geq 0$ for every obstacle individually that can lead to nonsmooth switching between active constraints, BarrierIK aggregates all obstacle-wise CBF conditions using a smooth log-sum-exp formulation

$$\tilde{B}(\mathbf{q}) = \frac{1}{\kappa} \log\left(\sum_{o \in \mathcal{O}(t)} \exp\big(\kappa\, B_o(\mathbf{q})\big) \right), \tag{3.8}$$

where $\kappa > 0$ controls the sharpness of the approximation. As $\kappa \to \infty$, the expression approaches $\max B_o(\mathbf{q})$, recovering the most restrictive CBF constraint.

BarrierIK enforces the multi-obstacle safety requirement by imposing

$$\tilde{B}(\mathbf{q}) \;\geq\; 0, \tag{3.9}$$

which ensures that the commanded joint velocity remains within the safe set defined by all obstacles in $\mathcal{O}(t)$.

With the obstacle-wise CBF constraints as (3.9), BarrierIK formulates inverse kinematics as a constrained optimization problem that preserves the flexible objective structure of RelaxedIK while guaranteeing geometric safety through CBF-enforced velocity limits. At each control cycle, the method computes a safe joint configuration $\mathbf{q}^\star$ by solving

$$\min_{\mathbf{q}} \quad \sum_{i=1}^{K} w_i \, G_i(F_i(\mathbf{q})) \tag{3.10a}$$

$$\text{s.t.} \quad \tilde{B}(\mathbf{q}) \geq 0, \tag{3.10b}$$

$$\mathbf{q}_{\text{min}} \leq \mathbf{q} \leq \mathbf{q}_{\text{max}}, \tag{3.10c}$$

$$\sigma_{\text{min}}(\mathbf{J}(\mathbf{q})) \geq \varepsilon \tag{3.10d}$$

Here, the objective retains all RelaxedIK feature terms, while the aggregated CBF constraint $\tilde{B}(\mathbf{q}) \geq 0$ ensures forward invariance of the collision-free set.

In this way, BarrierIK preserves the responsiveness and flexibility of optimization-based inverse kinematics while providing strict safety guarantees that soft-penalty formulations such as RelaxedIK and CollisionIK cannot offer.

3.2 Shared Autonomy

Shared autonomy aims to combine the operator's teleoperation input with robotic assistance to enhance manipulation performance, reduce user load, and prevent task failures in cluttered or constrained environments. In teleoperated manipulation, users typically specify motion commands or desired end-effector targets, while the robot must interpret these commands in the context of workspace geometry, joint limits, and task feasibility. This dual-objective structure, preserving user intent while ensuring feasible and safe execution, forms the foundation of modern shared autonomy research. A widely used approach is linear blending, where the final command is obtained as a weighted combination of human and autonomous actions [8, 49, 50]:

$$\mathbf{u}_{\text{blend}} = (1 - \alpha)\, \mathbf{u}_{\text{human}} + \alpha\, \mathbf{u}_{\text{auto}}, \tag{3.11}$$

Here, $\mathbf{u}_{\text{human}}$ denotes the operator-provided motion command, obtained from the teleoperation interface as a desired end-effector pose or an incremental motion target. The term

$\mathbf{u}_{\text{auto}}$ represents the autonomously generated motion suggested by the system, typically produced from the $\mathbf{x}_{\text{auto}}(t)$ or other task-level guidance signals. The end-effector pose of the manipulator is represented as (2.2). Both the operator-specified target pose $\mathbf{x}_{\text{human}}(t)$ and the autonomous target $x_{\text{auto}}(t)$ are represented in this form. The arbitration weight $\alpha(t)$ is computed from the discrepancy between the operator-specified target pose and the autonomous target. It reflects the consistency between human intention and autonomous guidance by combining both translational and rotational discrepancies. The position error is computed as the Euclidean distance

$$e_p(t) = \left\| \mathbf{p}_{\text{human}}(t) - \mathbf{p}_{\text{auto}}(t) \right\|, \tag{3.12}$$

while the orientation error is defined using the quaternion geodesic distance [51]

$$e_R(t) = 2 \arccos\Big(\left| \langle \mathbf{r}_{\text{human}}(t), \mathbf{r}_{\text{auto}}(t) \rangle \right| \Big), \tag{3.13}$$

where $\langle \cdot, \cdot \rangle$ denotes the quaternion inner product, ensuring invariance to the double cover of $SO(3)$. These errors are then mapped to a smooth arbitration coefficient through a sigmoid-based shaping function:

$$\alpha(t) = \sigma\left(c_p \frac{e_p(t)}{r_p} + c_R \frac{e_R(t)}{r_R} - h \right), \qquad \sigma(s) = \frac{1}{1 + e^{-s}}, \tag{3.14}$$

where $c_p, c_R > 0$ determine the steepness of the transition, $r_p, r_R > 0$ normalize the position and rotation scales, and $h > 0$ acts as a threshold. When the human and autonomous target are consistent, $\alpha(t) \approx 0$; as their difference grows, $\alpha(t)$ increases smoothly toward 1.

In this work, the teleoperation signal is provided by an external interface that continuously specifies a desired end-effector pose $\mathbf{x}_{\text{human}}(t)$. To regularize this user-specified motion and provide task-level guidance, an autonomous target $\mathbf{x}_{\text{auto}}(t) \in \mathbb{R}^7$ is computed from a higher-level task model, similar to assisted policies in manipulation tasks. The two poses are fused through the state-dependent arbitration weight $\alpha(t)$ defined in (3.14), which measures the positional and rotational discrepancies between the human target and the autonomous target.

The blended nominal pose is obtained by linearly interpolating the position and applying spherical linear interpolation (SLERP) [51] to the orientation:

$$\mathbf{p}_{\text{nom}}(t) = (1 - \alpha(t))\, \mathbf{p}_{\text{human}}(t) + \alpha(t)\, \mathbf{p}_{\text{auto}}(t), \tag{3.15}$$

$$\mathbf{r}_{\text{nom}}(t) = \text{slerp}\big(\mathbf{r}_{\text{human}}(t),\, \mathbf{r}_{\text{auto}}(t),\, \alpha(t)\big), \tag{3.16}$$

yielding

$$\mathbf{x}_{\text{nom}}(t) = \begin{bmatrix} \mathbf{p}_{\text{nom}}(t) \\ \mathbf{r}_{\text{nom}}(t) \end{bmatrix} \in \mathbb{R}^7.$$

This pose-blending strategy enables continuous and intuitive transitions between teleoperation and autonomous assistance, without abrupt switching effects reported in earlier shared-control systems.

The blended pose $\mathbf{x}_{\text{nom}}(t)$ is converted into a desired end-effector velocity using the operational-space tracking controller provided by the OSCBF framework [47]. Given the current end-effector pose $\mathbf{x}(t)$, the controller computes a Cartesian velocity command $\mathbf{v}_{\text{nom}}(t)$ whose detailed formulation will be presented in Section 3.3.

The resulting Cartesian velocity is then mapped into joint space through the manipulator Jacobian, using the differential kinematic relationship in (2.9):

$$\mathbf{u}_{\text{nom}}(t) = \mathbf{J}^{\dagger}(\mathbf{q})\, \mathbf{v}_{\text{nom}}(t), \tag{3.17}$$

yielding a joint-velocity-level representation of the shared-autonomy intent. Rather than executing $\mathbf{u}_{\text{nom}}(t)$ directly, we pass it to the OSCBF safety filter described in Section 3.3, ensuring that all executed commands respect geometric safety constraints such as obstacle avoidance, joint limits, and self-collision avoidance.

3.3 Operational-Space Control Barrier Functions

Control Barrier Functions (CBFs) introduced in Section 2.4 provide a general mechanism for enforcing safety through affine constraints of the form $h_i(\mathbf{z}) \geq 0$. For robotic manipulators, constraints must operate in concert with the operational-space control framework described in Section 2.2, where end-effector behavior and redundancy resolution are handled explicitly. OSCBFs [47] extend classical CBFs to this setting by embedding safety constraints directly within the operational-space hierarchy, ensuring that safety interventions remain consistent with the intended task behavior.

Following the kinematic OSC formulation, the nominal end-effector motion is first expressed as a proportional operational-space twist. Given a desired end-effector pose

$\mathbf{x}_{\text{des}} = [\mathbf{p}_{\text{des}}^{\mathsf{T}} \quad \mathbf{r}_{\text{des}}^{\mathsf{T}}]^{\mathsf{T}}$ and the current pose $\mathbf{x} = [\mathbf{p}^{\mathsf{T}} \quad \mathbf{r}^{\mathsf{T}}]^{\mathsf{T}}$, we construct a proportional operational-space twist by combining the position and orientation errors:

$$\boldsymbol{\nu}_{\text{nom}} = \boldsymbol{\nu}_{\text{des}} - K_{p,o} \begin{bmatrix} \mathbf{p} - \mathbf{p}_{\text{des}} \\ \delta\boldsymbol{\phi} \end{bmatrix}, \tag{3.18}$$

where $K_{p,o} \in \mathbb{R}^{6\times 6}$ is a positive (semi-)definite gain matrix, and $\delta\boldsymbol{\phi} \in \mathbb{R}^3$ denotes the instantaneous angular error between the current and desired orientations. The term $\boldsymbol{\nu}_{\text{des}}$ represents a feedforward operational-space twist, originating from higher-level task specifications such as reference motion commands. When no explicit feedforward twist is provided, a common choice also used in our implementation is to set $\boldsymbol{\nu}_{\text{des}} = \mathbf{0}$, so that the nominal twist is fully determined by the proportional correction on the pose error. The angular error is obtained using the quaternion representation of the current and desired orientations. $\mathbf{r}, \mathbf{r}_{\text{des}} \in \mathbb{R}^4$ denote the unit quaternions representing the current and desired orientations, respectively. The relative quaternion is

$$\mathbf{r}_{\text{err}} = \mathbf{r} \otimes \mathbf{r}_{\text{des}}^{-1}, \tag{3.19}$$

and the instantaneous orientation error is given by the vector part of this relative quaternion:

$$\delta\boldsymbol{\phi} = 2\,\text{vec}(\mathbf{r}_{\text{err}}) \in \mathbb{R}^3. \tag{3.20}$$

Equation (3.18) thus defines a task-consistent desired twist that drives the end-effector pose toward $\mathbf{x}_{\text{des}}$ with proportional feedback in both position and orientation. The corresponding joint-space motion is obtained from the Moore–Penrose pseudoinverse of the manipulator Jacobian:

$$\dot{\mathbf{q}}_{\text{task}} = \mathbf{J}^{\dagger}(\mathbf{q})\,\boldsymbol{\nu}_{\text{nom}}. \tag{3.21}$$

To regulate redundancy, a proportional posture term is projected into the Jacobian null space, given by (2.12)

$$\begin{aligned} \dot{\mathbf{q}}_N &= N(\mathbf{q})\left(\dot{\mathbf{q}}_{\text{des}} - K_{p,j}\big(\mathbf{q} - \mathbf{q}_{\text{des}}\big)\right), \\ N(\mathbf{q}) &= \mathbf{I}_n - \mathbf{J}^{\dagger}(\mathbf{q})\,J(\mathbf{q}), \end{aligned} \tag{3.22}$$

Here, $\dot{\mathbf{q}}_{\text{des}}$ denotes an optional feedforward joint-space velocity provided by the higher-level task specification. The term $K_{p,j} > 0$ is a scalar proportional gain that regulates the null-space posture, penalizing the joint deviation $\mathbf{q} - \mathbf{q}_{\text{des}}$ within the projector $N(\mathbf{q})$.

Combining the task-space and null-space components yields the nominal joint velocity prior to safety filtering in (2.13):

$$\dot{\mathbf{q}}_{\text{nom}} = \dot{\mathbf{q}}_{\text{task}} + \dot{\mathbf{q}}_N. \tag{3.23}$$

This expression enforces strict task hierarchy: redundancy objectives are pursued only when they do not conflict with task execution [52].

A crucial requirement in extending CBFs to operational space is task consistency. Within the OSC hierarchy, the end-effector task has strict priority, while redundancy objectives are confined to the Jacobian null space. Any safety-induced modification to the joint velocity must therefore preserve this hierarchy: corrections should be injected first in the null space and only affect the primary task when no feasible null-space adjustment exists. This property ensures that the safety filter does not introduce unintended motions in operational space or distort the task-consistent behavior defined by the OSC controller [47, 53].

In contrast to the general CBF formulation introduced in Section 2.4, the control input for a velocity-controlled manipulator is the joint velocity $\dot{\mathbf{q}}$. Consequently, the safety constraints $h_i(\mathbf{z}) \geq 0$ are enforced directly at the joint-velocity level, using the same Lie-derivative structure but now expressed with respect to $\dot{\mathbf{q}}$. Because the system input is $\dot{\mathbf{q}}$, the CBF condition takes the form

$$L_{\mathbf{f}} h_i(\mathbf{z}) + L_{\mathbf{g}} h_i(\mathbf{z})\, \dot{\mathbf{q}} + \alpha_i\big(h_i(\mathbf{z})\big) \geq -\delta_i, \tag{3.24}$$

where the slack variable $\delta_i \geq 0$ ensures feasibility when multiple constraints become simultaneously active. A key distinction from joint-space CBFs is that OSCBFs naturally incorporate task-space constraints through the Jacobian mapping. This allows OSCBFs to reason about both joint-level and task-level safety within a unified formulation, while remaining compatible with the operational-space control structure introduced in Section 2.2.

The aim of the OSCBF filter is to compute a safe joint velocity $\dot{\mathbf{q}}^\star$ that remains as close as possible to the nominal operational-space behavior while satisfying all CBF constraints. To preserve task hierarchy, deviations are penalized separately in task space and null

space [47]. This leads to the optimization problem:

$$\min_{\dot{\mathbf{q}},\, \delta_1, \ldots, \delta_N} \quad \left\|\dot{\mathbf{q}} - \dot{\mathbf{q}}_{\text{nom}}\right\|^2 + \sum_{i=1}^{N} \rho\, \delta_i^2 \tag{3.25a}$$

$$\text{s.t.} \quad L_{\mathbf{f}} h_i(\mathbf{z}) + L_{\mathbf{g}} h_i(\mathbf{z})\, \dot{\mathbf{q}} + \alpha_i\big(h_i(\mathbf{z})\big) \geq -\delta_i, \quad i = 1, \ldots, N, \tag{3.25b}$$

$$\dot{\mathbf{q}}_{\text{min}} \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}_{\text{max}}, \tag{3.25c}$$

$$\delta_i \geq 0, \quad i = 1, \ldots, N \tag{3.25d}$$

The quadratic objective penalizes deviation from the nominal joint velocity $\dot{\mathbf{q}}_{\text{nom}}$, which already encodes both task-space tracking and null-space regulation as defined in (3.22) and (3.23). Thus, the QP solution remains as close as possible to the desired OSC behavior while enforcing safety through CBF constraints, and the CBF constraints guarantee forward invariance of the safe set. Under standard assumptions, the QP is convex and admits a unique, locally Lipschitz solution [41, 43, 47].

The OSCBF formulation extends classical CBF-based filtering into the operational space by tightly coupling three components: the task-consistent motion generated by operational-space control, the redundancy resolution provided through null-space regulation, and the safety constraints expressed at the joint-velocity level. By embedding these elements within a single quadratic program, the OSCBF controller acts as a minimally invasive safety filter: whenever all safety constraints are satisfied, the solution $\dot{\mathbf{q}}^\star$ coincides with the nominal OSC command $\dot{\mathbf{q}}_{\text{nom}}$, and deviations only occur when required to maintain forward invariance of the safe set. This property ensures that safety enforcement remains aligned with the intended task behavior and preserves the hierarchical structure of OSC.

3.4 Unified CLF–CBF Quadratic Program

The control input in this method is the joint velocity command $\mathbf{u} \in \mathbb{R}^n$. Consequently, the configuration $\mathbf{q}$ evolves according to the velocity level

$$\dot{\mathbf{q}} = \mathbf{u}. \tag{3.26}$$

This formulation allows the CLF and CBF conditions to be written using the Lie derivative with respect to the input vector field. The human target $\mathbf{x}_{\text{human}}$ enters the controller

through the operational-space tracking law introduced as (3.18), (3.21) and (3.22),

$$\begin{aligned} \boldsymbol{\nu}_{\text{nom}} &= \boldsymbol{\nu}_{\text{des}} - K_{p,o} \begin{bmatrix} \mathbf{p}(\mathbf{q}) - \mathbf{p}_{\text{human}} \\ \delta\boldsymbol{\phi}\big(\mathbf{r}(\mathbf{q}),\, \mathbf{r}_{\text{human}}\big) \end{bmatrix}, \\ \dot{\mathbf{q}}_{\text{nom}} &= \mathbf{J}^{\dagger}(\mathbf{q})\, \boldsymbol{\nu}_{\text{nom}} + N(\mathbf{q}) \big(\dot{\mathbf{q}}_{\text{des}} - K_{p,j}\, (\mathbf{q} - \mathbf{q}_{\text{des}})\big), \end{aligned} \tag{3.27}$$

Here $\mathbf{p}(\mathbf{q})$ and $\mathbf{r}(\mathbf{q})$ denote the end–effector position and quaternion orientation obtained from (2.4). The terms $\mathbf{p}_{\text{human}}$ and $\mathbf{r}_{\text{human}}$ represent the position and quaternion extracted from the human-specified target $\mathbf{x}_{\text{human}}$. The vector $\delta\boldsymbol{\phi}\big(\mathbf{r}(\mathbf{q}), \mathbf{r}_{\text{human}}\big) \in \mathbb{R}^3$ is the instantaneous orientation error defined in (3.20), obtained from the vector part of the relative quaternion between $\mathbf{r}(\mathbf{q})$ and $\mathbf{r}_{\text{human}}$.

The matrix $\mathbf{J}^{\dagger}(\mathbf{q})$ is the Moore–Penrose pseudoinverse of the end–effector Jacobian, and $N(\mathbf{q})$ projects motions into the Jacobian null space.

The autonomous target $\mathbf{x}_{\text{auto}}$ specifies the reference for the Control Lyapunov Function associated with the end-effector task,

$$\begin{aligned} V(\mathbf{q}) &= \tfrac{1}{2} \big\| \mathbf{x}(\mathbf{q}) - \mathbf{x}_{\text{auto}} \big\|^2, \\ L_{\mathbf{f}} V(\mathbf{q}) + L_{\mathbf{g}} V(\mathbf{q})\, \mathbf{u} + \gamma\big(V(\mathbf{q})\big) \;&\leq\; \delta_{\text{clf}}, \end{aligned} \tag{3.28}$$

where $L_{\mathbf{f}} V(\mathbf{q})$ and $L_{\mathbf{g}} V(\mathbf{q})$ are the Lie derivative of V along the control vector field. The function $\gamma(\cdot)$ is a class-$\mathcal{K}$ function.

An important motivation of our design is to adjust the influence of the CLF and CBF terms based on the current configuration and task conditions. When the human operator is exploring or moving coarsely, the controller should give more authority to the human input. In contrast, when the robot is close to the autonomous target, autonomous assistance must become more dominant. To achieve this behavior, we introduce state-dependent relaxation weights for both the CLF and the CBF. To modulate the influence of the CLF, the discrepancy between the human and autonomous target is defined as

$$d_{\text{ref}} = \big\| \mathbf{p}_{\text{human}} - \mathbf{p}_{\text{auto}} \big\|. \tag{3.29}$$

The quantity d_{ref} in (3.29) measures the Euclidean distance between the human-specified target position $\mathbf{p}_{\text{human}}$ and the autonomous target $\mathbf{p}_{\text{auto}}$. When the two targets are far apart, the operator is still exploring and the CLF should exert little influence and when

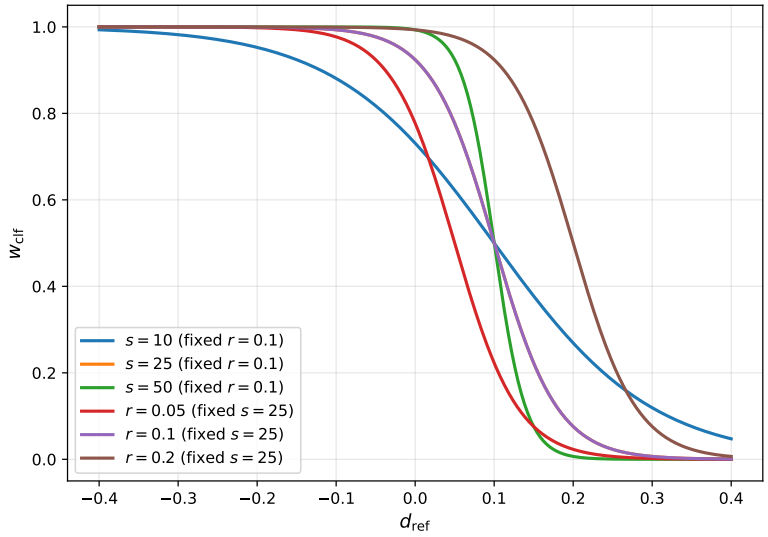


Figure 3.2: Logistic weighting used for CLF arbitration. The weighting w_{clf} decreases as the reference discrepancy d_{ref} between the human target and the autonomous target increases, so that the CLF has strong influence only when the two targets are close. Varying the sharpness parameter s changes how abruptly this decline occurs, while varying the offset parameter r shifts the distance at which the weight begins to drop. A smaller d_{ref} thus yields a higher CLF weight, promoting precise convergence, whereas larger discrepancies suppress the CLF and allow the operator to retain full control authority.

the targets nearly coincide, precise convergence is desired and the CLF should dominate. We model this transition using a smooth logistic weighting

$$w_{\text{clf}}(\mathbf{x}_{\text{human}}, \mathbf{x}_{\text{auto}}) = \frac{w_{\max}}{1 + \exp(s\,(d_{\text{ref}} - r))}, \tag{3.30}$$

in this expression, $w_{\max} > 0$ specifies the maximum emphasis on the CLF when the two targets coincide. The parameter $s > 0$ controls the sharpness of the transition, while $r > 0$ determines the distance at which the CLF begins to lose influence. These effects are illustrated in Fig. 3.2, where varying s changes the steepness of the logistic curve and varying r shifts the point at which the weight starts to decay.

In parallel, the CBF relaxation weight increases as the system moves closer to the safety

boundary. The smallest safety margin is defined as

$$h_{\text{min}} = \min_i h_i(\mathbf{q}), \tag{3.31}$$

and the state-dependent penalty is given by

$$w_{\text{cbf}}(\mathbf{q}) = w_0\big(1 + \exp(-k\, h_{\text{min}})\big), \tag{3.32}$$

which grows rapidly as $h_{\text{min}} \to 0^+$, causing the CBF constraints to dominate the optimization near the boundary. In this expression, $w_0 > 0$ determines the baseline penalty far from constraints, while $k > 0$ controls how sharply the weight increases as the robot approaches a potential violation. To prioritize safety over performance, the maximum CBF penalty is chosen to dominate the maximum CLF penalty,

$$w_0 \;\gg\; w_{\text{clf,max}}. \tag{3.33}$$

In all experiments we set

$$w_0 = 10^4, \qquad w_{\text{clf,max}} = 2 \times 10^2, \tag{3.34}$$

so that relaxing a CBF constraint is orders of magnitude more expensive than relaxing the CLF constraint, ensuring that safety is never traded for convergence.

Combining the nominal velocity generated from the human target, the autonomous target based Control Lyapunov Function, and the set of safety constraints encoded by the Control Barrier Functions leads naturally to a constrained optimization problem. The control input $\mathbf{u}$ and the relaxation variables associated with the CLF and CBF conditions are treated as optimization variables. The objective balances three contributions: deviation from the human-driven nominal command $\mathbf{u}_{\text{nom}}$, the relaxation required to satisfy the CLF condition, and the relaxation needed to keep the CBF constraints feasible. The state-dependent penalties w_{clf} and w_{cbf} modulate the relative influence of these contributions. In particular, w_{clf} increases when the human and autonomous target are close and decreases when they diverge, enabling a smooth transition between operator-dominated and autonomy-dominated behavior. Similarly, w_{cbf} strengthens the effect of the safety constraints as the system approaches the boundary of the admissible set.

$$
\begin{aligned}
\min_{\mathbf{u},\,\delta_{\text{clf}},\,\delta_{\text{cbf}}} \quad & \left\|\mathbf{u}-\mathbf{u}_{\text{nom}}\right\|^2 + w_{\text{clf}}(\mathbf{x}_{\text{human}},\mathbf{x}_{\text{auto}})\,\delta_{\text{clf}}^2 + w_{\text{cbf}}(\mathbf{q})\,\delta_{\text{cbf}}^2 && \text{(3.35a)}\\
\text{s.t.} \quad & L_{\mathbf{f}}V(\mathbf{q}) + L_{\mathbf{g}}V(\mathbf{q})\,\mathbf{u} + \gamma\big(V(\mathbf{q})\big) \le \delta_{\text{clf}}, && \text{(3.35b)}\\
& L_{\mathbf{f}}h_i(\mathbf{q}) + L_{\mathbf{g}}h_i(\mathbf{q})\,\mathbf{u} + \alpha\big(h_i(\mathbf{q})\big) \ge -\,\delta_{\text{cbf}}, \quad i=1,\dots,N, && \text{(3.35c)}\\
& \delta_{\text{clf}},\,\delta_{\text{cbf}} \ge 0, && \text{(3.35d)}\\
& \mathbf{u}_{\min} \le \mathbf{u} \le \mathbf{u}_{\max} && \text{(3.35e)}
\end{aligned}
$$

Under this formulation, the controller no longer simply follows the nominal velocity. Instead, it selects the joint command that balances human intent, helper-guided task convergence, and safety preservation, with their relative influence adjusted automatically through the state-dependent weighting functions. These elements come together in a unified optimization framework, in which the controller continuously reconciles operator input, autonomous assistance, and safety requirements, producing a joint velocity that reflects the appropriate priority dictated by the current state.

4 Experiments

This chapter presents a comprehensive evaluation of the proposed shared-autonomy CLF–CBF control framework on the Franka Emika Panda manipulator. The experiments are conducted across several cluttered and geometrically constrained teleoperation scenarios to examine how the controller balances operator intent with real-time safety guarantees. The evaluation focuses on situations where user-specified motion commands frequently conflict with environmental constraints such as obstacles, narrow passages, and joint-limit boundaries.

4.1 Experimental Setup

All experiments are carried out using a Franka Emika Panda robot simulated with the `JAXSim` accelerated physics framework [54], which internally relies on the NVIDIA PhysX engine, together with ROS and a Unity-based teleoperation interface. The system is designed to run at 100 Hz, although the actual frequency depends on the controller, since each method requires a different amount of computation.

The operator controls the robot through a keyboard interface in Unity. At each control cycle, the keyboard provides small position and rotation increments, which are integrated into a continuous human target pose $\mathbf{x}_{\text{human}}(t)$. This pose is updated at the same rate as the controller so that user commands are passed directly into the shared-autonomy module and the CLF–CBF controller. The autonomous target $\mathbf{x}_{\text{auto}}(t)$ is defined from fixed reference poses and corresponds to the desired grasping location in each scene.

Three environments of increasing difficulty are used to evaluate the teleoperation performance. The simplest scenario contains only two static obstacles that form a relatively wide narrow passage, providing a low-difficulty setting in which the fundamental behavior of the CLF–CBF controller can be observed. The second scenario, illustrated on the left

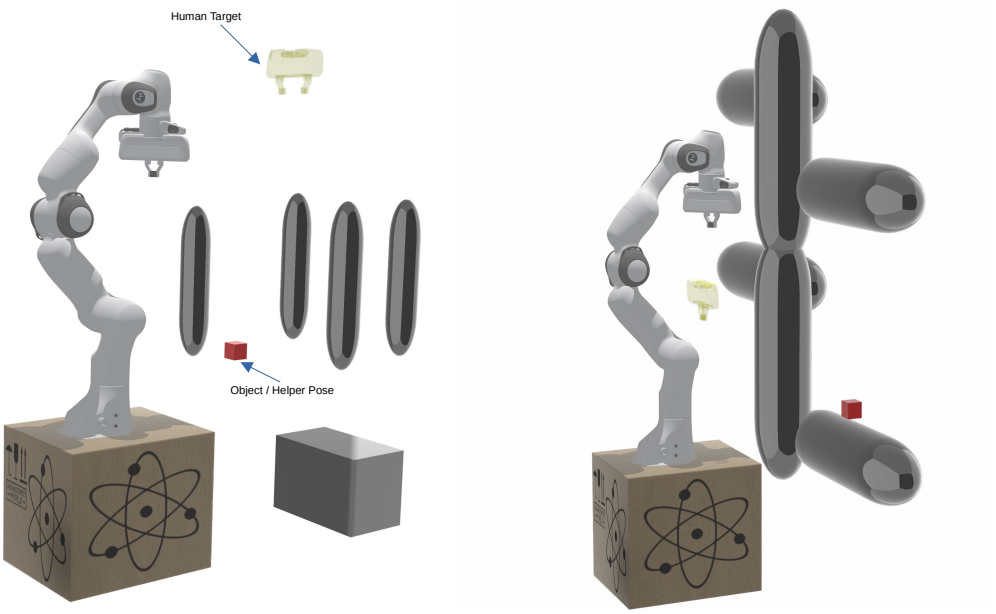


Figure 4.1: Overview of the two simulated environments used in the teleoperation experiments. **Left**: narrow-passage scenario containing multiple closely spaced obstacles. **Right**: shelf scenario in which the robot reaches toward a red cube placed inside a confined box-like structure.

side of Fig. 4.1, increases the complexity by placing four closely spaced obstacles around the approach path. This arrangement creates a much tighter corridor, in which small deviations of the human target pose can trigger strong safety interventions. The most challenging scenario, shown on the right side of Fig. 4.1, embeds the target cube inside a shelf-like structure, requiring the end-effector to maneuver within limited clearance and making shared autonomy particularly beneficial when the human input becomes imprecise.

All obstacle geometries are modeled as capsules, and the robot–obstacle signed distance is computed using a differentiable proximity function [55]. These distances are used inside the OSCBF constraints to enforce joint limits, avoid self-collision, and maintain clearance from obstacles. During teleoperation, the end-effector tracks the human target pose, while the shared-autonomy module slowly increases the influence of the autonomous target pose as the robot approaches the cube. Fine alignment and the final grasping motion remain

under full human control, with safety constraints active throughout the entire task.

Controller-behavior evaluation: In the first set of experiments, the human target is not provided by a real operator. Instead, it follows a predefined smooth trajectory generated in Unity. This trajectory passes near the autonomous target pose, but does not involve grasping. By removing human variability, this experiment provides a clear view of how the CLF–CBF controller reacts when approaching obstacles, moving through narrow passages, and entering or leaving the autonomous target's influence. The robot first follows the reference human target and then gradually shifts toward the autonomous target pose as the arbitration weight increases, before returning to the reference trajectory once the target moves away. Each scene is evaluated with five repeated trials. It is performed only in the simple two-obstacle narrow-passage and shelf scenes.

Teleoperated grasping tasks The second set of experiments involves real human operators controlling the target pose to reach and grasp a red cube placed among clutter. The three workspace layouts—ranging from a simple narrow passage to the dense narrow passage and the shelf environment in Fig. 4.1, offer increasing levels of difficulty. Each participant performs three trials in every scene. Operators attempt to drive the end-effector toward the cube, while the shared-autonomy module increases the influence of the autonomous target when the target enters the grasping region. The assistance does not override the operator: final alignment, orientation adjustment, and the last part of the reach-to-grasp motion remain fully under human control. Across all scenes, the controller should enforce joint limits, maintain clearance from obstacles, and prevent self-collision, thereby allowing users to focus on task execution without manually accounting for safety.

4.2 Baselines

Three controllers are used as baselines for evaluating the proposed shared-autonomy CLF–CBF framework. All baselines receive the same human-provided end-effector pose commands but differ in how autonomy is introduced and how safety constraints are enforced.

BarrierIK BarrierIK [48] is used as the pure-teleoperation baseline. Unlike velocity-level CBF controllers, BarrierIK formulates obstacle avoidance directly at the configuration level: at every control cycle, the method computes the next joint configuration $\mathbf{q}^\star$ by solving a position-based inverse-kinematics optimization as (3.10). The operator's desired end-effector pose is mapped to an IK objective, and BarrierIK enforces safety by constraining the joint increment $\Delta\mathbf{q}$ using discrete-time Control Barrier Functions.

As a baseline, BarrierIK represents a purely teleoperated controller. The robot tracks the human-provided pose command directly, without any form of shared autonomy or autonomous assistance. It guarantees geometric safety at the configuration level but does not guide the operator toward the red cube or modify the commanded motion based on task intent.

Baseline 2 The second baseline evaluates shared-autonomy blending in combination with OSCBF-based safety filtering as (3.25). At each control cycle, the operator specifies a target end-effector pose $\mathbf{x}_{\text{human}}(t)$, while the system provides an autonomous target pose $\mathbf{x}_{\text{auto}}(t)$. The arbitration weight $\alpha(t)$ is computed using the sigmoid formulation defined in (3.14), and the blended nominal pose is constructed as

$$\begin{aligned}\mathbf{p}_{\text{nom}}(t) &= \alpha\big(\mathbf{x}_{\text{human}}(t), \mathbf{x}_{\text{auto}}(t)\big)\ \mathbf{p}_{\text{human}}(t) + \Big(1 - \alpha\big(\mathbf{x}_{\text{human}}(t), \mathbf{x}_{\text{auto}}(t)\big)\Big)\ \mathbf{p}_{\text{auto}}(t), \\ \mathbf{r}_{\text{nom}}(t) &= \text{slerp}\big(\mathbf{r}_{\text{human}}(t),\ \mathbf{r}_{\text{auto}}(t),\ \alpha\big(\mathbf{x}_{\text{human}}(t), \mathbf{x}_{\text{auto}}(t)\big)\big)\ .\end{aligned} \tag{4.1}$$

yielding the blended pose $\mathbf{x}_{\text{nom}}(t)$

The nominal pose $\mathbf{x}_{\text{nom}}(t)$ is converted into a desired operational-space twist using the OSC formulation introduced in (3.18). Given the current pose $\mathbf{x}(t)$, the operational-space tracking law produces

$$\boldsymbol{\nu}_{\text{nom}}(t) = -K_{p,o} \begin{bmatrix} \mathbf{p}(t) - \mathbf{p}_{\text{nom}}(t) \\ \delta\boldsymbol{\phi}(t) \end{bmatrix}, \tag{4.2}$$

where $\delta\boldsymbol{\phi}(t)$ denotes the instantaneous angular error. This twist is mapped to joint space through the Jacobian pseudoinverse, representing the nominal shared-autonomy command.

$$\mathbf{u}_{\text{des}}(t) = \dot{\mathbf{q}}_{\text{nom}}(t) = \mathbf{J}^\dagger(\mathbf{q})\, \boldsymbol{\nu}_{\text{nom}}(t), \tag{4.3}$$

The nominal joint velocity $\mathbf{u}_{\text{des}}(t)$ is then passed to the OSCBF safety filter described in (3.25). The filter solves the QP, treating $\mathbf{u}_{\text{des}}(t)$ as the reference motion while enforcing

joint-limit, self-collision, and obstacle-avoidance CBF constraints. This yields the safe joint velocity $\dot{\mathbf{q}}^\star(t)$ which is guaranteed to remain within the safe set.

This baseline therefore evaluates shared-autonomy blending combined with OSCBF-based safety filtering, but without any CLF objective. The robot follows the blended human–helper intention as long as all safety constraints are satisfied, and deviations occur only when required to maintain forward invariance of the safe set.

Baseline 3 The third baseline evaluates an alternative shared-autonomy strategy in which blending is performed directly inside the safety-filtering optimization. Unlike the previous baseline, which first computes a blended nominal pose and then converts it into a nominal joint velocity $\dot{\mathbf{q}}_{\text{nom}}$, this method delivers two separate nominal joint velocities to the OSCBF controller: a human-driven velocity $\dot{\mathbf{q}}_{\text{human}}$ and an autonomous velocity $\dot{\mathbf{q}}_{\text{auto}}$.

At each control cycle, the operational-space controller described in (3.18) and (3.21) is used to generate two nominal joint velocities $\dot{\mathbf{q}}_{\text{human}}$ and $\dot{\mathbf{q}}_{\text{auto}}$. Both commands are generated using identical OSC tracking gains and the same end-effector Jacobian, ensuring comparable structure across the two velocity fields.

A state-dependent arbitration weight $w_{\text{human}}(\mathbf{x}_{\text{human}}, \mathbf{x}_{\text{auto}}) \in [0, 1]$ is computed from the discrepancy between the human target and autonomous target pose, following the shared-autonomy blending function described in (3.14). The complementary weight

$$w_{\text{auto}}(\mathbf{x}_{\text{human}}, \mathbf{x}_{\text{auto}}) = 1 - w_{\text{human}}(\mathbf{x}_{\text{human}}, \mathbf{x}_{\text{auto}}) \tag{4.4}$$

determines the influence of the autonomous component.

Instead of forming a blended nominal velocity externally, the OSCBF QP receives the two components separately and embeds the blending directly inside its cost function. The safety filter then computes a safe joint velocity $\dot{\mathbf{q}}^\star$ by solving the following quadratic program:

$$\begin{aligned}
\min_{\dot{\mathbf{q}},\, \delta_1, \ldots, \delta_N} \quad & w_{\text{human}}(t) \left\| \dot{\mathbf{q}} - \dot{\mathbf{q}}_{\text{human}} \right\|^2 + w_{\text{auto}}(t) \left\| \dot{\mathbf{q}} - \dot{\mathbf{q}}_{\text{auto}} \right\|^2 + \sum_{i=1}^{N} \rho\, \delta_i^2 && \text{(4.5a)} \\
\text{s.t.} \quad & L_{\mathbf{f}} h_i(\mathbf{z}) + L_{\mathbf{g}} h_i(\mathbf{z})\, \dot{\mathbf{q}} + \alpha_i(h_i(\mathbf{z})) \geq -\delta_i, \quad i = 1, \ldots, N, && \text{(4.5b)} \\
& \dot{\mathbf{q}}_{\text{min}} \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}_{\text{max}}, && \text{(4.5c)} \\
& \delta_i \geq 0, \quad i = 1, \ldots, N && \text{(4.5d)}
\end{aligned}$$

Here, the variables $\delta_i \geq 0$ are CBF slack variables that ensure the QP remains feasible when the human-driven and autonomous driven velocity fields temporarily conflict with the safety constraints. A large penalty $\rho = 10^4$ is applied to δ_i^2 so that any relaxation is heavily discouraged and only activated when strictly necessary. In practice, the slacks remain near zero in all experiments, indicating that the safety constraints are effectively preserved while maintaining numerical feasibility of the optimization. This formulation causes the safety filter to interpolate between the two velocity fields within the optimization itself. When the human and autonomous target poses are aligned, $w_{\text{human}}(t) \approx 1$ and the QP solution converges toward the operator's intended motion. As the discrepancy increases, the autonomous component receives greater weight, guiding the manipulator toward the autonomous target pose while still satisfying all CBF constraints such as obstacle avoidance and joint-limit enforcement. This baseline tests whether blending inside the optimization yields improved task performance or smoother arbitration compared with pre-filter blending (Baseline 2), while maintaining strict geometric safety.

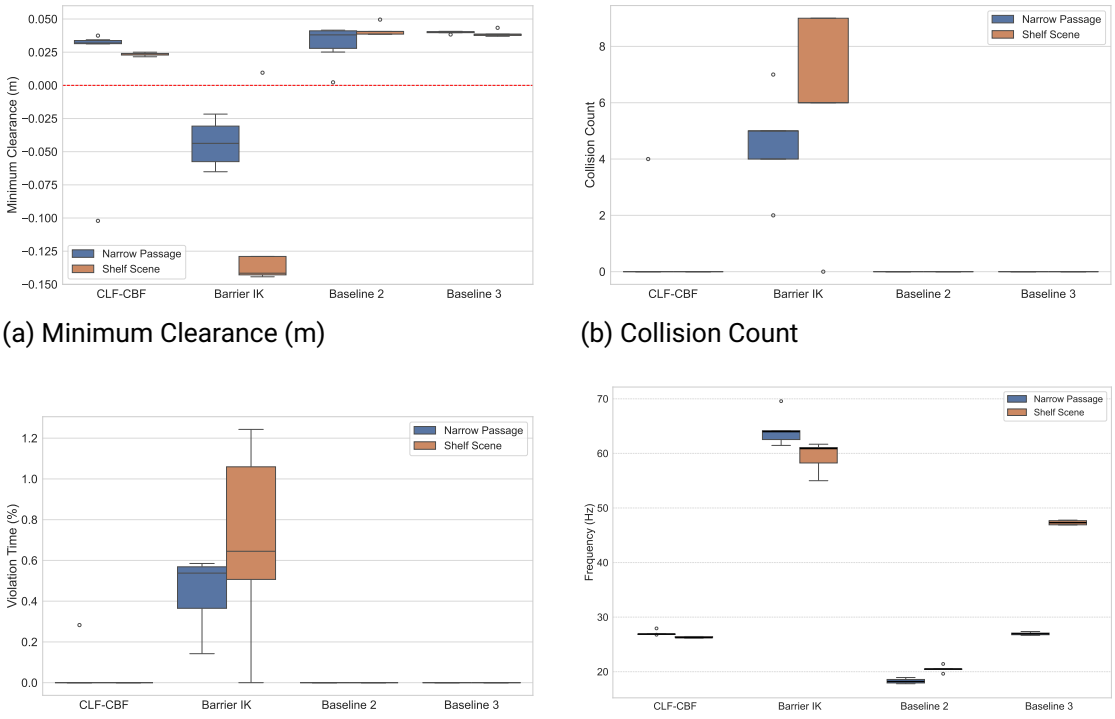
4.3 Results

We evaluate the proposed shared-autonomy CLF–CBF controller in two types of experiments that together show its safety properties and its effect on teleoperation. Taken together, the reference trajectories describe the controller's baseline behavior under repeatable conditions, while the user study shows its practical benefit during real teleoperation.

4.3.1 Controller Behavior Under Reference Trajectories

We use several simple and common metrics to compare the controllers. Minimum clearance, collision count, and violation time describe the safety of the motion. Controller frequency shows how fast each method runs in practice. Completion time measures how long the robot needs to follow the reference motion. Smoothness is computed from the joint jerk and reflects how continuous the motion is. Lower jerk means smoother trajectories.

Minimum clearance. Figure 4.2a reports the minimum robot–obstacle clearance for all controllers in both environments. Minimum clearance measures the smallest signed distance between the robot and any obstacle during the motion. Positive values indicate that the robot remains outside the obstacle geometry. The CLF–CBF controller keeps a



(a) Minimum Clearance (m)

(b) Collision Count

(c) Violation Time (%)

(d) Control Frequency (Hz)

Figure 4.2: Safety and performance comparison of all four controllers across both teleoperation scenarios. **(Top-left)** Minimum robot–obstacle clearance, where higher values indicate safer operation and a larger buffer to the environment. **(Top-right)** Total number of physical contacts detected during execution, with lower values indicating better obstacle avoidance. **(Bottom-left)** Percentage of time during which the CBF constraints were violated; a lower value reflects stronger safety preservation and better adherence to the barrier conditions. **(Bottom-right)** Control update frequency, reflecting the computational efficiency of each controller. Higher values enable smoother tracking and more responsive interaction.

positive clearance in every trial, with values between roughly 0.02 m and 0.04 m in both the narrow passage and the shelf scene. The variation across trials is small, indicating that the controller reacts in a consistent and predictable way as the robot approaches obstacles.

BarrierIK shows a very different result. In the narrow passage, the minimum clearance is negative in all trials, with typical values between -0.04 m and -0.07 m and occasional outliers close to -0.10 m. In the shelf scene, the clearances are even lower, falling between approximately -0.12 m and -0.14 m for every run. This means that the arm repeatedly enters the obstacle geometry in both scenarios and does so quite deeply in the shelf environment.

Baseline 2 and Baseline 3 maintain positive clearances throughout. Their values fall in the range of 0.02 m to 0.05 m, and the spread across trials is small. The minimum-clearance values of Baseline 2 and 3 remain comparable across the two environments.

Violation time. Figure 4.2c shows the percentage of the trajectory during which each controller violates at least one CBF constraint. Violation time reports the percentage of the trajectory during which any CBF constraint is violated, even if the robot does not enter the obstacle. For the CLF–CBF controller, the values in both environments are essentially zero. Apart from a single outlier in the narrow-passage scene, all trials lie exactly on the lower axis, indicating that the controller is able to maintain feasible CBF constraints at every time step.

BarrierIK shows nonzero violation times in both scenes. In the narrow passage, the values typically lie between about 0.15% and 0.6%, with a median slightly above 0.5%. In the shelf scene, the variation is larger: some runs show no violation, while others reach values above 1%. This spread reflects situations in which the robot moves close to several obstacles and the controller cannot maintain all constraints simultaneously.

Baseline 2 and Baseline 3 remain at or extremely close to zero across all trials. Their OSCBF safety constraints still remain feasible and no CBF condition is violated along the trajectory.

Collision count. Figure 4.2b reports the total number of collisions recorded in each reference trajectory. Collision count records the number of time steps in which this distance becomes negative, meaning that the robot has made physical contact with an obstacle. The CLF–CBF controller produces zero collisions in both environments. All trials

lie exactly at zero, which is consistent with the positive minimum clearances and zero violation times reported earlier.

BarrierIK shows repeated collisions in both scenes. In the narrow passage, most runs fall between four and five collisions, with an occasional outlier slightly lower. In the shelf scene, the collision count is higher, typically between six and nine per trial. These values match the negative minimum-clearance values observed for BarrierIK in Fig. 4.2a, where the robot frequently moves inside the obstacle geometry.

Baseline 2 and Baseline 3 show zero collisions in all trials for both environments. The reference trajectory in this experiment does not push them into contact, and their minimum-clearance values remain slightly positive throughout.

Controller frequency. Figure 4.2d shows the mean and standard deviation of the controller update frequency for the four methods in both environments. Baseline 2 runs at the lowest frequency, with average values around 18–$20\,\mathrm{Hz}$ in both scenes. The spread across trials is small, and the two environments look very similar for this method.

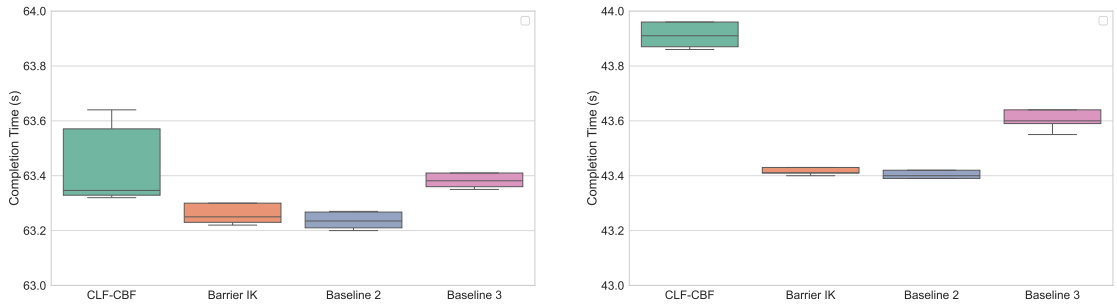
The CLF–CBF controller runs slightly faster, reaching about 25–$30\,\mathrm{Hz}$ in both environments. Its variability is also limited, and the two scenarios show nearly the same behavior. These rates reflect the computational cost of solving a larger QP with multiple safety constraints.

BarrierIK achieves the highest average frequency, with means near $60\,\mathrm{Hz}$ in both scenes. However, its variance is noticeably larger: the whiskers extend from roughly $40\,\mathrm{Hz}$ up to almost $90\,\mathrm{Hz}$, indicating that its update rate varies considerably between runs.

Baseline 3 falls between these extremes. In the narrow passage it averages around $30\,\mathrm{Hz}$, and in the shelf scene the mean increases to about 45–$50\,\mathrm{Hz}$. The spread is moderate in both environments.

Overall, the four methods differ clearly in update rate: Baseline 2 is the slowest, CLF–CBF occupies a middle range with stable performance, BarrierIK is the fastest but also the most variable, and Baseline 3 shows intermediate behavior that depends on the scene.

Completion Time Figure 4.3 reports the completion times for all controllers. In the Narrow Passage scene, all controllers complete the motion within a tight interval of roughly 63.2–$63.6\,\mathrm{s}$. BarrierIK and the two baselines are slightly faster and show very small variation across trials. The CLF–CBF controller shows a somewhat higher mean and a larger variation, so its runs are on average a bit longer than those of the other methods.



(a) Narrow Passage (b) Shelf Scene

Figure 4.3: Task completion times for the four controllers in the reference-trajectory evaluation. Error bars indicate variation across repeated trials. Across both environments, the overall completion times remain relatively similar among the four controllers.

The Shelf Scene shows a similar pattern. Completion times remain close across all methods approximately 43.4–44.0 s, but CLF–CBF again has the highest average duration. Baseline 2 is the fastest in this environment, with BarrierIK and Baseline 3 falling in between. The differences remain small compared to the absolute duration of the task.

Across the predefined-motion experiments, the CLF–CBF controller generally exhibits slightly longer completion times than the baselines. This behavior is consistent with the role of the CLF term, which encourages gradual reduction of the task error and avoids overly aggressive corrections. As a result, the controller tends to produce more controlled and well-regulated motions, even if this leads to marginally slower execution.

As discussed later in the smoothness analysis, this effect is most evident in the reference-trajectory experiments, where the end-effector follows a noise-free motion. Under these ideal conditions, the CLF term has a clear regularizing effect on the controller.

Smoothness Figure 4.4 shows the smoothness results. The CLF–CBF controller has the lowest jerk in both environments. Its motion stays smooth even in the shelf scene, where the robot must move in a tight space.

BarrierIK has much higher jerk in both scenes. The robot makes faster and more abrupt changes in its motion, which explains the large values in the plot.

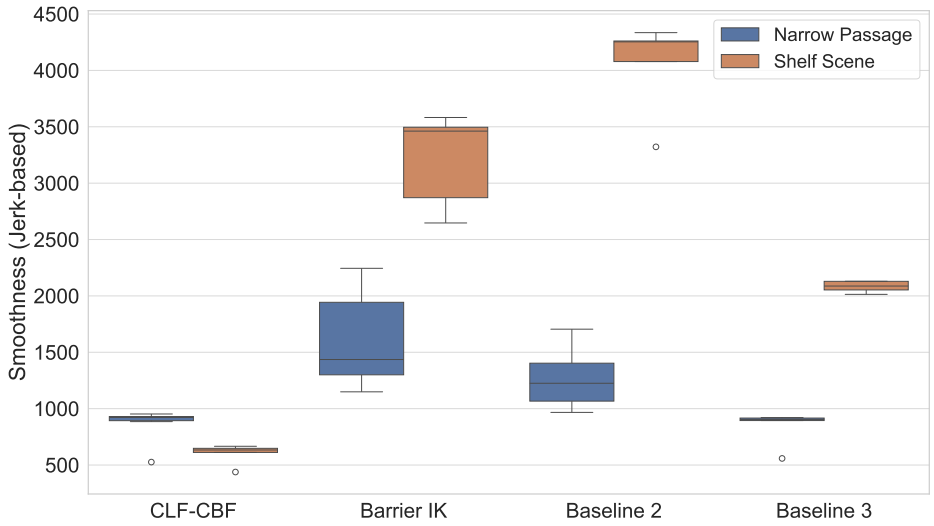


Figure 4.4: Smoothness comparison across controllers in the Narrow Passage and Shelf Scene environments. Lower values indicate smoother motion.

The two baseline controllers behave well in the narrow passage. Baseline 3 is the best one in this easy scene and produces very smooth motion. In the shelf scene, however, both baselines become much less smooth. When the robot moves close to the small opening, they make late corrections, and this increases the jerk.

Overall, the results suggest that CLF–CBF keeps smooth motion in both environments, while the other methods lose smoothness when the task becomes more difficult.

Statistical analysis. To complement the descriptive comparisons in Fig. 4.2, we conducted non-parametric pairwise Wilcoxon rank–sum tests on the reference-trajectory data. These tests are appropriate for the small number of repeated trials and do not rely on Gaussian assumptions.

For minimum clearance, the tests indicate a strong trend that CLF–CBF, Baseline 2, and Baseline 3 maintain consistently positive distances, whereas BarrierIK yields significantly lower values ($p < 0.01$). For violation time and collision count, BarrierIK again differs markedly from the other controllers ($p < 0.01$), while differences among the remaining three methods are small and not statistically conclusive. Control frequency shows significant differences across several controller pairs: Baseline 2 runs at a substantially lower

update rate ($p < 0.01$), BarrierIK at the highest, and CLF–CBF and Baseline 3 fall in the intermediate range.

Although the sample size prevents firm statistical claims, the tests support the overall tendencies visible in the plots: BarrierIK systematically violates safety-related metrics, Baseline 2 is primarily distinguished by its low update rate, and CLF–CBF maintains strong and consistent safety performance across environments.

4.3.2 Objective Metrics from the User Study

We recruited four participants for the user study. All participants were male and between 25 and 28 years old. A trial was counted as a failure if the participant could not grasp the target object within the allowed time; otherwise, the trial was marked as successful.

While the sample size is limited and the numerical results should be interpreted as indicative rather than statistically conclusive, the collected data still provides clear trends regarding how the controllers behave under realistic human-operated conditions.

Failure Rate Figure 4.5 reports the failure rates of all controllers in the three user-study environments. In the simple two-obstacle scene, all controllers show low failure rates, with values around 10–20%. In the four-obstacle narrow passage, the failure rate increases for every method. CLF–CBF remains below 20%, while BarrierIK and Baseline 2 reach around 30–35%. Baseline 3 also shows a clear increase in this environment.

In the shelf scene, CLF–CBF again achieves the lowest failure rate, remaining below 10%. BarrierIK and Baseline 3 show moderate failure rates around 20–35%, while Baseline 2 exhibits the highest failure rate, close to 40%. Overall, the results indicate a trend that CLF–CBF tends to perform more consistently across the three environments, while the other controllers tend to exhibit increased failures as the task becomes more difficult.

These failure patterns are also influenced by the nature of human-operated grasping. During the fine-alignment phase of the task, users often make small corrections, hesitate, or overshoot slightly when positioning the gripper near the cube. Such small inaccuracies can accumulate, especially in the shelf scene where the opening is tight. If the gripper drifts away from the optimal approach direction, users often need to reposition and try again, and a second attempt is usually more difficult. As observed during the study, these moments of human-induced jitter or hesitation frequently lead to unsuccessful grasps. Controllers that offer stronger stabilization toward the autonomous target, for example,

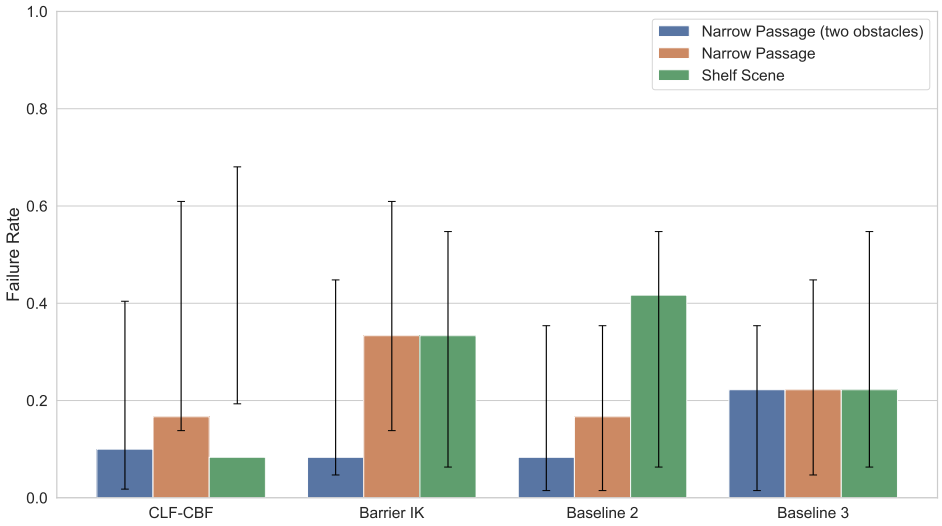
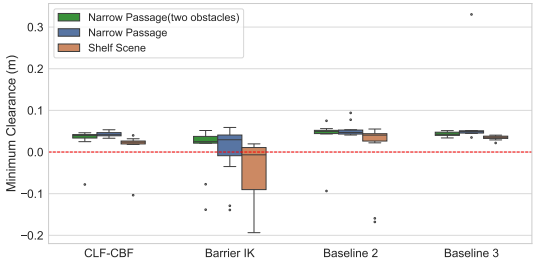


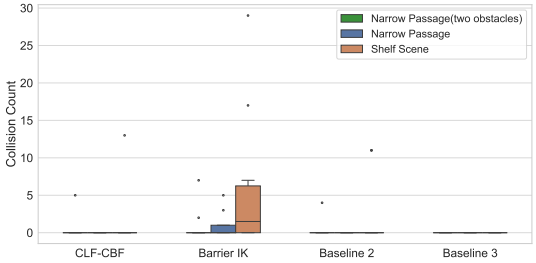
Figure 4.5: Failure rate across all controllers and environments in the user study. Each bar represents the proportion of failed trials (12 trials per controller per scene), and the error bars indicate the Wilson confidence interval for a binomial proportion. CLF–CBF achieves consistently low failure rates across all scenes, while BarrierIK shows noticeably higher failure rates in cluttered environments due to its weaker safety guarantees. Baseline 2 exhibits the highest failure rate in the shelf scene, reflecting the controller's latency, which often forced users to wait for the end-effector to catch up to the target and frequently led to unsuccessful grasps. Baseline 3 performs moderately across scenes, with variability but lower failure rates than BarrierIK and Baseline 2 in the most constrained setting.

CLF–CBF are better able to absorb these small user mistakes, which helps explain their lower failure rates.

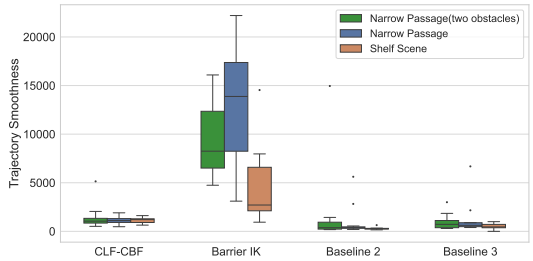
Minimum clearance. Figure 4.6a shows the minimum robot–obstacle clearance measured during the user-study trials. In all three environments, CLF–CBF, Baseline 2, and Baseline 3 keep a positive clearance in every trial, so users do not drive the robot into the obstacles. Among these three methods, CLF–CBF tends to keep a slightly larger and more consistent margin, especially in the shelf scene where the space around the target is tight. Baseline 2 and Baseline 3 remain safe as well, but their clearance values lie closer to the



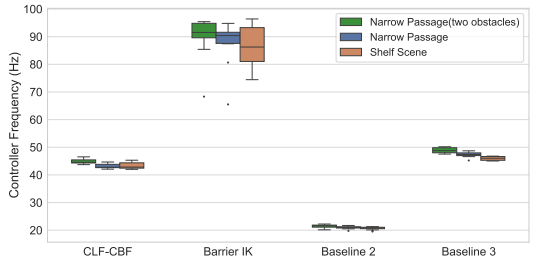
(a) Minimum Clearance (m)



(b) Collision Count



(c) Smoothness (jerk-based metric)



(d) Control Frequency (Hz)

Figure 4.6: Safety and performance results from the user study across all controllers. Top-left: minimum robot–obstacle clearance (higher is better). Top-right: number of collisions across user trials (lower is better). Bottom-left: trajectory smoothness quantified using a jerk-based metric (lower indicates smoother motion). Bottom-right: controller update frequency measured during execution.

zero line, in particular in the four-obstacle narrow passage and the shelf environment.

BarrierIK is the only controller that produces negative clearances. In both narrow-passage scenes, several trials cross the zero threshold, and in the shelf scene the minimum clearance can drop well below -0.1 m. This pattern suggests that the BarrierIK formulation itself is not able to guarantee a strict distance from the obstacles in cluttered layouts.

Compared with the reference experiments, the user-study results show slightly larger spread, which is expected when humans generate the target motion. Nevertheless, all three non–BarrierIK methods maintain a positive safety margin throughout, with CLF–CBF providing the most comfortable clearance.

Collision count. Figure 4.6b shows the number of collisions recorded in the user study. CLF–CBF, Baseline 2, and Baseline 3 all remain collision-free across every environment.

BarrierIK is the only controller that produces repeated collisions. When the workspace becomes tight, particularly in the shelf scene, its safety constraints cannot prevent the robot from entering the obstacle geometry, and several trials show multiple contacts. This reflects a limitation of the method itself rather than user behavior.

Overall, the collision results show that all controllers except BarrierIK avoid collisions in the user study, with CLF–CBF, Baseline 2 and Baseline 3 providing the strong safety guarantees.

Smoothness Figure 4.6c reports the smoothness of the executed trajectories in the user study. Across all three environments, CLF–CBF, Baseline 2, and Baseline 3 achieve very similar jerk levels. Their distributions overlap closely, and the overall variation across trials is small, suggesting that all three controllers are able to keep the motion reasonably smooth under human teleoperation. BarrierIK, on the other hand, shows clearly higher jerk values and a much wider spread, indicating that its outputs react more abruptly when users make small corrections or adjust the target pose near constrained regions.

Compared with the reference trajectory evaluation in Fig. 4.4, the difference between the controllers becomes smaller in the user study (Fig. 4.6c). In the automated setting, the reference trajectory is smooth and noise-free, so the jerk is determined almost entirely by the controller itself, making the smoothness gap between CLF–CBF and the baselines more visible. During teleoperation, however, users introduce small oscillations, brief hesitations, and fine adjustments near the grasp region. These input perturbations appear in the executed motion for all controllers and reduce the contrast between their jerk

values. Despite this reduced gap, BarrierIK remains the least smooth in both evaluations, confirming that its poor smoothness is due to the method itself rather than the presence of human input.

Control frequency. Figure 4.6d shows the controller update frequency measured in the user study. The results closely match the automated experiments: BarrierIK runs near 90–100 Hz, CLF–CBF operates around 45–50 Hz, Baseline 3 around 50 Hz, and Baseline 2 around 20 Hz.

Importantly, the presence of human input does not noticeably affect the frequency. Although users introduce small oscillations or irregular motion, these variations only change the target pose and do not modify the structure or size of the underlying optimization problems. As a result, the computational load remains almost the same as in the reference trajectory evaluations.

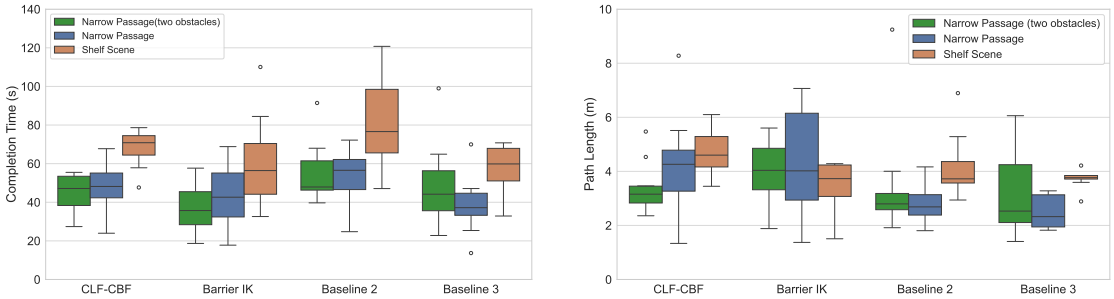
Overall, the control frequency is determined primarily by the controller design itself rather than by user behavior, which explains the close agreement between the automated and user-study results.

Baseline 2 operates at a consistently low update rate, far below the rate of CLF–CBF and Baseline 3. This slow feedback rate directly limits how quickly the robot can respond to changes in the human target. Even when users move smoothly or make only small adjustments, the arm updates too infrequently to follow these motions in real time, which aligns with participants' descriptions of Baseline 2 feeling noticeably delayed.

Figure 4.7 provides two complementary task-level metrics from the user study: completion time and executed end-effector path length. Taken together, they offer clear evidence of how controller latency affects human–robot teleoperation performance.

The completion-time results (Fig. 4.7a) show that Baseline 2 consistently requires substantially longer time to finish the task across all environments. In contrast, CLF–CBF, BarrierIK, and Baseline 3 complete the same trials noticeably faster. This behavior directly reflects the low control-update rate of Baseline 2 (Fig. 4.6d): because the robot updates its motion at only around 20 Hz, it approaches the human-specified target pose more slowly, forcing users to hold their input still and wait for the robot to catch up.

Importantly, the path-length metric (Fig. 4.7b) confirms that this increased execution time is not the result of users making excessive corrections or moving along unnecessarily long or oscillatory trajectories. Baseline 2 produces path lengths that are comparable to the other controllers, indicating that users generally follow similar spatial motions regardless



(a) Completion Time (s)

(b) End-Effector Path Length (m)

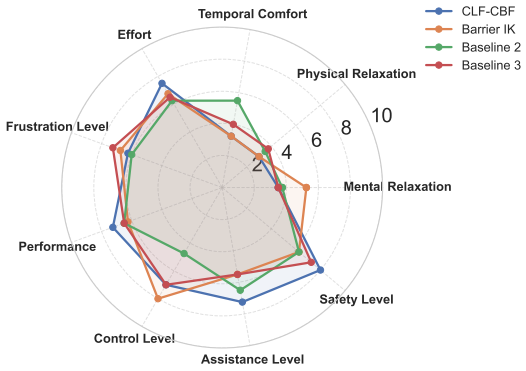
Figure 4.7: Objective performance metrics from the user study. Left: task completion time per controller and scene. Right: cumulative end-effector path length computed from the executed motion. Together, these metrics illustrate how control latency affects the task-level behavior, particularly for Baseline 2, which shows long completion times despite path lengths comparable to the other methods.

of the control method. The difference lies in how quickly the robot can respond to these motions.

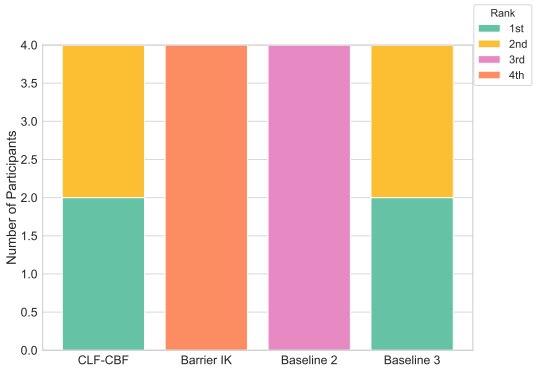
The combination of normal path length and significantly longer completion time is therefore characteristic of a controller with noticeable latency. Users are not traveling farther. They are simply waiting longer. These results align closely with the subjective reports describing Baseline 2 as “not following the hand” and confirm that its reduced responsiveness is the main factor behind its poorer task-level performance.

4.3.3 Subjective Metrics from the User Study

Figure 4.8a and Figure 4.8b summarize the subjective evaluation collected from the four study participants. To ensure consistency across metrics, all subjective scores are presented such that higher values indicate more desirable user experience. This corresponds to a reversed interpretation of several NASA–TLX dimensions: higher ratings on *Mental Relaxation*, *Physical Relaxation*, or *Frustration Level* indicate lower perceived workload. Meanwhile, higher scores on *Safety Level*, *Assistance Level*, *Control Level*, and *Performance* reflect a stronger sense of confidence, support, and controllability during teleoperation.



(a) Subjective scores across controllers (NASA-TLX).



(b) User ranking of the four controllers.

Figure 4.8: Subjective evaluation results from the user study. The radar plot (left) summarizes participant ratings across seven subjective dimensions, where all axes are oriented such that **higher scores indicate more desirable outcomes** (e.g., lower workload, higher perceived safety, and stronger sense of control). The ranking plot (right) shows the final preference ordering provided by each participant.

Consequently, across all axes of the radar plot, larger values consistently represent more desirable outcomes.

Overall, CLF–CBF and Baseline 3 obtain the most positive subjective evaluations. Across dimensions such as perceived safety, assistance level, control level, and overall performance, both controllers receive consistently high ratings. Participants frequently described these two methods as stable, predictable, and reliable, especially when maneuvering near obstacles or performing fine alignment tasks.

In contrast, BarrierIK, while highly responsive and providing strong direct control, receives noticeably lower scores in several dimensions—most prominently in perceived safety and frustration. Users noted that although the method reacts quickly, it tends to become mentally demanding in cluttered scenes, requiring greater attention and effort to maintain precise motion.

Baseline 2 receives the lowest overall subjective ratings. Participants consistently reported a clear delay and reduced responsiveness, often describing the controller as "not following the hand." This latency made accurate positioning near the target more challenging and increased both cognitive and physical workload.

These impressions are reflected in the final participant rankings (Fig. 4.8b). Most participants ranked CLF–CBF or Baseline 3 as their top choice, whereas BarrierIK and Baseline 2 appeared predominantly in the lowest positions. Although the sample size of four participants does not allow for statistically conclusive statements, the collected data reveal several consistent tendencies across controllers. Baseline 2 achieves the highest smoothness among all methods, but its update rate of around 20 Hz appears to introduce noticeable latency. Participants frequently reported that the controller was slow to react, making precise adjustments more difficult during fine manipulation.

In contrast, CLF–CBF provides strong assistance and safety, which many participants perceived as helpful when approaching cluttered regions. However, its stabilizing behavior also requires users to maintain attention, as the controller continuously guides the motion toward safer configurations.

Baseline 3 was generally described as the easiest to control. Participants noted that its responses felt intuitive and predictable, although this ease of use comes with slightly reduced safety margins compared to CLF–CBF. Together, these observations indicate a trend suggesting that each controller offers a different balance between responsiveness, assistance, and safety, which directly shapes the users' subjective experience during teleoperation.

5 Conclusion

5.1 Conclusion

This thesis presented a shared-autonomy control framework that combines a Control Lyapunov Function with Control Barrier Functions for safe teleoperation of a Franka Emika Panda robot. The objective was to support the operator during reach-to-grasp tasks while preserving strict geometric safety in cluttered workspaces.

The evaluation consisted of two complementary parts. The predefined-motion experiments provided repeatable conditions for assessing the intrinsic behavior of each controller. Under these conditions, the CLF–CBF controller maintained positive clearance, avoided collisions, and satisfied all safety constraints across both narrow-passage and shelf environments. BarrierIK, while computationally efficient and responsive, consistently entered obstacle geometry in cluttered scenes due to its configuration-level formulation. Baseline 2 and Baseline 3 preserved safety but showed smaller clearance margins, with Baseline 2 running at a noticeably lower frequency.

The user study introduced human variability into the evaluation. Participants performed grasping tasks in three environments of increasing difficulty. In this setting, humans naturally produced small oscillations or hesitations during fine alignment. The CLF–CBF controller handled these variations reliably and maintained safe operation while guiding the motion toward the helper pose. It achieved the lowest failure rate in the narrow and shelf scenes and preserved consistent safety margins. Baseline 3 also performed well and was frequently described as smooth and easy to use.

An important observation is that BarrierIK, despite its limited safety behavior, was consistently rated as the most responsive and easiest to control. Participants reported that it followed their motions closely and felt the most “direct.” However, this responsiveness came at the cost of reduced safety: BarrierIK produced negative clearance and repeated

collisions in both the objective and user-study evaluations. Baseline 2, although safe, suffered from noticeable latency due to its low update frequency.

Taken together, these findings show that CLF–CBF offers a balanced combination of safety, robustness, and shared-autonomy assistance. It preserves geometric safety in all tested environments, helps stabilize the end-effector during grasping, and improves task success in constrained layouts. The results further highlight that responsiveness alone does not guarantee safe teleoperation, and that structured shared autonomy combined with CBF-based safety can provide a more dependable experience in cluttered environments.

5.2 Future Work

Several directions remain for future investigation. First, although the proposed CLF–CBF controller demonstrates strong performance in obstacle-rich teleoperation tasks, the current formulation still relies on manually tuned penalties and relaxation parameters. Learning these quantities from demonstrations or adapting them online based on user intent, task context, or workload could further improve both efficiency and predictability. In particular, improving the tuning of the CLF-related weights may help reduce the mental and physical demand reported by some participants, enabling a more effortless interaction with the shared-autonomy system.

Another important limitation arises during fine manipulation or precise alignment. Participants noted that the controller provides insufficient assistance when very small or delicate adjustments are required. A promising direction for future research is therefore to incorporate higher-level strategies, such as skill learning, task primitives, or intent prediction, that can be integrated directly into the CLF objective. Such mechanisms may allow the controller to provide more anticipatory and context-aware guidance, ultimately improving the user experience in subtle, high-precision operations.

A broader user study is also an important next step. Because the present evaluation involved only four participants, the results reveal tendencies rather than statistically conclusive effects. A larger and more diverse participant population would enable a more systematic analysis of user preferences, workload, and perceived safety, and would support a more nuanced understanding of how different control strategies influence the teleoperation experience.

In addition, the relationship between controller smoothness and perceived control remains an open question. Although Baseline 3 was consistently rated as easy to control, its safety

margin was lower than that of CLF–CBF; conversely, CLF–CBF achieved strong safety and stability but required more deliberate motion from the user. A deeper investigation into this smoothness–assistance–safety trade-off, particularly between Baseline 3 and CLF–CBF, may help identify design principles for balancing responsiveness with robustness in shared-autonomy control.

Finally, deploying the method on a physical robot will allow us to study latency, sensor noise, and contact uncertainties that are not captured in simulation. Extending the framework to more complex tasks, such as multi-step manipulation, dynamic obstacle avoidance, or contact-rich behaviors, and exploring personalization based on user-specific preferences or risk sensitivity also represent promising directions for future work.

Bibliography

[1] S. Lichiardopol. *A survey on teleoperation*. English. DCT rapporten. DCT 2007.155. Technische Universiteit Eindhoven, 2007.

[2] Jean Vertut. *Teleoperation and robotics: applications and technology*. Vol. 3. Springer Science & Business Media, 2013.

[3] S Wibowo et al. "Improving teleoperation robots performance by eliminating view limit using 360 camera and enhancing the immersive experience utilizing VR headset". In: *IOP Conference Series: Materials Science and Engineering* 1073.1 (Feb. 2021), p. 012037. DOI: 10.1088/1757-899X/1073/1/012037. URL: https://doi.org/10.1088/1757-899X/1073/1/012037.

[4] Y. Yokokohji, T. Imaida, and T. Yoshikawa. "Bilateral teleoperation under time-varying communication delay". In: *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*. Vol. 3. 1999, 1854–1859 vol.3. DOI: 10.1109/IROS.1999.811748.

[5] J.W. Crandall and M.A. Goodrich. "Characterizing efficiency of human robot interaction: a case study of shared-control teleoperation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 2. 2002, 1290–1295 vol.2. DOI: 10.1109/IRDS.2002.1043932.

[6] J. Kofman et al. "Teleoperation of a robot manipulator using a vision-based human-robot interface". In: *IEEE Transactions on Industrial Electronics* 52.5 (2005), pp. 1206–1219. DOI: 10.1109/TIE.2005.855696.

[7] Tsung-Chi Lin, Achyuthan Unni Krishnan, and Zhi Li. "Shared Autonomous Interface for Reducing Physical Effort in Robot Teleoperation via Human Motion Mapping". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9157–9163. DOI: 10.1109/ICRA40945.2020.9197220.

[8] Anca Dragan and Siddhartha Srinivasa. “A Policy Blending Formalism for Shared Control”. In: *International Journal of Robotics Research* 32.7 (June 2013), pp. 790–805.

[9] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. “Shared autonomy via probabilistic inference”. In: *The International Journal of Robotics Research* 37.7 (2018), pp. 718–742.

[10] Mingzhang Pan et al. “Collision risk assessment and automatic obstacle avoidance strategy for teleoperation robots”. In: *Computers & Industrial Engineering* 169 (2022), p. 108275. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2022.108275. URL: https://www.sciencedirect.com/science/article/pii/S0360835222003424.

[11] Keshav Chintamani et al. “Systematic Tele-operation with Augmented Reality Path Planned Navigation Cues in Cluttered Environments”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 53.1 (2009), pp. 81–85. DOI: 10.1177/154193120905300118. eprint: https://doi.org/10.1177/154193120905300118. URL: https://doi.org/10.1177/154193120905300118.

[12] Dingjiang Zhou and Mac Schwager. “Assistive collision avoidance for quadrotor swarm teleoperation”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1249–1254. DOI: 10.1109/ICRA.2016.7487256.

[13] Domenico Tommasino et al. “Effect of End-Effector Compliance on Collisions in Robotic Teleoperation”. In: *Applied Sciences* 10.24 (2020). ISSN: 2076-3417. DOI: 10.3390/app10249077. URL: https://www.mdpi.com/2076-3417/10/24/9077.

[14] Shervin Javdani et al. *Shared Autonomy via Hindsight Optimization for Teleoperation and Teaming*. 2017. arXiv: 1706.00155 [cs.RO]. URL: https://arxiv.org/abs/1706.00155.

[15] Umur Atan, Varun R. Bharadwaj, and Chao Jiang. “Assistive Control of Robot Arms via Adaptive Shared Autonomy”. In: *2024 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. 2024, pp. 1096–1102. DOI: 10.1109/AIM55361.2024.10637239.

[16] Michael Bowman and Xiaoli Zhang. “Dimension-Specific Shared Autonomy for Handling Disagreement in Telemanipulation”. In: *IEEE Robotics and Automation Letters* 8.3 (2023), pp. 1415–1422. DOI: 10.1109/LRA.2023.3239313.

[17] Rolif Lima et al. “Augmenting Robot Teleoperation with Shared Autonomy via Model Predictive Control”. In: *2024 IEEE Conference on Telepresence*. 2024, pp. 42–48. DOI: 10.1109/Telepresence63209.2024.10841639.

[18] Mela Coffey and Alyssa Pierson. “Collaborative Teleoperation with Haptic Feedback for Collision-Free Navigation of Ground Robots”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 8141–8148. DOI: 10.1109/IROS47612.2022.9981426.

[19] Idil Ozdamar et al. “A Shared Autonomy Reconfigurable Control Framework for Telemanipulation of Multi-Arm Systems”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 9937–9944. DOI: 10.1109/LRA.2022.3191200.

[20] O. Khatib. “Real-time obstacle avoidance for manipulators and mobile robots”. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

[21] E. Rimon and D.E. Koditschek. “Exact robot navigation using artificial potential functions”. In: *IEEE Transactions on Robotics and Automation* 8.5 (1992), pp. 501–518. DOI: 10.1109/70.163777.

[22] Christoph Rösmann et al. “Efficient trajectory optimization using a sparse model”. In: *2013 European Conference on Mobile Robots*. 2013, pp. 138–143. DOI: 10.1109/ECMR.2013.6698833.

[23] Daniel Rakita et al. “CollisionIK: A Per-Instant Pose Optimization Method for Generating Robot Motions with Environment Collision Avoidance”. In: *arXiv preprint arXiv:2102.13187* (2021).

[24] Sami Haddadin et al. “Collision detection and reaction: A contribution to safe physical human-robot interaction”. English. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS ; Conference date: 22-09-2008 Through 26-09-2008. 2008, pp. 3356–3363. ISBN: 9781424420582. DOI: 10.1109/IROS.2008.4650764.

[25] Aaron D. Ames et al. “Control Barrier Functions: Theory and Applications”. In: *2019 18th European Control Conference (ECC)*. 2019, pp. 3420–3431. DOI: 10.23919/ECC.2019.8796030.

[26] Vipul K. Sharma et al. “Safe Human–Robot Collaboration With Risk Tunable Control Barrier Functions”. In: *IEEE/ASME Transactions on Mechatronics* 30.4 (2025), pp. 3133–3141. DOI: 10.1109/TMECH.2025.3572047.

[27] Carl D. Crane and Joseph Duffy. *Kinematic Analysis of Robot Manipulators*. USA: Cambridge University Press, 1998. ISBN: 0521570638.

[28] Robert J. Schilling. *Fundamentals of Robotics: Analysis and Control*. 1st. Simon & Schuster Trade, 1996. ISBN: 0133444333.

[29] J.J. Craig. *Introduction to Robotics, Global Edition*. Pearson Education, 2021. ISBN: 9781292164953. URL: https://books.google.de/books?id=Bjw1EAAAQBAJ.

[30] Serdar Kucuk and Zafer Bingul. *Robot kinematics: Forward and inverse kinematics*. Vol. 1. INTECH Open Access Publisher London, UK, 2006.

[31] B. Siciliano et al. *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2010. ISBN: 9781846286414. URL: https://books.google.de/books?id=jPCAFmE-logC.

[32] O. Khatib. "A unified approach for motion and force control of robot manipulators: The operational space formulation". In: *IEEE Journal on Robotics and Automation* 3.1 (1987), pp. 43–53. DOI: 10.1109/JRA.1987.1087068.

[33] Jun Nakanishi et al. "Operational Space Control: A Theoretical and Empirical Comparison". In: *The International Journal of Robotics Research* 27.6 (2008), pp. 737–757. DOI: 10.1177/0278364908091463. eprint: https://doi.org/10.1177/0278364908091463. URL: https://doi.org/10.1177/0278364908091463.

[34] Daniel E. Whitney. "Resolved Motion Rate Control of Manipulators and Human Prostheses". In: *IEEE Transactions on Man-Machine Systems* 10.2 (1969), pp. 47–53. DOI: 10.1109/TMMS.1969.299896.

[35] D. E. Whitney. "The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators". In: *Journal of Dynamic Systems, Measurement, and Control* 94.4 (Dec. 1972), pp. 303–309. ISSN: 0022-0434. DOI: 10.1115/1.3426611. eprint: https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/94/4/303/5528808/303_1.pdf. URL: https://doi.org/10.1115/1.3426611.

[36] B. Siciliano and J.-J.E. Slotine. "A general framework for managing multiple tasks in highly redundant robotic systems". In: *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*. 1991, 1211–1216 vol.2. DOI: 10.1109/ICAR.1991.240390.

[37] Eduardo D. Sontag. "A Lyapunov-Like Characterization of Asymptotic Controllability". In: *SIAM Journal on Control and Optimization* 21.3 (1983), pp. 462–471. DOI: 10.1137/0321028. eprint: https://doi.org/10.1137/0321028. URL: https://doi.org/10.1137/0321028.

[38] Eduardo D. Sontag. “A ‘universal’ construction of Artstein’s theorem on nonlinear stabilization”. In: *Systems & Control Letters* 13.2 (1989), pp. 117–123. ISSN: 0167-6911. DOI: 10.1016/0167-6911(89)90028-5. URL: https://www.sciencedirect.com/science/article/pii/0167691189900285.

[39] Hassan K Khalil. *Nonlinear systems*. Upper Saddle River, NJ: Prentice-Hall, 2002. URL: https://cds.cern.ch/record/1173048.

[40] Boqian Li et al. “A Survey on the Control Lyapunov Function and Control Barrier Function for Nonlinear-Affine Control Systems”. In: *IEEE/CAA Journal of Automatica Sinica* 10.3 (2023), pp. 584–602. DOI: 10.1109/JAS.2023.123075.

[41] Aaron D. Ames et al. “Control Barrier Function Based Quadratic Programs for Safety Critical Systems”. In: *IEEE Transactions on Automatic Control* 62.8 (2017), pp. 3861–3876. DOI: 10.1109/TAC.2016.2638961.

[42] Aaron D. Ames, Jessy W. Grizzle, and Paulo Tabuada. “Control barrier function based quadratic programs with application to adaptive cruise control”. In: *53rd IEEE Conference on Decision and Control*. 2014, pp. 6271–6278. DOI: 10.1109/CDC.2014.7040372.

[43] Wei Xiao and Calin Belta. “Control Barrier Functions for Systems with High Relative Degree”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 474–479. DOI: 10.1109/CDC40024.2019.9029455.

[44] F. Blanchini. “Set invariance in control”. In: *Automatica* 35.11 (1999), pp. 1747–1767. ISSN: 0005-1098. DOI: https://doi.org/10.1016/S0005-1098(99)00113-2. URL: https://www.sciencedirect.com/science/article/pii/S0005109899001132.

[45] Boqian Li et al. “A Survey on the Control Lyapunov Function and Control Barrier Function for Nonlinear-Affine Control Systems”. In: *IEEE/CAA Journal of Automatica Sinica* 10.3 (2023), pp. 584–602. DOI: 10.1109/JAS.2023.123075.

[46] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. “RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion”. In: *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, June 2018. DOI: 10.15607/RSS.2018.XIV.043.

[47] Daniel Morton and Marco Pavone. “Safe, task-consistent manipulation with operational space control barrier functions”. In: *arXiv preprint arXiv:2503.06736* (2025).

[48] Berk Gueler et al. “A Safety-Aware Shared Autonomy Framework with BarrierIK Using Control Barrier Functions”. Manuscript submitted to the IEEE International Conference on Robotics and Automation (ICRA). 2025.

[49] Katharina Muelling et al. “Autonomy Infused Teleoperation with Application to BCI Manipulation”. In: *Autonomous Robots* 41.6 (Aug. 2017), pp. 1401–1422.

[50] Mario Selvaggio et al. “Autonomy in Physical Human-Robot Interaction: A Brief Survey”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7989–7996. DOI: 10.1109/LRA.2021.3100603.

[51] Ken Shoemake. “Animating rotation with quaternion curves”. In: *SIGGRAPH Comput. Graph.* 19.3 (July 1985), pp. 245–254. ISSN: 0097-8930. DOI: 10.1145/325165.325242. URL: https://doi.org/10.1145/325165.325242.

[52] Hamid Sadeghian et al. “Task-Space Control of Robot Manipulators With Null-Space Compliance”. In: *IEEE Transactions on Robotics* 30.2 (2014), pp. 493–506. DOI: 10.1109/TRO.2013.2291630.

[53] Alexander Dietrich, Christian Ott, and Jaeheung Park. “The Hierarchical Operational Space Formulation: Stability Analysis for the Regulation Case”. In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1120–1127. DOI: 10.1109/LRA.2018.2792154.

[54] Filippo Luca Ferretti et al. *Hardware-Accelerated Morphology Optimization via Physically Consistent Differentiable Simulation*. 2025. URL: https://github.com/ami-iit/jaxsim.

[55] Kevin Tracy, Taylor A. Howell, and Zachary Manchester. *DiffPills: Differentiable Collision Detection for Capsules and Padded Polygons*. 2022. arXiv: 2207.00202 [cs.RO]. URL: https://arxiv.org/abs/2207.00202.