NORTHWESTERN UNIVERSITY

Generalizable Data-driven Models for Personalized Shared Control of Human-Machine Systems

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Alexander Broad

EVANSTON, ILLINOIS

May 2019

"You live and learn. At any rate, you live."

- Douglas Adams, *Mostly Harmless*

ABSTRACT

Generalizable Data-driven Models for

Personalized Shared Control of Human-Machine Systems

Alexander Broad

The theory of how humans and machines control and communicate with each other is at the core of the scientific field known as Human-Robot Interaction (HRI). Researchers in this sub-discipline of robotics are therefore particularly interested in developing methods to *reduce the inherent friction in this communication and control channel*. Just as can be observed in the analogous problem of collaboration between two human partners, solutions in this space require a tight coupling between a human partner and an autonomous partner. A conceptual framework that describes this exact relationship is known as shared control (SC). Shared control defines an abstract link between a set of partners (often a human operator and an autonomous agent) that are both responsible for providing control information to the same robotic device. This paradigm is especially useful as a method of extending the physical capabilities of a human operator, while simultaneously considering important constraints defined by the user and environment.

This dissertation is largely motivated by applications of shared control in the fields of assistive and rehabilitation medicine. Therefore, this thesis develops shared control solutions that

are designed specifically to improve, or restore, a human operator's ability to control complex mechanical devices. Example motivating systems include powered wheelchairs, exoskeletons, and robotic manipulators. In addition to increasing a human operator's capabilities, a particularly desirable attribute of any interactive system in assistive and rehabilitation medicine is the acceptance, and enjoyment, of the human-in-the-loop. For this reason, the SC algorithms described in this dissertation allocate the majority of the control authority to the human partner, while the autonomous partner is mainly responsible for providing control information to improve the stability and safety of the joint human-machine system.

The specific techniques described in this dissertation are motivated by the desire to generalize solutions in shared control to generic pairs of human and machine partners, while simultaneously developing a decision making framework that is responsive to the individual human-in-the-loop. To address this desire, this thesis introduces the notion of data-driven model-based shared control (MbSC). Data-driven MbSC extends the efficacy of standard shared control systems to scenarios in which we do not have any prior knowledge of the system dynamics or the human operator. Instead, data-driven MbSC relies on techniques from (1) machine learning to gain an understanding of the joint human-machine system from observation, and (2) optimal control (OC) to develop a control policy for the autonomous partner. The shared control system then allocates authority to each partner to improve desired outcomes (e.g. task-success, stability, and/or safety). Additionally, this dissertation describes data-driven techniques that further personalize the interaction paradigm to the individual human-in-the-loop. The proposed methodology uses a representation of the autonomy's trust in the human partner's control skill learned from observation data. This data-driven metric is then used to modulate the control authority granted to each partner in real-time. Taken together, the techniques described in this thesis describe a *generalizable* solution to the shared control problem that can be *personalized* to the individual human-in-the-loop to improve the capabilities of the joint system.

Acknowledgements

There are many people that I would like to thank for their help, guidance, and friendship during my PhD. I'll start with my advisor Brenna Argall who was the reason I initially chose to attend Northwestern University, and remains a great source of inspiration to this day. I owe a great deal of gratitude to Brenna's influence and constant drive. Whatever success it can be said I have had during my doctoral studies are certainly a result of her mentorship, and I am extremely thankful for her guidance. I would also be remiss if I did not single out Todd Murphey as an equally influential mentor. Todd was an invaluable resource during my time at Northwestern and I will never forget his excitement for discussing new research directions in great depth. Our conversations would often begin with him claiming to only have five minutes, and end with me interrupting him 45 minutes later because *I* needed to get back to work. I would also like to thank the rest of my thesis committee: Scott Niekum and Julie Shah. I have learned tremendously from each of their individual research programs, and the output of their labs. I truly appreciate the time and effort they put into providing feedback on my dissertation.

In viewing my research as a reflection of my own personal education, there is a long list of people who are also deserving of my gratitude. My first research mentor, Thomas Franke, is the reason I continued to work towards a career in science, albeit in a different field than his own. As my master's advisor at Washington University in St. Louis, Bill Smart, and his student Tom Erez, are responsible for my interest in data-driven methods of autonomous control for robotic systems. During my PhD I also had the pleasure of working on a short research project with Thomas Howard from the University of Rochester, who taught me a lot, quickly. Along the same lines, I spent a very fun summer at Mitsubishi Electric Research Laboratories where I was lucky enough to work with Michael Jones and Teng-Yok Lee. Equally important to these mentors have been the people I shared a lab with over the course of nearly 5 years. My time at Northwestern would not have been the same without Matthew Derry, Siddarth Jain, Deepak Gopinath, Mahdieh Nejati, Ahmetcan Erdogan, Michael Young, Chris Miller, Kyra Rudy, and Temesgen Gebrekristos. The same goes for all the professors and students in the Neuroscience and Robotics Lab–particularly Jarvis Schultz, Ian Abraham, and Aleksandra Kalinowska.

I would additionally like to thank the vast number of people who provided me with support during my doctorate, including all of the administrators in the EECS (now CS) department, and the more than 100 participants who took part in my research studies. Richard McNally, Emily's advisor at Harvard University, was a consistent supporter of our crazy travel schedule, and as such, he greatly improved our ability to enjoy the years we spent apart during grad school. It is a nice coincidence that we have all now spent so much time in Chicago. I would also like to pay special tribute to Todd and Ruth Warren, who's fellowship allowed me great freedom both academically, and personally, during my time at Northwestern. Their generosity significantly changed my experience for the better and I am extremely grateful for their support.

In the same vein, I would like to thank all of my friends for their encouragement, and constant reminders of the benefits of a healthy work/life balance. In particular, I would like to thank my friends in Cambridge (Adam, Alan, Jake, Scott, Colin, Ted), where I spent an inordinate amount of time during my PhD for someone who did not live there.

Finally, and above all, I would like to thank my entire family, who have always been the largest collective source of inspiration, encouragement and support. In particular, I would like to thank my parents, Richard and Valerie, my brother David, and of course Brenda. I would also like to thank all of my Grandparents: Joan and Bob, Irma and Julius; my Aunts and Uncles; and all of my cousins. Each one provides more love than I could ever ask for. My family now of course includes Emily and all of her relatives. In particular, Emily's parents Robert and Lisa, her sisters Allison and Jill, and Andy. Emily has been a constant source of support and the most upbeat and encouraging fellow scientist one could imagine. A hearty thanks to her, everyone I mentioned, and anyone else who fell prey to my poor memory but assuredly deserves my gratitude as well.

Dedication

To my family and friends.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Robotics autonomy offers great promise as a tool by which we can enhance, or restore, the natural abilities of a human partner. That is, in contrast to developing robotic technologies that aim to *replace* human functionality, this thesis describes a class of autonomous agents that are designed to *collaborate with*, and *assist* human beings in their quest to interact with their environment. This assistance-based paradigm not only aims to improve the lives of human partners, but offers an extremely practical foundation for the successful adoption of robotic devices into modern society, as it is mirrors how mechanical systems have been used as long as they have existed. Consider the variety of mechanical devices that have been developed to help mankind explore the ground, air and space. Similarly, assistive walkers and wheelchairs have been developed to restore lost, or limited, functionality to countless people in need. This thesis posits that we can further improve the efficacy of mechanical devices in broadly assisting human partners through the integration of artificial intelligence.

From a conceptual standpoint, the ideas in this thesis can be seen as intellectually descendant from the work of Norbert Weiner and his colleagues [**146**]. In particular, this thesis addresses questions that explore the fundamental relationship that describes how humans and machines control and communicate with each other. However, the scope of our work is significantly more narrow than this topic, which is considered an entire field of research at this point in time. Instead, this thesis proposes directions for progress in a subfield known as *shared control* or *shared autonomy*, in which a human and autonomous partner both provide control information to the same dynamic system.

In theory, shared control describes a general framework for improving the efficacy of generic human-machine systems by incorporating an autonomous partner into the control loop of a mechanical device [**5, 105**]. The autonomous partner is designed to account for deficiencies in the human partner's control due to either the inherent complexity of the system, the required fidelity in the control signal, or the physical limitations of the human partner. In essence, shared control can be used to further improve the efficacy of dynamic machines by offloading challenging aspects of the control problem to an autonomous partner in a shared control paradigm. In doing so, the human operator is freed to focus their mental and physical capacities on important high-level tasks like path planning and interaction with the environment. This general framework has the potential to provide great value in a wide range of application domains. For example, in assistive robotics, shared control can improve a user's ability to operate a powered wheelchair and therefore regain lost autonomy and re-engage with the world [**47**]. The same concept can be used to extend the efficacy of rehabilitative machines, like exoskeletons, that would otherwise require significant supervision from a team of trained therapists [**58, 67**]. Broadening the application domain even further, shared control can extend the effective range of search and rescue robots, the safety of robotic surgery tools, and the ability of a human operator to control complex ground and air vehicles.

The great promise of shared control then, is due to the generality of the framework. That is, shared control defines a partnership relationship that can be used to improve a human's ability to operate complex dynamic machinery and accomplish challenging tasks, while simultaneously improving the safety of the joint human-machine system. Beyond the constraint that the human and autonomous partners must both provide input to the same mechanical system, there are few (if any) requirements of a shared control system [**5**]. Instead, this paradigm spans a wide range of partnership relationships that range from human supervision to autonomous supervision, and everything in between (e.g., blending the human and autonomy's control input [**50**]). The particulars of a given human-machine partnership are therefore often defined based on the individual pair of partners and/or the task they are trying to accomplish.

This thesis describes approaches to shared control that are inspired by devices designed to assist people in their daily lives, and those that are designed to help people overcome physical limitations due to either injury or disease. For example, in assistive and rehabilitation medicine, powered wheelchairs, exoskeletons and robotic manipulators can be used as an aide for a human partner who has severely diminished motor capabilities. This use case suggests a few key desirable attributes that help guide the development of the different shared control frameworks that are discussed in this thesis. For example, a growing consensus in the literature [**53, 66, 71, 91**] indicates that users of assistive devices prefer to retain as much control authority as possible. Therefore, instead of developing methods that cede full authority to the autonomous partner (e.g., Level 4 & 5 in SAE's taxonomy of self-driving cars [**3**]), this thesis focuses on methods that can *dynamically adjust the level of assistance* given to a human partner to account for suboptimal user input that would otherwise destabilize, or endanger, the joint system.

A unique characteristic of the work presented in this thesis is that, due to the infinitely large number of potential human-robot partnerships, the solutions we discuss require no *a priori* knowledge of the mechanical system or the human-in-the-loop. Instead, this thesis describes a series of generalizable shared control paradigms that rely on state-of-the-art techniques in machine learning to develop actionable representations of the system dynamics and human partner directly from data. The main impact of this contribution is that it greatly reduces the costs and complexity associated with designing a novel shared control system for a given human-machine system. As a result of the work discussed in this thesis, *shared control can be used to improve the efficacy of any assistive robotic device through simple data-collection and model learning.*

To realize the above ideas in an assistive shared control paradigm, the following objectives are put forth :

- ***Generalizable:*** The autonomous partner should be able to learn a model of an unknown system's dynamics from observation, and use that information to regulate it's motion and account for deficiencies in the human partner's control.
- ***Personalized:*** The autonomous partner should be able to recognize important facets of the human partner's control skill during operation, and use that information to dynamically allocate control authority in a personalized manner.

Together, these concepts describe a generalizable methodology for designing a personalized shared control paradigm. To address these objectives, I outline the contributions of this thesis below.

1.1. Main Contributions

The main contributions of this thesis can be summarized as follows :

- Novel data-driven methods for user modeling, which can in turn be used to improve a human partner's ability to operate complex systems through personalized, dynamic autonomy allocation.
- The introduction of data-driven model-based shared control (MbSC) to the field. This data-driven method for offline, and online, system identification results in models that

can be used to generate autonomous control policies to achieve generic tasks, or adhere to desirable constraints.

- Intelligent control allocation techniques that rely on data-driven models of the human and machine partners to increase the control authority granted to the human partner while simultaneously improving the skill, stability and safety of the joint system.

1.2. Dissertation Outline

The following chapters relay key works undertaken during my PhD and describe how the results provide guidance for further exploration. This thesis begins with a description of related work and helpful background information in Chapter 2. This chapter also includes brief descriptions of results from related experiments that address the objectives outlined above, but that do not fully align with the main thrust of this thesis.

Chapter 3 describes a *data-driven notion of the trust the autonomous partner has in an individual human operator* that is calculated based on the quality of the interactions between a human and autonomous system. In contrast to prior literature, the described methodology does not rely on task-specific metrics, and instead builds a model of the user's understanding of the system dynamics and control skill without *a priori* knowledge of the system, the task or the human-in-the-loop. That is, the described measure of confidence is built upon a control-theoretic foundation that rewards stable operation of the system to give more trusted users additional control authority. The results of this study suggest that, through data-driven methods, we can dynamically adapt a personalized interaction paradigm between a human and machine to improve a user's control without *a priori* knowledge of the specific user.

Chapter 4 introduces the notion of data-driven, model-based shared control (MbSC) for human-machine systems. This paradigm extends the state-of-the-art in shared control/shared autonomy to scenarios in which we do not have *a priori* knowledge of the system dynamics. Instead, both the dynamics of the system and information about the user's interaction are learned from observation. This chapter describes the results of two human subjects studies. Both studies demonstrate that (1) expert knowledge of a system's dynamics is not a requirement for successful shared control, and (2) data-driven models of human-controlled dynamical systems *generalize across users*. The studies also provide additional evidence that geometric signal filters (e.g., MDA [**142**]) can be used to improve a human operator's ability to safely achieve pre-specified tasks, while systematically reducing the control authority of the autonomous partner. A comparison of the two human-subjects studies finds even stronger effects when the data-driven shared control paradigm is based on nonlinear (instead of linear) modeling and optimal control techniques. The second study also demonstrates the data-efficiency of the chosen modeling technique, and the efficacy of data-driven MbSC in an online learning setting.

Chapter 5 extends the notion of data-driven, model-based shared control to situations in which the user's desired motion and/or goal is not known *a priori*. This chapter introduces an autonomous partner that enforces safety and stability constraints *without knowledge of a desired task*. The approach incorporates *safety constraints* through control barrier functions [**11**] that penalize highly dynamic motion as well as actions that move the system towards potentially dangerous environmental risk factors. When the system is in a safe portion of the state space *and the human partner provides safe control*, the user retains full authority over the system. However, in unsafe portions of the state space, the autonomous partner gains control authority and returns the system to a safe state. The methodology described in this work combines the outer-loop stabilization technique described in prior work [**27, 60, 142**] with additional information about the system and environment. Importantly, the high-level algorithm generalizes to any dynamical system as the system model is learned from observation. Additionally, experimental results suggest that this technique can be viewed as a *training mechanism* for the human operator. That is, by constraining the motion of the system to known safe regions in the environment, the human operator can gain experience controlling the system and learn new skills without endangering the safety of either partner. Finally, additional analysis shows how the same safety-aware shared control paradigm can be used in conjunction with Imitation Learning to generate autonomous policies that recreate the behaviors demonstrated by the human-in-the-loop. This idea is valuable as a method of bootstrapping a task-agnostic shared control paradigm into one that incorporates task-specific information.

Chapter 6 continues to build on the idea of using data-driven, model-based shared control as a method of improving a user's ability to operate dynamic systems without *a priori* knowledge of the system, the user, or the task. The algorithm detailed in this chapter allocates control authority based on the user's input and a *prediction* of the safety of the joint human-machine system in the future, given a set of possible actions. This prediction is computed through techniques that build on ideas in model-based reinforcement learning. That is, the autonomous partner evaluates the safety of the joint system over a receding-horizon by analyzing predicted trajectories that stem from a large sample of potential actions the user may wish to take at each given moment. Importantly, these trajectories can be computed independently of each other and therefore evaluated *in parallel and in real-time*. The algorithm then selects the control sample that minimizes the impact of the autonomous partner, while remaining safe over a receding horizon. Notably, this approach places an additional emphasis on the user's acceptance of the shared control paradigm by *explicitly maximizing the authority allocated to the human partner at every instant to improve their sense of agency*. The results of the described study show that the shared control paradigm improves system safety without knowledge of the user's goal, while maintaining high-levels of user satisfaction and low-levels of frustration. To close the loop on the ideas presented in this thesis, this chapter also describes how data collected during run-time can be used to compute a model of the autonomy's trust in the human operator's control skill to further personalize the influence of the autonomous partner.

Chapter 7 presents an analysis of the results of the main four studies presented in Chapters 3-6 and highlights key takeaways. This chapter also describes limitations of the presented studies and concludes with a description of future directions.

CHAPTER 2

Background and Related Work

The following chapter provides relevant background information and details related work. The background information is useful in understanding the theory and engineering-methods proposed in this thesis. The related work is an overview of prior scientific innovations.

2.1. Shared Control of Human-Machine Systems

The work in this thesis is largely concerned with *shared control* systems. Shared control [**105**] is a concept used to describe how a human and autonomous partner communicate and share control over a mechanical device. Shared autonomy [**71**] is a related term often used to signify the incorporation of classical artificial intelligence technologies (e.g., model learning and reinforcement learning), however, at the core, these two expressions refer to the same concept. In this work, I will use the term shared control. From a birds eye view, shared control (SC) explores the question of how automation can be used to adjust to, and account for, the specific capabilities of a human partner. If done intelligently, and with appropriate knowledge of the individual capabilities of each team member, SC can be used to improve the overall efficiency, stability and safety of the joint system [**89**].

Shared control covers a wide range of interaction paradigms between a human and autonomous partner. One key distinguishing feature is the difference between task-level and signal-level SC [**5**]. In the former, a high level task is broken down into subtasks which are then often handled separately by the human and autonomous partners [**109**]. In the later, both partners are often required to provide continuous input to the same mechanical device. In this case, control authority is dynamically allocated to a particular partner depending on the requirements of the task [**105**] or with respect to the safety of the system [**89**]. Another distinguishing feature is whether the human and mechanical device are physically remote (e.g., controlled through teleoperation) or co-located. The former is common in applications like search and rescue, where the environment is generally considered too dangerous for the human partner [**96**]. The later is common in applications like assistance and rehabilitation [**21, 44**], where the mechanical device is often explicitly used to help the human partner regain lost functionality [**18**]. Finally, the presence of feedback, whether visual or physical [**97**], is another standard distinguishing feature in shared control. This thesis contributes a series of shared control paradigms that are designed for human-oriented applications (i.e., assistance and rehabilitation). In these domains, we assume that both partners are capable of providing continuous control (i.e., signal level SC) and that the human and machine are co-located, suggesting a primary focus on safety.

This thesis describes novel shared control algorithms that are well founded and viable even without *a priori* knowledge of the system or control dynamics. This contribution is important as it extends the efficacy of the described SC paradigm to a wide range of systems that are inherently challenging to model using hand-engineered techniques. That is, the vast majority of related work relies, either explicitly or implicitly, on known models of a given system's dynamics and the user's control influence to define a shared control paradigm. In this thesis, we instead design SC algorithms that generalize to any pair of human and machine partners through *data-driven solutions* that make use of modern techniques from machine learning. Additionally, the described data-driven approach greatly reduces the engineering burden associated with designing and implementing a shared control algorithm for a new pair of human and machine partners. In short, this thesis describes a generalizable methodology that can be used to define shared control paradigms for unknown human-machine systems from simple data collection.

Given the flexibility of the general shared control definition, one could imagine many aspects of the problem that could be solved through data-driven methods. For example, if the autonomous partner aims to improve the safety of the human-machine system, a real-time understanding of the environment is paramount. Data-driven methods have proven particularly useful in analyzing perceptual information like scene understanding [**30**] and obstacle detection [**137**], which are important pieces of information when developing safe autonomous control policies. Similarly, if the autonomous partner is designed to improve a user's ability to interact with the environment (e.g., manipulation of table-top objects [**28**]), or navigate through challenging portions of the environment (e.g., narrow doorways [**47, 145**]), the same perceptual data can be analyzed to form models of human intent and improve task-recognition.

This thesis focuses on developing data-driven models to describe (1) the autonomy's understanding of the human partner [**32, 33**], (2) the dynamics of the mechanical system [**31, 32**], and (3) the autonomous partner's control policy [**27, 31, 32**]. These three ideas come together to form the basis of *data-driven model-based shared control*, a class of algorithms that extend the efficacy of robotic assistance to any given human-machine partnership.

2.2. Data-driven Model-based Shared Control

One of the main contributions of this thesis is to introduce the concept of data-driven model-based shared control [**27, 31**] (MbSC) to the field. This approach extends standard shared control paradigms to scenarios in which we have no prior information about the system dynamics and user interaction schemes. The high-level algorithm relies on techniques that stem from model-based reinforcement learning and optimal control [**24, 131**]. That is, the key idea in MbSC is to learn an explicit model of the system dynamics from observation data. This model can be used for a variety of purposes such as system identification, intent prediction, and autonomous control. In data-driven MbSC, we collect the necessary data from observations of the human and machine interacting so as to capture important information about the human and machine partners. This model is then integrated into an autonomous policy generation method, and used to regulate how control authority is shared between the two partners.

Model-based shared control is therefore closely related to model-based reinforcement learning (MbRL), a paradigm that explicitly learns a model of the system dynamics in addition to learning an effective control policy. MbSC extends MbRL to scenarios in which it is preferential to integrate control from multiple sources (e.g., a human and autonomous partner) instead of relying on a fully autonomous solution. Early work in model-based reinforcement learning includes Barto et al. [**22**] and Kaelbling et al. [**76**]. More recently, researchers have considered integrating learned system models with optimal control algorithms to produce control trajectories in a more data-efficient manner [**104**]. These algorithms compute control through an online optimization process, instead of through further exploration [**22**]. There are, of course, many viable model learning techniques that can be used to describe the system and control dynamics. For example, Neural Networks [**149**], Gaussian Processes [**108**], and Gaussian Mixture Models [**79**] have all shown great promise in this area. Often the best choice of modeling algorithm is related specifically to the application domain. For example, Gaussian Processes perform well in low-data regimes, but scale poorly with the size of the dataset where Neural Networks fit naturally. A survey of learning for control can be found in Schaal et al. [**120**].

The following subsections provide background information on the main three features of MbSC : data-driven modeling (Section 2.2.1), autonomous policy generation (Section 2.2.2), and dynamic control allocation (Section 2.2.3).

2.2.1. Data-driven Modeling and System Identification

Data-driven model-based shared control relies on a model of the system and control dynamics that is learned from observation. As highlighted previously in this thesis, the vast majority of related work in shared control relies instead on *a priori* knowledge of the dynamics of the assistive robotic partner [**5, 18, 105**]. A novel aspect of this work then, is the development of techniques that extend shared control paradigms to scenarios in which we have no prior knowledge of this information [**107**]. Instead, data is collected through observation of interactions between the human partner and the dynamic machine, and then used to learn a model of the system and control dynamics. As the training data stems from interactions between the two partners (i.e., demonstrations), the model incorporates implicit control information about the human-in-the-loop.

This idea is another key insight that motivates this thesis. That is, instead of developing user-specific control allocation strategies, we posit that it is easier to both model and regulate the joint system than it would be to model and regulate each system independently. We therefore use the learned representation of the joint system to develop autonomous intervention policies that can improve system safety and stability constraints *without* explicit knowledge of the individual human partner. In theory, the shared control algorithms proposed in the following chapters are agnostic to the choice of machine learning algorithm used to learn and represent the dynamics [**26**]. However, the implementations described in this work often rely on a particular technique—the Koopman operator. The Koopman operator has only recently become popular in the robotics literature, and for that reason, we provide additional background information on this modeling technique in Section 2.3.

2.2.2. Autonomous Policy Generation

In addition to providing a quantitative understanding of each partner, data-driven models of the joint human-machine system can be used to generate autonomous control policies by integrating the learned representation of the system dynamics into an optimal control framework. This idea stems from model-based reinforcement learning and can be used to generate the autonomous partner's control and regulate the motion of the shared control robot. The main question we must address when considering what the autonomous partner's policy must be capable of is, "What portion of the control problem is the autonomous partner responsible for?"

Depending on the specific pair of human and machine partners, and the given task, one could imagine that the autonomous partner has a wide range of responsibilities. For example, if the human partner is operating a system they have significant experience with (e.g., driving a car), it is possible the autonomous partner only needs to be responsible for particularly challenging aspects of the control problem (e.g., anti-lock braking). In contrast, if the mechanical system is inherently complex (e.g., exoskeletons), or simply unfamiliar to the human operator, the autonomous partner may need to be responsible for the majority of the control problem to ensure system safety. In our motivating domains (i.e., assistive and rehabilitation robotics), the influence of the autonomous partner must also account for the physical capabilities of the human-in-the-loop which may increase over time with rehabilitation, or may decrease over time with the progression of a disease. For these reasons, the autonomous partner must be capable of providing assistance over a wide spectrum of influence, ranging from limited intervention to full control if required. Additionally, due to our choice to explore signal-level shared control paradigms, the autonomous partner must be capable of providing control input with the same frequency as the human partner. This thesis therefore places a particular importance on the use of policy generation methods that are capable of generalizing to a variety of tasks and constraints, and that are efficient enough to run in real-time.

The main contribution of this thesis in the area of autonomous policy generation is a series of methods that integrate learned system models with techniques from classical optimal control (e.g., derivative-based optimization) and stochastic optimal control (e.g., random shooting methods [**106**]). These methods (a) are data-efficient, (b) can be used in an online learning setting, and (c) are easy to integrate into optimal control algorithms. Details of these approaches are described in the methods sections of the following chapters. There are, of course, alternative techniques that can be used to generate the autonomous partner's control policy using data driven methods. For example, model-free reinforcement learning [**120**] describes a methodology for learning policies that directly map states to actions. These algorithms generally require significantly more data to converge than model-based approaches [**20, 116**] and can therefore be challenging to incorporate into shared control systems, particularly in an online learning setting (Chapter 4). Recent literature explores methods that aim to improve the data-efficiency of model-free learning algorithms through the integration of model-based reasoning [**6**]. This line of research suggests model-free policy generation techniques may offer an alternative, reliable solution for shared control paradigms in the future.

2.2.3. Dynamic Control Allocation

Given a model of the system dynamics and the policy of the autonomous partner, the final step in defining a model-based shared control paradigm is describing the strategy used to allocate control between the human and autonomous partners. How to best allocate control authority to maximize the effectiveness of the human-robot team is an open research question [**5, 105, 111**].

Related work in the shared control literature focuses on the development and analysis of techniques that statically and/or dynamically allocate control between human and robot partners. The main objective in these works is to improve system performance while reducing the control burden on the human operator [**105**]. In some applications, the autonomous partner is allocated the majority of the low-level control while the human operator acts in a supervisory role [**8**] while in others, the roles are reversed, and the autonomous partner takes on the supervisory role [**127**]. There is also related work in which neither partner acts as a supervisor and instead control is shared via a mixed-initiative paradigm. For example, researchers have developed techniques in which the autonomous partner is explicitly aware of the human operator's intention [**43**] as well as techniques in which the autonomous partner has an implicit understanding of the human operator's control policy [**31**]. Within the mixed-initiative literature, control allocation techniques range from the use of pre-defined, discretely adjustable functions [**84, 92**] to smooth interpolation [**46, 100**] to probabilistic reasoning [**72**] to formal policy blending methods [**50**]. In addition to sharing control in the signal space, dynamic allocation has been researched through haptic control [**110**] and compliant control [**81**].

The main contribution of this thesis in the area of dynamic control allocation is a series of data-driven methods that can be used to produce *personalized* control allocation strategies that respond to each individual user. This contribution is particularly important in our motivating domains (i.e., assistive and rehabilitation robotics) where an improved sense of independence and agency is paramount to the successful adoption of a specific shared control paradigm. One idea explored in this work is the use of an outer-loop filter [**60, 142**] designed to improve a user's ability to achieve pre-specified tasks without *a priori* knowledge of the system dynamics (Chapter 4). This thesis also extends the data-driven, outer-loop filter paradigm to scenarios in which the user's goal is unknown and safety of the joint system is paramount (Chapter 5). Finally, this thesis proposes a control allocation technique based on an optimization procedure that explicitly minimizes the influence of the autonomous partner over a receding horizon, thereby allocating the majority of the control authority to the human-in-the-loop (Chapter 6). Importantly, all methods described in the following chapters are computationally efficient enough to be run iteratively, inside a control loop. These methods are therefore capable of automatically allocating control in a personalized manner based solely on the state of the system and the human's input at each instant of the interaction. Finally, this thesis also proposes the use of the autonomous partner's trust in an individual human operator's control skill as a method of personalizing how control is allocated to improve system performance and user acceptance.

2.3. The Koopman Operator

As mentioned in Section 2.2.1, model-based shared control relies on a model of the joint system that is learned from observation data. From a scientific perspective, the choice of system modeling technique is relatively inconsequential. My work briefly explores some popular options (e.g., Neural Networks [**26, 29**]), but the vast majority of the work described in the following chapters model the system dynamics through an approximation to the Koopman operator. The Koopman operator has similarities to a variety of methods in the greater machine learning literature (e.g. linear kernel methods [**69**]), however, there are representations of the Koopman operator that make it particularly amenable to engineering systems (see Chapters 4 and 6), in addition to it's powerful theoretical properties [**82**]. As the Koopman operator has only recently been adopted into data-driven engineering applications, this section provides a short introduction to the theory.

The Koopman operator was first described by Bernard Koopman in 1931 as a method of representing any nonlinear dynamical system using a linear operator [**82**]. This is possible because, instead of acting on the original state space, the Koopman acts on a space of observables, in which the dynamics are linear but nominally infinite dimensional. To define the Koopman operator, let us consider a discrete time dynamic system $(\mathcal{X}, t, F)$:

$$x_{t+1} = F(x_t) \tag{2.1}$$

where $\mathcal{X} \subseteq \mathbb{R}^N$ is the state space, $t \in \mathbb{R}$ is time and $F : \mathcal{X} \to \mathcal{X}$ is the state evolution operator. We also define ϕ, a nominally infinite dimensional observation function

$$y_t = \phi(x_t) \tag{2.2}$$

where $\phi : \mathcal{X} \to \mathbb{C}$ defines the transformation from the original state space into the Hilbert space representation that the Koopman operator acts on. The Koopman operator $\mathcal{K}$ is defined as the composition of ϕ with F, such that

$$\mathcal{K}\phi = \phi \circ F. \tag{2.3}$$

By acting on this Hilbert state representation, the *linear* Koopman operator is able to capture the complex, nonlinear dynamics described by the state evolution operator.

While the Koopman operator is theoretically infinite dimensional, recent work has demonstrated the ability to approximate a finite dimensional representation using data-driven techniques [**37, 119**]. These techniques fall under a class of algorithms known as Dynamic Mode Decomposition (DMD) [**119, 121, 141**]. These algorithms use snapshots of observation data to approximate the Koopman modes that describe the dynamics of the observed quantities. In the limit of collected observation data, the approximation to the Koopman becomes exact [**150**]. Instead of relying on the infinite dimensional representation of the state described in the theory, DMD uses a basis as a projection operator to lift the state into a higher-dimensional representation. This thesis makes particular use of Extended Dynamic Mode Decomposition (EDMD) to approximate the Koopman operator. To provide a mathematical treatment of the EDMD algorithm, we start by defining the observation function ϕ as a vector valued set of basis functions chosen to compute a finite approximation to the Hilbert space representation. We can then define the following approximation to the Koopman operator

$$\phi(x_{t+1}) = \mathcal{K}\phi(x_t) + r(x_t) \tag{2.4}$$

where $r(x_t)$ is a residual term that represents the error in the model. The Koopman operator is therefore the solution to the optimization problem that minimizes this residual error term

$$\begin{aligned} J &= \frac{1}{2}\sum_{t=1}^{T} |r(x_t)|^2 \\ &= \frac{1}{2}\sum_{t=1}^{T} |\phi(x_{t+1}) - \phi(x_t)K)|^2 \end{aligned} \tag{2.5}$$

where T is the time horizon of the optimization procedure, and $|\cdot|$ is the absolute value. The solution to the least squares problem presented in Equation (2.5) is

$$K = G^{\dagger}A$$

where † denotes the Moore-Penrose pseudo inverse and

$$G = \frac{1}{T}\sum_{t=1}^{T} \phi(x_t)^T\phi(x_t)$$

$$A = \frac{1}{T}\sum_{t=1}^{T} \phi(x_t)^T\phi(x_{t+1})$$

These data-driven methods have renewed an interest in using the Koopman operator in applied engineering fields. In contemporary work, the Koopman operator has been successfully used to learn the dynamics of numerous challenging systems. This includes demonstrations that show the Koopman operator can differentiate between cyclic and non-cyclic stochastic signals in stock market data [**70**], it can detect specific signals in neural data that signify non-rapid eye movement (NREM) sleep [**36**], and it can be used to segment control modes in hybrid system dynamics [**77**]. Model-based control of robotic systems using a Koopman operator was first described in [**7**] and introduced in a shared control paradigm in our work [**27, 31**]. Deeper treatments of data-driven approaches to approximating the Koopman operator can be found in [**37, 83, 119, 140, 150**].

This thesis makes use of the Koopman operator representation because, in addition to its powerful theoretical properties, the Koopman is particularly amenable to modern computational resources. For example, the Koopman operator can be approximated with techniques that are efficient enough to be learned online [**27**]. Additionally, and perhaps most importantly for real-time shared control, the Koopman operator's simple, linear representation makes it easy to integrate with tools from optimal control [**7, 27, 31, 32**] to generate autonomous policies.

CHAPTER 3

Trust Adaptation Leads to Lower Control Effort in Shared Control of Crane Automation

The following chapter describes a data-driven approach to personalized shared control of a human-machine system. The described SC algorithm relies on a model of the autonomous partner's trust, or confidence, in an individual user's control skill, and increases the human partner's control authority inline with the learned trust metric. The trust level is dynamic and learned through repeated interaction with a given human partner. Trust increases as the human operator demonstrates a clear understanding of the control problem, and skill in operating the robot. A significant contribution of this chapter is that the trust metric is task-agnostic (i.e. not based on features like task-success), and is instead based on a control-theoretic foundation that rewards stable operation of the dynamic system. The results of a human-subjects study demonstrate that a personalized and adaptive trust in the human operator can be used to improve the user's ability to efficiently operate a dynamic system.

3.1. Introduction

The increasing pervasiveness, capabilities and complexity of autonomous robots in human environments has highlighted the need for more sophisticated control sharing techniques that allow humans to interact with, control and shape the behaviors of these systems, while also maintaining a high level of safety. Shared control enables a human and autonomous system to simultaneously control a system. As such, control sharing can create a system that leverages the strengths of each source of control while reducing the effects of the weaknesses. For instance, in the automotive domain shared control systems have contributed to safer driving by autonomously identifying and correcting common control mistakes made by humans [**139**]. Shared control research has accordingly seen a recent surge in interest and utility in areas ranging from rehabilitation and assistive robotics [**44, 92**], to search and rescue [**42, 64**], to transportation [**46, 65**].

Of particular importance to consider when developing these control sharing techniques is that systems exhibiting significant dynamics are difficult for humans to control. Human operators often cope with complex dynamics by sufficiently constraining the system to minimize the effect of the dynamics, for example a crane operator moving a payload very slowly. Alternatively, some form of automated assistance can help to mask the dynamics from the human, for example the unique shape of the JAS-39 airplane [**129**] creates aerodynamics that are uncontrollable by a human alone, thus significant automated assistance from the flight computer is added to stabilize the aircraft.

Our take on control sharing is to combine the relative advantages of the human and robot partners. In this chapter, we consider that automation and optimal control techniques are good at controlling highly dynamic systems, but require a reference trajectory to try to stabilize to. While these reference trajectories could come from automated path planners, engaging a human partner has the advantage of using the exceptional perceptual capabilities and situational awareness of humans to operate in dynamic environments. The key is for the human to provide reference trajectories that the automated controller can track.

In this chapter, one of the primary goals is to maximize stability in complex dynamic systems while taking advantage of humans' perceptual and cognitive adaptability. Towards that end, we develop a human-in-the-loop control framework that reasons explicitly about the amount of control authority that should be allocated to the human based on the trust, or confidence, that the autonomous system has *in the operator's control skill*. The purpose of this trust metric is to allow the system to learn how capable a user is in providing reference trajectories that can be easily tracked by the automated controller. The adaptive trust metric can then be used to develop a more stable shared control system. Our approach is novel in that we characterize interactions between the human and autonomous system within the framework of control theory in order to build this formalized notion of trust, with which the autonomous system can modulate control authority.

The proposed shared control framework is validated on a simulated planar crane robot platform in which a human operator is tasked with maneuvering a payload through a maze to different target locations. This platform provides a simulation that approximates a real-world, cyber-physical system problem that exhibits significant system dynamics. In analyzing our approach, we place particular importance on how well the system is able to learn and modulate control authority to a human based on their ability to provide suitable reference trajectories, as this has a direct effect on the stability of the system.

The remainder of this chapter is structured as follows: Section 3.2 reviews related literature in shared control and the foundation of optimal control theory that this proposed work is built upon. Section 3.3 presents our proposed trust formulation, and Section 3.4 describes the implementation of our experimental platform and details of the experiment, followed by results and discussion in Section 3.5 and conclusions in Section 3.6.

3.2. Background and Related Work

This section provides a brief review of the role that trust has played in shared control frameworks. Additionally, the foundation for the receding horizon optimal controller used in this chapter is presented.

3.2.1. Trust in Shared Control

Within the shared control literature, a subset of research discusses how to model *trust*, or *confidence*, in an effort to improve human-robot team performance [**48, 112, 153**]. In these works, the formulation of confidence represents the trust that the human has *in the autonomous system*, so that the automated system can use the model to choose actions to maximize trust [**152**]. Additionally, trust has been studied in the context of robot-robot teams [**112**]. Our proposed system is differentiated in that we are instead proposing a formulation of trust that represents the confidence that the autonomous system has *in the operator*. This metric is then used to allocate control authority in an effort to maximize stability of the full system. The explicit definition of trust that we use in this thesis is:

> The autonomy's trust in the human operator's control skill is a direct measure of the how closely the autonomous policy can track the human input. This measure represents the user's understanding of the system dynamics and their skill in providing inputs that are achievable by the robotic device.

This definition covers cases where the autonomy tries to track a reference trajectory (Chapter 3) or a simple control input (Chapter 6). Constraints can be applied to the autonomy's ability to track the system to ensure real-time operation, or additional features like stability and safety.

3.2.2. Receding-Horizon Optimal Control

In this chapter, the user's primary mode of interaction with the dynamic system is by defining reference trajectories for the automated controller to attempt to follow. The automated controller uses a discrete-time, receding-horizon, nonlinear optimization procedure to calculate controls to stabilize to the user-defined reference trajectory. The receding-horizon controller (RHC) is a type of Model Predictive Control [**101**] in which cost is minimized over a short time horizon. Our chosen methodology is based on an online projection-based trajectory optimization technique originally presented in [**68**]. This technique was later made interactive by adapting the RHC to discrete-time systems [**122**], and eventually reformulating it as a real-time, receding horizon procedure [**124**] that was used to experimentally stabilize the physical version of the planar crane system used in this work. Other methods are similarly capable of solving the presented optimization problem (e.g. [**49, 56, 133**]), however, the described approach was chosen because it provides both an optimal control and a feedback law, increasing the stability of the resulting receding horizon controller. Additionally, the trajectory optimization technique used is general such that it can be applied to different robots with relatively little effort.

In a general receding horizon control algorithm, at every timestep k a discrete-time trajectory optimization problem is solved over a reference trajectory defined over the next N timesteps. The optimization routine only has $\Delta t = t_{k+1} - t_k$ seconds to achieve convergence and produce a control signal for the present timestep, before a new measurement is taken and the procedure must begin again for the next timestep. One of the primary features of the particular algorithm utilized herein is that it produces a dynamically feasible system trajectory after every iteration. So even if Δt seconds does not provide sufficient computation time to achieve full convergence, as long as there is time to take a single step, the algorithm is still capable of producing system controls.

In this work, at timestep k the reference trajectory for the receding optimization is given by

$$\begin{aligned}\bar{\xi}_{ref,k} &= (x_{ref,k}, u_{ref,k}) \\ &= \left(\{x_{ref}(i)\}_{i=k}^{k+N},\, \{u_{ref}(i)\}_{i=k}^{k+N-1}\right)\end{aligned}$$

over the horizon $t_{ref,k} = \{t_{ref}(i) = i\Delta t \mid i = k..k+N\}$ where the reference state $x_{ref,k}$ and reference input $u_{ref,k}$ do not necessarily satisfy the system dynamics. The optimization problem statement at time k is then

$$\begin{aligned}\xi_k^* &= \arg\min_{\xi_k \in \mathscr{T}} J(\xi_k, k) \\ J(\xi_k) &= \sum_{i=k}^{k+N-1} l(x(i), u(i), i) + m(x(k+N))\end{aligned}$$

where ξ is used to indicate N-length sequences of both state x and input u, and $\mathscr{T}$ is the set of dynamically admissible states and input trajectories over the $t_{ref,k}$ time horizon. The running cost Lagrangian $l(\cdot)$ and the terminal cost $m(\cdot)$ are given by

$$\begin{aligned}l\left(x(k), u(k), k\right) =& \frac{1}{2}(x(k) - x_{ref}(k))^\intercal Q(x(k) - x_{ref}(k)) + \\ & \frac{1}{2}(u(k) - u_{ref}(k))^\intercal R(u(k) - u_{ref}(k))\end{aligned}$$

and

$$m(x(N)) = \frac{1}{2}(x(N) - x_{ref}(N))^\intercal P_1(x(N) - x_{ref}(N))$$

where Q, R, and P_1 are positive semidefinite, symmetric weighting matrices [**12**].

This optimization algorithm is an iterative, indirect optimal control algorithm with simultaneous variations of state and control at each iteration. As with many iterative optimization algorithms, it is guaranteed to converge to a local minimum, but has no guarantees on finding a global optimizer. Both first and second order decent directions are found by minimizing a local quadratic model. Once a descent direction is found, a step size is found to satisfy a sufficient decrease condition, and then the scaled descent direction is added to the current iterate. When the descent direction is added to the current iterate, the state-control trajectory pair no longer satisfies the system dynamics. The projection operator maps this infeasible system trajectory back to the system's trajectory manifold while maintaining convergence guarantees.

Important to note about this framework is that for a given timestep size, Δt, we only allow the optimization to run for up to Δt seconds. Since this computation takes a finite amount of time, the horizon for the optimizations are actually one timestep ahead of real-world time. This way the optimization will have completed by the time its result is needed for sending controls to the system. As a final point, note that the reference trajectory must be defined for at least $N\Delta t$ seconds into the future. The consequence is that the system always operates with $N\Delta t$ seconds of time delay from the user-provided reference.

3.3. Formulation of the Autonomy's Trust

In order to define, adapt and make use of a formal notion of trust, our proposed framework consists of two steps, *evaluation of user input* and *control modulation*. In the *evaluation of user input*, a trust metric is calculated as a function of the deviation from the reference trajectory. The *control modulation* step then uses the metric to allocate control authority. This approach allows us to asses and improve the system's understanding of the user's abilities.

3.3.1. Evaluation of User Input

After each interaction with the system, a trust metric is calculated using tools provided by optimal control theory. We take a control-theoretic viewpoint in which we use the *deviation* of the executed trajectory from the reference trajectory, as this measure indicates how well the receding horizon controller is able to track the user input. For a given trial i we calculate a deviation metric δ. The deviation from the reference trajectory can be captured by the Fréchet distance [**10**] between the executed and desired trajectories. To compute this metric in real time, we use a discrete variation of the Fréchet distance [**52**],

$$\delta_i(f,g) = \inf_{\alpha,\beta} \max_{t\in[0,1]} d(f(\alpha(t)), g(\beta(t))) \tag{3.1}$$

where $f : [a,b] \to V$ and $a,b \in \Re$, is the reference trajectory and $g : [a',b'] \to V$ and $a',b' \in \Re$, is the control trajectory and (V,d) is a metric space. α and β are continuous nondecreasing functions that map from $[0,1]$ onto $[a,b]$ and $[a',b']$, respectively. $\inf$ is the infinum, or greatest lower bound. We use the Fréchet distance to compute the deviation between the executed and desired trajectories as this measure accounts for the velocity and ordering of points along each curve, a property not shared by similar metrics such as the Hausdorff distance [**117**], which computes the distance between two geometries without explicitly considering the paths as time-based trajectories. Additionally, as mentioned, the discrete measure can be computed in real time which is important both for our experiments and for any resulting interactive system.

It should be noted that while we focus on the deviation from the reference trajectory in this work, there are other control-theoretic measures of performance that could be interesting.

Measures such as distance from the basin of attraction of the receding horizon controller could be incorporated into the calculation of δ.

We define the trust metric to decrease as the deviation δ increases. To calculate the trust τ_i at trial i, we represent the distribution of deviations as a Gaussian distribution, $\delta \sim N(\mu, \sigma^2)$, where μ and σ^2 are the mean and variance of an individual's deviation history. We then update the previous trust metric τ_{i-1} by computing the probability $\mathscr{P}$ of δ_i

$$\tau_i = \tau_{i-1} + \gamma \cdot \mathscr{P}\{\delta = \delta_i\} \tag{3.2}$$

where γ is a learning rate that determines how quickly the trust decays with performance, and $0.1 \leq \tau \leq 1$.

This Gaussian distribution represents the system's knowledge of the user, and is iteratively updated after each trial. Intuitively, if there is a large deviation from the desired trajectory, then τ_i should be low—because the user is providing reference trajectories that are difficult for the controller to track, which reduces the overall robustness of the system. We use a Gaussian distribution to represent the system's knowledge of the user as it provides a probabilistic method for weighting updates to the trust metric that places a larger emphasis on greater deviations and reduces the effect of smaller changes in deviation history. Additionally, as the distribution is parameterized by an individual user's history, we can recognize significant changes in the performance as either personal learning or a failure to understand the system.

3.3.2. Modulation of User Input via Trust

The proposed framework uses trust to allocate control authority. This approach is motivated by the fact that the human might be poor at accounting for the system dynamics in the low-level controls, but by using the learned trust level we can *regulate* the human's input to produce stable reference trajectories that require little effort to track.

Modulation of the user's input is realized through a combination of a low-pass filter and scaling the input speed. By removing the high-frequency content from the input signal, the receding horizon controller is better able to track the reference trajectory. Similarly, by scaling the input speed, users are better able to control for momentary mistakes that can lead to challenging reference trajectories. It is important to note that these transformations can adversely affect more typical task performance measures such as time to completion, but this is a trade-off for system stability.

Trust modulates the cutoff frequency of the low-pass filter ω according to

$$\omega = (\omega_{max} - \omega_{min}) \cdot (\tau_i)^{\lambda} + \omega_{min} \tag{3.3}$$

where ω_{min} is the minimum cutoff frequency that still allows the user to complete the task, ω_{max} is the maximum frequency that contains control information in the input signal, λ is a parameter that shapes the steepness and direction of the function, and τ_i is the user trust from Equation (3.2).

Trust also influences the magnitude of the human's input, according to

$$\widetilde{v_i} = \tau_i \cdot v_i \tag{3.4}$$

where $\widetilde{v_i}$ is the tempered 2D system velocity and v_i is the 2D input velocity.

As trust in the operator input increases, the user is given greater command bandwidth—with the expectation that skilled users generate high frequency inputs only when appropriate

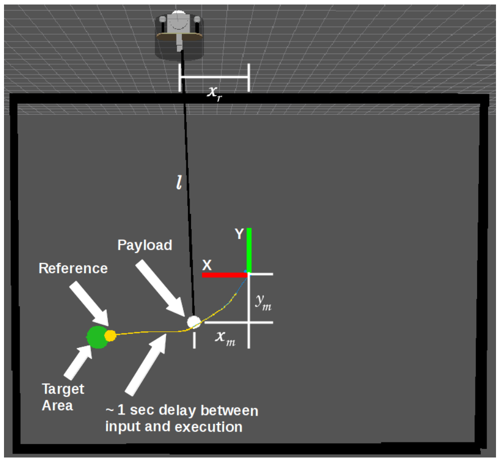


Figure 3.1. Definition of the system configuration variables x_m, y_m, x_r and l. The delay between user input (yellow line) and execution (blue line) is approximately 1 second.

and feasible for the controller to track, whereas with novice users the high frequency signals tend to be overshoot or corrective movements.

3.4. Experiment Design

Our trust-based shared autonomy framework is demonstrated on a simulated planar crane system, in which an overhead robot with a winch has a mass suspended by a string. The dynamic simulation and optimal control calculations for the planar crane system are completely done in the open-source Python module `trep`[1] [**73**], which has been used for a variety of real-time optimal control and estimation problems ranging in complexity from a single degree-of-freedom (DOF) pendulum up to a 40-DOF marionette [**74, 123**]. One of `trep`'s strengths is its integration with the Robot Operating System (ROS). `Trep`'s ability for real-time optimal control calculations, combined with ROS's ability to interface with hardware and share code, result in a software package that enables shared control research.

[1]`trep` is available at `http://nxr.northwestern.edu/trep`

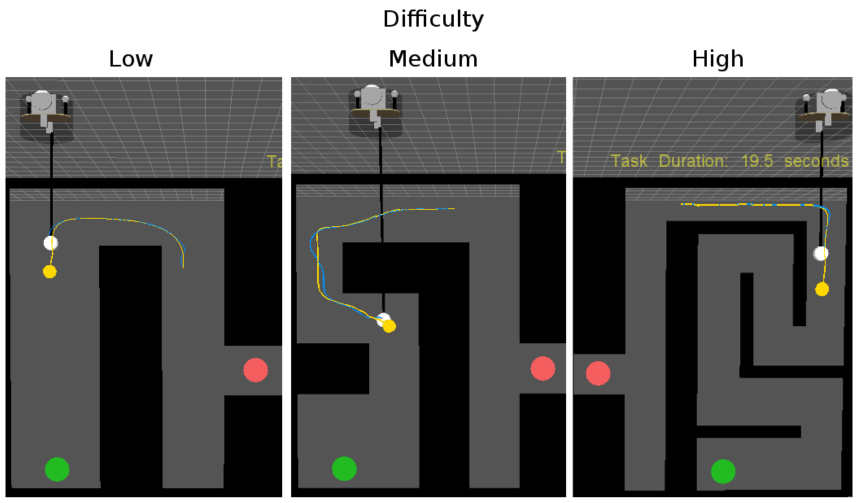


Figure 3.2. Example maze environments for the simulated planar crane task, showing initial position (red circle), target position (green circle), current position of the user input (yellow circle), reference trajectory (yellow line), executed trajectory (blue line), suspended mass (white circle), and maze walls (black). A robotic winch (grey) manipulates the location and length of the string supporting the payload.

The experiment was run on a Core i7 laptop with 8 GBs of RAM. The winch was initialized at the same location in each experiment (red circle in Fig. 3.2). The operator used the joystick of a Sony Playstation 3 (PS3) controller to provide the desired trajectory (or reference trajectory), which refers to the ordered set of target positions (yellow line in Fig. 3.2). As the user moves the target position (green circle in Fig. 3.2) through the environment, we maintain the desired position and velocity associated with each timestep.

3.4.1. Experimental System

The overhead robot is constrained to motion in a single dimension, moving only along the x-axis, and a pulley controls the length of the string. The resulting system has four configuration variables: horizontal position of the mass x_m, vertical position of the mass y_m, horizontal position of the robot x_r, and the length of the string l. The vertical position of the robot is fixed. Fig. 3.1 illustrates the coordinate system and configuration variables. Our model of the system assumes that the horizontal position of the robot x_r and the length of the string l can be treated as *kinematic inputs* [**38, 74**]. With this assumption the Lagrangian for the system is only a function of the dynamic configuration variables x_m and y_m, and it is given by

$$L(q, \dot{q}) = \frac{1}{2} m(\dot{x}_m^2 + \dot{y}_m^2) - m\, g\, y_r \tag{3.5}$$

where m is the mass of the payload and g is the acceleration due to gravity. A holonomic constraint enforces compatibility between the robot's kinematically-controlled horizontal position and string length and the two dynamic configuration variables. In continuous time, this results in a system with an eight-dimensional state vector defined below

$$X(t) = [x_m(t), y_m(t), x_r(t), l(t), \dot{x}_m(t), \dot{y}_m(t), \dot{x}_r(t), \dot{l}(t)]^T$$

and a two-dimensional input vector

$$U(k) = [\ddot{x}(t), \ddot{l}(t)]^T$$

comprised of accelerations of the robot position and string length. To discretize this system and obtain the discrete-time controller, we use variational integrators to represent the discrete-time system [**73, 74**].

We choose this system because it provides dynamics that are difficult for a human to control, while allowing for the definition of dynamic tasks that are representative of tasks for which control sharing would be beneficial. Fig. 3.3 illustrates what can happen when a human controls the robot position and winch speeds directly, without aid from the autonomy. In this case, the position of the mass along the x-axis oscillates considerably due to the pendulum-like dynamics of the system. Though the human's control inputs are uncomplicated (red line), the resulting dynamics are unaccounted for, as evidenced by the oscillation in the positional error of the suspended mass position over time (blue line).

3.4.2. Experimental Task

Inspired by a task that is currently part of real crane operator certification tests, we implement a maze navigation task within our simulated planar crane environment (Fig. 3.2). In the certification test task, the operator must negotiate a zigzag corridor with a payload [**2**]. Our task presents users with a target location within a maze, with walls arranged such that the path to the target is highly constrained. The task is complete once the mass dwells within the target location for 0.5 seconds.

We test three different task configurations of increasing difficulty (Fig. 3.2). First, a *low* difficulty configuration where the total path length is short, requires few turns (∼3) and the

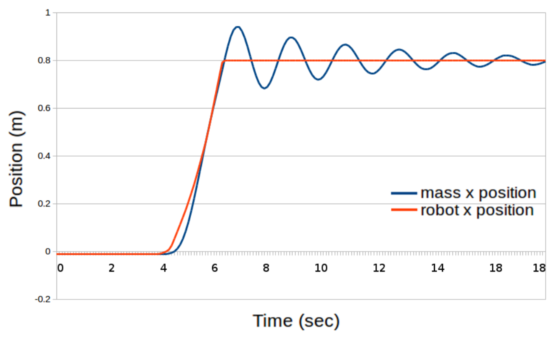


Figure 3.3. Robot (red) and suspended mass (blue) position versus mass position along the x-axis during direct control. Note the pendular dynamics of the system when the user does not attempt to issue controls that account for the system dynamics.

maze hallways are wide. Then, a *medium* difficulty configuration where the total path length is longer, requires more turns ($\sim$5) and the maze hallways are an average of 60% as wide as the low difficulty configuration. Finally, a *high* difficulty configuration where the total path length is long, includes many turns (~ 10) and the maze hallways are an average of 50% as wide as the low difficulty configuration. A total of 21 (8 low difficulty, 8 medium difficulty and 5 high difficulty) unique mazes are used in this experiment.

3.4.3. Autonomous Control

The automated control uses a receding-horizon controller which controls x_r and l to track a reference trajectory. The human specifies the desired reference trajectory with respect to the position of the mass x_m and y_m. A full reference trajectory actually consists of defining the complete position, velocity and momentum of the system at each time step, so, our simplifying assumptions set the velocity and momentum for both the robot and the suspended mass to zero. The variables x_r and l are calculated using a simple inverse kinematic solution that does not account for mass swing, where $x_r = x_m$ and $l = y_r - y_m$. While this is not a feasible trajectory itself, the receding horizon controller does a good job of tracking the user input variables x_m and y_m.

Unlike trajectory optimization techniques which usually require the entire trajectory prior to generating the optimal trajectory, the receding horizon controller optimizes the trajectory within the window of the receding horizon. This enables online control of the system by a user. This interactivity comes at the cost of not having a global optimal trajectory, due to an inability to look ahead along the trajectory over the entire time period. Consequently, if the receding horizon window is set too small, the controller is unable to account for enough of the system dynamics and does not perform well. Alternatively, if the window is set too large, the optimization can take too long to run in real time. We found that for this system the range of window sizes that work well is between 5 and 20 time steps (approximately 0.5 to 2 seconds). For all of our experiments, we use a window size of 10, meaning there is a delay of about one second between user input and system response (Fig. 3.1).

We tune the optimization parameters (Q, R and P_1) such that much higher weights are placed on having the mass follow the reference configuration. That is, the diagonal terms corresponding to x_m and y_m had much higher weights than any other entries. N was chosen to be high enough as to achieve good performance, and low enough as to allow the optimization to reliably fully converge in δt seconds.[2] These weights also are the same as the tuned weights for the real-world experimental system [**124**].

3.4.4. Trust Computation

In order to determine the appropriate range of low-pass filter cutoff frequencies to map to trust, we performed a Fourier analysis of user input that contained high and low frequency movements. Fig. 3.4 shows the analysis which highlights that nearly all of the control information exists in the 0.1 to 2.5 Hz bandwidth. Thus the range of our cutoff frequency for the low-pass, 4th-order Butterworth filter is from 0.1 to 3.0 Hz, where 0.1 Hz maps to zero confidence in the user input and 3.0 Hz maps to full confidence.

From the Fourier analysis, we see that the control content decreases monotonically. To distribute the capabilities of the user uniformly over the confidence range, we use Equation (3.3) to map confidence to cutoff frequency ω with $\omega_{max} = 3.0$, $\omega_{min} = 0.1$, and $\lambda = 3$. The intuition

[2]Optimization parameter values : Q = diag([$20, 20, 0.1, 0.1, 0.1, 0.1, 0.25, 0.25$]), R = diag([$0.1, 0.1$]), P1 = Q, δt = 0.1 (10 Hz), N = 10 (with 10 timesteps, each RHC window considers a 1-second horizon).

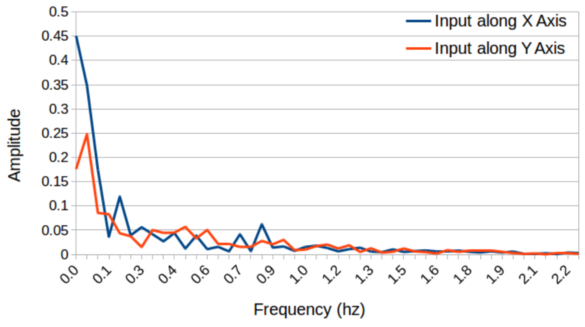


Figure 3.4. Fourier analysis of the user input along the x axis (blue) and along the y axis (red). The majority of the signal content of the user input happens below 0.2 Hz. Above 2.5 Hz there is almost no signal content (not visualized).

behind this is to require large changes in confidence for small changes in cutoff frequency when between ω_{min} and ω_{max}.

3.4.5. Experimental Protocol

Twenty-two users were recruited from the Northwestern University community.[3] Users were randomly grouped into two cohorts:

- Static: The trust level was held static after the initial training period.
- Adaptive: The trust level evolved throughout the experimental procedure.

Each experiment assumed the following protocol:

- Initialization: Three trials of shared control with fixed trust values on the low-difficulty maze task. Each trial was initialized to one of the (low 0.1, middle 0.5 or high 1.0) fixed trust values, and after each trial the trust was updated according to Equation 3.2.

[3]Two participants were removed from the study before analysis due to a poor understanding of the task requirements. On average, users completed 11.7 of the 15 test trials. The two removed users completed one trial each.

The presentation order (of which fixed trust value) was random and balanced across subjects.

- Low: Five trials of shared control on the low-difficulty maze task. Trust was initialized in the first trial to the average of the three updated trust values from the initialization phase, for both cohorts. Trust was then held fixed for cohort *static*, and updated according to Equation 3.2 after each trial for cohort *adaptive*.
- Medium: Five trials of shared control on the medium-difficulty maze task. Trust for the *adaptive* cohort was initialized to the final value updated in phase *Low*. Trust for the *static* cohort remained fixed.
- High: Five trials of shared control on the high-difficulty maze task. Trust for the *adaptive* cohort was initialized to the final value updated in phase *Medium*. Trust for the *static* cohort remained fixed.

Each experiment additionally included five trials of direct control on the low-difficulty maze, for a total of 23 trials.

3.5. Results and Discussion

The validation of the proposed formulation consists of an analysis of the system's *understanding of*, and faculty in *accounting for*, user specific abilities to provide references trajectories that can be easily tracked by the automated controller. One premise underlying this formulation of trust is that the system should be able to learn from an individual user's inputs and leverage this information to modulate the control authority afforded to that user. In our formulation, the goal of the system is to modulate the user input to produce stable references trajectories as defined by a minimization of control effort. We compute the controller effort as the magnitude of the two dimensional control vector, $u(t)$. This vector is comprised of the finite-differenced velocity of the robot's horizontal position and the finite-differenced velocity of the string length. We compute the magnitude at each time step using the Euclidean norm. The average controller effort over the course of an entire trial is defined as

$$U = \frac{\sum_{t=0}^{N} \|u(t)\|}{N} \tag{3.6}$$

where t is time, and N is the final time-step in a given trajectory. Therefore, larger controller effort indicates that the controller is experiencing some combination of increased error from the reference trajectory and increased input effort. Either scenario indicates that the automated controller has had to work harder to track the reference trajectory.

In this chapter, we evaluate the performance of the operator on a given task entirely through an analysis of the average required controller magnitude. There are other possible measures including task-specific metrics like number of collisions; however, we chose average controller magnitude as our sole performance metric as it is task-agnostic and will generalize to other applications in which a user provides a reference trajectory.

3.5.1. Adaptive vs. Static Trust

Here we analyze the system's ability to modulate a user's trust metric to produce stable reference trajectories. We perform a statistical analysis comparing the average controller magnitude, U, between the adaptive and static trust cohorts in each maze configuration. All statistical analysis is done using a two-tailed Student's t-test. In both the static ($p < 0.01$) and adaptive ($p < 0.01$) trust cohorts, we see a statistically significant decrease in the average controller magnitude required to track the user's reference trajectory in the final maze configuration when compared with the initial maze configuration. This suggests that users in both cohorts are able to learn pertinent aspects of the system dynamics and how to provide stable reference trajectories from the viewpoint of the automated controller.

We also find (Fig. 3.5) a statistically significant difference between the average controller magnitude, U, required to track reference trajectories provided by users in the static trust cohort versus those in the adaptive trust cohort, in the final maze configuration ($p < 0.01$). As we see no statistical evidence that one cohort outperforms the other in the first two maze configurations, we can infer that the adaptive trust formulation allows the system to adapt to the (possibly changing) abilities of the user, and so modulates the user input to provide reference trajectories that require less effort for the controller to stabilize to—*regardless of the abilities of the individual user*.

3.5.2. Average Controller Magnitude in Adaptive Trust Cohort

A more detailed analysis of how trust evolves in the adaptive trust cohort can be seen in Fig. 3.6. This plot presents preliminary results[4] that further suggest the controller is adapting to the abilities of the user, resulting in reference trajectories that require less effort to track. This plot breaks down the evolution of required controller effort over the course of the experiment based on users who finished the study with higher trust (red) than they began with, and those who finished the study with lower trust (blue).

[4]We say preliminary results because this test included only the members of the adaptive trust cohort, and divides them into two groups (consisting of 7 (red, Fig. 3.6) and 3 (blue, Fig. 3.6) subjects), and therefore provided a smaller sample size from which we draw conclusions.

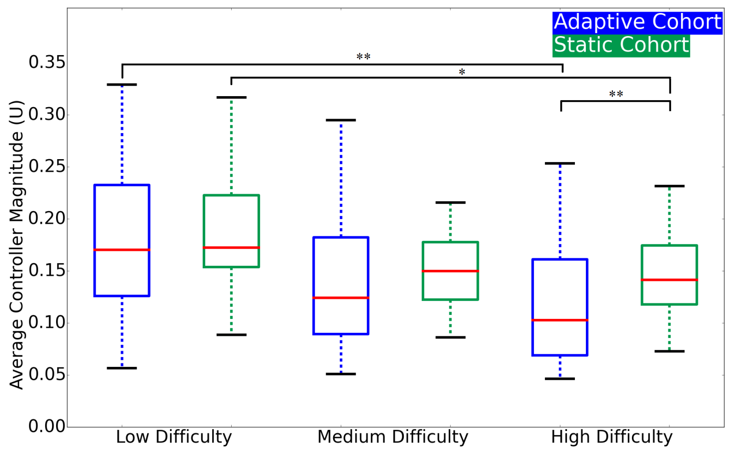


Figure 3.5. Average controller magnitude (U) per maze configuration for the static (green) and adaptive (blue) trust cohorts. We see a significant ($p < 0.01$) decrease in average controller magnitude between the initial maze configuration (low difficulty) and the final maze configuration (high difficulty) in both cohorts. We also see that the adaptive cohort requires a significantly ($p < 0.01$) diminished average controller magnitude than the static cohort in the final maze configuration. We do not see this difference in either the low or medium difficulty maze configuration which demonstrates that the rate of learning is significantly accelerated in the adaptive trust cohort. Key : * $p < 0.05$ and ** $p < 0.01$.

This plot shows that in cases where the automated system thinks the person can handle more trust, which corresponds with the user being given *greater* control bandwidth, there is a reduction in average control magnitude ($p < 0.05$). Additionally, when the system thinks the person requires less trust, which corresponds to *lower* control bandwidth, and we also see a reduction in average control magnitude ($p < 0.01$). This demonstrates that the results are not due to a specific modulation of the user input (e.g. saturation of the execution speed) as our hypothesis holds true whether the user sees an improvement or decline in their abilities to solve the maze task. That is, when the system's trust is adaptive, the operator is able to produce reference trajectories that require less effort to track than when the system's trust is static.

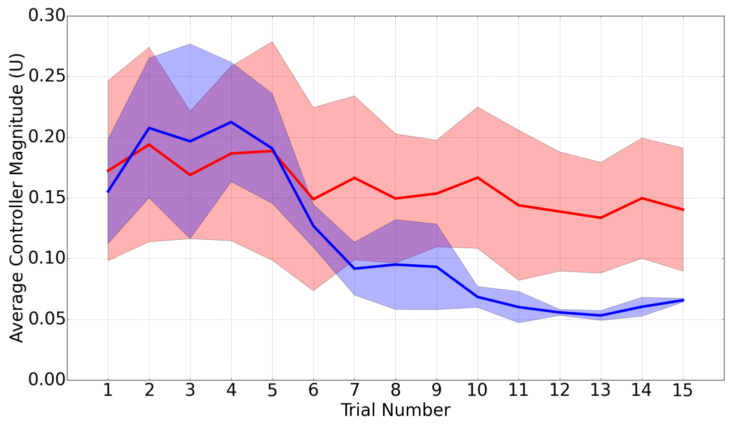


Figure 3.6. Evolution of the average controller magnitude (U) per trial. The data are divided into those subjects whose final trust value was higher (red) versus lower (blue) than the initial trust value. Mean (line) and standard deviation (variance envelope) are presented. We see a significant decrease in the required average controller magnitude both in users whose final trust value was lower ($p < 0.01$) and in users whose final trust value was higher ($p < 0.05$) than their initial value. This demonstrates that the results hold regardless of whether the initial control authority allocation is an over- or under-estimate of the user's expertise.

3.5.3. Trust Metric Over Time

Additionally, we find no single trend in the evolution of the trust values that produce the trend of decreasing average controller magnitude (Fig. 3.7). This helps elucidate the point that it is not simply an increase or decrease in the trust metric that allows a user to produce superior reference trajectories. Rather, from the standpoint of the automated controller, it is a combination of user performance, system learning and the adaptive trust level, which produces this trend.

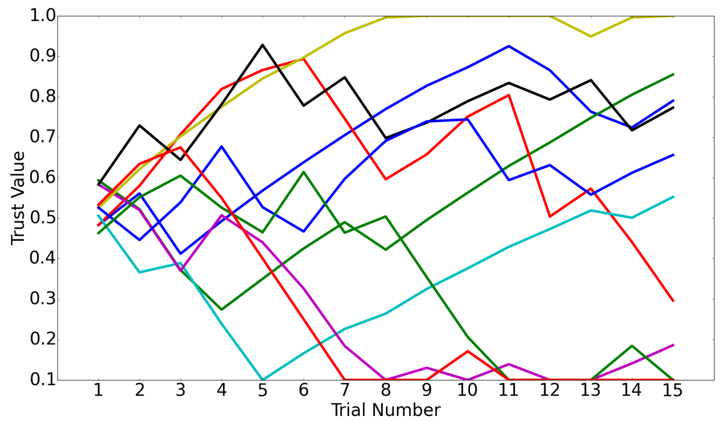


Figure 3.7. Evolution of trust over the course of the experiment (15 trials). Each line represents a single user in the adaptive trust cohort. No single pattern emerges, suggesting that the adaptive trust metric based on user performance is the causal variable in reducing the required average controller magnitude of the shared control system.

3.6. Conclusion

This chapter has presented a trust-based shared control framework that utilizes a control-theoretic measure of trust that an automated controller has in the operator. Results show that an adaptive trust metric, based on our control-theoretic formulation, was able to improve the ability of the shared control system to produce reference trajectories that require significantly ($p < 0.01$) less effort for the controller to track than those provided by users with a static trust metric. The reduced average controller magnitude, U, reflects the system's ability to learn appropriate methods for modulating the operator's input, resulting in reference trajectories that are easier to track. This work creates a foundation upon which to expand the trust-based shared control framework to include the online, continuous adaptation of trust, more granular user skill level classification, as well as applications to additional tasks and robot platforms.

CHAPTER 4

Data-driven Model-based Shared Control of Human-Machine Systems

The following chapter introduces the notion of data-driven, model-based shared control (MbSC) for human-machine systems. This methodology extends the efficacy of SC systems to scenarios in which we have no *a priori* knowledge of the system dynamics, or the human-in-the-loop. By integrating techniques from machine learning and optimal control, MbSC is able to generalize to any pair of human and machine partners. Additionally, as the data used to model the system dynamics is provided through demonstration, MbSC incorporates implicit information about the human partner in the learned model (which can be computed offline or online). Results of two human-subjects studies demonstrate that this approach can be used to improve the control skill demonstrated by a user in comparison to a user-only control paradigm. The associated code base is available online for free: `https://github.com/asbroad/model_based_shared_control`.

4.1. Introduction

Robot autonomy offers great promise as a tool by which we can enhance, or restore, the natural abilities of a human partner. For example, in the fields of assistive and rehabilitative medicine, devices such as exoskeletons and powered wheelchairs can be used to assist a human who has severely diminished motor capabilities. However, many assistive devices can be difficult to control. This can be due to the inherent complexity of the system, the required fidelity in the control signal, or the physical limitations of the human partner. We can, therefore, further

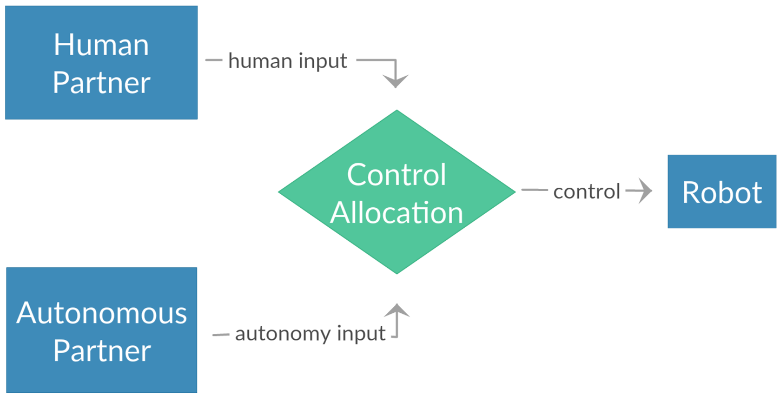


Figure 4.1. Pictorial representation of a shared control paradigm. Both the human and autonomy are capable of controlling the mechanical system, and a dynamic control allocation algorithm selects which agent is in control at any given moment.

improve the efficacy of these devices by offloading challenging aspects of the control problem to an autonomous partner. In doing so, the human operator is freed to focus their mental and physical capacities on important high-level tasks like path planning and interaction with the environment. This idea forms the basis of *shared control* (see Figure 4.1), a paradigm that aims to produce joint human-machine systems that are more capable than either the human or machine on their own.

A primary challenge that researchers and engineers face when developing shared control paradigms for generic human-machine systems is a lack of *a priori* knowledge of the human and robot partners. This issue is compounded by the fact that, in the real world, many users may operate the same mechanical device. It is therefore necessary to consider solutions that generalize to a variety of potential human and machine partners. In this chapter, we propose a data-driven methodology that learns all relevant information about how a given human and machine pair interact directly from observation. We then integrate the learned model of the joint system into a single shared control paradigm. We refer to this idea as *model-based shared control*.

In this chapter, we learn a model of the joint human-machine system through an approximation to the Koopman operator [**82**], though any machine learning approach could be used. This model is trained on observation data collected during demonstration of the human and machine interacting and therefore describes both the human's input to the system, and the robot's response to the human input and system state. We can then integrate the portion of the learned model that specifically describes the system and control dynamics of the mechanical device into an optimal control algorithm to produce autonomous policies. Finally, the input provided by the human and autonomous partners are integrated via a geometric signal filter to provide real-time, dynamic shared control of unknown systems.

We validate our thesis that modeling the joint human-machine system is sufficient for the purpose of automating assistance with a human subjects study consisting of 16 participants. We also demonstrate that our modeling technique is generalizable across users with results that suggest that individualizing the model offline, based on a user's own data, does not affect the ability to learn a useful representation of the dynamical system. We then compare and contrast the efficacy of linear model-based shared control (where linear constraints are placed on the modeling and control algorithms) with nonlinear model-based shared control (where these constraints are relaxed). This comparison incorporates data from a separate human subjects studying consisting of a second group of 16 participants. We find that nonlinear methods improve performance more than linear methods. Finally, we evaluate the efficacy of our shared control paradigm in an online learning scenario, demonstrating the sample efficiency of the model-based shared control paradigm.

We provide background and related work in Section 4.2. We then define model-based shared control in Section 4.3. In Section 4.4 we describe the human subjects study we perform and detail the results in Section 4.5. We describe important takeaways in Section 4.6 and conclude in Section 4.7.

4.2. Background and Related Work

This section presents background and related work in the shared control literature for human-machine systems. We also identify alternative methods of autonomous policy generation for shared control, and provide background on the Koopman operator [**82**] with a particular focus on its use in learning system dynamics.

4.2.1. Data-driven Shared Control

The effects of shared control (SC) have been explored in numerous fields in which the addition of a robot partner could benefit a human operator. For example, in assistive and rehabilitation robotics, researchers have explored the effects of shared control on teleoperation of smart wheelchairs [**53, 138**] and robotic manipulators [**80**]. Similarly, researchers have explored shared control as it applies to the teleoperation of larger mobile robots and human-machine systems, such as cars [**46**] and aircraft [**100**]. When dealing with systems of this size, safety is often a primary concern.

Our work is similar to prior art in shared control as we use automation to facilitate control of a robot by a human partner. However, in this work, we do not augment the user's control based on an explicit model of the user. Instead, we use observations of the user demonstrations to build a *model of the joint human-robot system*. The effect of the human partner on the shared control system is implicitly encoded in the model learned from their interactions.

4.2.2. Model-Based Optimal Control and Reinforcement Learning

Most closely related to the approach we describe in this paper is recent work that computes control trajectories by integrating learned dynamics models with model predictive control (MPC) algorithms [**149, 51**]. These algorithms are defined by an iterative, receding horizon optimization process instead of using an infinite-horizon. Similar to our own work, these researchers first collect observations from live demonstrations of the mechanical device to learn a model of the system dynamics. They then integrate the model with an MPC algorithm to develop control policies. Beyond methodological differences (e.g., choice of machine learning and optimal control algorithms), the key theoretical distinction between these works and our own is our focus on shared control of joint human-machine systems, instead of developing fully autonomous solutions. In particular, we learn a model of the joint system that is integrated into a shared control system to improve a human operator's control of a dynamic system. We therefore consider the influence of the human operator both during the data-collection process and at run-time in the control of the dynamic system. In this work, we learn a model of the system and control dynamics through an approximation to the Koopman operator [**82**].

4.3. Model-based Shared Control

Our primary goal is to develop a shared control methodology that improves the skill of human-machine systems without relying on *a priori* knowledge of the relationship between the human and the machine. To define our model-based shared control algorithm we now describe

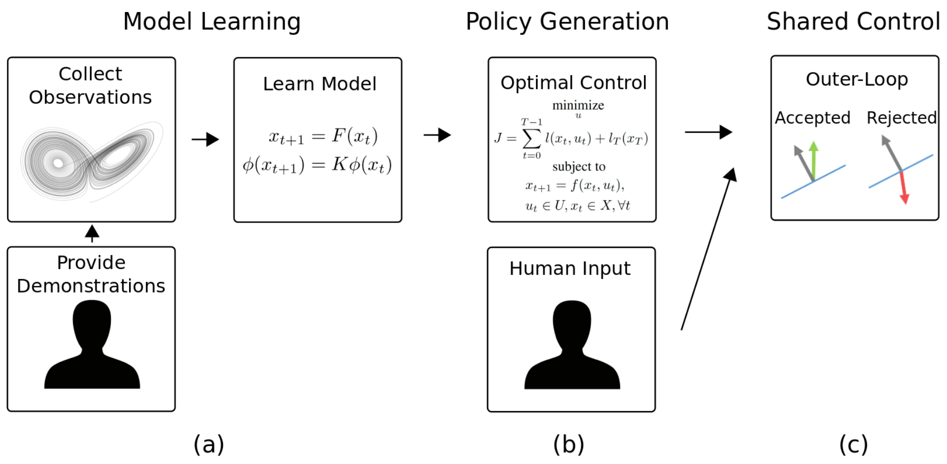


Figure 4.2. Pictorial depiction of the our model-based shared control paradigm. (a) Collect observations from user interaction and learn a model of the joint human-machine system through an approximation to the Koopman operator. This can be computed offline or online. (b) Compute control policy of autonomous agent by solving optimal control problem using the learned model. (c) Allocate control to integrate autonomy (gray) and user input (green/red).

the (1) model learning process, (2) method for computing the policy of the autonomous agent (*autonomy input* in Figure 4.1) and (3) control allocation method (the *green box* in Figure 4.1). A pictorial depiction of our model-based shared control paradigm can be found in Figure 4.2. Our learning-based approach develops a model of the joint human-machine system solely from observation, and this model can be used by the policy generation method to develop autonomous control trajectories. The control allocation method then describes how we integrate the input provided by the human partner and the autonomous agent into a single command that is sent to the dynamic system.

4.3.1. Model Learning via the Koopman Operator

When designing assistive shared control systems, it is important to consider both the human and autonomous partners. To ensure that our paradigm is valid for generic human-machine systems, we learn both the *system dynamics* and information about the *user interaction* directly from data. In this work, we develop a model of the joint human-machine through an approximation to the Koopman operator (see Chapter 2), which can be computed offline or online (discussed further in Section 4.5.3). The model learning process is depicted in Figure 4.2(a).

4.3.1.1. Basis. To approximate the Koopman operator using Extended Dynamic Mode Decomposition [**150**], we must define a basis or observation function (ϕ). This basis describes the space the approximate model operates on. In this work, we require that the finite basis ϕ *includes both the state and control variables* [**114**]. This ensures that the Koopman operator models both the natural dynamics of the mechanical system and the control dynamics as provided by the user demonstration. Here we choose ϕ such that

$$\begin{aligned}\phi =&[1, x_1, x_2, x_3, x_4, x_5, x_6, u_1, u_2, u_1 * x_1, u_1 * x_2, u_1 * x_3, u_1 * x_4, u_1 * x_5,\\ &u_1 * x_6, u_2 * x_1, u_2 * x_2, u_2 * x_3, u_2 * x_4, u_2 * x_5, u_2 * x_6, u_1 * cos(x_3),\\ &u_1 * sin(x_3), u_2 * cos(x_3), u_2 * sin(x_3)].\end{aligned} \tag{4.1}$$

These 25 basis functions were chosen to combine information about the geometry of the task (e.g., the trigonometric functions capture specific nonlinearities present in the system dynamics, see Section 4.4.1) with information related to how the user responds to system state. One can also choose the set of basis functions through data-driven techniques. For example, Sparsity Promoting DMD [**75**] imposes an ℓ_1 penalty during the learning process and therefore algorithmically decides which basis functions are the most relevant to the observable dynamics [**136**]. In this chapter we empirically select a fixed set of basis functions to ensure that all models (across the different users in our validation study) are learned using the same basis.

4.3.2. Autonomous Policy Generation

To generate an autonomous control policy, we can integrate the portion of the learned model that relates to the system and control dynamics into a model predictive control (MPC) algorithm. In particular, we use Koopman operator model-based control [**7, 31**], which we detail now in full. To compute the optimal control sequence, u, we must solve the following Model Predictive Control (MPC) problem

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & J = \sum_{t=0}^{T-1} l(x_t, u_t) + l_T(x_T) \\ \text{subject to} \quad & x_{t+1} = f(x_t, u_t), \\ & u_t \in U, x_t \in X, \forall t \end{aligned} \tag{4.2}$$

where $f(x_t, u_t)$ is the system dynamics, l and l_T are the running and terminal cost, and U and X are the set of valid control and state values, respectively.

To integrate our learned system model, we re-write the system dynamics as such:

$$\phi(x_{t+1}) = f_{\mathcal{K}}(x_t, u_t) \tag{4.3}$$

where $f_{\mathcal{K}} = \mathcal{K}^T \phi(x_t, u_t)$ is the learned system dynamics parameterized by a Koopman operator $\mathcal{K}$. This equation demonstrates the fact that the Koopman operator does not map directly from state to state, but rather operates on functions of state. We can then evaluate the evolved state by recovering the portion of the basis that represents the system's state

$$x_{t+1} = \phi(x_{t+t})_{1:N} \tag{4.4}$$

where values $1:N$ are the state variables, as per our definition in Equation (4.1), and N is the dimension of the state space. The policy generation process is depicted in Figure 4.2(b).

4.3.2.1. Nonlinear Model Predictive Control Algorithm. We solve Equation (4.2) with Sequential Action Control [**17**] (SAC). SAC is a real-time, model-based non-linear optimal control algorithm that is designed to iteratively find a single value, a time to act, and a duration that maximally improves performance. Other viable nonlinear optimal control algorithms include iLQR [**95**] and DDP [**102, 134**]. SAC is particularly well suited for our shared control algorithm because it searches for single, short burst actions which aligns well with our control allocation algorithm (described in detail in Section 4.3.3). Additionally, SAC can compute closed-loop trajectories very quickly (1 kHz), an important feature for interactive human-machine systems such as the one presented in this chapter.

4.3.2.2. Integrating the Koopman model and SAC. Sequential Action Control is a gradient-based optimization technique and it is therefore necessary to compute derivatives of a system during the optimization process. The linearization of the discrete time system is defined by the following equation

$$x_{t+1} = Ax_t + Bu_t.$$

By selecting a differentiable ϕ, one can compute A and B

$$\begin{aligned} A &= \mathcal{K}_{1:N}^T \frac{\partial \phi}{\partial x} \\ B &= \mathcal{K}_{N:N+P}^T \frac{\partial \phi}{\partial u} \end{aligned} \tag{4.5}$$

where N is again the dimension of the state space, and P is the dimension of the control space.

4.3.3. Control Allocation Method

To close the loop on our shared control paradigm, we define a control allocation method that uses the solution from the optimal control algorithm to provide outer-loop stabilization. We use a geometric signal filter that is capable of dynamically shifting which partner is in control at any given instant based on optimality criteria. This technique is known as Maxwell's Demon Algorithm (MDA) [**142**]. Our specific implementation of MDA is detailed in Algorithm 1.

Algorithm 1 Maxwell's Demon Algorithm (MDA)

```
if $\langle u_h, u_a \rangle \geq 0$ then
    $u = u_h$;
else
    $u = 0$;
end if
```

where u_h is the control input from the human operator, u_a is the control produced by the autonomy, and u is applied to the dynamic system. We also provide a pictorial representation of the algorithm in Figure 4.3.

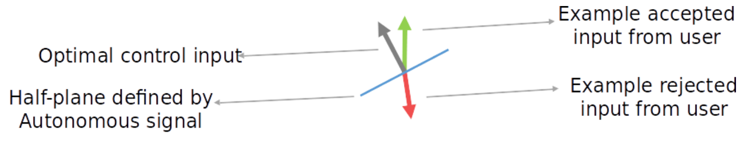

Optimal control input
Half-plane defined by Autonomous signal
Example accepted input from user
Example rejected input from user

Figure 4.3. Maxwell's Demon Algorithm (MDA)

This control allocation method restricts the user's input to the system to be in the same half-plane as the optimal control solution, and places no other limitations on the human-machine interaction. If the user's input is in the opposite half-plane, no input is provided to the system. This control allocation method is lenient to the human partner, as notably, *the autonomous agent does not add any information into the system* and instead only blocks particularly bad input from the user. Therefore, *any signal sent to the system originates from the human partner*. We use this filter because we are motivated by assistive robotics, in which prior research has shown that there is no consensus across users on desired assistance level [**53**]. By allowing the user a high level of control freedom, the system encourages input from the human operator and restricts how much authority is granted to the autonomous partner. This method is depicted in Figure 4.2(c).

4.4. Human Subjects Study

Here, we detail the experimental setup that we use to study three main aspects of the described system.

- First, our aim is to evaluate the efficacy of model-based shared control as it relates to task success and control skill. Concurrently, we aim to evaluate the generalizability of the learned system models with respect to a wide range of human operators.
- Second, we aim to evaluate the efficacy of model-based shared control under an online learning paradigm—specifically, the sample-efficiency of the Koopman operator representation.
- Finally, we aim to evaluate the impact of nonlinear modeling and policy generation techniques through a comparison to a second human-subjects study that enforces linear constraints on our model-based shared control algorithm.

4.4.1. Experimental Environment

The proposed shared control framework is evaluated using a simulated lunar lander (see Figure 4.4). We use a simulated lunar lander (rocket) as our experimental environment for a

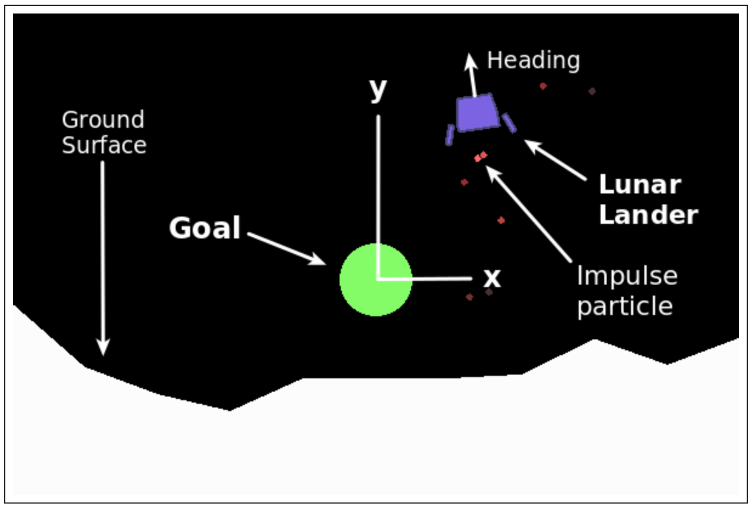


Figure 4.4. Simulated lunar lander system. The green circle is the goal location. The red dots represent an engine firing.

number of reasons. This environment is challenging for a novice user, but performance can be improved (and sometimes mastered) given enough time and experience. Similar to a real rocket, one of the main control challenges is the stability of the system. As the rocket rotates along its yaw axis, firing the main thruster can produce nonintuitive dynamics for a novice. Furthermore, once the rocket has begun to rotate, momentum can easily overwhelm a user who is unfamiliar with such systems. Therefore, it is often imperative—particularly for non-expert users—to maintain a high degree of stability at all times in order to successfully complete the task. In addition to the control challenges, we choose this environment because the simulator abstracts the system dynamics through calls to the Box2D physics engine; therefore, we do not have an exact model and thus have an *explicit need to learn one*.

4.4.2. System Description

The dynamic system is a modified version of an open-source environment implemented in the Box2D physics engine and released by OpenAI [**34**]. Our modifications (1) allow for continuous-valued multi-dimensional user control via a joystick, and (2) incorporate the code-base into the open-source ROS framework. We have made our code available online at `https://github.com/asbroad/model_based_shared_control`.

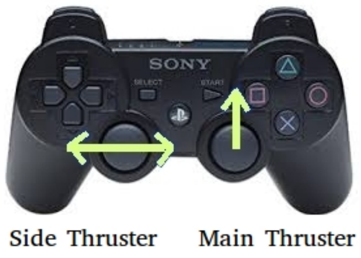
The lunar lander is defined by a 6D state space made up of the position (x, y), heading (θ), and their rates of change $(\dot{x}, \dot{y}, \dot{\theta})$. The control input to the system is a continuous two dimensional vector (u_1, u_2) which represents the throttle of the main and rotational thrusters. The main engine can only apply positive force. The left engine fires when the second input is negative, while the right engine fires when the second input is positive. The main engine applies an impulse that acts on the center of mass of the lunar lander, while the left and right engines apply impulses that act on either side of the rocket. We remind the reader that our goal is to learn both the system dynamics and user interaction. For this reason, we must collect data both on the system state and also the control input. Together, this defines an eight dimensional system:

$$\mathcal{X} = [x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}, u_1, u_2]$$

where the first six terms define the lunar lander state and u_1, u_2 are the main and rotational thruster values, through which the user interacts with the system.

4.4.3. Trial Description

The task in this environment requires the user to navigate the lander from its initial location to the goal location (represented by the green circle in Figure 4.4) and to arrive with a heading nearly perpendicular to the ground plane and with linear and rotational velocities near zero. A trial is considered complete either (1) when the center of an upright lunar lander is fully contained within the goal circle (i.e., the Euclidean distance between the center of the lander and the center of the goal is less than 0.9 m) and the linear and angular velocities are near zero (i.e., the linear velocities must be less than 1.0 m/s and the angular velocity must be less than 0.3 rad/sec), or (2) when the lander moves outside the bounded environment (i.e., when the lander moves off the screen to the left or right) or crashes into the ground.



In each trial, the lunar lander is initialized to the same x, y position (10.0 m, 13.3 m), to which we added a small amount of Gaussian noise ($\mu = 0.2$ m). Additionally, a random force is applied at the start of each trial (uniform(-1000 N,1000 N)). The goal location (10.0 m, 6.0 m) is constant throughout all trials and is displayed to the user as a green circle (see Figure **??**).

The operator uses a PS3 controller to interact with the system. The joystick controlled by the participant's dominant hand fires the main thruster, and the opposing joystick fires the side thrusters. As the user moves through the environment, we keep track of the full state space at each timestep (10 Hz). We provide a video of the system, task and user interaction under shared control as part of the supplementary material.

4.4.4. Analysis I : Efficacy and Generalizability of Model-based Shared Control

4.4.4.1. Control Conditions. To study the efficacy and generalizability of our shard control system and the generalizability of the learned system dynamics, we compare four distinct control conditions.

- In the first condition, the user is in full control of the lander and is not assisted by the autonomy in any way; we call this approach *User Only* control. As each user undergoes repeated trials with the same goal, this can also be considered a natural learning paradigm.

In the remaining three conditions an autonomous agent provides outer-loop stabilization on the user's input as described in Section 4.3. The main distinction between these three control conditions is the source of the data used to compute the model of the joint system.

- In the second condition, the model is defined by a Koopman operator learned on data captured from earlier observations of the current user; we call this approach *Individual Koopman*.
- In the third condition, the model is defined by a Koopman operator learned on data captured from observations of three novice participants prior to the experiment (who were not included in our analysis); we call this approach *General Koopman*.
- In the fourth condition, the model is defined by a Koopman operator learned on data captured from observations of an expert user (the first author of the paper, who has

significant practice controlling the simulated system); we call this approach *Expert Koopman*.

We analyze the viability of model-based shared control by comparing the *User Only* condition to each of the shared control conditions. We analyze the generalizability of the learned models by comparing the results from the *Individual Koopman*, *General Koopman* and *Expert Koopman* conditions.

4.4.4.2. Protocol and Participants. Each experiment begins with a training period for the user to become accustomed to the dynamics of the system and the interface. This training period continues until the user is able to successfully achieve the task three times in a row or 15 minutes elapses. During the next phase of the experiment, we collect data from 10 user-controlled trials, which we then use to develop a model. Finally, each user performs the task under the four conditions detailed above (10 trials each). The order in which the control paradigms are presented to the user is randomized and counter-balanced to reduce the effect of experience.

The study consisted of 16 participants (11 female, 5 male). All subjects gave their informed consent and the experiment was approved by Northwestern University's Institutional Review Board.

4.4.5. Analysis II : Online Model-based Shared Control

To study the efficacy of our model-based shared control algorithm in an online learning paradigm, we collect data from a fifth experimental condition, which we call *Online Koopman*.

4.4.5.1. Control Condition.

- The main difference between the *Online Koopman* paradigm and the three previously described shared control conditions is that the model of the joint human-machine system is learned online in real-time. In all other control conditions, all models were trained offline from observations gathered during a data collection phase. In the online paradigm, the model is updated continuously starting with the first collected set of observations.

In addition to the lack of a separate data collection phase, the online learning paradigm is distinct from the other shared control conditions because of the data that we use to learn the model. In the shared control conditions that use a model learned offline, we use all of the observations collected from the user demonstrations to learn the model. In the online learning paradigm we only update the model when the user input is admitted by the MDA controller. We choose this learning paradigm because it fits well conceptually with our long term goal of using the outer-loop algorithm to provide stability and safety constraints on the shared control system.

4.4.5.2. Protocol and Participants. The online learning paradigm consists of 15 trials per user to allow us to evaluate possible learning curves. The model is updated at the same rate as the simulator (10 Hz) and is initialized naively (i.e., all values are sampled from a uniform distribution [0,1)). This paradigm is presented as the final experimental condition to all subjects. The subjects are the same 16 participants as in Section 4.4.4.

4.4.6. Analysis III : Comparison of Linear and Nonlinear Model-based Shared Control

To study the impact of nonlinear modeling and policy generation techniques on our model-based shared control paradigm, we compare results from the above study to a second study (consisting of a separate group of 16 participants) that enforces linear constraints on these parts of the system.

4.4.6.1. Control Conditions. The same four control conditions from Analysis I are evaluated. The differences lie in (1) the choice of basis used to approximate the Koopman operator and (2) the choice of optimal control algorithm used to generate the autonomous policy. In this study, we use a linear basis, instead of a nonlinear basis, to approximate the Koopman operator, which consists of the first nine terms in Equation (4.1). We furthermore use a Linear Quadratic Regulator, instead of a nonlinear model predictive control (MPC) algorithm (Sequential Action Control (SAC)) to generate the autonomous partner's control policy.

4.4.6.2. Protocol and Participants. The same experimental protocol described in Section 4.4.4 was used, allowing us to perform a direct comparison between the two studies. The data was previously analyzed in [**31**] and was collected from 16 additional subjects, resulting in 32 total participants. The code for the linear study is also provided online for free : `https://github.com/asbroad/koopman_operator_model_learning`.

4.4.7. Statistical Analysis

We analyze the results of the human-subjects studies using statistical tests to compare the performance of participants along a set of pertinent metrics under the control conditions described in Section 4.4.4. Our analysis consists of one-way ANOVA tests conducted to evaluate the effect of the shared control paradigm on each of the dependent variables in the study. These tests allow us to statistically analyze the effect of each condition while controlling for overinflated type I errors that are common with repeated t-tests. Each test is computed at a significance value of 0.05. When the omnibus F-test produces significant results, we conduct post-hoc pair-wise Student's t-tests using Holm-Bonferroni adjusted alpha values [**151**]. The post-hoc t-tests allow us to further evaluate the cause of the significance demonstrated by the ANOVA by comparing each pair of control paradigms separately. Similar to the ANOVA test, the Holm-Bonferroni correction is used to reduce the likelihood of type I errors in the post-hoc t-tests.

In addition to reporting the results of the statistical tests, we also use box-and-whisker diagrams to display specific metrics. In these plots, the box represents the *interquartile range (IQR)* which refers to the data that lies between the first and third quartiles. This area contains 50% of the data. The line inside the box represents the median value and the whiskers above and below the box are the minimum and maximum values inside 1.5 times the interquartile range. The small circles are outliers. The plots also depict the results of the reported statistical tests. That is, if a post-hoc t-test finds statistically significant differences between two conditions, we depict these results on the box-and-whisker diagrams using asterisks to represent the significance level ($* : p < 0.05, ** : p < 0.01, *** : p < 0.005$).

We note that this analysis is used for *all reported results*. Therefore, if we present the results of a t-test, it signifies that we have previously run an ANOVA and found a statistically significant difference. The reader can also assume that any unreported post-hoc t-tests failed to reject the null hypothesis.

4.5. Results

We now present the results of the desired analyses described in Sections 4.4.4, 4.4.5, and 4.4.6. Our analyses support the premise that model-based shared control is a valid and effective data-driven method for improving a human operator's control of an *a priori* unknown dynamic system. We also find the learned system models are generalizable across a population of users. Finally, we find that these models can be learned online in a fast, data-efficient manner.

4.5.1. Efficacy of Model-based Shared Control

To evaluate the efficacy of our model-based shared control algorithm, we compute the average success rate under each control paradigm and examine the distribution of executed trajectories. Our analysis compares the User Only control condition to each of the shared control conditions (Individual Koopman, General Koopman and Expert Koopman).

4.5.1.1. Task Success and User Skill. A trial is considered a success when the user is able to meet the conditions defined in Section 4.4.3. We can interpret the success rate of a user, or shared control system, on a set of trials as a measure of skill. The greater the skill, the higher the success rate. By comparing the average success rate under the User Only control paradigm with the average success rate under the shared control paradigms, we can analyze the impact of the assistance provided by the autonomous partner.

The average number of successful trials produced in each control condition are displayed in Figure 4.5. An analysis of variance shows that the choice of control paradigm has a significant effect on the success rate ($F(3, 59) = 4.58, p < 0.01$). Post-hoc t-tests find that users under the shared control conditions show statistically significant improvements in the success rate when compared to their performance under the User Only control condition ($p < 0.01$, for all

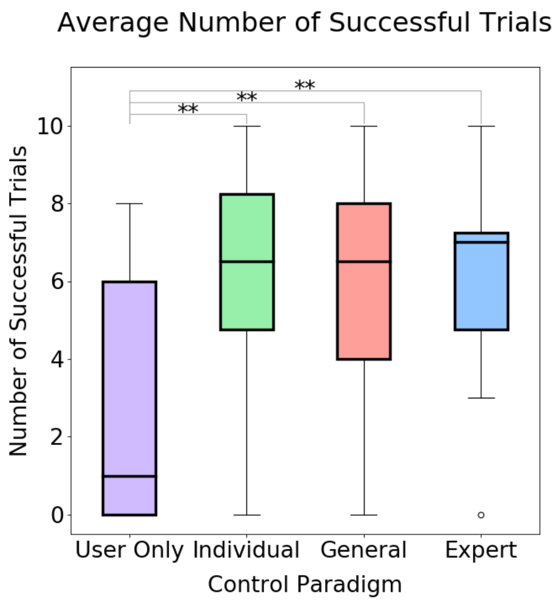


Figure 4.5. Number of successful trials under each control condition.

cases). No other pairings are found to be statistically distinct. This result demonstrates that the assistance provided by the autonomous agent significantly improves the skill of the joint human-machine system, thereby validating the efficacy of model-based shared control. It also suggests that the source of the data used to learn the model may not be important in developing helpful autonomous assistance in the shared control of dynamic systems (discussed further in Section 4.5.2).

4.5.1.2. Distribution of Trajectories—Qualitative. We further analyze the different control conditions through a comparison of the distribution of trajectories we observe in each condition. Unlike the success metric, this analysis is not based on task performance, and is instead performed to evaluate the control skill exhibited by either the human operator alone or the joint human-machine system. Figure 4.6 depicts trajectory plots which represent the most frequently occupied sections of the state space. The plots are generated using data separated based on the

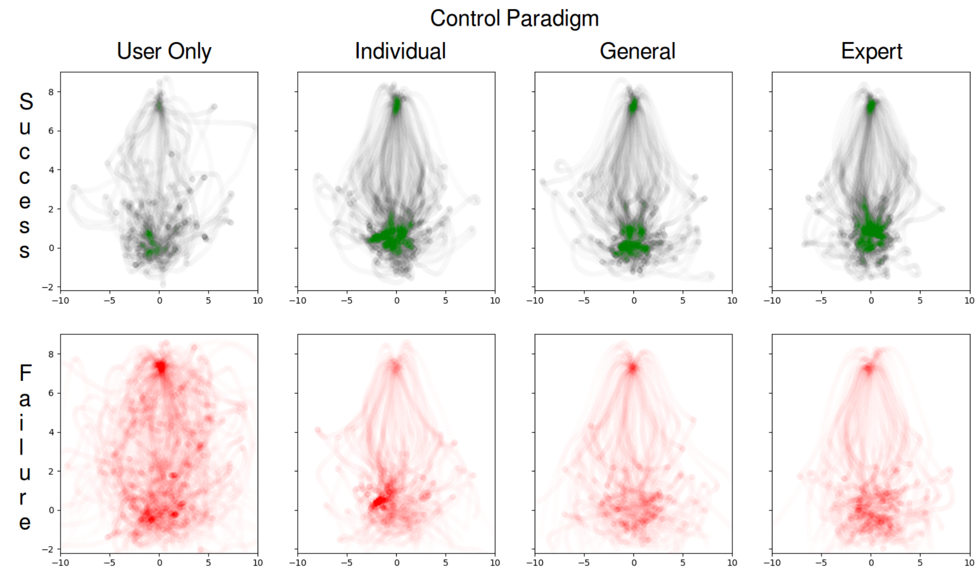


Figure 4.6. Trajectory plots which visualize the most frequently visited parts of the state space. The data is broken down by control condition (columns) and whether the trial was successful (rows). The plots are generated by overlaying each trajectory with a low opacity and the intensity of the plots therefore represents more frequently visited portions of the state space.

control condition (columns) and whether the user was able to complete the task on a given trial (rows).

The first distinction we draw is between the User Only control condition and the three shared control conditions. In particular, the distribution of trajectories in the User Only condition depicts both larger excursions away from the target and lower levels of similarity between individual executions. When we focus specifically on which parts of the state space users spend the most time in (as represented by the intensity of the plots), we see two main clusters of high intensity (around the start and goal locations) in the shared control conditions, whereas we see a wider spread of high-intensity values in the User Only control condition. This suggests more purposeful motions under the shared control conditions.

The second distinction we draw focuses on a comparison between the successful and unsuccessful trials. Specifically, we note that trajectory plots computed from the failed trials under the shared control conditions demonstrate similar properties (e.g., the extent of the excursions away from the target, as well as two main clusters of intensity) to the trajectory plots computed from successful trials under the shared control conditions. This suggests that users may have been closer to succeeding in these tasks than the binary success metric gives them credit for. By comparison, the trajectory plot computed from the failed trials under the User Only control condition depicts a significantly different distribution of trajectories with less structure. Specifically, we observe numerous clusters of intensity that represent time spent far away from the start and goal locations. This suggests that users were particularly struggling to control the system in these cases.

4.5.1.3. Distribution of Trajectories—Quantitative. These observations are supported by an evaluation of the ergodicity [**59, 99**] of the distributions of trajectories described above. We find users under the shared control paradigm are able to produce trajectories that are more ergodic with respect to the goal location then users under User Only control, which means that they spend more a significantly larger proportion of their time navigating near the goal location under shared control. To perform this comparison, we compute the ergodicity of each trajectory with respect to a probability distribution defined by a Gaussian centered at the goal location (which represents highly desirable states). This metric can be calculated as the weighted Euclidean distance between the Fourier coefficients of the spatial distribution and the trajectory [**103**].

Similar to our qualitative analysis of the trajectory plots in Figure 4.6, we first compare ergodicity between the different control conditions by analyzing *all* the trajectories observed under each condition. An analysis of variance showed that the effect of the shared control paradigm on trajectory ergodicity is significant ($F(3, 640) = 12.97, p < 0.00001$). Post-hoc t-tests find statistically significant differences between the performance of the users in the User Only control condition and users in the shared control conditions based on the individual, general and expert datasets ($p < 0.0005, p < 0.001, p < 0.0005$, respectively). No other pairings demonstrate statistically distinct results. We interpret this result as additional evidence that model-based shared control improves the skill of the human partner in controlling the dynamic system.

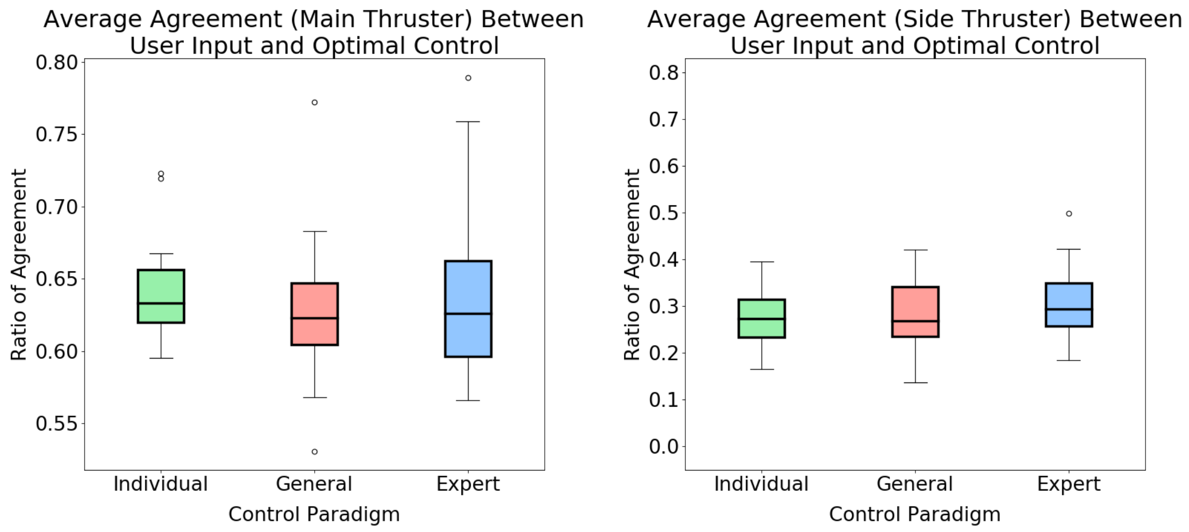
We further analyze the ergodicity results by separating the trajectories based on whether they come from an unsuccessful or successful trial. An analysis of variance computed over all control conditions showed that the effect of the shared control paradigm on trajectory ergodicity is significant for both unsuccessful ($F(3, 310) = 6.60, p < 0.0005$) and successful ($F(3, 325) = 7.20, p < 0.0005$) trials. Post-hoc t-tests find statistically significant differences between the performance of the users in the User Only control condition and users in the shared control conditions ($p < 0.005$ in all unsuccessful cases, $p < 0.05$ in all successful cases). No other pairings reject the null hypothesis. These results suggest that the shared control paradigm is helpful in improving the user's skills even when they provide input that is ultimately unsuccessful in achieving the task. Furthermore, our shared control paradigm is helpful, even when user's are performing at their best. Thus, for both failed and successful trials, users exhibit a greater amount of control skill than when there is no assistance.

4.5.2. Generalizability of Shared Control Paradigm

We continue the evaluation of our human subjects study with an analysis of the generalizability of the learned system models and our model-based shared control algorithm. As reported in Section 4.5.1, we find no statistical evidence that the source of the data used to train the model impacts the efficacy of the shared control paradigm. When we compare the success rate of users in each shared control condition, we find no statistically significant difference. However, we do find a significant difference between the user's performance under each shared control condition and the User Only condition. The same result holds when we compare each control condition along the ergodic metric described in Section 4.5.1 and visualized by trajectory plots in Figure 4.6. Taken together, these results suggest that the efficacy of the assistance provided by the autonomous agent is *independent of the source of the data used to learn a model of the joint system*. That is, models trained on data collected from an individual user generalize to a larger population of human partners.

To further analyze the generalizability of the model-based shared control paradigm, we evaluate the participants' interactions with the outer-loop autonomous control. We are interested in whether or not users agree more often with the autonomy when control signals are produced based on models learned from their personal demonstration data. To evaluate this idea, we look at the percentage of user inputs that are let through as control to the dynamic system based on our control allocation method (MDA). The average agreement metric is broken down by control condition and presented in Figure 4.7.

An analysis of variance shows that the effect of the source of the model data on the average agreement is not significant in either the main thruster ($F(2, 44) = 0.87, p = 0.43$) nor the side thruster ($F(2, 44) = 0.38, p = 0.69$). These results show a uniformity in the response to system



(a) (b)

Figure 4.7. Average agreement between user and optimal control algorithm as defined by the Maxwell's Demon Algorithm (Equation (1)) along the (a) main and (b) side thrusters.

state across users and suggest that the system is able to adapt to the person, instead of requiring a personalized notion of the user and system.

We interpret this finding as further evidence of the generalizability of our model-based shared control paradigm. In particular, we find that it is not necessary to incorporate demonstration data from individual users when developing model-based shared control. This result replicates findings from our analysis of data collected under a shared control paradigm that enforced a linear constraint on the model learning and policy generation techniques [**31**].

4.5.3. Online Learning Shared Control

We next evaluate our model-based shared control algorithm in an online learning paradigm. Our evaluation considers the sample complexity of our model-based learning algorithm through a comparison of the *impact each shared control paradigm has on the skill of the joint system* *over time*. Our statistical analysis is a comparison of the percent of participants who succeed under each paradigm *by trial number*, shown in Figure 4.8. We remind the reader that users participate in 15 trials of the Online Koopman condition while they participate in 10 trials of the four other experimental conditions. For comparison we only plot the first 10 trials of the Online Koopman data, though we note that the improved success rate is sustained over the final five trials. From this plot, we can see that users in the Online Koopman shared control condition start off performing poorly, but by around trial 7 start performing on par with the other shared control conditions.

Here, we also note the number of trials used to train the model of the system and control dynamics in each condition. In the Individual and Expert conditions, data is collected from 10 trials to train the model. In the General condition, data is collected from three different users who each control the system for 10 trials each, which means the model is trained from a total of 30 trials. Finally, as discussed above, in the Online condition, the model is learned continuously over the course of 15 trials.

To provide quantitative evidence of this visual trend, we perform the same types of statistical analyses as in previous sections, but now include data from the *Online Koopman* as a fifth experimental condition. For ease of discussion we refer to the *Individual*, *General* and *Expert Koopman* model-based shared control conditions as the offline learning conditions, and the *Online Koopman* model-based shared control as the online learning condition. As users provide more data in the Online Koopman condition than in all other conditions, we perform two sets of analyses. First, we compare the data from the first ten trials from the Online Koopman condition to all other control conditions. We then re-perform the same tests, but use the final ten trials from the Online Koopman condition. By comparing these results, we can evaluate the efficacy

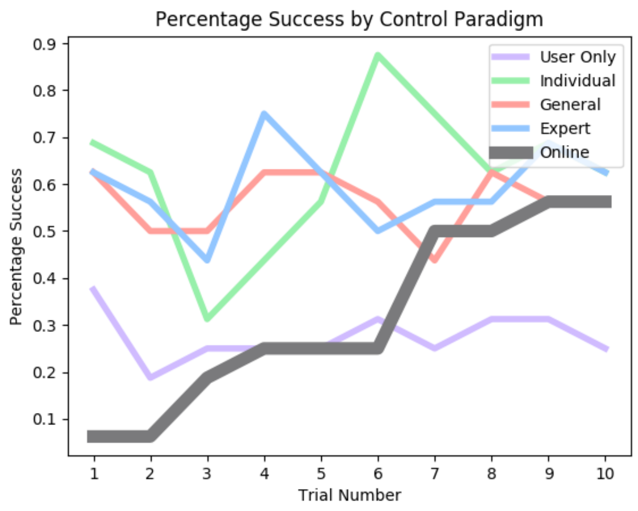


Figure 4.8. Average percentage success by trial for the first 10 trials by control condition. Users under all shared control conditions using models learned offline (Individual, General, Expert) outperform the User Only control condition across all trials. Users under the shared control condition using models learned online (Online) start off performing poorly, but quickly begin to outperform the User Only control condition and, in the end, achieve the same level of success as those under the offline shared control conditions.

of the online learning paradigm, and also analyze the effect of the amount of data used during the learning process.

4.5.3.1. Statistical Analysis of the First Ten Trials. An analysis of variance finds a statistically significant difference between the various control conditions along the primary success metric ($F(4, 74) = 5.35, p < 0.001$). Post-hoc t-tests find that all offline learning conditions significantly outperform the Online Koopman and User Only control conditions ($p < 0.05$ for all cases). We do not find the same statistically significant difference between the User Only and Online Koopman conditions. These results suggest that users under the online learning paradigm initially perform on par with how they perform under the user only control paradigm, but worse than under the offline control conditions. This analysis is consistent with our expectations since, in the online condition, the model of the joint system is initialized randomly and therefore does a poor job of assisting the user. However, it is also important that this online shared control does not degrade performance in comparison to the User Only paradigm, suggesting that there is little downside to employing the online learning paradigm during learning.

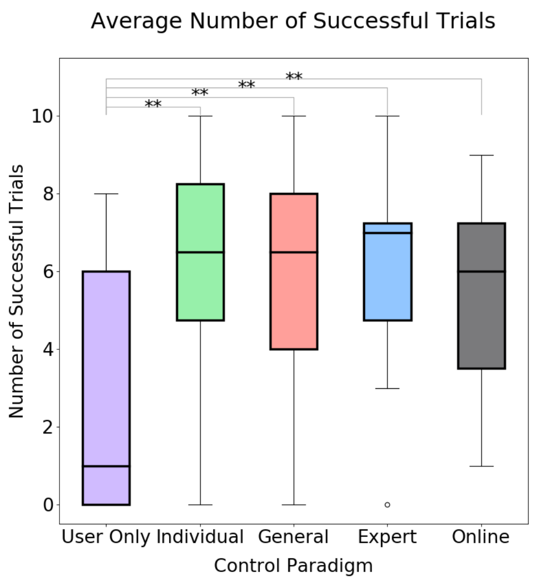


Figure 4.9. Number of successful trials under each control condition (including an online learning paradigm). We find statistically significant differences between the User Only condition and each shared control condition ($p < 0.01$).

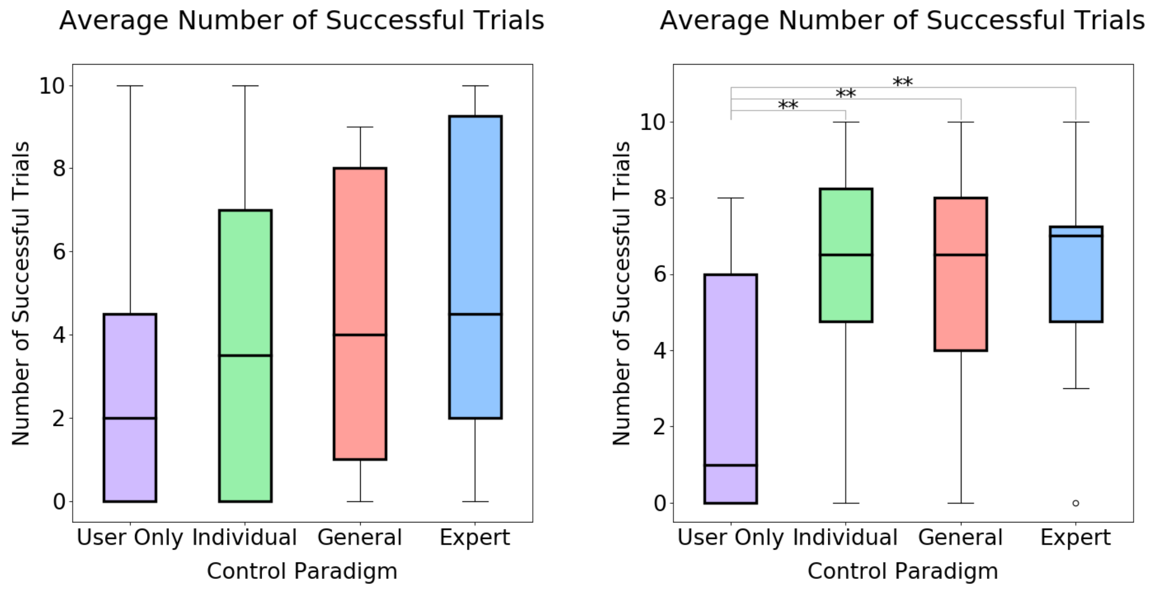
4.5.3.2. Statistical Analysis of the Final Ten Trials. As a point of comparison, we now re-run the same statistical tests using the final ten trials from the Online Koopman condition. An analysis of variance finds a statistically significant difference between the various control conditions along the primary success metric ($F(4, 74) = 3.55, p < 0.05$) (see Figure 4.9). Post-hoc t-tests find that all shared control conditions (using models learned offline and online) significantly out perform the User Only control paradigm ($p < 0.01$ for all conditions). This result is different from our analysis of the first ten trials and suggests that the learned model improves significantly with more data and now is on par with the models learned in the offline conditions. No other pairings show statistically significant differences.

The visual trend present in Figure 4.8 and the statistical analysis demonstrated in Figure 4.9 suggest that the Koooman operator is able to quickly learn an actionable representation of the joint human-machine system. These results also demonstrate the efficacy of our model-based shared control algorithm in an online learning scenario and in limited data regimes.

4.5.3.3. Covariate Shift. One final piece of qualitative analysis that we provide here relates to a difference in the data used to learn the system dynamics model in the offline and online learning settings. In particular, in the offline learning paradigm, the data used to learn the system dynamics model includes all of the observations collected during the human partner's control of the lunar lander (i.e. without any assistance). In contrast, in the online learning paradigm, data is only collected, and used to learn the system dynamics model, when the input passes through the MDA filter. One might imagine that this difference could lead to an issue known as the covariate shift problem [**118**] (see Chapter 5); however, the similarity of the results in both settings suggest this is not an issue for our system. One reason this may be true is that the covariate shift problem tends to be less pervasive in system identification than in policy learning [**20**]. It can pose an issue when the system exhibits state or time dependent dynamics, but the lunar lander exhibits the same dynamics everywhere in the state space. Early experiments that tried to learn the system dynamics model from random inputs suggest that it is particularly important to learn how the lunar lander behaves near points of equilibrium, as this allows the autonomous controller to develop stable policies. For this reason, we include a human-in-the-loop during model learning, as the human operator often explicitly tries to maintain system safety.

4.5.4. Linear and Nonlinear Model-based Shared Control

The final piece of analysis we perform in this chapter is an related to the impact that non-linear modeling and policy generation techniques have on our model-based shared control paradigm. For this analysis we compare the User Only control condition to the three offline shared control conditions.



(a) Average success rate under linear model-based shared control and user only control.

(b) Average success rate under nonlinear model-based shared control and user only control.

Figure 4.10. Comparison of linear and nonlinear model-based shared control paradigms.

The average success rate of users under each control paradigm for both studies is presented in Figure 4.10. In the linear study [**31**] we observe a trend (see Figure 4.10a) that suggests users perform better under the shared control paradigm, but we do not find statistically significant evidence of this observation. In contrast, we find that model-based shared control using nonlinear modeling and policy generation techniques does statistically improve the success rate when compared to a User Only control paradigm.

One potential explanation for the difference we find in the results of the two studies is that the nonlinear basis produces more accurate models of the system dynamics then the linear basis. To explore this explanation, we evaluate the predictive capabilities of a Koopman operator learned with a linear basis to one learned with a nonlinear basis. This analysis is performed by comparing the predicted system states with ground truth data. We evaluate the error (mean and variance) as a function of prediction horizon (a.k.a. the H-step error). Figure 4.11 depicts the raw error (in meters) of Koopman operators trained using linear and nonlinear bases.

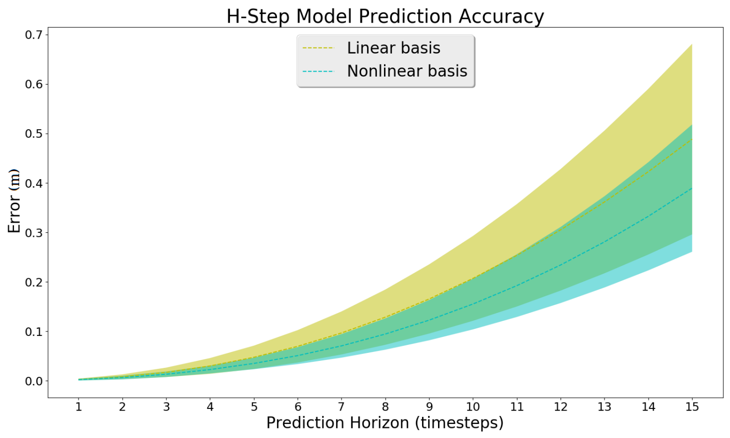


Figure 4.11. H-step prediction accuracy of Koopman operator models based on linear and nonlinear bases. Error is computed as the Euclidean distance between the predicted (x, y) values and the ground truth (x, y).

Our analysis of the predictive capabilities of the Koopman operator models demonstrates that each is highly accurate. The nonlinear model does slightly outperform the linear model as the prediction horizon grows, however, we find that both models are able to produce single-step predictions with error on the scale of 10^{-3} meters. As a reminder to the reader, the state space is bounded with $X \in (-10, 10), Y \in (0, 16)$. This analysis suggests that the choice of basis function does not cause the observed difference in average success rate between the two studies. Instead, the important design decision may be the choice of model predictive control algorithm. In the linear study we use an infinite horizon LQR to produce autonomous control policies, whereas in the nonlinear study we use a receding-horizon Model Predictive Control (MPC) to produce autonomous control. Our interpretation of these results is that the receding horizon nature of MPC is better suited to the visual planning approach that human operators use when solving the lunar lander task.

4.6. Discussion

In this section, we highlight a number of main takeaways that stem from our analysis. To begin, the results of our human-subjects studies demonstrate that our model-based shared control paradigm is able to (1) successfully learn a model of the joint human-machine system from observation data, and (2) use the learned system model to generate autonomous policies that can help assist a human partner achieve a desired goal. We evaluate the predictive capabilities of the learned system models through a comparison to ground truth trajectory data (see Figure 4.11) and evaluate the impact of the assistive shared control system through a comparison of performance (success rate, see Figure 4.5) with a User Only (or natural learning) control paradigm. All analyses support the idea that MbSC can help improve the control skill of a human operator both when they are able to achieve a task on their own and when they are not.

Additional evaluations demonstrate that the learned system and control dynamics generalize across users, and suggests that,unlike in other human-machine interaction paradigms, personalization is not required for defining shared control paradigms of generic human-machine systems. Specifically, we find that the demonstration data used to learn the system and control models does not need to come from an optimal, or expert, controller, and can instead come from *any* human operator. Therefore, at a base level, the controller does not need to be personalized to each individual user as the learned model captures all necessary information. This idea is important for application in real-world scenarios where personalization of control paradigms can be time-consuming, costly, and challenging to appropriately define.

We also demonstrate that our approach can be used in an online learning setting. Importantly, we find that the model is able to learn very quickly, from limited amounts of data. In the Online Koopman condition, each trial took an average of 18 seconds, and therefore provided 180 data points. From our analysis in Section 4.5.3.2, we find that we are able to learn an effective model of the joint system after only 5 trials (or about 900 data points). Our model learning technique is also well suited for an online learning paradigm as it is not computationally intensive and can easily run at 50Hz on a Core i7 laptop with 8 GB of RAM. Additionally, we find that, even during the learning process, the application of the online model-based shared control algorithm does not significantly degrade the performance of the human operator.

Finally, we also evaluate the impact that nonlinear modeling and policy generation techniques have on our model-based shared control algorithm [**31**]. In particular, we replace the nonlinear modeling and policy generation techniques with linear counterparts and compare how they impact the ability of a human operator to achieve a desired task. This requires using a nonlinear basis when computing the approximation to the Koopman operator and using nonlinear model predictive control (SAC) to generate the autonomous policy. We find that the nonlinear model-based shared control paradigm produces a joint human-machine system that is significantly better along the primary performance metric (task success) then users under a user only control paradigm. The same result is not found from the data collected under a shared control paradigm that enforced linear constraints (see Figure 4.10).

4.7. Conclusion

In this work, we introduce model-based shared control (MbSC). A particularly important aspect of this work is that *we do not rely on a priori knowledge, or a high-fidelity model, of the system dynamics*. Instead, we learn the system dynamics *and* information about the user interaction with the system directly from data. We learn this model through an approximation to the Koopman operator, an infinite dimensional linear operator that can exactly model non-linear dynamics. By learning the joint system dynamics through user interaction, the robot's understanding of the human is implicit to the system definition.

Results from two human subjects studies (consisting of 32 total participants) demonstrate that incorporating the learned models into our shared control framework statistically improves the performance of the operator along a number of pertinent metrics. Furthermore, an analysis of trajectory ergodicity demonstrates that our shared control framework is able encourage the human-machine system to spend a significantly greater percentage of time in desirable states. We also find that the learned system models are able to be used in shared control systems that generalize across a population of users. Finally, we find that, using this approach, models can be efficiently learned online. In conclusion, we believe that our approach is an effective step towards shared control of human-machine systems with unknown dynamics. This framework is sufficiently general that it could be applied to any robotic system with a human in the loop. Additionally, we have made our code available online at `https://github.com/asbroad/model_based_shared_control` + `https://github.com/asbroad/koopman_` `operator_model_learning`, and include a video depicting a user's control of the dynamic system and the impact of model-based shared control in the supplementary material.

CHAPTER 5

Operation and Imitation under Safety-Aware Shared Control

The following chapter extends our data-driven model-based shared control (MbSC) paradigm to scenarios in which we have no *a priori* knowledge of the user's desired objective. In these situations, the assistance paradigm cannot allocate control based on information related to a desired task, or goal. Instead, we describe a SC paradigm that considers only information related to the safety and stability of the dynamic system. The results of a human-subjects study demonstrate that control-theoretic barrier constraints can be used to increase the general safety of the joint system, without restricting the human partner from achieving unspecified tasks. Additional analysis exhibits how the same shared control paradigm can be used to generate autonomous policies that safely imitate the demonstrations provided by the human partner.

5.1. Introduction

Mechanical devices can be used to extend the abilities of a human operator in many domains, from travel to manufacturing to surgery. In this chapter, we are interested in developing a shared control methodology that further enhances the ability of a human user to operate dynamic systems in scenarios that would otherwise prove challenging due to the complexity of the control problem (e.g., modern aircraft), the complexity of the environment (e.g., navigation in a densely populated area), or the skill of the user (e.g., due to physical injury). A particularly motivating domain is assistive and rehabilitation medicine. Consider, for example, the use of an exoskeleton in rehabilitating the leg muscles of a spinal cord injured subject [**45**]. While these devices are designed explicitly to aid a user in recovering from trauma by rebuilding lost muscular control, the complexity of the machine itself often requires that one or more physical therapists assist the subject in operating the device during therapy (e.g., to provide stabilization). Artificial intelligence can further improve the efficacy of these devices by incorporating autonomy into the control loop to reduce the burden on the human user. That is, if the autonomous agent accounts for subpar (and potentially dangerous) control input, the human operator and therapist(s) are freed to focus on important therapeutic skills.

In this chapter, we improve the effectiveness of joint human-machine systems by developing a safety-aware shared control (SaSC) algorithm that assists a human operator in controlling a dynamic machine without a priori knowledge of the human's desired objective. In general, shared control is a paradigm that can be used to produce human-machine systems that are more capable than either partner on their own [**105**]. However, in practice, shared control systems often require the autonomous agent to know the goal (or a set of discrete, potential goals). While a priori knowledge of a desired set of goals may be a valid assumption in some domains, it can also be a severely limiting assumption in many other scenarios. Therefore, instead of allocating control based on whether the human operator's input will improve the likelihood of achieving a goal, we aim to allocate control based on whether the user's control commands will lead to dangerous states and actions in the future. Under this paradigm, the autonomous partner develops a control strategy that is *only concerned with the safety of the system, and is otherwise indifferent to the control of the human operator.*

Our safety-aware shared control algorithm can be used to improve the efficacy of human-machine systems in two main ways:

G1.: Improve a human operator's control, and understanding, of a dynamic system in complex and potentially unsafe environments.

G2.: Improve the value of Imitation Learning (IL) in the same domains, both by facilitating demonstration and addressing the covariate shift problem [**118**].

Item **G1** is important because control challenges can stem from a variety of issues including the inherent complexity of the system, the required fidelity in the control signal, or the physical limitations of the human partner. For this reason, there is often an explicit need for assistance from an autonomous partner in controlling the mechanical device. Item **G2** is important because Imitation Learning can be used to further extend the capabilities of human-machine systems, however demonstration may not always be feasible for the human partner given the aforementioned control challenges.

The main contribution of this work is a safety-aware shared control algorithm that improves the efficacy of human-machine collaboration in complex environments, *with the key feature that it is possible for the user's desired objective to remain unknown*. In this algorithm, an autonomous partner accounts for system- and environment-based safety concerns, thereby freeing the human operator to focus their mental and physical capacities on achieving high-level goals. Our algorithm (Section 5.3) describes a novel interaction paradigm that extends the viability of complex human-machine systems in various scenarios including those in which the human's skill is limited or impaired. We also provide an analysis of our algorithm (Section 5.4) with a human subjects study consisting of 20 participants conducted in a series of challenging simulated environments. Finally, we show how the same algorithm can be used to improve the human operator's control skill and the power of Imitation Learning (Section 6.7).

5.2. Background and Related Work

The most closely related shared control paradigms in prior literature are those that are safety, or context, aware. In this area, researchers have explored the impact of autonomous obstacle avoidance on teleoperated robots in search and rescue [**127**]. Additionally, safety is a particular concern when the human and robot partner are co-located, such as with autonomous vehicles [**15**]. Co-located partners are also common in assistive robotics where researchers have developed environment-aware smart walkers [**87**] to help visually-impaired people avoid dynamic obstacles.

Related to our goal of generating autonomous policies that recreate the behavior demonstrated during system operation, there is prior work in the field of Imitation Learning (IL). Most commonly, the demonstration data is provided by a human partner [**19**], though it can come from variety of sources including trajectory optimization and simulation [**94**]. Example data is commonly assumed to come from an expert (or optimal) controller [**4**]; however, researchers also have explored techniques that use demonstrations provided by novice (or suboptimal) sources [**154**]. In this work we describe how Imitation Learning can be used even when *the human operator is not able to provide demonstration data on their own*. We further describe how our safety-aware shared control algorithm can be used to address the covariate shift problem [**118**], a common issue in Imitation Learning that stems from the fact that the data used to train the policy may come from a different distribution than the data observed at runtime.

Lastly, there is also related work in the subfield of safety-aware Reinforcement Learning (RL). In this domain, safe autonomous control policies are learned from exploration instead

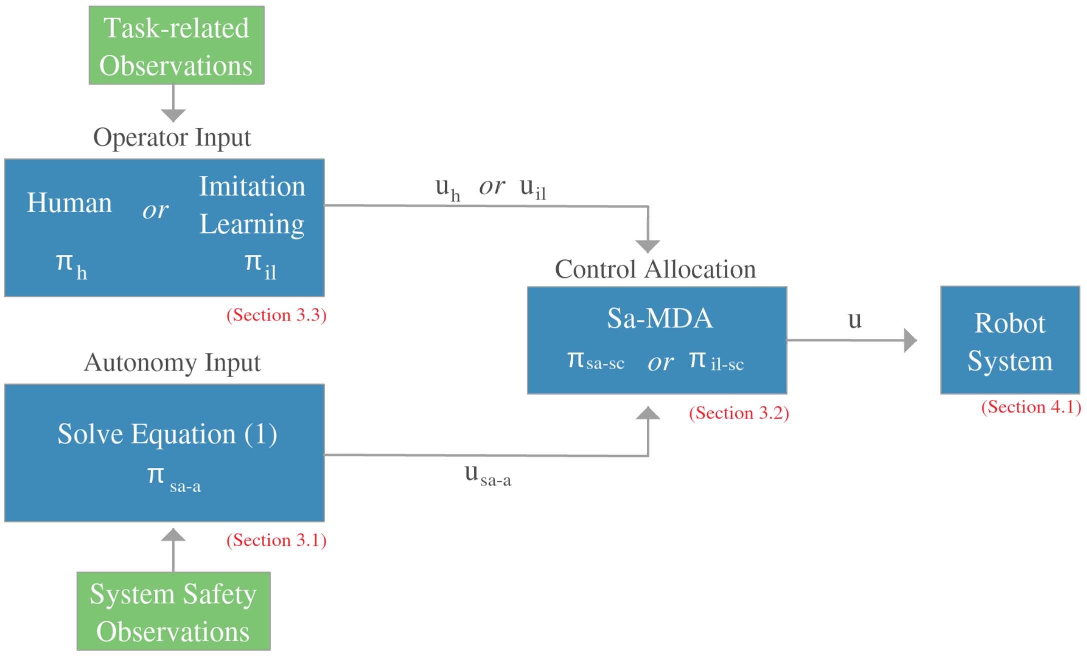


Figure 5.1. Flow chart of our Safety-aware Shared Control (SaSC) algorithm. The operator focuses on task objectives, while the autonomy accounts for safety.

of demonstration. Examples include techniques that enforce safety during the exploration process through linear temporal logic [**9**] and formal methods [**63**]. Researchers also have explored model-free RL as a paradigm to integrate the feedback of a human operator through the reward structure during the learning process, and to share control with a human operator at run-time [**116**]. Finally, researchers have considered learning safe policies from demonstration by computing probabilistic performance bounds that allow an autonomous agent to adaptively sample demonstration data to ensure safety in the learned policy [**35**].

5.3. Safety-Aware Shared Control

In this chapter, we contribute a specific implementation of a class of algorithms that we refer to as Safety-aware Shared Control (SaSC). SaSC can help users operate dynamic systems in challenging and potentially unsafe environments that would normally require expert-level control. An analogous engineering solution is fly-by-wire control of modern aircraft [**130**]. In these systems, the onboard computer accounts for many of the intricacies of the control problem allowing the pilot to focus on high-level tasks. Our SaSC algorithm takes this idea a step further to account for unsafe actions related to both the system dynamics and the environment. Safety-aware shared control consists of two components:

1.: A safe policy generation method for the autonomous agent.

2.: A control allocation algorithm to safely integrate input from both partners.

The high-level algorithm is depicted in Figure 5.1. Points **1** and **2** are described in more detail in subsections 5.3.1 and 5.3.2. Relevant policy notation is below:

- π_h Human operator's policy
- π_{sa-a} Safety-aware autonomous policy (has no task information)
- π_{sa-sc} Safety-aware shared control policy
- π_{il} Imitation Learning policy
- π_{il-sc} Imitation Learning policy under safety-aware shared control

5.3.1. Safety-Aware Autonomous Policy Generation

To implement our safety-aware shared control algorithm, we must first develop an autonomous control policy that is capable of *safely controlling* the dynamic system in question (policy π_{sa-a} in Section 5.3). In this chapter, we utilize Model-based Optimal Control [**20**] (MbOC). MbOC learns a model of the dynamic system directly from data, which is then incorporated into an optimal control algorithm to produce autonomous policies. Here, we learn a model of the system and control dynamics through an approximation to the Koopman operator [**82**]. Further details of this modeling technique are presented in Section 5.4.2.1.

Given a learned model of the system dynamics, we can then compute a control policy by solving the finite-horizon nonlinear Model Predictive Control (MPC) [**115**] problem defined by a cost function of the form

$$J(x(t), u(t)) = \int_{t=0}^{t_f} l(x(t), u(t)) + l_{t_f}(x(t)) \tag{5.1}$$

where

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \tag{5.2}$$

and f defines the nonlinear system dynamics, $x(t)$ and $u(t)$ are the state and control trajectories, and l and l_{t_f} are the running and terminal costs, respectively.

To solve the optimal control problem, we use Sequential Action Control (SAC) [**17**], an algorithm designed to iteratively find a single action (and time to act) that maximally improves performance. Data-driven model-based, and model-free, optimal control algorithms have been experimentally validated with numerous dynamic systems [**7**] including joint human-machine systems [**31**] [**116**].

To address the main focus of this chapter, *we specify a control objective that relates only to system safety*. Safety is defined with respect to *geometric constraints* based on obstacles in the environment. Specifically, we use quadratic costs to deter unstable states and higher order polynomial penalty functions to keep the system away from dangerous locations. Notably, the cost function *does not* incorporate any task information. Therefore, if the autonomous partner's policy is applied directly, the system will work to maintain a safe state but will not move towards any specific goal. The cost function used in our work is described in Section 5.4.2.2.

5.3.2. Safety-Aware Dynamic Control Allocation

To assist the human partner in safely controlling the dynamic system, we define an outer-loop control allocation algorithm that incorporates input signals from the human and autonomous partners (policy π_{sa-sc} in Section 5.3). There is, of course, a balance to strike between the control authority given to the each partner. If the outer-loop controller is too permissive and accepts a significant portion of the human operator's input, it may do a poor job enforcing the necessary safety requirements. However, if the outer-loop controller is too stringent, it can negatively impact the ability of the human operator to produce their desired motion. In this work, we balance the control authority between the human and autonomous partners to increase the authority of the human operator when the system is deemed to be in a safe state, and increase the authority of the autonomy when the system is deemed to be in a dangerous state. Here, the autonomy *adds information into the system only when it is necessary to ensure safety.*

Specifically, we allocate control using a variant of Maxwell's Demon Algorithm (MDA) [**142**]. MDA uses information from an optimal control algorithm as a guide by which to evaluate the input from another source. In this chapter, we contribute a safety-aware variant that we call Safety-Aware MDA (Sa-MDA). Sa-MDA is described in full in Algorithm 2. Here, the **unsafe** function describes whether the system is in an unstable or dangerous configuration with respect to the environment (e.g., through barrier functions, see Section 5.4.2.2), $\boldsymbol{u_h}$ is the input from the human partner, $\boldsymbol{u_a}$ is the input produced by the autonomy, $\langle\cdot\rangle$ is the inner product, and $\boldsymbol{u}$

Algorithm 2 Safety-Aware Maxwell's Demon Algorithm

```
1: if unsafe (system, environment) then
2:     $u = u_{sa-a}$;
3: else
4:     if $\langle u_h, u_{sa-a} \rangle \geq 0$ then    ⊳ When used with an IL policy
5:         $u = u_h$;                                   ⊳ $u_h$ is replaced by $u_{il}$
6:     else
7:         $u = 0$;
8:     end if
9: end if
```

Lines 4–8: Maxwell's Demon Algorithm

is the applied control. When a learned policy is used to mimic a demonstration $\boldsymbol{u_h}$ is replaced with $\boldsymbol{u_{il}}$ (see Section 5.3.3).

5.3.3. Safety-Aware Shared Control Imitation Learning

We now describe how we use the data collected under shared control to produce autonomous policies through Imitation Learning [**19**]. The goal here is to learn a policy π_{il} that mimics the behavior demonstrated by the human operator. To achieve this goal, we treat the data collected under shared control π_{sa-sc} as a supervisor in the policy learning process. Notably, this data, and the associated learned policy, *now contain task-relevant information*, as provided by the human operator during demonstration. Our goal, then, is to learn an autonomous policy that minimizes the following objective

$$J(\pi_{il}, \pi_h) = \min_{\theta} \sum_{s \in \xi \in \mathbb{D}} ||\pi_{il}(s) - \pi_h(s)||_2^2 \tag{5.3}$$

where J is the cost to be minimized and s is the state. By minimizing J we learn a policy that closely matches the policy demonstrated by the human partner. $\pi(s) : s \to u$ defines a control policy which is parameterized by θ, π_h represents the (supervisor) human's policy and π_{il} represents the Imitation Learning policy.

The autonomous policy is learned from a set $\mathbb{D}$ of trajectory data ξ recorded during the demonstration phase. To generate π_{il}, we use behavior cloning [**88**], a standard offline imitation learning algorithm. Details of our specific implementation are provided in Section 5.4.2. As described in Section 6.2, behavior cloning can fail to reproduce the desired behavior due to the covariate shift problem [**118**] [**88**]. We address this issue with an autonomous policy π_{il-sc} that *combines* the learned policy π_{il} with the safety-aware autonomous policy π_{sa-a} (Algorithm 2). By incorporating the same shared control algorithm used during data collection, we encourage the system to operate in a similar distribution of the state space to what was observed during demonstration. One can view this solution as a shared control paradigm in which the control is shared between *two autonomous agents*: the autonomy mimicking the human control and the autonomy enforcing safety constraints.

5.4. Empirical Evaluation

To evaluate the efficacy of our algorithm, we perform a human subjects study on a simulated system exhibiting nonlinear dynamics in complex environments. In this evaluation, we compare the efficacy of each policy presented in Section 5.3 except π_{sa-a}, which is never executed and is only used to keep π_h and π_{il} safe.

5.4.1. Experimental System

The experimental system consists of a simulated "lunar lander" (Figure 5.2), chosen to demonstrate the impact that shared control can have on the safety of a joint human-machine system when the control problem and environment are complex. The lunar lander exhibits non-linear dynamics and can easily become unstable as it rotates away from its point of equilibrium.

Again, in this experiment *safety is defined with respect to hard geometric constraints.* Additionally, two of the experimental environments contain obstacles that must be avoided to stay safe (see Figure 5.2). The system and environment are implemented in the Box2D physics engine based on the environment defined in OpenAI's Gym [**34**].

The lunar lander is defined by a six dimensional vector which includes the 2D position and heading (x_{1-3}) and their rates of change (x_{4-6}). The control input is a continuous two dimensional vector which represents the throttle of the main (u_1) and rotational (u_2) thrusters. The *first* environment includes only the lunar lander and the ground surface (Fig. 5.2, left). This environment illuminates the challenges associated with maintaining the stability of a complex dynamic system, while simultaneously executing unspecified behaviors. The *second* environment incorporates dynamic obstacles that obstruct the motion of the system (Fig. 5.2, middle). In this environment, a series of circular obstacles move across the screen at the same height as the lander (one at a time). The *third* environment includes two static obstacles that force the operator to navigate through a narrow passageway, increasing the required control fidelity (Fig. 5.2, right).

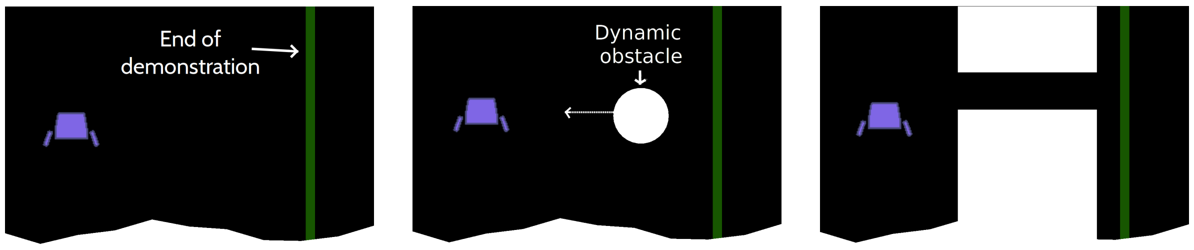


Figure 5.2. Visualization of lunar lander (enlarged) and experimental environments. A trial is complete when the lander moves across the green line boundary.

5.4.2. Implementation Details

5.4.2.1. Model Learning. We learn a model of the system and control dynamics using an approximation to the Koopman operator [**150**], which has been validated on numerous systems [**7**], including human-machine systems [**31**]. The Koopman is a linear operator that can model all relevant features of nonlinear dynamical systems by operating on a nominally infinite dimensional representation of the state [**82**]. To approximate the true Koopman operator one must define a basis. In this chapter, we define $\phi = [1, x_1, x_2, x_3, x_4, x_5, x_6, u_1, u_2, u_1 \cdot x_1, u_1 \cdot x_2, u_1 \cdot x_3, u_1 \cdot x_4, u_1 \cdot x_5, u_1 \cdot x_6, u_2 \cdot x_1, u_2 \cdot x_2, u_2 \cdot x_3, u_2 \cdot x_4, u_2 \cdot x_5, u_2 \cdot x_6, u_1 \cdot cos(x_3), u_1 \cdot sin(x_3), u_2 \cdot cos(x_3), u_2 \cdot sin(x_3)]$, where x_{1-6} represent the system state variables and u_{1-2} represent the control input variables. The specific basis elements used in this (and the previous) chapter are chosen empirically and represent a reduced set of features that describe the full state and control space, as well as the interaction between the user's input and the state. Data-driven methods (e.g. sparsity-promoting DMD [**75**]) can be used to automatically choose a proper set of basis functions (see Chapter 6).

5.4.2.2. Safety-Aware Autonomous Policy Generation. To compute an autonomous policy that is solely concerned with the safety of the system, we define a cost function (Equation (5.1)) that considers two notions of safety: stability around points of equilibrium and collision avoidance,

$$l(x) =$$

$$\overbrace{Diag[0, 0, 15 \cdot x_3, 1 \cdot x_4, 1 \cdot x_5, 10 \cdot x_6]^2}^{\text{stabilization}} + \overbrace{Diag[(x_1 - o_1), (x_2 - o_2), 0, 0, 0, 0]^8}^{\text{obstacle avoidance}}$$

where (o_1, o_2) is the position of the nearest obstacle in 2D space. This cost function (i) penalizes states that are far from points of equilibrium using a quadratic cost and (ii) prevents the system from entering dangerous portions of the state space using polynomial barrier functions [**25**]. The stabilization term ensures that the lunar lander does not rotate too far away from upright—if this happens, the lander's main thruster can no longer be used to counteract gravity, a situation that commonly leads to catastrophic failure in rockets. We therefore penalize both the position and velocity of the heading term. We additionally penalize the x and y velocity terms as momentum can significantly reduce the time a controller has to avoid collision. Finally, the obstacle avoidance term simply acts to repel the system from the nearest obstacle. Importantly, if this policy is applied on its own (without any input from the human partner), the lander will simply attempt to hover in a safe region, and will not advance towards any goal state. In our implementation, the obstacle avoidance term does not include any penalties based on the velocity (or momentum) of the system and therefore cannot guarantee safety. However, in future work, we plan to define barrier functions that provide strong safety guarantee [**93**].

Notably, the safety-aware autonomous policy only *adds* information into the control loop when the system is deemed to be **unsafe** (see Algorithm 2). Otherwise the user's commands are, at most, simply blocked by the autonomy. This can be thought of as an accept-reject-replace shared control paradigm [**78**]. We define the **unsafe** function based on an empirically chosen distance to the nearest obstacle. Therefore, if the system gets too close to an obstacle, the autonomy's signal is sent to the system, otherwise the human's input is accepted or rejected, according to the MDA filter. Here we note that the structure and weights in the defined cost function, as well as the pre-defined distance metric, are specific to the experimental system; however, there are generalizable principles that can be used to develop similar cost function for other systems. For example, system stability can generally be improved by defining costs that help reduce dynamic features to kinematic features, while obstacle avoidance terms can be defined using additional information from the learned system model.

5.4.2.3. Imitation Learning. A neural network is used to learn a control policy that mimics successful trials demonstrated by the human partner. The input is the current state (x_{1-6}) and the output is the control signal (u_{1-2}) sent to the system. The control signal is discretized (-1.0 to 1.0 in increments of 0.5) and the problem is therefore cast as a classification instead of regression. There are three hidden layers in the neural network—the first has 32 nodes, and the following two layers have 64 nodes. Each hidden layer uses ReLu as an activation function. The final layer uses a softmax activation. We use categorical cross entropy to compute the loss and RMSProp as the optimizer.

5.4.3. Study Protocol

The human subjects study consisted of 20 participants (16 female, 4 male). All subjects gave their informed consent and the experiment was approved by Northwestern University's Institutional Review Board. Each participant provided demonstrations of novel behaviors in all three of the environments, under both a *user-only control* paradigm and our *safety-aware shared control* paradigm. There was no goal location specified to the participants; instead a trial was considered complete when the human operator navigated the lunar lander across a barrier defined by the green line in the environment (see Figure 5.2). The specific trajectory taken by the lander during a demonstration was up to the participant. The operator used a PS3 controller to interact with the system. The joystick controlled by the participant's dominant hand fired the main thruster, and the opposing joystick fired the side thrusters.

Subjects were asked to provide 10 demonstrations per environment (3 total) and per control paradigm (2 total), resulting in a total of 60 demonstrations per participant. The environments were presented in a randomized and counterbalanced order. Participants were assigned to one of two groups, where Group A (10 subjects) provided demonstration data in each environment under *user-only control* first, and Group B (10 subjects) provided demonstration data under *shared control* first. Group assignment was random and balanced across subjects.

5.5. Experimental Results

We find that our SaSC algorithm significantly improves the user's skill with respect to a no-assistance baseline (Figure 5.3). Additionally, we find that our SaSC algorithm can be used as a training mechanism to improve a subject's understanding and control of the dynamic system (Table 5.2). Finally, we show how the same shared control technique can be used to extend the Imitation Learning paradigm (Figure 5.4 and 5.5). These findings are discussed in detail in the following subsections.

5.5.1. Safety-Aware Shared Control Enables Successful Demonstration

To address item **G1**, the primary metric we evaluate is a binary indicator of control competency: the occurrence of safe, successful demonstrations, indicated by navigating the lunar lander beyond the green border. We first analyze data collected from all three experimental environments together. We then segment the data based on the specific environment in which it was collected and re-perform our analysis (Figure 5.3). We use the non-parametric Wilcoxon signed-rank test to statistically analyze the data.

The results of the statistical tests revealed that our described shared control paradigm significantly improved the human partner's control skill ($p < 0.005$). In particular, we find that participants provided safe demonstrations of the desired behavior in 96.0% of the trials produced under the shared control paradigm versus 38.5% of the trials produced under the user-only control paradigm. Additionally, the statistically significant result holds when we compare the control paradigms in each experimental environment separately ($p < 0.005$ in all cases).

Recall that the shared control paradigm does not provide any *task*-related assistance, but rather only *safety*-related assistance. We therefore interpret the increase in task success as evidence that our SaSC algorithm helps subjects exhibit greater control skill, and an associated increased ability to provide demonstrations of novel behaviors. Additionally, we note that in some experiments conducted under user-only control, subjects were *not able to provide any successful demonstrations* in a given environment. This suggests that safety-aware shared control

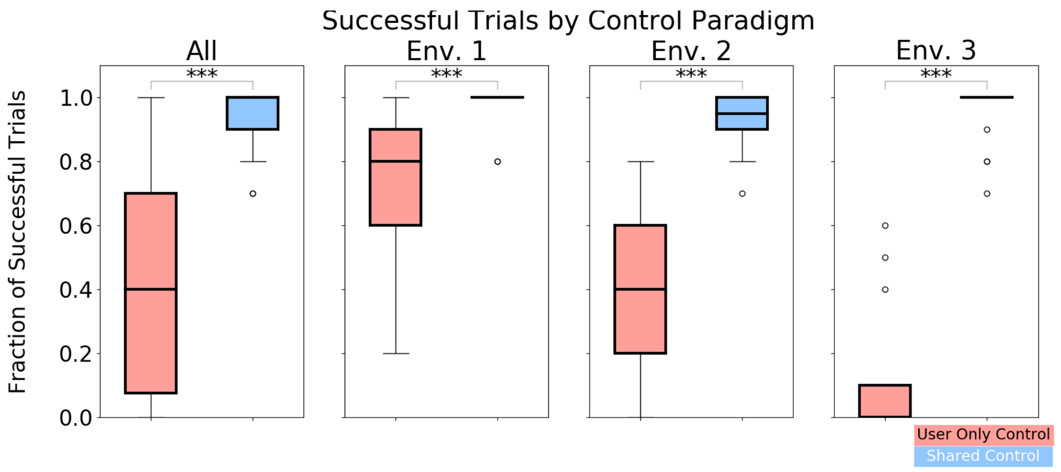


Figure 5.3. Average fraction of successful trials under each control paradigm in each environment. The plots represent data collected in all environments (left), and broken down by each individual environment (right). In all cases, participants under the shared control paradigm provide safe demonstrations significantly more often than under the user-only control paradigm (*** : $p < 0.005$).

may be a *requirement* for functional usage of dynamic systems in some of the more challenging domains that motivate this work. This finding is also important when considering our ability to train autonomous policies that mimic behaviors demonstrated by the human operator (see Section 5.5.4).

5.5.2. Impact of Safety-Aware Shared Control on Trajectory Features

As discussed in Section 5.3.2, safety-aware shared control impacts features of the trajectories produced by the human operator. For example, by rejecting a majority of the user's inputs the SaSC algorithm can ensure system safety, but it will not allow the user to execute desired behaviors. To evaluate how our SaSC algorithm impacts the user's abilities to demonstrate a novel behavior, we compare a number of quantitative metrics that go beyond safety and relate specifically to features of the trajectories (Table 5.1). Here, we analyze only the *successful demonstrations* provided under each control paradigm.

Metric	Control	Env 1.	Env 2.	Env 3
Path Length (m)	User	30.0 ±2.4	33.8 ±4.8	28.3 ±0.8
	Shared	27.7 ±1.1	34.4 ±2.3	30.5 ±2.2
Trial Time (s)	User	15.5 ±3.5	18.6 ±5.6	22.4 ±5.9
	Shared	27.0 ±7.8	30.0 ±7.4	30.4 ±6.8
Final Speed (m/s)	User	23.6 ±6.8	23.8 ±6.2	14.5 ±5.0
	Shared	7.9 ±2.8	9.6 ±4.5	7.9 ±1.9
Final Heading (deg)	User	45.5 ±73.5	79.5 ±79.8	39.6 ±46.0
	Shared	2.7 ±10.4	6.4 ±22.0	3.5 ±5.9

Table 5.1. Mean and standard deviation of trajectory metrics computed from successful demonstrations. Path length is not impacted significantly by SaSC, but shared control does result in trajectories that take longer to execute, have slower final speeds, and are more upright (stable) in their final configurations.

Our analysis shows that the safety-aware shared control paradigm impacts not only the *ability* of a user to provide demonstrations, but also *how* they provide demonstrations. In all environments, we find that participants produced trajectories of nearly equal length under both control paradigms. However, under the user-only control paradigm participants produced successful demonstrations in *less time*, with a *greater final speed* and in a state that is *rotated further away from the point of equilibrium* than under the shared control paradigm. Moreover, in Environments 1 and 2 (the less constrained environments), participants were able to produce demonstrations that were safe over the course of the demonstrated trajectory, *but unstable in the final configuration*. While these were counted as successful demonstrations, they require less control skill than trajectories that end in a stable configuration. This suggests that SaSC improves control skill in ways not fully captured by the binary success metric.

5.5.3. Safety-Aware Shared Control as a Training Mechanism

As a final piece of analysis addressing item **G1**, we examine whether experience under a safety-aware shared control algorithm improves human skill learning. To evaluate this idea, we compare the user-only control trials of Group A (*user-only condition first*) with those of Group B (*user-only control condition second*). We segment the data based on the specific environment in which it was collected and use the non-parametric Mann-Whitney U test to statistically analyze the results. We display data and the results of the described statistical tests in Table 5.2.

The important take-away from these results is that shared control allows users *to operate human-machine systems safely during their own skill learning*, and that this practice then *translates to skill retention when the assistance is removed*. In the most challenging environment,

Environment 3, we see the largest raw difference ($+18\%$) in success rate between the two cohorts and a statistically significant result ($p < 0.005$). Notably, *zero* participants provided *any* successful demonstrations in this environment when the data was provided under the user-only control paradigm first. We also see a relatively large, but not statistically significant, difference in raw percentage points in Environments 1 and 2 ($+10\%$ and $+9\%$) which might become statistically significant with more data.

Under this paradigm, we can allow users to learn naturally while simultaneously ensuring the safety of both partners. For systems where failure during learning is not acceptable (e.g., exoskeleton balancing), safety-aware shared control becomes a requirement in enabling human skill acquisition: without some sort of safety assistance, operators are simply not able to control the system.

5.5.4. Safety-Aware Shared Control Improves Imitation Learning

To address item **G2**, we examine whether learned policies are capable of reproducing the behavior demonstrated by the human operator. Our evaluation is based on a comparison of trajectories generated by Imitation Learning with (π_{il-sc}) and without (π_{il}) safety-aware shared control at runtime. We provide visualizations of 10 successful reproductions of the demonstrated behaviors in Environments 2 and 3 in Figures 5.4 and 5.5, respectively.

	User-Only First	**User-Only Second**	**Difference**	**Stat. Significance**
Env 1.	67 %	77 %	+10 %	$p > 0.05$
Env 2.	35 %	44 %	+9 %	$p > 0.05$
Env 3.	0 %	18 %	+18 %	$p < 0.005$
All	34 %	46 %	+12 %	$p = 0.06$

Table 5.2. Average success rate under *user-only* control over time.

Notably, *all* trajectories produced by π_{il-sc} safely avoid both the static and dynamic obstacles. In Figure 5.4 we see that the learned policy π_{il-sc} is able to mimic the behavior demonstrated by the human operator. In Figure 5.5, we include visualizations of the trajectories provided under user-only control (π_h) and Imitation Learning without shared control (π_{il}) in Environment 3. Here, we also see that the user was unable to provide any successful demonstrations without safety assistance. Similarly, the learned control policy (π_{il}) was unable to avoid obstacles in the environment without the safety assistance.

These two final points elucidate the *need for our safety-aware shared control system in both the demonstration **and** imitation phases.* Without assistance from the SaSC algorithm, not only is the human operator unable to demonstrate desired behaviors, but the learned neural network policy fails to generalize.

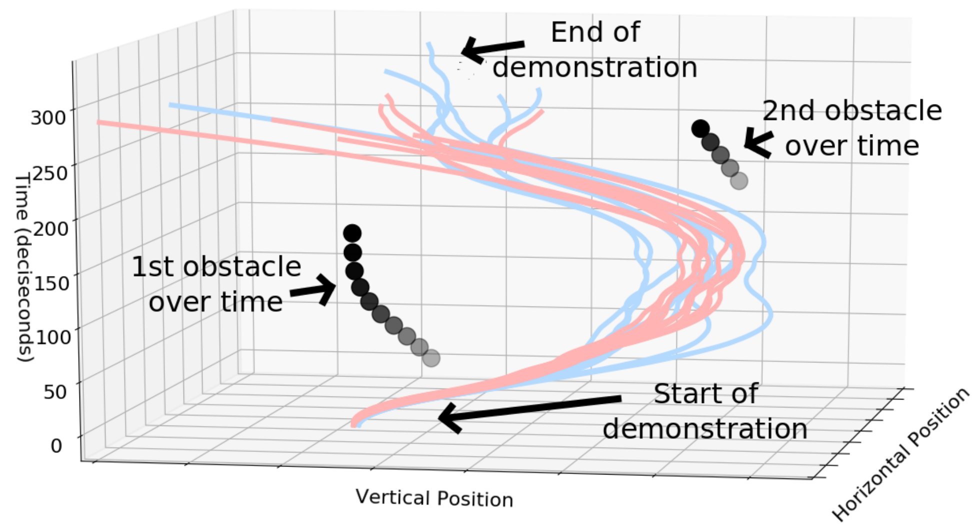


Figure 5.4. Environment 2. A visualization of data provided by the human partner under safety-aware shared control (π_{sa-sc} : blue) and trajectories produced autonomously (π_{il-sc} : pink) using the learned control policy. The vertical and horizontal position of the dynamic obstacles (black) over time are also displayed.

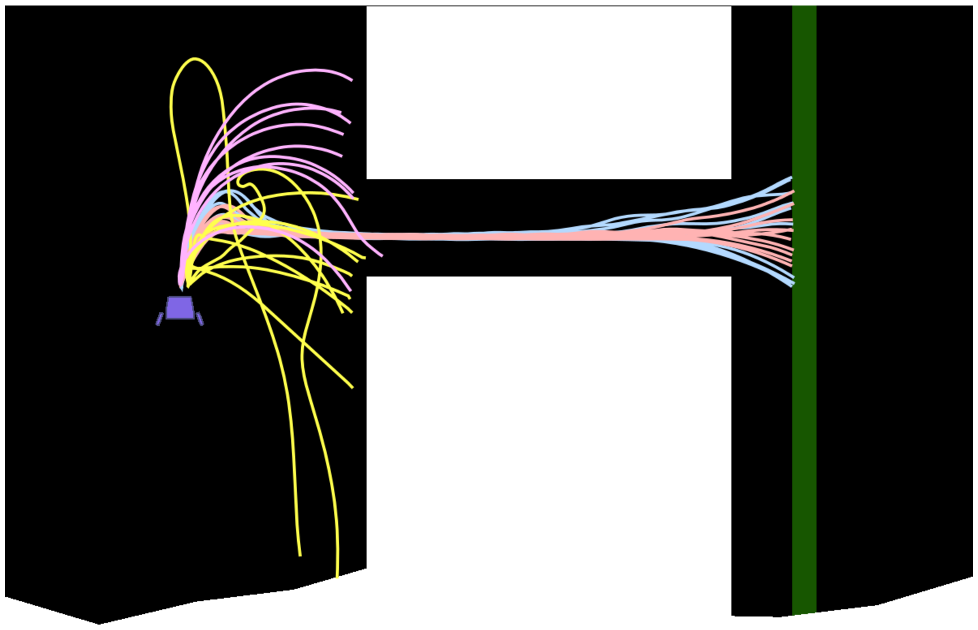


Figure 5.5. Environment 3. Visualization of (1) demonstrations provided under the user-only control paradigm (π_h : yellow), (2) demonstrations provided under our safety-aware shared control paradigm (π_{sa-sc} : blue), (3) trajectories produced autonomously using solely the Imitation Learning policy (π_{il} : purple, learned from π_{sa-sc} demonstrations), and (4) trajectories produced autonomously using the safety aware shared control imitation learning policy (π_{il-sc} : pink).

5.6. Discussion and Conclusion

In this chapter, we contribute a shared control paradigm that allows users to provide demonstrations of desired actions in scenarios that would otherwise prove too difficult due to the complexity of the control problem, the complexity of the environment, or the skill of the user. We solve this problem through the application of shared control, allowing a human operator to provide demonstrations while an autonomous partner ensures the safety of the system. We validate our approach with a human subjects study comprised of 20 participants.

The results of our human subjects study show that our safety-aware shared control paradigm is able to help human partners provide demonstrations of novel behaviors in situations in which they would otherwise not be able (Figure 5.3). Additionally, we find that our SaSC algorithm can be used as a training mechanism to improve a human operator's control skill by ensuring safety during training (Table 5.2). Furthermore, we find that a combination of Imitation Learning with our safety-aware shared control paradigm produces autonomous policies that are capable of safely reproducing the demonstrated behaviors. In this work, the autonomous policies are learned from data provided under shared control and for this reason, one must also consider how the autonomy affects the demonstration data. We find that the shared control paradigm slows the average speed of the system, but generally increases the stability (Table 5.1). Of course, it is also possible to learn autonomous policies from the data provide under user-only control. However, when system safety is a requirement (e.g. co-located human-machine systems), shared control can be thought of as fundamental for allowing users to provide demonstrations of new behaviors. Allowing a system to fail during demonstration is often an unrealistic assumption with real-world systems.

In future work, we plan to explore additional uses of data collected under a shared control paradigm in learning autonomous policies that do not rely on continued safety assistance (e.g. as seeds in Guided Policy Search [**94**]). Relatedly, there is evidence to suggest that policies learned via Inverse Reinforcement Learning (IRL) may provide more robust autonomous policies [**9, 35**] than standard Behavior Cloning techniques, as described in this work. While full IRL solutions likely require more data then our safety-aware imitation learning algorithm, the improved robustness may be worth the training effort, and therefore remains an area of future research. Finally, we also plan to explore a bootstrapped notion of shared control, in which the outer-loop autonomous controller originally considers only the safety of the joint system and then dynamically updates to consider both the safety of the system system and task-level metrics that describe the desired behavior of the human operator.

CHAPTER 6

Highly Parallelized Data-driven MPC for Minimal Intervention Shared Control

The following chapter builds on the general task-agnostic, data-driven model-based shared control paradigm introduced in the previous chapter. Instead of relying on pre-specified control barrier functions to improve safety, this chapter uses the learned system dynamics model to *predict safety* over a receding horizon. The algorithm described in this chapter also introduces an additional optimization procedure into the control loop to explicitly maximize the authority granted to the human-in-the-loop. A human-subjects study evaluated with two different simulated environments demonstrates that the described SC system improves the safety of the joint system, while simultaneously adhering to the instantaneous desires of the human operator. A post-study questionnaire revealed that our minimal intervention shared control paradigm was preferred to a user-only control paradigm and produced very low levels of frustration, suggesting that the described paradigm is a promising research direction for human-oriented shared control. The end of this chapter highlights the results of a preliminary study that aims to integrate data-driven MbSC with the trust-based personalization described in Chapter 3.

6.1. Introduction

Shared control is a paradigm that incorporates an autonomous partner into the control loop of a robotic system to help a human partner achieve tasks they would otherwise be unable to on their own [**5**]. This approach offers an alternative to fully autonomous robotic systems, and can be used to extend the efficacy of modern robots in human-oriented domains (e.g., surgery, assistance and rehabilitation, search and rescue) through collaboration. In these domains the two partners often need to communicate frequently and may even be co-located. The most important consideration of the autonomous partner is therefore the safety of the joint system, as unsafe behavior can lead to injury to the human operator. However, there are other features that may be equally important when we consider the range of behaviors the human partner may wish to perform, and the user's acceptance of the assistance provided the autonomous partner.

In the majority of related work, the decision of which partner should be in control at a given time is made based on (1) the safety of the system and (2) an evaluation of who would provide better input to achieve a perceived, or known, task goal [**105**]. These task-specific systems can frustrate the human partner when the user's intended goal is difficult to predict, and they fail out-right when there is no desired trajectory, task or goal the user is trying to achieve. Consider, for example, a person operating a lower-limb exoskeleton for rehabilitation purposes. In this scenario, task-level and performance-based metrics (e.g., the amount of area covered during a search mission), are not important in determining which partner should be in control at a given time, as there is often no explicit notion of a goal (i.e., the human operator may simply want to wander around aimlessly). Instead, the person's instantaneous desires are often the most relevant feature, conditioned on the general safety of the system.

The question we consider in this chapter then is, how does one define safety constraints that can enhance a user's ability to operate complex, dynamic machines without artificially constraining their capacity to achieve unspecified behaviors? We address this gap in the literature with a task-agnostic control allocation strategy that balances the human's desires to achieve a wide range of possible behaviors, while simultaneously improving the safety of the joint system. Our shared control paradigm therefore adheres to the following three ideals:

(1) safety is paramount,

(2) the user has no explicit task goal, and

(3) the autonomy should exert as little influence as possible.

In other words, the goal of our shared control paradigm is to allow the user to do whatever they would like, so long as the safety of the joint system is satisfied. Additionally, when the autonomous partner does intervene, it should only minimally modify the user's input (a.k.a., the minimal intervention principal [**13**]). By adhering to these ideals, we hope to increase the influence of the human partner and consequently improve their acceptance of the assistance provided by the autonomy.

In this chapter, we develop a shared control algorithm that uses a highly parallelizable sampling-based model predictive control (MPC) algorithm to generate the autonomous partner's policy. By sampling densely at uniform over the input space, we can evaluate a large set of potential actions that the user may wish to take, without *a priori* knowledge of a specific goal. We then use ideas from model-based reinforcement learning and model predictive control to generate predicted trajectories (or imagined rollouts) that represent the configuration (and safety) of the robot over a receding horizon. Conditioned on the predicted safety of the robotic system, we iteratively select the sampled action that most closely matches the human partner's input, allowing the user to more safely move around the environment without adhering to a single objective. Our approach relies on a representation of the human-machine system that is valid both with, and without, a known analytical model. We focus on the latter case and therefore learn a model of the joint system offline from data. We evaluate the efficacy of our approach with a human subjects study in two simulated environments. Additionally, we provide an open-source, scalable implementation of our algorithm in both environments that uses a GPU for real-time interaction.

The main contributions of this chapter are therefore:

- A highly parallelizable sampling-based model predictive control algorithm for autonomous policy generation.
- A predictive notion of safety that can evaluate the impact of a current action over a receding horizon.
- A human-motivated cost function that only considers the instantaneous desires of the human-in-the-loop and requires no knowledge of an explicit goal.
- The results of a human subjects study that evaluates the efficacy of, and user experience with, our shared control paradigm.
- A GPU-implementation for real-time control of two simulated systems.

In Section 6.2 we provide background information and related work. In Section 6.3 we detail the theoretically exact solution to our problem. In Section 6.4 we describe our approximation, why it scales well to the majority of devices in our target domain (human-centered robotics), and provide analytical bounds on the sub-optimality of the applied control with respect to the human's desired motion. In Section 6.6 we describe the experiment we use to validate our shared control paradigm through a human subjects study consisting of 20 participants in two simulated environments. We also detail pertinent metrics to analyze the human partner's control skill and style, which are easily computable due to our sampling-based approach. In Section 6.7 we present the study results which we discuss in Section 6.8. Finally, we describe the results of a second, preliminary study that integrates the data-driven trust paradigm detailed in Chapter 3 with our model-based shared control algorithm in Section 6.9. We then conclude in Section 6.10.

6.2. Background and Related Work

In this section we discuss background and related literature in both shared and fully autonomous control, with a particular focus on human-oriented domains.

6.2.1. Shared Control

The majority of the shared control literature focuses on assisting a human operator when a desired task is known *a priori* [**80, 40**] or predicted based on a model of the operator's intent [**50**]. For this reason, task success is often the primary metric of concern in analyzing shared control systems, while the user's desires relating to *how* a motion is achieved are often disregarded. In some application domains there is a welcome trade-off between achieving the desired high-level goal and intervention from the autonomous partner (e.g., with ground vehicles on a roadway where a large amount of structure is enforced on the motion of the dynamic system). In more human-centered domains, such as assistive and rehabilitation robotics, there is often significantly less structure imposed on the motion of the machine and there may be no explicit goal. For this reason, the same trade-off in task success and autonomous intervention is not consistently accepted by users [**54**]. In these domains, it is instead of utmost importance that the user retains a sense of personal agency and a feeling of control over the mechanical device.

Despite these differences, the most closely related work to our own (from a methodological standpoint) can be found in the semi-autonomous vehicle literature. The semi-autonomous vehicle paradigm is distinct from the concept of a fully autonomous self-driving cars as the autonomy's goal is not to take full control of the vehicle, but instead to act as a guardian or intelligent co-pilot [**14**], intervening on the person's control when deemed necessary to ensure safety. For this reason, there is a growing line of work that follows the minimum intervention principle in the semi-autonomous vehicle domain [**93**]. For example, Schwarting et al. [**125, 126**] describe a parallel autonomy framework that develops control trajectories for semi-autonomous vehicles that minimize deviation from user-input and achieve task-specific metrics like road following and contour tracking. Anderson et al. [**13, 16**] describe a geometric, homotopy-based algorithm for computing *free space* in the environment. The human operator is then allocated full control of the dynamic system so long as their input will not violate the constraints defined by the geometric constructs. In this work we provide an alternative method (prediction) of computing safe space in the environment that is simpler (e.g., there is no need to integrate constraints defined by potentially complex geometries into the optimization problem), and acts over the entire control space (i.e., instead of only the steering angle [**13, 40, 55**]). Additionally, all prior work in this area assumes *a priori* knowledge of the system dynamics whereas our technique extends to models learned from data.

6.2.2. Sampling-based and Stochastic Optimal Control

From a control-theoretic standpoint, related work includes shared control algorithms that build on ideas from sampling-based optimal control. For example, Carlson et al. have explored the idea of controlling a wheelchair using *safe mini-trajectories* [**40, 39**], however, this work again relies on *a priori* knowledge of the human operator's goal (or predicted goals based on sensor information). Relatedly, Shia et al. [**128**] use a probabilistic model of the user's potential future inputs to achieve a pre-specified task, instead of allowing the operator the freedom to move however they would like at each instant.

Our approach is also related to fully autonomous control solutions that rely on sampling-based and stochastic optimal control methods. For example, Lavalle et al. propose Rapidly-expanding Random Trees (RRTs) [**90**], which develop random trajectories through the state space that are achievable as they are constrained by the system dynamics. More recently, Kousik et al. extend this idea through a model predictive control algorithm that is based on the notion of a forward reachability set [**85, 86**] to ensure safety. Finally, Williams et al. have proposed Model Predictive Path Integral (MPPI) [**135, 147**] control as a method of solving the optimization problem through path integrals. These ideas build on similar theory to our own (approximating an optimal solution from a nominally infinite set of trajectories), however, they are again all standardly defined in relation to a specified start and goal configuration. We instead consider an approximation to the infinite set of trajectories that stem from a single point and extend in *all* directions for a given time-horizon.

6.3. Safe Minimal Intervention Shared Control

In this section, we describe a theoretically correct (though computationally infeasible) solution to the problem of safe minimal intervention shared control. We begin by defining the shared control problem mathematically. This problem can be posed in standard optimal control terms with an additional constraint to incorporate information from both partners. That is, our goal is to find an optimal action sequence $\bar{\mathbf{u}}$, and corresponding state sequence $\bar{\mathbf{x}}$, that minimizes the

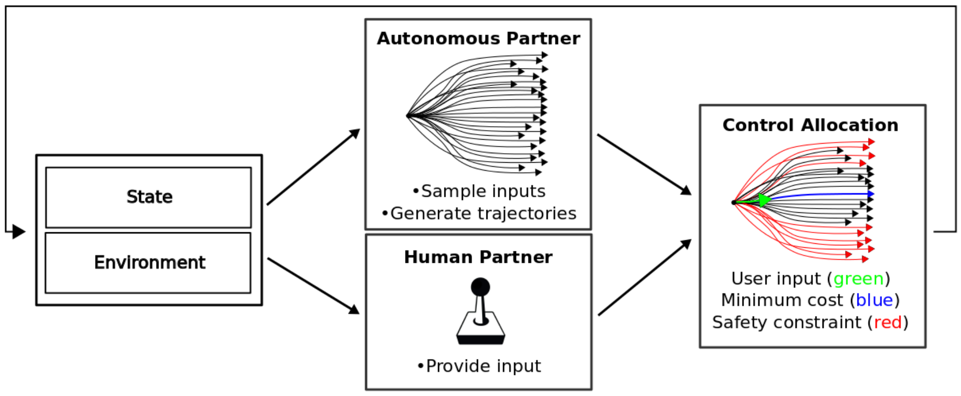


Figure 6.1. Pictorial representation of our model predictive minimal intervention shared control (MPMI-SC) paradigm. The autonomous partner samples densely from the input space and generates a cloud of potential trajectories using a massively parallel processor. The human partner provides their desired input. The optimal control solution is then computed according to the Minimal Intervention Principle (MIP) and constrained based on the safety of the system over a receding horizon.

cost

$$\begin{aligned} \underset{J}{\text{minimize}} \quad & J(x(t),u(t)) = \int_{t=0}^{T} l(x(t),u(t)) + l_T(x(t)) \\ \text{subject to} \quad & \dot{x}(t) = f(x(t),u(t)), \\ & u(t) = g(u_h(t),u_a(t)) \end{aligned} \tag{6.1}$$

where $x(t)$ and $u(t)$ are the state and control trajectories, and l and l_T are the running and terminal costs. The optimization is subject to constraints f representing the nonlinear system dynamics, and g representing the control allocation between the human (u_h) and autonomous partners (u_a).

Algorithm 3 Minimal Intervention Shared Control

```
1: procedure MI-SC($x_t, u_h$)
2:     $\Gamma \leftarrow$ all possible trajectories ($\gamma$) from $x(t)$
3:     $\Gamma_{safe} \leftarrow \gamma \in \Gamma$ where $\gamma \notin ICS$
4:     $u_r \leftarrow \arg\min(u_h, cost(\Gamma_{safe}))$
5:     return $u_r$
6: end procedure
```

6.3.1. Safe Control and Inevitable Collision States

The primary constraint in g relates to the safety of the human-machine system. That is, the autonomous partner should only produce trajectories that remain safe over a receding horizon. This requires ensuring that the system does not enter an Inevitable Collision State (ICS) [**23, 62**]. ICSs refer to configurations from which it is impossible to safely recover, regardless of the control trajectory taken. So long as the system does not enter an ICS, it is possible to develop a control strategy that results in continued safe interaction.

6.3.2. Minimum Intervention Principle

The second feature we embed in g is known as the minimal intervention principle (MIP) which states that “an autonomous partner should only augment the human partner’s control by the minimum amount necessary to achieve the desired result” [**13, 93, 125**]. By developing a shared control algorithm that adheres to the MIP we maximize the influence of the human partner’s control at each given moment.

6.3.3. Minimal Intervention Shared Control

The described solution to minimal intervention shared control is outlined in Algorithm 3. Here Γ is the set of all possible trajectories γ that stem from the current state $x(t)$, and Γ_{safe}

is the subset of safe trajectories. The inputs u_r and u_h are the signal sent to the robot, and the signal provide by the human partner, respectively. The $cost$ function describes how desirable a trajectory is based on it's distance from the human operator's input. If the human's command does not lead to an ICS, their exact input will be passed to the system; otherwise, a perturbation (computed as the minimum deviation required to ensure safety) will be applied to the control signal.

This exact (full information) algorithm is computationally infeasible to compute online for any reasonably complex human-in-the-loop system. In particular, the solution requires exploring an infinite set of potential actions over a receding horizon to (1) ensure collision-free trajectories [**144**], and (2) select the input that most closely matches the human's desired action. In this work, we instead propose an approximation to this solution that can be computed in real-time.

6.4. Model Predictive Minimal Intervention Shared Control

To compute an approximation to the optimal solution described in the prior section, we make a few key methodological choices. First, instead of considering *all* possible trajectories from the current sate, we only consider a representative set that we generate by sampling densely (from a uniform prior) over the input space and only at the present time. We then predict the motion of the system over a receding horizon and reject any inputs that generate a trajectory that violate the defined safety constraints. From the subset of inputs that do not produce unsafe trajectories, we select the control signal that is closest to the human partner's input. Notably, this approach does not require a model of the user's actions, or knowledge of a desired goal, as the user is free to dynamically adjust their objective at each timestep. We describe each step of this algorithm in detail in the following subsections.

6.4.1. Model Representation and Data-driven Approximations

The majority of related work requires hand-written (and potentially complex) models of the system and control dynamics [**13, 93, 125**]. In this work, we instead use a representation that is simple (i.e., linear) regardless of the underlying dynamics and can be learned from data when the model is not known *a priori*. This representation is known as the Koopman operator [**82**] and it can be used to model nonlinear dynamical systems as a linear operator because it *maps functions of state to functions of state* instead operating in the original state space. This representation is valid both when we know an analytical model of the system in the standard state space [**82**], and when we must learn the model from data [**150**].

In this work, we opt to learn the system model from data to demonstrate the efficacy of our approach when we have no prior knowledge of the robotic system. In particular, we use sparsity-promoting Dynamic Mode Decomposition [**75**] to select an appropriate basis and approximate the Koopman operator [**82**]. Data-driven Koopman operators have recently been explored as a method of generating model-based control in robotics applications [**7**]. Unlike in prior work, however, the choice of the Koopman operator representation is explicitly motivated by our sampling-based policy generation method (see Section 6.4.2) and modern computational resources (e.g., multi-core CPUs and GPUs). That is, the Koopman operator representation is particularly well suited for sampling-based control as forward predicting the state of the system requires only a single matrix-vector multiplication, and generating a large set of potential trajectories requires only a single matrix-matrix multiplication. Both of these operations can be easily parallelized on a GPU [**143**]. Related work in model-based control for dynamic systems has utilized linear representations (e.g., Bayesian linear regression [**148**]), however, to the best of our knowledge, ours is the first work to develop a model-based controller the integrates a Koopman operator representation with sampling-based optimal control.

6.4.2. Sampling-based Optimal Control and Predictive Safety

To generate a set of *unconstrained* potential trajectories the user may wish to execute, we sample N inputs from an equally-spaced discretization of the control space. By relying on a uniform prior, we make no assumptions about the user's desired action at the next step (i.e., *there is no model of the user*). These samples can also be generated stochastically. However, stochasticity has known downsides in human-in-the-loop systems. For example, inputs that generate motion directly along a single dimension—a common desire of human operators—are unlikely to be sampled as they exist in segments of the input space that have near zero probability mass when sampling from a continuous distribution.

To ensure that our shared control algorithm only considers trajectories that satisfy the safety constraint described in Section 6.3, we evaluate the configuration of the system at each time step over the receding horizon in each predicted trajectory. If the system violates *hard geometric constraints* (i.e. our definition of safety) that are defined with respect to the environment, we reject the input that generated that trajectory. If, however, the system is safe over the entire course of the predicted trajectory, we consider that input as a viable solution. This can be seen as a *predictive notion of safety* as we evaluate the likelihood of a particular signal leading to a catastrophic failure by observing how we expect the controlled device to evolve over time.

Algorithm 4 MPMI-SC

```
1:  procedure MPMI-SC($t, x_t, u_h$)
2:      $\xi \sim \mathbb{U}^{MxN}$                                ▷ unbiased control samples
3:      for i in N in parallel do                                  ▷ forward predict system
4:          $t_p, x_p, safe \leftarrow t, x_t,$ True
5:          while $t_p < t + T$ and $safe$ do                      ▷ prediction
6:              $x_p \leftarrow f(x_p, \xi(i)) + z_i$              ▷ $z_i$ is Gaussian noise
7:              $safe =$ isSafe($x_p$)                             ▷ system safe at $x_p$
8:              $t_p = t_p + \Delta t$                             ▷ $\Delta t$ is the timestep
9:          end while
10:         if $safe =$ True then                                  ▷ safe over full trajectory
11:             store $\leftarrow \xi(i)$
12:         end if
13:     end for
14:     $u_r \leftarrow \arg\min(cost(\xi(i), u_h)) \; \forall \, \xi(i) \in$ store
15:     return $u_r$                                               ▷ signal is safe and adheres to MIP
16: end procedure
```

6.4.3. Model Predictive Minimal Intervention Shared Control

The full Model Predictive Minimal Intervention Shared Control (MPMI-SC) approach can be seen in Fig. 6.1 and is outlined in Algorithm 4. The inputs are the current time t, the current state x_t and the human partner's input u_h. ξ is the sampled control, $\mathbb{U}$ is the distribution the control is sampled from, M is the dimensionality of the input space, N is the number of samples, and T is the prediction horizon. During forward prediction (Line 6), z_i is sampled i.i.d. from a white noise Gaussian process to account for inaccuracies in the dynamics model. Notably, this computation is done *in parallel on a GPU*. The learned system model $f(x, u)$ predicts the state x_p at the next timestep t_p, which is evaluated for safety (based on hard geometric constraints). The $cost$ function minimizes the influence of the autonomous partner, and u_r is the control signal sent to the robot.

6.4.4. Minimal Intervention Principle and Expected Deviation

An additional benefit of our sampling-based approach is that we can provide an explicit bound on the sub-optimality of the applied control *with respect to the user's instantaneous desires*. In particular, the deviation between the user's input and the applied control signal (*when the user input is safe*) is upper bounded by half the distance between the sampled inputs[1]. This can be computed based on the number of samples generated at each timestep and the Lebesgue measure [**132**] of the control space. This relationship is described in Equation (6.2).

$$\mathbb{E}[||u_h, u_r||] = \frac{\lambda^*(U)}{2N} \tag{6.2}$$

where u_h is the human partners input, u_r is the signal sent to the robot, and $||u, v||$ is the Euclidean distance. $\lambda^*(U)$ is the n-dimensional volume (i.e., Lebesgue measure) of the bounded input space and N is the number of samples. As the number of samples grows ($N \to \infty$), the maximum deviation between the user's input and the applied signal shrinks to 0. However, we also note that while the number of samples (denominator) grows linearly, the Lebesgue measure (numerator) grows exponentially with each additional control dimension (i.e., the measure is defined as the Cartesian product of the intervals of each dimension). This describes a potential issue in the scalability of our sampling based solution; however, this issue is mitigated in the majority (if not all) of our application domains as the dimensionality of the input generally remains low as the human operator must capable of providing input to the system (e.g., using a joystick). The interval of each control dimension also generally remains small (e.g., [-1, 1]

[1]Note, an equally spaced discretization (or a uniform prior) provides the tightest bound on the maximum deviation between the user's input and the applied control signal when we have no knowledge of their desired motion.

or [0, 1]) so that it is understandable by the human partner. A more in-depth discussion of the scalability of our algorithm is presented in the following section.

6.5. Details of Highly Parallelized Implementation

Sampling-based optimal control algorithms are a classic example of "embarrassingly parallel" computation. To compute an optimal control strategy these approaches integrate information from a large number of sampled trajectories that are generated completely independently of one another. Importantly, the expected optimality of the solution generated through these techniques increases with the number of trajectories considered. For this reason, massively parallelized computer architectures (e.g., GPUs and FPGAs) are a natural choice to improve the speed and efficacy of sampling-based optimal control algorithms for real-time applications. We begin this section with a brief introduction to how GPU architectures differ from standard CPU architectures (see Section 6.5.1). We then describe what portions of our algorithm are parallelizable and provide a visualization for the aid of the reader (see Section 6.5.2). Finally, we end with a discussion of the scalability of our algorithm (see Section 6.5.3).

6.5.1. CPU vs GPU Architectures

Central processing units (CPUs) are designed for serial computation that requires potentially complex control logic, while graphics processing units (GPUs) are designed to perform highly parallel multi-threaded computation. From a hardware design perspective this means that CPUs have larger control blocks, larger on-board cache and a smaller number of arithmetic logic units (ALUs). In contrast, GPUs are designed with smaller control blocks, less on-board memory and a significantly larger number of ALUs (see Fig. 6.2).

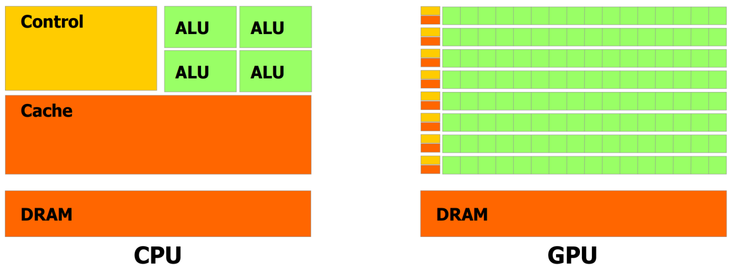


Figure 6.2. Pictorial representation of generic CPU and GPU architectures [1].

GPUs are therefore well suited to solve arithmetic-heavy computations in data-parallel scenarios such as we find in the algorithm described in this paper. In the next section, we describe key implementation details we consider to properly take advantage of the GPU hardware and improve the speed of our shared control paradigm.

6.5.2. GPU Implementation of MPMI-SC

In this section, we focus our description on how we achieve *instruction-level parallelism*. Parallel computation devices can also be used to speed up calculations through *algorithm-level parallelism*, however, we note that our shared control paradigm is already highly-parallelized at this level as it relies on sampling-based techniques.

The important insight from Algorithm 4 is that the *for loop* on Line 3 is performed in parallel. That is, each control sample (from a uniform distribution) is used to generate a potential trajectory that the human partner may wish to take. Each resulting trajectory is therefore generated independently of the other trajectories and is done so in parallel on the GPU. Here, we note that the main part of the MPMI-SC algorithm happens inside this initial for loop, suggesting that the vast majority of the computation can be parallelized.

To implement this algorithm on a GPU, we rely on NVIDIA's CUDA API [**1**]. This platform refers to functions as *compute kernels* which are launched in parallel blocks of threads on an NVIDIA GPU. The order in which a given thread runs on a block is not guaranteed, but CUDA supports syncing functions to ensure that all threads have finished running before moving on to additional computation. To improve the instruction-level parallelism of our code, we follow the principles collected by [**113**]. In particular, key implementations features are:

- minimize memory bandwidth delays by reducing the number of cross-device copies,
- minimize kernel launches through *kernel fusion* [**57**], and
- use *shared memory* when possible to allow fast data access by all threads in a single block.

In contrast to [**113**], we found the cuBLAS optimizations beneficial during the forward prediction step of our algorithm. This relates directly to our choice of system modeling technique—the Koopman operator [**82**]. Unlike related work that uses complex representations (e.g., Neural Networks) to perform sampling-based optimal control [**149**], the Koopman operator can be represented as a single matrix and can therefore forward predict the motion of a system through highly optimized matrix-multiplications. Additionally, we note that selecting the block size and number of blocks has a large impact on the efficiency of the algorithm. We use a principled formula to select the optimal number of blocks based on the block size and the number of samples (see code).

Figure 6.3 is the data flow diagram used to guide our implementation and it highlights the principles defined above. Again, following the structure described by [**113**], we use the CPU as a high-level controller to ensure correct serial processing of the data and minimize the number

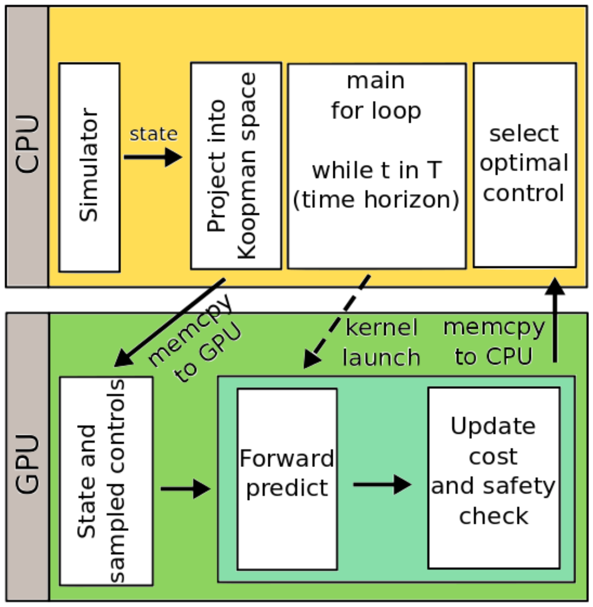


Figure 6.3. Pictorial representation of the data flow of our algorithm on a CPU and GPU. This data flow can be observed in the main callback (i.e., **mpmi_cb**) in the two main files in the associated codebase (i.e., **shared_control_balance_bot.cpp** and **shared_control_race_car.cpp**).

of kernel calls to reduce overhead. Notably there are only two copy operations per iteration—once at the beginning and once at the end—to maximize throughput. Within the main for loop, there is a non-parallelizable **while loop** used to forward predict the state of the system. This computation must be done in sequence as the state of the system at each time-step depends on the prior state. Importantly, however, this computation can still be carried out on the GPU. Additionally, it only requires a single kernel call per time-step and zero memory copies. To ensure fast safety checks on the GPU, all relevant environmental information is loaded into *shared memory* so all parallel threads have access to the information. A final important implementation detail is that we re-project (on the GPU) the state into the Koopman space after each prediction iteration.

6.5.3. Scalability

In this section we discuss the scalability of our proposed algorithm. The main factors we must consider here are:

- the number of control samples,
- the length of the receding horizon, and
- the size of the basis used to project the state into the Koopman space.

The number of control samples defines (1) the set of potential actions the user can choose from, and (2) the number of trajectories we must evaluate for safety. The later computation requires forward predicting the state of the system over a receding horizon. While each trajectory must propagate sequentially, the *set of trajectories* can be computed independently in parallel on a GPU. The limiting factor therefore is the number of blocks and threads on the particular GPU.

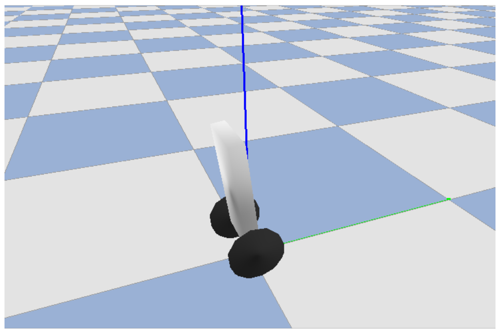
The length of the receding horizon also has a direct impact on the run-time of our algorithm. This computation must be performed in serial for each individual control sample. Serial computation is less efficient on a GPU then on a standard CPU [**1**] which means that shorter time-horizons result in faster overall run-times. However, time-horizons that are too short will result in collision predictions that are not computed early enough to be useful when intervening to improve safety. The limiting factor here is the clockrate of the particular GPU.

Finally, the size of the basis used to project the state into the Koopman space also impacts the efficiency of our algorithm. The main effect of this variable is on the efficiency of the forward prediction computation which requires a series of matrix multiplications. It takes between $\mathcal{O}(n^{2.4}) - \mathcal{O}(n^3)$ to multiple two n x n matrices, depending on the matrix multiplication algorithm used. These computations can be done quicker on a GPU than on a CPU [**143**], but again, a larger basis (and therefore larger matrices) require more powerful GPU hardware. In this work we used an nVidia GeForce GTX 860M, a low-power 2GB GPU, and therefore note that the speed of this system can be further increased with more powerful hardware.

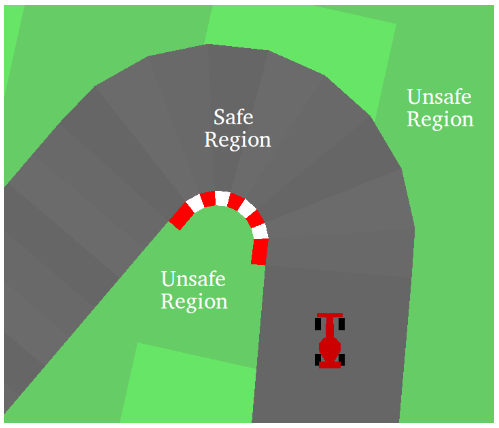
6.6. Experimental Evaluation

We validate our Model Predictive Minimal Intervention Shared Control (MPMI-SC) algorithm with a human subjects study in two simulated environments which we describe below.

6.6.1. Simulated Environments



(a) Simulated balance bot.



(b) Simulated race car.

Figure 6.4. Pictorial representation of simulated environments.

The first dynamic system that users operate is a simulated balance bot (Fig. 6.4a). When controlling this system, users are told (1) that they are free to move around the environment however they would like, and (2) that they should try to make sure that body of the robot does not collide with the ground (i.e., the safety constraint). This can be challenging for a novice operator due to the stabilization requirements. The system is based on an open-source package [**41**] developed using the PyBullet physics engine. The observation space is a three-dimensional continuous vector that includes the angular position and velocity of the robot's body, and the linear velocity of the system. The input is a one dimensional continuous signal that sets a target velocity for both wheels.

The second dynamic system that users operate is a simulated race car (Fig. 6.4b). When controlling this system, users are told (1) that they are free to move around the environment however they would like, and (2) that they should take caution not to drive off of the road (i.e the safety constraint). This can be challenging for a novice operator due to the narrowness of the road and the fact that the car can go unstable (e.g., skid out) if the force applied to the system exceeds the friction limit of the ground surface. This system is based on an open-source package released by OpenAI [**34**] and developed using the Box2D physics engine. The observation space is a six dimensional continuous vector that includes the (x, y, θ) position of the car and the associated velocities $(\dot{x}, \dot{y}, \dot{\theta})$. The input to the system is a three dimensional continuous vector that defines the desired heading for the robot, the positive acceleration (gas) and negative acceleration (break).

The complexity of these two systems makes them useful as testbeds to validate the impact of our shared control algorithm. In particular, stabilization and environmental constraints are important challenges for robotic systems in human-centered domains where the human partner is often co-located with the mechanical system. Our implementation is provided online for free : `https://github.com/asbroad/mpmi_shared_control`.

6.6.2. Experimental Design

To evaluate the impact of our shared control algorithm we ran a human-subjects study (n=20) in which participants operated the system under two distinct interaction paradigms:

- User-only control (No assistance)
- Model Predictive Minimal Intervention Shared Control

The order in which the participants saw the two paradigms was randomized and counter-balanced to account for ordering effects. In each trial, participants were told to perform whatever action they would like so long as the system remained safe. They were also told that in some conditions an autonomous partner would help maintain safety, but they were not told *how* it would help. A trial would end when the system violated one of the defined safety constraints or after a maximum alloted time (balance bot: 20 seconds, race car: 30 seconds). Each participant interacted with the system 10 times in each environment under each control condition. In the race car environment, the morphology of the road was generated randomly at the start of each trial, however, the same random seeds were maintained across participants to ensure that each subject saw the same road configurations.

6.6.3. Implementation Details

6.6.3.1. Safety Computation. In both experimental environments, the system is considered unsafe when it violates hard geometric constraints that represent physical barriers. For the balance bot this is defined as the robot body colliding with the ground. For the race car this is defined as driving off the track. To compute the safety of a predicted trajectory we evaluate the configuration of the system at each discrete timestep. Theoretically, these geometric checks provide strong safety guarantees as we always pick the applied control signal from the subset of sampled inputs that do not violate the defined bounds. In practice, the accuracy of this method relies both on precise system models and our ability to account for noise in the dynamics and/or sensors. In this work, we account for these errors by adding an inflated barrier beyond the natural collision points to reduce the impact of inaccurate predictions. Importantly, even with errors in the dynamics model, and noise in the sensor measurements, one can define an inflation radius that provides strong guarantees on the safety of the system [**86, 85**]. However, in this initial work, we simply rely on a hand-tuned inflation radius.

6.6.3.2. Algorithm Parameters. The implementation of our algorithm requires defining a small set of parameters, the two most important of which are the number of control samples N, and the length of the receding horizon T. As these parameters increase the approximation to the true solution improves, however, this comes at a cost of increased computational complexity. To address this issue, we provide a highly parallelized implementation of our algorithm that evaluates each trajectory independently on an NVIDIA GeForce 860M GPU (details in Chapter 6.5). To provide a good user experience and demonstrate the scalability of our algorithm, we select a large N: 10,000 for the balance bot and 10,120 [2] for the race car. The receding horizon T ($T = 30$ for the balance bot, $T = 25$ for the race car) was chosen based on observations of each system under MPMI-SC with random inputs.

6.6.3.3. Basis Selection. Koopman operator system identification requires defining a basis to approximate the nominally infinite dimensional Hilbert space that the Koopman operator acts on. We select this basis through data-driven methods (as described in Section 6.4.1). In particular, we generate a set of 50 and 150 random basis functions for the balance bot and race car environments, respectively. The chosen learning algorithm [**75**] selects the most relevant

[2]Chosen to produce equally spaced samples in all 3 input dimensions.

basis functions for modeling each system, resulting in 6 and 26 basis functions for the balance bot and race car environments, respectively. All parameter choices are well documented in the open-source code.

6.7. Results

We first explore the impact of our algorithm on the safety of the human-machine system, and the users' response to the intervention of the autonomy. We then detail the computational performance of our system using the defined parameter settings. Finally, we provide a secondary analysis of metrics that relate to the human operator's control skill and style.

6.7.1. Impact of Shared Control on Safety

To evaluate the impact of MPMI-SC on the safety of the joint system we compare (1) the average fraction of safe interactions to unsafe interactions and (2) the average time it took for

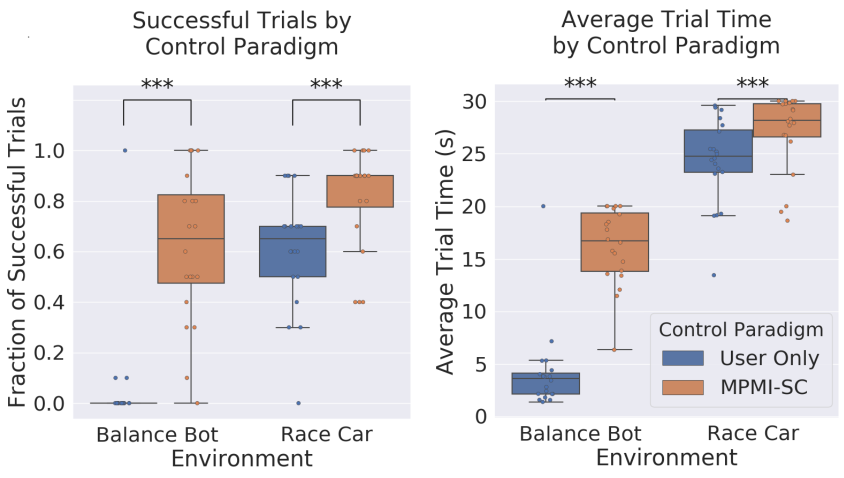


Figure 6.5. Average success rate (left) and average time to failure (right), broken down by environment and control paradigm. The maximum interaction time was 20s in the balance bot and 30s in the race car. Both metrics improve significantly ($* * * : p < 0.005$) under shared control.

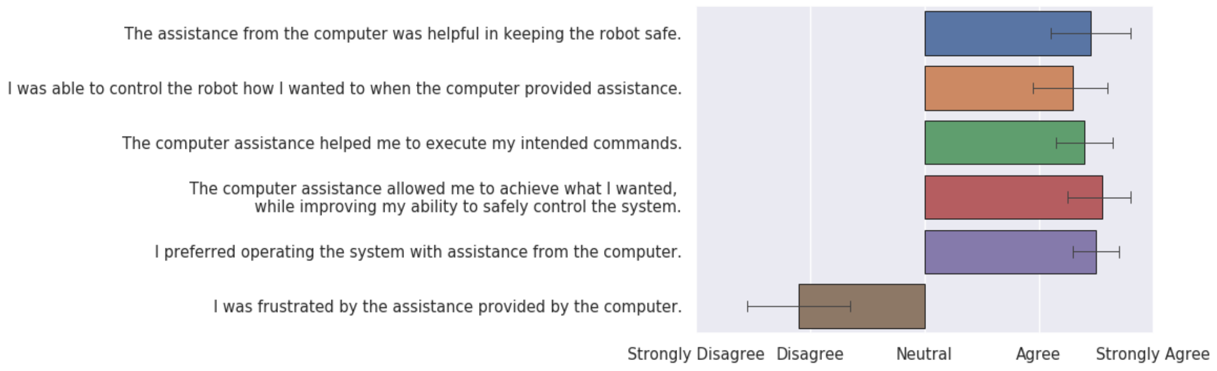


Figure 6.6. Average user response (agreement) to post-experiment questionnaire. Black bars are standard deviations.

the system to enter an unsafe state under each control paradigm (Fig. 6.5). To compare these values, we use a non-parametric Wilcoxon signed-rank test. In both experimental environments we find that MPMI-SC significantly improves the rate at which users are able to safely control the system when compared to a user-only control paradigm ($p < 0.005$). Similarly, we find that users are able to safely control the system for a significantly longer amount of time under shared control then when under user-only control ($p < 0.005$).

6.7.2. User Acceptance of Shared Control Paradigm

We next evaluate the users' acceptance of the assistance provided by the autonomous system. As mentioned in Section 6.2, the majority of shared control systems assist a user in achieving a specific task. The assistance can therefore come at the expense of user satisfaction as the human partner often feels that they are fighting the autonomy [**54**]. In contrast, we develop a task-agnostic shared control paradigm that adheres to the human partner's desires at each instant.

To evaluate the user experience under MPMI-SC we asked participants to fill out a post-experiment questionnaire (Fig. 6.6). Statements were rated on a 5-point Likert-type scale where 1 represents strong disagreement and 5 represents strong agreement. Overall, user's felt the assistance provided by MPMI-SC helped them keep the robot safe and execute their indented commands. Perhaps most telling is that the participants strongly preferred operating the system with assistance from the computer, and did not feel frustrated by the assistance.

6.7.3. System Performance

We now describe the computational efficiency of MPMI-SC using the parameters defined in Section 6.6.3.2. Our algorithm is capable of generating trajectories at $\sim$7000 Hz and $\sim$3500 Hz in the balance bot and race car environments. Incorporating the safety checks, the system runs at $\sim$100 Hz and $\sim$60 Hz, respectively (i.e., between 600,000 and 1,000,000 trajectories every second). MPMI-SC is faster in the balance bot as it relies on a smaller basis and more efficient safety checks. A detailed description of our GPU implementation and the scalability of MPMI-SC is provided in Chapter 6.5.

We also compute the maximum possible deviation between a user's input and the closest safe signal at each timestep. This value is determined by evaluating Equation (6.2) with a known n-dimensional input volume ($\lambda*$) and number of samples (N). If the user's input is considered safe over the receding horizon (i.e., through our prediction method described in Section 6.4.2), this value represents *the maximum influence of the autonomous partner on the applied signal*. If the user's input is not safe over the predicted trajectory, this bound represents *the maximum possible difference between the true closest safe signal and the applied signal*. For the balance bot, the maximum deviation is $\mathbb{E} = \mathbf{0.0001}$ because $\lambda^*(U)$ = 2 (i.e., 1D input that spans $[-1, 1]$)

and $N = 10,000$. For the race car, the maximum deviation is $\mathbb{E} = \mathbf{0.0002}$ because $\lambda^*(U)$ = 4 (i.e., 3D input where the heading spans $[-1, 1]$, the gas spans $[0, 1]$, and the break spans $[-1, 0]$) and $N = 10,120$.

How large an impact an intervention of the described magnitudes has on the dynamics of a dynamic system is dependent on each particular machine. However, we note that both values described in this work are *very* small. We therefore hypothesize that there is likely no discernible difference between the effect of the intended signal and the applied signal when the user's input is considered safe. If there were a large impact on the system dynamics due to an intervention of this magnitude, the system may be chaotic and therefore challenging to operate no matter what controller was used [**61**].

6.7.4. User Control Skill and Style

Finally, we provide a secondary analysis of the data collected during our experiment to evaluate the users' control skill and style. The proposed metrics are easy to compute based on the users' input and the trajectories generated at each timestep.

6.7.4.1. Operator Control Skill. We first calculate the average observed deviation between the user input and the *closest safe signal* as a proxy for the users' understanding of the system dynamics and their control skill. To illustrate this relationship consider a car making a tight turn. If the human partner understands that they need to change their heading and speed to stay on the road during the turn, this metric will remain low (Sec. 6.4.4). If, however, the human partner does not understand this relationship and relies on the autonomous partner to maintain the safety of the system during the turn, this metric will increase. The average value of this metric will be within the described bound ($\mathbb{E}$, above) if, and only if, *all* of the user's controls

Metric	Control	Balance Bot	Race Car
Avg. Deviation	User-only	0.46 ± 0.37	0.05 ± 0.03
	MPMI-SC	0.21 ± 0.21	0.07 ± 0.06
Avg. % Safe Rollouts	User-only	0.28 ± 0.15	0.38 ± 0.09
	MPMI-SC	0.50 ± 0.10	0.35 ± 0.08

Table 6.1. Average deviation and average percentage safe rollouts, broken down by environment and by control condition.

are considered safe during the course of their interaction. Otherwise this value will increase according to the minimal amount required to maintain safety.

6.7.4.2. Operator Control Style. We then calculate the average percentage of sampled rollouts that are safe at each timestep as a proxy for the user's control style. A lower percentage of safe trajectories suggests that the user is operating the system in a more *dangerous* manner (i.e., the system may be closer to entering an ICS). However, we note that this metric alone does not directly relate to the user's control skill as there are many cases in which one may trade off notions of safety for performance. To illustrate this relationship consider, again, a person in a car taking a tight turn to decrease the time of their drive. If this value correlates positively with safe control of the system, we can say that the operator likely has a high degree of skill and is explicitly trading off conventional notions of safety (e.g., distance from an obstacle) for improved performance. However, if this metric correlates positively with unsafe behavior, it is likely that the operator is unskilled and making poor control decisions.

6.7.4.3. Analysis. In the balance bot environment, we find that the user's input more closely aligns with the closest safe signal under MPMI-SC (Table 6.1). We also observe that the system remains in a state where more potential actions remain safe over the horizon. We interpret these results as evidence that the users provided more competent control with assistance than without. In the race car environment, we find that, in both control conditions, the user's input is nearly equally aligned with the closest safe signal, and that there are similar percentages of potentially safe actions. We interpret these results as evidence that MPMI-SC had less impact in this environment then in the balance bot. Evidence of this interpretation can also be seen in Figure 6.5 where we find larger raw differences between the safety metrics in the balance bot environment than in the race car environment. However, the differences between each of these primary safety metrics are statistically significant suggesting that, even when MPMI-SC is less impactful, it still meaningfully improves the safety of the joint system. Both proposed metrics are indirect measures of the user's control skill and style, but the preliminary results (Table 6.1) suggest they warrant further investigation. In future work we plan to evaluate how they evolve over time for an individual operator.

6.8. Discussion

In this section we provide additional observations from our study and describe a limitation of our approach.

6.8.1. Study Observations

One piece of information that is not reflected in our analysis is how people alter their *control strategy* when they are under different paradigms. For example, participants were observed *testing* the limits of the assistance. That is, users would intentionally operate the system at the boundary of safety, and even act in an adversarial manner to test the reliability of MPMI-SC. In contrast, under user-only control, participants were much more cautious. This is potentially a consequence of not explaining *how* the autonomous partner would provide assistance; however, we also believe this behavior aligns with human nature as people often *explore* while they learn.

Another observation relates to why MPPI-SC had a larger impact on the balance bot than on the race car. In particular, participants were more likely to have prior experience operating car-like systems then unstable machines like the balance bot, which could mean they were better able to provide safe control based solely on experience and intuition. Additionally, a main challenge users faced in controlling the race car is that it can enter a *skidding state*. This behavior was not explicitly modeled in our system and therefore not accounted for by the autonomous partner. It is likely we would see further improvements in safety if we (1) explicitly considered the hybrid dynamics of the system or (2) incorporated a cost to penalize the skidding behavior. To ensure safety in the real-world it would be important to explicitly account for these properties when implementing MPMI-SC on new systems.

6.8.2. Limitation in Prediction of Safety

As described in Section 6.4, the control signal sent to the robot is selected based on the user's desires at each instant and therefore does not require a model of the user. However, to compute the safety of a given control action, we implicitly embed a naïve model of the user that assumes the human partner will continue to apply nearly the same input over the time-horizon. The negative implication of this assumption is that we will occasionally reject user inputs that *seem* dangerous, but are not in reality if the user quickly adjusts their strategy. Therefore, there will be trajectories that the person would like to execute and are safe (by recovering at a later timestep) that MPMI-SC incorrectly rejects. Despite this limitation, it is possible that our iterative, receding horizon approach alleviates the impact on the user's acceptance by recomputing the set of dangerous actions at each timestep. We leave a deeper evaluation of this idea to future work.

6.9. Data-driven Trust-based Model-based Shared Control

As a supplementary result to the current chapter, and final piece of analysis in this dissertation, I describe the outcome of a preliminary study that evaluates the intersection of our MPMI-SC framework and the trust-based shared control paradigm proposed in Chapter 3. This evaluation is based data collected from a third experimental condition in the human-subjects study detailed in Section 6.6.2. That is, in addition to the two previously discussed control paradigms, participants operated the robot under MPMI-SC *with a dynamic trust level*. Just as described in Chapter 3, this trust metric evolves based on the observed quality of the interactions between an individual human partner and the autonomous system. The learned trust metric is then used to modulate the control authority granted to the human-in-the-loop to personalize the influence of the autonomous partner, and improve system safety.

6.9.1. Formulation of Trust

To define, and make use of, a formal notion of trust, we detail two important steps as described in Section 3.3. That is, we define the

- Evaluation of User Input
- Modulation of User Input via Trust

The notion of trust discussed in this work is motivated by the same theoretical understanding of the concept described in Chapter 3. We therefore take a control-theoretic viewpoint that rewards users with greater authority when they exhibit a clear understanding of the system dynamics and control problem. In contrast, we grant greater authority to the autonomous partner when there is a lack of quantitative evidence supporting the same hypothesis with the goal of improving the safety of the joint system. This metric therefore includes no task-specific information, and is generalizable to any human-machine system. To *evaluate the user input*, the trust metric is calculated as a function of the deviation between an individual user's input and the closest control input that is deemed safe by our MPMI-SC algorithm. As described in 3.3, we model this variable as a single dimensional Gaussian distribution. We then use this metric to modulate the user input by increasing or decreasing the inflation radius applied to obstacles in the environment. In turn, the inflation radius impacts the MPMI-SC system's prediction of the safety of the robot over a receding horizon. A small inflation radius allows the robot to move close to dangerous regions of the state space, while a large inflation radius restricts the motion of the robot to enforce stronger safety constraints. The trust metric evolves over time according the same probabilistic update equation described in Section 3.3. This work differs from the paradigm presented in Chapter 3 as it aims to improve the safety of the joint system, instead of reducing control effort.

6.9.2. Experimental Design

To evaluate the impact of a dynamic trust metric in our MPMI-SC paradigm, we include a third experimental condition in the human-subjects study described in Section 6.6.2. This third condition requires a small modification to the MPMI-SC paradigm (Algorithm 4) to incorporate the autonomy's trust in the human partner. In particular, instead of relying on a predefined obstacle inflation radius[3] when performing the safety check (Line 7), the new condition uses a dynamic inflation radius. The size of the inflation radius is initially set to a high level as we assume no *a priori* knowledge of the human-in-the-loop and therefore take a conservative

[3]It is standard practice to "inflate" obstacles beyond their natural (or sensed) borders to account for errors in the system model and sensor noise.

approach to ensure the safety of the joint system. The inflation radius is then updated according to the trust metric described above.

It is important to note that, unlike the other two experimental conditions described in this study, MPMI-SC with dynamic trust is always presented to the participant at the end of the study. For this reason, the experimental design does not account for user experience and natural human learning in this new condition. As such, we remind the reader that these results should only be seen as preliminary. On this basis, we explicitly choose not to statistically analyze the results of this experimental condition, and only describe high level trends that are apparent from the study.

6.9.3. Results

The primary metrics we evaluate in this section are the same two metrics described in Section 6.7. As one can observe in Figure 6.7, participants under MPMI-SC with Dynamic Trust

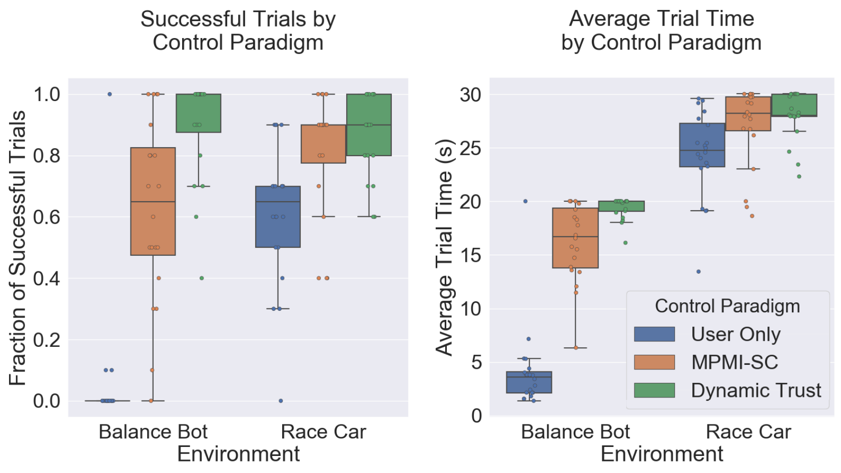


Figure 6.7. Average success rate (left) and average time to failure (right), broken down by environment and control paradigm. The maximum interaction time was 20s in the balance bot and 30s in the race car.

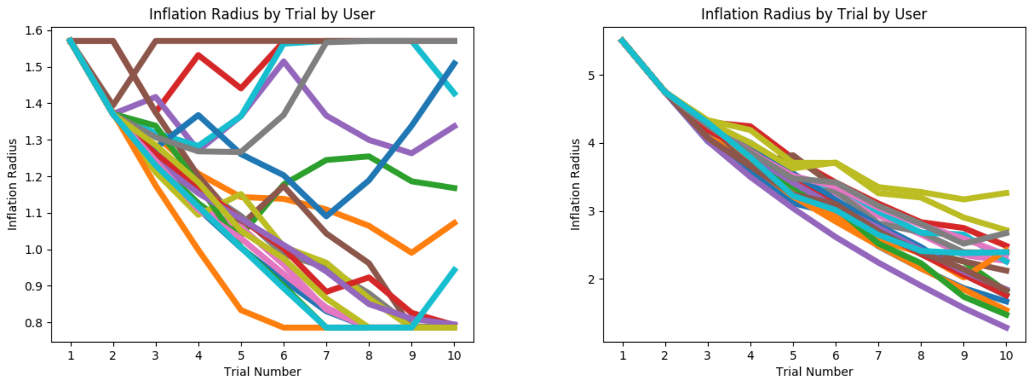


Figure 6.8. Evolution of inflation radius by user in balance bot (left) and race car (right) environments. Each line represents a single user.

perform better, both in terms of the average safety of the interaction and the average length of the interaction, than under user-only control in both environments.

In addition to the primary metrics, we also depict the evolution of the inflation radius by trial number (as a proxy for the trust metric) for each participant in each environment (see Figure 6.8). In the subfigure that represents the evolution of the inflation radius in the balance bot environment (left), we observe a similar patter to the initial result in Section 3.5 (See Figure 3.7). That is, there is no clear trend in the evolution of this metric that leads to the improved performance. This suggests that the influence of trust on the performance of the human-in-the-loop cannot be prescribed to a predefined evolution over time that can be applied to every user. Instead, it appears that the trust metric adapts to each individual user and therefore personalizes the assistance from the autonomous partner.

There is, however, a noticeable difference in the evolution of the inflation radius in the race car environment (Figure 6.8 (right)). In this environment, trust appears to reliably increase for each participant in the study (as observed by the shrinking of the inflation radius). While this result may appear contradictory at first glance, we believe it actually highlights the strength of the proposed metric in evaluating each user's understanding of the system dynamics. That is, as described in Section 6.7, we found that overall, MPMI-SC had a smaller impact on the participants' performance in the race car environment then in the balance bot environment. We hypothesized that one explanation for this result was that participants were more likely to have prior knowledge of how to operate a car-like system. This interpretation was supported by the data when comparing the average performance in the user-only control condition across environments (see Figure 6.5). Under this same interpretation, one would expect trust to grow quickly for each user in the race car environment, as the participants come into the study with useful prior information relating to the system dynamics. As such, participants were able to quickly demonstrate their competency through their interaction, and trust grew accordingly, allocating the vast majority of the control authority to the human-in-the-loop.

6.9.4. Main Takeaways and Caveat

In this supplementary piece of analysis, we provide a preliminary evaluation of a method for incorporating the autonomous partner's trust in the human operator with model predictive minimal intervention shared control. In particular, we demonstrate that the same samples that are used to compute the optimal control solution can be used to compute an adaptive trust metric. This metric can then be used to modulate how control is allocated to each partner in a human-machine system. The results of the described human-subjects study reveal a pattern that suggests that dynamic trust *may* improve the performance of a joint human-machine system with respect to a user-only control baseline. However, the described experiment does not account for experience or human learning and we therefore claim that this first experiment only suggests that incorporating a dynamic notion of the autonomy's trust in an individual user does not degrade the performance of the joint system. Given the trends observed in this preliminary study, we believe the proposed approach could be a fruitful direction for future research.

6.10. Conclusion

In conclusion, we described a shared control paradigm that enhances a human partner's ability to operate complex, dynamic machines by incorporating safety constraints without explicit knowledge of the user's long-term objective. Our approach relies on a simple representation of the joint system (i.e., the Koopman operator) which, in turn, means we can very quickly generate and evaluate the safety of a large number of potential trajectories through the parallelization capabilities of a GPU. Importantly, this representation can be learned from data and therefore generalizes to any pair of partners. Finally, our approach adheres to the minimal intervention principle to ensure that the human partner is allocated the majority of the decision making authority throughout the interaction.

We evaluated the efficacy of our Model Predictive Minimal Intervention Shared Control (MPMI-SC) paradigm with a human-subjects study consisting of 20 participants. The results demonstrated that our approach is able to improve the general safety of the joint system without *a priori* knowledge of the user's desires. Additionally, we found that the participants enjoyed the assistance provided by the autonomy (and reported low levels of frustration), a feature lacking in many shared control paradigms [**54**]. Finally, a preliminary follow up study demonstrates how one could integrate ideas from trust-based shared control into our MPMI-SC framework. Our code is available online for free : `https://github.com/asbroad/mpmi_shared_control`.

CHAPTER 7

Conclusion

Shared control, as a paradigm, is a fundamental building block in the field of human-machine interaction. The long term promise of SC is based on the idea that we can use techniques from autonomous robotics to simultaneously (1) improve a human's ability to operate a dynamic system by offloading challenging aspects of control problem to an autonomous partner, and (2) regulate the user input to enforce safety and stability constraints on the joint system. Continued advancement in the theory, and engineering, of shared control systems is therefore necessary for the successful adoption of human-oriented robotic technologies into our society. In particular, by incorporating an autonomous partner into the control loop of a dynamical system, we can improve, or restore, the natural abilities of a human partner in fields as diverse as assistive and rehabilitative medicine, search and rescue, and transportation.

The majority of modern shared control systems rely on *a priori* knowledge of the system dynamics and the human in the loop. Additionally, SC systems often depend on *a priori* knowledge of, or the ability to accurately predict, the user's desired goal. In this dissertation, I propose a set of data-driven techniques that can be used to develop *platform- and user-independent* shared control paradigms. Additionally, motivated by the fields of assistive and rehabilitative medicine, control is allocated such that the human partner retains a large amount of freedom to achieve unspecified behaviors. The autonomous partner is therefore primarily concerned with task-agnostic features such as stability and safety.

Trust Adaptation Leads to Lower Control Effort in Shared Control of Crane Automation (Chapter 3) In this chapter I describe a data-driven notion of the autonomous partner's confidence in a specific human partner. This trust metric is task-agnostic, and instead based on a control-theoretic foundation that describes the user's understanding of the system dynamics, and their skill in operating the robot. The metric is then used to develop a *personalized* shared control paradigm that adapts to the user in real-time, and is shown to reduce the control effort required to achieve a specific task.

Data-driven Model-based Shared Control of Human-Machine Systems (Chapter 4) In this chapter I introduce data-driven model-based shared control to the field. Data-driven MbSC generalizes the standard SC paradigm to generic pairs of human-machine partners. That is, I describe an efficient model learning methodology for learning quantitative representations of the joint human-machine system from observation. These models are then integrated with techniques from optimal control to generate the policy of the autonomous partner. Finally, the autonomous partner's policy can be used regulate the input of the human partner. Additionally, this chapter demonstrates the data-efficiency of the chosen modeling algorithm as well as the computational-efficiency of the learning algorithm. Importantly, results from the human-subjects study demonstrate the efficacy of the descried data-driven MbSC paradigm in an online learning scenario where we have no *a priori* knowledge of either partner, and no prior observation data to learn from.

Operation and Imitation under Safety-Aware Shared Control (Chapter 5) In this chapter I extend data-driven MbSC to scenarios in which the human operator's desired behavior (or goal) is unknown. The autonomous partner is therefore solely responsible for improving the safety and stability of the joint human-machine system. This paradigm increases the influence of the human operator by removing the assumption that we can predict (or have *a priori* knowledge of) a specific desired goal or motion. Results from a human-subjects study illustrate the significant impact of the described SC paradigm on the safety of the joint system. This chapter also shows that data collected under the safety-aware shared control paradigm can be used to develop autonomous control policies that safely mimic the behavior demonstrated by the human operator. This suggests that task-agnostic SC algorithms can also be used to learn actions a user frequently repeats, which can then be bootstrapped into a task-aware shared control algorithm that uses information from the learned autonomous policies.

Highly Parallelized Data-driven MPC for Minimal Intervention Shared Control (Chapter 6) In this chapter I further extend the influence of the human partner in a task-agnostic data-driven MbSC paradigm. In particular, I describe a SC algorithm that explicitly maximizes the authority granted to the human partner while simultaneously accounting for the safety of the joint system, without *a priori* knowledge of a desired task. A human-subjects study demonstrates (1) the significant impact of our SC system on the safety of the robot, and (2) the fact that the participants' *prefer* our shared control paradigm to a user-only control paradigm, and report *low levels of frustration* when using the system. Finally, I describe two metrics that relate to the user's understanding of the system dynamics and their control skill. These metrics are analogous to the metric proposed in Chapter 3 and, as such, can be interpreted as the autonomous partner's trust in the individual human operator. The results of a preliminary analysis of this idea mirrors the results from Chapter 3.

Taken together, the works presented in this thesis suggest a direction for future shared control research. The described SC algorithms rely on a tight integration between the human and autonomous partners (i.e., the entire autonomous decision making process must be very fast) and appropriately tuned assistance levels (i.e., the human partner is allocated the majority of the control authority, while the autonomous partner is primarily concerned solely with features related to system safety). To address these points, I have described data-driven SC algorithms that are generalizable to any pair of human and machine partners. Additionally, through an iterative, real-time optimization process, the MbSC algorithms are able to respond to, and account for the control deficiencies of, each individual human partner. Finally, we extend the personalization of MbSC algorithms through a data-driven modeling technique that relies on a control-theoretic notion of the autonomous partner's trust in the human partner.

As informative as the chapters of this thesis are, there are of course additional limitations that we have not addressed. The truly exciting opportunity, therefore, lies in future work that builds on the ideas (and open-source code) proposed in this dissertation. For example, while the implementations described in this work significantly improve the safety of the joint human-machine systems we evaluated, they do not provide a guarantee of safety (a feature that could be considered a requirement in certain scenarios). There are, of course, various formal techniques that one could imagine incorporating into the general data-driven model-based shared control paradigm to guarantee safety (e.g., control barrier functions, linear temporal logic, etc...). Additionally, the work presented in this thesis assumes access to high fidelity state information that would otherwise need to be captured via noisy sensors in the real world. How to best integrate real-time sensor information [**30**] is therefore another open research question that significantly improve our ability to transition this research out of the laboratory environment. Similarly, the experiments described in this thesis assume deterministic motion, which allow us to define hard geometric safety constraints without concern for stochasticity in the system dynamics. In future work, we plan to explore alternative methods that can be used to define safety constraints for non-deterministic systems [**35, 98**]. In conclusion, the continued development of data-driven model-based shared control algorithms offers the promise of extending a human beings ability to achieve tasks that were previously too complex or dangerous to consider without the aid of a robotic partner.

References

[1] NVIDIA CUDA C Programming Guide. *Nvidia Corporation*, 120(18):8, 2011.

[2] Mobile Crane Operator Certification Exam, 2014.

[3] Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. SAE J3016, SAE International, Warrendale, PA, 2016.

[4] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine learning*, page 1. ACM, 2004.

[5] David A Abbink, Tom Carlson, Mark Mulder, Joost CF de Winter, Farzad Aminravan, Tricia L Gibo, and Erwin R Boer. A Topology of Shared Control Systems—Finding Common Ground in Diversity. *Transactions on Human-Machine Systems*, (99):1–17, 2018.

[6] Ian Abraham, Alexander Broad, Brenna Argall, and Todd Murphey. SMPO : Switching Mode Policy Optimization. In *In Submission*, 2019.

[7] Ian Abraham, Gerardo De La Torre, and Todd D Murphey. Model-Based Control Using Koopman Operators. 2017.

[8] Peter Aigner and Brenan J McCarragher. Modeling and Constraining Human Interactions in Shared Control Utilizing a Discrete Event Framework. *Transactions on Systems, Man, and Cybernetics*, 30(3):369–379, 2000.

[9] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe Reinforcement Learning via Shielding. In *AAAI Conference on Artificial Intelligence*, 2018.

[10] Helmut Alt and Michael Godau. Computing the Fréchet Distance Between Two Polygonal Curves. *International Journal of Computational Geometry & Applications*, 5(1):75–91, 1995.

[11] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control Barrier Functions: Theory and Applications. *arXiv:1903.11199*, 2019.

[12] Brian D. O. Anderson and John B. Moore. *Optimal Control: Linear Quadratic Methods*. Dover Publications, 2007.

[13] Sterling J Anderson, Sisir B Karumanchi, and Karl Iagnemma. Constraint-based Planning and Control for Safe, Shared Control of Ground Vehicles. In *IEEE International Vehicles Symposium*, 2012.

[14] Sterling J Anderson, Sisir B Karumanchi, Karl Iagnemma, and James M Walker. The Intelligent CoPilot: A Constraint-based Approach to Shared-Adaptive Control of Ground Vehicles. *Intelligent Transportation Systems Magazine*, 5(2):45–54, 2013.

[15] Sterling J Anderson, Steven C Peters, Tom E Pilutti, and Karl Iagnemma. An Optimal-Control-Based Framework for Trajectory Planning, Threat Assessment, and Semi-Autonomous Control of Passenger Vehicles in Hazard Avoidance Scenarios. *International Journal of Vehicle Autonomous Systems*, 8(2-4):190–216, 2010.

[16] Sterling J Anderson, James M Walker, and Karl Iagnemma. Experimental Performance Analysis of a Homotopy-based Shared Autonomy Framework. *Transactions on Human-Machine Systems*, 44(2):190–199, 2014.

[17] Alexander R Ansari and Todd D Murphey. Sequential Action Control: Closed-form Optimal Control for Nonlinear and Nonsmooth Systems. *Transactions on Robotics*, 32(5):1196–1214, 2016.

[18] Brenna D. Argall. Autonomy in Rehabilitation Robotics: An Intersection. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):441–463, 2018.

[19] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[20] Christopher G Atkeson and Juan Carlos Santamaria. A Comparison of Direct and Model-Based Reinforcement Learning. In *International Conference on Robotics and Automation*, volume 4, pages 3557–3564. IEEE, 1997.

[21] S.K. Au, P. Bonato, and H. Herr. An EMG-position Controlled System for an Active Ankle-foot Prosthesis: An Initial Experimental Study. In *IEEE International Conference on Rehabilitation Robotics (ICORR)*, 2005.

[22] Andrew G Barto, Steven J Bradtke, and Satinder P Singh. Learning to Act Using Real-Time Dynamic Programming. *Artificial Intelligence*, 72(1-2):81–138, 1995.

[23] Antoine Bautin, Luis Martinez-Gomez, and Thierry Fraichard. Inevitable Collision States: A Probabilistic Perspective. In *International Conference on Robotics and Automation*, pages 4022–4027. IEEE, 2010.

[24] Dimitri P Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena scientific Belmont, MA, 1995.

[25] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[26] Alexander Broad, Ian Abraham, Todd Murphey, and Brenna Argall. Structured Neural Network Dynamics for Model-based Control. In *RSS Workshop on Learning and Inference in Robotics*, 2018.

[27] Alexander Broad, Ian Abraham, Todd Murphey, and Brenna Argall. Data-Driven Model-based Shared Control of Human-Machine Systems. In *In Submission*, 2019.

[28] Alexander Broad and Brenna Argall. Geometry-Based Region Proposals for Real-Time Robot Detection of Tabletop Objects. *arXiv:1703.04665*, 2017.

[29] Alexander Broad, Deepak Gopinath, Todd Murphey, and Brenna D Argall. An Empirical Analysis of Methods for Learning Robot Kinematics from Demonstration. In *Midwest Robotics Workshop*, 2017.

[30] Alexander Broad, Michael Jones, and Teng-Yok Lee. Recurrent Multi-frame Single Shot Detector for Video Object Detection. In *British Machine Vision Conference*, 2018.

[31] Alexander Broad, Todd Murphey, and Brenna Argall. Learning Models for Shared Control of Human-Machine Systems with Unknown Dynamics. In *Robotics: Science and Systems*, 2017.

[32] Alexander Broad, Todd Murphey, and Brenna Argall. Highly Parallelized Data-driven MPC for Minimal Intervention Shared Control. In *Robotics: Science and Systems*, 2019.

[33] Alexander Broad, Jarvis Schultz, Matthew Derry, Todd Murphey, and Brenna Argall. Trust Adaptation Leads to Lower Control Effort in Shared Control of Crane Automation. *Robotics and Automation Letters*, 2(1):239–246, 2017.

[34] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv*, abs/1606.01540, 2016.

[35] Daniel S Brown and Scott Niekum. Efficient Probabilistic Performance Bounds for Inverse Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, 2018.

[36] Bingni W. Brunton, Lise A. Johnson, Jeffrey G. Ojemann, and J. Nathan Kutz. Extracting Spatial-Temporal Coherent Patterns in Large-Scale Neural Recordings using Dynamic Mode Decomposition. *Journal of Neuroscience Methods*, 258:1–15, 2016.

[37] Marko Budišić, Ryan Mohr, and Igor Mezić. Applied Koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012.

[38] Francesco Bullo and Andrew D. Lewis. *Geometric Control of Mechanical Systems*, volume 49 of *Texts in Applied Mathematics*. Springer Verlag, New York-Heidelberg-Berlin, 2004.

[39] Tom Carlson and Yiannis Demiris. Collaborative Control for a Robotic Wheelchair: Evaluation of Performance, Attention, and Workload. *Transactions on Systems, Man, and Cybernetics*, 42(3):876–888.

[40] Tom Carlson and Yiannis Demiris. Human-Wheelchair Collaboration through Prediction of Intention and Adaptive Assistance. In *IEEE International Conference on Robotics and Automation*, pages 3926–3931, 2008.

[41] Yannis Chatzikonstantinou. Balance bot. `https://github.com/yconst/balance-bot`, 2018.

[42] R. Chipalkatty, H. Daepp, M. Egerstedt, and W. Book. Human-in-the-loop: MPC for Shared Control of a Quadruped Rescue Robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[43] Rahul Chipalkatty, Greg Droge, and Magnus B Egerstedt. Less is More: Mixed-Initiative Model-Predictive Control with Human Inputs. *IEEE Transactions on Robotics*, 29(3):695–703, 2013.

[44] Christian Cipriani, Franco Zaccone, Silvestro Micera, , and M. Chiara Carrozza. On the Shared Control of an EMG-Controlled Prosthetic Hand: Analysis of User-Prosthesis Interaction. *IEEE Transactions on Robotics*, 24(1):170–184, 2008.

[45] Gery Colombo, Matthias Joerg, Reinhard Schreier, and Volker Dietz. Treadmill Training of Paraplegic Patients using a Robotic Orthosis. *Journal of Rehabilitation Research and Development*, 37(6):693–700, 2000.

[46] J.C.F. de Winter and D. Dodou. Preparing drivers for dangerous situations: A critical reflection on continuous shared control. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2011.

[47] Matthew Derry and Brenna Argall. Automated Doorway Detection for Assistive Shared-Control Wheelchairs. In *International Conference on Robotics and Automation*, pages 1254–1259. IEEE, 2013.

[48] Munjal Desai. *Modeling Trust to Improve Human-robot Interaction*. PhD thesis, University of Massachusetts Lowell, 2012.

[49] Moritz Diehl, Hans Georg Bock, and Johannes P Schlöder. A Real-time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

[50] Anca Dragan and Siddhartha Srinivasa. A Policy Blending Formalism for Shared Control. *International Journal of Robotics Research*, 32(7):790–805, May 2013.

[51] Paul Drews, Grady Williams, Brian Goldfain, Evangelos A. Theodorou, and James M. Rehg. Aggressive Deep Driving: Combining Convolutional Neural Networks and Model Predictive Control. In *Conference on Robot Learning*, volume 78, pages 133–142, Nov 2017.

[52] Thomas Eiter and Heikki Mannila. Computing Discrete Fréchet Distance. Technical report, Citeseer, 1994.

[53] Ahmetcan Erdogan and Brenna Argall. The Effect of Robotic Wheelchair Control Paradigm and Interface on User Performance, Effort and Preference: An Experimental Assessment. *Robotics and Autonomous Systems*, 94:282–297, 2017.

[54] Ahmetcan Erdogan and Brenna D Argall. Prediction of User Preference over Shared Control Paradigms for a Robotic Wheelchair. In *IEEE International Conference on Rehabilitation Robotics*, July 2017.

[55] Stephen M Erlien, Susumu Fujita, and Joseph Christian Gerdes. Shared Steering Control using Safe Envelopes for Obstacle Avoidance and Vehicle Stability. *Transactions on Intelligent Transportation Systems*, 17(2):441–451, 2016.

[56] Hans Joachim Ferreau, Hans Georg Bock, and Moritz Diehl. An Online Active Set Strategy to Overcome the Limitations of Explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[57] Jiří Filipovič, Matúš Madzin, Jan Fousek, and Luděk Matyska. Optimizing CUDA Code by Kernel Fusion: Application on BLAS. *The Journal of Supercomputing*, 71(10):3934–3957, 2015.

[58] Christian Fisahn, Mirko Aach, Oliver Jansen, Marc Moisi, Angeli Mayadev, Krystle T Pagarigan, Joseph R Dettori, and Thomas A Schildhauer. The Effectiveness and Safety of Exoskeletons as Assistive and Rehabilitation Devices in the Treatment of Neurologic Gait Disorders in Patients with Spinal Cord Injury: A Systematic Review. *Global Spine Journal*, 6(08):822–841, 2016.

[59] Kathleen Fitzsimons, Ana Maria Acosta, Julius P. A. Dewald, and Todd D. Murphey. Ergodicity Reveals Assistance and Learning from Physical Human-Robot Interaction. *Science Robotics*, 4(29), 2019.

[60] Kathleen Fitzsimons, Emmanouil Tzorakoleftherakis, and Todd Murphey. Optimal Human-In-The-Loop Interfaces Based on Maxwell's Demon. In *American Control Conference*, pages 4397–4402, 2016.

[61] Alexander L Fradkov and Robin J Evans. Control of Chaos: Methods and Applications in Engineering. *Annual Reviews in Control*, 29(1):33–56, 2005.

[62] Thierry Fraichard and Hajime Asama. Inevitable Collision States—A Step Towards Safer Robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[63] Nathan Fulton and André Platzer. Safe Reinforcement Learning via Formal Methods. In *AAAI Conference on Artificial Intelligence*, 2018.

[64] Fei Gao, M.L. Cummings, and E.T. Solovey. Modeling Teamwork in Supervisory Control of Multiple Robots. *IEEE Transactions on Human-Machine Systems*, 44(4):441–453, 2014.

[65] Rui Gonçalves, Sérgio Ferreira, José Pinto, João Sousa, and Gil Gonçalves. Authority Sharing in Mixed Initiative Control of Multiple Uninhabited Aerial Vehicles. In *Engineering Psychology and Cognitive Ergonomics*, volume 6781, pages 530–539. Springer Berlin Heidelberg, 2011.

[66] Deepak Gopinath, Siddarth Jain, and Brenna D Argall. Human-in-the-Loop Optimization of Shared Autonomy in Assistive Robotics. *Robotics and Automation Letters*, 2(1):247–254, 2017.

[67] Ashraf S Gorgey. Robotic Exoskeletons: The Current Pros and Cons. *World Journal of Orthopedics*, 9(9):112, 2018.

[68] John Hauser. A Projection Operator Approach to the Optimization of Trajectory Functionals. volume 35, pages 377–382, 2002.

[69] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel Methods in Machine Learning. *The Annals of Statistics*, pages 1171–1220, 2008.

[70] Jia-Chen Hua, Sukesh Roy, Joseph L. McCauley, and Gemunu H. Gunaratne. Using Dynamic Mode Decomposition to Extract Cyclic Behavior in the Stock Market. *Physica A: Statistical Mechanics and its Applications*, 448:172–180, 2016.

[71] Shervin Javdani, Henny Admoni, Stefania Pellegrinelli, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared Autonomy via Hindsight Optimization for Teleoperation and Teaming. *The International Journal of Robotics Research*, 37(7):717–742, 2018.

[72] Shervin Javdani, Siddhartha Srinivasa, and J. Andrew Bagnell. Shared Autonomy via Hindsight Optimization. In *Robotics: Science and Systems*, Rome, Italy, 2015.

[73] Elliot Johnson, Jarvis Schultz, and Todd Murphey. Structured Linearization of Discrete Mechanical Systems for Analysis and Optimal Control. *IEEE Trans. on Automation Sci. and Eng.*, 12(1):140–152, 2014.

[74] Elliot R. Johnson and Todd D. Murphey. Scalable Variational Integrators for Constrained Mechanical Systems in Generalized Coordinates. *IEEE Trans. on Robotics*, 25:1249–1261, October 2009.

[75] Mihailo R Jovanović, Peter J Schmid, and Joseph W Nichols. Sparsity-Promoting Dynamic Mode Decomposition. *Physics of Fluids*, 26(2):024103, 2014.

[76] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[77] Aleksandra Kalinowska, Thomas Berrueta, and Todd D Murphey. Data-Driven Gait Segmentation for Walking Assistance in a Lower-Limb Assistive Device. In *International Conference on Robotics and Automation*, 2019.

[78] Aleksandra Kalinowska, Kathleen Fitzsimons, Julius Dewald, and Todd D Murphey. Online User Assessment for Minimal Intervention During Task-Based Robotic Assistance. In *Robotics: Science and Systems*, 2018.

[79] S Mohammad Khansari-Zadeh and Aude Billard. Learning Stable Nonlinear Dynamical Systems with Gaussian Mixture Models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[80] Hyun K Kim, J Biggs, W Schloerb, M Carmena, Mikhail A Lebedev, Miguel AL Nicolelis, and Mandayam A Srinivasan. Continuous Shared Control for Stabilizing Reaching and Grasping with Brain-Machine Interfaces. *IEEE Transactions on Biomedical Engineering*, 53(6):1164–1173, 2006.

[81] Won S Kim, Blake Hannaford, and AK Fejczy. Force-Reflection and Shared Compliant Control in Operating Telemanipulators with Time Delay. *Transactions on Robotics and Automation*, 8(2):176–185, 1992.

[82] Bernard O Koopman. Hamiltonian Systems and Transformation in Hilbert Space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[83] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[84] David Kortenkamp, Debra Keirn-Schreckenghost, and R Peter Bonasso. Adjustable Control Autonomy for Manned Space Flight. In *Aerospace Conference*, volume 7, pages 629–640. IEEE, 2000.

[85] Shreyas Kousik, Sean Vaskov, Fan Bu, Matthew Johnson-Roberson, and Ram Vasudevan. Bridging the Gap Between Safety and Real-Time Performance in Receding-Horizon Trajectory Design for Mobile Robots. *arXiv:1809.06746*, 2018.

[86] Shreyas Kousik, Sean Vaskov, Matthew Johnson-Roberson, and Ram Vasudevan. Safe Trajectory Synthesis for Autonomous Driving in Unforeseen Environments. In *ASME Dynamic Systems and Control Conference*, 2017.

[87] Gerard Lacey and Shane MacNamara. Context-Aware Shared Control of a Robot Mobility Aid for the Elderly Blind. *International Journal of Robotics Research*, 19(11):1054–1065, 2000.

[88] Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. DART: Noise Injection for Robust Imitation Learning. In *Conference on Robot Learning*, pages 143–156, 2017.

[89] Przemyslaw A Lasota, Terrence Fong, Julie A Shah, et al. A Survey of Methods for Safe Human-Robot Interaction. *Foundations and Trends in Robotics*, 5(4):261–349, 2017.

[90] Steven M LaValle and James J Kuffner Jr. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[91] Martin Lawitzky, Melanie Kimmel, Peter Ritzer, and Sandra Hirche. Trajectory Generation Under the Least Action Principle for Physical Human-Robot Cooperation. In *IEEE International Conference on Robotics and Automation*, pages 4285–4290. IEEE, 2013.

[92] Adam Eric Leeper, Kaijen Hsiao, Matei Ciocarlie, Leila Takayama, and David Gossow. Strategies for Human-in-the-loop Robotic Grasping. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2012.

[93] K. Leung, E. Schmerling, M. Chen, J. Talbot, J. C. Gerdes, and M. Pavone. On Infusing Reachability-Based Safety Assurance within Probabilistic Planning Frameworks for Human-Robot Vehicle Interactions. In *International Symposium on Experimental Robotics*, 2018.

[94] Sergey Levine and Vladlen Koltun. Guided Policy Search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[95] Weiwei Li and Emanuel Todorov. Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems. In *International Conference on Informatics in Control, Automation and Robotics*, volume 1, pages 222–229, August 2004.

[96] Yugang Liu and Goldie Nejat. Robotic Urban Search and Rescue: A Survey from the Control Perspective. *Journal of Intelligent & Robotic Systems*, 72(2):147–165, 2013.

[97] Dylan P Losey, Craig G McDonald, Edoardo Battaglia, and Marcia K O'Malley. A Review of Intent Detection, Arbitration, and Communication Aspects of Shared Control for Physical Human–Robot Interaction. *Applied Mechanics Reviews*, 70(1):010804, 2018.

[98] Anirudha Majumdar and Russ Tedrake. Funnel Libraries for Real-time Robust Feedback Motion Planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[99] George Mathew and Igor Mezić. Metrics for Ergodicity and Design of Ergodic Dynamics for Multi-Agent Systems. *Physica D: Nonlinear Phenomena*, 240(4):432–442, 2011.

[100] N. Matni and M. Oishi. Reachability-based Abstraction for an Aircraft Landing under Shared Control. In *Proceedings of the American Control Conference (ACC)*, 2008.

[101] Jacob Mattingley, Yang Wang, and Stephen Boyd. Receding Horizon Control. *Control Systems, IEEE*, 31(3):52–65, 2011.

[102] David Mayne. A Second-Order Gradient Method for Determining Optimal Trajectories of Non-Linear Discrete-Time Systems. *International Journal of Control*, 3(1):85–95, 1966.

[103] Lauren M Miller, Yonatan Silverman, Malcolm A MacIver, and Todd D Murphey. Ergodic Exploration of Distributed Information. *IEEE Transactions on Robotics*, 32(1):36–52, 2016.

[104] Djordje Mitrovic, Stefan Klanke, and Sethu Vijayakumar. Adaptive Optimal Feedback Control with Learned Internal Dynamics Models. In *From Motor Learning to Interaction Learning in Robots*, pages 65–84. Springer, 2010.

[105] Selma Musić and Sandra Hirche. Control Sharing in Human-Robot Team Interaction. *Annual Reviews in Control*, 44:342–354, 2017.

[106] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural Network Dynamics for Model-based Deep Reinforcement Learning with Model-free Fine-tuning. In *International Conference on Robotics and Automation*, pages 7559–7566. IEEE, 2018.

[107] Duy Nguyen-Tuong and Jan Peters. Model Learning For Robot Control: A Survey. *Cognitive processing*, 12(4):319–340, 2011.

[108] Duy Nguyen-Tuong, Jan R Peters, and Matthias Seeger. Local Gaussian Process Regression for Real Time Online Model Learning. In *Advances in Neural Information Processing Systems*, pages 1193–1200, 2009.

[109] I. Nourbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion. Human-Robot Teaming for Search and Rescue. *Pervasive Computing*, 4(1):72–79, 2005.

[110] Shahin S Nudehi, Ranjan Mukherjee, and Moji Ghodoussi. A Shared-Control Approach to Haptic Interface Design for Minimally Invasive Telesurgical Training. *Transactions on Control Systems Technology*, 13(4):588–592, 2005.

[111] Meeko Oishi. Assessing Information Availability for User-Interfaces of Shared Control Systems under Reference Tracking. In *Proceedings of the American Control Conference (ACC)*, 2014.

[112] Charles Pippin and Henrik Christensen. Trust Modeling in Multi-Robot Patrolling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[113] Brian Plancher and Scott Kuindersma. A Performance Analysis of Parallel Differential Dynamic Programming on a GPU. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018.

[114] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Generalizing Koopman Theory to Allow for Inputs and Control. *arXiv*, abs/1602.07647, 2016.

[115] J.B. Rawlings, D.Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[116] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Shared Autonomy via Deep Reinforcement Learning. *Robotics: Science and Systems*, 2018.

[117] R Tyrrell Rockafellar and Roger J-B Wets. *Variational Analysis*, volume 317. Springer Science & Business Media, 2009.

[118] Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[119] Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral Analysis of Nonlinear Flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.

[120] Stefan Schaal and Christopher G Atkeson. Learning Control in Robotics. *Robotics and Automation Magazine*, 17(2):20–29, 2010.

[121] Peter J Schmid. Dynamic Mode Decomposition of Numerical and Experimental Data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[122] Jarvis Schultz, Elliot Johnson, and Todd D. Murphey. Trajectory Optimization in Discrete Mechanics. In *Differential-Geometric Methods in Computational Multibody System Dynamics*. Springer International Publishing, 2015.

[123] Jarvis Schultz and Todd D. Murphey. Extending Filter Performance Through Structured Integration. In *American Controls Conf. (ACC)*, 2014.

[124] Jarvis Schultz and Todd D. Murphey. Real-time Trajectory Generation for a Planar Crane Using Discrete Mechanics. In *IROS Workshop on Real-Time Motion Generation and Control*, 2014.

[125] Wilko Schwarting, Javier Alonso-Mora, Liam Paull, Sertac Karaman, and Daniela Rus. Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention. In *IEEE International Conference on Robotics and Automation*, 2017.

[126] Wilko Schwarting, Javier Alonso-Mora, Liam Paull, Sertac Karaman, and Daniela Rus. Safe Nonlinear Trajectory Generation for Parallel Autonomy with a Dynamic Vehicle Model. *IEEE Transactions on Intelligent Transportation Systems*, (99):1–15, 2017.

[127] Jian Shen, Javier Ibanez-Guzman, Teck Chew Ng, and Boon Seng Chew. A Collaborative-Shared Control System with Safe Obstacle Avoidance Capability. In *IEEE International Conference on Robotics, Automation and Mechatronics*, pages 119–123, 2004.

[128] Victor A Shia, Yiqi Gao, Ramanarayan Vasudevan, Katherine Driggs Campbell, Theresa Lin, Francesco Borrelli, and Ruzena Bajcsy. Semiautonomous Vehicular Control using Driver Modeling. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2696–2709, 2014.

[129] Gunter Stein. Respect the Unstable. *IEEE Control Systems Magazine*, 272(1708/03), 2003.

[130] JP Sutherland. Fly-by-Wire Flight Control Systems. Technical report, Air Force Flight Dynamics Lab Wright-Patterson AFB OH, 1968.

[131] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[132] Terence Tao. *An Introduction to Measure Theory*. American Mathematical Society, 2011.

[133] Yuval Tassa, Tom Erez, and William D Smart. Receding Horizon Differential Dynamic Programming. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2008.

[134] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited Differential Dynamic Programming. In *International Conference on Robotics and Automation*, pages 1168–1175. IEEE, 2014.

[135] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A Generalized Path Integral Control Approach to Reinforcement Learning. *Journal of Machine Learning Research*, 11(Nov):3137–3181, 2010.

[136] Robert Tibshirani. Regression Shrinkage and Selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[137] Luca Tonin, Robert Leeb, Michele Tavella, Serafeim Perdikis, and José del R Millán. The Role of Shared-Control in BCI-based Telepresence. In *International Conference on Systems, Man and Cybernetics*, pages 1462–1466. IEEE, 2010.

[138] Hoang T Trieu, Hung T Nguyen, and Keith Willey. Shared Control Strategies for Obstacle Avoidance Tasks in an Intelligent Wheelchair. In *International Conference on Engineering in Medicine and Biology Society*, pages 4254–4257, 2008.

[139] Kakin K. Tsoi, Mark Mulder, and David A. Abbink. Balancing Safety and Support: Changing Lanes with a Haptic Lane-keeping Support System. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2010.

[140] Jonathan H Tu, Clarence W Rowley, Dirk M Luchtenburg, Steven L Brunton, and J Nathan Kutz. On Dynamic Mode Decomposition: Theory and Applications. *Journal of Computational Dynamics*, 1:391, 2014.

[141] Jonathan H Tu, Clarence W Rowley, Dirk M Luchtenburg, Steven L Brunton, and J Nathan Kutz. On Dynamic Mode Decomposition: Theory and Applications. *Journal of Computational Dynamics*, 1:391–421, 2014.

[142] Emmanouil Tzorakoleftherakis and Todd D Murphey. Controllers as Filters: Noise-Driven Swing-Up Control Based on Maxwells Demon. In *IEEE Conference on Decision and Control*, 2015.

[143] Vasily Volkov and James W Demmel. Benchmarking GPUs to Tune Dense Linear Algebra. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2008.

[144] Li Wang, Aaron D Ames, and Magnus Egerstedt. Safety Barrier Certificates for Collisions-Free Multirobot Systems. *IEEE Transactions on Robotics*, 33(3):661–674, 2017.

[145] Zhixuan Wei, Weidong Chen, and Jingchuan Wang. 3D Semantic Map-based Shared Control for Smart Wheelchair. In *International Conference on Intelligent Robotics and Applications*, pages 41–51. Springer, 2012.

[146] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*, volume 25. MIT press, 1965.

[147] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model Predictive Path Integral Control: From Theory to Parallel Computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[148] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive Driving with Model Predictive Path Integral Control. In *International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[149] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information Theoretic MPC for Model-Based Reinforcement Learning. In *International Conference on Robotics and Automation*, 2017.

[150] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A Data–driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[151] S. Paul Wright. Adjusted P-Values for Simultaneous Inference. *Biometrics*, 48(4):1005–1013, 1992.

[152] Anqi Xu and Gregory Dudek. Trust-Driven Interactive Visual Navigation for Autonomous Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[153] Anqi Xu and Gregory Dudek. OPTIMo: Online Probabilistic Trust Inference Model for Asymmetric Human-Robot Collaborations. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2015.

[154] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, volume 8, pages 1433–1438, 2008.