Universität Stuttgart

Adaptive Human-Robot Policy Blending for Shared Control Teleoperation

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Yoojin Oh

aus Cheonan, Republik Korea

Hauptberichter:	Prof. Dr. rer. nat. Marc Toussaint
Mitberichter:	Prof. Dr. Katherine J. Kuchenbecker
Tag der mündlichen Prüfung:	08.03.2024

Institut für Parallele und Verteilte Systeme der Universität Stuttgart

2024

Abstract

Technology is rapidly advancing toward intelligent systems driven by artificial intelligence (AI). Expectations are growing for fully autonomous systems that can assist humans in efficiently completing tasks in various domains, such as autonomous vehicles and mobile robots, industrial robots, medical and assistive robots, service robots, and rescue robots. Existing robotic systems still lack robustness and adaptability when operating in highly dynamic environments. Teleoperation enables the deployment of robots with partial autonomy to execute tasks with the help of human operators. Robots can be more agile, execute highly diverse tasks, and achieve long-term goals with the help of human assistance, but humans are prone to making operational errors when teleoperating robots under pressure. Shared control aims to solve this problem by combining the advantages of both humans and robots, where the robot and human agents support each other to maximize the overall performance of the system. In shared control, it is essential to determine an acceptable blending strategy (arbitration), that combines human and autonomous robot policies to generate an arbitrated action.

This dissertation proposes novel arbitration strategies for robot teleoperation, that focus on maximizing the control authority of the human operator while allowing the robot to assist the human in avoiding operational errors. In the dissertation, it is assumed that a goal-conditioned autonomous robot policy is provided that can generate optimized actions toward a target goal at any state. The methods utilize this autonomous robot policy to infer intrinsic information about the task, to assist the human operator in critical situations. The thesis formulates shared control as an optimization problem to find the blended policy that maximizes the operator's internal action-value function, while maintaining a trust region to keep the policy close to the autonomous robot policy. This leads to a state update rule in the form of natural gradient descent, which emerges from sampling local robot policies and computing the second derivative. In addition, a method that actively allocates more control authority to the human at a decision point is introduced, reasoning on the modality of the marginalized probability distribution of all goal-conditioned autonomous robot policies. The concept is used to train a nonlinear differentiable arbitration strategy using reinforcement learning, by learning a policy through user interaction. Finally, a teleoperation system is presented that combines perception and manipulation solutions using hand gestures as the teleoperation interaction method. The results of the methods presented in this dissertation demonstrate an improvement in safe teleoperation while allowing the operator to retain control authority.

Zusammenfassung

Die Technologie entwickelt sich rasch zu intelligenten Systemen, die durch künstliche Intelligenz (KI) angetrieben werden. Die Erwartungen an vollständig autonome Systeme, die den Menschen bei der effizienten Erledigung von Aufgaben in verschiedenen Bereichen unterstützen können, wie z. B. autonome Fahrzeuge und mobile Roboter, Industrieroboter, medizinische Roboter und Assistenzroboter, Serviceroboter und Rettungsroboter, steigen. Bestehenden Robotersystemen mangelt es noch an Robustheit und Anpassungsfähigkeit beim Einsatz in hochdynamischen Umgebungen. Die Teleoperation ermöglicht den Einsatz von Robotern mit teilweiser Autonomie zur Ausführung von Aufgaben mit Hilfe von menschlichen Steuerungen. Roboter können mit menschlicher Unterstützung agiler sein, sehr unterschiedliche Aufgaben ausführen und langfristige Ziele erreichen, aber Menschen sind anfällig für Bedienungsfehler, wenn sie unter Druck mit Robotern arbeiten. Die gemeinsame Steuerung zielt darauf ab, dieses Problem zu lösen, indem die Vorteile von Menschen und Robotern kombiniert werden, wobei sich die Roboter und die menschlichen Agenten gegenseitig unterstützen, um die Gesamtleistung des Systems zu maximieren. Bei der gemeinsamen Steuerung ist es wichtig, eine annehmbare Mischstrategie (Arbitrierung) zu bestimmen, die die Richtlinien des Menschen und des autonomen Roboters kombiniert, um eine abgestimmte Aktion zu erzeugen.

In dieser Arbeit wird die gemeinsame Steuerung als Optimierungsproblem formuliert, bei dem es darum geht, die gemischte Strategie zu finden, die die interne Aktionswertfunktion des Bedieners maximiert und gleichzeitig einen Vertrauensbereich aufrechterhält, um die Strategie nahe an der autonomen Roboterstrategie zu halten. Dies führt zu einer Zustandsaktualisierungsregel in Form eines natürlichen Gradientenabstiegs, der sich aus dem Abtasten lokaler Roboterstrategien und der Berechnung der zweiten Ableitung ergibt. Darüber hinaus wird eine Methode eingeführt, die dem Menschen an einem Entscheidungspunkt aktiv mehr Kontrollbefugnisse zuweist, indem sie auf der Modalität der marginalisierten Wahrscheinlichkeitsverteilung aller zielbedingten autonomen Roboterstrategien beruht. Das Konzept wird verwendet, um eine nichtlineare differenzierbare Arbitrierungsstrategie mit Hilfe von Reinforcement Learning zu trainieren, die eine Strategie durch Benutzerinteraktion lernt. Schließlich wird ein Teleoperationssystem vorgestellt, das Wahrnehmungs- und Manipulationslösungen unter Verwendung von Handgesten als Interaktionsmethode für die Teleoperation kombiniert. Die Ergebnisse der in dieser Dissertation vorgestellten Methoden zeigen eine Verbesserung der sicheren Teleoperation bei gleichzeitiger Beibehaltung der Steuerungskompetenz des Bedieners.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Dr. Jim Mainprice for not only allowing me to pursue my Ph.D. in Germany under his supervision but also supporting me until the end of my Ph.D. journey. I am grateful for the detailed and constructive guidance and valuable feedback he gave me. His direct involvement in research allowed me to gain fruitful insights as well as hands-on experience in practical programming skills, which I truly value. I would also like to thank Prof. Marc Toussaint for his support with my Ph.D. process and the passion he showed me for science. I enjoyed supporting him as a teaching assistant in his courses and am grateful to gain valuable insights through his teaching.

I am grateful to be a scholar of the International Max Planck Research School for Intelligent Systems (IMPRS-IS) and I would like to thank Dr. Katherine Kuchenbecker and Dr. Jörg Stückler for being a member of my IMPRS-IS thesis committee and for providing me with feedback and guidance. I would especially like to express my gratitude to Dr. Kuchenbecker for allowing me to finish my research in the Haptic Intelligence department at the Max Planck Institute for Intelligent Systems (MPI-IS) and providing me with support and advice on my career.

I am thankful to have met wonderful colleagues and students over the past years. A big thank you to Carola Stahl for helping me with all the administrative tasks that I had to deal with in Germany, and to Philipp Kratzer and Janik Hager who were the last members of the remaining Machine Learning and Robotics group at the University of Stuttgart. Thank you Dr. Jean-Claude Passy for providing technical support and motivating me to complete my last research project. I truly enjoyed working with him and gaining new technical insights during my last months at MPI. Thanks to all my other colleagues for their support.

My appreciation goes to the students I supervised: Hangbeom Kim, Shaowen Wu, Gunjan Gupta, Benedikt Rüther, Tim Bacher, and Likith Rana Vayala. Thank you for bringing your creative insights and helping me conduct experiments and collect data.

Last but not least, I am forever grateful for all the love and support my family and my partner Florian Behrle provided me throughout this journey.

List of Figures

List of Tables

List of Symbols

The symbols frequently used throughout the dissertation are listed below for convenience. They follow a general rule: scalar values are shown in lowercase (e.g., t), vectors in bold lowercase (e.g., $\boldsymbol{x}$), and matrices are shown in bold uppercase letters (e.g., $\boldsymbol{J}$). Some symbols may play a dual role. In this case, the meaning should be clear from the context.

Indices and Scalars

i	Enumeration index
t	Time index
α	Arbitration factor
γ	Discount rate
η	Step size
r	Reward
g	Goal object index
g^*	Human operator's intended goal object index
b	Goal confidence (belief) of the object
G	Return

Vectors

$\boldsymbol{x}$	State in Cartesian space
$\boldsymbol{s}$	System's state
$\boldsymbol{a}$	Action
$\boldsymbol{a}^R$	Autonomous agent's action
$\boldsymbol{a}^H$	Human agent's action
$\boldsymbol{u}$	Arbitrated action that the robot executes, control action (Chapter 3.2.1)
$\boldsymbol{q}$	Joint configuration
$\boldsymbol{p}$	Position
$\boldsymbol{p}_{gripper}$	Robot end-effector position
$\boldsymbol{p}_{goal}$	Goal position
$\boldsymbol{p}_{approach}$	Approaching point when grasping
$\boldsymbol{v}$	Velocity
$\boldsymbol{v}_{gripper}$	Robot end-effector velocity
$\boldsymbol{v}_H$	Human agent's velocity command
$\boldsymbol{a}$	Acceleration
$\boldsymbol{t}$	Translation
$\mathbf{v}_{rot_Z}$	Robot end-effector rotation vector
$\boldsymbol{\theta}$	Parameters

Matrices

$\boldsymbol{I}$	Identity matrix
$\boldsymbol{J}$	Jacobian

$\boldsymbol{J}^{\dagger}$	Jacobian pseudo-inverse
$\boldsymbol{H}$	Hessian matrix
$\boldsymbol{G}$	Riemannian metric tensor
$\boldsymbol{W}$	Riemannian metric in the configuration
$\boldsymbol{F}$	Fisher information matrix
$\boldsymbol{A}$	State transition matrix
$\boldsymbol{B}$	Control multiplier
$\boldsymbol{Q}$	State cost matrix
$\boldsymbol{R}$	Control cost matrix (Chapter 3.2.1), replay buffer (Chapter 6), rotation matrix (Chapter 7)

Others

v_{π}	State value function
v^{*}	Optimal state value function
q_{π}	State-action value function
Q_H	Human agent's internal action-value function
π	Policy
π^{*}	Optimal policy
π_H	Human agent's policy
π_R	Autonomous agent's policy
π_S	Shared control policy
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$\mathcal{U}$	Shared control action space

$\mathcal{R}$	Reward function
$\mathcal{G}$	Goal space
$\mathcal{M}$	Manifold
$\mathcal{Q}$	Configuration manifold
$\mathcal{T}$	State-transition function
V	State log partition function
ξ^{ref}	Reference trajectory
ϕ	Forward kinematics map
μ	Actor network policy
Q	Critic network

Contents

Chapter 1

Introduction

Technology is rapidly advancing towards autonomous systems with Artificial Intelligence (AI). Intelligent robots are integrated increasingly into our daily lives: from semi-autonomous vehicles, surgical robots, social and collaborative robots, to disaster relief and field robots. Such robots can assist or even substitute for people, with their ability to access inhospitable areas, fast computation skills, and precision and strength exceeding human capabilities. Yet, contrary to our anticipation of futuristic and versatile robots, current robotic systems lack resilience in unpredictable scenarios.

By contrast, humans are skilled at interpreting such complex environments and planning comprehensive sequential decisions. While they can provide supervision and guidance to robotic systems, they are prone to making operational mistakes when it comes to controlling robots with low situational awareness. In fact, operator mistakes were the major cause of robot errors during the DARPA Robotics challenge [1].

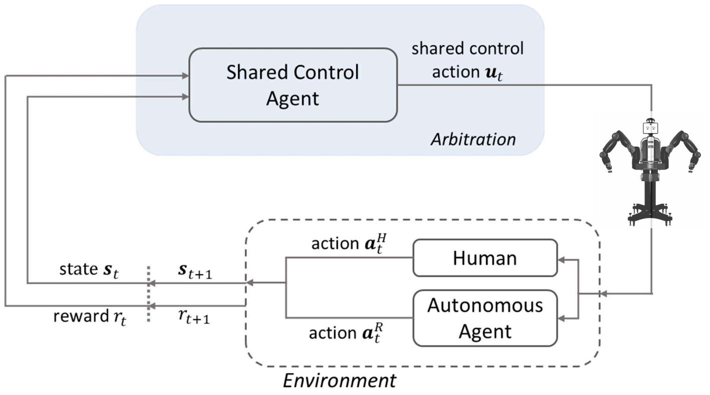


Figure 1.1: Shared control agent interacting with the environment

Shared control is a framework that combines the best of both worlds. It aims to leverage human intelligence and the robot's partial autonomy to enhance the overall performance of the robotic system. As shown in Figure 1.1, the framework consists of three agents: the human agent provides long-term goals, forestalling unreliable actions from the robot; the (semi-) autonomous agent provides optimal solutions for precision and efficiency; and the shared control agent combines the commands from both agents to filter out human mistakes, but efficiently performs the task in the way the human prefers. However, balancing this level of assistance, i.e., arbitration, can be challenging as users exhibit different preferences for robot assistance. Some anticipate active assistance to ensure the execution of optimal behaviors while others feel being in control is more important. According to a user study in [2], when participants were unable to understand the purpose of the automated assistance, there was a decrease in trust and a preference for control over automated aids. Therefore, there is a strong demand for flexible arbitration methods that seamlessly blend both forms of control.

This dissertation attempts to design such adaptable arbitration methods to improve teleoperation performance during robot manipulation tasks. The focus is on allowing the users to retain control authority without risking the safety and stability of the robotic system. Beginning with a description of how to represent a robot agent with optimal behaviors, I present multiple approaches for arbitration. I then introduce a teleoperation system that combines solutions to both perception and manipulation. This work demonstrates that the proposed novel approaches can improve the performance of safe and efficient teleoperation, while still allowing the user to take control.

1.1 Dissertation Outline

The focus of the dissertation is on arbitration methods that combine robot autonomy and human assistance for a human-robot teleoperation system. The following chapters are structured as follows:

Chapter 2: Background provides relevant background knowledge necessary for understanding the techniques described in the dissertation.

Chapter 3: Representing the Semi-Autonomous Agent discusses the teleoperation task addressed in the dissertation and presents how a global policy is obtained that represents the semi-autonomous robot agent.

Chapter 4: Natural Gradient Shared Control presents a method to arbitrate human and robot controls, by observing the local landscape of the vector field that the semi-autonomous agent policy forms, to alter the robot assistance level depending on the complexity of the state. The Fisher information matrix expresses the sensitivity of the policy within a close range, and the human operator's controls are augmented accordingly to provide robot assistance in critical situations.

Chapter 5: Augmenting Human Policies using Riemannian Metrics for Human-Robot Shared Control extends the work of the previous chapter to higher dimensional spaces, integrating human and robot control in different coordinate systems, demonstrating the method using a simulated teleoperation setup allowing interaction via a VR system.

Chapter 6: Learning to Arbitrate using Disagreement Between Robot Sub-policies introduces a method to learn an arbitrated shared control policy using reinforcement learning. The von Mises distribution measures the uncertainty of the states that could lead to a disagreement between the human and the semi-autonomous agent. The reinforcement learning agent then reduces the level of assistance in these areas to give the human operator more control.

Chapter 7: A System for Traded Control Teleoperation Combining Manipulation and Perception proposes a tele-operation system setup using traded control, which is a special form of shared control. State-of-the-art perception methods are combined to allow for the most intuitive form of teleoperation, where commands are given as if the human is executing them.

Chapter 8: Conclusion is a summary of the findings presented throughout the dissertation.

Chapter 2

Background

This chapter provides prior knowledge that is necessary for understanding the theory and techniques described in the dissertation.

2.1 Teleoperation

Teleoperation refers to "operating a system at a distance", where a physical distance or change in scale separates the work being performed from the human operator [3] and the robot/system is operated using human intelligence [4]. Its main function is to assist the operator in accomplishing complex tasks in hazardous and unstructured environments that are difficult to access and to reduce the cost of the mission or safety risks. For this reason, teleoperation systems were initially widely used to operate unmanned vehicles underwater, in space, and for military and defense applications since the 1970s [5].

A teleoperation system consists of a human operator controlling a remote robot using one or more human-robot interface devices (e.g., mouse, keyboard, buttons, joystick, manipulators, etc.). Depending on the system, special controllers [6]–[8] or control stations [5] that consist of multiple interface devices were designed to control the robot while perceiving the robot's status and its surroundings through sensors and visual interfaces.

The design of the teleoperation interface largely impacts the performance of the robot. According to Murphy [9], more than 50 percent of all robot failures at disaster sites are the result of human error. In addition, the DARPA Robotics Challenge (DRC) held from 2012 to 2015 well reflected the recent advances in the development of semi-autonomous robots for disaster or emergency response. The competition focused on building robots that are capable of accomplishing complex mobility and manipulation tasks in a disaster scenario. It was reported that one of the major causes of failures originated from human operator errors [1] and the performance of the robots was heavily dependent on the operator's skill which required extensive training and experience [10]. Atkeson et al. [1] emphasized that designing effective teleoperation interfaces is the most cost-effective research area to improve robot performance and that interfaces should help eliminate errors or mistakes that humans make while controlling the robots under pressure. The main focus of the dissertation is to explore different human-robot interaction methods from the perspective of shared control for teleoperation that increase efficiency and prevent human operator errors.

2.2 Shared Control and Shared Autonomy

Shared control is an intermediary control method in the levels of robot autonomy taxonomy, that ranges from human direct control to fully autonomous systems. It refers to a concept where the human and robot autonomy share control over a system together at the operational level [11]–[13].

According to Musić and Hirche [14], shared control can be classified into the *complementary interaction paradigm* and *overlapping interaction paradigm*. In the *complementary interaction paradigm*, the control is shared at the task level [15]. The human takes on a supervisory role and provides high-level task commands such as managing a team of robots, navigating between points or exploring an area, while subtasks are allocated to the robot's autonomy and the robot supports the human by executing these sub-tasks such as computing low-level control inputs, collision avoidance [16]–[18].

The *overlapping interaction paradigm* directly arbitrates human and robot actions, and control sharing is done at the servo-level. The human and the robot simultaneously provide task-related controls and the resulting control commands are a blend of their control inputs. Since the arbitration is happening more repeatedly compared to the previous paradigm, the human operators must always be engaged in the control loop and the performance of the shared control system is highly influenced by how the control inputs are blended [15]. Nevertheless, it is assumed that the paradigm can be a practical solution when working with semi-autonomous systems with lower autonomy levels or a fallback plan to take over the controls of autonomous systems since it does not involve long-horizon tasks and motion planning with complex tasks. The shared control setting discussed in the dissertation follows the overlapping interaction paradigm.

The term *shared autonomy* emerged as an emphasis on combining humans with an intelligent system equipped with some degree of autonomy. Selvaggio et al. [19] recently distinguished the terms depending on the subject that is in charge of adjusting the level of autonomy. They distinguished shared control, where humans manually tune the assistance levels, from shared autonomy, where the robot regulates its autonomy based on its understanding of human intentions and the environment. The methods presented in the dissertation are more related to shared autonomy based on their definition. However, the term *shared control* is used throughout the dissertation to emphasize the idea of shared authority over the robot, and the authority is assigned dynamically. It is worthwhile to note that both terms are still often used interchangeably in many prior publications.

2.3 Arbitration

The arbitration function $\alpha : (\boldsymbol{a}^H, \boldsymbol{a}^R) \mapsto \boldsymbol{u}$, given user and robot actions $\boldsymbol{a}^H, \boldsymbol{a}^R$, outputs an arbitrated action $\boldsymbol{u}$ that is executed by the robot. The arbitration function can depend on different factors such as confidence in the user intent prediction [20]–[22], or considering the difference between each command [23]. One common form of blending is through a linear combination between the human user and autonomous robot agent policies [20]–[23].

When the robot predicts the user's intent with high confidence, the user often loses *control authority*. This has been reported to generate mixed preferences from users where some users prefer to keep control authority despite longer completion times [24], [25]. Additionally, when assistance is against the user's intention, this approach can aggravate the user's workload [20]; the user "fights" against the assistance rather than gain help from it. Defining an arbitration function that is not too *timid* (i.e., only assists when very confident) nor to *aggressive*, is generally difficult and interpreting the noisy confidence estimate of the intent prediction is error-prone.

2.4 Formulation as a Reinforcement Learning Problem

Humans reason about the long-term consequences of their actions, rather than making myopic decisions, and they utilize their previous experience to improve their future decisions [26]. Their behaviors are considered optimal (or suboptimal); their sensorimotor system is the result of processes that act constantly to enhance behavioral performance and optimal control models have been used to explain biological behaviors based on optimal performance [27]. Human behavior is regarded as an expert who acts optimally (or suboptimally) and is observed to infer the

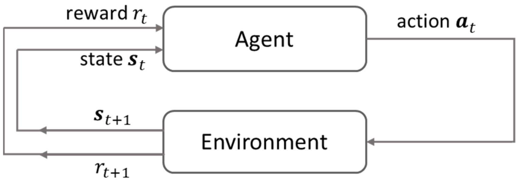


Figure 2.1: Agent interacting with the environment in an MDP

reward function that explains their behaviors in inverse reinforcement learning (IRL) [28].

The purpose of reinforcement learning (RL) is to find an optimal strategy in the long term, similar to how humans behave and therefore suitable to describe the shared control problem setting. I summarize the key components of a reinforcement learning problem setup that are adopted throughout the dissertation to describe the problem setting, from Sutton and Barto [29].

Markov Decision Process A *Markov Decision Process (MDP)* is a formulation of sequential decision-making in a stochastic setting. It is a straightforward formulation of the problem of learning from interaction to achieve a goal, and it is widely used in solving optimization problems using dynamic programming and reinforcement learning.

In an MDP, an *agent* learns and makes decisions by interacting with the *environment*. At each discrete time step t, the agent observes a representation of the environment's *state* $\boldsymbol{s}_t \in \mathcal{S}$ and selects and *action* $\boldsymbol{a}_t \in \mathcal{A}$. In the next step, the agent receives a *reward* $r_{t+1} \in \mathcal{R}$ and observes a new state $\boldsymbol{s}_{t+1}$, as shown in Figure 2.1. The state-transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ describes the probability of landing in a state $\boldsymbol{s}_{t+1}$ given the previous state $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$.

Policy A policy is a strategy that directs the agent's actions to maximize its long-term reward. A policy can either be *stochastic* or *deterministic* depending on the number of actions an agent can take at a state, and it assigns either a probability or directly an action.

A *stochastic policy* $\pi : \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ represents the probability of choosing a possible action based on a state. That is, a policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ indicates the probability of taking an action $\boldsymbol{a}_t$ at a state $\boldsymbol{s}_t$. This implies that there can be multiple actions to select from in a certain state.

A *deterministic policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ directly maps states to actions, and a policy $\pi(\boldsymbol{s}_t) = \boldsymbol{a}_t$ prescribes an action $\boldsymbol{a}_t$ at a state $\boldsymbol{s}_t$. When the policy is deterministic, there is only one action that the agent can take at a certain state.

Value Function The *state-value function* $v_\pi(\boldsymbol{s})$ expresses how good a state is for the agent to be in. It is the expected return (discounted sum of rewards) when following a policy π starting at a state $\boldsymbol{s}$

$$v_\pi(\boldsymbol{s}) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = \boldsymbol{s}] \tag{2.1}$$

$$= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1}], \text{ for all } \boldsymbol{s} \in \mathcal{S} \tag{2.2}$$

where γ is the discount rate, r_t is the reward, and G_t is the return. From the perspective of optimal control, the objective is to minimize the cost rather than the reward. Therefore, the value function measures the long-term cost at the given state, which is known as the cost-to-go function [30].

The *action-value function* (state-action value function) $q_\pi(\boldsymbol{s}, \boldsymbol{a})$ represents the value for the state-action pair. It is the expected return of taking action a in state $\boldsymbol{s}$:

$$q_\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = \boldsymbol{s}, A_t = \boldsymbol{a}]. \tag{2.3}$$

for all $\boldsymbol{s} \in \mathcal{S}$ and $\boldsymbol{a} \in \mathcal{A}$.

In reinforcement learning, the agent's goal is to find the optimal policy π^* that maximizes the cumulative reward. The value of following the optimal policy π^* is represented by the *optimal state-value function*, defined as

$$v^*(\boldsymbol{s}) = \max_\pi v_\pi(\boldsymbol{s}) \text{ for all } \boldsymbol{s} \in \mathcal{S} \tag{2.4}$$

Likewise, following π^* generates an *optimal action-value function* defined as:

$$\begin{align} q^*(\boldsymbol{s}, \boldsymbol{a}) &= \max_\pi q_\pi(\boldsymbol{s}, \boldsymbol{a}), \text{ for all } \boldsymbol{s} \in \mathcal{S} \text{ and } \boldsymbol{a} \in \mathcal{A} \tag{2.5} \\ &= \mathbb{E}[r_{t+1} + \gamma v^*(S_{t+1}) | S_t = \boldsymbol{s}, A_t = \boldsymbol{a}]. \tag{2.6} \end{align}$$

2.5 Riemannian Geometry

This section introduces the mathematical background of the methods presented later in Chapters 4 and 5 concerning Riemannian geometry, by summarizing topics from [31]–[33]. For an in-depth introduction, see the mentioned references.

2.5.1 Riemannian Geometry

Manifold A manifold $\mathcal{M}$ is a topological space that appears locally Euclidean. The set of all derivatives of all smooth curves passing through a point $\boldsymbol{p} \in \mathcal{M}$ defines a vector space called the *tangent space*, denoted as $T_{\boldsymbol{p}}\mathcal{M}$. *Tangent vectors* are vectors on $T_{\boldsymbol{p}}\mathcal{M}$ that are tangent to the manifold $\mathcal{M}$ at point $\boldsymbol{p}$. The union of all tangent spaces at all points on the manifold $\mathcal{M}$ forms a new manifold called the *tangent bundle* $T\mathcal{M}$.

Riemannian metric A Riemannian metric g on a manifold $\mathcal{M}$ is a smooth, covariant 2-tensor field whose value $g_{\boldsymbol{p}}$ at each point $\boldsymbol{p} \in \mathcal{M}$ is a positive, definite inner product on the tangent space $T_{\boldsymbol{p}}\mathcal{M}$

$$\langle \boldsymbol{v}, \boldsymbol{w} \rangle_g = g_{\boldsymbol{p}}(\boldsymbol{v}, \boldsymbol{w}) \tag{2.7}$$

for vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ The inner product induces a Riemannian norm defined as:

$$||\boldsymbol{v}|| \coloneqq \sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_g} = \sqrt{\boldsymbol{v}^\top g \boldsymbol{v}} \tag{2.8}$$

Naturally, the metric g augments the space along its Eigenspectrum proportional to the square roots of its Eigenvalues.

The Euclidean metric is a special case of the Riemannian metric where the value at each point is the Euclidean dot product on the tangent space. On the Euclidean plane, the metric tensor corresponds to an Identity matrix and the Euclidean distance is then defined as $||\boldsymbol{v}|| = \sqrt{\boldsymbol{v}^\top \boldsymbol{v}}$.

Riemannian manifold A Riemannian manifold $\mathcal{M}$ is an n-dimensional, differentiable smooth manifold, which is endowed with the Riemannian metric g.

Geodesic A geodesic is a parameterized, smooth curve on the Riemannian manifold that is locally the shortest possible path between two points.

Pushforward and pullback Given a smooth map $\varphi : \mathcal{M} \to \mathcal{N}$ between two smooth manifolds $\mathcal{M}$ and $\mathcal{N}$, the *differential* maps the tangent space of the domain manifold $\mathcal{M}$ to the tangent space of the co-domain manifold $\mathcal{N}$ at a point $\boldsymbol{p}$, $d\phi_{\boldsymbol{p}} : T_{\boldsymbol{p}}\mathcal{M} \to T_{\boldsymbol{p}}\mathcal{N}$. Conceptually, the differential "pushes" the tangent vectors from the tangent space $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ to the cotangent space $\mathcal{T}_{\phi(\boldsymbol{p})}\mathcal{N}$, hence the term *pushforward.*

The *pullback* refers to the dual linear map $d\varphi^*_{\boldsymbol{p}}$ that maps the tangent space of the co-domain $\mathcal{N}$ back to its domain $\mathcal{M}$.

2.5.2 Riemannian Geometry of Manipulators

Riemannian geometry provides a mathematical framework explaining the movements of multi-joint manipulators, including human and robot arms. The geometric properties of the human arm such as the path and the posture, follow geodesics, i.e., the shortest possible path in the Riemannian configuration space that generates less muscular effort when ignoring external forces [34]. Human motions are described as *sequences of geodesic synergies*, series of minimum energy movements in the configuration manifold, which can then be transferred to the robot motion as geodesic paths in the robot configuration space [35].

Any n-dimensional, multi-linked mechanical system's configuration space can be characterized by a Riemannian manifold, referred to as the configuration manifold $\mathcal{Q}$ [31] and it is the space spanned by the degrees of freedom of multi-joint robots [36]. Each robot configuration $\boldsymbol{q}$ is a point on $\mathcal{Q}$ and the tangent vector on the point corresponds to the velocity. Multiple points on the manifold $\mathcal{Q}$ can be connected to form a curve, and the robot generates a trajectory following the sequence of configurations along the curve.

The forward kinematics map $\phi : \boldsymbol{q} \mapsto \boldsymbol{x}$ links the robot configuration manifold to the robot's end-effector workspace manifold, by mapping the robot configuration $\boldsymbol{q}$ to the end-effector position $\boldsymbol{x}$ in Cartesian coordinates. Specifically, the *Jacobian* $\boldsymbol{J}_\phi \equiv \frac{\partial \phi}{\partial \boldsymbol{q}}$ is the differential that transforms the configuration velocity to the task space velocity:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_\phi \dot{\boldsymbol{q}} \tag{2.9}$$

which gives the best linear approximation near the given point.

For a redundant manipulator, the aim is to find a solution that minimizes the quadratic cost functional of joint velocities

$$c(\dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}\boldsymbol{W}\dot{\boldsymbol{q}} \tag{2.10}$$

where $\boldsymbol{W}$ enables us to compute path lengths using the norm. Among the possible curves $\xi = (\boldsymbol{q}_1, \cdots, \boldsymbol{q}_T)^\top$ in the Riemannian configuration space, the manipulator follows the shortest path – i.e., *geodesics*– which minimizes the cost [37]:

$$\psi(\xi) = \frac{1}{2}\sum_{t=1}^{T} \dot{\boldsymbol{q}}_t^\top \boldsymbol{W}(\boldsymbol{q}_t)\dot{\boldsymbol{q}}_t \Delta t. \tag{2.11}$$

The solution to minimizing the functional (Equation (2.10)) can be derived using the method of Lagrange multipliers as:

$$\dot{\boldsymbol{q}} = \boldsymbol{W}^{-1}\boldsymbol{J}^\top(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^\top)^{-1}\dot{\boldsymbol{x}}_e \tag{2.12}$$

where $\dot{\boldsymbol{x}}_e$ denotes the end-effector velocity defined in task space. Here, the subscript from $\boldsymbol{J}_\phi$ is dropped and the notation $\boldsymbol{J}$ is used for simplicity. When $\boldsymbol{W}$ is the identity matrix, the solution locally minimizes the norm of joint velocities and Equation (2.12) is simplified into $\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}\dot{\boldsymbol{x}}_e$, where $\boldsymbol{J}^{\dagger} = \boldsymbol{J}^{\top}(\boldsymbol{J}\boldsymbol{J}^{\top})^{-1}$ is known as the *pseudo-inverse* of $\boldsymbol{J}_{\phi}$ [38].

2.6 Natural Gradients

2.6.1 Gradient Descent

Gradient descent is a first-order optimization procedure to locally minimize a real-valued function $f : \mathbb{R}^d \rightarrow \mathbb{R}$

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}). \tag{2.13}$$

The local minimum is found by iteratively taking steps in the direction of the negative gradient $-\frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$, the steepest descent direction, until it converges to a local minimum:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{2.14}$$

where η is the step size [39].

The choice of η influences the speed of convergence, and it is generally difficult to choose the suitable η that guarantees fast convergence from any initial $\boldsymbol{\theta}_0$. This is because each gradient component $\frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i}$ varies in size and direction along each dimension. Without considering the structure of the underlying parameter space, it is difficult to choose η that assures fast convergence of each $\boldsymbol{\theta}_i$ [40].

2.6.2 Natural Gradient Descent

In the general non-linear optimization framework, the parameter space is often non-Euclidean, but rather a Riemannian space. In this case, the natural gradient represents its steepest direction, rather than the ordinary gradient [41].

Recalling gradient descent, we write it as an optimization problem where we minimize a first-order approximation of the function $f(\boldsymbol{\theta})$ subject to the constraint that the distance in the step $\delta\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\theta}_t$ is marginal [39]:

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\arg\min} \quad & f(\boldsymbol{\theta}_t) + \nabla f(\boldsymbol{\theta}_t)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_t) \\ \text{subject to} \quad & ||\boldsymbol{\theta} - \boldsymbol{\theta}_t||_{\boldsymbol{G}}^2 = \epsilon^2 \end{aligned} \tag{2.15}$$

The natural gradient descent computes the distance over the manifold in the Riemannian space that the coordinates parameterize, rather than the Euclidean space. $\boldsymbol{G}(\boldsymbol{\theta})$ is the *Riemannian metric tensor* described in Equation (2.7), which characterizes the intrinsic curvature of a particular manifold in n-dimensional space. The natural gradient is then defined as

$$\tilde{\nabla} f = \boldsymbol{G}^{-1} \nabla f \tag{2.16}$$

It is equivalent to standard gradient descent (in Euclidean space) when $\boldsymbol{G}(\boldsymbol{\theta})$ is the identity matrix, such that Equation (2.8) is equivalent to the squared Euclidean norm [41]. When $\boldsymbol{G}(\boldsymbol{\theta})$ is a positive-definite Hessian, it is analogous to Newton's method. When $\boldsymbol{G}(\boldsymbol{\theta})$ lies in the parameter space of a statistical model, it refers to the Fisher information matrix $\boldsymbol{F}(\boldsymbol{\theta})$.

2.6.3 Fisher Information Matrix

The Fisher information $\boldsymbol{F}(\boldsymbol{\theta})$, by definition, measures the expectation of the overall sensitivity of a probability distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$ to changes of parameters $d\boldsymbol{\theta}$. It is defined as the variance of the $score = \frac{d}{d\boldsymbol{\theta}} \log p(\boldsymbol{x}|\boldsymbol{\theta})$, which indicates the sensitivity of the model to changes in $\boldsymbol{\theta}$ [42].

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{\theta}) &= \mathop{\mathbb{E}}_{p(\boldsymbol{x}|\boldsymbol{\theta})} \left[\left(\frac{d}{d\boldsymbol{\theta}} \log p(\boldsymbol{x}|\boldsymbol{\theta}) \right)^2 \right] \\ &= - \int p(\boldsymbol{x}|\boldsymbol{\theta}) \frac{d^2}{d\boldsymbol{\theta}^2} \log p(\boldsymbol{x}|\boldsymbol{\theta}) d\boldsymbol{x}^1 \end{aligned} \tag{2.17}$$

Most importantly to our interest, the Fisher information is the second-order derivative (Hessian, $\boldsymbol{H}$) of the Kullback–Leibler (KL) divergence [43].

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{\theta}) &= - \int p(\boldsymbol{x}|\boldsymbol{\theta}) \nabla^2_{\boldsymbol{\theta}'} \log p(\boldsymbol{x}|\boldsymbol{\theta}') \big|_{\boldsymbol{\theta}'=\boldsymbol{\theta}} d\boldsymbol{x} \\ &= \boldsymbol{H}_{\mathrm{KL}(p(\boldsymbol{x}|\boldsymbol{\theta}||p(\boldsymbol{x}|\boldsymbol{\theta}')))} \end{aligned} \tag{2.18}$$

This gives the key to the connection between natural gradient and the KL divergence, where KL divergence is the function to measure the

[1] It is equivalently defined under mild regularity conditions [42]

"distance[2]" in gradient descent [44] as below:

$$\mathrm{KL}(p(\boldsymbol{x}|\boldsymbol{\theta})||p(\boldsymbol{x}|\boldsymbol{\theta}')) \approx \frac{1}{2}\delta\boldsymbol{\theta}^\top \boldsymbol{F}(\boldsymbol{\theta})\delta\boldsymbol{\theta} \tag{2.19}$$

2.7 Summary

The chapter begins by defining shared control and what it means to share control between the human operator and the semi-autonomous agent. It also introduces the main techniques underlying the algorithms that are presented in the dissertation, such as the formulation of the problem using terminology from reinforcement learning, Riemannian geometry, and natural gradients.

[2]KL divergence is not formally a distance metric since it is not symmetric

Chapter 3

Representing the Semi-Autonomous Agent

In the proposed shared control setting, the *overlapping interaction paradigm* is chosen where both the human operator and the semi-autonomous agent simultaneously provide the system with control inputs at every time step. Since the resulting robot's actions are influenced by both the human and the semi-autonomous agents' commands, the robot never follows a pre-planned, fine-tuned, optimal trajectory. From the semi-autonomous agent's point of view, its trajectory is constantly perturbed by human control. The semi-autonomous agent should therefore be responsive enough to generate optimal control inputs from any state of the environment.

In this chapter, I describe how the policies of semi-autonomous agents are represented which are used to blend control inputs for shared control. I first start with defining the task scenario for shared control and lay out the key assumptions and the relationship between the policy and the value function which are the key concepts that support the contributions made in the following Chapters 4 and 5. Different techniques, including the use of neural networks, are demonstrated to learn an optimal robot policy for reaching that is fast to query the action from any state of the robot. These policies are utilized in the next chapters to present novel arbitration methods for shared control, which is the main focus of the dissertation.

Parts of the work presented in this chapter are based on the following publications:

- Y. Oh, S.-W. Wu, M. Toussaint, and J. Mainprice, 'Natural Gradient Shared Control', *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 1223–1229, 2020, © 2020 IEEE.
- Y. Oh, M. Toussaint, and J. Mainprice, 'Learning to Arbitrate Human and Robot Control Using Disagreement Between Sub-Policies', *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5305–5311, 2021, © 2021 IEEE.
- Y. Oh, J.-C. Passy, and J. Mainprice, 'Augmenting Human Policies Using Riemannian Metrics for Human-Robot Shared Control', *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2023, © 2023 IEEE.

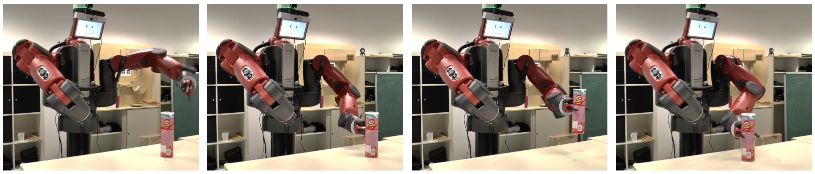


Figure 3.1: Baxter robot performing pick and place (a chain of reaching tasks)

3.1 The Reaching Task

In the dissertation, I investigate the problem of shared control teleoperation for reaching tasks. Whether teleoperation in manipulation or navigation, reaching a point in space by moving the robot (or a part of the robot) is the most elemental subtask. For example, manipulation tasks in disaster response, such as clearing debris or operating tools, can be broken down into a chain of reaching subtasks including moving to the object-grasping point, grasping, and moving to the target point, similar to Figure 3.1. Navigation of a robotic wheelchair or a mobile robot can be viewed as a long-horizon reaching task.

When a robot has a comprehensive model of its environment, reaching can be simple; the robot can compute an optimal policy toward the target point. If not (e.g., poor perception, limited autonomy), it can be challenging to scale the reaching task to become robust.

A teleoperation manipulation scenario in which the goal is to direct the gripper to grasp an object from a set of graspable objects is considered, similar to Figure 3.2. It is critical to highlight that it is presumed that

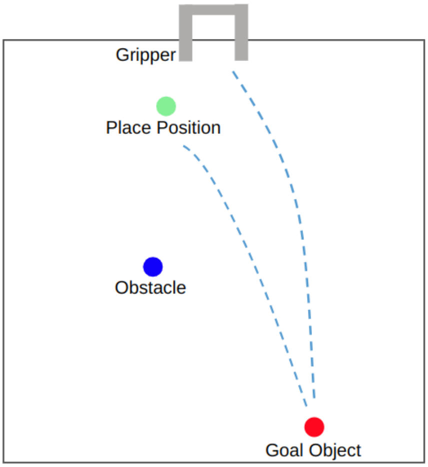


Figure 3.2: Manipulation environment setup

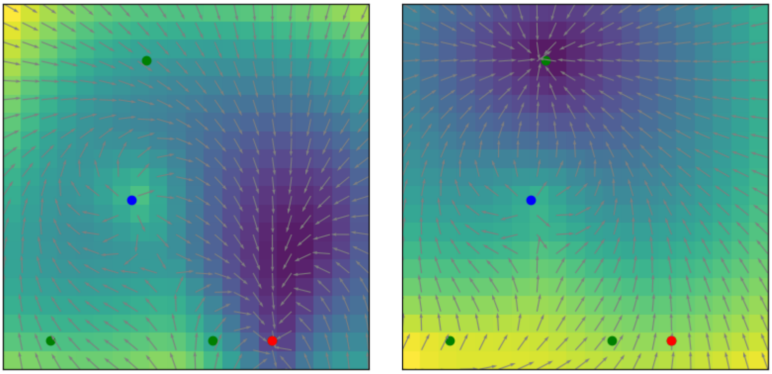
the semi-autonomous agent has access to the optimal policy for reaching the goal given its precise location. Details on obtaining the optimal policy will follow in Section 3.3.

3.2 Insight into the Value Function Through Policies

A significant assumption made throughout the dissertation (especially Chapters 4 and 5) is that information about the value function can be derived from the policy. The value of a state indicates how desirable a state is for the agent to be in. Estimating the landscape of the value function through queried policies enables us to infer information about the robot's perceived environment through the tendency of their values. Intuitively, large changes in the values in the local neighboring states indicate a strong preference towards being in the state.

Figure 3.3 shows the values calculated for each state of the environment as a heatmap, from the environment shown in Figure 3.2 The goal state has a higher value than its surroundings and all the neighboring states generate an optimal action that leads to the goal. In contrast, a state near an obstacle has a lower value than its neighbor, causing actions to move away from it.

From Equation (2.5), the action-value function is at its maximum when following the optimal policy. In other words, the optimal policy



(a) Object pick policy (b) Object place policy

Figure 3.3: Computed values with respect to the state using trained policies

π^* at a state $\boldsymbol{s}$ chooses the action $\boldsymbol{a}$ that maximizes the action-value function $q_\pi(\boldsymbol{s}, \boldsymbol{a})$:

$$\pi^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} q_\pi(\boldsymbol{s}, \boldsymbol{a}) \tag{3.1}$$

Combining the equation with the Bellman optimality equation, i.e. $v^*(\boldsymbol{s}) = \max_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{s})} q^*(\boldsymbol{s}, \boldsymbol{a})$, the optimal policy at a state corresponds to the action that maximizes the value function $v^*(s)$:

$$\pi^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} v^*(\boldsymbol{s}) \tag{3.2}$$

The policy cannot be used to directly estimate the value of a state. In special circumstances where the system dynamics are linear and the reward function (cost function) is quadratic, the policy represents the steepest descent direction of the value function as discussed in the next section.

3.2.1 Linear Dynamics and Quadratic Cost Assumption

When it is approximated that the dynamics of the system are linear and the cost function is quadratic, the value function is a quadratic function of the state [48]:

$$\text{dynamics: } f(\boldsymbol{x}, \boldsymbol{u}) = \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{3.3}$$

$$\text{cost function: } C(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u} \tag{3.4}$$

$$\text{value function: } V(\boldsymbol{x}, t) = \boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} \tag{3.5}$$

where $\boldsymbol{x}$ is the state vector, $\boldsymbol{u}$ is the control vector[1], $\boldsymbol{A}$ and $\boldsymbol{B}$ refer to the state transition matrix and control multipliers, and $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the state and control cost matrices.

The optimal control becomes:

$$\boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}\boldsymbol{P}\boldsymbol{x} \tag{3.6}$$

which is known as a *Linear Quadratic Regulator*. The solution can be driven by substituting Equations (3.3)–(3.5) in the Hamilton-Jacobi-Bellman equation:

$$-\frac{\partial V(\boldsymbol{x},t)}{\partial t} = \min_{\boldsymbol{u}} \left\{ C(\boldsymbol{x},\boldsymbol{u}) + \frac{\partial V(\boldsymbol{x},t)}{\partial \boldsymbol{x}} \cdot f(\boldsymbol{x},\boldsymbol{u}) \right\} \tag{3.7}$$

where $V(\boldsymbol{x},t)$ is the value function, $C(\boldsymbol{x},\boldsymbol{u})$ is the cost function (i.e., negative reward function), and $f(\boldsymbol{x},\boldsymbol{u})$ is the dynamics of the system [30].

The optimal control action $\boldsymbol{u}$ is a linear transformation of the gradient of the value function $V(\boldsymbol{x},t)$. In Equation (3.6), $-\boldsymbol{P}\boldsymbol{x}$ indicates the steepest descent direction of the value function. $\boldsymbol{B}^{\top}$ projects the steepest descent onto the control space, and $-\boldsymbol{B}^{\top}\boldsymbol{P}\boldsymbol{x}$ is the steepest descent achieved with the control vector $\boldsymbol{u}$. $\boldsymbol{R}^{-1}$ scales the direction of descent based on different control inputs [49]. This idea is utilized to estimate a scalar map of the value function (except for the constant term) from its derivatives, to determine the level of arbitration during shared control based on the landscape of the underlying value function.

[1]The notations follow the conventional optimal control theory notations: $\boldsymbol{x}$ as state, $\boldsymbol{u}$ as action, V as value function, to prevent confusion.

3.2.2 Action as a Gradient of the Value Function

Specifically, in a linear dynamics system where action $\boldsymbol{u}$ is a tangent vector of the state manifold, e.g., the position as the state and the velocity as the action, the optimal action $\boldsymbol{u}^*$ can be expressed using Equation (3.7) and $\boldsymbol{u} = \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \dot{\boldsymbol{x}})$:

$$\dot{\boldsymbol{u}}^* = \arg\min_{\dot{\boldsymbol{x}}} \left[C(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \frac{\partial V}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}} \right] \tag{3.8}$$

The minimum of $[\cdot]$ is obtained when $\frac{\partial}{\partial \dot{\boldsymbol{x}}} \left(C(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \frac{\partial V}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}} \right) = 0$. In special cases when the cost function $C(\boldsymbol{x}, \dot{\boldsymbol{x}})$ is only dependent on the state, i.e., $C(\boldsymbol{x}, \dot{\boldsymbol{x}}) = C(\boldsymbol{x})$, Equation (3.8) can then be further simplified:

$$\frac{\partial^2 V}{\partial \boldsymbol{x} \partial \dot{\boldsymbol{x}}} \dot{\boldsymbol{x}} + \frac{\partial V}{\partial \boldsymbol{x}} = 0 \Rightarrow \dot{\boldsymbol{x}}^* = -\frac{\partial V}{\partial \boldsymbol{x}}$$

The equation leads to the optimal action being directly the negative gradient of the value function with respect to the state. This is an important assumption used in the following chapters to locally estimate the value function.

3.3 Learning a Goal-Conditioned Autonomous Agent Policy

From the perspective of the semi-autonomous agent, the robot's trajectory is consistently perturbed by human controls in a shared control setting. While some work such as Wang et al. [15] explicitly model the uncertainty in the interaction, the focus here is to obtain a robust and reactive optimal policy that can handle disruptions and generate optimal actions for any state, given a single known goal location. The semi-autonomous agent is capable of being fully “autonomous” when it has an accurate model of the environment; in other words, it can achieve the task without the presence of the human agent when the goal is known.

The trained optimal policy in a three-dimensional task space (translation in x, y plane + rotation) is described. The policy illustrated in this section serves as a foundation for Chapters 4 and 6. Next, approaches to acquiring an optimal policy in a seven-dimensional configuration space are described, which is later utilized in Chapter 5.

3.3.1 Learning an Optimal Policy in Task Space

The robot policy is represented using a neural network that outputs robot actions that resemble an optimized trajectory, which was computed using a trajectory optimization algorithm. When trained effectively, the network is generalized over the entire state space, allowing it to infer an optimal action for the robot at any state.

Given the system's state $\tilde{\boldsymbol{s}}_t$ and a goal $\boldsymbol{g}$, the network infers a unit velocity vector $\boldsymbol{v}_{gripper}$ and end-effector rotation $\mathbf{v}_{rot_Z}$ that represent the next optimal action towards the goal. The state $\tilde{\boldsymbol{s}}_t$ is a concatenation of robot and environment states consisting of: end-effector position $\boldsymbol{p}_{gripper}$, relative obstacle position $\boldsymbol{p}_{obstacle}$, relative goal position $\boldsymbol{p}_{goal}$, and the rotation of the gripper $\mathbf{v}_{rot_Z}$ in the axis orthogonal to the workspace plane. That is, for the two-dimensional rotation matrix $\boldsymbol{M} \in SO(2)$, where

$$\boldsymbol{M} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$$

the rotation is represented by its first column vector allowing the representation to be continuous [50]:

$$\mathbf{v}_{rot_Z} = [\cos\varphi, \sin\varphi]^\top \tag{3.9}$$

The state and action space of the robot policy is solely defined in task space. Therefore, the network must learn to predict end-effector actions that also avoid collisions after computing the joint configurations using inverse kinematics. The training data is meticulously collected by producing pick-and-place trajectories with random starting positions and environment configurations. Trajectories were generated using a Rapidly exploring Random Tree (RRT) followed by a Gauss-Newton trajectory optimizer [51] that is tuned to perform precise pick-and-place motions considering the full-arm kinematics and the workspace, as shown in Figure 3.4. A generous number of trajectories is required for training to cover the whole state space of the network input during training, however, this can be obtained effortlessly from simulated robot motions.

Figure 3.5 illustrates the architecture of the neural network that

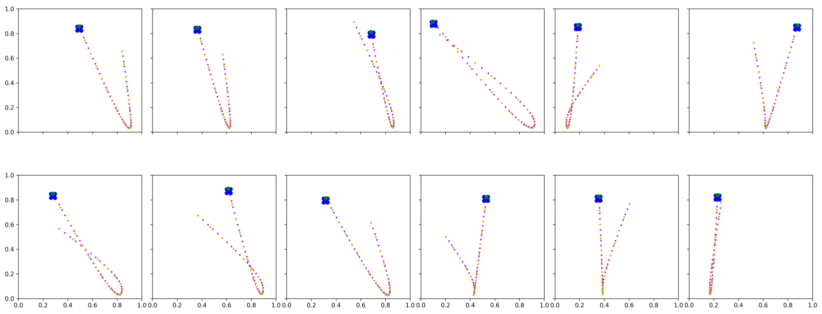
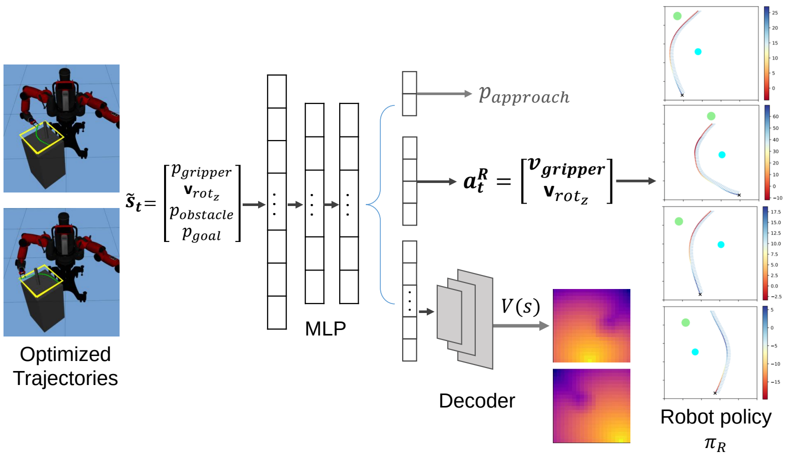


Figure 3.4: Pick-and-place trajectories generated using a Gauss-Newton trajectory optimizer

represents the autonomous robot agent. The network consists of multiple fully connected dense layers that predict the optimal robot actions $\boldsymbol{a}_t^R$ at a given state. A multi-loss function is used to penalize/enforce specific behaviors of the robot policy.

In the pick phase, the network predicts an approach point $\boldsymbol{p}_{approach}$, which is a point in the optimal trajectory that the gripper reaches when approaching the object, and an additional cost term is added to minimize its prediction error. This ensures that the network generates actions that lead the gripper to approach the object parallel to the gripper's fingers for grasping.

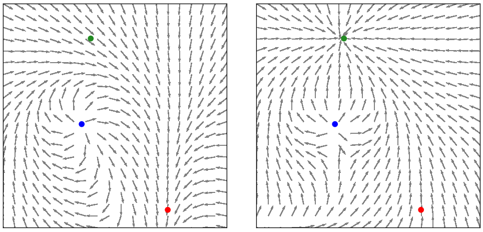
In addition, the network predicts a cost map of the environment (shown as heatmaps in Figure 3.5) that replicates the state log partition function $V(\boldsymbol{s})$ computed with Maximum Entropy Inverse Optimization Control (MaxEnt IOC) [52][53]. The state log partition function $V(\boldsymbol{s})$



Optimized Trajectories
$\tilde{s}_t = \begin{bmatrix} p_{gripper} \\ \mathbf{v}_{rot_z} \\ p_{obstacle} \\ p_{goal} \end{bmatrix}$
MLP
$p_{approach}$
$a_t^R = \begin{bmatrix} v_{gripper} \\ \mathbf{v}_{rot_z} \end{bmatrix}$
$V(s)$
Decoder
Robot policy π_R

Figure 3.5: Overview of the autonomous robot agent's policy network

is a soft estimate of the expected cost to reach the goal from a state $\boldsymbol{s}$. The network generates smoother trajectories by enforcing the network to reconstruct the learned $V(\boldsymbol{s})$ from the latent layer of the network using a decoder structure with a series of convolution and up-sampling layers. After training with 24K trajectories, the network generates autonomous pick-and-place trajectories with a success rate of 95% in random environment conditions when following inferred action at each state. Figure 3.6 shows the vector field of the robot policy inferred over the whole workspace. The successfully learned policies show positive divergence from the obstacle and negative divergence towards a goal.


(a) Inferred pick and place policy actions over the state space

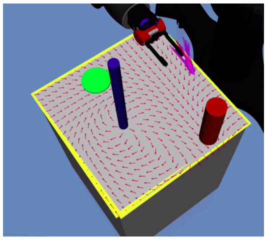
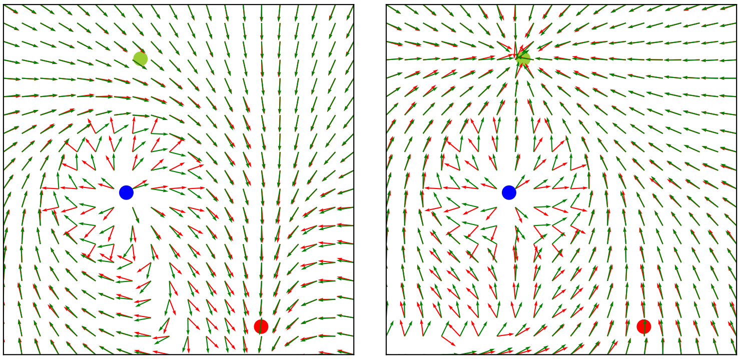

(b) The policy projected onto the environment

Figure 3.6: Quiver plot of the actions generated by the trained autonomous robot policies and its projection onto the simulated environment

3.3.2 Fitting Locally Weighted Regression Models

Locally Weighted Regression (LWR) fits a regression model that is valid only in the local region given a set of data points around the point of interest [54]. Since the resulting model is linear, the robot policy is fitted to an LWR model to reduce any non-linearity generated by the output of the network. The model is used to query data points to compute smoother Jacobians using the finite difference method, as part of the process to compute the Fisher information matrix $\boldsymbol{F}$ described in Section 4.3.3.

Another advantage of using LWR is that the model can be augmented by providing additional data points since the model is fit only in the local region. This is useful to improve the behavior of the learned policy. For instance, we prefer the learned policy to enforce stronger obstacle



(a) Object pick scenario (b) Object place scenario

Figure 3.7: Comparison between the neural network policy (green arrows) and the augmented policy using LWR (red arrows)

avoidance; thus, we provide additional data points computed using a signed distance field when fitting LWR near the obstacle.

The vector fields of the policies regressed using LWR are shown as red arrows in the top row in Figure 3.7. The LWR model successfully approximates the neural network policy (shown in green arrows), as well as augmenting a stronger repulsion in neighboring points of the obstacle.

3.3.3 Learning an Optimal Policy in Configuration Space

A neural network model is trained to represent the optimal robot policy in a seven-dimensional configuration space. The network directly infers the configuration velocity $\Delta \boldsymbol{q} \in \mathbb{R}^7$ given the gripper position $\boldsymbol{p}_{gripper}$, relative goal object position $\boldsymbol{p}_{goal}$, rotation of the goal object $R_{goal_{6D}}$, and current configuration $\boldsymbol{q}$. The rotation of the goal object $R_{goal_{6D}}$ is represented as a 6-dimensional vector composed of the first two columns of the rotation matrix to ensure continuity in the rotation representation [50][55].

Similar to the robot policy network for task space, a multi-layer perceptron network (MLP) composed of fully connected dense layers is used to predict the configuration velocity $\boldsymbol{\Delta q}$, where the network outputs both the unit velocity $\hat{\boldsymbol{q}} = \frac{\boldsymbol{\Delta q}}{||\boldsymbol{\Delta q}||}$ and the norm of the velocity $||\boldsymbol{\Delta q}||$ individually. The loss function is computed as a weighted sum of the mean squared errors of the outputs. The trained network generates autonomous pick trajectories with a success rate of $92 \sim 97\%$ when following the inferred action at each state.

As the feature space of the network becomes high dimensional, it becomes nearly impossible to collect training data that covers the whole state space. The goal position $\boldsymbol{p}_{goal}$ is restricted to a fixed position to reduce the complexity when training the model. Despite the effort, many blind spots occur especially when training the network from trajectory data, as shown in Figure 3.8. The figure shows that the data is not distributed along the joint limit boundaries (shown as vertical red lines), albeit collecting almost 300K grasp trajectories from random starting positions.

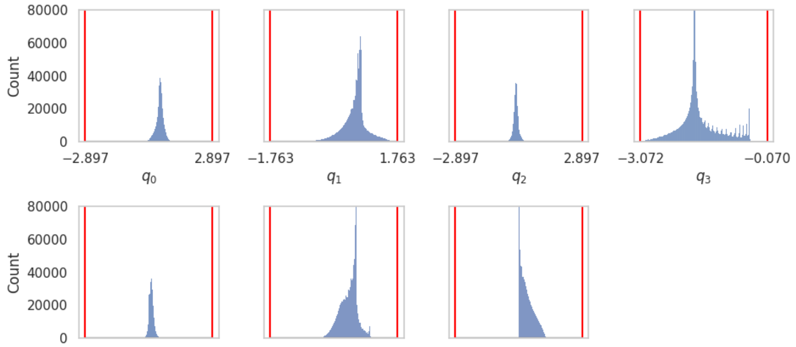


Figure 3.8: Distribution of the joint configuration values from the training data

3.3.4 Generating Local Policies Using Linear Quadratic Regulator

Given a single, optimized reference trajectory ξ^{ref}, a Linear-quadratic regulator (LQR) controller is used to generate policies that steer the system toward the reference trajectory ξ^{ref}, such that $\lim_{t\to\infty} \boldsymbol{x} - \boldsymbol{x}_d = 0$, where $\boldsymbol{x}$ and $\boldsymbol{x}_d$ are states of the robot and the reference trajectory. This is known as *trajectory tracking* [31].

Assuming that the error is small, a linear system using the state error $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_d$ and the control error $\boldsymbol{\nu} = \boldsymbol{a} - \boldsymbol{a}_d$ can be written as:

$$\dot{\boldsymbol{e}} = \boldsymbol{A}\boldsymbol{e} + \boldsymbol{B}\boldsymbol{\nu} \tag{3.10}$$

The optimal control controller is computed using the following equation

$$\boldsymbol{\nu} = -\boldsymbol{K}\boldsymbol{e} \tag{3.11}$$

which leads to the control $\boldsymbol{u}_t$ for state $\boldsymbol{x}_t$ at time t

$$\boldsymbol{u}_t = -\boldsymbol{K}(\boldsymbol{x}_t - \boldsymbol{x}_d) + \boldsymbol{u}_d \tag{3.12}$$

The control matrix $\boldsymbol{K}$ is obtained by solving the discrete-time algebraic Riccati equation, given the coefficient matrices that define the LQR problem. The overall algorithm for computing the policy is summarized in Algorithm 1. At a robot configuration $\boldsymbol{q}_t$, we find the index i_t of the configuration $\boldsymbol{q}_{td}$ of the reference trajectory ξ^{ref}, that is the closest to $\boldsymbol{q}_t$. The state and control errors are computed using a segment of the reference trajectory starting from the index. The velocity of the trajectory is computed using backward finite difference, where the initial velocity is set to zero. Our main interest is to determine the optimal velocity at $\boldsymbol{q}_t$, as opposed to the entire LQR trajectory. The velocity is computed using the equation of motion.

Figure 3.9 illustrates the generated trajectories using LQR trajectory tracking which converge to the reference trajectory. The gray dotted line indicates the reference trajectory, whereas the black solid line represents the segment of the reference trajectory. The colored lines show the trajectories starting from nearby states, and the red arrows show the calculated velocity.

The state cost matrix $\boldsymbol{Q}$ and the control cost matrix $\boldsymbol{R}$ define the quadratic cost function, which alters the behavior of the tracking trajectory.

Algorithm 1: Generating Robot Policies using LQR

```
Input:
    • Robot configuration $\boldsymbol{q}_t$
    • reference trajectory $\xi^{ref} = \{\boldsymbol{q}_n^{ref}\}_{n=1}^{N}$
    • control matrix $\boldsymbol{K}$
Output: velocity $\boldsymbol{v}_t$
Function compute_policy($\boldsymbol{q}_t, \xi^{ref}, \boldsymbol{K}, dt$):
    Find the closest configuration $\boldsymbol{q}_{td} = \xi^{ref}[i_t]$
    Calculate robot state $\boldsymbol{x}_t = [\boldsymbol{q}_t, \boldsymbol{v}_t]$
    Calculate reference state $\boldsymbol{x}_{td} = [\boldsymbol{q}_{td}, \boldsymbol{v}_{td}]$
    Compute acceleration using Equation (3.12)
    Integrate to obtain velocity
    return $\boldsymbol{v}_t$
End Function
```

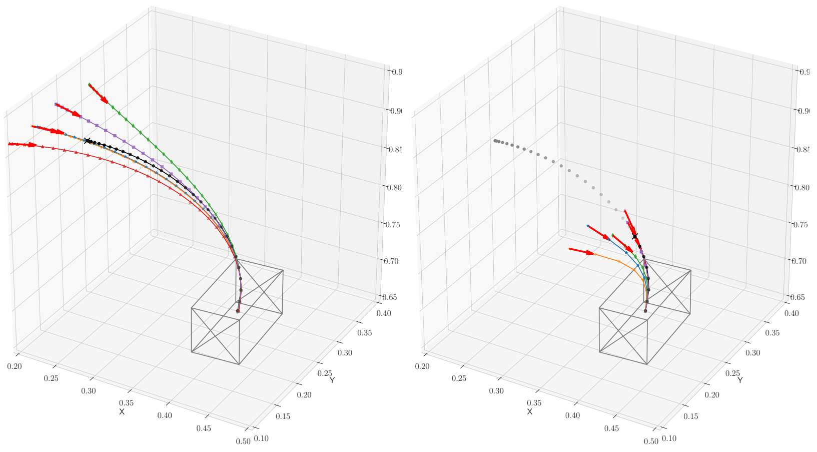


Figure 3.9: Trajectories generated using LQR at different locations

3.4 Summary

This chapter focuses on describing the semi-autonomous agent (goal-conditioned autonomous agent) that is utilized throughout the next chapters, including the problem formulation, key assumptions and the technical methods for training a global policy using data-driven architectures. Since the state of the robot is constantly changing due to the inputs from the human agent, and the autonomous agent has to compute a global optimal policy from each new state at each time step, a reactive autonomous agent is essential for the proposed shared control setting.

Providing a powerful, robust autonomous agent that can solve manipulation tasks in diverse settings is beyond the scope of the research. With the help of complex data-driven models and advanced reinforcement learning techniques such as [56]–[58], it is anticipated that the burden of obtaining a robust optimal policy will decrease, allowing easy access to the semi-autonomous robot agent.

Chapter 4

Natural Gradient Shared Control

In this chapter, I introduce a novel method to arbitrate the commands of human and semi-autonomous agents. The method allows the human operator to retain maximum control whenever possible, so that the human operator does not have to fight against the robot's autonomy, but still receives assistance in critical situations. For background theory related to the chapter, refer to Section 2.6.

The work presented in this chapter has been published in:

- Y. Oh, S.-W. Wu, M. Toussaint, and J. Mainprice, 'Natural Gradient Shared Control', *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 1223–1229, 2020, © 2020 IEEE.

4.1 Introduction

A simple form of policy blending that is still widely used in many shared control frameworks [59]–[61] is the linear blending paradigm introduced by Dragan et al. [20]. The blended policy $\boldsymbol{u}$ is represented as

$$\boldsymbol{u} = \alpha \boldsymbol{a}^R + (1 - \alpha)\boldsymbol{a}^H \tag{4.1}$$

where $\boldsymbol{a}^H$ and $\boldsymbol{a}^R$ are human and autonomous robot policies and $\alpha \in [0, 1]$ is a coefficient to adjust the level of arbitration. In this approach, the amount of arbitration depends on how well the intentions of the human operator are predicted. The blended policy follows the autonomous policy when the robot predicts the user's intent with high confidence. However, this has been reported to result in mixed user preference, with some users preferring to retain control despite longer completion times [24], [25]. In addition, if the assistance is against the user's intention, this approach can increase the user's workload [20]; the user "fights" against the assistance rather than gain help from it.

Some works have taken the option of giving the user maximum control authority by providing minimal assistance only when necessary. Broad et. al. introduced minimal intervention shared control, which computes whether the control signal will lead to an unsafe state and, if so, replaces the user's control [62], [63]. However, in these works, the problem is usually modeled by the selection of an action within a feasible set.

In this work, the shared control is formulated as an optimization

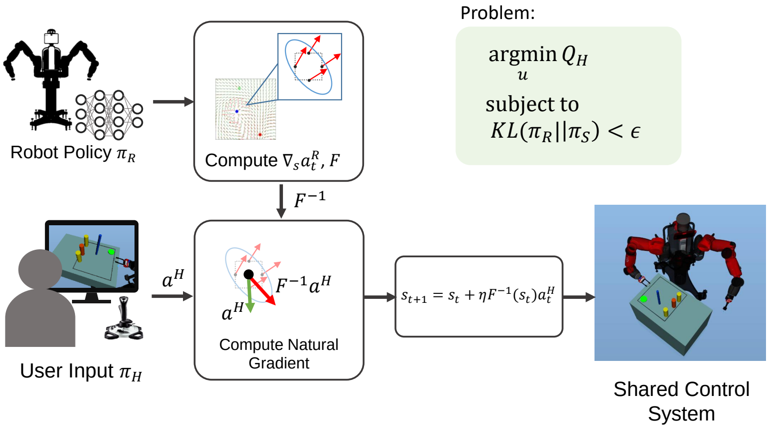

Robot Policy π_R
Compute $\nabla_s a_t^R, F$
Problem:
$\underset{u}{\text{argmin}}\, Q_H$
subject to
$KL(\pi_R||\pi_S) < \epsilon$
F^{-1}
User Input π_H
a^H
$F^{-1}a^H$
a^H
Compute Natural Gradient
$s_{t+1} = s_t + \eta F^{-1}(s_t) a_t^H$
Shared Control System

Figure 4.1: Overview of arbitration strategy using Natural Gradient Shared Control

problem, as shown in Figure 4.1. The shared control action is chosen to maximize the internal action-value function of the user, while the shared control policy is constrained not to deviate from the policy of the autonomous robot. The Fisher information matrix is constructed, which is an expression of how sensitive a distribution is to changes in the local neighborhood of the state. When the autonomous robot's policy is represented as a vector field over the state space, assistance is provided where the policy locally diverges. The inverse Fisher information matrix adapts the user's actions so that the robot gains more authority when the robot's policy changes rapidly in the local region (e.g., near an obstacle or near a goal). In the regions where the policy does not locally diverge, the user can retain more control authority Through the use of the Fisher information matrix, the term “Natural Gradient Shared Control” is introduced.

In order to evaluate the efficacy of the approach, we define a teleoperation task and compare the quantitative metrics, where a user performs pick-and-place tasks with a simulated robotic arm. We show that our shared control paradigm can assist the user in achieving the goal while simultaneously giving the user more control authority.

The main contributions are summarized as:

- A shared control paradigm that depends on natural gradients emerging from the divergence constraint between the robot and the shared policy
- Approximating the Fisher information matrix by sampling the autonomous robot’s policy and computing the local gradient using finite difference
- Generation of an autonomous robot policy that is based on task space states and can be regressed over the entire state space
- Quantitative results of a preliminary human user study with 16 participants, demonstrating the effectiveness of the proposed paradigm to give more user control

4.2 Related Work

4.2.1 Different Methods of Sharing Control

Shared control refers to the cooperation between the user and the autonomous robot agent to achieve a common task at a control level [13]. Different paradigms for shared control have been developed depending on the task and purpose of the application. We categorize the paradigms into three groups depending on how the agents share control: control switching, direct blending, and indirect blending.

Control switching is the allocation of all-or-none assistance during control. It is a discrete switch between full autonomy and direct control depending on a predefined circumstance. Control switching can be momentary: at each state, the robot evaluates whether to take over, depending on its confidence in the user's intentions [20] (aggressive mode); or to prevent the system from entering an unsafe state [62]. It is also called *traded control* when the robot fully takes over and executes a sub-task over a sequence of time steps [64]–[66].

Direct blending involves explicitly combining both agents' control using an arbitration function. The above *control switching* corresponds to when the step function is used as the arbitration function. Approaches include using a linear function [20], [60], [61], a sigmoid function [67], specifically tuned [21], [68], or learned [59], [69]. It can be tedious to define and tune an arbitration function that generalizes across users and tasks. As a result, the blended action may be worse than following either policy.

Indirect blending occurs when the shared control is a result of an optimization problem. Javdani et al. [25] formulated the problem as a POMDP (Partially Observable Markov Decision Process) and approximated using hindsight optimization to provide assistance which minimizes the expected cost-to-go for an unknown goal. Reddy et al. [70] used deep Q-learning to select an action closest to the user's suggestion while being suboptimal in discrete states. Broad et al. [63] introduced shared control with "minimal intervention", such that the human's command is perturbed when the system leads to an inevitable collision state. Similarly, our work does not directly blend the controls despite the presence of the autonomous robot action but rather computes a metric based on the topology of the autonomous policy in the local neighborhood and adjusts the user commands.

4.2.2 Natural Gradients

The natural gradient adjusts the direction of the standard gradient according to the underlying Riemannian structure of the parameter space. It is effective when the direction of the standard gradient descent does not represent the steepest descent direction of the cost function in the parameter space, which leads to poor convergence [40], [41].

The natural gradient is advantageous as it is invariant under coordinate transformations and unlike Newton's method, it doesn't assume the cost function is locally quadratic. It is widely applicable to non-linear functions in neural networks [41], including reinforcement learning methods such as policy gradient [71] and actor-critic [72]. In Schulman et al. [73], a trust region is defined by constraining the KL divergence between the old and the new policy during policy updates. Similarly, our shared control framework imposes a constraint on the KL divergence of the autonomous policy and the shared control policy such that the shared control policy does not diverge far from the autonomous robot policy.

4.3 Natural Gradient Shared Control

4.3.1 Expressing Shared Control as an Optimization Problem

Let $\boldsymbol{s} \in \mathcal{S}$ be the state of the system. Let $\boldsymbol{a}^H \in \mathcal{A}^{\mathcal{H}}$ as the user action, $\boldsymbol{a}^R \in \mathcal{A}^{\mathcal{R}}$ be the autonomous robot action, and $\boldsymbol{u} \in \mathcal{U}$ be the shared control action. The human and the robot agent both select actions following their stochastic policies, π_H, and π_R. Our goal is to find a shared control policy π_S that satisfies the following optimization problem.

$$\begin{aligned} \underset{\boldsymbol{u}_t}{\arg\max} \quad & Q_H(\boldsymbol{s}_t, \boldsymbol{u}_t) \\ \text{subject to} \quad & \mathrm{KL}(\pi_R \| \pi_S) < \epsilon \end{aligned} \tag{4.2}$$

At each time step, the shared control policy is chosen to maximize the user's internal action-value function $Q_H(\boldsymbol{s}_t, \boldsymbol{u}_t)$. The KL divergence constraint between the robot policy and the shared policy ensures that

Algorithm 2: Natural Gradient Shared Control

```
for $t = 1, T$ do
    Observe user action $\boldsymbol{a}_t^H$
    foreach $g \in \mathcal{G}$ do
        Compute belief $b_{t,g}$
        $\boldsymbol{F}_g(\boldsymbol{s}_t) = \texttt{ComputeFisher}(\boldsymbol{s}_t, g)$
    $\boldsymbol{F}(\boldsymbol{s}_t)^{-1} = \sum_g b_{t,g} \boldsymbol{F}_{t,g}^{-1}(\boldsymbol{s}_t)$        ▷ weighted sum over goals
    $\boldsymbol{u}_t \leftarrow \boldsymbol{F}(\boldsymbol{s}_t)^{-1} \boldsymbol{a}_t^H$        ▷ compute shared action
    $\boldsymbol{s} \leftarrow \boldsymbol{s} + \eta \boldsymbol{u}_t$        ▷ update state
```

the shared policy does not deviate far from the autonomous robot policy.

The user's internal action-value function $Q_H(\boldsymbol{s}_t, \boldsymbol{u}_t)$ is approximated using the user's action. We view the user's action as an estimate of $\nabla_{\boldsymbol{s}} Q_H(\boldsymbol{s}_t, \boldsymbol{a}_t^H)$ at each step (refer to Section 3.2.2), assuming that people take actions in the direction that maximizes their internal value function. Otherwise, it is possible to learn Q_H using methods such as Maximum Entropy Inverse Optimal Control (MaxEnt IOC) [52], but predicting the user controls $\boldsymbol{a}_t^H$ can be challenging due to interpersonal differences.

The problem can be expressed using a Lagrange Multiplier, assuming a linear approximation of our objective $Q_H(\boldsymbol{s}_t, \boldsymbol{a}_t^H)$ and a quadratic approximation of the KL divergence constraint. Solving this approximation of the Lagrangian leads to an update rule that introduces natural gradient adaptation.

4.3.2 Natural Gradient Shared Control

We introduce a state update function for shared control using natural gradient adaptation. Note that our goal is to find the action that maximizes Q_H, resulting in taking gradient steps in the direction of ascent.

$$\begin{aligned} \boldsymbol{s}_{t+1} &= \boldsymbol{s}_t + \eta \boldsymbol{F}(\boldsymbol{s}_t)^{-1} \nabla_{\boldsymbol{s}} Q_H(\boldsymbol{s}_t, \boldsymbol{a}_t^H) \\ &= \boldsymbol{s}_t + \eta \boldsymbol{u}_t \end{aligned} \tag{4.3}$$

η is the step size and the natural gradient in Equation (2.16) corresponds to the shared control action $\boldsymbol{u}_t \sim \pi_S(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t^H, \boldsymbol{a}_t^R)$. We utilize the approximation $\boldsymbol{a}_t^H \propto \nabla_{\boldsymbol{s}} Q_H(\boldsymbol{s}_t, \boldsymbol{a}_t^H)$ in Equation (4.5). The proportionality constant is absorbed by the step size η. The overall algorithm is summarized in Algorithm 2.

$$\boldsymbol{u}_t = \boldsymbol{F}(\boldsymbol{s}_t)^{-1} \nabla_{\boldsymbol{s}} Q_H(\boldsymbol{s}_t, \boldsymbol{a}_t^H) \tag{4.4}$$

$$= \boldsymbol{F}(\boldsymbol{s}_t)^{-1} \boldsymbol{a}_t^H \tag{4.5}$$

The Fisher information matrix $\boldsymbol{F}(\boldsymbol{s}_t)$ can be interpreted as the sensitivity of the autonomous robot policy π_R to changes in the parameter.

Intuitively, a vector field is defined by regressing a deterministic robot policy over the entire state space. This vector field contains information about which optimality and constraint trade-offs are made about the underlying actions. When an obstacle is in an environment, it acts as a *source* (positive divergence) in the vector field resulting

Algorithm 3: Computing Fisher Information Matrix

```
**Init:** Load pre-trained robot policy model $f_{nn}$
**Input:** $\boldsymbol{s}_t, g$
**Output:** $\boldsymbol{F}_{t,g}$
**Function** ComputeFisher($\boldsymbol{s}_t, g$):
    Sample set of states $\mathcal{S} = \{(\boldsymbol{s}_i, g)\}_{i=1}^{N}$
    Infer robot actions $\mathcal{D} = f_{nn}(\mathcal{S})$
    Fit LWR model $L_g = \text{LWR}(\mathcal{S}, \mathcal{D})$
    $\tilde{\boldsymbol{a}}_{t,g}^{R} \leftarrow L_g(\boldsymbol{s}_t)$          ▷ query action from LWR
    $\boldsymbol{H}_{t,g} \leftarrow \nabla_{\boldsymbol{s}} \tilde{\boldsymbol{a}}_{t,g}^{R}$          ▷ compute Jacobian of action
    $\boldsymbol{F}_{t,g} \leftarrow \frac{1}{2}(\boldsymbol{H}_{t,g} + \boldsymbol{H}_{t,g}^{\top})$          ▷ ensure symmetry
    **return** $\boldsymbol{F}_{t,g}$
**End Function**
```

in a repulsive action. When the policy is goal-directed, the goal acts as a *sink* (negative divergence) and the vectors around the goal point inward. $\boldsymbol{F}$ measures how sensitive the field changes and emphasizes or discounts towards certain directions of $\boldsymbol{u}_t$.

4.3.3 Computing the Fisher Information Matrix

We approximate $\boldsymbol{F}$ as the curvature of the robot's action-value function at a given state:

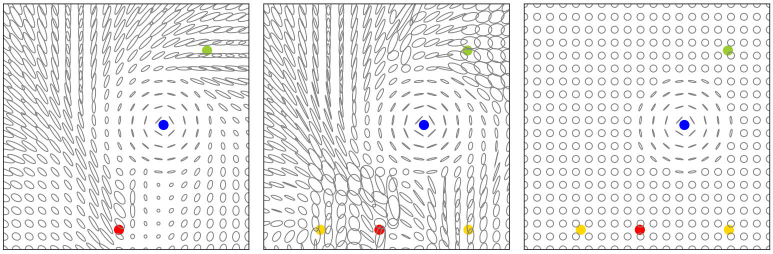
$$\boldsymbol{F}(\boldsymbol{s}_t) = \mathop{\mathbb{E}}_{\pi_H} [\nabla_{\boldsymbol{s}} \log \pi_R(\boldsymbol{a}_t^R | \boldsymbol{s}_t) \nabla_{\boldsymbol{s}} \log \pi_R(\boldsymbol{a}_t^R | \boldsymbol{s}_t)^{\top}] \tag{4.6}$$

$$\approx \nabla_{\boldsymbol{s}}^2 Q_R(\boldsymbol{s}_t, \boldsymbol{a}_t) \tag{4.7}$$

Keeping in mind the connection between $\boldsymbol{F}$ and the KL divergence and the asymmetry of the KL divergence, one may ask how dependent is $\boldsymbol{F}$ to the underlying assumption of $\mathrm{KL}(\boldsymbol{p}||\boldsymbol{q})$ or $\mathrm{KL}(\boldsymbol{q}||\boldsymbol{p})$. It turns out that when $\boldsymbol{p}$ and $\boldsymbol{q}$ are close, the KL divergence is locally/asymptotically symmetric [44]. Hence, our definition for $\boldsymbol{F} \approx \nabla^2_{\boldsymbol{s}} Q_R$ is equivalent to integrating the user actions over all possible robot actions.

We describe our method for computing $\boldsymbol{F}$ in Algorithm 3, where $\boldsymbol{F}$ is computed at each state for each goal. We use Locally Weighted Regression (LWR) to fit a local model L_g using a set of sampled states and actions inferred using a pre-trained model. A detailed explanation regarding LWR is described in Section 3.3.2. As we consider action as an approximation of the first derivative of the Q-function, we consider the Jacobian of the robot action as the Hessian of the Q-function. $\nabla_{\boldsymbol{s}} \tilde{\boldsymbol{a}}^R_t$ is the Jacobian computed using the finite difference method with actions $\tilde{\boldsymbol{a}}^R_{t,g}$ from L_g. The Fisher information matrix $\boldsymbol{F}$ is positive-definite by definition. However, the Jacobian computed using the finite difference method may not always be symmetric. Thus, we decompose the matrix into a symmetric and a skew-symmetric matrix, and we apply the symmetric matrix.

The objective of the robot assistance can be flexibly defined depending on the cost function that the robot policy optimizes. Figure 4.2 shows the ellipses computed over the state space, using eigenvalues and eigenvectors of $\boldsymbol{F}(\boldsymbol{s})^{-1}$ for each assistance mode; the robot assistance can be goal-directed (in Figure 4.2(a)), or it can minimally assist to avoid obstacles (in Figure 4.2(c)). The direction of the ellipse represents the direction along which the user's action is stretched. When the



(a) Single goal (b) Belief-weighted over goals in a multi-goal environment (c) Obstacle avoidance

Figure 4.2: Plots showing the computed ellipses over the state space

ellipse is close to a circle, the user has more control authority over the system. When the ellipse is narrow, for example near an obstacle, the robot augments the user's action toward one direction.

In the case of goal-directed assistance with multiple goals, the robot must predict a single goal among multiple candidates. We compute $\boldsymbol{F}(\boldsymbol{s}_t)^{-1}$ as a weighted sum over the beliefs representing the confidence in the goal prediction. Figure 4.2(b) shows $\boldsymbol{F}(\boldsymbol{s}_t)^{-1}$ computed using beliefs as a naive distance-based goal prediction. It is shown that when the confidence is low (right top corner, or above the left two objects), the user gains more control authority.

4.4 Experiments

4.4.1 Experiment Setup

To assess the efficacy of the proposed method, we conducted a study including human participants. We defined a simulated teleoperation environment consisting of the Baxter robot operating in a 50 cm $\times$ 50 cm workspace over a table and several cylinders representing objects and an obstacle, as shown in Figure 4.3. In the environment, physical collisions were not simulated and grabbing was implemented by attaching the object to the gripper when grasping is initiated by the user. A joystick (Logitech Extreme 3D Pro) was used to control the robot's right end-effector and the robot was controlled at approximately 30 Hz.

We hypothesized:

- Shared control method based on natural gradients allows the user to have greater control over the task while maintaining both safety and efficiency.

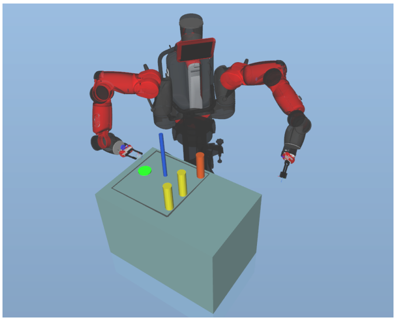
4.4.2 User Study Procedure

The user study included 16 individuals (12 males and 4 females) who used the right hand as their dominant hand. The participants had no apparent prior experience with robots and gave their informed consent before beginning the study.

As demonstrated in Figure 4.3b, participants were instructed to teleoperate the robot's gripper using the joystick to pick up the red cylinder and return it to the green goal position while avoiding the blue pole. The participants controlled the gripper in task space, using the joystick's yaw motion to rotate the z-axis and pitch and roll motions to manipulate the end-effector velocity. The trigger button on the joystick was used to initiate a grasp. Participants were given a perspective view of the robot's workspace to replicate a teleoperation scenario with a limited camera view.

We designed a within-subjects study in which all participants took part in all the conditions, namely:

- **DC**: Direct Control
- **NG**: Natural Gradient Shared Control
- **LB**: Linear Blending


(a) Simulated teleoperation environment

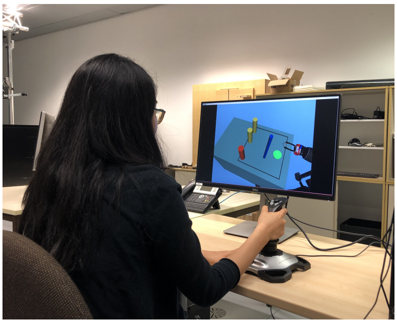

(b) A user interacting with the environment using a joystick

Figure 4.3: The simulated teleoperation environment and an example of the user interaction setup

- **OA**: Obstacle Avoidance

Each participant carried out three sets of demonstrations, each set consisting of four different environment settings repeated across the four different control conditions. The random order of the conditions and the environment settings were predefined and balanced.

For **LB**, we implemented the "timid" mode for linear arbitration suggested in Dragan et al. [20], in which the assistance is proportional to the goal confidence level but never fully taking charge. The robot policy described in Section 3.3.1 was used as the optimal robot policy to linearly blend the policies. In **OA**, minimal assistance was provided to avoid the obstacle using a signed distance function. In all the conditions, the participant was responsible for determining the resulting action's speed.

4.5 Results

A repeated-measures analysis is used to analyze the results, with the assistance method as a factor. As shown in Table 4.1 and Figure 4.4, we compared four quantitative measures: task duration, travel distance, minimum proximity to the obstacle, and the cosine distance between actions. In Figure 4.4, the mean value is indicated using a white dot and the median is indicated as the orange horizontal line in the box. The outliers are indicated as plus symbols.

The results indicate that our method successfully assisted the user in completing the task while allowing the user to maintain control authority.

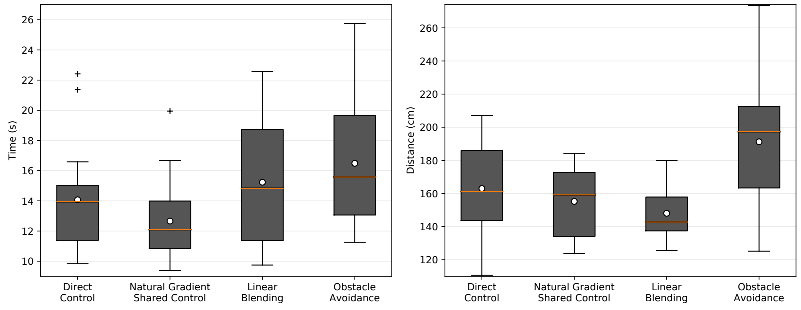
Table 4.1: User study results: mean and standard deviation

Method	**Duration** (s)	**Travel Dist.** (cm)	**Proximity** (cm)	**Cosine Dist.** ($\times 10^{-2}$)
DC	14.1 ± 3.5	163.0 ± 26.2	5.5 ± 1.8	0.4 ± 0.25
NG	12.7 ± 2.8	155.2 ± 20.5	7.4 ± 1.0	6.6 ± 1.0
LB	15.2 ± 4.0	148.0 ± 14.6	6.6 ± 1.1	19.2 ± 5.2
OA	16.5 ± 4.2	191.2 ± 40.4	6.5 ± 0.9	1.5 ± 0.6

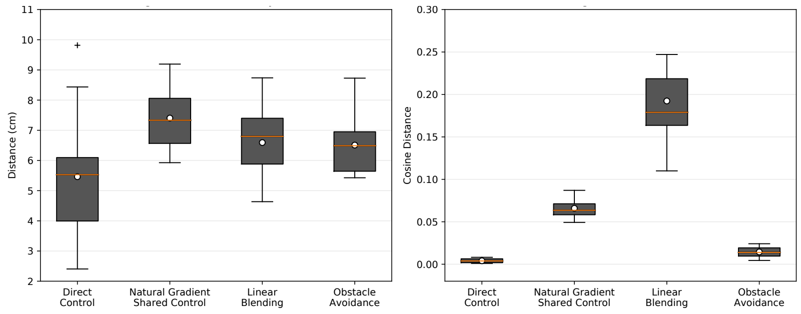
As seen in Figure 4.4 (a), **NG** reported the shortest average task duration with statistical significance: **DC**($f(4,12) = 20.67,\ p = 0.0004$), **LB**($f(4,12) = 16.93,\ p = 0.0009$), and **LB**($f(4,12) = 61.50,\ p < 0.0001$). Comparing the end-effector travel distance in Figure 4.4 (b), **LB** exhibited the shorted average distance followed by **NG**. However, the results were statistically equivalent to **NG** ($f(4,12) = 1.7933,\ p = 0.2005$)

Interestingly, although the **LB** had the shortest average travel distance, its duration was not the shortest, and it had the greatest average cosine distance. This suggests that there may have been a mismatch between the participants' intentions and the actions of the robot, forcing them to resist the robot's behavior. On the other hand, this was not shown in **NG**, indicating that the arbitrated action was acceptable to the user while still being efficient.

As a measure of safety, we compared the average minimum distance between the gripper and the obstacle. The shorter distance



(a) Duration (b) Travel distance



(c) Minimum proximity to obstacle (d) Cosine distance

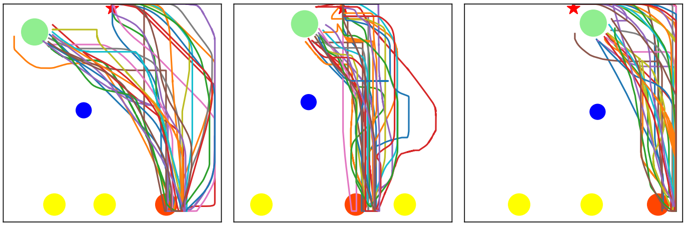
Figure 4.4: Comparison of control paradigms across all users for (a) execution time, (b) travel distance, (c) minimum proximity to the obstacle, (d) cosine distance

signifies a higher risk of collision with the obstacle. **NG** demonstrated the most effective obstacle avoidance with statistical significance: **DC**$(f(4,12) = 31.42, p < 0.001)$, **LB**$(f(4,12) = 18.7363, p = 0.0006)$, and **OA**$(f(4,12) = 23.2720, p = 0.0002)$, while still retaining a reasonably short overall travel distance In **DC** and **OA** where there is minimal to no robot assistance, the participants were responsible for avoiding obstacles.

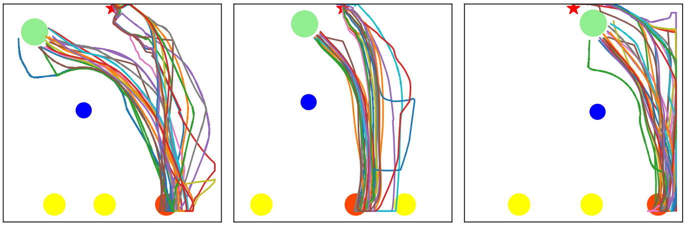
The cosine distance indicates the degree of disagreement between the human command and the executed command. As depicted in Figure 4.4 (d), the average cosine distance of **NG** is significantly less than that of **LB** ($f(4, 12) = 104.36, p < 0.001$), indicating that the **NG** method executed actions that were closer to what the user intended, verifying our hypothesis that **NG** would allocate more control authority.

Methods **DC** and **OA** provided no assistance toward the goal, which explains the smallest cosine distances. The trajectory was minimally influenced, thus resulted in a larger variance compared to **NG** or **LB** when comparing the duration, travel distance, and the minimum proximity to the obstacle.

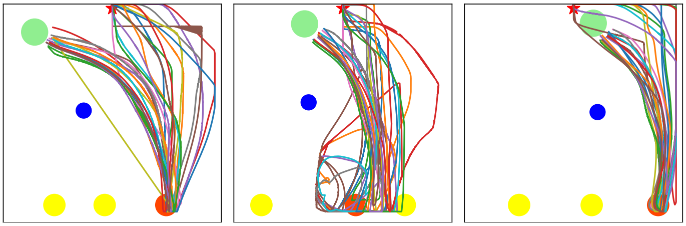
Figure 4.5 shows the participants' trajectories during the user study for three different environments. Participants were instructed to start from the red star, grab the red object, and retrieve back to the green position, while avoiding the blue obstacle. As seen from quantitative results, **DC** and **OA** exhibit divergent paths compared to **NG** or **LB**. Trajectories include dragging along the workspace boundary or approaching the place position from various directions. A notable characteristic of the **LB** method is shown in the second column of Figure 4.5. Since the target object and another object were relatively close in this environment, assistance was often provided toward the wrong goal. The user had to fight against the assistance and noisy trajectories are shown near the objects. Overall, **NG** showed reliable performance in task execution while still maintaining compliance with user commands. The results show that our method can be an option for reducing the discrepancy and increasing user satisfaction during teleoperation.



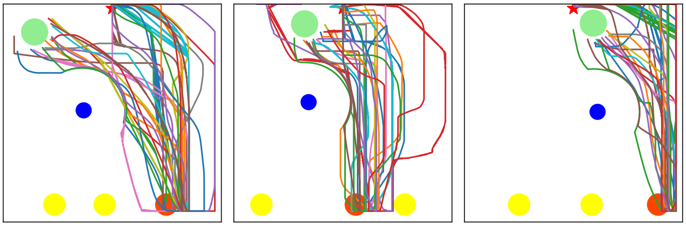
(a) Direct control



(b) Natural Gradient Shared Control



(c) Linear Blending



(d) Obstacle Avoidance

Figure 4.5: Top-down visualization of user demonstrations for tele-operation method (rows) of three different environments (columns)

4.6 Discussion

The novelty of the chapter lies in defining the shared control problem as an optimization problem which leads to the use of the Fisher information matrix to define a trust region to keep the shared control action close to the optimal robot action. The Fisher information matrix $\boldsymbol{F}$ measures the sensitivity of the autonomous robot policy to changes in the state.

Since the method is based on the assumption that there exists a global policy that represents the semi-autonomous agent, the result of the shared control action is dependent on the quality of the semi-autonomous robot policy. The more accurate the optimal policy is, the better the quality of the arbitration between the policies. For this reason, the method presented in this chapter was limited to the end-effector space and the teleoperation task was designed simple enough to acquire an optimal policy. Even so, it was a laborious effort to obtain a feasible optimal policy that can be applied to the method.

In practice, it can be challenging to acquire a global policy for a bigger state space. However, with the help of current advancements in robot control such as task and motion planning (TAMP), reinforcement learning, and parallel computing, sampling diverse optimal robot policies are becoming more accessible with little cost. Further investigations with a larger state space are discussed in the next chapter.

4.7 Summary

This chapter presented a novel shared control paradigm based on natural gradients, that provides effective task completion without compromising the human operator's control authority. Shared control is formulated as an optimization problem, in which the cost function maximizes the human operator's internal value function while minimizing the KL divergence between the autonomous agent policy and the shared control policy. This provides an effective balance where the robot's policy is not compromised while maintaining a high level of user control. Comparisons with other methods of human-in-the-loop control such as linear blending and direct control are shown in the results section, which described the effectiveness of the proposed framework.

Chapter 5

Augmenting Human Policies Using Riemannian Metrics for Shared Control

The work presented in Chapter 4 is extended to higher dimensional spaces, to enable human and robotic agents to operate in different coordinate systems.

The work presented in this chapter has been published in:

- Y. Oh, J.-C. Passy, and J. Mainprice, 'Augmenting Human Policies Using Riemannian Metrics for Human-Robot Shared Control', *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2023, © 2023 IEEE.

5.1 Introduction

Manipulation is an essential feature of teleoperation robots and robots are expected to perform tasks that require a high level of manipulability in a limited amount of time. When it comes to manipulating a high-dimensional robot arm with low-dimensional human inputs, many works relied on the inverse kinematics (IK) solver to control the end-effector of the robot arm to achieve manipulation tasks [74]–[76].

We tackle the task of reaching an object through teleoperation. Reaching a point in space is the most basic subtask when teleoperating a robot arm, and various manipulation tasks in disaster response, such as clearing debris or operating tools, can be broken down into a chain of reaching subtasks. The focus is to develop a smooth teleoperation method that allows the human operator to control the robot manipulator as if they are embodied in the teleoperation robot.

In the previous chapter, a shared control paradigm that prioritizes human inputs while augmenting them with robot assistance only when necessary is presented. The Fisher information matrix is computed, which reflects the changes in the distribution of robot assistance in the neighborhood, and the human inputs are augmented accordingly. The chapter extends the paradigm to higher-dimensional spaces, enabling human and robotic agents to provide commands in their most natural settings.

The human input and robot assistance are in velocities, which are considered to be gradients of the value function existing in the tangent space of the manifold. Each agent commands actions that maximize their internal value functions and the human inputs are supplemented according to the curvature of the robot's optimal value function, a Riemannian metric. The measure keeps track of the cost terms (e.g., avoiding collisions, grasp orientation, joint limits, etc.) that the robot assistance takes into account. In this way, the assistance can help minimize any cost terms or constraints that the human operator might miss during teleoperation. The curvature of the robot's value function is calculated by sampling robot policies in the local neighborhood states. This is useful when the robot's value function is hard to obtain, such as when the policy is modeled by a neural network.

The main contributions are summarized as the following:

- generalizing the shared control framework from Chapter 4 to the configuration space to enable human and robot agents to operate in different spaces
- approximating the curvature of the autonomous robot's value function from policy samples of neighboring states
- locally approximating the global robot policy using a single optimized robot trajectory and computing the curvature using finite differences
- presenting results using simulated human behavior policies that show the potential of the proposed paradigm

5.2 Related Work

5.2.1 Teleoperation in Different Workspaces

The morphological difference between humans and robots results in an asymmetry in the degrees of freedom (DoF), and different workspaces between the human and robot can increase operator workload by demanding intensive training and focus [77]. Prior studies have focused on developing teleoperation systems that reduce the physical workload of the operator through motion mapping [78]–[80], relying on the redundancy mechanism to allow the operator to manipulate the end-effector. Simultaneously, the robot ensures automatic collision avoidance [76], [81], or using learning from demonstration to estimate the desired robot action given the operator's inputs [82], or by learning a correspondence between human and robot arm poses [22]. Another common solution is using virtual fixtures that constrain the manipulability of the robot so that the operator can focus on the high-level task objective while the assistive system takes care of tasks such as collision avoidance or orientation assistance [83]. The virtual fixtures are hand-coded [84], or learned from demonstration [83], [85], [86]. The method is often combined with haptic feedback, known as Haptic Shared Control [87], [88]. Our approach is similar to virtual fixtures as it augments the human inputs to prevent any collisions when teleoperating a redundant manipulator, but the information is acquired from the optimal robot motions rather than demonstrations from the human operator.

5.2.2 Riemannian Approach to Shared Control

Riemannian geometry provides a principled way to extend algorithms from Euclidean space to other manifolds [89]. The configuration space of an n-dimensional, multi-linked robot can be characterized by a Riemannian manifold, referred to as the configuration manifold $\mathcal{Q}$ [31], and previous works such as [37], [90], [91] formulated robot motion using Riemannian geometry, which we take as theoretical inspiration for our method.

From the perspective of shared autonomy, Zeestraten et al. [85] developed a Riemannian approach for Programming by Demonstration to define virtual fixtures that restrict the manipulability of the robot for shared control. In [89], task-parameterized Gaussian Mixture models combined with model predictive control were used to teleoperate a bimanual underwater robot. Also, Ti et al. [92] proposed a Learning from Demonstration (LfD) approach, where the demonstration data distribution is represented using Gaussian distributions on Riemannian manifolds to reduce the number of demonstrations required by the robot to acquire manipulation skills. In a shared control setting, the robot is guided by the operator while automatically determining the grasping strategy. Our approach expresses the local curvature of the autonomous agent's value function using a Riemannian metric, that is used to augment the human operator's command to support grasping while preventing collision.

5.3 Shared Control as an Optimization Problem

In a teleoperation scenario, the human operator and the autonomous robot work together to complete a task. We define two different methods of defining shared control as an optimization problem that combines both agents operating in different dimension spaces, which leads to utilizing the Riemannian metric to modify the human operator's commands.

5.3.1 Constrained Optimization Problem

Recalling our previous work [45], we formulate the optimization problem:

$$\begin{aligned} \underset{\boldsymbol{u}}{\arg\max} \quad & Q_H(\boldsymbol{s}, \boldsymbol{u}) \\ \text{subject to} \quad & \mathrm{KL}(\pi_R \| \pi_S) < \epsilon \end{aligned} \tag{5.1}$$

where $\boldsymbol{s}$ and $\boldsymbol{u}$ denote the state and the shared control action. The objective is to find a shared control policy π_S that maximizes the user's internal action-value function $Q_H(\boldsymbol{s}, \boldsymbol{u})$ while retaining a trust region to keep π_S in the local neighborhood of the autonomous robot policy π_R. We referred to this as "Natural Gradient Shared Control" as the constraint induces a natural gradient update, with respect to the distance in the parameter space. Here, the Fisher metric was used to measure the distance in the distribution space.

In the case of deterministic policies, the policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ directly maps the state space $\mathcal{S}$ to the action space $\mathcal{A}$ rather than outputting a probability associated with the action. Thus, the optimization problem can be rewritten in a similar form as Equation (2.15):

$$\begin{aligned} \underset{\boldsymbol{u}}{\arg\max} \quad & Q_H(\boldsymbol{s}, \boldsymbol{u}) \\ \text{subject to} \quad & \|\boldsymbol{a}_R - \boldsymbol{u}\|^2_{\boldsymbol{W}} < \epsilon \end{aligned} \tag{5.2}$$

where the constraint is defined to ensure that $\boldsymbol{u}$ is close to the autonomous robot action $\boldsymbol{a}_R$.

$\boldsymbol{W}$ is a symmetric, positive definite $n \times n$ matrix that defines an inner product on the vector space $\mathbb{R}^n$. The inner product induces a norm, enabling us to compute the distances between two vectors. It is analogous to the Riemannian metric g described in Section 2.5 since an n-dimensional robot configuration space can be characterized by a Riemannian manifold.

In our prior work [45], we investigated combining policies of both agents operating in the same task space. We now investigate combining both agents' policies operating in different spaces, where the human operator specifies end-effector commands in the task space and the autonomous robot computes appropriate configurations in the joint space. This leads to the following state update rule in the form of $\boldsymbol{q}_{t+1} = \boldsymbol{q}_t + \eta\dot{\boldsymbol{q}}_t$, which adapts the joint variables $\boldsymbol{q}$:

$$\boldsymbol{q}_{t+1} = \boldsymbol{q}_t + \eta \boldsymbol{W}^{-1} \nabla_{\boldsymbol{s}} Q_H(\boldsymbol{s}_t, \boldsymbol{u}^H_t) \tag{5.3}$$

where η is the step size and we approximate $\nabla_{\boldsymbol{s}} Q_H(\boldsymbol{s}_t, \boldsymbol{u}_t) = \boldsymbol{J}^\dagger \boldsymbol{v}_H$

as the human operator's action converted into the configuration space. Intuitively, the human operator naturally decides the action they believe to be optimal according to their policy. This is comparable to the optimal action being proportional to the value function for a system with linear dynamics and quadratic costs. At each time step, the metric $\boldsymbol{W}$ directly augments the user's action according to the curvature of the autonomous robot policy space.

5.3.2 Least-Squares Problem (Sum-of-Squares)

We formulate the shared control problem for manipulation as an unconstrained optimization problem, where the cost function is a sum of squares that jointly minimizes the distance in the end-effector space and the configuration space:

$$\boldsymbol{q}_{t+1} = \arg\min_{\boldsymbol{q}} \|\phi(\boldsymbol{q}) - \boldsymbol{x}^*\|^2_{\boldsymbol{C}} + \|\boldsymbol{q} - \boldsymbol{q}_t\|^2_{\boldsymbol{W}} \tag{5.4}$$

where $\boldsymbol{C}$ and $\boldsymbol{W}$ are task and the configuration space metrics, respectively. The first norm penalizes the quadratic cost between the position of the end-effector and the intended position of the human operator $\boldsymbol{x}^*$, and the second norm penalizes the quadratic cost of the joint velocities. The optimum can be obtained by differentiating Equation (5.4) and utilizing $\phi(\boldsymbol{q}) \approx \boldsymbol{x}_t + \boldsymbol{J}(\boldsymbol{q} - \boldsymbol{q}_t)$ which expresses the local linearization of ϕ at $\boldsymbol{q}_t$.

$$
\begin{aligned}
f(\boldsymbol{q}) &= \|\phi(\boldsymbol{q}) - \boldsymbol{x}^*\|^2_{\boldsymbol{C}} + \|\boldsymbol{q} - \boldsymbol{q}_t\|^2_{\boldsymbol{W}} \\
&= \|\boldsymbol{x}_t + \boldsymbol{J}(\boldsymbol{q} - \boldsymbol{q}_t) - \boldsymbol{x}^*\|^2_{\boldsymbol{C}} + \|\boldsymbol{q} - \boldsymbol{q}_t\|^2_{\boldsymbol{W}} \\
\frac{\partial}{\partial q} f(\boldsymbol{q}) = 0^\top &= 2(\boldsymbol{x}_t - \boldsymbol{x}^* + \boldsymbol{J}(\boldsymbol{q} - \boldsymbol{q}_t))^\top \boldsymbol{C}\boldsymbol{J} + 2(\boldsymbol{q} - \boldsymbol{q}_t)^\top \boldsymbol{W}
\end{aligned}
$$

Rearranging the equation leads to the following update rule, analogous to the Gauss-Newton method:

$$
\begin{aligned}
\boldsymbol{q}_{t+1} &= \boldsymbol{q}_t + (\boldsymbol{J}^\top \boldsymbol{C}\boldsymbol{J} + \boldsymbol{W})^{-1} \boldsymbol{J}^\top \boldsymbol{C}(\boldsymbol{x}^* - \boldsymbol{x}_t) \\
&= \boldsymbol{q}_t + \boldsymbol{W}^{-1}\boldsymbol{J}^\top (\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^\top + \boldsymbol{C}^{-1})^{-1}(\boldsymbol{x}^* - \boldsymbol{x}_t) \\
&= \boldsymbol{q}_t + \eta \boldsymbol{W}^{-1}\boldsymbol{J}^\top (\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^\top + \epsilon \boldsymbol{I})^{-1} \boldsymbol{v}_H
\end{aligned}
$$

where $\boldsymbol{v}_H = \boldsymbol{x}^* - \boldsymbol{x}_t$ is the human operator's commands. The equation is equivalent to Equation (2.12) when $\boldsymbol{C} \to \infty$, which strictly enforces the robot's end-effector position to be at the user's intended position $\boldsymbol{x}^* = \phi(\boldsymbol{q})$. We substitute $\boldsymbol{C}^{-1} = \epsilon \boldsymbol{I}$ in the last line of the calculation to ensure numerical stability when inverting near singularity.

The first cost objective in Equation (5.4) is analogous to maximizing $Q_H(\boldsymbol{s}, \boldsymbol{u})$ given that the user's action-value is the maximal when executing the user's commands that lead to the user's intended position $\boldsymbol{x}^*$.

When the human operator provides only the linear velocity commands $\boldsymbol{v}_H \in \mathbb{R}^3$, the operator can rely on the shared control to modify the orientation accordingly. This is useful in teleoperation scenarios where the human conveniently controls the end-effector position, without having to worry about detailed grasp orientations when controlling redundant manipulators. For full-dimension control, the human can provide both linear velocity $\dot{\boldsymbol{p}}_H$ and angular velocity $\boldsymbol{\omega}_H$, $\tilde{\boldsymbol{v}}_H = [\dot{\boldsymbol{p}}_H \; \boldsymbol{\omega}_H]^\top$, and the geometric Jacobian $\tilde{\boldsymbol{J}} = [\boldsymbol{J}_{pos} \; \boldsymbol{J}_{ori}]^\top$ is applied.

5.4 Recovering the Riemannian Metric

The work is motivated by [37], [90], and [91] where robot motion is optimized by solving a constrained sum-of-squares problem of the objective. The objective incorporates cost terms or constraints (task-related or transitional costs) along the trajectory (for example, Equation (2.11) penalizes square velocities). The optimized robot motion creates an optimal robot policy that minimizes its "cost-to-go", in other words, that yields the optimal value function. Its second derivative (Hessian), comparable to the metric $\boldsymbol{W}$, expresses the value function's local curvature.

Inversely, given an optimized autonomous robot policy, we can estimate the value function and its curvature by computing the finite differences of configuration space variables. This reveals the underlying cost terms (collision avoidance, grasp orientation, joint limits etc.) that the autonomous robot takes into account, and our idea is to augment the human operator's commands using the metric $\boldsymbol{W}$ to stretch the tangent space according to the topology of the value function to consider any cost terms or constraints that the human operator might overlook during teleoperation.

It is important to mention that the actions of the robot are velocities which are regarded as gradients of the value function that exists in the tangent space of the manifold, which enables the estimation of the metric $\boldsymbol{W}$ using finite difference approximation.

5.4.1 Estimating the Hessian Using Finite Differences

Theoretically, the second derivative Hessian can be directly computed as the Jacobian of the gradient. However, directly estimating the Hessian matrix by computing first derivatives from velocities can lead to asymmetrical Hessian matrices, which violates the definition that the Hessian matrix of a scalar-valued function is symmetric. Instead, it is computed by first recovering the local values of the scalar field and directly constructing the Hessian using finite difference approximation, as shown in Figure 5.1.

That is,

$$\boldsymbol{W} = \Phi_2 \Phi_1^{-1} \dot{\boldsymbol{q}}_{robot}$$

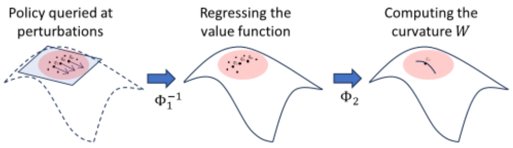

Policy queried at perturbations
Regressing the value function
Computing the curvature W
Φ_1^{-1}
Φ_2

Figure 5.1: Estimating the Hessian using finite differences

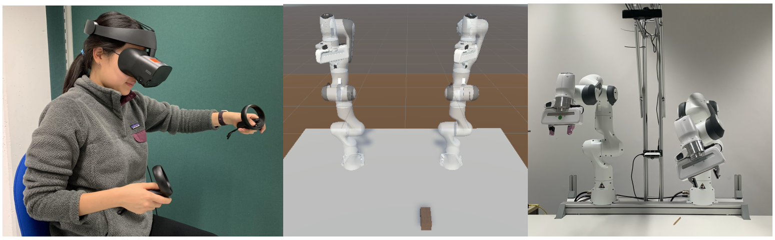
where $\mathit{\Phi}_1$ and $\mathit{\Phi}_2$ are finite difference operators for computing the first and second derivatives of the scalar-valued function. Specifically, $\mathit{\Phi}_1$ estimates the local function values $f(\boldsymbol{q} + h_i\boldsymbol{e}_i)$ for a perturbation along the i-th unit vector $\boldsymbol{e}_i$ defined by the step size h_i:

$$f(\boldsymbol{q} + h_i\boldsymbol{e}_i) = f(\boldsymbol{q})'h_i + f(\boldsymbol{q}) \tag{5.5}$$

where the original function value $f(\boldsymbol{q})$ is considered as $f(\boldsymbol{q}) \approx 0$, which is an offset of the function that vanishes when computing the second derivative. Based on the recovered function values in the local region, $\mathit{\Phi}_2$ computes the second derivatives [93]:

$$\frac{\partial^2 f}{\partial \boldsymbol{q}_i \partial \boldsymbol{q}_j} = \frac{f(\boldsymbol{q} + h_i\boldsymbol{e}_i + h_j\boldsymbol{e}_j) - f(\boldsymbol{q} + h_i\boldsymbol{e}_i) - f(\boldsymbol{q} + h_j\boldsymbol{e}_j) + f(\boldsymbol{q})}{h_i h_j}. \tag{5.6}$$

Utilizing the central finite difference approximation would generate more accurate estimates. However, it would also require more function calls. Since the Hessian is updated at every time step, we favor speed over accuracy and adopt the forward difference, which requires only $1 + n + n(n+1)/2 = 1 + n(n+3)/2$ additional function calls, where n is the number of degrees of freedom. A seven-dimensional robot arm requires 36 function calls from neighboring states to estimate a Hessian matrix. This emphasizes the need for a global robot policy that can rapidly query actions at perturbed states to reduce computation costs.



(a) Using VR headset and controllers (b) Simulation setup (c) Physical robot setup

Figure 5.2: Teleoperation interface using VR and robot setup

5.5 Experiments

5.5.1 Experiment Setup

We created a simulated teleoperation environment with two Franka Emika Panda robot arms [94]. The robot arms are mounted on a table facing the same direction, and a rectangular box (6 cm $\times$ 6 cm $\times$ 12 cm) is placed on the table in front of the robot. The objective is to reach and grab the rectangular box, starting from a random initial position and a random object position. We consider it a successful grasp when the object is between the gripper's fingers. We used KOMO [90] to generate reference trajectories ξ^{ref} in the joint space and included cost terms and constraints such as collision, joint limits, target position, grasp approach position, and gripper orientation. The robot executes a trajectory that first reaches the grasp approach position above the object and grabs the middle of the object to avoid collisions. When grasping, the orientation of the gripper is defined to point downwards

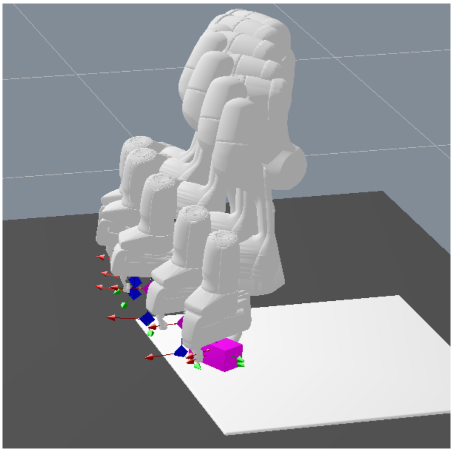


Figure 5.3: Simulating optimized reference trajectories using KOMO

with its lateral axis perpendicular to the long axis of the gripper (See Figure 5.3).

5.5.2 Simulating Human Operator Behaviors

To test our method, we simulate human policies inspired by [70] and [95]. The simulated human is modeled using an inverse kinematics solver and generates different behavior policies:

- *Straightline*: Actions that point directly toward the object
- *Approach*: Actions to reach a pre-grasp position above the object and descend to grab the object
- *Noisy-k*: Approach policy action with injected uniform noise on the joints with size k

- *Biased*: Actions reaching the incorrect object location with max. 5 cm error radius
- *Laggy-m*: Actions based on the previous state observed m steps ago

Noisy, *Biased*, and *Laggy* policies are variants of the *Approach* policy which simulates the disturbance in human actions leading to inaccurate commands.

5.5.3 Simulation Results

Table 5.1 summarizes the results for 100 reach-and-grasp tasks using the simulated human policies. The task is considered a success when the object lies between the gripper's fingers and the object has not deviated from its initial position. We terminate the episode as collided when the gripper collides with the object before reaching a grasp, and timeout occurs when the task duration exceeds the provided time limit.

The constrained optimization method (Constr.) and the least squares method (SoS) outperform the *NoAssist* method regarding the number of succeeded tasks. The two proposed methods formed robot trajectories similar to the reference trajectory. This indicated that optimizing one reference trajectory at the beginning of the episode and using LQR to approximate its action was sufficient to represent the autonomous robot agent without accessing a traditional motion optimizer at each time step. Figure 5.4 shows generated trajectories using different simulated human policies. Each represents the following: human policy (*NoAssist*, blue),

RIGHT	Straightline			Approach			Noisy-0.1			Noisy-0.2			Biased			Laggy-5			Laggy-10			Avg. Cossim		
	Collision	Timeout	**Success**	Collision	Timeout	**Success**	Collision	Timeout	**Success**	Collision	Timeout	**Success**	Collision	Timeout	**Success**	Collision	Timeout	**Success**	Collision	Timeout	**Success**	x	y	z
NoAssist	66	0	34	20	0	80	38	0	62	79	0	21	18	0	82	20	0	80	19	0	81	0.88	0.86	0.90
Constr.	26	0	74	2	7	91	6	2	92	7	5	88	1	4	95	2	9	89	2	9	89	0.99	0.99	0.99
SoS	1	1	98	1	3	96	16	0	84	24	2	74	0	2	98	0	1	99	1	1	98	0.99	0.99	0.99

LEFT	Straightline			Approach			Noisy-0.1			Noisy-0.2			Biased			Laggy-5			Laggy-10			Avg. Cossim		
	Collision	Timeout	**Success**	Collision	Timeout	**Success**	Collision	Timeout	**Success**	Collision	Timeout	**Success**	Collision	Timeout	**Success**	Collision	Timeout	**Success**	Collision	Timeout	**Success**	x	y	z
NoAssist	63	0	37	17	0	83	44	0	56	58	0	42	18	0	82	17	0	83	15	0	85	0.90	0.86	0.91
Constr.	18	0	82	1	13	86	4	4	92	3	2	95	0	9	91	1	12	87	1	15	84	0.99	0.99	0.99
SoS	2	0	98	0	3	97	12	2	86	28	1	71	1	2	97	0	2	98	0	1	99	0.99	0.99	0.99

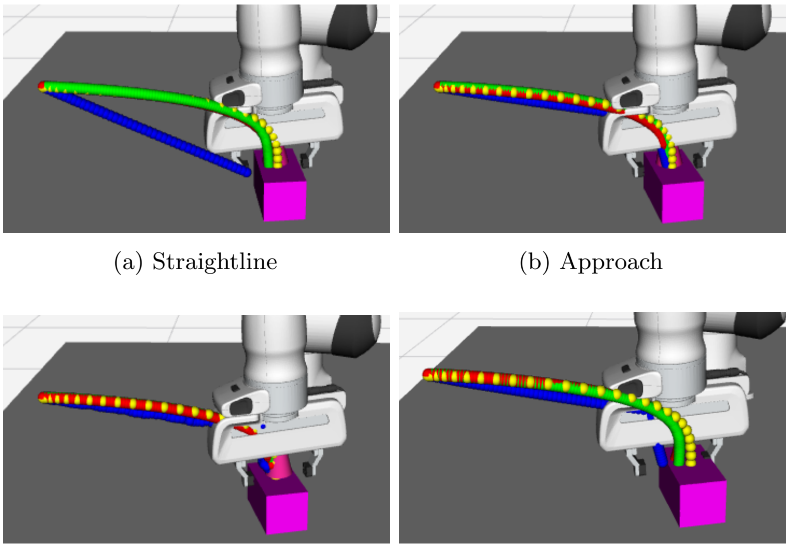
Table 5.1: Results of 100 reach-and-grasp tasks with simulated human policies

constrained (*Constr*, green), least-squares (*SoS*, red), and autonomous robot reference trajectory ξ^{ref} (yellow). As shown in Figure 5.4, our methods adjusted the orientation of the gripper and altered the trajectory to achieve the task, even though the simulated human policies only provided three-dimensional linear velocities excluding the orientation.

The proposed methods also improve the average cosine similarity between the object and the final gripper orientation results. Since the autonomous robot policy follows the reference trajectory ξ^{ref}, which satisfies grasp orientation cost terms (described in Section 5.5.1), the metric $\boldsymbol{W}$ also contributed to correcting the gripper's orientation.

This supports our hypothesis that through the estimated curvature, we can infer the underlying information about the cost and constraint terms of the autonomous robot policy and apply it to adjust the human operator's actions.

The least-squares (SoS) method generally performed better than the constrained method in all simulated users except the *Noisy* users. We assume that in Sections 5.3.1 and 5.3.2, the constrained method imposes harder constraints on the optimization problem, whereas the least-squares method adds softer constraints by optimizing all cost terms altogether. Since the constrained method directly applied the metric $\boldsymbol{W}$ in Equation (5.3), the behavior was more "aggressive" and led to better success rates with the *Noisy* policies by damping out the gripper tremble.



(a) Straightline (b) Approach

(c) Noisy-0.1 (d) Biased

Figure 5.4: Trajectory comparison between simulated policies and methods

5.6 Teleoperation System Design

5.6.1 System Implementation

One of the challenges of implementing a teleoperation system using a VR headset was bridging two different OS systems. The majority of the robot's implementation was done using ROS in Ubuntu, but the VR headset (Meta's Oculus Rift S) required development in Windows. We created a simulated teleoperation environment with two Franka Emika Panda robot arms using the RAI interface, which included the Bullet

]

physics engine. The two robot arms are mounted on a table both facing the same direction similar to the setup using real Panda robot arms, as shown in Figure 5.2.

The environment data such as robot and object configurations and states were published as ROS messages, and the simulation environment was recreated in Unity so that it could be accessed with the VR headset and its touch controllers.

The recent development of Windows Subsystem for Linux (WSL) and Unity Robotics Hub allowed easy implementation and communication between the robotic system and the VR interface without providing separate computers for each OS. The robot was controlled at a frequency of 15 Hz.

The user wears the VR headset and perceives the virtual teleoperation environment from the first-person viewpoint, through a virtual camera located between the robot's arms and facing toward the middle of the table. The touch controllers were used to command movements in free space as well as to send start and grasp signals via the controllers' buttons.

5.6.2 Pilot User Study

Using the Oculus Rift S VR headset and its controllers [96], we developed a teleoperation system where the human operator can give natural task space motion commands as if performing the task themselves. The simulated robot interface was created using Unity Robotics Hub [97], and the robot was controlled at a frequency of 15 Hz.

We conducted a pilot user study to test the system's efficacy, comparing 6-dimensional *NoAssist*, 3- and 6-dimensional *constrained* modes. The 6-dimensional modes refer to the control scheme where the human operator provides linear and angular velocities.

We recruited 11 participants (4 males and 7 females, 9 right-handed and 2 left-handed) to test the system's efficacy, all of whom had prior experience with robots or VR systems but none with our system. The user study followed procedures approved under the Haptic Intelligence framework agreement from the Max Planck Ethics Council (protocol F028A).

- <u>Condition 1</u>: The human and the robot control the 3-dimensional position of the end-effector, and the robot takes care of the 3-dimensional orientation of the gripper, i.e.,

  $$\varDelta \boldsymbol{q} = \boldsymbol{W}^{-1} \boldsymbol{J}_{pos}^{\dagger} \boldsymbol{v}_{user} \tag{5.7}$$

  where $\boldsymbol{J}_{pos}^{\dagger} \in \mathbb{R}^{7\times 3}$ and $\boldsymbol{v}_{user} \in \mathbb{R}^{3}$

- <u>Condition 2</u>: Only the human controls both the 3-dimensional position and the 3-dimensional orientation of the end-effector, i.e.,

  $$\varDelta \boldsymbol{q} = \boldsymbol{I} \boldsymbol{J}_{pose}^{\dagger} \boldsymbol{v}_{user} \tag{5.8}$$

  where $\boldsymbol{W}^{-1} = \boldsymbol{I}$, $\boldsymbol{J}_{pose}^{\dagger} \in \mathbb{R}^{7\times 6}$ and $\boldsymbol{v}_{user} \in \mathbb{R}^{6}$

- Condition 3: The human and the robot together control the 3-dimensional position and the 3-dimensional orientation of the end-effector, i.e.,

  $$\varDelta \boldsymbol{q} = \boldsymbol{W}^{-1} \boldsymbol{J}^{\dagger} \boldsymbol{v}_{user} \tag{5.9}$$

  where $\boldsymbol{J}^{\dagger}_{pose} \in \mathbb{R}^{7\times 6}$ and $\boldsymbol{v}_{user} \in \mathbb{R}^{6}$

For condition 1, the human user's command $\boldsymbol{v}_{user}$ was computed as the difference between the commanded position and the current position of the end-effector. For conditions 2 and 3, the human user's command $\boldsymbol{v}_{user}$ was computed as the pose difference including the orientation represented as 3-dimensional Euler angles.

The condition was considered a within-subjects factor, as each subject experienced all conditions. The order of the conditions was randomized across all participants and each participant was randomly assigned to one of the orders.

The study began with the participants providing their consent and completing a pre-session questionnaire. The study consisted of three sessions, one per condition, during which the participant performed ten rounds of reach-and-grab tasks for each hand using the teleoperation system. Participants had time to adjust to the new condition before starting the data collection. Upon completion of the session, they completed the NASA Task Load Index (NASA-TLX) workload assessment questionnaires and questionnaires inspired by [25] and [98]:

(Q1) I felt in control,

(**Q2**) The robot did what I wanted,

(**Q3**) I was able to accomplish the tasks quickly,

(**Q4**) The robot and I collaborated well together,

(**Q5**) The assistance from the robot was useful in accomplishing the task,

(**Q6**) The robot's actions were reasonable,

(**Q7**) The robot was responsive to me,

(**Q8**) The robot was trustworthy,

(**Q9**) If I were going to teleoperate a robotic arm, I would like to use the system.

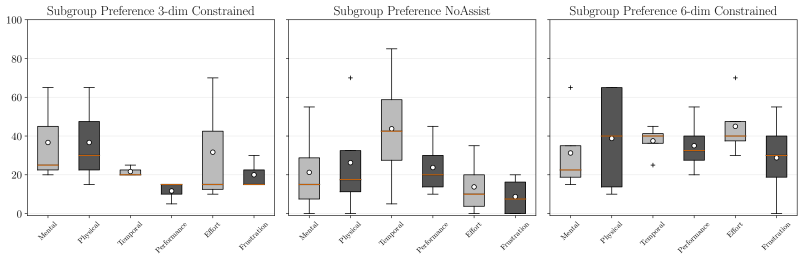
Each session consisted of 10 reach-and-grab tasks with a randomized starting configuration and a random object orientation. The random environment set was pre-defined before the study, and the participants used the same set but in a different order for each condition. For each round, the participants started by matching their virtual end-effector's pose to the given start pose. The participants started the round by clicking on a button on the hand controller and finished by clicking on another button when they believed the robot could successfully grab the object. User commands were given to the robot by moving and turning the hand controls in the roll/pitch/yaw direction.

After the user study, we asked the participants to rank their preferred control methods.

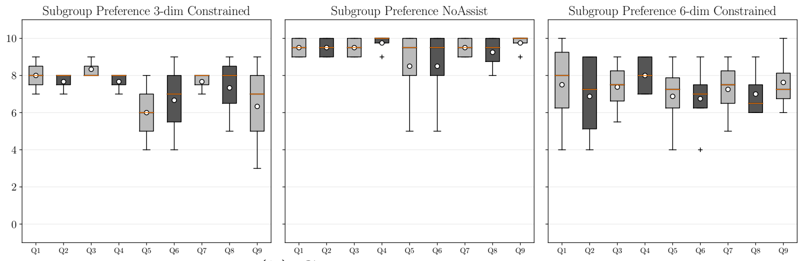
5.7 Results

The results were mixed between the *NoAssist* and the 6-dimensional *constrained* modes, each with four votes as their most preferred control method, with the rest of the participants voting for the 3-dimensional *constrained* mode. Participants who favored the *NoAssist* mode reported that it was more straightforward when controlling the gripper orientation. The robot gripper did not comply with their instructions when using other modes. They showed, on average, low effort and frustration levels for their preferred method on the NASA-TLX questionnaires and high scores for custom questionnaires which indicate more control authority such as "*I felt in control*" or "*The robot did what I wanted*". On the other hand, participants who favored the 6-dimensional *constrained* mode also showed on average the lowest frustration levels when using the mode and high scores for questions such as "*The robot and I collaborated well together*" and "*The robot's actions were acceptable*", but still gave positive scores for "I felt in control". This suggests that participants viewed the shared control as tolerable and reasonable.

Participants who favored the 3-dimensional shared control reported low frustration and high performance on the NASA-TLX questionnaires. They also showed high ratings for the custom questionnaire "*I was able to accomplish the tasks quickly*" and low levels of temporal demand, suggesting that they could finish the task naturally without being hurried or rushed.



(a) NASA-TLX



(b) Custom questionnaires

Figure 5.5: Qualitative results of the user study

We present the total number of collisions for all participants, average task duration, average path distance, and the distance between the gripper center and the object when the grasp command was triggered. There is little or no improvement in the quantitative results compared to the *NoAssist* mode, as shown in Table 5.2. Using repeated-measures analysis of variance (ANOVA) with the assistance mode as the factor, the 3-dimensional *constrained* mode outperforms the other modes with statistical significance ($f(2, 20) = 4.89$, $p = 0.01$) in task duration, but no statistical significance is observed for the path distance ($f(2, 20) =$

Mode	Collision	Duration (s)	Path (m)	Dist. at grasp (cm)
NoAssist	48	10.36 ± 7.86	0.66 ± 0.36	1.5 ± 2.5
3-dim	54	8.20 ± 3.56	0.68 ± 0.33	1.6 ± 2.2
6-dim	49	10.80 ± 6.38	0.63 ± 0.34	1.3 ± 2.2

Table 5.2: Quantitative results showing the total number of collisions, average task duration, path length, and distance between the gripper center and the object at grasp

0.60, $p = 0.56$) and the distance at grasp ($f(2, 20) = 0.30$, $p = 0.74$). The main reason for the weak improvement in the user experiment is likely due to the designed task being too short and easy to highlight the advantages of the proposed methods. It was easy enough with the *NoAssist* mode, which made it unnecessary to require assistance. Also, we noticed that the 6-dimensional *constrained* mode was slower than the other modes despite the same frame rate. The mode modified more joints simultaneously to match the desired end-effector pose rather than a few joints to only reach the desired end-effector position, which resulted in a slower end-effector motion after normalizing the configuration in Equation (5.3) to prevent sudden movements. Therefore, adjustments should be made to adapt its speed to follow the human input commands.

5.8 Discussion

This chapter showed how the method described in Chapter 4 can be extended to higher dimensional spaces. Although the method was justified through mathematical derivation, no sufficient improvement in the experiment with human subjects was found with the proposed method. One possible reason is the lack of complexity in the experiment design. The teleoperation task that the participants had to carry out was not complicated enough to require such a system; thus it lacked showing the benefits of the proposed system. Results from a user study with a teleoperation task that requires longer times and more precision would be necessary to justify the effectiveness of the proposed method.

As discussed in the previous chapter, a major limitation of the proposed method is the necessity of an optimal controller that is capable of generating optimal policies at each step. Controlling the robot in configuration space greatly increases the complexity of sampling, making it very difficult to obtain an optimal policy from regression by simply sampling policy across the state space using data-driven methods. Collecting policy samples from viable robot grasp trajectories covers only part of the state space as seen in Figure 3.8, and the rest of the state space includes infeasible joint configurations that may be inaccessible. This makes it difficult for the robot to recover once it enters an unseen state because the semi-autonomous agent starts to generate random actions. Thus, a more intelligent method is required to represent the semi-autonomous agent for a robust shared control system.

5.9 Summary

The method presented in the previous chapter is extended using Riemannian geometry to allow shared control in higher dimensional spaces. The proposed method augments the human inputs using the Riemannian metric to consider any cost terms or limitations the human operator may overlook when operating a redundant manipulator, which enables straightforward, low-workload teleoperation of redundant manipulators for complex tasks. In addition, we described how the curvature of the value function is approximated using finite differences. Although simulated teleoperation experiments showed an improvement in teleoperation performance, the pilot user study with human subjects did not show sufficient improvement in task performance.

Chapter 6

Learning to Arbitrate Using Disagreement Between Robot Sub-policies

Many prior works consider it crucial to predict the user's intent to promote teleoperation efficiency [99], [100] and to actively determine the level of assistance during teleoperation [20], [101]–[103]. However, accurately inferring the user intent can be a challenging task and unreliable predictions from an immature prediction system can aggravate the teleoperation experience of the user. The chapter presents an arbitration strategy for shared control teleoperation that is robust to uncertain predictions of the user's intent. While the previous chapters provide a direct strategy to combine human and semi-autonomous agents, this chapter proposes to dynamically tune the arbitration using reinforcement learning.

The work presented in this chapter has been published in:

- Y. Oh, M. Toussaint, and J. Mainprice, 'Learning to Arbitrate Human and Robot Control Using Disagreement Between Sub-Policies', *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5305–5311, 2021, © 2021 IEEE.

6.1 Introduction

Shared autonomy systems rely on two components:

1. **prediction** of user intent
2. **blending** user vs. autonomous controls

Trading off the autonomous robot and human control is usually directly based on the confidence scores associated with the intent prediction module. The chapter focuses on the blending strategy (i.e., arbitration), and leaves the prediction to other works.

The proposed approach (see Figure 6.1) learns an arbitration strategy from user interaction. This is generally beneficial as different users may have different preferences. Though tempting, it is difficult to learn an arbitration function using supervised learning. Indeed, the introduction of arbitration will inevitably shift the data distribution and hence the behavior of the human in return.

Several works have studied the problem of using reinforcement learning in shared control. However, to the best of our knowledge, no work has focused on directly learning the arbitration module while leveraging the

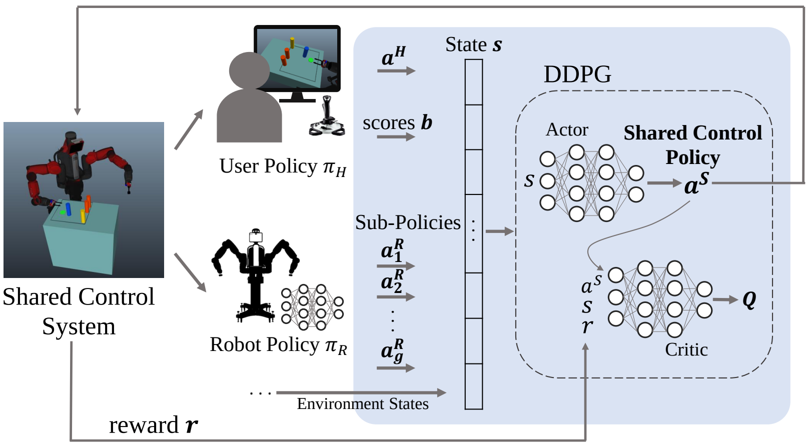


Figure 6.1: Overview of the method using the Deep Deterministic Policy Gradient algorithm

local geometry of the available sub-policies. This is desirable to limit the sample complexity by 1) limiting the search space by using readily available optimal actions, 2) making use of the disagreement between the sub-policies and the shared actions to create implicit feedback, resulting in dense rewards. To solve control tasks in the continuous action-space, we apply the modern actor-critic algorithm Deep Deterministic Policy Gradient (DDPG) [104].

The approach explicitly lowers the arbitration level at states when the robot encounters *decision points* (i.e., intersections, moving past an obstacle, deciding which object to grasp, shown in 6.2) to actively allow the user to guide the robot rather than providing assistance when the user's intent is not clear. This enables precise execution of the user's intentions and provides the user with a sense of control over the robot.

Decision points are identified by observing the modality of the marginal of the goal-conditioned sub-policies of the robot policy. Implicit feedback is then generated as a specific reward term in the reinforcement learning training process to tune the level of assistance through user interaction. Results of a simulated user study indicate that this method can be used to effectively learn an arbitration policy that recognizes when to hand over control authority while maintaining safe and accurate teleoperation despite an imperfect goal prediction system.

The main contributions of the chapter are the following:

- A generic arbitration learning formulation, which we cast as an actor-critic reinforcement learning problem
- Identification of "decision points" in the environment, where the robot encounters multiple options

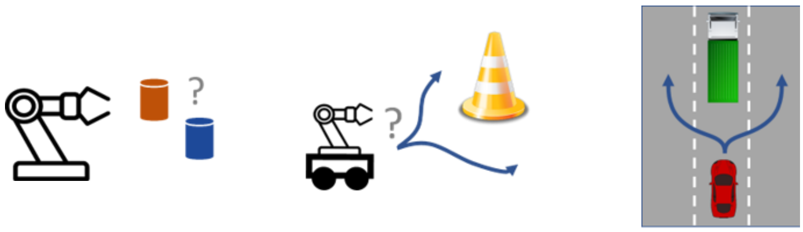


Figure 6.2: Examples of scenarios encountering a decision point where one has to choose an option among multiple options

- A new reward term that allows maximizing human control authority near decision points, by implicit feedback in continuous domains
- Quantitative results in simulation that show the effectiveness of the approach on a realistic robot manipulation scenario.

6.2 Related Work

Recent works have proposed the use of reinforcement learning for shared control (RLSC) in different application contexts. In [105], SARSA, an on-policy reinforcement learning algorithm, is used to produce an arbitration weight for a walking-aid robot. The reward function penalizes collision with the environment and promotes smoothness. In contrast to our work, no *implicit feedback* from the user is considered.

The notion of implicit feedback as a reward term is introduced in [106] to train an agent for an X-to-Text application. The error correction input (i.e., backspace) from the user is used to penalize the wrong actions taken by the autonomous agent.

Recently model-free RLSC methods have been proposed. The earliest work is [70], where the agent maximizes a combination of task performance and user feedback rewards using Deep-Q learning. In [107], the approach is extended to maximize human-control authority using residual policy learning. While general, model-free approaches do not consider the structure of the problem to maximize adaptability at the cost of sample complexity. Instead, our approach investigates the case where robot sub-policies are available. In our experiments, these correspond to goal-conditioned policies, but they do not have to be.

Fernandez and Caarls [108] developed an RLSC agent to learn a haptic policy using a type of implicit feedback, which compares the velocity the user applies with the forces exerted. While similar to our approach, their approach does not make use of an available optimal policy, which is instead learned online.

The ideas developed in RLSC have been generalized in [95], which formalizes a framework for assistive systems where the notion of human *empowerment* is introduced to denote reward terms that promote the operator's control over the state. The reward function we propose can be viewed as a type of empowerment proxy, specially designed to account for sub-policies in continuous action spaces, which are common in robot control problems.

6.3 Identifying Disagreement Between Sub-policies

6.3.1 Identifying Decision Points

We propose an arbitration strategy that allocates more control authority when the robot needs to decide, that is, at a *decision point.* A *decision point* is like a fork in the road, where one must choose an option among multiple presented options that lead to different possibilities in the subsequent results (examples in Figure 6.2). These points can be explicit, such as physically encountering an intersection of different paths or bypassing an obstacle, or implicit by having to make implications or decisions from multiple prospective goals. In human-robot collaboration tasks such as teleoperation, the human's intent is inferred by estimating the belief over the possible goals based on the robot's past trajectory or other cues given by the human. The true human's intent only becomes certain when the robot has passed the decision point. Rather than relying on early intent predictions and providing assistance near a decision point, the proposed method actively lets the user decide by lowering the assistance level. As a result, the human has more control authority and can direct the robot to their intended goal instead of fighting against the robot that provides assistance toward a wrong goal.

The decision points are identified by observing how the sub-policies diverge at a state. When the robot encounters an obstacle, the sub-policies generate different actions to maneuver around the obstacle. Similarly, when the robot is between two goal objects, the policies for each object produce two divergent paths that the robot can take. This property is expressed through observing its modality of the mixture model $\pi(\boldsymbol{a}|\boldsymbol{s})$

$$\pi(\boldsymbol{a}|\boldsymbol{s}) = \mathit{\Sigma}_{g \in \mathcal{G}} \pi(\boldsymbol{a}|\boldsymbol{s}, g) p(g) \tag{6.1}$$

which is a marginalized probability distribution of goal-conditioned sub-policies $\pi(\boldsymbol{a}|\boldsymbol{s}, g)$ with their probabilities $p(g)$, for each goal $g \in \mathcal{G}$. If all sub-policies are similar, then the marginalized distribution would be uni-modal. On the contrary, the marginalized distribution starts to become multi-modal if the sub-policies are diverse. Hence, a decision point exists.

6.3.2 Directional Deterministic Policies on the Plane

We consider the case where the action is a directional vector, which is a one-dimensional rotation angle along the axis that is surface normal to the plane. Therefore, it is natural to express the sub-policy $\pi(\boldsymbol{a}|\boldsymbol{s}, g)$ as a von Mises distribution. The von Mises distribution is a continuous probability distribution along a unit circle [109]:

$$f(\theta|\ \mu, \kappa) = \frac{1}{2\pi I_0(\kappa)} \exp^{\kappa \cos(\theta - \mu)} \tag{6.2}$$

for an angle θ. The mode μ and dispersion $1/\kappa$ are comparable to the mean μ and the variance σ^2 in the normal distribution. $I_0(\kappa)$ is the modified Bessel function of order 0.

For each sub-policy, the mode μ points in the direction of the action and the dispersion κ is associated with the goal prediction scores. The marginal distribution $\pi(\boldsymbol{a}|\boldsymbol{s})$ becomes a Finite von Mises Mixture model (FVMM).

The FVMM is **unimodal** when: 1) the robot is bypassing an obstacle or 2) attempting a precise grasp. In the first case, all sub-policies are co-directed (pointing in the same direction) to avoid collision with the obstacle. In the second case, one sub-policy is predominant among others due to a high confidence score. These situations indicate when the robot should take more control authority and provide greater assistance to enhance safety and improve efficiency.

The FVMM is **multimodal** when: 1) the robot is approaching an obstacle or 2) in between multiple goal objects. The diverse sub-policies lead to the resulting mixture distribution being multimodal with multiple modes, each facing different directions. This indicates the *decision point*, where the robot provides less assistance so that the human user's intentions are more conveyed in the robot's actions.

6.3.3 Modality Estimation

Carreira-Perpinan [110] proves that there exists at most M modes in any one-dimensional Gaussian mixture. To accurately determine the number of modes, a solution is to use a hill-climbing algorithm or gradient ascent to locate the local maxima in the distribution. These methods, however, can be exhaustive to compute at each time step of the teleoperation loop.

Rather, the distribution's modality is approximated by sampling several points from the FVMM and evaluating whether the value exceeds a certain threshold. According to Carreira-Perpinan [110], "it is clear that every centroid μ_m of the mixture must be near, if not coincident, with one of the modes, since the modes are contained in the convex hull of the centroids (as the mean is)"(p. 3), the modes are located near the means of each distribution. Furthermore, due to the superposition of the individual distributions, a unimodal mixture model exhibits higher values at the modes. To estimate the multi-modality of the FVMM, the value of the FVMM is sampled at the locations of each mean of the sub-policy, which is then compared to a threshold value.

6.4 Reinforcement Learning for Arbitration

The reinforcement learning problem is modeled as a Markov Decision Process (MDP).

The state space $\mathcal{S}$ includes user action $\boldsymbol{a}^H \in \mathcal{A}^H$ and autonomous robot actions $\boldsymbol{a}_g^R \in \mathcal{A}_g^R$ conditioned on a goal $g \in \mathcal{G}$. The shared control agent generates an arbitrated action $\boldsymbol{u} \in \mathcal{U}$, executed by the robot. Each goal is associated with a confidence score or belief b_g to describe the likelihood of that goal matching the user's intentions.

The shared control agent learns a policy $\mu(\boldsymbol{u}_t|\boldsymbol{s}_t)$ that outputs $\boldsymbol{u}_t$, given the following as input:

$$\boldsymbol{s}_t = \{\boldsymbol{x}, \boldsymbol{a}^H, \boldsymbol{a}_1^R, \ldots, \boldsymbol{a}_g^R, b_1, \ldots, b_g\}$$

where $\boldsymbol{x}$ denotes the environment states (e.g., gripper position, distance to goals, distance to obstacle).

6.4.1 DDPG for Arbitration Learning

The Deep Deterministic Policy Gradient (DDPG) algorithm [104] is used to train the policy μ, as shown in Algorithm 4. Adjustments are made that distinguish the method from the vanilla DDPG, as discussed in the following paragraphs.

Algorithm 4: DDPG for Arbitration Learning

```
Init:
Load actor $\mu(\boldsymbol{s}|\boldsymbol{\theta}^{\mu})$, critic $Q(\boldsymbol{s},\boldsymbol{a}|\boldsymbol{\theta}^{Q})$ weights $\boldsymbol{\theta}^{\mu}$, $\boldsymbol{\theta}^{Q}$
Initialize target network $\mu'$, $Q'$ with weights $\boldsymbol{\theta}^{\mu'} \leftarrow \boldsymbol{\theta}^{\mu}$,$\boldsymbol{\theta}^{Q'} \leftarrow \boldsymbol{\theta}^{Q}$
Initialize replay buffer $\boldsymbol{R}$, episode buffer $\boldsymbol{R}_E$
for episode $= 1, M$ do
    for $t = 1, T$ do
        Observe user action $\boldsymbol{a}_t^H$
        foreach $g \in \mathcal{G}$ do
            Query sub-policy action $\boldsymbol{a}_{t,g}^R$
            Compute score $b_{t,g}$
        Select action $\boldsymbol{u}_t = \mu(\boldsymbol{u}_t|\boldsymbol{\theta}^{\mu}) + \mathcal{N}_t$
        Execute $\boldsymbol{u}_t$, observe next state $\boldsymbol{s}_{t+1}$, done $d_t$
        Store transition $(\boldsymbol{s}_t, \boldsymbol{u}_t, \boldsymbol{s}_{t+1}, d_t)$ in $\boldsymbol{R}_E$
        if $t > n$ then
            Sample a random mini-batch of $N$ transitions
             $(\boldsymbol{s}_i, \boldsymbol{u}_i, r_i, \boldsymbol{s}_{i+1}, d_t)$ from $\boldsymbol{R}$
            Update $\boldsymbol{\theta}^{\mu}$, $\boldsymbol{\theta}^{\mu'}$, $\boldsymbol{\theta}^{Q}$, $\boldsymbol{\theta}^{Q'}$

    Observe true goal $g^*$
    foreach $o \in \boldsymbol{R}_E$ do
        Compute reward $r_i = \mathit{Reward}(o, g^*)$
        Store transition $(\boldsymbol{s}_i, \boldsymbol{u}_i, r_i, \boldsymbol{s}_{i+1}, d_t)$ in $\boldsymbol{R}$
```

Sub-policies A sub-policy refers to the autonomous robot policy that is conditioned on a variable, i.e., a goal. Each robot action $\boldsymbol{a}_g^R$ is sampled according to the goal-conditioned sub-policy $\pi(\boldsymbol{a}_g^R|\boldsymbol{x}, g)$ for each prospective goal g. $\pi(\boldsymbol{a}_g^R|\boldsymbol{x}, g)$ is considered as a deterministic policy, as described in Section 3. That is,

$$\boldsymbol{a}_g^R = \pi(\boldsymbol{a}_g^R|\boldsymbol{x}, g) \tag{6.3}$$

The score b_g denotes how likely an object is the target goal, and it can be described as the posterior probability given a history of observed features $\xi_{S \to U}$ [20], [102].

$$b_g = P(g|\xi_{S \to U}) \tag{6.4}$$

All sub-policies along with their scores are considered at every time step when learning the arbitration, rather than using only the sub-policy with the highest confidence score. This allows the arbitration network to take into account the uncertainty of the intent prediction.

Hindsight goal labeling The true user's intended goal g^* is verified to the robot once the episode is terminated. Therefore, the reward for each state-action pair cannot be computed in real time without knowing g^*. The state-action pairs are stored in a separate episode buffer $\boldsymbol{R}_E$ and their rewards are computed in hindsight once the episode terminates. The off-policy characteristic of the DDPG algorithm enables the update of the replay buffer $\boldsymbol{R}$ in a delayed manner, since the network samples a mini-batch from $\boldsymbol{R}$ to update its parameters. The replay buffer $\boldsymbol{R}$ is filled with transitions $(\boldsymbol{s}_i, \boldsymbol{u}_i, r_i, \boldsymbol{s}_{i+1}, d_t)$ of n episodes before updating the network to compensate for the delayed population of $\boldsymbol{R}$.

6.4.2 Reward Function Using Disagreement

The goal is to learn an arbitrated, shared control policy using reinforcement learning and commands from both the human and the robot. The

Algorithm 5: Reward Function

```
Input: $\boldsymbol{s}_t, \boldsymbol{a}^S_t, g^*$
Output: $r_t$
Function Reward($\boldsymbol{s}_t, \boldsymbol{a}^S_t, g^*$):
    Compute $r_{env}$
    Means $\mathcal{M} = \{\boldsymbol{a}^R_{t,g}\}_{g=1}^G$
    Dispersions $\mathcal{K} = \{b_{t,g}\}_{g=1}^G$
    Construct FVMM $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t) = \frac{1}{2\pi I_0(\kappa)} \exp^{\kappa \cos(\boldsymbol{a}_t - \mu)}$
    if multimodal then
        $r_{agree} = -||\boldsymbol{a}^H_t - \boldsymbol{u}_t||^2$
    else
        $r_{agree} = -||\boldsymbol{a}^R_{t,g^*} - \boldsymbol{u}_t||^2$
    $R_{speed} = -\big|\, ||\boldsymbol{a}^H_t|| - ||\boldsymbol{u}_t||\, \big|$
    $r_{agree} \leftarrow r_{agree} + R_{speed}$
    $r_t \leftarrow r_{agree} + r_{env}$
    return $r_t$
End Function
```

policy is learned by using implicit feedback in the form of a specific reward term by reasoning on the FVMM modality, as opposed to the user providing external feedback to adjust the level of assistance.

Algorithm 5 provides a summary of the overall algorithm used to compute the reward. The reward function consists of a reward term based on an agreement between the policies r_{agree} and r_{env} to punish or reward certain acts taken when interacting with the environment.

$$R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = r_{agree} + r_{env} \tag{6.5}$$

The value *Renv* includes a negative reward when the robot collides with the obstacle (-10) or the workspace border (-2) and positive reward when the gripper reaches the desired object (+10). By penalizing policy differences between agents based on the FVMM's modality, r_{agree} serves as a type of implicit feedback. When the FVMM is multimodal, r_{agree} is calculated using the negative Euclidean distance between the human agent $\boldsymbol{a}^H$ and the shared control action $\boldsymbol{u}$. On the other hand, if the FVMM is unimodal, r_{agree} is determined by computing the negative Euclidean distance between the arbitrated action $\boldsymbol{u}$ and the robot action $\boldsymbol{a}^R_{g^*}$. Note that $\boldsymbol{a}^R_{g^*}$ denotes the robot sub-policy for the actual aim g^* that the user intended, which is available once the episode has concluded. The actions are normalized before computing the L2 norm, and the speed difference is deducted from r_{agree} to adjust the policy's speed to that of the human operator (referred to as R_{speed}).

6.5 Experiments

The section describes the experimental procedures for building the neural network model and assess how well the trained shared control agent performed. The following hypotheses were proposed:

- When the prediction is uncertain, the shared control agent can learn to delegate more control authority to the human and provide more assistance when the goal is clear.
- The custom reward function enables faster training convergence as well as safe and accurate execution towards the user's intended goal.

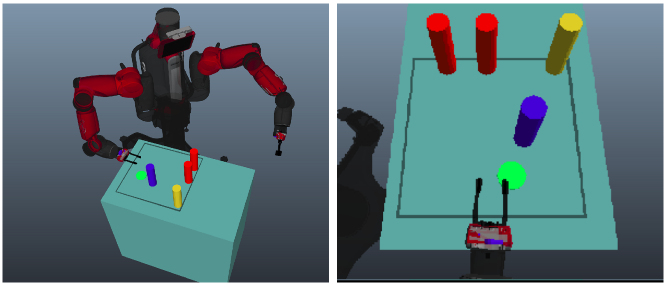


Figure 6.3: The simulated environment and an example image of the visual interface for human operators

Environment details We consider a teleoperation task in which the human agent controls the robot manipulator's end-effector to move toward and grab the goal object while avoiding an obstacle. Figure 6.3 shows the Baxter robot and a table with objects that comprise the simulated environment. The robot's end-effector can be moved on a parallel plane above the 50 cm $\times$ 50 cm table workspace. Graspable cylinders (colored yellow and red) are placed toward the workspace's edge, while the obstacle is positioned at a random point near the center (blue cylinder). The gripper position is initialized to begin behind the obstacle to ensure that the obstacle is encountered when the robot arm reaches for the desired object. The simulated environment does not include physical collisions.

Obtaining robot sub-policies Each robot sub-policy $\pi(\boldsymbol{a}_g^R|\boldsymbol{x}, g)$ is represented as a neural network policy, described in Section 3.3.1.

Predicting user intent We consider a simple intent inference model proposed in Dragan and Srinivasa [20], based on distance and direction towards the goal.

The score b_g describes how likely the object is the goal object. Here, we use the posterior probability value for each object. The goal object g^* can be chosen as the object that maximizes the posterior probability given a history of observed features $\xi_{S \to U}$ [20], [102].

$$g^* = \arg\max_{g \in \mathcal{G}} P(g|\xi_{S \to U}) \tag{6.6}$$

The $P(g|\xi_{S \to U})$ is computed using the sum of squared velocity magnitudes as the cost function.

Simulating human users In the experiments, we simulate human policies to replace the interaction with the reinforcement learning agent during training and evaluation:

- *Noisy* user as a sub-optimal noisy policy
- *Straight* user executes a policy that points directly to the goal
- *Biased* user misperceives the goal location due to imprecise perception (e.g., perceiving the goal closer than it actually is)

Both *Noisy* and *Biased* users make use of the policy described in Section 3.3.1. The *Noisy* user adds random noise at each time step to the policy, whereas the *Biased* user adds a random offset to the goal position.

Training details The 1seven-dimensional input $\boldsymbol{s}_t$ to the DDPG arbitration module contains the user command, all sub-policies and their scores, and environmental states (end-effector position, distance to objects, and distance to the obstacle). The input $\boldsymbol{s}_t$, except the user input, is processed through three dense layers of 32 units each, followed by a layer of two units. This portion of the network is referred to as the "head".

The head is then concatenated with the user input and passed through three dense layers of 16 units, producing a continuous two-dimensional action that represents the arbitrated action. This is known as the actor network.

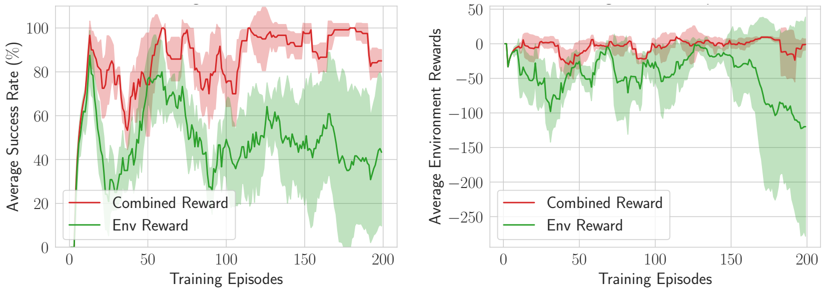
The same head structure is used in the critic network. The output of the critic network's head is concatenated with the user action and the arbitrated action and processed through two dense layers of 128 units. The critic network's output is a one-dimensional value that calculates the expected return following an action at a given state.

6.6 Results

6.6.1 Effects of Penalizing Disagreement During Training

We compare the training outcomes between the proposed combined reward function ($r_{agree} + r_{env}$) that also penalizes disagreement and the environment reward function (r_{env}) to see if the suggested reward function promotes conversion during training. Figure 6.4a shows that after 110 training episodes, the model with the combined reward function had a high success rate ($> 90\%$), whereas the success rate of the model with only the environment reward function dropped over training. Considering that our goal prediction model is imprecise, we can presume that the model with the combined reward function has learned to delegate more control authority to the user at decision points where the FVMM is multimodal, as opposed to relying exclusively on the prediction and guiding the robot toward the goal with the highest confidence.

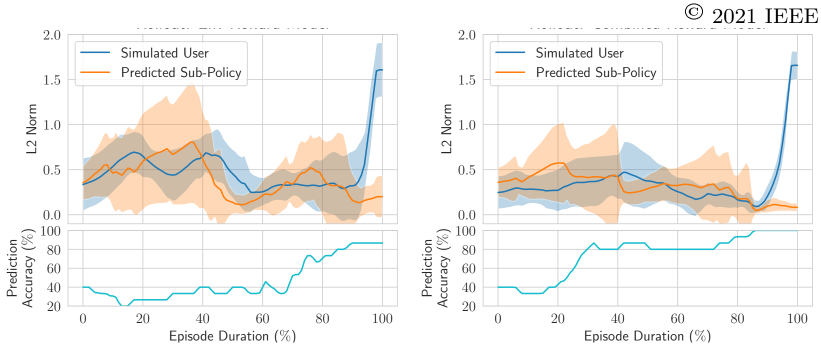
The combined reward model also demonstrated safe robot motion throughout the episode by avoiding collisions with the environment. The agent earns negative rewards at each time step if the robot collides with the obstacle or goes out of bounds, and a positive reward if it successfully reaches the correct goal object. When an episode is completed without any collisions, the agent earns the maximum envi-



(a) Averaged success rate of episodes throughout training

(b) Comparison of r_{env} received throughout training

Figure 6.4: Comparison between the policies with different reward functions during training



(a) Environment Reward Model (b) Combined Reward Model

Figure 6.5: Performance of the trained models during an episode

ronment reward ($\max r_{env} = 10$). Figure 6.4 shows the performance of the models with different reward functions during training. The metrics are computed by averaging across 12 trained models and smoothed using a moving average (window size: 10 episodes).

As seen in Figure 6.4b, the combined reward model agent remained stable across the training episodes and eventually gained positive rewards after 170 episodes. This implies that the agent has learned to allocate more control authority in areas where the FVMM is unimodal, i.e., near the obstacle or while approaching a grasp.

6.6.2 Demonstration Using Trained Agents

Using the trained models, we demonstrated episodes using the trained agents to observe their behavior. The agents with the highest cumulative reward after training were selected and gathered demonstrations from

15 randomly generated episode settings.

Figure 6.5a and Figure 6.5b show demonstrations of the trained models, each averaged across 15 random setting episodes. The figures show the difference between the actions of the trained agents, the human user, and the autonomous robot policy by comparing the Euclidean norm (L2 norm). The blue line indicates the L2 norm between the human action and the arbitrated action and the orange line indicates the L2 norm between the action from the sub-policy with the highest score (predicted sub-policy) and the arbitrated action. Here, the simulated user with the *straight* policy was used to demonstrate the human user, and the sub-policy with the highest goal confidence was used as the robot policy.

As seen in Figure 6.5b, the trained agent with the combined reward function flexibly assigns the control authority depending on the situation. At the beginning of the episode (duration $< 30\%$), the robot manipulator approaches the first decision point, and the agent should decide which way to get around the obstacle. The policy of the trained agent was closer to the human policy despite the low prediction accuracy. In the middle of the episode when the robot gripper must avoid colliding with the obstacle (duration $40\% \sim 60\%$), the trained agent's policy was closer to the predicted sub-policy, indicating more robot assistance. Towards the end of the episode (duration $60\% \sim 80\%$), the robot manipulator approaches the second decision point for determining the target object. The agent's policy resembles the human policy, and the human acquires more control authority. By the end of the episode, the agent's policy is closer to the predicted sub-policy, which makes grasping the object easier.

The behavior was not shown in the demonstration with the environment reward model. As seen in Figure 6.5a, the L2 norms for the environment reward function agent were comparably higher than those for the combination reward. The arbitrated policy was neither near to the human nor the predicted sub-policy, indicating that it generated policies that do not conform with both policies.

6.6.3 Comparison Between Different Simulated Users

Table 6.1 shows the results of demonstrations for all user modes based on the learned policies: *Noisy* user with two different noise sizes, the *Straight* user, and the *Biased* user. Direct control refers to operating the robot without any assistance.

Although the combined reward policy did not result in the shortest travel distance compared to other assistance methods, it did result in more successful episodes and fewer collisions for all users than the environment reward policy. A compromise in the travel distance would have been required to safely avoid collision from a distance. When compared to the direct control method, the improved success rate, increased travel distance, and decreased number of collisions in the *Straight* user proves that the combined reward policy curved the user's straight course to avoid collision. For the *Noisy* user, the success rate and the number of collisions did not improve drastically since the *Noisy*

Table 6.1: Rollouts with simulated users over 15 episodes

User Mode	**Assistance Method**	**Success**	**Travel Dist.** (cm)	**Collisions**
Noisy 0.5	Combined Reward	**14/15**	53.68 ± 3.17	**0/15**
	Environment Reward	12/15	**53.06 ± 2.64**	2/15
	Direct Control	**14/15**	59.12 ± 16.09	1/15
Noisy 1.0	Combined Reward	**14/15**	59.41 ± 18.70	**0/15**
	Environment Reward	12/15	**53.99 ± 4.06**	2/15
	Direct Control	**14/15**	71.16 ± 13.23	**0/15**
Straight	Combined Reward	**13/15**	51.63 ± 1.65	**2/15**
	Environment Reward	10/15	54.02 ± 2.38	9/15
	Direct Control	11/15	**49.19 ± 2.17**	7/15
Biased	Combined Reward	**8/15**	**54.60 ± 11.81**	**0/15**
	Environment Reward	6/15	62.54 ± 21.46	1/15
	Direct Control	3/15	65.27 ± 22.21	**0/15**

user policy is created based on optimized trajectories that guarantee a successful grasp. However, the shorter travel distance in the combined reward policy implies that the combined policy compensated for the noise. The combined reward policy was advantageous for the *Biased* user as well, as it corrected the imprecise user commands to grab the target object. The results indicate that the r_{agree} term in the combined reward function leads to the successful learning of a policy that appropriately supports the user in achieving the objective while ensuring safety.

6.7 Discussion

As the arbitration policy was learned through interactions with simulated human behaviors, it would be interesting to investigate how the system reacts to actual humans. Further research can focus on whether transfer learning is possible, i.e., training the agent with simulated behavior and then fine-tuning it with actual human interaction to reduce the burden of collecting user interaction. The key is to find a strategy that minimizes agent training with real human interaction data without compromising system effectiveness. Also, it would be interesting to study whether the arbitration agent can pick up on different behaviors of the users, such as their adaptability to assistance, through interaction. This would imply that the agent can be fine-tuned to provide suitable assistance depending on the user's reaction by calibrating the agent using a few interactions with the actual user. Overall, this chapter describes how to learn arbitrated policies directly through user interaction in a continuous state space, and lays the groundwork for learning arbitration directly through data-driven methods.

6.8 Summary

A framework is proposed to learn an arbitrated policy for shared control using DDPG that can dynamically hand over control authority to the user at a decision point. The decision points are defined by looking at the diversity of the sub-policies and constructing a Finite von Mises

Mixture model to observe the modality of the distribution. Experiment results indicate that incorporating implicit feedback allows the agent to effectively learn when to hand over control authority while maintaining safe and accurate teleoperation despite an imperfect goal prediction system.

Chapter 7

A System for Traded Control Teleoperation Combining Manipulation and Perception

In the previous chapters, I discussed various methods of blending control between the human and the autonomous agent to provide a single robot command. With shared control, human-intended actions can be optimally executed at the servo level, because human control is modified by autonomous agent control at each step. However, for fast and intuitive teleoperation, it may not be necessary to directly mix the two controls at each time step.

Assuming a perfect world where intelligent systems can better predict user intent and robots are capable of planning robust motions that match their intent, human operators may not need to provide continuous commands but let the autonomous agent take action when the intent is clear. Instead of sharing control at the servo-level, this concept moves toward task-level arbitration where the human and the autonomous agents share tasks at a higher level. In this chapter, the approach is based on traded control, in which the operator specifies only the target object to initiate the subtask and the robot executes the task using autonomous motion planning.

The work presented in this chapter has been published in:

- Y. Oh, T. Schäfer, B. Rüther, M. Toussaint, and J. Mainprice, 'A System for Traded Control Teleoperation of Manipulation Tasks Using Intent Prediction from Hand Gestures', *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2021, © 2021 IEEE.

7.1 Introduction

When using interfaces such as hand gesture controllers, direct teleoperation is often nearly impossible as the mismatch between the kinematics of the robot and hand gestures is too large to produce fluid movements. In addition, the noisy output from the hand gesture makes it difficult to perform precise manipulation. However, motion-tracking controllers such as hand gesture controllers can provide unrestricted and intuitive motion control.

In this chapter, we demonstrate a complete *traded control* teleoperation implementation, where the user specifies the task objectives and executes the motion autonomously. Contrary to the previous ap-

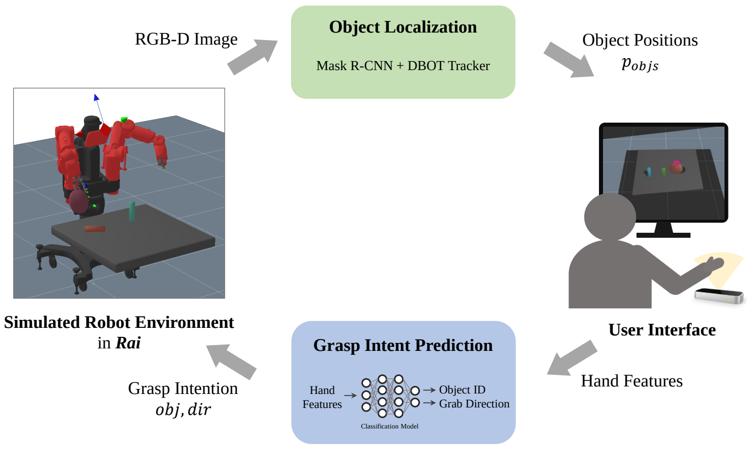


Figure 7.1: Overview of the traded control teleoperation system

proaches, it does not blend between direct and autonomous control at each time step but allows the robot to take control when the intent is clear.

Hand gestures are used to specify the goal objects for a sequential manipulation task; the robot then autonomously generates a grasping or a retrieving motion using trajectory optimization. The perception module identifies the objects present in the robot workspace and the intent prediction module which object the user likely wants to grasp. It relies on the model-based tracker to accurately track the 6-dim pose of the objects and makes use of a state-of-the-art learning-based object detection and segmentation method, to initialize the tracker by automatically detecting objects in the scene. Goal objects are identified from user hand gestures using a trained multi-layer perceptron classifier. The goal of this chapter is not to present novel components but to present a teleoperation system that makes use of existing models, for practical applications.

We present and evaluate all components needed for the system: 1) a perception pipeline capable of identifying and tracking objects, 2) an intent estimation system that can identify which objects to grab and how, 3) a motion planning system that can produce accurate manipulation motion according to the human operator's intent.

We assess the accuracy of the different modules on dedicated tasks. The performance of the object localization and tracking module is evaluated on several objects in simulation. The accuracy of the grasp intent prediction module is tested using a dataset of trajectories. Finally, we present results using our grasp intent inference module, where various users are simulated by modifying user-demonstrated trajectories that were collected using the hand gesture controller.

The main contributions of the chapter are summarized as the following:

- A teleoperation system capable of traded control using hand gestures through prediction of the operator's intent
- A solution for automatic initialization of an existing object tracking module using Mask R-CNN [112]
- Simulated user experiment assessing the capacity of our grasp intent prediction module to perform teleoperation of pick and place motions

7.2 Related Work

7.2.1 Traded Control in Teleoperation

Traded control is a discrete switching mechanism between high-level robot autonomy and low-level control depending on predefined circumstances. It is also referred to as *control switching*, as the system allocates all-or-none assistance rather than a blended spectrum between user and robot controls. The operator initiates a sub-task or behavior for the robot, and the robot performs the sub-task autonomously while the operator monitors the robot [64], [66]. Bohren et al. [113] showed that intent-based traded control can improve teleoperation performance and alleviate difficulties in high-latency teleoperation scenarios.

7.2.2 Hand Gesture Recognition for Robot Control

The Leap Motion controller (Ultra Leap, https://www.ultraleap.com/) is a consumer-grade, marker-less motion capture sensor that tracks hand gestures and finger movements up to 200 Hz. Weichert et al. [114] showed that its accuracy is below 2.5 mm; however, the controller shows inconsistent performance due to its limited sensory range [115]. Nevertheless, its simplicity and its capability to track the hand in 6-Dof are the reasons for its application.

Prior works used deep learning to improve the accuracy of gesture recognition, such as support vector machines (SVM) and random forests [116], or neural networks using radial basis functions (RBF) [117]. Similar to [118], we propose to train a gesture classifier (i.e., which object is intended) for hand motion recognition rather than mapping hand features directly to robot configurations. Achieving higher accuracy is easier on classification than regression (i.e., predicting accurate positions) tasks, which is one of the justifications for our traded control approach.

7.2.3 Depth Based Object Tracking (DBOT)

We utilize the implementation of depth-based object tracking methods described in [119] ("particle tracker") and [120] ("Gaussian tracker") to acquire the 6-dimensional pose of objects during teleoperation. Compared to recent learning-based methods such as PoseCNN [121] and DenseFusion [122], the methods take a model-based approach.

The particle tracker in DBOT tracks objects by computing a posterior distribution over the object using a dynamic Bayesian network for inference [119], while the Gaussian tracker improves the performance of a Gaussian filter using a robustification method as well as reducing the filter's computational complexity [120]. This approach has the advantage of being robust without requiring any extra tuning or pre-training.

7.3 Manipulation: Hand-Gesture-Based Robot Control

The user issues commands by performing reach-and-grasp gestures with the right-hand as if they were naturally reaching and grasping an object in front of themselves, while viewing the environment through the robot's perspective. We use the Leap Motion controller to capture the hand movement at a frame rate of 180 Hz and access its features, such as the palm position, hand direction vector, the normal vector to the palm, and roll/pitch/yaw rotation of the hand, through ROS topics.

The gripping positions vary substantially since the user is actually reaching for an invisible object. Figure 7.2 shows actual grasp positions from a single participant who repeatedly performed reach-and-grasp trajectories for a single object, where the blue dot and red dots in the figure represent the user's attempts to grab the side and top of the cylinder, respectively.

As it is evident that using real user commands (i.e., position commands) for accurate teleoperation would be challenging, we overcome this issue by adopting a traded control paradigm and utilizing the user's intent instead of their actual controls.

7.3.1 Traded Control

Instead of using a continuous shared control paradigm, we propose a traded control strategy to overcome the inconsistent hand-tracking performance. This additionally alleviates the issues caused by the physical mismatch between the human arm and the robot manipulator.

After detecting the graspable objects in the environment using the method described in Section 7.4, we infer the user's intent with respect to the target object and the grasping direction. The robot executes the reach-and-grasp motion as soon as the intent is identified. The user retains control authority over the task by deciding in which order to grab the set of objects. In other words, the human is responsible for making high-level decisions while the robot handles low-level control and motion planning.

7.3.2 Grasp Intent Prediction

We use supervised learning to train a multi-layer perceptron (fully connected dense layers) to classify the target object and the grasp direction. As seen in Figure 7.3, we assume a fixed number of items (m=3) and two possible grasp directions (top grasp/right grasp, n=2).

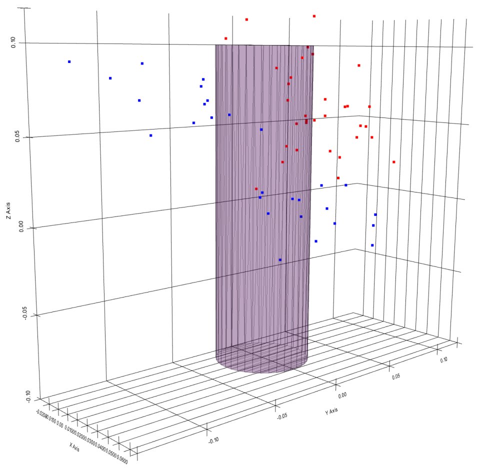
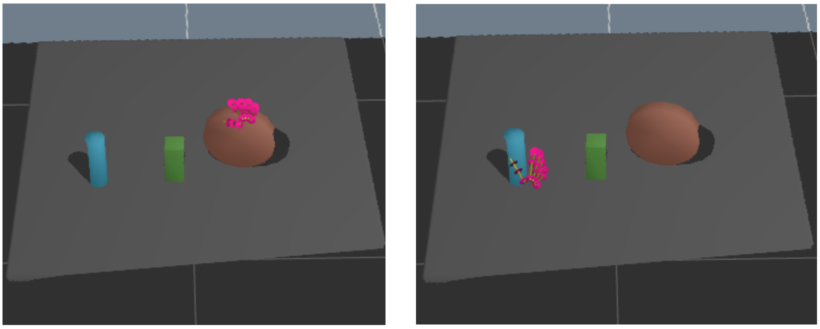


Figure 7.2: Varied grasp positions when virtually grasping the cylinder using the Leap Motion sensor



(a) Disk top grab (b) Cylinder right grab

Figure 7.3: User interface for reach and grab motion in a setting with three objects

The input of the network consists of eight features: distances from hand to objects, the x-component of the hand position, the x-component of the hand direction, the x, y-components for the palm normal vector, and the y-rotation of the hand. The model generates class labels and consists of three dense layers of 64 hidden units attached to two separate layers of two units.

7.4 Perception: Object Tracking Pipeline

The object tracking pipeline automatically tracks the 6-dimensional pose of rigid objects. We utilize the already existing object tracking library DBOT [119], [120] but provide a solution to avoid manual initialization, which is required when using the libraries.

The pipeline includes two components: object tracker initializer

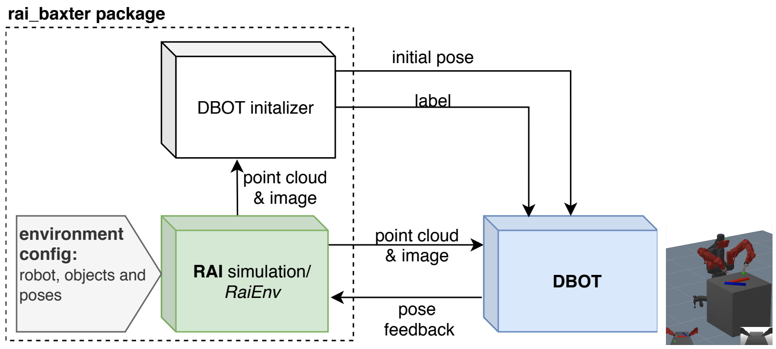


Figure 7.4: Overview of the object tracking pipeline and an image of the simulated robot environment

(DBOT initializer) and the object tracker (DBOT tracker), as shown in Figure 7.4. The DBOT initializer obtains camera images of the surrounding environment and predicts both the initial pose (position + orientation) and semantic label of observed objects. Given the initial pose and label of the object, the DBOT tracker constantly tracks the object. The following sections provide detailed descriptions of each component.

7.4.1 DBOT Initializer: Automated Object Tracker Initialization

The depth-based object tracking library (DBOT) follows the general model of a Bayes filter, which predicts the current object pose given the previous pose based on observations. However, the authors stated that the initialization of the tracking was outside the scope of their study.

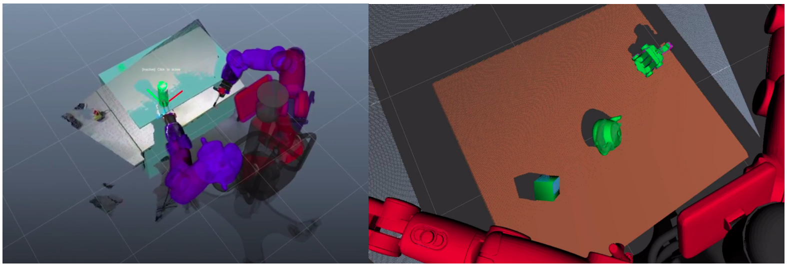
In practice, the initialization is done manually by specifying the object pose using an interactive marker.

We utilize the state-of-the-art instance segmentation method Mask R-CNN [112] to automate the initialization procedure by identifying the masks and labels. The label identifies the object for tracking while the mask is used to segment the depth image and extract depth pixels corresponding to the object.

After segmenting the depth pixels, the 6-dimensional pose of the object is computed using point cloud registration. The matching mesh model of the object is loaded using the labels from the Mask R-CNN, and points are sampled from the mesh to create a reference point cloud for registration. We perform rigid registration using the coherent-point-drift (CPD) algorithm [123] and obtain a 4×4 homogeneous transformation matrix. This estimated pose is referred to as *mesh_pose*.

7.4.2 DBOT Tracker

Once the DBOT trackers have been successfully configured, the refined poses can be subscribed to execute precise object manipulation, as shown in Figure 7.5. We tested the DBOT up to seven objects per CPU core at 10 Hz. By utilizing GPU, the performance can be enhanced further.



(a) Tracking a real environment (b) Tracking simulated objects

Figure 7.5: Demonstration of the DBOT tracker

7.5 Experiments

7.5.1 Simulated Robot Environment

We create a simulated environment using the RAI[1] interface. RAI provides a physics-simulated environment as well as a robot motion optimization solver using k-Order Motion Optimization (KOMO) [124] The interface offers simple and straightforward approaches for defining cost objectives for robot motion optimization problems, by listing the optimization objects that represent cost terms or in/equality constraints.

As shown in Figure 7.6, the environment consists of a Baxter robot with graspable items on a table. A virtual depth camera is added to the Baxter's head display to provide first-person view images.

[1] https://github.com/MarcToussaint/rai

7.5.2 Transfer Learning on a Synthetic Dataset

We tune the Mask R-CNN using transfer learning to detect custom objects that are used in the simulation. A dataset of images that include the following objects is gathered to train the network: cube, sphere, toy, teapot, cup, jug, and bowl. The objects are shown in Figure 7.7.

Simulated images lack noise, shadows, inconsistent lighting conditions, and varying textures. The images are modified to guarantee that the network can generalize to real or noisy images. Modifications include flipping, affine transformation, light contrast, blur/sharpening, and color adjustments. Additionally, the dataset is collected with random robot arm joint positions included in the image, so that the Mask R-CNN network can learn to ignore the appearance of robot arms during

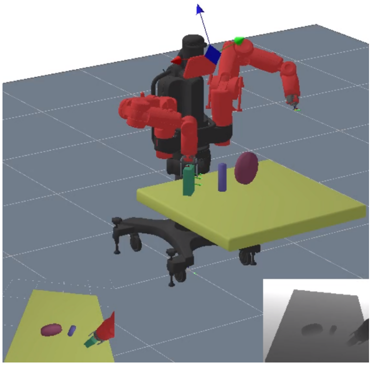
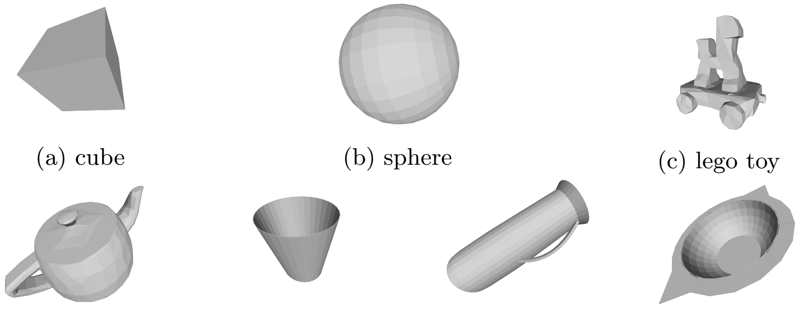


Figure 7.6: Simulated robot environment using RAI



(a) cube (b) sphere (c) lego toy

(d) teapot (e) cup (f) jug (g) bowl

Figure 7.7: Objects used to train Mask R-CNN

object detection.

The detection rate of the retrained Mask R-CNN network ranged between 92% to 95%, with an error rate between 0.16% to 0.74% based on different training settings, such as the dataset size (107 to 50K samples) and the number of epochs (10 to 80 epochs). The definitions of the detection rate and error rate are shown in Equations (7.1) and (7.2)

$$\text{detection rate} = \frac{\text{\# of successfully classified objects}}{\text{\# of objects}} \tag{7.1}$$

$$\text{error rate} = \frac{\text{\# of wrongly classified objects}}{\text{\# of classified objects}} \tag{7.2}$$

The model used to evaluate the pose initialization results displayed in Table 7.1 was trained using 40 epochs of unmodified images. Out of 4798 sample images that were excluded during the training of the network, the detection rate of the retrained Mask R-CNN was 92.73%, with an error rate of 0.16%.

The influence of each network parameter was found to be minor when comparing the outcomes of the entire initialization pipeline. As long as the detected label is accurate, a decent mask is sufficient for computing the initial pose. Thus, we do not present specific results for each configuration.

7.5.3 Accuracy of the Initial Object Pose Estimation

Using the average distance (ADD) metric proposed in [125], we evaluate the accuracy of the estimated pose. The average distance computes the mean of the pairwise distances of all model points $\boldsymbol{x}$ between two 3D models with ground truth pose (translation $\boldsymbol{t}$, rotation $\boldsymbol{R}$) and estimated pose (translation $\hat{\boldsymbol{t}}$, rotation $\hat{\boldsymbol{R}}$):

$$\text{ADD} = \frac{1}{m} \sum_{\boldsymbol{x} \in \mathcal{M}} ||(\boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}) - (\hat{\boldsymbol{R}}\boldsymbol{x} + \hat{\boldsymbol{t}})|| \tag{7.3}$$

where m is the number of 3D model points and $\mathcal{M}$ is the set. For symmetric objects, the average distance is calculated using the closest point distance:

$$\text{ADD-S} = \frac{1}{m} \sum_{\boldsymbol{x}_1 \in \mathcal{M}} \min_{\boldsymbol{x}_2 \in \mathcal{M}} ||(\boldsymbol{R}\boldsymbol{x}_1 + \boldsymbol{t}) - (\hat{\boldsymbol{R}}\boldsymbol{x}_2 + \hat{\boldsymbol{t}})|| \tag{7.4}$$

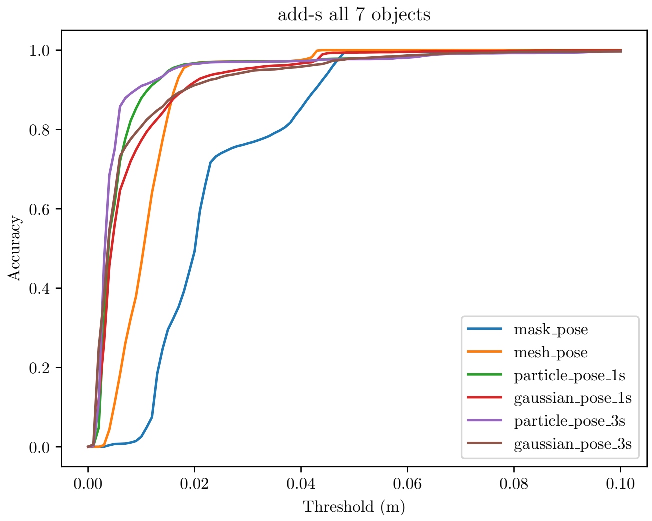


Figure 7.8: Accuracy-threshold curve of the average ADD-S for seven objects with a maximum threshold of 10 cm

Following recent work [121], [122], we report the area under the curve (AUC) by computing the pose accuracy while increasing the threshold up to 0.1 m. We also measure the proportion of AUC below a 2 cm threshold, which is the lowest tolerance for robot grasping.

Table 7.1 and Figure 7.8 show the accuracy of object tracking approaches, including the method described in Section 7.4.1. In Table 7.1, the upper table shows the accuracy of the position and the orientation. The lower table shows the accuracy computed using the area under the ADD-S curve (AUC) and the <2cm metric. The *mask_pose* (indicated as *mask* in Table 7.1) refers to the center position of the masked point cloud, and it is used as a baseline when comparing translation to other methods. However, the *mask_pose* does not have rotation, which explains the poor AUC. The *mesh_pose*, which was calculated using rigid registration, is slightly more precise in translation than the *mask_pose*, but it is sufficient to provide an initial rotation estimate for DBOT initialization.

7.5.4 Accuracy of the DBOT Tracker

We report the accuracy of each estimated pose after obtaining the initial *mesh_pose* and after the DBOT tracker has been set up. Using both the particle filter and Gaussian filter methods, the DBOT object pose is measured at one and three seconds after startup while the object is stationary. As shown in Table 7.1, all DBOT trackers outperform the *mesh_pose*, which indicates that the tracker was able to function successfully after receiving an initial pose from *mesh_pose*.

In reality, the particle tracker corrected the pose faster and more accurately than the Gaussian tracker. It is likely that the Gaussian tracker is less resistant to inaccurate initialization. The authors of DBOT [120] stated that the particle tracker is slightly more robust, while the Gaussian tracker is more accurate. The Gaussian tracker is tolerant to distortions in the input point cloud and occluded settings in which the particle tracker is incapable of tracking.

As shown in Table 7.2, we compare the tracking accuracy of our objects to comparable objects in the YCB dataset [126]. The mean AUC of the

	mask	mesh		particle 1s		Gaussian 1s		particle 3s		Gaussian 3s	
	t	t	R	t	R	t	R	t	R	t	R
cube	89.1	90.3	96.0	97.4	97.0	93.9	97.0	**97.7**	**97.2**	90.7	96.9
sphere	76.1	79.2	**97.0**	97.9	**97.0**	97.5	**97.0**	97.9	**97.0**	**98.0**	**97.0**
lego_toy	66.7	80.2	90.2	**85.0**	92.2	84.8	92.3	84.9	**92.4**	83.0	92.3
teapot	77.8	80.0	90.7	91.7	93.3	86.5	92.8	**94.0**	**95.2**	91.2	94.8
cup	90.9	90.2	95.0	94.0	95.9	85.1	95.6	**94.3**	**96.2**	77.1	95.7
jug	69.8	77.3	90.2	93.6	92.4	74.8	92.8	**95.1**	95.3	89.9	**96.2**
bowl	69.3	70.4	84.4	81.0	87.7	**86.6**	84.9	81.1	**88.7**	80.8	85.6
MEAN	77.1	81.1	91.9	91.5	93.7	87.0	93.2	**92.1**	**94.6**	87.2	94.1

	mesh		particle 1s		Gaussian 1s		particle 3s		Gaussian 3s	
	AUC	<2cm	AUC	<2cm	AUC	<2cm	AUC	<2cm	AUC	<2cm
cube	93.8	100.0	96.8	100.0	95.6	99.7	**97.0**	100.0	94.3	98.8
sphere	88.6	100.0	**96.1**	100.0	96.0	100.0	**96.1**	100.0	**96.1**	100.0
lego_toy	90.8	99.3	93.3	98.1	**93.4**	96.7	**93.4**	97.3	92.3	88.4
teapot	88.2	100.0	93.7	100.0	91.3	94.0	**95.2**	100.0	93.8	95.2
cup	92.9	100.0	95.1	99.0	92.0	92.7	**95.3**	98.9	87.8	86.4
jug	84.6	98.8	91.8	99.6	86.6	82.7	**95.2**	99.6	93.3	94.5
bowl	80.4	78.7	83.6	80.2	**85.5**	78.1	82.9	80.2	83.9	75.0
MEAN	88.5	**96.7**	92.9	**96.7**	91.5	92.0	**93.6**	96.6	91.6	91.2

Table 7.1: Accuracy of the object tracking pipeline

	PoseCNN [121]		DenseFusion [122]		own	Own results mesh		particle 3s	
	AUC	<2cm	AUC	<2cm	class	AUC	<2cm	AUC	<2cm
pitcher base	97.8	100.0	97.1	100.0	jug	84.6	98.8	95.2	99.6
bowl	81.0	54.9	88.2	98.8	bowl	80.4	78.7	82.9	80.2
mug	95.0	99.8	97.1	100.0	cup	92.9	100.0	95.3	98.9
wood block	87.6	80.2	89.7	94.6	cube	93.8	100.0	97.0	100.0
MEAN	90.35	83.73	93.03	98.35		87.93	94.38	92.6	94.68

Table 7.2: Tracking accuracy on similar objects in comparison to PoseCNN and DenseFusion

particle tracker at 3 seconds after initialization is 92.6%, which is slightly lower than DenseFusion [122] but higher than PoseCNN [121]. A direct comparison with these previous works is not entirely fair because of the different datasets used for the evaluation and the use of simulated images. Nevertheless, it shows the feasibility of the proposed object-tracking pipeline and its potential application in robot manipulation.

7.5.5 Accuracy of the Grasp Intent Prediction

We obtained the reach-and-grasp motion trajectories of two individuals (one male and one female). A total of 350 trajectories were collected in four different environment settings, with each trajectory lasting approximately two to five seconds. Users started with their right hand over the Leap Motion controller and reached forward to mimic grasping an object in a particular direction (right or top) while looking at the display interface. The start and end of the trajectories were set by pressing a key on the keyboard.

Figure 7.9 illustrates the average accuracy of prediction for 18 grasping

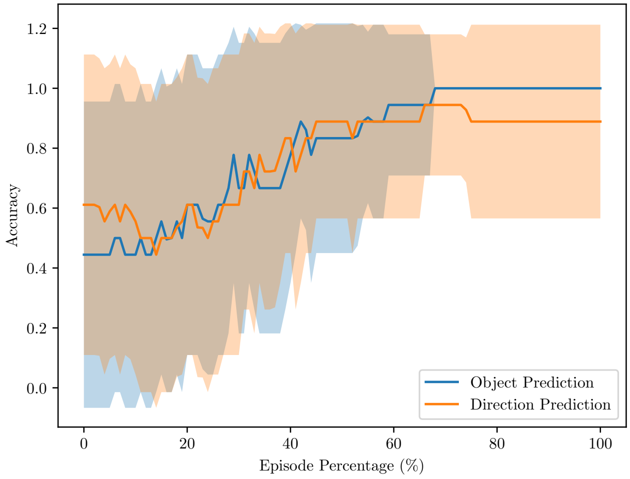


Figure 7.9: Prediction accuracy of the grasp intention prediction averaged throughout the episode

episodes in a single environment using trajectories that were excluded during training. The target object prediction and the grab direction prediction had an accuracy of 79.4% and 77.4%, respectively. The accuracy of the target prediction reached perfect accuracy before reaching 70% of the episode, and the accuracy of the direction prediction reached an average of 89% at the end of the episode. The low accuracy during the first twenty percent of the episode was caused by the delay between the beginning of the recording and the beginning of the movement. Further optimization of the hyperparameters and the neural network structure can be performed to improve prediction outcomes. The use of recurrent neural networks may also improve the accuracy of early predictions.

7.5.6 Experiment Setup for a Teleoperation Task

To evaluate the efficacy of the system, we designed a manipulation-based teleoperation task. We anticipate that the quicker the target object is identified, the sooner the robot can begin motion planning, resulting in faster task completion in a traded control scenario. We conducted a simulated user study by modeling different human user behaviors:

- *Normal* user consists of human user trajectory data
- *Noisy* user is simulated by adding Gaussian noise at each time step to the normal user
- *Biased* user is simulated by adding a constant random offset to the normal user trajectory.

The *Biased* user simulates the human user with poor perception, such as perceiving the object to be closer than it is. The difference between a *Biased* user and a *Noisy* user is that the noise in the *Biased* user stays constant throughout the trajectory, but the noise in the *Noisy* user varies with each time step.

The task is to conduct a series of picking motions to retrieve three objects off a table. The user selects the target object and demonstrates the picking motion. By combining the framework from Section 7.4, we presume that all object poses are known. The robot has to determine the order in which the objects are collected. An object is identified as the goal object when the robot predicts the same target for t consecutive time steps ($t = 80$) and the first 300 time steps are discarded to lower the prediction error at the beginning of the episode.

7.5.7 Evaluation of Different Simulated Users

To demonstrate that early prediction increases teleoperation efficiency, we refer to *Early* mode as the control mode in which, during user demonstration, the robot begins to plan its grasping trajectory toward the predicted target object. This mode is compared to *Late* mode, where the motion planning does not begin until the user completes the user demonstration.

The teleoperation system was evaluated based on the following criteria: time required to predict the goal object (a.k.a. time until execution), episode duration, and the accuracy of prediction (goal prediction, direction prediction) when the robot identified a goal. The time until execution is the sum of the time required for the robot to initiate the grasp motion for the three objects and the episode duration represents the total time for picking all three objects, including the time it took for motion planning. Table 7.3 displays the average results for 12 episodes.

Despite a reduction in prediction accuracy, early motion planning and execution based on goal prediction resulted in reduced episode duration, approximately five seconds faster for all user modes. The *Noisy* and *Biased* users required more time for the robot to confidently identify a goal. Also, except for the direction prediction for the *Biased* user in *Early* mode, the accuracy of predictions did not decrease throughout user modes. This indicates that the prediction model was robust enough

Criteria	Control Mode	User Mode Normal	Noisy	Biased	MEAN
Time until Execution (s)	**Early**	9.6 ± 1.1	10.3 ± 0.9	10.1 ± 1.3	10.0
	Late	14.6 ± 1.4	14.7 ± 1.2	14.1 ± 1.5	14.5
Episode Duration (s)	**Early**	33.1 ± 5.6	32.5 ± 3.7	33.9 ± 5.3	33.2
	Late	38.6 ± 5.1	39.2 ± 3.1	38.1 ± 5.4	38.6
Goal Prediction (%)	**Early**	0.86 ± 0.49	0.89 ± 0.47	0.89 ± 0.62	0.88
	Late	0.97 ± 0.28	0.97 ± 0.28	0.89 ± 0.62	0.94
Direction Prediction (%)	**Early**	0.94 ± 0.37	0.97 ± 0.28	0.89 ± 0.62	0.93
	Late	1.00 ± 0.00	0.97 ± 0.28	0.97 ± 0.28	0.98

Table 7.3: Teleoperation results for different simulated users and control modes

to withstand the noisy conditions. Overall, the results demonstrate that the suggested traded control system can enhance teleoperation performance while utilizing noisy hand motions to control the robot.

7.6 Summary

Traded control can be a useful method to improve teleoperation performance and handle high-latency teleoperation situations. To successfully implement such a paradigm, the following schemes are a prerequisite for the autonomous system. First, an accurate prediction of the human operator's intent to avoid conflicts between the human operator and the operating robot. Second, a comprehensive understanding of its surrounding environment to interpret and associate the operator's intentions with tasks.

In this chapter, I present ideas to satisfy the prerequisites to enable traded control. I presented a teleoperation system that utilizes intuitive human-grabbing hand gestures to perform sequential manipulation tasks. The operator's intention is acquired using a prediction model from hand gestures. This allows us to mitigate the issues that arise when using hand gestures, where the control signals are noisy and inconsistent. The robot autonomously generates a grasping or retrieving motion using trajectory optimization as soon as the robot identifies the user's intention. To obtain an understanding of the robot's environment, the combination of Mask R-CNN [112] and the model-based object tracker DBOT [120] is used to locate objects of interest and initialize the tracker. Results showed the successful mask detection and initial pose estimation through the proposed method and showed the feasibility of its application in robot manipulation. The simulated user study indicated that using intent prediction brought down the overall task execution time.

Chapter 8

Conclusion

The dissertation explored how intelligent systems and human operators can share control to ensure accurate and safe execution, yet allow the human operator to maintain a sense of control during teleoperation. In the chapters, I investigated different methods for arbitrating the commands of human and semi-autonomous agents, specifically for reaching tasks, in which both agents are providing control inputs to the system at each time step.

Throughout the dissertation, the semi-autonomous agent was assumed to be able to autonomously accomplish the task, when the goal is known. I first described in Chapter 3 how to represent a robust semi-autonomous agent that generates optimal policies at any given state. Building on top of this global optimal policy of the semi-autonomous agent, I investigated how we can extract information about the robot's surrounding environment by observing the semi-autonomous agent's policies and use this to modify the human operator's commands.

Chapters 4 and 5 focused on representing a metric that expresses the divergence of the semi-autonomous agent's policy within a close range. The metric emerged from the formulation of an optimization problem to find the policy that favors the human operator while remaining within a trust region to prevent the arbitrated policy from deviating too much from the autonomous policy. This metric is then used to augment the human operator's policy, to prevent the robot from collisions or assist the robot near a goal state. The positive results from Chapter 4 showed the effectiveness and the potential of the proposed method in a low-dimensional setting, where both the human and the semi-autonomous agent operated in the same workspace. Chapter 5 generalized the framework to higher dimensional spaces and settings using Riemannian geometry considering the kinematic differences between the human operator and the robot.

While Chapters 4 and 5 provide methods for combining human and semi-autonomous agents policies, Chapter 6 provides a framework for tuning the level of arbitration interactively. By utilizing the goal-conditioned semi-autonomous agent policies, states that possess uncertainty can be identified. These states are referred to as decision points, and the human operator's commands are prioritized in these states to clear out any ambiguity rather than relying on unstable intent prediction. This prevents fighting against the autonomous system during teleoperation caused by incorrect prediction of the user intent. Here, the reinforcement learning approach is used to learn a model that outputs an arbitrated action given both agents' actions, through interaction with the system. The experiments showed that the method learned to effectively lower its arbitration in the decision points while avoiding collisions and helping to reach the goal. Although the chapter only showed results from simulated users, it is expected that the method can be further refined through interactions with a real human operator.

Chapter 7 showed an example of a complete teleoperation setup including a perception and manipulation pipeline. In this chapter, a traded control framework was implemented to enable reach-and-grasp teleoperation tasks using hand gestures, allowing the human operator to give commands as if performing the task in first-person perspective. From the operator's hand gesture commands, the high-level commands were extracted such as the intent and the grasping order rather than the exact signals of the hand motion. The intention was interpreted by a trained neural network classifier. The classifier identified the grasp direction and the object. For the perception, the combination of Mask-CNN and the depth-based object tracker enabled the automatic initialization of the object tracker to identify the goal space. The experiment results showed successful mask detection and initial pose estimation using the perception pipeline and showed the feasibility of its application in robot manipulation. Altogether, the simulation study showed that the traded control system reduced the total task execution time while using noisy hand gestures of the human operator.

8.1 Future Work

The possible future directions are laid out in this section.

User-specific Adaptability during Arbitration When teleoperating a robot, some operators prefer to retain control over the performance. Others are more accepting of assistance from the autonomy. In addition, their preference toward assistance and their teleoperation skills tend to vary through interaction. Previous works such as Nikoladis et al. [98] have addressed the issue of human-robot mutual adaptation during short teleoperation experiments in a discrete state setting. It would be interesting to extend the ideas proposed in Chapter 6 where an arbitration agent is trained through interaction using reinforcement learning, to experiment with whether different arbitration strategies could be fine-tuned by interactions with users with different preferences.

Arbitration in Variable Autonomy A variable autonomy system refers to a system where the degree of robot assistance can be dynamically adjusted within the *Levels of Autonomy* [127], [128], ranging from complete direct control to full autonomy [129]. It is also referred to as adjustable autonomy [130], [131]. In contrast to shared control where the arbitration is happening at the servo-level, variable autonomy combines both agents at the task or mission level depending on the task and situation. Designing such interfaces that can seamlessly leverage human and robot controls by fluently switching between autonomy levels (i.e., using data-driven approaches) can enhance the effectiveness of the teleoperation interface. Specifically, further work is required to answer the following research questions:

- When does the operator require a shift in the level? Is it task-dependent, interface-dependent, or user-dependent?
- What are the implicit/explicit reactions towards the changed autonomy? How does the operator react to unwanted autonomy?

The Future of Teleoperation The enhancement of teleoperation through balancing between human intelligence and robot autonomy can enable robots to perform a range of manipulation tasks, from simple pick-and-place operations to precision-requiring and heavy-duty tasks across different domains such as disaster relief, outer space, underwater operations, semi-autonomous driving, assistive robots, medical robots, and so much more. Often teleoperation is regarded as an intermediate solution toward developing fully autonomous robots and human intervention is required to control the robot [132]–[134]. Does this mean that one day, when intelligent robots achieve full autonomy, there will be no need to teleoperate robots?

Teleoperation and shared control is not solely an interim solution before full robot autonomy is realized. Even when systems achieve full autonomy, human supervision (teleoperation) will still be required to maintain control of the intelligent systems at any moment. In addition, as robots work with humans in close proximity, safer teleoperation methods may be needed to ensure that the robot does not harm nearby humans. Thus, many exciting research topics remain in developing intelligent shared control methods for teleoperation.

Bibliography

[1] C. G. Atkeson, P. B. Benzun, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, *et al.*, 'What Happened at the DARPA Robotics Challenge Finals', *The DARPA Robotics Challenge Finals: Humanoid Robots to the Rescue*, pp. 667–684, 2018 (cit. on pp. 25, 32).

[2] M. T. Dzindolet, L. G. Pierce, H. P. Beck, L. A. Dawe, 'The Perceived Utility of Human and Automated Aids in a Visual Detection Task', *Human factors*, vol. 44, no. 1, pp. 79–94, 2002 (cit. on p. 26).

[3] S. Lichiardopol, 'A Survey on Teleoperation', 2007 (cit. on p. 31).

[4] J. Cui, S. Tosunoglu, R. Roberts, C. Moore, D. W. Repperger, 'A Review of Teleoperation System Control', in *Proceedings of the Florida Conference on Recent Advances in Robotics*, Citeseer, 2003, pp. 1–12 (cit. on p. 31).

[5] T. Fong, C. Thorpe, 'Vehicle Teleoperation Interfaces', *Autonomous Robots*, vol. 11, pp. 9–18, 2001 (cit. on pp. 31, 32).

[6] W. Conklin, S. Tosunoglu, 'Conceptual Design of a Universal Bilateral Manual Controller', in *Proceedings of the Florida Conference on Recent Advances in Robotics*, 1996, pp. 187–191 (cit. on p. 32).

[7] W. S. Kim, A. K. Bejczy, 'Demonstration of a High-Fidelity Predictive/Preview Display Technique for Telerobotic Servicing in Space', *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 698–702, 1993 (cit. on p. 32).

[8] P. Batsomboon, S. Tosunoglu, 'A Review of Teleoperation and Telesensation System', in *Proceedings of the Florida Conference on Recent Advanced in Robotics*, 1996 (cit. on p. 32).

[9] R. R. Murphy, *Disaster Robotics.* MIT press, 2014 (cit. on p. 32).

[10] H. A. Yanco, A. Norton, W. Ober, D. Shane, A. Skinner, J. Vice, 'Analysis of Human-Robot Interaction at the DARPA Robotics Challenge Trials', *Journal of Field Robotics*, vol. 32, no. 3, pp. 420–444, 2015 (cit. on p. 32).

[11] P. Griffiths, R. B. Gillespie, 'Shared Control Between Human and Machine: Haptic Display of Automation During Manual Control of Vehicle Heading', in *Proceedings of the International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, IEEE, 2004, pp. 358–366 (cit. on p. 33).

[12] F. Flemisch, M. Heesen, J. Kelsch, J. Schindler, C. Preusche, J. Dittrich, 'Shared and Cooperative Movement Control of Intelligent Technical Systems: Sketch of the Design Space of Haptic-Multimodal Coupling Between Operator, Co-Automation, Base System and Environment', *IFAC Proceedings Volumes*, vol. 43, no. 13, pp. 304–309, 2010 (cit. on p. 33).

[13] F. Flemisch, D. Abbink, M. Itoh, M.-P. Pacaux-Lemoine, G. Weßel, 'Shared Control is the Sharp End of Cooperation: Towards a Common Framework of Joint Action, Shared Control and Human Machine

Cooperation', *IFAC-PapersOnLine*, vol. 49, no. 19, pp. 72–77, 2016 (cit. on pp. 33, 71).

[14] S. Musić, S. Hirche, 'Control Sharing in Human-Robot Team Interaction', *Annual Reviews in Control*, vol. 44, pp. 342–354, 2017 (cit. on p. 33).

[15] W. Wang, X. Na, D. Cao, J. Gong, J. Xi, Y. Xing, F.-Y. Wang, 'Decision-Making in Driver-Automation Shared Control: A Review and Perspectives', *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 5, pp. 1289–1307, 2020 (cit. on pp. 33, 54).

[16] A. Franchi, C. Secchi, M. Ryll, H. H. Bulthoff, P. R. Giordano, 'Shared Control: Balancing Autonomy and Human Assistance with a Group of Quadrotor UAVs', *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 57–68, 2012 (cit. on p. 33).

[17] E. J. Rodríguez-Seda, J. J. Troy, C. A. Erignac, P. Murray, D. M. Stipanovic, M. W. Spong, 'Bilateral Teleoperation of Multiple Mobile Agents: Coordinated Motion and Collision Avoidance', *IEEE Transactions on Control Systems Technology*, vol. 18, no. 4, pp. 984–992, 2009 (cit. on p. 33).

[18] J. Tan, C. Xu, L. Li, F.-Y. Wang, D. Cao, L. Li, 'Guidance Control for parallel parking Tasks', *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 1, pp. 301–306, 2019 (cit. on p. 33).

[19] M. Selvaggio, M. Cognetti, S. Nikolaidis, S. Ivaldi, B. Siciliano, 'Autonomy in Physical Human-Robot Interaction: A Brief Survey', *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7989–7996, 2021 (cit. on p. 34).

[20] A. D. Dragan, S. S. Srinivasa, ‘A Policy-Blending Formalism for Shared Control’, *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, 2013 (cit. on pp. 34, 35, 68, 71, 81, 117, 128, 132).

[21] D. Gopinath, S. Jain, B. D. Argall, ‘Human-in-the-Loop Optimization of Shared Autonomy in Assistive Robotics’, *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 247–254, 2016 (cit. on pp. 34, 35, 71).

[22] C. Schultz, S. Gaurav, M. Monfort, L. Zhang, B. D. Ziebart, ‘Goal-Predictive Robotic Teleoperation from Noisy Sensors’, *IEEE International Conference on Robotics and Automation (ICRA)*, 2017 (cit. on pp. 34, 35, 92).

[23] A. A. Allaban, V. Dimitrov, T. Padır, ‘A Blended Human-Robot Shared Control Framework to Handle Drift and Latency’, in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2019 (cit. on p. 35).

[24] D.-J. Kim, R. Hazlett-Knudsen, H. Culver-Godfrey, G. Rucks, T. Cunningham, D. Portee, J. Bricout, Z. Wang, A. Behal, ‘How Autonomy impacts Performance and Satisfaction: Results from a Study with Spinal Cord Injured Subjects Using an Assistive Robot’, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 42, no. 1, pp. 2–14, 2011 (cit. on pp. 35, 68).

[25] S. Javdani, H. Admoni, S. Pellegrinelli, S.S. Srinivasa, J. A. Bagnell, ‘Shared Autonomy via Hindsight Optimization for Teleoperation and Teaming’, *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, 2018 (cit. on pp. 35, 68, 72, 109).

[26] D. Lee, H. Seo, M. W. Jung, ‘Neural Basis of Reinforcement Learning and Decision Making’, *Annual Review of Neuroscience*, vol. 35, pp. 287–308, 2012 (cit. on p. 35).

[27] E. Todorov, ‘Optimality Principles in Sensorimotor Control’, *Nature Neuroscience*, vol. 7, no. 9, pp. 907–915, 2004 (cit. on p. 35).

[28] D. Hadfield-Menell, S. J. Russell, P. Abbeel, A. Dragan, ‘Cooperative Inverse Reinforcement Learning’, *Advances in Neural Information Processing Systems*, vol. 29, 2016 (cit. on p. 36).

[29] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction.* MIT press, 2018 (cit. on p. 36).

[30] E. Todorov *et al.*, ‘Optimal Control Theory’, *Bayesian Brain: Probabilistic Approaches to Neural Coding*, pp. 268–298, 2006 (cit. on pp. 38, 53).

[31] F. Bullo, A. D. Lewis, ‘Geometric Control of Mechanical Systems’, 2005 (cit. on pp. 39, 41, 62, 93).

[32] J. M. Lee, *Introduction to Riemannian Manifolds.* Springer, 2018, vol. 2 (cit. on p. 39).

[33] ——, *Smooth Manifolds.* Springer, 2012 (cit. on p. 39).

[34] P. D. Neilson, M. D. Neilson, R. T. Bye, ‘A Riemannian Geometry Theory of Human Movement: The Geodesic Synergy Hypothesis’, *Human Movement Science*, vol. 44, pp. 42–72, 2015 (cit. on p. 41).

[35] H. Klein, N. Jaquier, A. Meixner, T. Asfour, ‘A Riemannian Take on Human Motion Analysis and Retargeting’, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5210–5217, 2022 (cit. on p. 41).

[36] S. Arimoto, M. Yoshida, M. Sekimoto, K. Tahara, 'A Riemannian-Geometry Approach for Control of Robotic Systems under Constraints', *SICE Journal of Control, Measurement, and System Integration*, vol. 2, no. 2, pp. 107–116, 2009 (cit. on p. 41).

[37] N. Ratliff, M. Toussaint, S. Schaal, 'Understanding the Geometry of Workspace Obstacles in Motion Optimization', *IEEE International Conference on Robotics and Automation (ICRA)*, 2015 (cit. on pp. 42, 93, 98).

[38] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, 'Differential Kinematics and Statics', *Robotics: Modelling, Planning and Control*, pp. 105–160, 2009 (cit. on p. 43).

[39] M. Toussaint, 'Lecture Notes: Some Notes on Gradient Descent', 2012. [Online]. Available: `https://www.user.tu-berlin.de/mtoussai/notes/gradientDescent.pdf` (cit. on pp. 43, 44).

[40] S.-I. Amari, S. C. Douglas, 'Why Natural Gradient?', in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, vol. 2, 1998, pp. 1213–1216 (cit. on pp. 43, 72).

[41] S.-I. Amari, 'Natural Gradient Works Efficiently in Learning', *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998 (cit. on pp. 44, 72).

[42] A. Ly, M. Marsman, J. Verhagen, R. P. Grasman, E.-J. Wagenmakers, 'A Tutorial on Fisher Information', *Journal of Mathematical Psychology*, vol. 80, pp. 40–55, 2017 (cit. on p. 45).

[43] A. Kristiadi. (Mar. 2018). Natural Gradient Descent, [Online]. Available: https://wiseodd.github.io/techblog/2018/03/14/natural-gradient/ (cit. on p. 45).

[44] J. Martens, 'New Insights and Perspectives on the Natural Gradient Method', *arXiv preprint arXiv:1412.1193*, 2014 (cit. on pp. 46, 77).

[45] Y. Oh, S.-W. Wu, M. Toussaint, J. Mainprice, 'Natural Gradient Shared Control', *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 1223–1229, 2020 (cit. on pp. 48, 67, 94, 95).

[46] Y. Oh, M. Toussaint, J. Mainprice, 'Learning to Arbitrate Human and Robot Control Using Disagreement Between Sub-Policies', *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5305–5311, 2021 (cit. on pp. 48, 118).

[47] Y. Oh, J.-C. Passy, J. Mainprice, 'Augmenting Human Policies Using Riemannian Metrics for Human-Robot Shared Control', *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2023 (cit. on pp. 48, 89).

[48] M. Toussaint, 'Introduction to Robotics', 2016. [Online]. Available: https://www.user.tu-berlin.de/mtoussai/teaching/Lecture-Robotics.pdf (cit. on p. 52).

[49] R. Tedrake, *Underactuated Robotics, Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. 2023. [Online]. Available: https://underactuated.csail.mit.edu (cit. on p. 53).

[50] Y. Zhou, C. Barnes, J. Lu, J. Yang, H. Li, 'On the Continuity of Rotation Representations in Neural Networks', in *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5745–5753 (cit. on pp. 56, 61).

[51] J. Mainprice, N. Ratliff, S. Schaal, 'Warping the Workspace Geometry with Electric Potentials for Motion Optimization of Manipulation Tasks', *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016 (cit. on p. 56).

[52] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, S. Srinivasa, 'Planning-Based Prediction for Pedestrians', *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3931–3936, 2009 (cit. on pp. 57, 74).

[53] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, M. Hebert, 'Activity Forecasting', in *European Conference on Computer Vision*, Springer, 2012, pp. 201–214 (cit. on p. 57).

[54] C. G. Atkeson, A. W. Moore, S. Schaal, 'Locally Weighted Learning', in *Lazy learning*, Springer, 1997, pp. 11–73 (cit. on p. 59).

[55] G. Wang, F. Manhardt, F. Tombari, X. Ji, 'GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 611–16 621 (cit. on p. 61).

[56] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, P. Abbeel, 'Overcoming Exploration in Reinforcement Learning with Demonstrations', *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299, 2018 (cit. on p. 66).

[57] L. Marzari, A. Pore, D. Dall'Alba, G. Aragon-Camarasa, A. Farinelli, P. Fiorini, 'Towards Hierarchical Task Decomposition Using Deep

Reinforcement Learning for Pick and Place Subtasks', in *2021 20th International Conference on Advanced Robotics (ICAR)*, IEEE, 2021, pp. 640–645 (cit. on p. 66).

[58] S. Nasiriany, H. Liu, Y. Zhu, 'Augmenting Reinforcement Learning with Behavior Primitives for Diverse Manipulation Tasks', *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7477–7484, 2022 (cit. on p. 66).

[59] A. Goil, M. Derry, B. D. Argall, 'Using Machine Learning to Blend Human and Robot Controls for Assisted Wheelchair Navigation', in *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*, IEEE, 2013, pp. 1–6 (cit. on pp. 68, 71).

[60] S. J. Anderson, J. M. Walker, K. Iagnemma, 'Experimental Performance Analysis of a Homotopy-Based Shared Autonomy Framework', *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 2, pp. 190–199, 2014 (cit. on pp. 68, 71).

[61] M. Gao, J. Oberländer, T. Schamm, J. M. Zöllner, 'Contextual Task-aware Shared Autonomy for Assistive Mobile Robot Teleoperation', *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3311–3318, 2014 (cit. on pp. 68, 71).

[62] A. Broad, T. Murphey, B. Argall, 'Operation and Imitation under Safety-Aware Shared Control', in *Workshop on the Algorithmic Foundations of Robotics*, 2018 (cit. on pp. 68, 71).

[63] A. Broad, T. Murphey, B. Argall, 'Highly Parallelized Data-Driven MPC for Minimal Intervention Shared Control', in *Robotics: Science and Systems*, 2019 (cit. on pp. 68, 72).

[64] J. Kofman, X. Wu, T. J. Luu, S. Verma, 'Teleoperation of a Robot Manipulator Using a Vision-Based Human-Robot Interface', *IEEE Transactions on Industrial Electronics*, vol. 52, no. 5, pp. 1206–1219, 2005 (cit. on pp. 71, 145).

[65] C. Smith, M. Bratt, H. I. Christensen, 'Teleoperation for a Ball-Catching Task with Significant Dynamics', *Neural Networks*, vol. 21, no. 4, pp. 604–620, 2008 (cit. on p. 71).

[66] C. Phillips-Grafflin, H. B. Suay, J. Mainprice, N. Alunni, D. Lofaro, D. Berenson, S. Chernova, R. W. Lindeman, P. Oh, 'From Autonomy to Cooperative Traded Control of Humanoid Manipulation Tasks with Unreliable Communication', *Journal of Intelligent & Robotic Systems*, vol. 82, no. 3-4, pp. 341–361, 2016 (cit. on pp. 71, 145).

[67] K. Muelling, A. Venkatraman, J.-S. Valois, J. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, J. A. Bagnell, 'Autonomy Infused Teleoperation with Application to BCI Manipulation', *arXiv preprint arXiv:1503.05451*, 2015 (cit. on p. 71).

[68] S. Jain, B. Argall, 'An Approach for Online User Customization of Shared Autonomy for Intelligent Assistive Devices', in *Proceedings of the IEEE International Conference on Robotics and Automomation, Stockholm, Sweden*, 2016 (cit. on p. 71).

[69] Y. Oh, M. Toussaint, J. Mainprice, 'Learning Arbitration for Shared Autonomy by Hindsight Data Aggregation', in *Robotics: Science and Systems, Workshop on AI and Its Alternatives for Shared Autonomy in Assistive and Collaborative Robotics*, 2019 (cit. on p. 71).

[70] S. Reddy, A. D. Dragan, S. Levine, ‘Shared Autonomy via Deep Reinforcement Learning’, *arXiv preprint arXiv:1802.01744*, 2018 (cit. on pp. 72, 102, 121).

[71] S. M. Kakade, ‘A Natural Policy Gradient’, in *Advances in Neural Information Processing Systems*, 2002, pp. 1531–1538 (cit. on p. 73).

[72] J. Peters, S. Schaal, ‘Natural Actor-Critic’, *Neurocomputing*, vol. 71, no. 7-9, pp. 1180–1190, 2008 (cit. on p. 73).

[73] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, ‘Trust Region Policy Optimization’, in *International Conference on Machine Learning*, 2015, pp. 1889–1897 (cit. on p. 73).

[74] T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. Mccrory, *et al.*, ‘Summary of Team IHMC’s Virtual Robotics Challenge Entry’, *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 307–314, 2013 (cit. on p. 90).

[75] M. DeDonato, V. Dimitrov, R. Du, R. Giovacchini, K. Knoedler, X. Long, F. Polido, M. A. Gennert, T. Padır, S. Feng, *et al.*, ‘Human-in-the-Loop Control of a Humanoid Robot for Disaster Response: A Report from the DARPA Robotics Challenge Trials’, *Journal of Field Robotics*, vol. 32, no. 2, pp. 275–292, 2015 (cit. on p. 90).

[76] C. Yang, X. Wang, L. Cheng, H. Ma, ‘Neural-Learning-Based Telerobot Control with Guaranteed Performance’, *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3148–3159, 2016 (cit. on pp. 90, 92).

[77] H. Wang, X. P. Liu, ‘Adaptive Shared Control for a Novel Mobile Assistive Robot’, *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 6, pp. 1725–1736, 2014 (cit. on p. 92).

[78] M. Laghi, M. Maimeri, M. Marchand, C. Leparoux, M. Catalano, A. Ajoudani, A. Bicchi, ‘Shared-Autonomy Control for Intuitive Bimanual Tele-Manipulation’, *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2018 (cit. on p. 92).

[79] T.-C. Lin, A. U. Krishnan, Z. Li, ‘Shared Autonomous Interface for Reducing Physical Effort in Robot Teleoperation via Human Motion Mapping’, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9157–9163, 2020 (cit. on p. 92).

[80] X. Gao, J. Silvério, E. Pignat, S. Calinon, M. Li, X. Xiao, ‘Motion Mappings for Continuous Bilateral Teleoperation’, *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5048–5055, 2021 (cit. on p. 92).

[81] X. Wang, C. Yang, H. Ma, L. Cheng, ‘Shared Control for Teleoperation Enhanced by Autonomous Obstacle Avoidance of Robot Manipulator’, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4575–4580, 2015 (cit. on p. 92).

[82] B. Xi, S. Wang, X. Ye, Y. Cai, T. Lu, R. Wang, ‘A Robotic Shared Control Teleoperation Method Based on Learning from Demonstrations’, *International Journal of Advanced Robotic Systems*, vol. 16, no. 4, p. 1 729 881 419 857 428, 2019 (cit. on p. 92).

[83] K. H. Khokar, R. Alqasemi, S. Sarkar, R. V. Dubey, ‘Human Motion Intention Based Scaled Teleoperation for Orientation Assistance in Preshaping for Grasping’, in *2013 IEEE 13th International Conference*

on Rehabilitation Robotics (ICORR), IEEE, 2013, pp. 1–6 (cit. on p. 92).

[84] J. J. Abbott, P. Marayong, A. M. Okamura, 'Haptic Virtual Fixtures for Robot-Assisted Manipulation', in *Robotics Research: Results of the 12th International Symposium ISRR*, Springer, 2007, pp. 49–64 (cit. on p. 92).

[85] M. J. Zeestraten, I. Havoutis, S. Calinon, 'Programming by Demonstration for Shared Control with an Application in Teleoperation', *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1848–1855, 2018 (cit. on pp. 92, 93).

[86] M. Ewerton, G. Maeda, D. Koert, Z. Kolev, M. Takahashi, J. Peters, 'Reinforcement Learning of Trajectory Distributions: Applications in Assisted Teleoperation and Motion Planning', *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4294–4300, 2019 (cit. on p. 92).

[87] D. A. Abbink, M. Mulder, E. R. Boer, 'Haptic Shared Control: Smoothly Shifting Control Authority?', *Cognition, Technology & Work*, vol. 14, pp. 19–28, 2012 (cit. on p. 92).

[88] G. Li, F. Caponetto, X. Wu, I. Sarakoglou, N. G. Tsagarakis, 'A Haptic Shared Autonomy with Partial Orientation Regulation for DoF Deficiency in Remote Side', *IEEE Transactions on Haptics*, vol. 16, no. 1, pp. 86–95, 2023 (cit. on p. 92).

[89] S. Calinon, 'Gaussians on Riemannian Manifolds: Applications for Robot Learning and Adaptive Control', *IEEE Robotics & Automation Magazine*, vol. 27, no. 2, pp. 33–45, 2020 (cit. on p. 93).

[90] M. Toussaint, ‘Newton Methods for K-Order Markov Constrained Motion Problems’, *arXiv preprint arXiv:1407.0414*, 2014 (cit. on pp. 93, 98, 101).

[91] J. Mainprice, R. Hayne, D. Berenson, ‘Goal Set Inverse Optimal Control and Iterative Replanning for Predicting Human Reaching Motions in Shared Workspaces’, *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 897–908, 2016 (cit. on pp. 93, 98).

[92] B. Ti, A. Razmjoo, Y. Gao, J. Zhao, S. Calinon, ‘A Geometric Optimal Control Approach for Imitation and Generalization of Manipulation Skills’, *Robotics and Autonomous Systems*, vol. 164, p. 104 413, 2023 (cit. on p. 93).

[93] J. E. Dennis Jr, R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, 1996 (cit. on p. 100).

[94] Franka Emika, *Franka Emika Panda Robot*, `https://www.franka.de/`, Accessed: 2024–02-22 (cit. on p. 101).

[95] Y. Du, S. Tiomkin, E. Kiciman, D. Polani, P. Abbeel, A. Dragan, ‘AvE: Assistance via Empowerment’, *Neural Information Processing Systems (NeurIPS)*, 2020 (cit. on pp. 102, 122).

[96] Facebook Technologies & Lenovo, *Oculus Rift S*, `https://www.oculus.com/rift-s/`, Accessed: 2024–02-22 (cit. on p. 107).

[97] Unity, *Unity Robotics Hub*, `https://github.com/Unity-Technologies/Unity-Robotics-Hub`, Accessed: 2024–02-22, 2019 (cit. on p. 107).

[98] S. Nikolaidis, Y. X. Zhu, D. Hsu, S. Srinivasa, ‘Human-Robot Mutual Adaptation in Shared Autonomy’, *ACM/IEEE International Confer-*

ence on Human-Robot Interaction (HRI), pp. 294–302, 2017 (cit. on pp. 109, 168).

[99] S. Li, M. Bowman, H. Nobarani, X. Zhang, 'Inference of Manipulation Intent in Teleoperation for Robotic Assistance', *Journal of Intelligent & Robotic Systems*, vol. 99, pp. 29–43, 2020 (cit. on p. 117).

[100] L. Wang, Q. Li, J. Lam, Z. Wang, Z. Zhang, 'Intent Inference in Shared-Control Teleoperation System in Consideration of User Behavior', *Complex & Intelligent Systems*, pp. 1–11, (cit. on p. 117).

[101] D. E. Gopinath, B. D. Argall, 'Active Intent Disambiguation for Shared Control Robots', *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 6, pp. 1497–1506, 2020 (cit. on p. 117).

[102] S. Jain, B. Argall, 'Probabilistic Human Intent Recognition for Shared Autonomy in Assistive Robotics', *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 9, no. 1, pp. 1–23, 2019 (cit. on pp. 117, 128, 132).

[103] R. M. Aronson, T. Santini, T. C. Kübler, E. Kasneci, S. Srinivasa, H. Admoni, 'Eye-Hand Behavior in Human-Robot Shared Manipulation', in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 4–13 (cit. on p. 117).

[104] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, 'Continuous Control with Deep Reinforcement Learning', *arXiv preprint arXiv:1509.02971*, 2015 (cit. on pp. 119, 126).

[105] W. Xu, J. Huang, Y. Wang, C. Tao, L. Cheng, 'Reinforcement Learning-Based Shared Control for Walking-Aid Robot and Its Experimental Verification', *Advanced Robotics*, vol. 29, no. 22, pp. 1463–1481, 2015 (cit. on p. 121).

[106] J. Gao, S. Reddy, G. Berseth, A. D. Dragan, S. Levine, 'Xt2: Training an X-to-Text Typing Interface with Online Learning from Implicit Feedback', in *International Conference on Learning Representations (ICLR)*, 2021 (cit. on p. 121).

[107] C. Schaff, M. R. Walter, 'Residual Policy Learning for Shared Autonomy', *arXiv preprint arXiv:2004.05097*, 2020 (cit. on p. 121).

[108] F. C. Fernandez, W. Caarls, 'Deep Reinforcement Learning for Haptic Shared Control in Unknown Tasks', *arXiv preprint arXiv:2101.06227*, 2021 (cit. on p. 122).

[109] K. V. Mardia, P. E. Jupp, K. Mardia, *Directional Statistics.* Wiley Online Library, 2000, vol. 2 (cit. on p. 124).

[110] M. A. Carreira-Perpinan, 'Mode-Finding for Mixtures of Gaussian Distributions', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1318–1323, 2000 (cit. on p. 125).

[111] Y. Oh, T. Schäfer, B. Rüther, M. Toussaint, J. Mainprice, 'A System for Traded Control Teleoperation of Manipulation Tasks Using Intent Prediction from Hand Gestures', *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2021 (cit. on p. 142).

[112] K. He, G. Gkioxari, P. Dollár, R. Girshick, ‘Mask R-CNN’, *IEEE International Conference on Computer Vision (ICCV)*, 2017 (cit. on pp. 144, 151, 164).

[113] J. Bohren, L. L. Whitcomb, ‘A Preliminary Study of an Intent-Recognition-Based Traded Control Architecture for High Latency Telemanipulation’, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017 (cit. on p. 145).

[114] F. Weichert, D. Bachmann, B. Rudak, D. Fisseler, ‘Analysis of the Accuracy and Robustness of the Leap Motion Controller’, *Sensors*, vol. 13, no. 5, 2013 (cit. on p. 145).

[115] J. Guna, G. Jakus, M. Pogačnik, S. Tomažič, J. Sodnik, ‘An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking’, *Sensors*, vol. 14, no. 2, 2014 (cit. on p. 145).

[116] G. Marin, F. Dominio, P. Zanuttigh, ‘Hand Gesture Recognition with Jointly Calibrated Leap Motion and Depth Sensor’, *Multimedia Tools and Applications*, vol. 75, no. 22, 2016 (cit. on p. 145).

[117] W. Zeng, C. Wang, Q. Wang, ‘Hand Gesture Recognition Using Leap Motion via Deterministic Learning’, *Multimedia Tools and Applications*, vol. 77, no. 21, 2018 (cit. on p. 145).

[118] W. Qi, S. E. Ovur, Z. Li, A. Marzullo, R. Song, ‘Multi-Sensor Guided Hand Gestures Recognition for Teleoperated Robot Using Recurrent Neural Network’, *IEEE Robotics and Automation Letters*, 2021 (cit. on p. 145).

[119] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, S. Schaal, 'Probabilistic Object Tracking Using a Range Camera', *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013 (cit. on pp. 146, 149).

[120] J. Issac, M. Wüthrich, C. G. Cifuentes, J. Bohg, S. Trimpe, S. Schaal, 'Depth-Based Object Tracking Using a Robust Gaussian Filter', *IEEE International Conference on Robotics and Automation (ICRA)*, 2016 (cit. on pp. 146, 149, 157, 164).

[121] Y. Xiang, T. Schmidt, V. Narayanan, D. Fox, 'PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes', *arXiv preprint arXiv:1711.00199*, 2017 (cit. on pp. 146, 156, 159).

[122] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, S. Savarese, 'DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion', 2019 (cit. on pp. 146, 156, 159).

[123] Kenta-Tanaka et al., *Probreg*, version 0.1.6. [Online]. Available: `https://probreg.readthedocs.io/en/latest/` (cit. on p. 151).

[124] M. Toussaint, 'Komo: Newton Methods for K-Order Markov Constrained Motion Problems', *arXiv preprint arXiv:1407.0414*, 2014 (cit. on p. 152).

[125] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, N. Navab, 'Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes', in *Asian Conference on Computer Vision*, Springer, 2013, pp. 548–562 (cit. on p. 155).

[126] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, A. M. Dollar, 'The YCB Object and Model Set: Towards Common benchmarks for Manipulation Research', in *International Conference on Advanced Robotics (ICAR)*, IEEE, 2015, pp. 510–517 (cit. on p. 157).

[127] T. B. Sheridan, W. L. Verplank, T. Brooks, 'Human/Computer Control of Undersea Teleoperators', in *NASA. Ames Res. Center The 14th Ann. Conf. on Manual Control*, 1978 (cit. on p. 168).

[128] M. R. Endsley, D. B. Kaber, 'Level of Automation Effects on Performance, Situation Awareness and Workload in a Dynamic Control Task', *Ergonomics*, vol. 42, no. 3, pp. 462–492, 1999 (cit. on p. 168).

[129] M. Chiou, N. Hawes, R. Stolkin, 'Mixed-Initiative Variable Autonomy for Remotely Operated Mobile Robots', *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 10, no. 4, pp. 1–34, 2021 (cit. on p. 168).

[130] P. Scerri, D. V. Pynadath, M. Tambe, 'Towards Adjustable Autonomy for the Real World', *Journal of Artificial Intelligence Research*, vol. 17, pp. 171–228, 2002 (cit. on p. 168).

[131] M. A. Goodrich, D. R. Olsen, J. W. Crandall, T. J. Palmer, 'Experiments in Adjustable Autonomy', in *Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, Seattle, WA, 2001, pp. 1624–1629 (cit. on p. 168).

[132] P. Kaiser, D. Kanoulas, M. Grotz, L. Muratore, A. Rocchi, E. M. Hoffman, N. G. Tsagarakis, T. Asfour, 'An Affordance-Based Pilot Interface for High-Level Control of Humanoid Robots in Supervised Autonomy', *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 621–628, 2016 (cit. on p. 169).

[133] J. Luo, Z. Lin, Y. Li, C. Yang, ‘A Teleoperation Framework for Mobile Robots Based on Shared Control’, *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 377–384, 2019 (cit. on p. 169).

[134] K. T. Ly, M. Poozhiyil, H. Pandya, G. Neumann, A. Kucukyilmaz, ‘Intent-Aware Predictive Haptic Guidance and Its Application to Shared Control Teleoperation’, *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 565–572, 2021 (cit. on p. 169).

All URLs were last accessed on 22.02.2024.