A FRAMEWORK FOR IMPROVING THE SPEED AND PERFORMANCE OF TELEOPERATED MOBILE MANIPULATORS

by

Steven Eric Vozar

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2013

Doctoral Committee:

Professor Dawn Tilbury, Chair
Associate Professor Brent Gillespie
Paul Muench, US Army TARDEC
Assistant Professor Edwin Olson
Professor Nadine Sarter

This dissertation is dedicated to the memory of my mother,
Hinda Vozar

ACKNOWLEDGEMENTS

I would like to thank the Rackham Graduate School as well as the Automotive Research Center (ARC) at the University of Michigan, with funding from government contract DoD-DoA W56H2V-04-2-0001 through the US Army Tank Automotive Research, Development, and Engineering Center for financial support for this doctoral work. This research would not have been possible without this generous support.

I owe a tremendous debt of gratitude to Professor Dawn Tilbury for being all that I could ask for (and more!) in a PhD advisor. Her willingness to let me define my own research and experiment with new ideas with a watchful yet patient eye has allowed me to take risks that might not otherwise have been possible. I can say without hesitation that her teaching and mentoring have had extraordinarily positive impacts on my academic and professional development.

Thanks also to others who have provided me with mentorship over the years: Professor Volker Sick, who gave me my first independent research project, found funding for and advised my masters thesis, and encouraged me to pursue a doctoral degree. Professor A. John Hart, who took an unselfish interest in my career aspirations and academic work, and has given me immeasurable advice. Dr. Paul Muench, who has gone above and beyond his duty as TARDEC liaison, taking an interest not just in my research, but in me and my future.

To my student colleagues, past and present, whom I can always count on for an interesting conversation about robots: Dhananjay Anand, John Broderick, Justin

Storms, Josh Langsfeld, Ryan Morton, and Andrew Richardson. Thanks for checking my math, my code, and my grammar. Also, to the research assistants that have worked for me: Peter Turpel, Matthew Ko, and Jeff Abromowitz. You have helped me grow as a supervisor, and your efforts are greatly appreciated.

Thanks also to the administrative staff of the ARC, the Mechanical Engineering Department, and the College of Engineering who run the show behind the scenes, as well as any other staff who have helped me fill out a form, buy a servo, or navigate red tape. Your dedication makes research in higher education possible.

To my friends, whom I don't dare list out for fear of glaring omission: Thanks for the memories, the jokes, the trips, the hijinks, the burgers, the beer, the trivia, the sports, and the sanity checks.

Finally, to my family: nothing I write here will come close describing the amount of gratitude I owe you all for helping me get to this point. Your enthusiastic support of all my endeavors means the world to me. Thanks for shaping the person I have become.

Go Blue!

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

A Framework for Improving the Speed and Performance of Teleoperated Mobile Manipulators

by
Steven Eric Vozar

Chair: Professor Dawn Tilbury

Despite recent advances in robot autonomy, teleoperation remains an integral part of many robot tasks. In situations where it is hazardous or difficult for humans to be present, but which require human judgment and decision-making skills, the use of a human operator is the only option. However, there are many issues resulting from limited feedback channels that degrade perception and manipulation abilities in remote environments, causing even basic robot tasks to be difficult and time-consuming. For robots to become more useful tools for humans in remote environments, the speed and ease of teleoperated tasks must be increased.

This purpose of this dissertation is to develop a framework for increasing speed and performance of teleoperated mobile robot tasks. First, the key issues affecting teleoperated robot system performance are defined and characterized. These factors are incorporated into an optimization-based approach for evaluating multiple design options for teleoperated systems. This optimization may require models for system components that are not readily available, and must be estimated or measured empirically.

Modeling user performance in teleoperation tasks can be particularly difficult.

This dissertation focuses on obtaining such models by performing several user studies designed to predict the teleoperator performance in response to multiple manual input devices and visual feedback mechanisms, as well as varying system latencies.

The overall framework for improving system performance is based on incorporating the derived, estimated, and measured component models into the implementation of the design optimization over a series of operations in the teleoperation system's required task set.

The contributions of this dissertation are as follows: 1) An identification of the factors limiting teleoperation system performance. 2) A framework for performing design optimization of teleoperated mobile robot speed and performance. 3) An evaluation of teleoperator performance with two types of manual interfaces and two types of visualization interfaces. 4) The development of a performance model for a path-following steering task under different latency conditions that indicates a possible mapping between performance under constant latency and variable latency. 5) The development and validation of a driver model capable of generating human-like steering inputs to a mobile robot.

CHAPTER I

Introduction

1.1 Motivation

Teleoperation is the process of controlling a device from a distance with a human operator. In mobile robotics, teleoperation (in direct and advanced forms) is the default mode of control for tasks requiring human judgment and decision-making skills. Today, unmanned ground vehicles (UGVs) are primarily controlled by teleoperation in military [44] and emergency-response [28] applications because of the highly unstructured nature of the missions. Even in cases where autonomous operation is possible, some operators prefer to remain in the control loop to quickly access and interpret the information gathered by the robot [42]. In the domain of mine rescue and recovery, teleoperation control has been ranked as a more important consideration than both autonomous navigation and self-localization and mapping [41]. Therefore, despite recent advances in robot autonomy, teleoperation continues to be a relevant mobile robot control modality for the foreseeable future.

In this work we distinguish between remote control, in which the operator can see the robot, and teleoperation, in which there is no line of sight. In both scenarios, commands to the robot must travel over a communications network. However, during teleoperation, all the feedback necessary to effectively control the device must also

be communicated over the network, limiting the amount and fidelity of information that can be provided to the teleoperator. For ease of implementation, operator commands and feedback signals generally travel over the same network, which can result in high network traffic and long delays. Additionally, mobile robots generally use wireless network protocols, often resulting in lower communications bandwidths and stochastic latency profiles.

This limited feedback channel results in multiple issues affecting perception and manipulation abilities in teleoperated robot tasks, identified by Chen *et al.* as [14]:

1. **Video field of view:** Viewing the remote scene through a camera feed strips users of peripheral vision and can lead to decreased spatial judgment and driving abilities.
2. **Operator sense of robot orientation:** Users need information about both the location and orientation of the robot in the remote environment, as well as the configuration of the robot itself (such as arm position and chassis pitch and roll).
3. **Camera viewpoint and frame of reference:** Some camera placement locations may lead to unnatural viewing angles for the user. Egocentric (first-person) and exocentric (third-person) camera views have benefits and drawbacks for different tasks.
4. **Depth perception:** Pure video feeds often lack a significant amount of depth-perception information, foreshortening objects in the remote environment. This effect is particularly pronounced for ground robots because of their low viewpoints.
5. **Video image quality:** Poor image fidelity or low frame rate can degrade an operator's ability to understand the remote scene, including spatial orientation

and target identification, and can lead to increased delays in the operator's reaction time.

6. **Time delay:** With increased system latency, operators often change their command input strategy from a continuous control to a "move and wait" style of control. Communication delays propagate throughout the operator-robot control loop, causing even longer total system delays.
7. **Switching between multiple camera views:** Switching between camera views may increase operator cognitive load as the user must rapidly switch between contexts. Additionally, change blindness may occur when operators switch views.
8. **Robot motion:** Motion of the robot can induce motion sickness in the operator, and vibrations caused by motion can make both interpreting visual displays and using manual interfaces more difficult.

These issues cause teleoperated tasks to be difficult and time-consuming to execute, and can lead to operator slips (failures caused by fallacies in unconscious processing, such as accidentally crashing a robot arm into the ground), and mistakes (failures caused by fallacies in conscious processing, such as taking a wrong turn during robot navigation) [13]. Ultimately, this results in decreased teleoperated system performance and reliability.

Many potential design solutions exist for mitigating these performance issues [14], each with expected benefits as well as associated costs. This raises the question: Which solutions should be implemented? The answer is not always clear, and can be highly system- and task-specific. It may not even be obvious which *issue* is the most pressing, let alone what solution should be implemented. Also, because robot subsystems are often interdependent, implementing a new design in one component

may have unintended consequences in another subsystem.

There is therefore a need for a systematic framework to evaluate the impact and cost-effectiveness of design choices for teleoperated UGVs. This dissertation introduces a framework for this purpose, and gives examples of its implementation. Because this methodology requires models of human teleoperator behavior that may not yet exist, this work details the development of several such models for use in the framework.

1.2 Contributions

This work has five main contributions applicable to the design and implementation of teleoperated mobile manipulators.

The first contribution is the identification of the factors limiting teleoperated system performance. These factors are present in all remotely-operated robots, and understanding the underlying causes of performance issues is a key step in overcoming such limitations. The factors are presented as elements of a closed-loop feedback system with a human operator acting as the controller. The second contribution is a framework for optimizing the speed and performance of teleoperated robots. This framework is based on systematically addressing the limiting factors by applying a design optimization approach to determine the most effective way to improve system performance and indicate the tradeoffs between performance and costs such as price, power usage, and reliability. Two examples are provided: one demonstrating the overall framework, and another showing how models required for the optimization can be developed. Both of these contributions are presented in Chapter II.

The third contribution, presented in Chapter III, is an evaluation of teleoperator performance under two types of manual input devices and two types of visual

feedback mechanisms, as measured by task completion time and accuracy. The results of this study indicate that under the conditions tested, a Master-Slave (MS) manual input method resulted in significantly better performance than a traditional gamepad, and that a Mixed Reality (MR) visualization interface resulted in slower task completion times than a video-only feedback scenario. Another finding was that users that did well on the tasks overall received less of a performance boost from the MS input device than did users who performed poorly on the tasks. Overall, these user tests reinforce the notion that a task's scenario, objectives, and operator must be carefully considered when designing teleoperation interfaces.

The fourth and fifth contributions, presented in Chapter IV, result from a set of user tests on teleoperated steering performance in the presence of system latency. First, a model of human teleoperation steering performance for a line-following task under constant and variable latency conditions was developed. This model shows that under the conditions tested, there is an equivalence between variable latency and constant latency. This equivalence was speed-independent and applied to both the objective performance measure and the teleoperator's subjective sense of the delay. If broadly applicable, this equivalence has the potential to greatly simplify the process of modeling human responses to latency. Second, a driver model for teleoperated steering tasks was developed. To our knowledge, this is the first steering model specifically designed for teleoperated mobile robot driving. This model can be used to generate human teleoperator-like steering commands under different latency conditions for use in UGV development, testing, and simulation.

1.3 Dissertation Organization

This dissertation is composed of three papers that have been submitted for journal publication, each in one chapter. For clarity, the word "paper" that appears in the journal manuscripts when used as a reference to itself has been replaced with the word "chapter" in this dissertation. Chapter II, titled "A System-Level Methodology for Design Optimization of Teleoperated Mobile Robot Speeds," enumerates the system factors limiting teleoperation performance and identifies their locations in the teleoperation feedback loop. This chapter also introduces the performance optimization framework. An example of the overall methodology and a sample human modeling procedure for a coin detection task are also presented. Chapter III, titled "Performance Evaluation of Visual and Manual UIs for Teleoperated Mobile Manipulators," presents a user study testing an MS manual interface and an MR visualization feedback interface for representative teleoperation tasks, and discusses an interpretation of the results. Chapter IV, titled "Modeling Teleoperated Mobile Robot Steering Behavior in the Presence of Latency," presents a second user study in which a simulated robot was steered around a virtual test track in the presence of varying amounts and types of latency using two different speeds and two different points of view. Path-following performance was evaluated for the different scenarios, and a model of a human teleoperator for steering tasks was developed that can be used to emulate a human driver for simulation and testing purposes. Finally, Chapter V summarizes the conclusions of this dissertation and discusses areas for future work.

CHAPTER II

A System-Level Methodology for Design Optimization of Teleoperated Mobile Robot Speeds

Portions of this chapter have been published in [63].

2.1 Introduction

Despite recent advances in robot autonomy, human-in-the-loop interaction (including direct and advanced forms of teleoperation) remains an integral part of many mobile robot tasks. In situations where it is hazardous or difficult for humans to be present, but which require human judgment and decision-making skills, the use of a human teleoperator is the only option. There are also cases, such as search-and-rescue tasks, in which human operators prefer to remain in the loop [42]. Additionally, Murphy *et al.* [41] have rated the importance of teleoperated control as higher than both autonomous navigation and self-localization and mapping for the use of mobile robots in mine rescue and recovery.

For robots to become more useful tools for humans in the future, the speed at which robot-assisted tasks can be completed must be increased. However, robot speed during teleoperation tasks is not generally limited by actuator speeds, but by other bottlenecks.

Consider a simplified teleoperation scenario, as shown in Fig. 2.1, in which a

mobile robot with maximum speed as governed by actuator saturation v_{sat} travels in a straight path at a given speed, taken as the magnitude of the robot's velocity, $v = |\vec{v}|$. Multiple sources contribute to the total system latency, with each individual delay given by δ_i. If an obstacle appears in the robot's path, the operator must recognize it and execute an avoidance maneuver before the robot collides with the obstacle. If the total distance needed to perform the obstacle avoidance ($v\Sigma_i\delta_i$) is greater than the maximum distance at which the obstacle can be detected (ℓ), then a collision is unavoidable. Rearranging terms, we can determine the following inequality providing a bound for maximum operational speed of the robot:

$$v \leq \min\left(\frac{\ell}{\Sigma_i\delta_i}, v_{sat}\right) \tag{2.1}$$

Thus, assuming the actuators are not saturated, if one wishes to increase the speed at which a particular teleoperated robot task can be executed then either the total system latency must be decreased or the detection distance must be increased. However, it is not always obvious how to most effectively choose between possible design options to achieve such a speed increase. Different components can have widely different costs of implementation, and can affect other system properties. It is thus desirable to have a systematic methodology for making such decisions. The contributions of this chapter are the framing of a design optimization-based methodology for efficiently increasing teleoperated robot speed for a given application, a categorization and discussion of the components contributing to teleoperated robotic system speed, an example of the application of this method, and examples of the type of tests that can be run to obtain the models required for this framework. The examples presented in this chapter focus on the type of teleoperation scenario described in Fig. 2.1. The framework described in Section 2.3 and the discussion of the limiting

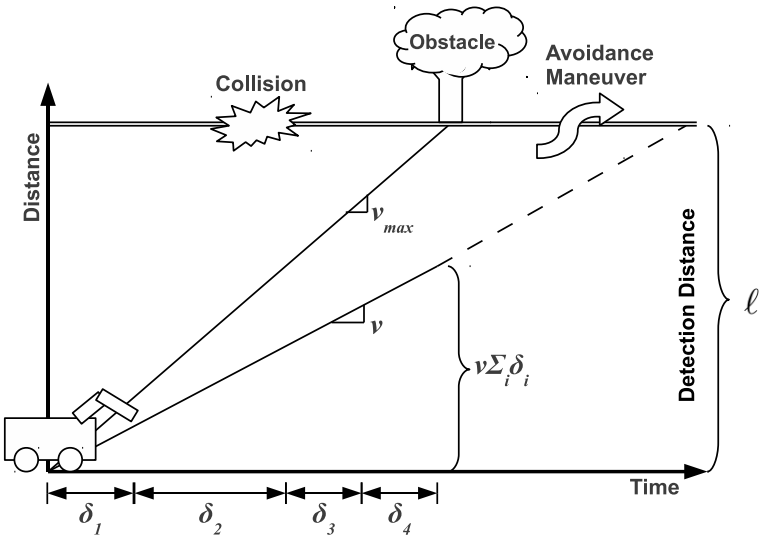


Figure 2.1: Schematic illustration of a simple teleoperation task. A robot travels at a constant speed v while an operator scans the robot's path and initiates an avoidance maneuver if an obstacle is detected. If not limited by actuator saturation, the maximum speed that can be successfully achieved in teleoperation is a function of the total time delay ($\Sigma_i \delta_i$) and the maximum distance at which an obstacle can be detected (ℓ).

factors in Section 2.2 are broad enough to be applied to any teleoperation scenario, including robot driving and mobile manipulation tasks.

2.2 Characterizing Limiting Factors

This section aims to broadly categorize the different factors that limit total system speed of teleoperated robots. Figure 2.2 shows a block diagram of a robot teleoperation system with the locations of some of the factors. These factors are applicable to the examples discussed in this chapter, as well as any other robot task that is performed with a human operator in the loop. To our knowledge, this is the first time these factors have been enumerated as a part of the teleoperation system feedback control loop. While it may not constitute a detailed list of potential limiting factors (such a list would be application-specific), designers can use this categorization during system design as a preliminary checklist for identifying the root causes that may limit system speed.

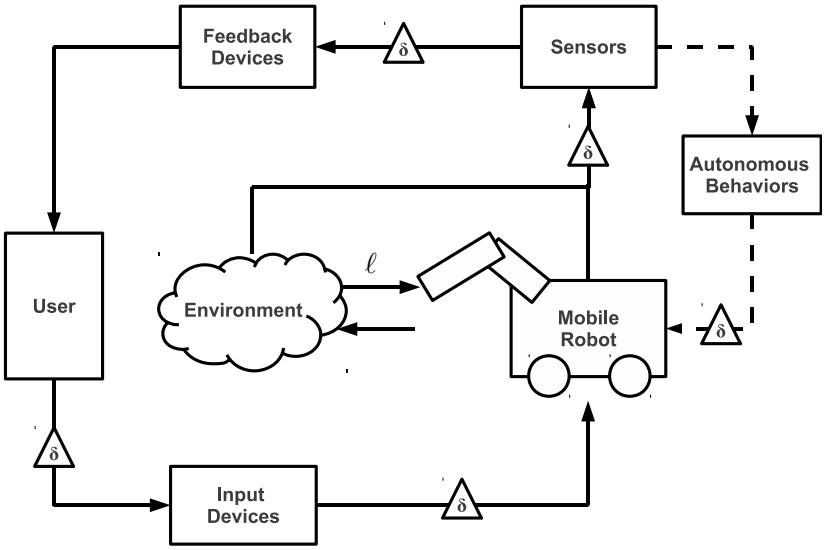


Figure 2.2: Block diagram of the teleoperation feedback loop. The location of some of the factors limiting teleoperation system performance are shown in the diagram. Each δ block represents a delay, while the detection distance is represented by ℓ. Delay blocks in the diagram may consist of multiple types of delay. Semi-autonomous behaviors may be present in some teleoperated systems. This block diagram facilitates a system-level analysis of the teleoperation process.

2.2.1 Actuator Saturation

Perhaps the most straightforward factor limiting the speed of teleoperated mobile robots is actuator saturation. Clearly, a robot cannot operate at a faster speed than its motors can accommodate. More powerful motors and motor drivers as well as batteries capable of a higher discharge current could all increase the maximum speed of the actuators, but often these components cost more, weigh more, and use more power. The concept of actuator saturation can be extended to more complex systems beyond individual motors and servos. For example, robot rollover during tight turns at high speeds could be considered a type of actuator saturation, as it stems from the robot's physical inability to perform the desired task. However, most teleoperated robots are not limited by the actuators themselves, but by the ability of the operator to safely navigate the robot through remote environments in the presence of system

latency.

2.2.2 Network Delays

Communication between a human operator and a mobile robot typically takes place over a network. Both operator control signals to the robot and feedback information from the robot generally travel over the same network for ease of implementation. Common communications networks used for mobile robotics include various standards of wireless Ethernet, Bluetooth, cellular communications networks, and satellite communications. Experimental results using wireless networks found round-trip delays on the order of a few milliseconds for different types of 802.11 networks, but tens of milliseconds for Bluetooth networks, and that delays in the wireless domain increase with distance, packet size, and interference [4]. In the wireless case, the stochastic nature of the transmission channel also leads to a significant variation in the time delays. Experimental results in the field have shown delays in video transmission from a teleoperated mobile robot on the order of seconds [9].

2.2.3 Sensing Delays

Sensing delay is defined as the delay associated with the raw data coming from a robot sensor (before any data is processed). For example, a charge-coupled device (CCD) camera must transfer a charge across a capacitor array until it reaches a storage element, which causes a small delay from the time that original image was recorded; however this delay is generally insignificant when compared to the rest of the latency in a robot system, and even compared to the frame rate of the camera. On the other hand, scan times for laser rangefinders can be up to hundreds of milliseconds [30], which can have a significant effect on teleoperation performance, especially considering that an object generally must be seen in multiple laser scans

before it can be properly recognized.

2.2.4 Processing Delays

Both on the robot and at the operator control unit (OCU), raw sensor data must be processed before it can be useful to the system. In general, powerful computers can minimize this delay, but low-power micro-controllers on a robot performing computationally expensive operations, or portable handheld devices used for operator user interface (UI) [33] may increase the impact of this type of delay. For example, a simple embedded video encoder on a robot can add a delay from tens to hundreds of milliseconds due to video compression alone [2]. Complex or inefficiently written software can increase this delay at both the robot and OCU.

2.2.5 Operator Delays

The category of operator delays includes everything in the "User" block in Fig. 2.2. The user must receive feedback from the robot (generally through a video feed), decide on an action to take, and then give the appropriate commands to the robot. There are both physical (the time taken to process and react to stimuli) and cognitive (the time taken by a user to determine the proper action to take) delays. Physical reaction time has been found to be around 400-500ms for a simple go/no-go image categorization task [57], though this could vary for other types of tasks. On the other hand, the cognitive delay for complicated tasks can depend heavily on the other types of delay within the system as well as the user interface itself. For example, it has been found that when facing a delay of more than 1 second, users adopt a "move-and-wait" control strategy for bilateral teleoperated manipulation tasks [54]. Additionally, it has been shown that users can sense latency elsewhere in the system that is higher than 10-20ms, and that system feedback delays higher than 170-320ms

can degrade performance for certain tasks related to robot teleoperation [14]. While operators can sometimes become accustomed to and adjust the input commands for some amount of feedback delay, if the delay time varies, such compensation is no longer possible [68].

2.2.6 Detection Distance

The detection distance, ℓ, of a system is defined as the maximum distance at which an obstacle can be detected, either by a human operator or an automated detection algorithm. This can depend on multiple factors, including the types of sensors used and the size of the object being detected. Laser scanning rangefinders have maximum detection distances up to 30 meters [30], but this range can be affected by object reflectivity and ambient lighting conditions. The detection distance for video feeds can also depend on video frame rate and video quality [15], as well as lighting, weather, and other environmental conditions.

2.3 Design Optimization for Teleoperated Robot Speed

Now that the key factors limiting teleoperation speed have been identified, we want to find a way to address each issue systematically in the context of a design problem. For example, we may not want to increase the detection distance if the system bottleneck actually lies within the network delay. A design optimization process can help determine the most effective way to increase the speed of a teleoperated robot as well as indicate cost/performance tradeoffs. However, the complexity of teleoperated robot systems makes such an optimization difficult. This section discusses how such an optimization can be applied to teleoperated robots, focusing on the particular challenge of optimizing such a complex system.

Prior work on design optimization for industrial robots [10, 70], unmanned ground

vehicles [8, 67], and mobile manipulators [34] generally aim to optimize robot performance (defined in multiple ways, including manipulator weight, reliability, and workspace size) using robot geometry and actuator selection as design variables. However, these optimizations do not consider the overall robot system performance when a human operator is placed in the control loop. An optimization analysis has been performed in [7] for determining the ideal collaboration level for a human-robot target recognition system, but this work did not focus on optimizing the hardware and system parameters. We believe that this work is the first to address robot hardware design optimization specifically for teleoperated robots.

It is important to understand that this optimization framework does not automate the design process. Rather, it shifts the robot designer's job from that of individual component selection to the more holistic task of understanding the design objectives and choosing appropriate objective function weights. Therefore, the selection of weights is difficult to generalize, and should be carefully considered for each individual design scenario. Comparing the results of the optimization using various weighting values can also give insights into the design space offered by the design choices as well as the trade-offs associated with each design objective.

The steps of this process can be framed as a multiobjective optimization problem:

1. Identify design objectives.
2. Enumerate the possible design variables and options.
3. Model the relationships between objective functions and design variables.
4. Assign trade-off weights and evaluate optimization.

Applying these steps to a teleoperated robot design problem presents some unique challenges, which are discussed in the following subsections. Additionally, while this

optimization process is broad enough that any design objectives could be considered, using speed as an objective complicates the process, primarily because the relationship between potential design variables and robot speed are not generally well-understood. Thus, the examples given in Sections 2.4 and 2.5 are provided to demonstrate how to estimate or empirically model the relationships required for optimization of teleoperated robot speeds.

2.3.1 Identify Design Objectives

The difficulty in designing a teleoperated robot system capable of high speeds lies in the inherent trade-off between performance and practicality. If one could always choose the best-performing components, the optimization would be trivial. Thus, the first step in the optimization framework is to determine which other factors, such as component cost, weight, power requirements, and working range, are significantly constraining the teleoperation system design. For teleoperated robots however, different tasks will have different objectives. For example, a robot may be required to drive to a object and pick it up. It is therefore helpful to break up the optimization into smaller sub-tasks and optimize each separately (in this case, analyze the driving and manipulation tasks individually), even though both tasks may use some of the same components.

Additionally, the large number of competing objectives in teleoperated robot design could result in a very complicated high-dimensional optimization problem. By converting some of the design objectives into constraints (*e.g.* rather than try to minimize power usage, we just set a maximum power bound), the search space of the optimization can be significantly simplified.

2.3.2 Enumerate Possible Design Variables and Options

The sheer number of design choices that have a potential impact on the design objectives can make this a daunting task. While generating an exhaustive list of design options may be impractical, a designer can generate a partial list by selecting a representative sample of viable design options. Similarly, design options that are continuous rather than discrete (the choice of video frame rate, for example) can be discretized for simplicity. If further resolution between similar options is desired, the optimization can always be run iteratively with designs of increasing similarity. When designing an objective function based on robot speed, the main factors limiting the speed of teleoperation listed in Section 2.2 are of particular importance, regardless of the specific task being optimized.

2.3.3 Model Relationships between Objective Functions and Design Variables

This is perhaps the most difficult step in setting up the optimization process for teleoperated robots, as it can often be difficult to find models that accurately predict the performance (as defined by the objective functions) of the available components. In the absence of such models, one may be able to determine empirical models by performing a series of tests on the components, but this requires that each design option be available for testing. An example of this model generation is presented in Section 2.5. Alternatively, one may attempt to define a model for each component using a combination of literature searches, manufacturer specifications, and estimation. Unfortunately, the accuracy of models made in this manner have no way of being verified without physical testing. However, if the results of the optimization based on first-pass estimated models can clearly eliminate some design choices, then perhaps physical testing can be reduced to a smaller set of design options.

2.3.4 Assign Trade-Off Weights and Evaluate the Optimization Function

There are multiple ways to construct a multiobjective optimization problem, but one of the simplest is that of the weighted average technique [43]. Given n objective functions $\{f_1(x),...,f_n(x)\}$ corresponding to different goals that depend on the state vector of decision variables (or design options) x, we can assign each one a weight w_i, and the compromise solution can be found by minimizing the multiobjective function [43]:

$$f(x) = \sum_{i=1}^{n} w_i f_i(x) \tag{2.2}$$

Example objective functions relevant to teleoperated mobile robots include speed, cost, weight, size, payload capacity, power usage, and reliability. If there is a particular objective function that should be maximized (*e.g.* robot speed), we simply set $w_i < 0$.

Because the competing objective functions necessary for teleoperated robot design have different units (dollars, m/s, kg, *etc*), it is difficult to assign weights that accurately express the relative importance of each objective. Normalizing each objective by comparing it to some baseline case can make choosing the weights more intuitive. However, even with normalized objectives, it's not clear what impact defining *e.g.* robot price as twice as important as robot weight has on the optimization result. Therefore, it is advisable to try the optimization over a variety of different trade-off weights at first to get an understanding of the design space and sensitivity of the optimization.

2.4 Example: Designing a Teleoperated Robot for High-Speed Operation

As a demonstration of the optimization described in Section 2.3, let us return to the scenario described in Section 2.1 and use it as a highly simplified example. This optimization framework could be applied to other teleoperation scenarios such as mobile manipulation by following the steps outlined in Section 2.3, keeping in mind the limiting factors discussed in Section 2.2.

2.4.1 Problem Definition and Design Objectives

Suppose the operator's only job is to monitor a video feed coming from the robot and press a button to perform an emergency avoidance maneuver if he/she perceives that there is a static obstacle in the path of the robot. We wish to maximize the speed at which the robot can run, and we want to take the system cost into consideration, but do not consider any other penalty factors such as weight. We also assume the robot's actuators are powerful enough to drive the robot at any speed we desire. Our optimal configuration is determined by minimizing the weighted average objective function:

$$f(x) = -w_{speed} f_{speed}(x) + w_{cost} f_{cost}(x) \tag{2.3}$$

which is a specific instance of Eq. 2.2.

We consider a simplified system in which video is recorded by a camera on the robot, compressed by an embedded video processor on the robot, and sent over a network to the operator control unit which decodes the video signal and displays the video on the screen. Once the user recognizes an obstacle, they must then press a physical button that sends a signal over the network to the robot indicating that the

robot should swerve immediately to avoid the obstacle. In this simplified example, we ignore the design of the swerving maneuver, and do not consider the risk of robot rollover or any other dynamic effects. This example serves as a "toy" problem to demonstrate the methodology, using linear approximations of models for system interactions and simplified hardware specifications.

2.4.2 Design Variables

We limit the design choices for this example to three types of cameras, three types of processors, and three types of network protocols, all summarized in Table 2.1. For the camera choices, 640x480, 1280x720, and 1920x1080pixels are standard video resolutions of increasing quality. We assume processors and networks are capable of providing their maxiumum bitrate continuously, though this may not be true for real systems. Additionally, we consider adding an enhanced user interface that adds a fixed amount of processing delay to the OCU, but reduces the cognitive delay of the user by 50%, and increases the distance at which the user can recognize the obstacle by 10%, but also adds a cost of $300. Finally, we fix the frame rate at 10 FPS, but allow the video resolution to vary continuously up to 1920x1080. The video size design variable is sampled as 100 equally-spaced points between 0 and 2073600px, which is the native size of 1920x1080 video.

2.4.3 Objective Function Modeling

Since a complete model of the whole system is not available, we must make some simplifying assumptions about how the design variables affect the objective function. Examples of the types of tests that could be used to validate these assumptions are discussed in Section 2.5. For now, we estimate models for the system behavior

Table 2.1: Summary of the available hardware selections used in the simplified example.

Camera	Resolution	Native Video Size	Cost
C_1	640x480	307200px	$50
C_2	1280x720	921600px	$100
C_3	1920x1080	2073600px	$500
Processor	**Max Bitrate**	**Max Latency**	**Cost**
P_1	10 Mbit/s	250ms	$25
P_2	10 Mbit/s	80ms	$100
P_3	20 Mbit/s	40ms	$200
Network	**Max Bitrate**	**Max Latency**	**Cost**
N_1	1 Mbit/s	500ms	$30
N_2	11 Mbit/s	1000ms	$50
N_3	54 Mbit/s	1000ms	$70
Enhanced UI	**Cognition Delay**	**Processing Delay**	**Cost**
off	500ms	0ms	$0
on	250ms	20ms	$300

under different design variables. These models represent first-pass approximations of component behavior for the purposes of demonstrating this example within the optimization framework. All model parameters are described in Table 2.2, and their relationships to the model are discussed in further detail below.

First, we estimate that the processing is dominated by the image compression routine, and that the processing delay is proportional to the amount of compression being done by the processor plus a baseline delay, with the max delay occurring at maximum compression of the video from its native video size in pixels (A_{native}):

$$\delta_P = \alpha\delta_P^{max} + (1-\alpha)\delta_P^{max}\left(1 - \frac{A}{A_{native}}\right) \tag{2.4}$$

where α is the proportion of the maximum delay that is inherent in the processor and is assumed to be 0.25 for all processors, A is the size of the compressed image in pixels, and δ_P^{max} is the maximum processor delay.

Table 2.2: Parameters used in the optimization, with descriptions and numerical ranges used in this example.

Parameter	Description	Value
A	Compressed Video Size	Varies up to A_{native}
A_{native}	Native Video Size	See Table 2.1
A_{native}^{max}	Max Native Video Size	$1920 \times 1080 = 2073600$px
α	Delay Inherent to Processor	25%
β	Video Throughput Scaling	5×10^{-7} Mbits/pixel
f	Video Frame Rate	$f = 10$ FPS
δ_{HC}	Human Cognition Delay	See Table 2.1
δ_{HR}	Human Reaction Time Delay	450ms
δ_P^{max}	Maximum Processing Delay	See Table 2.1
δ_{UI}	Processing Delay due to UI	See Table 2.1
δ_{VB}	Video Buffer Delay	$\frac{A}{A_{native}^{max}}\delta_{VB}^{max} + \delta_{UI}$
δ_{VB}^{max}	Max Video Buffer Delay	100ms
γ_{UI}	UI Range Scaling Factor	$\begin{cases} 1.0 \text{ Predictive UI} \\ 1.1 \text{ No Predictive UI} \end{cases}$
ℓ^{max}	Max Detection Distance	10m
R	Video Throughput	$R = \beta f a$
R_N^{max}	Max Network Throughput	See Table 2.1

Let us also assume that the processor uses a fixed compression ratio, and that the throughput (R) needed to transmit video is a linear function of the compressed video size:

$$R = \beta f A \tag{2.5}$$

where β is assumed to be 5×10^{-7} Mbits/pixel, and f is 10 FPS.

Assume also that the network delay is constant and can be estimated as a linear function of the network bandwidth in use, up to some maximum delay for the network. Assuming the majority of the network bandwidth is used to transmit video, and that the delay is symmetric for both the robot-to-human, and the human-to-robot directions, the one-way network delay is thus estimated by:

$$\delta_N = \frac{R}{R_N^{max}}\delta_N^{max} \tag{2.6}$$

We further assume that the time taken to decode the video at the OCU is a linear function of video size, up to a maximum of $\delta_{VB}^{max} = 100$ms for 1080p video, plus any

delay from an enhanced UI, if used. Thus, we estimate the video buffer delay at the OCU to be:

$$\delta_{VB} = \frac{A}{A^{max}_{native}} \delta^{max}_{VB} + \delta_{UI} \tag{2.7}$$

where $\delta_{UI} = 20$ms if the enhanced UI is used, and 0ms if not.

Assume the delay due to human cognition δ_{HC} to be 500ms without the enhanced user interface, and by 250ms with the enhanced interface. The reaction time of the operator δ_{HR} is taken as 450ms. The total operator delay is given by $\delta_O = \delta_{HC} + \delta_{HR}$. User commands to the robot are subject to the same delay as that associated with video transmission, δ_N. Assume all other system delays are negligible.

Finally, assume that distance at which the user can recognize an obstacle on the video screen is proportional to the video size, up to $\ell^{max} = 10$m for 1920x1080 video, and can be increased by adding the enhanced UI.

$$\ell = \left(\ell^{max} \frac{A}{A^{max}_{native}} \right) \gamma_{UI} \tag{2.8}$$

where γ_{UI} is 1 without the enhanced UI and 1.1 with it implemented.

The objective function for speed is given by the maximum possible speed, which depends on the detection distance and the delays:

$$f_{speed}(x) = v_{max} = \frac{\ell}{\delta_P + 2\delta_N + \delta_O} \tag{2.9}$$

and the objective function for cost is simply the sum of the costs of the individual components.

Table 2.3: Comparison of the optimal, fastest, and cheapest hardware combinations. The optimal configuration was determined using scaled weights of $\overline{w}_{cost} = 1$ and $\overline{w}_{speed} = 1.2$.

System Type	Camera	Processor	Network	UI	Video Size	Speed	Cost
Fastest	C_3	P_3	N_3	on	2073600px	9.1m/s	\$1070
Cheapest	C_1	P_1	N_1	off	188509px	0.4m/s	\$105
Nominal Compromise	C_2	P_2	N_2	on	921600px	3.0m/s	\$550
Optimal	$\boldsymbol{C_3}$	$\boldsymbol{P_1}$	$\boldsymbol{N_3}$	**off**	**1989818px**	**6.5m/s**	**\$595**

2.4.4 Optimization Weights and Evaluation

Because our individual objective functions are in different units (m/s and dollars), it is difficult to choose weights for the compromise solution that accurately reflect the trade-off we wish to achieve. To better choose values for the weights, we can scale them by the best-case scenario for speed, and worst-case scenario for cost (see Table 2.3), thus using weights of $\overline{w}_{cost} = \frac{w_{cost}}{max(f_{cost}(x))}$ and $\overline{w}_{speed} = \frac{w_{speed}}{max\big(f_{speed}(x)\big)}$.

Suppose that in this example, the designers decide that the multiobjective function should consider the robot speed 1.2 times more important than the system price when the weights are scaled as described above. Thus, we select weights of $\overline{w}_{cost} = 1$ and $\overline{w}_{speed} = 1.2$. Performing the optimization using an exhaustive search over all of the design variable options, we arrive at the optimal system described in Table 2.3.

We can compare the optimal system against the slowest (also lowest-cost), and fastest systems (highest-cost), as well as a "nominal compromise" system, in which the mid-price, mid-performance components (C_2, P_2, and N_2) were selected, and the improved UI was included. In this example, the optimized system is able to achieve a speed more than 15 times higher than the cheapest system. The optimized system's speed is more than double that of the nominal compromise system while the system cost was increased by less than 10%. While the simplified hardware specifications of this toy problem lead to possibly unrealistic predicted performance improvements, this example demonstrates the utility of the optimization process in determing design

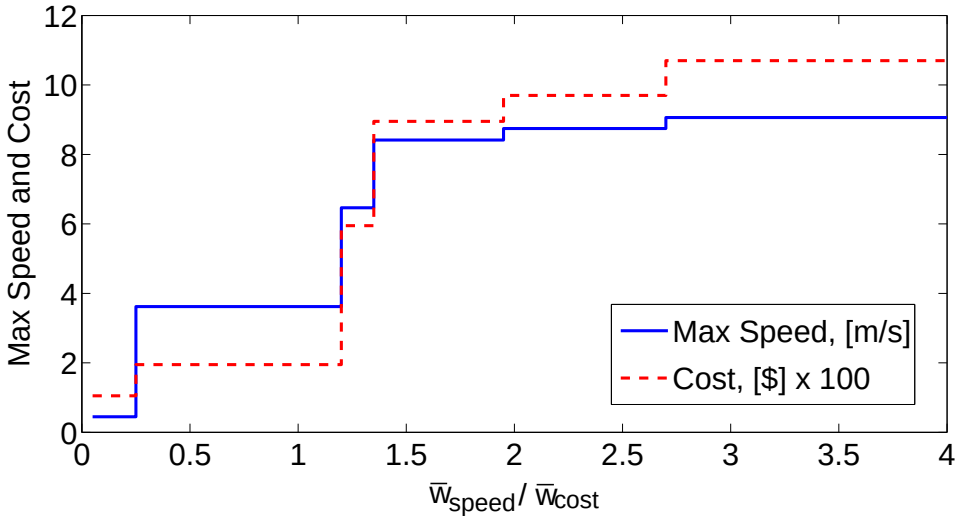


Figure 2.3: The optimized system speeds and prices for different ratios between the weighting values $\overline{w}_{speed}$ and $\overline{w}_{cost}$. Selecting a weighting ratio of $\overline{w}_{speed}/\overline{w}_{cost} = 1.2$ results in the optimized system design described in Table 2.3. This plot also shows the sensitivity of the optimization results to the weighting ratio.

solutions that may not seem obvious to human designers.

More than half of the cost of the nominal compromise solution is due to the use of the enhanced UI, however the optimized system does not include this feature. The optimization has revealed that for our choice of multiobjective function weights, the cost of the UI relative to the speed increase it afforded was too high to justify its use, and money would be better spent elsewhere in the system.

It is also of interest to see how the optimal solution would change if we vary relative weights of the speed and cost objective functions in the multiobjective function (see Fig. 2.3). The highest gain in speed comes with the first change in hardware from the cheapest option, and subsequent speed increases are less dramatic.

2.5 Example: Modeling System Performance with Experimental Tests

While the previous analysis is useful for providing a simple example of the optimization methodology, it relies on estimated models to describe the relationships

between the hardware and the factors limiting teleoperation. Performing some basic experimental tests can provide another interesting example in which to study the framework. This is not meant to be an exhaustive exercise, but rather a demonstration of the types of tests that could be performed to help inform teleoperation system designers about the optimization framework and how it fits into the design process. The aim of this example is to demonstrate how one can construct useful models of a complicated physical teleoperation system that could be used in an optimization analysis.

In this example, we seek to create a model of system performance of a detection task for an off-the-shelf robot with adjustable speed and video quality. In this simplified task, a teleoperator navigates a robot down a straight hallway looking for a coin placed randomly on the ground and attempts to stop the robot as quickly as possible after finding the coin. For the purpose of this model, we define"performance" to be the distance in front of the coin that the robot is able to stop.

2.5.1 Robot Hardware and Software

A user teleoperates the robot (SuperDroid skid-steer robot chassis) in a straight line through a hallway, using a gamepad (Logitech Cordless Rumblepad 2) to provide input to the chassis. A simplified control scheme is used wherein the user presses a gamepad button to command a constant speed to the robot, and releases the button to command the robot to stop. Steering commands are issued to the robot by the user via the gamepad's thumb stick only to keep it moving in a straight line down the test track.

Low-level commands are communicated to the robot via a wired Ethernet connection from a laptop mounted on the robot chassis. A camera (Microsoft LifeCam Cinema, model 1393), connected via USB to the robot laptop is mounted on a

robotic arm, which remains fixed throughout the tests. This on-board laptop receives commands from and provides video to the operator control unit (OCU) using the Lightweight Communications and Marshalling (LCM) libraries [31], over a tethered Ethernet connection using multicast User Datagram Protocol (UDP). The video is transmitted as a series of JPEG images, with the video resolution determined by hardware-based Motion JPEG video encoding. The tether is long enough that the robot can travel the length of the hallway and the user can be placed out of line of sight of the robot.

2.5.2 Experimental Procedure

All teleoperation tasks were performed by a single expert user without line of sight to the robot. More users would be required if the model needed more generalizable results.

For each test, a researcher placed a coin on the ground randomly within the boundaries of a 20x1.25m test track, outside the initial visible range of the robot. The robot was then driven down the test track and the operator commanded the robot to stop once a coin was identified. The distance from the front of the robot to the center of the coin was then measured manually.

First, the robot was driven at a very low speed (just enough to overcome the stiction of the robot's motors), while the teleoperator searched for three different US coin types (a penny, a quarter, and a Sacajawea dollar, see Fig. 2.5) using video feeds with resolutions of 1280x720px, 640x480px, and 320x280px, all scaled without interpolation or stretching to have a width of 1280 pixels on a 25" (63.5cm) monitor with a resolution of 1920x1200px. Five different tests were performed for each combination of coin type and video resolution. Because the robot's speed was very low for these tests, the distance the robot traveled after the teleoperator recognized the

coin was negligible, so the results of this test indicate the detection distance for each of the coins under the different video resolutions.

Next, tests at higher speeds were performed using the same three video formats, operating the robot at speeds of approximately 0.3, 0.6, and 0.9 m/s in the same manner as the previous tests. Note that 0.9 m/s was the highest achievable speed for this robot chassis due to actuator saturation. This test was repeated ten times for each speed/resolution pairing. All tests were performed with a video frame rate of 10 FPS, using a penny as the detection target.

To try to isolate the effects of controlling the robot via teleoperation as opposed to with a line of sight view, the line of sight stopping distance (LoSSD) was measured for the three different robot speeds listed previously. The robot was driven in a straight line while the operator stood next to the test track at a designated stopping line with an unobstructed line of sight to the robot. When the user perceived that the front of the robot passed over the stopping line, the robot was commanded to stop. The distance that the robot took to stop was then measured. The user was positioned as close to the stopping line as possible to prevent parallax errors. Ten trials were performed at each speed setting. The LoSSD captures the impact of the inertial properties of the robot as well as the system delays due to the network (one-way from the user to the robot), processor, and physical reaction of the user.

2.5.3 Results and Discussion

Figure 2.6 shows boxplots of the detection distance (ℓ) of each of the three types of coins for three different image resolutions. During the tests it became clear that the detection distance was highly influenced by lighting and carpet patterns, and it was particularly difficult to see the silver-colored quarter on the blue-speckled carpet. The large, gold-colored dollar coin was the easiest to see, and even though

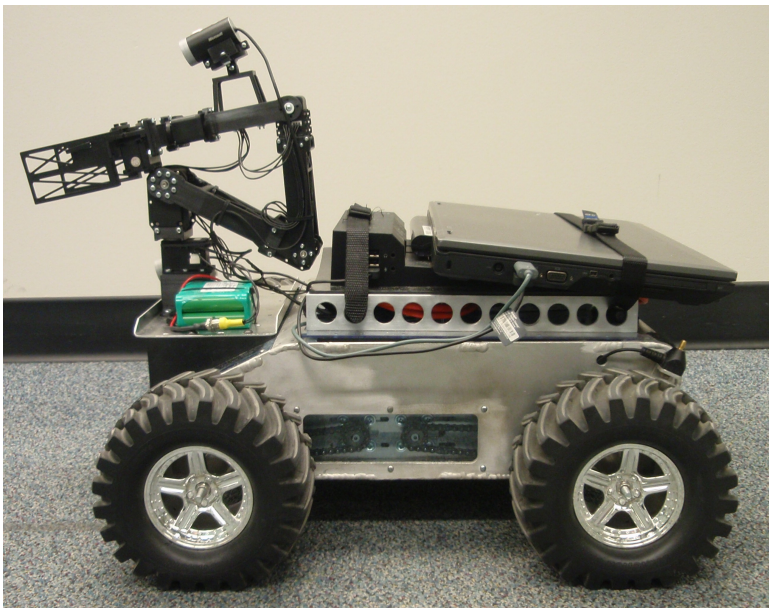


Figure 2.4: Photograph of the skid-steer robot used for the experiments. The robot arm, and thus the camera position, was locked in a stable configuration throughout the duration of the tests.

Figure 2.5: Photograph of the three types of US coins used in the experiments. From left to right, a penny, quarter, and Sacajawea dollar. In addition to the size differences between the coins, the blue coloring of the carpet makes the quarter more challenging to identify during teleoperation.

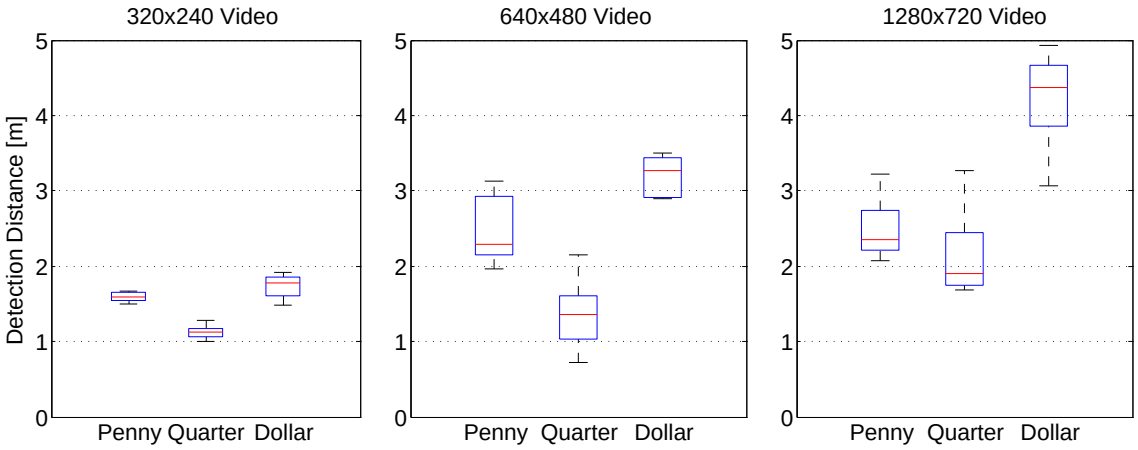


Figure 2.6: Boxplot showing the detection distance, ℓ, of three types of US coins for the three video resolutions tested. For all boxplots in this chapter, the center line of each box represents the data median, while the edges of the box correspond to the 25th and 75th percentiles (the innerquartile range, IQR), and the whiskers extend to the most extreme data points within 1.5 IQR of the 25th and 75th percentiles. Data points outside this range are considered outliers [24]. Against the blue carpeted floor, the large gold-colored Sacajawea dollar coin was by far the easiest to detect. The detection distance of all coin types increases from a 320x240 resolution to 640x480, though for the penny and dollar coins there may be diminishing returns when moving to the 1280x720 video resolution.

the penny is smaller than the quarter, its copper color stood out more against the floor. Also note the low variability of the low-resolution tests. This may indicate that the nuances of lighting and floor pattern play less of a role at the close distances required to see the coins at the low resolutions.

Additionally, this plot shows that the detection distances for the 640x480 and 1280x720 resolutions are both higher than that of the lowest resolution tested, but are relatively close to one another. A key takeaway from this is that for this task there may be diminishing returns for using higher resolution video feeds. Thus, full 1280x720 resolution video may not be necessary for this application and the bandwidth that could be saved by dropping down to 640x480 may be put to better use elsewhere.

Figure 2.7 shows the Line of Sight Stopping Distance (LoSSD) of the robot at the three speeds tested. As expected, the stopping distance and distance variance

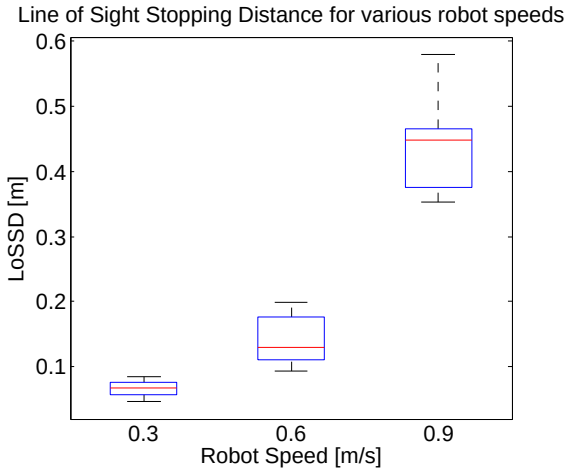


Figure 2.7: A boxplot showing the Line of Sight Stopping Distance (LoSSD) of the robot for various robot speeds. The LoSSD is a lumped parameter that captures the inertial properties of the robot, processing and network delays in the system, as well as user physical reaction time. Both the absolute distance and the variance of the LoSSD increase with robot speed.

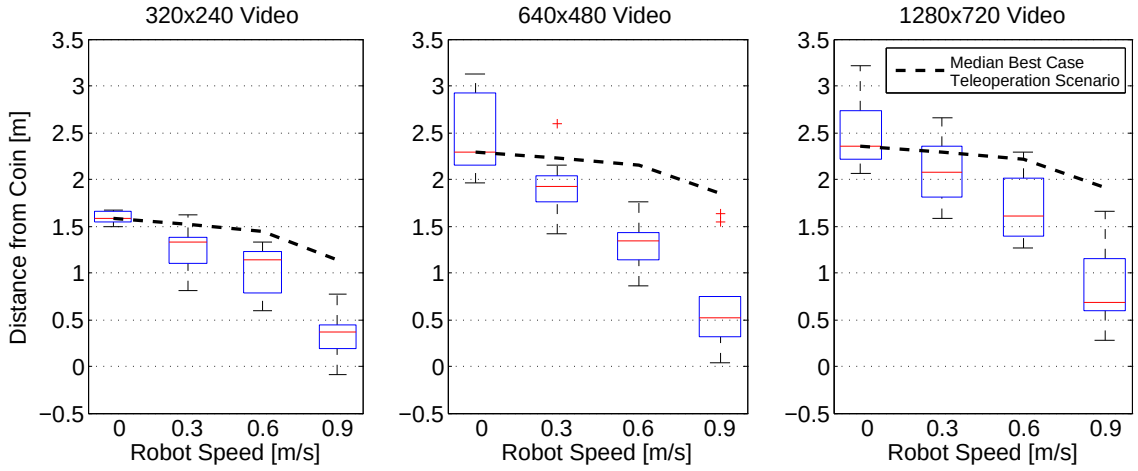


Figure 2.8: Boxplots showing the distance from the robot to the coin after detection and braking versus the speed of the robot, for three video resolutions. The median best-case teleoperation scenarios are shown for comparison. They are defined by subtracting the LoSSD from the detection distance, and represent the predicted median performance limit for each scenario. Similar to the results of detection distance, the performance of the lowest video resolution is much lower than that of the other two video formats, but there is little distinction between the two higher video resolutions. All tests were performed with a video frame rate of 10 FPS, detecting a penny.

increase with speed. If the LoSSD only depended on delays, we would expect it to have a linear relationship with speed. The trend shown in Fig. 2.7 indicates that the inertial effects are significantly contributing to the stopping distance.

Figure 2.8 shows a plot of the distance between the robot and the coin after detection and braking at three different robot speeds for various video resolutions. The distance generally decreases with increasing robot speed. Though there is more variation in this data than that of the detection distance tests, similar trends are observed. For each speed, the lowest video resolution is generally worse than the two higher resolutions, but there is less of a difference between the distances observed when teleoperating with 640x480 vs. 1280x720 video streams.

The cause of the increase in variation between tests of the same type is likely due to several factors: First, during these tests, the robot is moving fast enough that while the user is scanning the video feed for targets, if a coin comes into view while the user is looking at a different part of the image, there could be a significant amount of distance covered before the user can register that a coin is within sight. Similarly, any variation in reaction or decision-making time on the user's part has a higher impact on the variation of the results at higher speeds than when the robot is going very slowly.

The median best case teleoperation scenario for each trial is determined by subtracting the LoSSD from the detection distance for each data point. There appears to be a higher discrepancy between the best case scenarios for the medium-resolution video. Because the LoSSD measurement captures all of the delays in the system except the robot-to-user network delay (which should not be higher for lower image areas) and user cognitive delay, this data indicates that the cognitive load is highest for the 640x480 video, and the user must do more work to identify the coin in

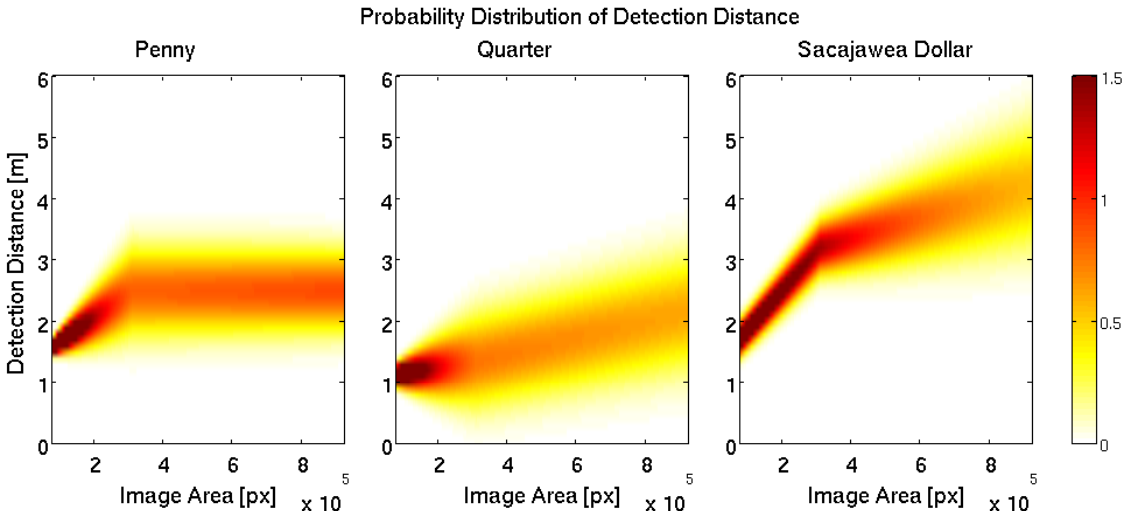


Figure 2.9: The extrapolated probability density functions of coin detection distance (ℓ) conditioned on video image area. The underlying probability distribution at each image area is assumed to be Gaussian. In all cases, the distribution has a higher variance for larger image areas.

the medium resolution scenario, despite the overall performance of this video being comparable to that of the higher-resolution video. Because the low-resolution video has such a low detection distance, this "instant recognition" may lessen the cognitive load felt by the user, despite the overall reduced performance.

2.5.4 Performance Modeling

Using the data presented in Figs. 2.6 and 2.8 and making some assumptions about the underlying probability distributions of the test data, we can determine a model for system performance under varying conditions that were not explicitly tested. For the purposes of developing such models, the test data at each condition is assumed to have a Gaussian probability distribution, characterized by a mean and standard deviation. To predict detection distance for image areas between the explicitly measured test points, these mean and standard deviation parameters were linearly interpolated as a function of image area between the points of experimental data for each coin.

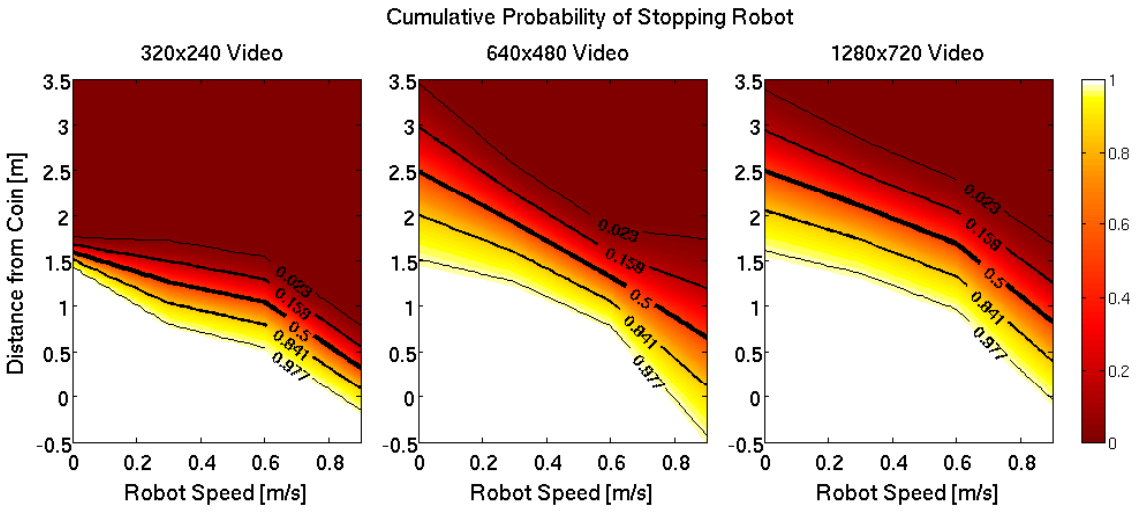


Figure 2.10: A model showing the cumulative probability of successfully stopping the robot at a given distance from a penny over different robot speeds. Contour lines represent one standard deviation from the median distance. The underlying probability distribution at each image area is assumed to be Gaussian.

Figure 2.9 shows a model of detection distance for each coin, represented by a normal probability distribution function, throughout the range of video image areas tested. This model indicates that there are diminishing returns for increasing image area for the penny and dollar coins, but the expected value of the detection distance for the quarter increases linearly over the range of image areas tested, which is consistent with the assumption made in Eq. 2.5 for the optimization example presented in Section 2.4. This type of empirical model could be used to replace the estimated model described by Eq. 2.5 for a given UI. It could be used in its current probabilistic state for simulations, or if a simpler model is desired it could be integrated over the detection distance to create a cumulative distribution function describing the probability of detecting a coin at a given distance.

Figure 2.10 depicts a model of system performance for this specific task (defined as the distance between the stopped robot and the coin) by showing the probability of achieving a performance specification over the range of speeds tested using the

penny as the detection object. These models were determined by integrating the probability distribution function at each video resolution over distance to obtain the cumulative distribution function. For example, if we desire the robot to have a 98% probability of stopping at least 1 meter in front of the penny, the maximum speeds allowable for video feeds of size 320x240, 640x480, and 1280x720 pixels are 0.2, 0.5, and 0.6 m/s, respectively. Because this model represents a performance metric, it is possible to use this empirical model of $f_{speed}(x)$ directly in an optimization that minimizes Eq. 2.3 without having to estimate how the various design parameters affect the system delays directly. The drawback to this approach is that because the model represents an aggregate of the system delays and detection distance, it would be difficult to use it to predict performance for any design options that have not already been tested explicitly.

2.6 Conclusions and Future Work

This chapter presents a new application of design optimization for the purpose of optimizing the speed of teleoperated robots. The contributions of this chapter include a discussion of the challenges specific to the application of design optimization for human-in-the-loop tasks in Section 2.3, and a categorization of the common factors limiting robot speed in Section 2.2. A simplified example demonstrating the optimization method has been worked through in Section 2.4. While it may be difficult to find real-world data that would be needed to perform this analysis, examples of some experimental tests that could be used to build the kinds of models necessary to use this methodology are discussed in Section 2.5.

This design framework offers several advantages over design methods that focus on individual sub-systems. First, as shown in the example in Section 2.4, the results of

the optimization may suggest a non-obvious compromise solution that uses resources more effectively than would other compromise solutions. Second, by shifting the focus of the designers from the individual design choices to the choice of weights for the objective functions, designers will get a more holistic system-focused view of the teleoperation system and the tasks it must accomplish.

The application of this optimization technique assumes that the designer has knowledge about (or can obtain a reasonable estimation for) the relationships between the design variables and the objective function. In practice, these models may be difficult to obtain, even in the lumped-form described in Section 2.5. Additionally, the delays used in these analyses were considered constant relative to factors such as robot distance from the communications hub, and stochastic effects were not considered. However, this framework can provide a first-pass evaluation of different hardware choices using some estimates of unknown parameters. Choosing relative weights for each objective function is not necessarily a straightforward task, and even small changes in their relative weights could have a significant impact on the design solution. A sensitivity or uncertainty analysis over both the model parameter space and the optimization function weights could be used to understand the robustness of the optimization.

The implementation of the framework discussed in this work relies on an exhaustive search over the design space, which may become impractical as the number of design variables increases. Additionally, using speed as an objective function could be problematic for more complicated scenarios, as it depends nonlinearly on system delays. One workaround for this could be to use the multiobjective function to minimize delay and maximize detection distance, but this does not fully capture speed as the end-objective.

The examples presented in this work focused on very simple driving tasks. More work needs to be done to understand how this methodology could be applied to more complicated systems involving more realistic tasks such as mobile manipulation. One possible way to expand the scope of this analysis would be to set up the teleoperation model as a standard block diagram, which could then be analyzed by systems theory if the necessary models for each block could be measured or derived. It may then be possible to concatenate models for different types of simple operations (such as driving, braking, turning, and manipulator placement) to scale this approach to more complex mission-level tasks to get an understanding of overall system performance beyond what can be modeled with a simple block diagram.

CHAPTER III

Performance Evaluation of Visual and Manual UIs for Teleoperated Mobile Manipulators

Preliminary results related to this chapter were published in [62] and [64], and the design of the User Interface used in these experiments is published in [65].

3.1 Introduction

While many rote tasks can be performed by robots without human intervention, direct teleoperation remains the default mode of control for mobile manipulators when human judgment and decision-making skills are required for a task. However, many of the issues associated with teleoperation, including (but not limited to) field of view, user understanding of robot state, depth perception, time delays, and motion effects [14] are still prevalent in currently-fielded teleoperated robot systems. Such issues result in teleoperation missions that are slow and laborious to complete, even with significant user expertise. In time-critical life-or-death situations, such as urban search-and-rescue following a terrorist attack or natural disaster, there is a desperate need to perform teleoperated missions quickly and with few mistakes.

Well-designed User Interfaces (UIs) can help mitigate some of the issues that limit teleoperation speed and performance, but robot designers often fail to consider the importance of UI when designing teleoperation systems. In particular, there are few

UIs that are well-suited to both mobile robot navigation and path-planning as well as manipulator arm control, even though both control modes are necessary to complete many teleoperated robot tasks.

Both the method by which the user inputs commands to the robot and the way in which the user receives feedback from the robot are parts of the UI of the teleoperation system. In this work, we limit the scope of discussion to manual input devices (as opposed to voice commands, gestural input, or other input methods) and visual feedback mechanisms (as opposed to tactile, audio, or other types of feedback). Manual and visual interface components are often developed independently of one another, and their individual contributions to overall system performance are not well-understood. Finally, many user studies do not take into account the skill-levels of users when making recommendations.

The contributions of this chapter stem from a 22-subject user study designed to determine the effects of UI visualizations and manual input methods on user performance, objectively measured by task completion time and accuracy. This study resulted in four key findings: 1) As expected, teleoperators performed significantly better with a Master-Slave (MS) manual interface than with a traditional computer gamepad. 2) In contrast to expectations, performance decreased when users switched from a video-only visualization to a Mixed Reality (MR) visualization. 3) Users that did well on the tasks overall derived less benefit from the MS interface than did users that performed poorly overall. 4) A regression model was derived showing that the two most important factors predicting task performance in this study are the user's prior experience with video games and the type of manual input used. Additionally, the results of the user study are used to make general UI design recommendations for teleoperated mobile manipulation tasks.

3.2 Background

This section provides a brief overview of previous work that has been performed in the areas of Mixed Reality (MR) and Master-Slave (MS) interfaces, as well as a description of our prior work with similar UIs.

3.2.1 Mixed Reality

Mixed Reality displays are a class of technologies in which real-time video information is blended with computer graphics, creating an enhanced visualization. Mixed Reality displays are classified on a "virtuality continuum," defined by Milgram [38], which extends from the pure video displays through Augmented Reality (AR) (primarily video with some virtual elements), Augmented Virtuality (AV) (primarily virtual renderings with some video elements), to complete Virtual Reality (VR). Presently, MR is used in a variety of applications, including UIs, training programs, video games, sports entertainment, advertising campaigns, and scientific visualization [61].

Mixed Reality is an attractive tool for teleoperated mobile robot UIs, as it can help improve communication between robots and users with an intuitive spatial and visual dialogue [26], potentially reducing operator slips and mistakes (Defined by Carlson and Murphy [13] as failures caused by unconscious and conscious human processing errors, respectively) and training times. Note that other human factors researchers can define alternate differentiations between error types. For example, Reason [49] defines slips, lapses, trips and fumbles as execution failures, and categorizes mistakes as failures resulting from planning or problem solving issues. Additionally, MR can be used to communicate auxiliary sensor information to the user, such as LIDAR scans, which may be difficult for the operator to parse without accompanying video

data [17].

Many MR interfaces for both static manipulators and mobile robots have been previously developed and tested. The ARGOS system developed by Milgram *et al.* [40] was one of the first implementations of AR for human-robot interaction (HRI). In this work, virtual indicators were superimposed on a stereographic video feed of an industrial-style manipulator, showing spatial relationships between objects on the screen. These markers included virtual pointers (indicating the position of a point), virtual tape-measures (indicating the distance between points) and other geometric indicators [39]. It was found that such overlays had a positive effect on user performance [52].

Chintamani *et al.* [16] used an AR interface to assist teleoperators controlling a manipulator arm with alignment tasks. It was found that the implementation of AR cues in the form of virtual coordinate axes significantly reduced errors and end effector path distance during teleoperation.

Green *et al.* [27] tested a multi-modal interface in which the operator was shown an exocentric (third-person) AR view of the robot's workspace and communicated with the robot via speech and gesture commands to teleoperate and path-plan a virtual robot's trajectory around a maze. It was found that this interface resulted in a slower task completion time, but higher task completion percentage and fewer "close calls" with collisions. Additionally, Nielsen [45] developed an Augmented Virtuality system for robot navigation in which the robot's 2D video feed is projected into a 3D virtual map of the robot workspace, which also shows the robot pose. Thus, all relevant information was integrated into a single perspective. It was found that such an interface improved user performance in both navigation and exploration tasks. Collett and MacDonald [17] have developed an AR interface to assist robot

programmers during debugging by providing visualizations of the robot's world view.

While MR interfaces for robotics have generally focused on Unmanned Ground Vehicle (UGV) navigation [45, 19, 3] and path planning [25], or the use of immobile industrial-style manipulator arms [16, 40], many UGV teleoperation tasks require both chassis navigation and manipulator arm control, and there is a knowledge gap in mobile robot UIs that are well-suited for both tasks. Because UIs used for static manipulators may not be suitable for tasks involving mobile robot navigation, and vice versa, creation of such a UI is not trivial. For example, fixed cameras providing third-person views are often used in industrial settings when controlling robotic arms, but this setup is not possible with an arm mounted on a mobile chassis.

3.2.2 Master-Slave Interfaces

A MS interface is a method of controlling a teleoperated robot arm (slave) by manually orienting a control arm (master) to the desired position. Such position control can be classified as direct, in which each joint is controlled individually, or resolved, in which a reference point is mapped from the master to the slave, but not necessarily with the same joint angles [23]. Position control can also be either unilateral, in which no force information is communicated back to the master arm, or bilateral, in which forces felt by the slave arm are reflected back to the master manipulator and thus to the user [23]. In contrast, manual interfaces like joysticks and gamepads afford only rate (velocity) control, which can lead to a slower work pace, and are not an effective form of control for tasks requiring high dexterity [23]. Direct rate control can also lead to safety and efficiency problems when the operator fails to understand the mappings between the control inputs and the movement outputs [23].

Master-Slave teleoperation has a rich history dating back to the 1940s [29], but

much of that work has focused on immobile robot arms, not mobile manipulators. Additionally, many of the interesting control problems in the field stem from the issues associated with maintaining stability and telepresence despite the existence of communication delays during bilateral teleoperation [29], so the literature tends to focus more on force-reflective systems.

In the past decade, several researchers have developed systems implementing unilateral MS control for mobile manipulators. Rogers [50] used hobby radio control components to develop a low-cost teleoperated MS controller without force feedback for use in under-vehicle inspection tasks for military applications. Suganuma *et al.* [56] used a system of three unilateral position controllers instead of bilateral feedback to communicate force information back to an operator for a gripper system mounted on a walking robot.

3.2.3 Previous Work

Our previous work describes the implementation [64] and pilot testing [62] of a visualization interface that used an MR display to enhance the visual feedback presented to the user during a simple teleoperated mobile manipulation task. In this task, users were required to navigate around a test arena, pick up a small box and then deposit it into a goal bin, and then repeat the operation for a second box. While not enough user tests were performed to obtain statistically significant results, the qualitative tests provided valuable information for the design and implementation of future UIs. Survey results indicated that users felt more present in the remote workspace with the MR visualizations, especially when the both AR and VR scenes were available. Thus, the MR interface showed promise warranting future development and testing. In these tests, a gamepad was used for both arm manipulation and robot navigation. However, users generally found it very difficult to mentally

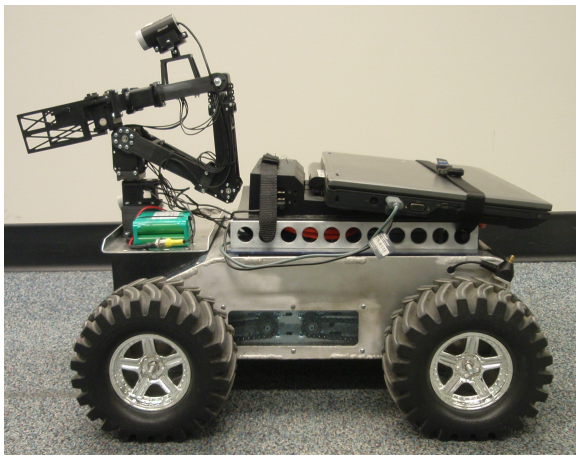


Figure 3.1: A photograph of the robot platform. A custom-built five DoF robot arm is mounted on a skid-steer robot chassis. An HD camera is statically affixed to the third link of the arm. The robot's processor is a laptop computer mounted on the back of the chassis.

map commands from the gamepad to the robot arm.

3.3 Task Setup

The user tests in this study are performed using a custom robot system in a specially-designed robot arena. The teleoperated robot system consists of a chassis and a manipulator arm, and is controlled by the user with the help of a desktop computer serving as an Operator Control Unit (OCU).

3.3.1 Robot System

The skid-steer chassis is a SuperDroid 4WD All Terrain Robot Kit with upgraded motors and motor drivers (see Fig. 3.1). A laptop computer running Ubuntu Linux is mounted on the chassis, which functions as the robot's processor. Heading and velocity commands are communicated via wireless Ethernet from the OCU to a Gumstix single-board computer in the chassis, which in turn sends low-level torque commands to the wheels' motor drivers. An HD camera (Microsoft LifeCam Cinema, model 1393) is mounted on the front of the chassis that is used for robot localization, but does not provide video for the user.

The manipulator is a five degree-of-freedom arm with a linearly-reciprocating gripper, mounted on the front of the robot chassis. Joint and gripper actuation is accomplished using Dynamixel servos, and the manipulator links are made of custom 3D-printed plastic. The servos are controlled over USB by the onboard computer. An HD camera (Microsoft LifeCam Cinema, model 1393) is rigidly mounted on the fourth link of the robot arm, and broadcasts video to the robot teleoperator via the onboard laptop.

The user interacts with the robot through the OCU, consisting of a desktop computer (running Ubuntu) with a 25" (63.5cm) monitor with a resolution of 1920x1200px, a keyboard and mouse, as well as the manual input devices listed below. The OCU is connected via wired Ethernet to a router (Buffalo AirStation Wireless G), which connects wirelessly to the laptop and Gumstix computers onboard the robot.

3.3.2 Manual Interfaces

Two different manual interfaces are tested in this study. Both interfaces are connected via USB to the OCU, and custom Java software interprets the interface input and transmits both manipulator and chassis commands to the robot.

Direct Velocity Control

One manual interface consists of a standard gamepad (Logitech Cordless Rumblepad 2). This interface offers a traditional direct velocity control paradigm for the chassis and the manipulator wherein each joint of the robot's manipulator arm is mapped to a different axis on the gamepad. The chassis is driven by holding down a button and using two axes on the same control stick for velocity and steering commands.

Figure 3.2: The master arm controller. The controller is a 1:1 scale replica of the slave arm mounted on the robot's chassis, and is affixed to a platform that is the same size as the robot chassis. Operators manually position the master controller, and the remote slave arm matches the master arm's state at the remote site.

Direct Unilateral Position Control

The second manual controller consists of a 1:1 scale replica of the robot arm, shown in Fig. 3.2. The initial development of this master controller is discussed in [65]. This master arm controller enables direct unilateral control, in which the manipulator arm mounted on the remote robot matches the state of the master controller. A switch on the side of the gripper toggles the master arm between locked (in which all servos hold their positions) and unlocked modes (in which all servos can be backdriven and the arm moves freely). The state of the gripper is controlled by a wheel mounted next to the master arm base. The master arm is mounted on a wooden platform that acts as an analog for the remote robot's chassis. The user provides heading and velocity commands to the chassis with a 3D Mouse (3Dconnexion SpaceNavigator). The user is allowed to position and orient the master arm and 3D mouse setup in any way he or she chooses.

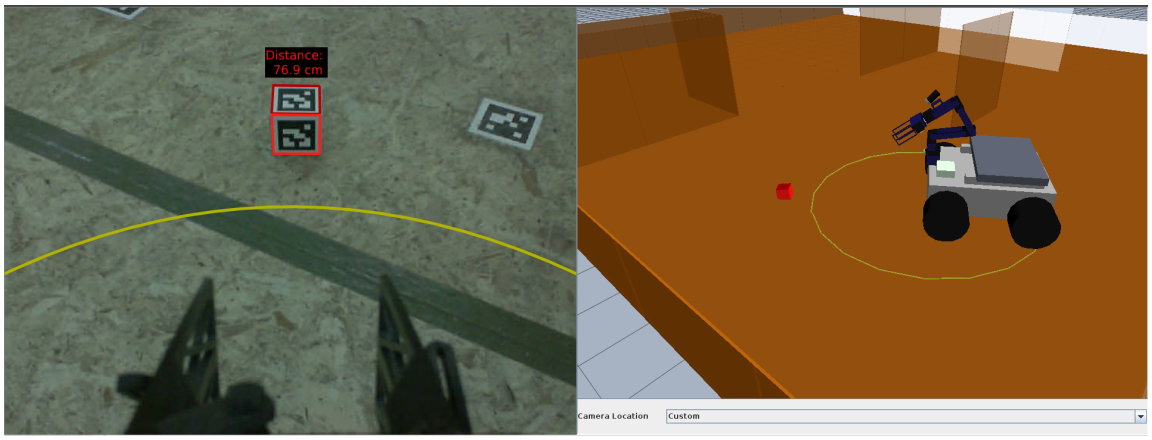


Figure 3.3: A screen shot of the Mixed Reality (MR) visualization interface. The Augmented Reality scene (left) shows a video feed from the camera attached to the robot arm with virtual objects superimposed over the image, including distance information. The Virtual Reality scene (right) shows a third-person view of the robot scene, which can be manipulated to show the robot workspace from any perspective. A yellow halo indicates the projection of the reachable workspace of the manipulator arm on the floor

3.3.3 Visual Interfaces

Two different visual interfaces are tested in this study. Both are displayed on the OCU's external monitor, and the height of each display window is approximately 1/2 the height of the monitor. For both visualizations, the image on the screen was refreshed at a rate of 15Hz.

Video-only Visual Feedback

This visual feedback interface provides a streaming video feed from the camera mounted on the robot's manipulator arm. The position of the camera is controlled by the user by moving the arm.

Mixed Reality Visual Feedback

This interface provides the user with two views of the robot scene, as shown in Fig. 3.3. The left view shows an egocentric (first-person) video feed from the camera mounted on the robot's arm, with Augmented Reality overlays of information about

the robot scene. When a box is recognized, the distance from the camera to the center of the box is indicated in centimeters, and the box is highlighted either in red if it is outside of the reach of the robot's arm, or in green if it can be reached without moving the chassis. Additionally, a yellow arc appears on the floor indicating the reach of the robot's arm. As in the video-only interface, the position of the camera is controlled by the user by moving the arm.

The right side of the interface provides the user an exocentric (third-person) Virtual Reality scene of the robot workspace. The outer boundaries appear automatically, but objects inside the arena (walls and boxes) appear in the virtual scene only after they are discovered by the user. After they are discovered, objects remain in the virtual scene for the remainder of the trial. The operator can manipulate this scene to any desired orientation and position using the mouse. Additionally, there are two pre-set conditions (a follower mode and a birds's-eye view, which can be toggled via the drop-down menu under the virtual scene). The boxes appear on the screen as green or red as in the AR scene, and the halo around the robot arm indicates the arm's reach. The development of this visualization is discussed further in [65].

3.3.4 Software and Communications

Custom Java software runs on three different computers during the user tests: the robot-mounted laptop, the OCU, and the experimenter's control computer. Most of the network communication is managed by the Lightweight Communications and Marshalling (LCM) libraries [31], which uses the UDP Multicast transport layer. For bandwidth reasons, video broadcasting from the robot to the OCU is accomplished using the UDP Unicast transport layer. There was latency between the operator command and visual feedback from the OCU, which was primarily due to the delay from the wireless network. The delay was typically around 0.5 seconds,

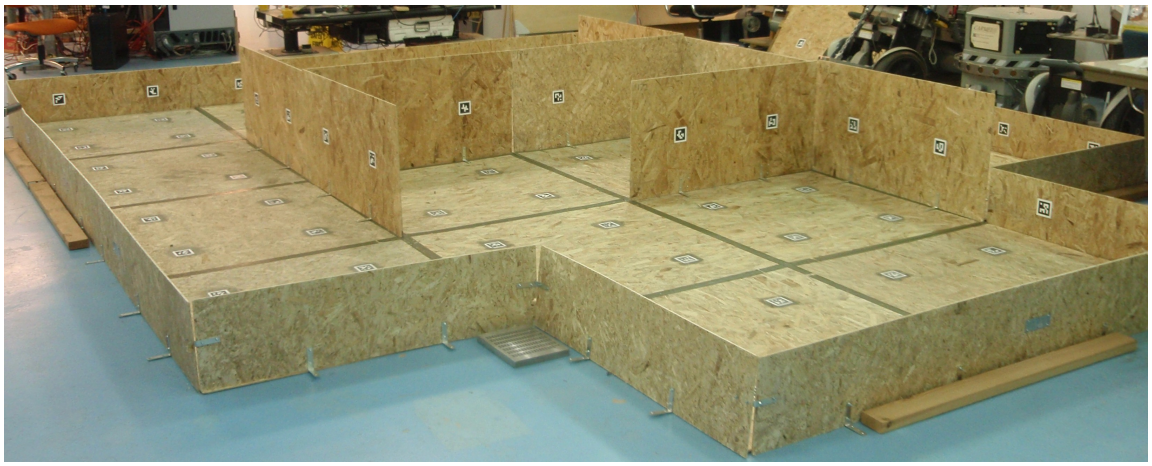


Figure 3.4: Photograph of one configuration of the robot arena. Four different inner wall configurations were used in the trials, all with the same outer envelope. Inner walls were taller than outer walls to prevent users from using the manipulator arm to peek over barriers. Visual fiducial markers were affixed to the walls and floor of the arena to assist the robot with localization.

but occasionally fluctuated up to 4-5 seconds momentarily during the tests.

3.3.5 Test Environment

The robot was teleoperated in a self-contained arena constructed out of plywood specifically for these user tests (see Fig. 3.4). The outer geometry of the arena was fixed with an overall envelope of 6.1×6.1m, but inner walls could be repositioned to create maze-like corridors within the arena. Four different inner wall configurations were used in these tests. The walls and floor were marked with visual fiducial tracking markers [47] that the robot used for localization. A goal bin (26×21×14cm), partitioned into two equal halves (left and right), was located near the starting location of the robot, and both the start position and goal bin were the same for each configuration.

A number of paper boxes (5.72cm cubes), also containing tracking markers on each side, were placed around the arena for each test. Each inner wall configuration had a corresponding paper cube layout, which was consistent throughout the tests.

Users teleoperated the robot in a control room adjacent to the laboratory housing

the test arena, and were unable to see the robot during testing. Occasionally, the robot was audible from the control room, but it is unlikely that this gave users any significant advantage. The control room contained a desk for the operator, which had a control computer, as well as a computer monitor and the manual controllers for the robot, and a desk for one experimenter.

3.4 Procedure

User tests were conducted with 28 volunteers recruited via flier and email advertisements from a population of undergraduate and graduate engineering students. Six participants started the study, but for various reasons were unable to complete the tests, so they are not included, leaving 22 users in the data set. A total of 17 men and 5 women completed the study, ranging in ages from 18 to 30, with a mean age of 22.5 years (standard deviation = 2.7 years). Users were given $20 for participating, with the knowledge that a $25 bonus would be awarded to the top three overall performers as determined at the end of the trials. The tests were designed to take less than three hours total, and many participants needed significantly less time. These tests were approved by the University of Michigan Health Sciences and Behavioral Sciences Institutional Review Board. (UM IRB #HUM00044265).

Users performed the same task four times with four different manual/visual interface combinations: Gamepad/Video (GP+Vid), Gamepad/Mixed Reality (GP+MR), Master-Slave/Video (MS+Vid), and Master-Slave/Mixed Reality (MS+MR). The order in which the interfaces were used was randomized to compensate for any potential learning effects. Additionally, there were four different arena maps (consisting of a set of internal barriers and box locations), which were evenly distributed among the different interface combinations.

3.4.1 Pre-Test

Test volunteers were first greeted by two experimenters and brought into the laboratory housing the robot arena (with all internal walls removed) and robot (in the initial start position). After signing consent forms and being informed of the nature of the experiment, as well as the test procedure itself, users were able to inspect the robot and test arena, and ask any questions.

Users were then brought to the control room with one experimenter, and were given an opportunity to practice with each interface combination in the same order that the tests were to be performed. Each practice session consisted of the users familiarizing themselves with the controls before navigating the robot to the center of the arena (with no internal barriers present), picking up a box with the manipulator, and then driving the robot to the goal bin and depositing the box in the bin. Users were given as much time as they desired to complete this practice task, and their performance did not count towards their overall score. An experimenter ensured that all features of each interface (manual and visual) were explored during the practice tests.

After all of the practice sessions had concluded, the users filled out a demographic survey while experimenters set up the first timed test.

3.4.2 Timed Tests

Four timed tests were performed with the four different interfaces, as outlined in the following sections. Users were allowed to control the robot with the interface provided in any way they saw fit during the tests, and were free to use or ignore any interface feature at their discretion.

Test Tasks

Beginning from the start location (which was the same for each test), users teleoperated the robot around the arena and performed three subtasks in the following order:

Subtask A: The user navigated the robot around the test arena to determine the number of boxes present in the environment (B_{actual}). Once the user has determined what they believe to be the correct number of boxes (B_{found}), they returned to the start position and verbally indicated their answer to the experimenter.

Subtask B: Beginning from the start location, reached at the end of the last subtask, the user navigated the robot to what they believed was the closest box to the start position (as measured by a straight-line path), used the manipulator arm and attached gripper to pick up the box, and then navigated the robot to the goal bin and deposited it into the left partition.

Subtask C: Beginning from the goal location, reached at the end of the last subtask, the user navigated the robot to what they believed was the farthest box from the start position (as measured by a straight-line path), used the manipulator arm and attached gripper to pick up the box, and then navigated the robot to the goal bin and deposited it into the right partition.

The user's objective was to complete the above subtasks as quickly and accurately as possible. Users were scored for each task based on task completion time, with time penalties being given for the following slips (S_i) and mistakes (M_j):

- Miscounting the boxes in Subtask A: 30 seconds $\times |B_{actual} - B_{found}| = M_{count}$
- Choosing the wrong box for either Subtask B or C: 30 seconds per box (M_{wrong}).

- Putting the box in the wrong partition, missing the goal bin entirely, or accidentally dropping the box for Subtask B or C: 15 seconds per drop (S_{drops}). Note that dropped boxes were placed back in the gripper to prevent the user from having to pick up the box multiple times.
- Contacting a wall or occlusion with any part of the robot (chassis and/or arm): 5 seconds per incident (S_{wall}).
- Contacting a box with any part of the robot other than the gripper: 10 seconds per incident (S_{box}).

The adjusted total time in seconds (T_{adj}) by which the users were scored for each test is thus given by:

$$T_{adj} = T_A + T_B + T_C + 30M_{count} + 30M_{wrong} + 15S_{drops} + 5S_{wall} + 10S_{box} \tag{3.1}$$

where T_A, T_B, and T_C are the times taken for Subtasks A, B, and C, respectively.

Survey

Once the entire task was complete, the user filled out a survey about the interface they just used, which included questions about how easy and intuitive the interface felt to the user. There was no time limit for the survey.

Data Recording

Data was recorded in the following ways:

- Experimenters manually tracked the test time and penalties accrued.
- Two videos of the test arena were recorded during all tests. This video was used to verify the times and penalties. One video was recorded by an experimenter following the robot, and the other video is taken from a fixed overhead view of the robot arena.

Table 3.1: P-values of factors potentially affecting user performance for different metrics. Factors with $p < 0.10$ are bolded, and those with $p < 0.05$ are underlined and bolded. T_A, T_B, and T_C are the completion times for Subtasks A, B, and C. $\sum S_i$ and $\sum M_i$ are number of slips and mistakes, respectively. T_{adj} is the total adjusted task completion time, including all time penalties.

Factor	T_A	T_B	T_C	$\sum S_i$	$\sum M_j$	T_{adj}
Manual Interface	**0.083**	**<u>0.000</u>**	**<u>0.000</u>**	0.954	0.509	**<u>0.000</u>**
Visual Interface	**<u>0.016</u>**	0.820	**<u>0.049</u>**	0.793	0.737	**0.055**
Manual Interface×Visual Interface	**0.054**	0.659	0.680	0.423	0.407	0.372
Arena	0.395	0.177	**0.055**	0.838	**<u>0.049</u>**	0.173
Trial	0.153	0.166	0.139	0.127	0.438	**<u>0.025</u>**

- Audio of the test subject's running commentary was recorded during each test. Users were encouraged to verbalize their thoughts as they were teleoperating the robot.
- The operator control computer's screen was was recorded during all tests.
- The OCU logged all network communications that went through LCM, including user input and the robot state data.

3.4.3 Post-Test

After the timed tests were completed, the volunteers participated in a cued debriefing with an experimenter about their overall experience with the tests. Following this interview, volunteers had completed the experiment.

3.5 Results

The results of the user tests are summarized in the following sections.

3.5.1 Significant Factors

Table 3.1 shows a summary of the statistical significances (p-values) for potential factors and interactions over various objective metrics for all trials from all users in

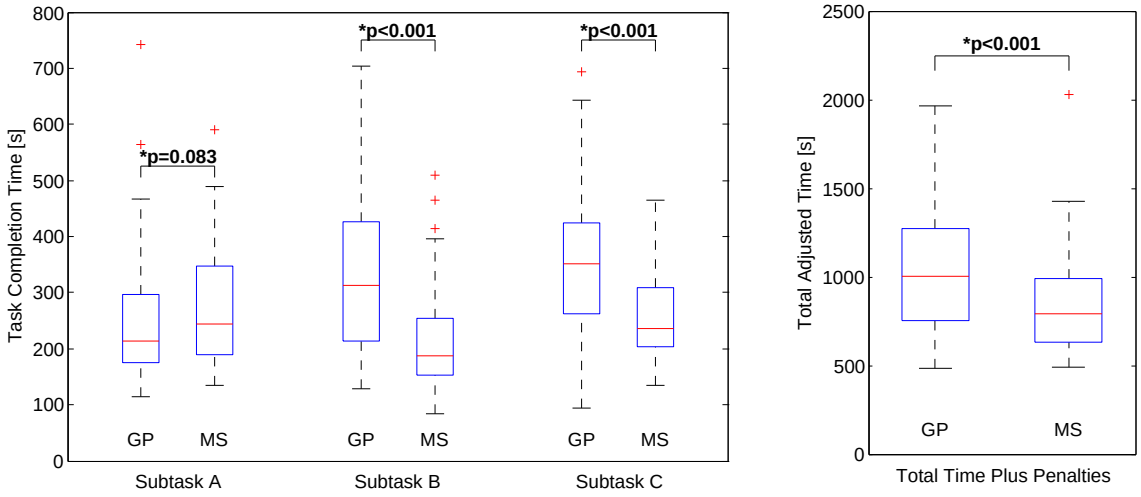


Figure 3.5: Boxplots comparing the completion times for each subtask, as well as the total task time including penalties assessed for slips and mistakes for both types of manual interface. For all boxplots in this chapter, the center line of each box represents the data median, while the edges of the box correspond to the 25th and 75th percentiles (the innerquartile range, IQR), and the whiskers extend to the most extreme data points within 1.5 IQR of the 25th and 75th percentiles. Data points outside this range are considered outliers [24]. The MS interface resulted in significantly slower task completion times for Subtask A, but significantly lower task completion times for Subtasks B and C, as well a significantly lower total adjusted time.

the study. From an experimental-design perspective, Table 3.1 indicates that the type of arena was a significant factor affecting the number of mistakes, and that there was a significant learning effect over the different trials.

Other than the arena type, there was no significant single factor affecting slips or mistakes. Subtask A time (which was primarily a navigation and exploration subtask), Subtask C time, which also involved a fair amount of navigation, and total adjusted time were significantly affected by the type of visual interface, while Subtask A as well as Subtask B and C times (subtasks involving both navigation and manipulation) and total adjusted time were significantly affected by the type of manual interface. There was a significant interaction effect between the manual and visual interfaces for Subtask A time.

Figure 3.5 shows that as a whole, operators using the MS controller performed bet-

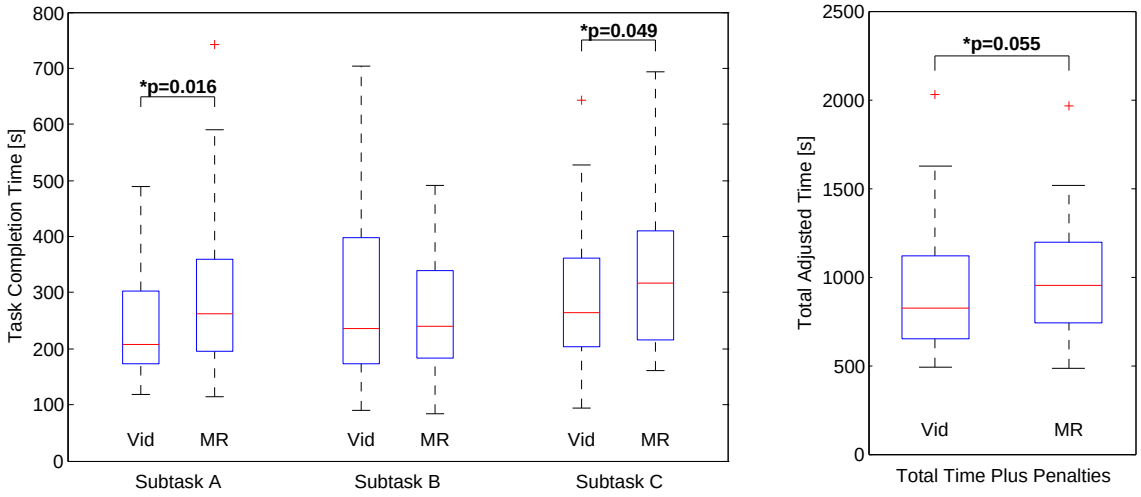


Figure 3.6: Boxplots comparing the completion times for each subtask, as well as the total task time including penalties assessed for slips and mistakes for both types of visual interface. The MR interface resulted in significantly slower task completion times for Subtask A, Subtask C, and total adjusted time.

ter as measured by total adjusted time than operators using the gamepad, however they performed worse on the navigation and exploration sections of the trial (Subtask A).

Figure 3.6 indicates that the MR interface caused operators to take longer on the navigation subtasks, as well as the overall task. There was no difference in subtask completion time for the manipulation subtasks. However, Fig. 3.7 shows there is an interaction effect between manual and visual interfaces for Subtask A, but there is no interaction effect between the two interfaces for Subtasks B and C.

3.5.2 Percent Improvement by Overall Performance

Figure 3.8 shows the percentage improvement in manipulation subtask performance when the user switched from the gamepad interface to the MS controller, averaged over both visualization types, as a function of their overall performance

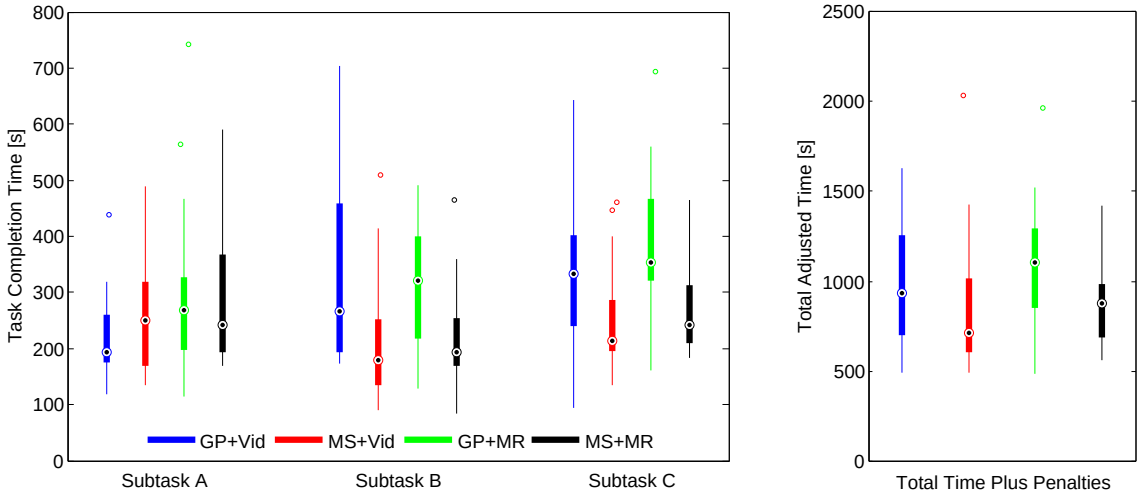


Figure 3.7: Boxplots comparing the completion times for each subtask, as well as the total task time including penalties assessed for slips and mistakes for each interface combination. There is an interaction effect between the visual and manual interfaces for Subtask A that is not present for the other subtasks nor the overall adjusted completion time.

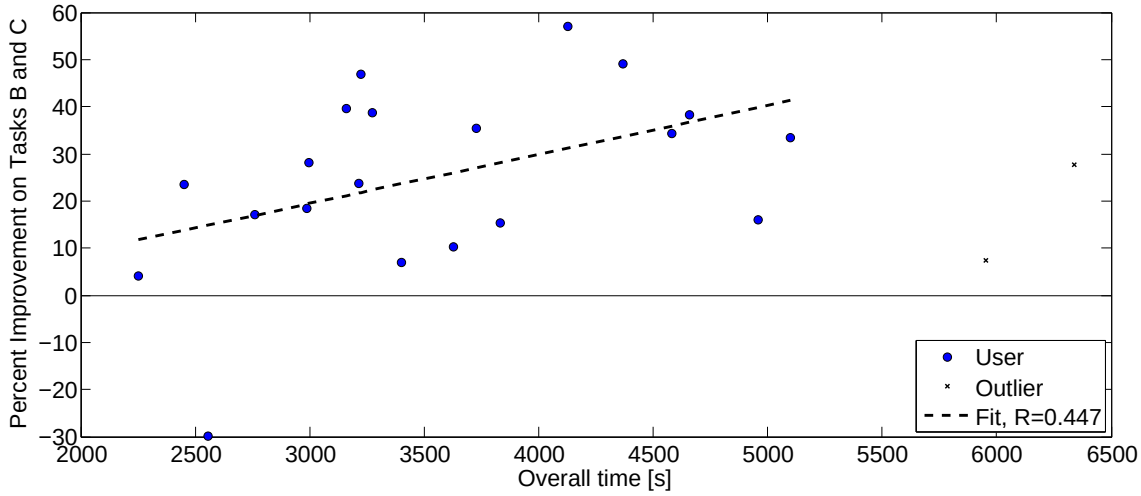


Figure 3.8: Plot showing the percent improvement in performance on manipulation subtasks (Subtasks B and C) when users switched from the gamepad to the MS controller plotted for each user against overall completion time, defined as sum of the total adjusted time for all trials for each user. A negative value indicates the user performed worse with the MS controller. The two outliers had overall times more than 15 minutes longer than the next highest overall times.

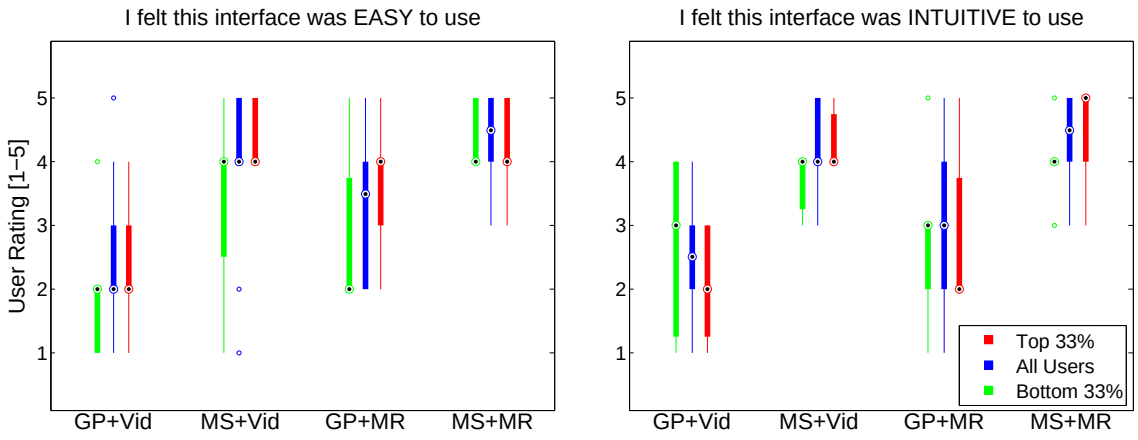


Figure 3.9: Boxplots showing responses to the questions "I thought this interface was easy to use" and "I thought this interface was intuitive to use," separated by overall user performance.

(defined as the sum of each user's four total adjusted times). The trend indicates that the users that performed better as a whole derived less benefit from the MS controller than those that did poorly overall.

3.5.3 User Ratings of Interfaces

Figure 3.9 shows the user ratings for questions about interface ease of use and intuitiveness. The MS interface rated as significantly easier ($p < 0.05$) and more intuitive ($p < 0.05$) than the gamepad, and the MR visualization was rated as easier ($p < 0.05$) and more intuitive ($p < 0.1$) overall, despite adjusted total task times being higher for the MR visualization. There was also an interaction effect between visual interface and manual interface for user rating of interface ease ($p < 0.1$). Additionally, the ratings of the top third of users generally mirrored those of the rest of the test population, even though they did not benefit as highly as the rest of the users. Finally, when asked during the post-test interview, 76% of users with a preference stated they most enjoyed using the MS+MR interface, and 73% of users would choose to use the MS+MR interface if they were asked to participate in

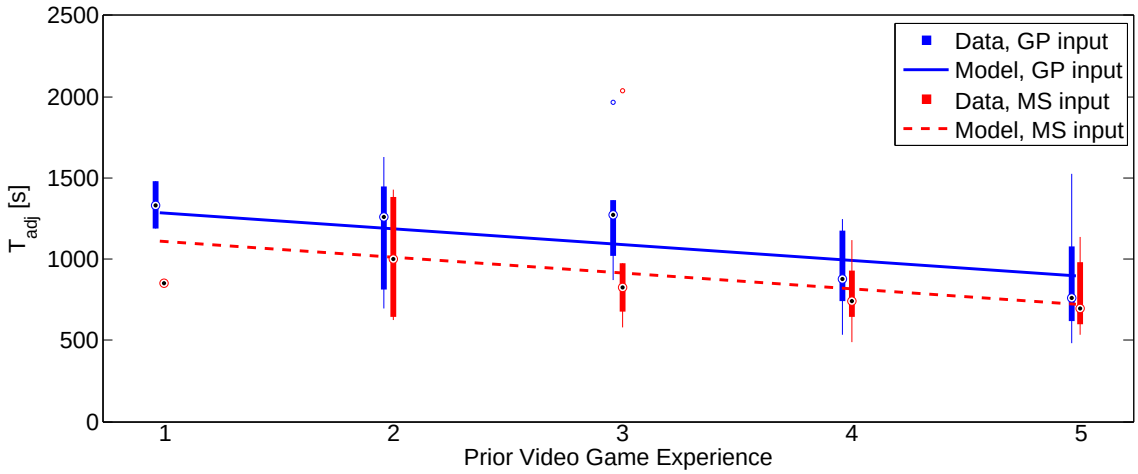


Figure 3.10: The derived performance model for total adjusted time (T_{adj}) with boxplots showing the experimental data. Eq. 3.2 shows that the two most important factors in predicting performance are operator prior video game experience and the type of manual interface used.

another test to be evaluated in a similar way.

3.5.4 User Performance Model

To predict a potential user's performance on the tasks in this user study, a linear regression analysis was performed to obtain a task performance model. Because there were many (possibly correlated) potential factors that could be used in the model, the Bayesian information criterion (BIC) [53], which penalizes models with many regressors, was used to avoid overfitting. From a list of potential regressors, including the test parameters of input device, feedback device, and trial number, and user self-reported prior experience with video games, AR, and VR technologies, the regression model for expected value of total adjusted task time is given by:

$$E\left[T_{adj}\right] = 1385 - 175x_{MI} - 98x_{VG} \tag{3.2}$$

where x_{MI} is a dummy variable representing the type of manual interface (0 – Gamepad, 1 – Master-Slave), and x_{VG} is the user's previous experience playing video

games, on a scale from 1 (None) through 5 (Very Experienced). The model given in Eq. 3.2 is plotted against the experimental test data in Fig. 3.10.

3.6 Discussion

3.6.1 Manual Interface

The results presented in Section 3.5.1 indicate that the MS manual interface resulted in significantly better performance in the manipulation subtasks, as well as the overall task time. This is consistent with the previous literature indicating that position controllers are preferable for dexterous tasks [23]. The MS interface also had the added benefit of acting as a type of visual feedback for the arm orientation, which users often cited as being very useful during tests. This type of visual feedback would not necessarily be present for a resolved position controller, as the user would only be able to visualize the end-effector position.

Some users complained that the 3D mouse used with the MS interface was very sensitive, and was an unfamiliar computer peripheral to many users, so this may account for the slightly decreased performance during the navigation subtask. The 3D mouse could be swapped out for a more familiar physical interface to try to mitigate this issue.

As shown in Fig. 3.9, users found the MS interface easier and more intuitive to operate than the gamepad, mirroring the results of the performance metrics.

While gamepad interfaces are generally portable, simple, and readily available, these user tests reinforce the idea that they are not necessarily the best choice for tasks that require high dexterity, and other input methods should be considered during the design phase of teleoperated systems.

3.6.2 Visual Interface

Section 3.5.1 also indicates that the MR interface actually made users perform worse than the video-only feedback. This is inconsistent with some of the previous literature on MR interfaces, so this result likely stems from the implementation of this particular interface for these tasks and metrics.

The tasks in this study involved both navigation and manipulation. Previous work, such as [16] and [27], only tested MR interfaces for either navigation or manipulation, so no evaluation of switching tasks could be made. Additionally, neither of those studies required the user to manipulate their point of view manually. In the tests performed in this study, users were able to manually control the virtual point of view, which was often done when changing from a driving to a manipulation task. Users often took time to obtain their desired point of view, which could have negated any time advantage gained by having an enhanced understanding of the remote environment. Indeed, a comparison of the total user input control effort for both arm control (defined as the sum of the absolute value of the change in commanded arm position) and chassis navigation (the time-integrated absolute value of the chassis velocity and steering commands) showed no significant difference between the feedback interfaces ($p > 0.1$). This indicates that the extra time task completion time was not time spent commanding the robot.

Previous work also implemented MR interfaces that were very specific to the task being performed. For example, the interface in [16] was specifically designed for space-telerobotics orientation tasks. Our MR visualization was designed to provide more spatial information about the remote environment, but did not implement any features that were specific to the required tasks (for example, there was no indication to the user when the gripper was positioned to pick up a box, which would have been

specific to this set of tasks). Additionally, the implementation of the MR with dual displays may have caused some added cognitive load due to users switching between viewpoints that would not be present in single-display systems.

The performance metric used in these test was heavily based on task completion time, whereas previous studies often focus more on path efficiency, distance from obstacles, and accuracy. For task completion time, these results agree with the findings of Green [27], in which navigation time increased with an exocentric MR interface in comparison to an egocentric view, and Chintamani [16], wherein no significant difference was found between video and MR feedback methods for a manipulation task. Our results contrast the findings of Nielsen [45], in which completion times for navigation tasks using an MR interface were significantly reduced when compared to the completion times when using a traditional interface.

Despite the increased task completion time Fig. 3.9 shows that users rated the MR interface as easier and more intuitive to operate than the video-only visualization. This may be due to the users' perceived novelty of the interface, or it could be because it did effectively reduce the users' cognitive load, despite taking more time to operate.

In context with previous work, these results have implications for UI design for teleoperated mobile manipulators. In particular, it may be worthwhile to implement MR for a teleoperation system that is used for well-defined tasks that require detailed operator understanding of the robot state. For systems that may be used for a variety of unstructured tasks for which task-specific MR overlays cannot be designed in advance, situations involving very simple tasks, or cases where speed is valued over accuracy, the utility of an MR interface may be diminished.

3.6.3 Interaction between Input and Feedback Interfaces

Figure 3.7 indicates that for the navigation and exploration subtask, there was an interaction effect between the manual and visual interfaces. In this case, it may be that the MR visualization either mitigates some of the negative effect of the 3D mouse input for chassis navigation, or that the users spent less time adjusting the virtual scene when using the 3D mouse. There is no interaction effect for the manipulation subtasks, despite the fact that MS input provides a type of visual feedback similar to a third-person view in the virtual scene. This indicates that the arm position information contained in the virtual scene may not have been particularly useful for the box-picking tasks in this test.

3.6.4 Slips and Mistakes

Neither the manual interface nor the visualization significantly affected the number of slips and mistakes committed by the user. For mistakes, this is likely due to the task not inducing many mental errors, as there was an average of only $\mu_M = 0.81$ mistakes per task ($\sigma_M = 0.76$). There were far more slips per task ($\mu_S = 11.8$, $\sigma_S = 7.8$), but no interface had a significant effect on the number of slips. This is surprising given the exocentric view available in the MR interface, which should in theory enable users to determine whether or not the robot was contacting the arena. A possible explanation for this is that occasionally during the tests, the VR scene would briefly render the robot scene incorrectly. As a result, users sometimes commented that they did not trust the VR scene to be accurate, and often ignored the virtual indications that the robot was about to hit something. This is consistent with prior work showing that failures early in an interaction lead to lower overall real-time trust in the robot system [22].

3.6.5 User Skill vs. Interface Benefits

During the experiments, it was observed that the users who generally did well at the tasks seemingly received less of a benefit from the MS controller than those who struggled with the tasks. This observation is confirmed in Fig. 3.8.

We hypothesize several mechanisms for this differentiation in user improvement. First, the top users generally self-identified as being highly experienced with video games, so they would have been more used to the gamepad controller than those with little gaming experience. Additionally, those predisposed to do well on the types of tasks presented in this study likely have better spatial-reasoning skills than others, and the mental mapping from the gamepad to the manipulator arm may not have been as taxing for them. For these tasks, the required arm manipulation was relatively simple and repeatable, so if users figured out a sequence of commands to pick up a box once, it was easy to perform the sequence again. Since the gamepad required users to physically move less than the MS controller (thumbs and fingers vs. arms, and in some cases, entire bodies), the efficiency of the gamepad could have outweighed the intuition of the MS controller. Different results may have occurred if the manipulation tasks required more diverse sequences of movements.

Curiously, as shown in Fig. 3.9 all users rated the MS interface highly. It might be that the users who did not see as much of an improvement were subject to the novelty effect of using something they had not previously experienced, or it could be that they were actually less mentally taxed, but they were proficient enough with the gamepad that they did not show much performance improvement with the MS interface.

Once again, these results have implications for robot UI designers. Primarily, this demonstrates that the designer must understand the types of users and tasks

that the system will need to accommodate. In this case, it seems that if the task is to be performed by novices, the MS interface would be preferred. However, if the manipulation tasks were highly repeatable, and the users were experts with gamepads, then it might not be worthwhile to implement the MS system.

3.6.6 Performance Model

The linear regression model in Eq. 3.2 only contains the regressors of manual interface and prior video game experience, indicating that these are the two most powerful predictors of task completion time for these tests. The BIC procedure used to determine the model indicates that the inclusion of factors representing Visual Interface and trial number overly constrain the model. Additionally, this model does not include any interaction effect between the two factors, meaning it does not capture the diminishing returns on performance improvement from the implementation of the MS interface for "power users" discussed above.

While the performance model given in Eq. 3.2 is limited in direct applicability in that it can only predict task performance for this specific set of tasks, it has broader implications as well. The model indicates that for tasks of this type, the strongest indicators of user performance (of the set of indicators measured) are video game experience and manual input method. Interestingly, the experienced video game players performed better with both the gamepad and the MS interface, indicating that the advantage gained in this task for video game players goes beyond simply having more prior experience with gamepads. Additionally, the trial number does not have a strong enough impact to factor into the model, indicating that short-term prior experience with the specific task is less important than long-term prior gaming experience. However, these results cannot predict the impact of any long-term learning that may occur if a longitudinal study were performed for users repeating

the task many times.

3.7 Conclusions and Future Work

This chapter presents the results of a 22-subject user study exploring the effects of two types of feedback visualizations and two types of manual input methods for a teleoperated mobile robot with a manipulator arm on a set of tasks that required both robot navigation and object manipulation. The performance metric used in the study was primarily task completion time, with a component penalizing slips and mistakes. The results of the user tests indicate that an MS direct position controller was more effective than a computer gamepad for most people, however users who were very good at the tasks overall derived less benefit from the MS control input. An MR visualization was found to cause decreased performance in the tests, but this could be because users spend a significant amount of time adjusting the point of view of a virtual scene. There was an interaction effect between visualization and manual input method for navigation and exploration subtasks, but not for arm manipulation subtasks. No interface was found to affect the number of slips and mistakes committed by the user, but this may be due to users' lack of trust in the exocentric view presented in MR interface. Despite the seemingly counterintuitive performance results, users rated both the MS input and MR visualization as easier and more intuitive to use. A model of predicted user performance for these tests based on user background and interface type is presented, which shows that the two strongest indicators affecting predicted performance for this task are a user's previous experience with video games and the type of manual interface. The results of these tests highlight the need for robot designers to understand both the users and the tasks for which they are designing.

Future work includes an analysis of this same data set with a focus on understanding how user sense of presence plays a role in task performance. Additionally, it would be of interest to compare the implemented unilateral MS system with one that uses haptic feedback for these tasks to determine if there is any significant improvement.

CHAPTER IV

Modeling Teleoperated Mobile Robot Steering Behavior in the Presence of Latency

4.1 Introduction

Latency is a significant factor affecting teleoperated robot performance. Whether the latency originates in the system communications network, processing routines, or sensing hardware, it can negatively impact a human operator's ability to perform even basic remote tasks. With enough delay, a user's entire control strategy is often switched to a move-and-wait open-loop methodology [54], making it impossible to maneuver a robot quickly or efficiently. Moreover, while operators can sometimes adapt to a system delay if it remains relatively consistent, variable latency makes it difficult for humans to predict how the robot will respond [35, 20]. While many teleoperated industrial or surgical robots have the benefit of communicating over wired networks, mobile robots generally must utilize wireless protocols, which have higher latency and latency variation. However, the effects of variable latency are not well-characterized in teleoperated systems.

Understanding how teleoperators interact with robots is key to designing better teleoperation systems. Because it is often not feasible to test large numbers of different design iterations with real human operators, it is desirable to have a model of a human teleoperator that could be used to evaluate multiple designs quickly and

easily. Driver models used for similar purposes have a long history in the automotive industry [36], but these models may not be directly applicable to teleoperation, as the two tasks are quite different. While vehicle drivers have a wide variety of sensory feedback available, teleoperators are generally limited to relying solely on visual feedback, which is often delayed and has a limited field of view [14]. Additionally, input devices for automobiles (*e.g.* steering wheels with haptic feedback) are generally not the same as those used in teleoperation, which can be as simple as off-the shelf video game controllers. Finally, the internal models automobile drivers have for their vehicles are likely far more developed than those of even experienced teleoperators. Thus, there is a need develop new models of humans performing teleoperation tasks in remote environments.

The results in this chapter are based on a 31-subject user study designed to measure the effects of both constant and variable latency on a simulated teleoperation steering task using a commercially available gamepad as an input device. The input commands from the human to the robot were recorded with the aim of developing a driver model to simulate human behavior for simple steering tasks under different latency conditions. The study resulted in three key findings: 1) Variable latency scenarios resulted in worse path-following performance than did constant latency scenarios having the same mean delay time. 2) Variable latency can be considered as an equivalent constant latency under the conditions tested in this study. This equivalent constant latency is greater than the mean delay of the variable distribution. 3) A teleoperator's steering commands with a gamepad can be reasonably modeled as PD controller based on the preview of the robot's anticipated lateral displacement. To the our knowledge, this is the first steering model developed specifically for teleoperated robots.

The chapter is organized as follows: First, prior work regarding latency's effects on teleoperation as well as steering models in both the automotive and Human-Computer Interaction (HCI) domains are discussed in Section 4.2. Then Sections 4.3 and 4.4 present the design and implementation of a user study with the aim of characterizing teleoperators' responses to different latency scenarios. The user study also gathered experimental data on operator driving style to be reproduced by a driving model. The results of the study are presented and discussed in Sections 4.5 and 4.6, respectively. A driver model is developed and validated using the test data in Section 4.7. Finally, Section 4.8 discusses conclusions and future work regarding the direction of this research.

4.2 Background

4.2.1 Latency in Teleoperation

It is well-established that latency has a detrimental impact on teleoperation performance, and time delay is known to be one of the most significant factors affecting remote perception and manipulation [14]. Sources of latency in a teleoperated robot system include network delays, sensing delays, and processing delays, as well as delays caused by the operator's cognitive and physical processing, which can themselves be affected by the delay in the rest of the feedback loop (see Chapter II).

One of the earliest studies in this domain investigated open-loop position control of a remote manipulator, and found that users adopted a move-and-wait strategy when the delay was above 1.0 second [54]. More recent studies have examined mobile robot teleoperation performance under other conditions including 2D driving [35, 20] and 3D underwater navigation tasks [18]. Prior work has shown that variable latency leads to worse performance than constant latency, because users are less able to compensate for the changing delays [35, 20]. The directionality of the latency

(whether user-to-robot or robot-to-user) has also been investigated, where it has been found that users felt robot control was more difficult when the latency was in the robot-to-user direction, but no objective difference in performance was observed [35].

4.2.2 Steering Models

Modeling human driver behavior has a rich history in the automotive domain [36, 60, 21]. From transfer functions models to nonlinear and adaptive controllers to neural networks, genetic algorithms, and fuzzy logic controllers [36, 21], there are myriad methodologies for modeling vehicle lateral control (steering), longitudinal control (acceleration and braking), and combined control. These models can be used to simulate human drivers when testing new vehicle designs and technologies [36, 21], and despite the complexity of human behavior, low order models are often sufficient for many control tasks [11].

Regardless of overarching approach, all driver models aim to capture the key characteristics of the human driver as a controller in a feedback loop. MacAdam [36] notes that the essential requirements of a model should include: a time delay due to human processing, a preview of the upcoming control requirements, the ability to adapt to different vehicle and operating conditions, and an internal model to predict vehicle responses. Our modeling efforts adhere to these requirements.

Automotive steering models can use one or more feedback cues as inputs to the driver model, including any or all of the following: lateral displacement, lateral acceleration, roll angle, heading angle, and yaw rate. The cues can be processed by the model in one or more forms, which may include visual cues, motion effects, sound, and tactile information [36]. However, typically only limited visual signals are available in teleoperation scenarios.

4.2.3 Steering and Latency in HCI

Steering has also been addressed by researchers in the field of Human-Computer Interaction (HCI). There exist several laws quantitatively linking human performance in steering tasks to the spatial constraints of the scenario. Notably, Accot and Zhai developed a law (derived from Fitts' pointing law) showing that the time T to steer through a path is governed by [1]:

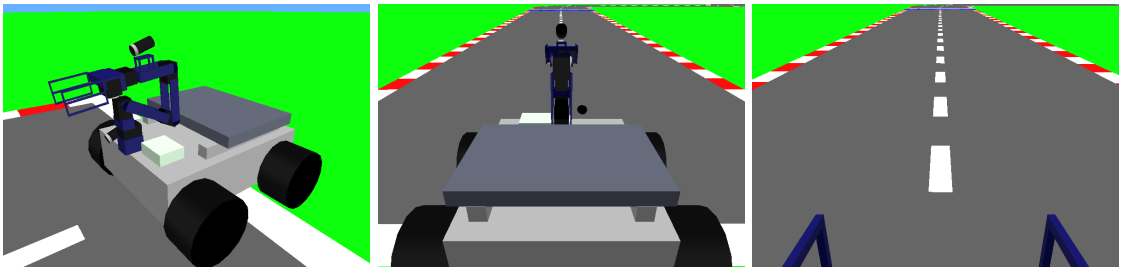
$$T = a + bID \tag{4.1}$$

where a and b are constants and ID is a difficulty index. While this steering law was originally developed for 2D trajectory-based interactions, such as menu navigation with a mouse [1], it has been demonstrated that this relationship also holds for locomotive steering tasks in virtual environments [69].

While this law has not been directly tested in the presence of latency, similar work has found that performance in 2D target-following tasks using a mouse is significantly degraded by latencies over 110ms and for latency variations over 40ms [48]. Additionally, Fitts' law requires corrective terms in the presence of time delays when applied to scenarios ranging from planar mouse-pointing tasks [37] to haptic Virtual Reality surgical simulations [32], indicating that a steering law may also require corrective terms to accommodate latency. Another limitation of this steering law is that it only applies to successful steering tasks, and cannot account for failed trials in which a user steers outside the constraints of the scenario [69].

4.3 Task Setup

A user study was performed to gather data on task performance and operator driving style in robot steering tasks using a low-fidelity simulation of a teleoperated mobile robot driving on a simulated test track of constant width. The user's goal



(a) Simulated Robot (b) Third-Person Viewpoint (c) First-Person Viewpoint

Figure 4.1: Renderings of (a) the simulated robot, (b) the exocentric (third-person) viewpoint, and (c) the egocentric (first-person) viewpoint presented to the users in the study.

was to steer the robot such that it followed the track's indicated centerline as closely as possible.

4.3.1 Simulation Environment

A custom simulation environment for this set of tests (with a look inspired by the classic arcade game *Pole Position*) was written in Java using the April Robotics Toolkit [6] and Lightweight Communications and Marshalling (LCM) [31] libraries. The simulation uses a simple kinematic driving model of a representative skid-steer robot chassis (see Fig. 4.1a), calculating the robot's Cartesian position (x_1,x_2) and orientation (θ) for each time step ($k+1$) using the commanded robot speed (v) and turn rate (ω):

$$\begin{bmatrix} x_1 \\ x_2 \\ \theta \end{bmatrix}_{k+1} = \begin{bmatrix} x_1 \\ x_2 \\ \theta \end{bmatrix}_{k} + \begin{bmatrix} v_k \cos\theta_{k+1}\Delta t \\ v_k \sin\theta_{k+1}\Delta t \\ \omega_k \Delta t \end{bmatrix} \tag{4.2}$$

The simulation runs in its own Java thread at a rate of 60Hz.

4.3.2 User Interface

The user gives steering commands to the robot using one of the two analog mini joysticks of a standard computer gamepad (Logitech Cordless Rumblepad 2), which

is read by the Operator Control Unit (OCU) at a rate of 40Hz. A small amount of noise is artificially added the gamepad input, generated from a uniform distribution on the range of $[-10\%, 10\%]$ of the maximum possible input command, to simulate the noise present in a physical robot system. Operators do not have control over the robot speed; it is constant for the duration of the trial unless the robot is driven off of the track.

The simulation visualization is displayed to the user via the OCU on a 25" (63.5cm) monitor with a resolution of 1920x1200px in a full-screen window. The visualization refreshes the displayed frame at a rate of 15Hz. Two different viewpoints are used in this study. A third-person view (Fig. 4.1b) shows the scene from a virtual camera following behind the robot. A first-person view (Fig. 4.1c) shows the scene from the point of view of the robot's camera, with a small portion of the robot's gripper visible in the bottom portion of the window. Both viewpoints are aimed at a point the same distance in front of the robot, giving both views identical lookahead distances.

4.3.3 Insertion of Delay

Gamepad instruction packets are read by the OCU and enter a queue waiting to be read by the simulation, representing a delay in the human-to-robot direction. For each incoming instruction, a simulated delay (δ) inserted between the gamepad instruction and the simulation is determined by:

$$\delta(\delta_{min}, \sigma) = \delta_{min} + |\delta_X| \tag{4.3}$$

where δ_{min} is a minimum baseline delay, and $\delta_X \sim \mathcal{N}(0, \sigma^2)$ is a continuous random variable having a normal distribution with variance σ^2. This results in a stochastic delay distribution approximating the qualitative shape of wireless packet intervals

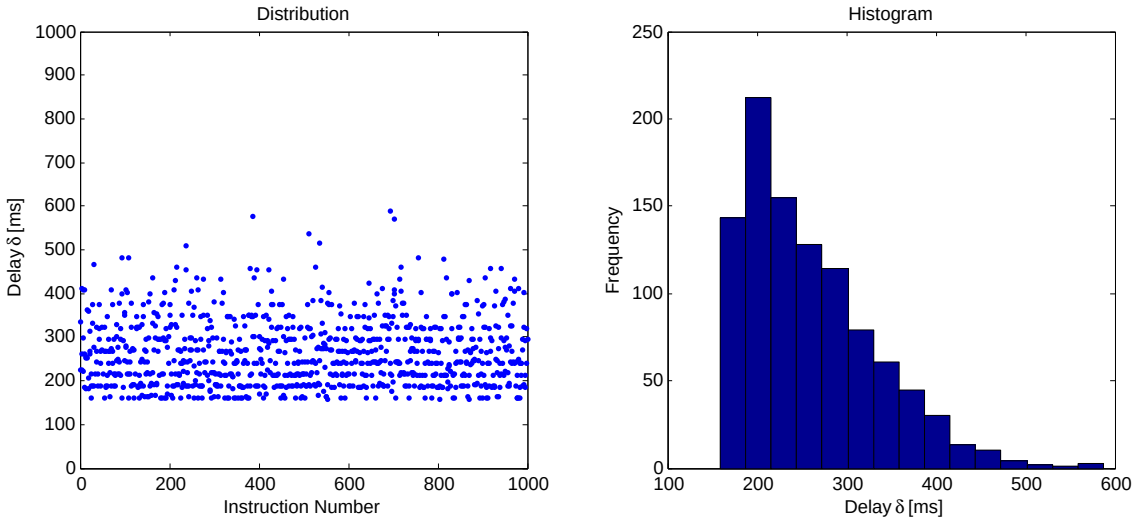


Figure 4.2: Distribution of gamepad instruction packet delays with δ_{min}=150ms, σ=125ms, and mean delay $E[\delta]$=250ms. The quantization is due to the gamepad sampling rate of 40Hz, which also causes the delay minimum (and therefore the mean delay) to be slightly greater (<10ms) than the nominal value, but this is negligible compared to the induced delay.

reported in [5]. For simulation of constant delay, we set $\sigma = 0$ such that $\delta = \delta_{min}$. Once the delay is determined for the current time step, the newest gamepad instruction in the queue that is at least δ ms old is used as the command input to the driving simulation, and older instructions are discarded. If no instruction is older than the desired delay, the previous instruction is used until the oldest instruction is older than δ ms. A sample distribution of instruction delay values is shown in Fig. 4.2.

This induced delay is added to the system and does not include or compensate for any further computer processing delays or delays due to the display device, which are assumed to be negligible compared to the magnitude of the latencies induced in the trials. Additionally, there is a slight delay from the sampling time of the gamepad, which is also negligible.

4.3.4 Test Track

Sixteen non-intersecting test tracks were randomly generated that each contain exactly one of the following elements:

- Right Turn
- Left Turn
- U-Turn Right
- U-Turn Left
- S-Turn Right-Left
- S-Turn Left-Right

All turns have a constant radius of 2m, and the width of the track is 2m, with 0.125m borders on either side. The width of the robot (wheel-to-wheel) is 0.74m. The turn gain of the gamepad input is scaled by robot speed such that the minimum turning radius of the robot is always 1.6m, preventing users from relying on the actuator limits of the gamepad to execute ideal turning motions. Each track element has a section of straight-line path at least 5m long immediately following it to allow the user to try to recover from any deviation sustained during the turn. Additionally, there is a 10m straight-line practice section at the start of each track in which the user can familiarize him/herself with the test condition. A sample track is shown in Fig. 4.3.

4.3.5 Scoring

The path-following score for each trial is determined as a function of the robot's distance from the centerline over the course of the path. Scoring begins after the robot has passed the start line indicating the end of the practice section of track.

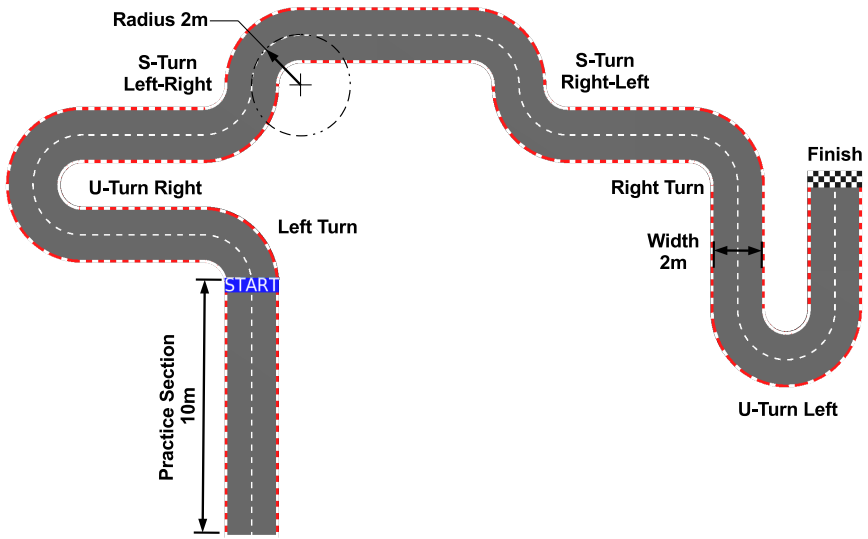


Figure 4.3: Representative track with dimensions for simulated driving tasks. Sixteen total tracks were randomly generated for use in the study, all with the same set of features and dimensions. A practice section was included at the beginning of the track to enable users to familiarize themselves with the test conditions. Scoring for each trial commenced after the robot passed the "Start" line.

The score at time step i is given by:

$$S_i = \max(0, 1 - |y_i|) \tag{4.4}$$

where y_i is the lateral displacement at step i. Then the total score is determined as the average of the of the scores at each time step:

$$S = \frac{1}{n} \sum_{i=1}^{n} S_i \tag{4.5}$$

Therefore, a score of 1 indicates that the path was followed perfectly, and a score of 0 indicates that the robot was never on the test track.

4.4 Procedure

User tests were conducted with 32 volunteers recruited via flier and email advertisements distributed to a population of undergraduate and graduate engineering students. One participant withdrew from the study, leaving 31 users in the data set. A total of 22 men and 9 women completed the tasks, ranging in ages from 18

Table 4.1: List of latency types used in the user study. All values are listed in ms. $E[\delta]$ is the expected value of the random variable δ, representing the delay inserted between the user and the robot for each latency type.

Latency Type	δ_{min}	σ	$E[\delta]$
A	0	0	0
B	250	0	250
C	500	0	500
D	750	0	750
E	150	125	250
F	300	250	500

Table 4.2: Number of users participating in each scenario. All users participated in six of the scenarios (starred), while the remaining six scenarios were distributed evenly among the participants.

	Constant Latency				**Variable Latency**	
Visualization POV	A	B	C	D	E	F
Third Person	31*	11	31*	10	31*	11
First Person	31*	10	31*	10	31*	10

to 37, with a mean age of 23.5 years (standard deviation = 4.2 years). Users were given $10 for participating, with the knowledge that an additional $10 bonus would be awarded to the top performer of six different tasks as determined at the end of the trials, with a $30 bonus cap. The tests were designed to take less than one hour, and most participants needed approximately 45 minutes. These tests were approved by the University of Michigan Health Sciences and Behavioral Sciences Institutional Review Board. (UM IRB #HUM00044265).

4.4.1 Study Design

This study used a repeated-measures test design, with three independent variables: Viewpoint – first- and third-person; latency type – see Table 4.1; and robot speed – 1.0 m/s, and 1.5 m/s. However, to make the study more efficient, both speed levels for a given viewpoint-latency scenario were tested consecutively, and a survey was administered once per scenario, instead of after each individual trial. Therefore,

there were 12 scenarios tested in the study, with two speeds per scenario. Due to time constraints, users did not experience all 12 test conditions; instead, each individual saw eight scenarios. Six of the scenarios were performed by all users (these were used as the basis for the bonus payments), and the remaining two scenarios for each user were evenly drawn from the secondary set of scenarios (see Table 4.2). Users were not informed which scenarios were common to all participants. The order of the scenarios was randomized to counterbalance any learning or fatigue effects. Additionally, the order of the speeds in each scenario was randomized, as was the selection of track for each trial.

4.4.2 Test Procedure

Test subjects were first greeted by an experimenter, and brought into the testing room. After being informed of the nature of the experiment and signing consent forms, the users filled out a demographic survey. The experimenter then explained in detail the procedure of the trials, and answered any questions. Users were informed of the scoring mechanism prior to the tests, but were not told their test scores so as to not bias their survey responses.

Once the user was ready to move on to the tests, the experimenter would remotely trigger the first trial of the first scenario. Users would push a button on the gamepad to initiate driving, and would steer the robot along the test track with one of the two mini joysticks on the gamepad, at a constant pre-determined speed, attempting to keep the center of the robot in line with the center of the test track. If the center of the robot passed completely off the track, the speed of the robot was automatically reduced to half of the original speed, and returned to normal when the user was able to get the robot back onto the track.

Once the first trial was completed, the experimenter would trigger the second trial

of the first scenario, in which the robot would drive at a different speed, and the user would again navigate the test track (a different track was used for each trial). Once both trials were finished for a given scenario, the user filled out a survey about the test condition s/he just experienced by responding to a series of seven-point Likert items. Questions about the users' sense of presence in the robot's workspace were derived from Witmer [66].

This process was then repeated for the remaining seven scenarios.

Data was recorded in the following ways:

1. The OCU screen was recorded during all tests.
2. Audio of the test subjects running commentary was recorded during each test. Users were encouraged to verbalize their thoughts as they were steering the robot.
3. Data logs recorded the user's input to the robot as well as the robot state during all tests.
4. Surveys were administered on the OCU via web browser.

One data log for latency scenario C with a first-person view at a speed of 1.5 m/s did not record properly, so this trial is omitted from the dataset, but the survey results are included.

4.5 Results

The results of the user tests are summarized in the following subsections.

4.5.1 Significant Factors Affecting Objective Performance

Table 4.3 shows the p-values for the factors and interaction effects potentially affecting users' objective performance, as measured by Eq. 4.5, for the path-following

Table 4.3: Table indicating the p-values of factors and interaction effects potentially affecting the path-following score. Factors with p <0.001 are bolded.

Factor	p-value
Speed	**0.000**
Viewpoint	0.130
Latency	**0.000**
Track	0.302
Trial Number	**0.000**
Speed×Viewpoint	0.126
Speed×Latency	**0.000**
Viewpoint×Latency	0.155
Speed×Trial Number	0.852

task. The robot speed, latency type, and trial number all significantly affect the trial score, and there is an interaction effect present between the robot speed and the type of latency. The viewpoint type and track number did not significantly affect the path-following score.

Figure 4.4 shows a boxplot of trial scores under different latency and speed conditions. The plot shows decreasing scores and increasing score variance for increasing amounts of latency. Both of these trends agree qualitatively with the findings in [18], which measured task completion time in a 3D navigation task. There is a clear decrease in performance for higher robot speeds under all latency conditions, and the performance dropoff is particularly stark above 500ms of delay. Additionally, the interaction effect between speed and latency type is apparent in the plot. Finally, the scores for trials under variable latency are lower than the scores for trials under constant latency with the same mean, as scenario E shows worse performance than B, and F shows worse performance than C.

4.5.2 Survey Responses

Figure 4.5 shows the participant responses to the questions on the survey measuring the teleoperators' sense of the latency on a seven-point Likert scale. The responses show good internal consistency (Chronbach's α=0.87), in measuring how

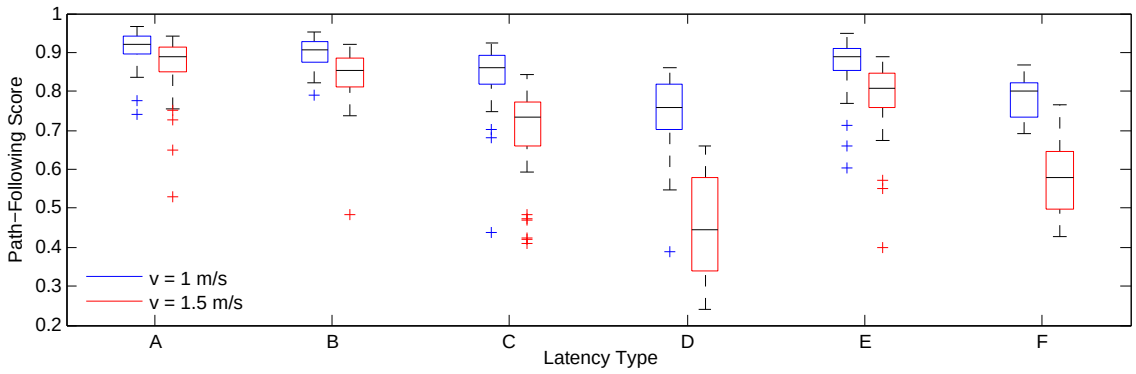


Figure 4.4: Boxplot showing the path-following score, as defined by Eq. 4.5, indicating user performance under varying latency and speed conditions. For all boxplots in this chapter, the center line of each box represents the data median, while the edges of the box correspond to the 25th and 75th percentiles (the innerquartile range, IQR), and the whiskers extend to the most extreme data points within 1.5 IQR of the 25th and 75th percentiles. Data points outside this range are considered outliers [24].

much delay the user felt in the system, and indicate that users felt more delay with increasing delay mean, and also felt more delay in scenarios involving variable latency than for scenarios having constant latency with the same mean.

4.6 Discussion

4.6.1 Significant Factors

The results of Section 4.5.1 indicate that, as expected, robot speed and latency type significantly affect performance. Additionally, the trial number had an effect on performance, indicating that there was a significant learning effect. Because the order of the scenarios and speeds were randomized, this should not affect the results of this study. The track number was not a significant factor, indicating that all tracks were of equal difficulty.

In contrast to our expectations, the viewpoint presented to the user did not have a significant effect on performance. This comes despite videogamers' general preference for a 3rd-person point of view [51]. This result may have several explanations. First, a part of the robot arm's gripper was visible in the first-person view. This was done

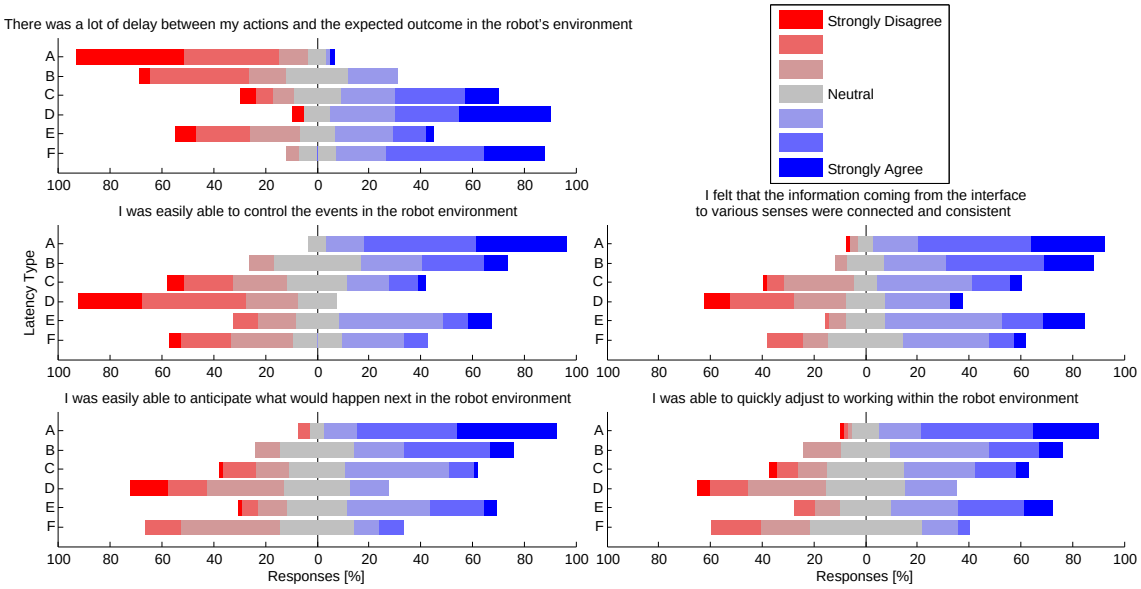


Figure 4.5: User responses to questions designed to assess how much delay the user felt in the system for each latency type. The survey with 7-point Likert items was administered at the conclusion of the two speed trials for each scenario.

purposely to accurately depict the physical robot represented by the simulation. However, this gave users a reference point for the camera's position with respect to the robot, which may have aided their performance. Additionally, this task was relatively simple and did not involve interaction with any objects other than the track, so the benefits of a third-person view may not have been apparent for this situation. One participant specifically commented that while he normally uses a third-person view in racing video games, the fact that he did not have to account for anything in the environment other than the track meant that the third-person view did not help as much as he anticipated.

4.6.2 Performance Modeling and Variable Latency Equivalence

Figure 4.6 shows the score distribution of the constant latency scenarios plotted with a trend line for each of the two speeds, with which we can predict path-following scores for delays not explicitly tested in this study.

By determining where the path-following score results of the variable latency scenarios intersect with the trend for the constant latency case, we can estimate an equivalent constant-latency value for each variable latency scenario. Figure 4.6 indicates that for variable latency scenario E, the median scores at the two different speeds correlate with the same equivalent constant delay, 380ms. Similarly, latency scenario F corresponds to a constant delay of 660ms for both speeds. We conjecture that for this steering task, variable sources of delay can be mapped to an equivalent constant delay, independent of speed. This could simplify the process of understanding user responses to delay for teleoperated tasks, as once an equivalence is found, it may be possible to use this equivalence instead of the delay distribution for modeling and predicting user behavior. Note that this equivalence is with regards to the system performance under the different latency scenarios, and does not imply that pure delay and latency variability have equivalent underlying mechanisms resulting in decreased performance.

Figure 4.5 shows that users also reported experiencing the variable latency at a similar equivalence, in that the users' sense of the delay in the system followed the same trend as the objective score, wherein variable latency type E fell in between constant latency types B and C, and type F was between and C and D. Figure 4.7 shows the sum of responses to these five Likert items (with the responses to "There was a lot of delay between my actions and the expected outcome in the robots environment" reversed to make the sentiment of the statements consistent). A linear trend can be found between the constant delay scenarios and the sum of the ratings, and plotting the survey results of the variable latencies on this same scale shows that the constant equivalences determined by the objective scores are consistent with the trend in the survey responses. This indicates that users' perceptions of the variable

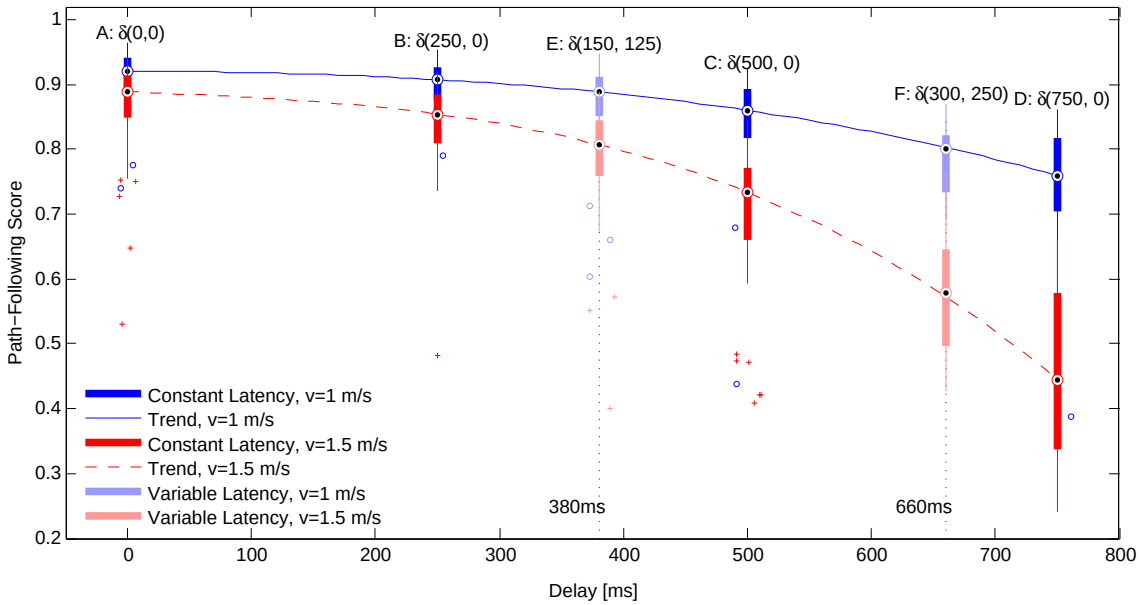


Figure 4.6: Model of user performance as measured by the path-following score for various latencies and robot speeds. The data in this plot is the same as in Fig. 4.4, but the scores are now plotted by numerical latency value on the horizontal axis. The trend line runs through the median score for each constant latency case (A-D), and the variable latency cases (E-F) are then shown at their corresponding equivalent constant delays.

delay agrees with their objective performance on the tasks, despite their not being informed of their scores during the trials.

4.7 Driver Model

This section discusses the development of a model for simulating the steering commands issued by the teleoperator under different system latency conditions. This model could be useful as a substitute for a real teleoperator when testing mobile robot designs for tasks requiring steering inputs.

4.7.1 Driver Behavior

To act as an acceptable substitute for a human driver, the steering model must accurately replicate the key characteristics of the human driver. Figure 4.8 shows two example datasets from runs under low latency (Latency A), and under high latency

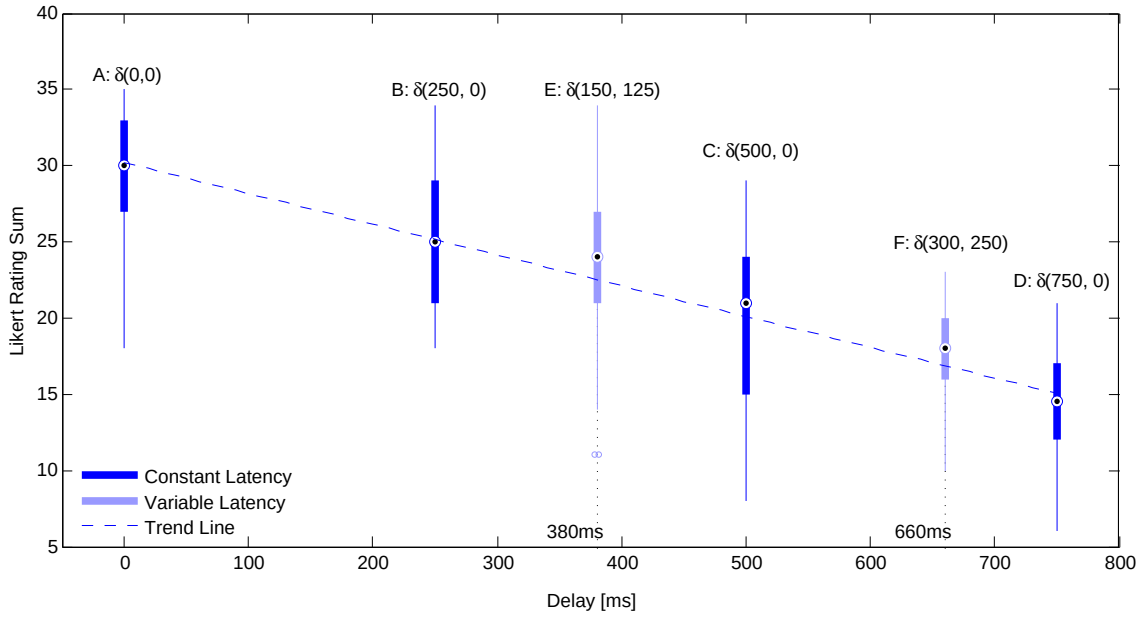


Figure 4.7: Boxplot of user responses to survey questions related to operator sense of delay. The fit line is generated from the constant-latency cases (A-D), and the variable latency scenarios (E-F) are plotted at their constant latency equivalents, as determined by path-following score in Fig. 4.6.

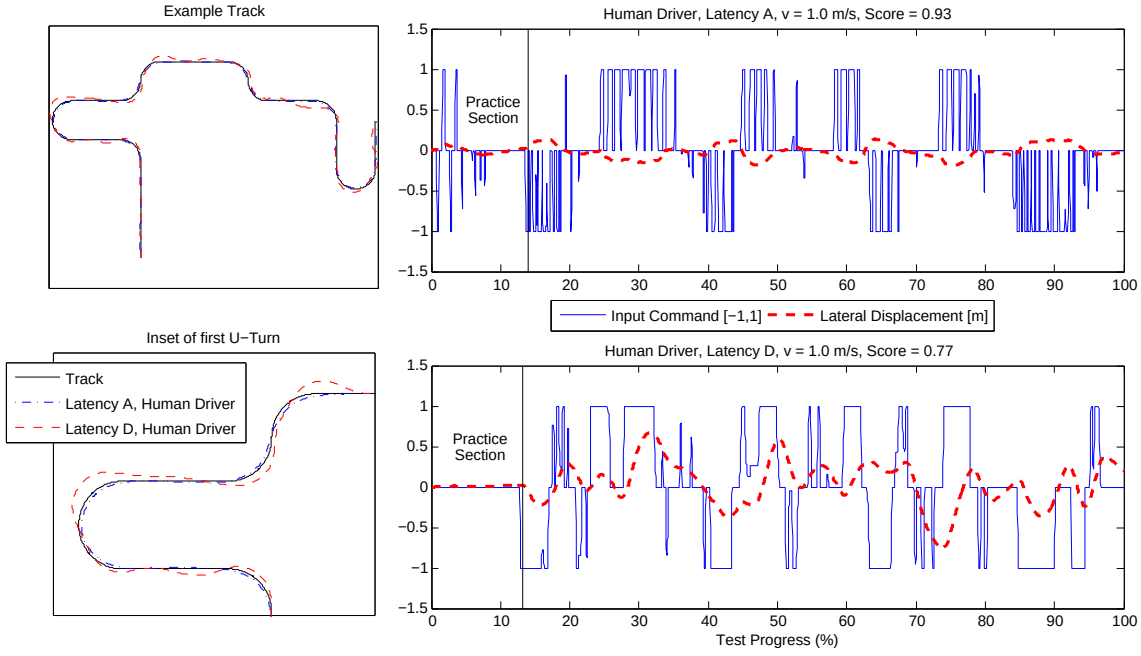


Figure 4.8: Plots showing example datasets of low-latency and high-latency test cases. The datasets are from two different users. These datasets were chosen as representative of the median user performance in the test. Scores were not accumulated during the practice section. Note that even though the operators could use the joystick to command any value between -1 and 1 to the robot, users generally only toggled between 0 and ± 1.

(Latency D), which are representative of many of the test datasets.

The first characteristic that the model should accurately reproduce is the profile of the lateral displacement of the robot along the path. This can be represented by the path-following score, but the simulated displacement should also show similar patterns to the measured data. Two convenient measures to characterize the shape of the path are the maximum lateral displacement (maximum error), and the length of the path (as meandering paths will be longer overall).

We also wish to emulate the characteristics of the input command. As shown in both traces in Fig. 4.8, the input command is almost always saturated. This is because most users tended to move the control stick on the gamepad as far to the left or right as its travel allowed rather than use an intermediate input. This tendency was present under all test conditions: over all the trials, users kept the joystick centered 55.5% of the time, pushed the stick to the far left or right 33.5% of the time, and only 11% of commands were any value in between. This type of gamepad input behavior has been previously observed in computer racing games [12]. Whether the inputs are in the form of quick, frequent adjustments, or long sustained turning commands (both strategies were employed by users in this study), one way to further characterize the input command is by measuring the average magnitude of the command, giving a measure of control effort. An average magnitude of 1 means the driver was constantly turning, while an average of 0 means no input was given. We can also measure the rate at which the operator toggles the gamepad joystick from center to either side. While a simulated controller with high gain may tend to chatter back and forth very quickly, a human operator may not be physically able give this type of input. By dividing the number of toggles by the trial time, we obtain an overall toggle rate for the trial, by which we can determine if the command profile

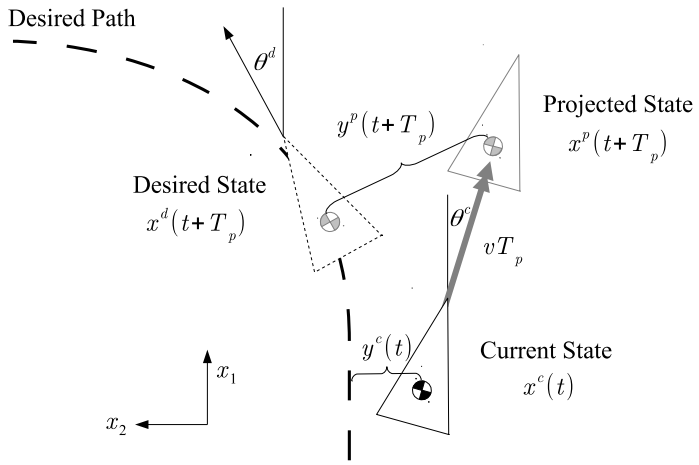


Figure 4.9: Diagram illustrating the determination of the projected lateral displacement. The projected state is the location of the robot at a future time $t + T_p$, assuming a constant angle and velocity. The desired state is then defined to be the position and orientation of the desired path that is closest to the projected state. The perpendicular distance between the desired state and the projected state is then taken as the projected lateral displacement $y^p(t + T_p)$.

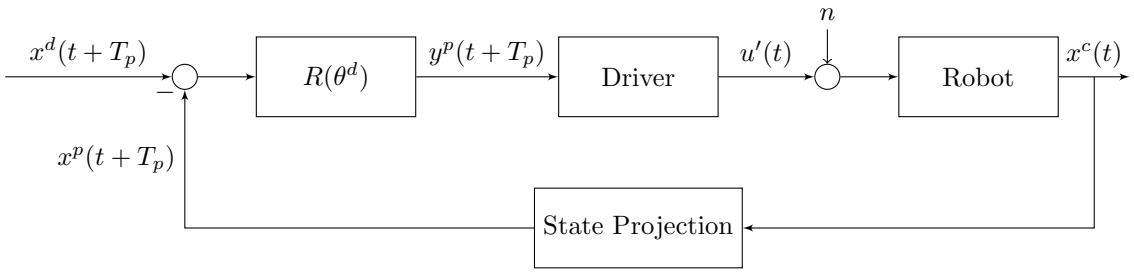


Figure 4.10: Block diagram showing the steering control loop. The lateral displacement $y^p(t + T_p)$ is determined from the difference between the projected and desired robot states at time $t + T_p$. The $R(\theta^d)$ block represents the rotation operation described in Eq. 4.8. The n term represents the noise injected into the command signal.

was feasible.

4.7.2 Model Development

To develop a model for teleoperated robot steering, we can draw from some of the techniques previously developed for automotive steering models. Specifically, we use a preview of the desired path combined with an internal model of the vehicle kinematics and an operator time delay.

A simple steering model can be developed based on the driver's anticipated deviation from the desired path at some future time $(t + T_p)$. In this case, we choose the

projected lateral displacement $y^p(t+T_p)$ of the robot as the feedback cue. Figure 4.9 illustrates the process of determining the projected lateral displacement, which is based on the projected state of the robot $x^p(t+T_p)$, assuming it continues its trajectory from the current state $x^c(t)$ at a constant velocity:

$$x^p(t+T_p) = \begin{bmatrix} x_1^p \\ x_2^p \\ \theta^p \end{bmatrix} = \begin{bmatrix} x_1^c \\ x_2^c \\ \theta^c \end{bmatrix} + \begin{bmatrix} v\cos\theta^c \\ v\sin\theta^c \\ 0 \end{bmatrix} T_p \tag{4.6}$$

The desired future state of the robot $x^d(t+T_p)$, is the point along the desired path closest to the projected state, as measured by Euclidian distance:

$$x^d(t+T_p) = \arg\min_{x \in path} \sqrt{(x_1 - x_1^p)^2 + (x_2 - x_2^p)^2} \tag{4.7}$$

The projected lateral displacement is the component of the difference between the projected and desired states perpendicular to the direction of the desired path. It is obtained by rotating the vector from x^d to x^p by the desired heading angle θ^d and taking the component perpendicular to the path:

$$y^p(t+T_p) = (x_2^d - x_2^p)\cos\theta^d - (x_1^d - x_1^p)\sin\theta^d \tag{4.8}$$

For a continuous path, this is equivalent to taking the length of the vector, but for paths consisting of a discrete set of points this method results in smaller computational errors due to gaps in the path.

We now model the steering action of the user as a PD controller [46] based on the anticipated lateral displacement feedback cue, with an additional delay δ_H representing the driver's physical reaction:

$$u(t+\delta_H) = K_p y^p(t+T_p) + K_d \dot{y}^p(t+T_p) \tag{4.9}$$

The control signal generated by Eq. 4.9 is continuous and unbounded. However, the gamepad input device is only capable of generating control inputs on the interval $[-1, 1]$, and it was noted in Section 4.7.1 that users tend to issue commands at one extreme of the interval or the other. We can capture both the actuator saturation and the users' tendency to max out the limits of the gamepad by conditioning the control input with a simple threshold ($\mu > 0$):

$$u'(t) = \begin{cases} -1 & \text{if} \quad u(t) \leq -\mu \\ 0 & \text{if } \mu > u(t) > -\mu \\ 1 & \text{if } \mu \leq u(t) \end{cases} \tag{4.10}$$

and using $u'(t)$ as the simulated gamepad steering command issued to the robot. Figure 4.10 shows a block diagram of the overall steering control loop.

4.7.3 Model Parameter Tuning

We focus here on the case in which the robot speed is 1 m/s. To simplify the process of tuning of this model to reflect the driver behaviors measured in the user tests, we can make some assumptions about the parameters. First, we assume that the physical reaction time (δ_H) of the model driver is 200ms, as the gamepad log data indicates that users generally actuated the joystick from its center to its limit within that amount of time. Also, we assume a lookahead time (T_p) of 1250ms. Additionally, we set the threshold for conditioning the control input to $\mu = 0.5$. Therefore the only two parameters left to tune are the control gains K_p and K_d. The gains were tuned by hand to reflect the path-following score and average control input of the users, discussed in Section 4.7.1, for each constant latency case, using a MATLAB model of the robot system in place of the Java simulation. A summary of these constant and tuned parameters is shown in Table 4.4.

Table 4.4: Tuned control gains and parameter values for constant latency cases.

Type	δ [ms]	K_p	K_d	T_p [ms]	δ_H [ms]	μ
A	0	1.7	0.0	1250	200	0.5
B	250	1.6	0.3	1250	200	0.5
C	500	1.3	0.7	1250	200	0.5
D	750	1.0	1.0	1250	200	0.5

For the zero latency case, the K_d value of zero is consistent with vehicle steering models having only proportional feedback to errors in projected lateral displacement [58]. For the scenarios with latency, the ratio of K_p/K_d decreases as the latency increases, demonstrating that the steering model more heavily weighs the projected error for low latency, and relies more on the predicted displacement trend when the delay is high. Intuitively, this reflects the strategy employed by a human teleoperator in the control loop, who must rely more on prediction based on anticipation of the track's features when the latency is high rather than direct visual feedback. Additionally, the decreasing proportional gain reflects the users' increased tolerance for steady state errors in the difficult-to-control high latency cases.

Because the noise n injected into the input command propagates through the robot system, the $\dot{y}^p(t+T_p)$ term can also be quite noisy, significantly affecting the derivative portion of the controller. Therefore, the derivative term is smoothed by averaging the values of $\dot{y}^p(t+T_p)$ over 10 samples (for a controller running at 40Hz).

4.7.4 Model Validation

To test the performance of the steering model, the teleoperation scenarios were run with the gamepad command simulated in real-time by a MATLAB script running the steering model at 40Hz and communicating to the robot simulation via LCM over the gamepad channel. Everything else about the simulation was the same as the

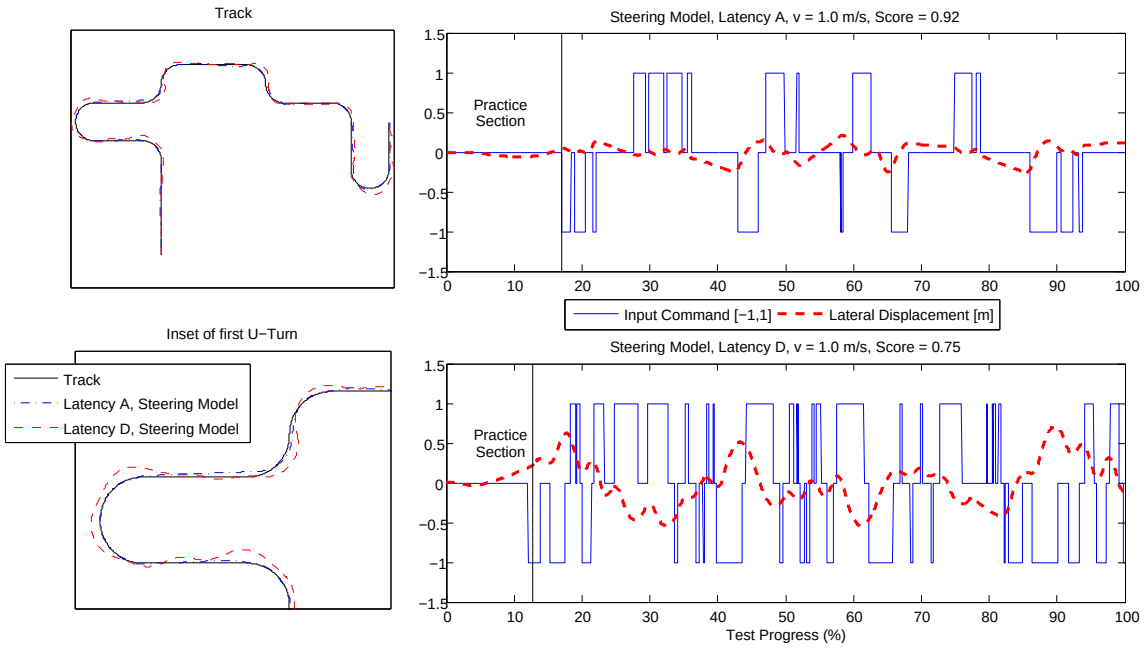


Figure 4.11: Example paths and input profiles of the robot as commanded by the steering model. These paths and inputs show similar qualitative characteristics to those produced by human drivers. Scores were not accumulated during the practice section.

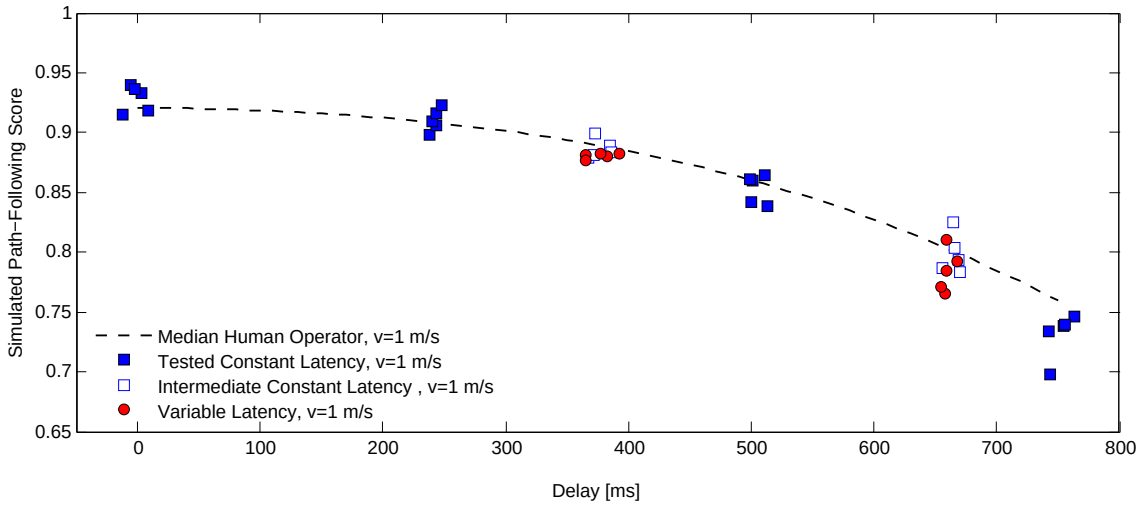


Figure 4.12: Scores of path-following simulations of the robot at a speed of 1 m/s with input commands from the steering model. The model was tuned using the constant latency scenarios from the user trials, and additionally tested with the variable latency scenarios. The gains used in the constant latency cases were linearly interpolated from the tuned gains given in Table 4.4, and the variable latency gains were tuned to the equivalent latency cases shown in Fig. 4.6

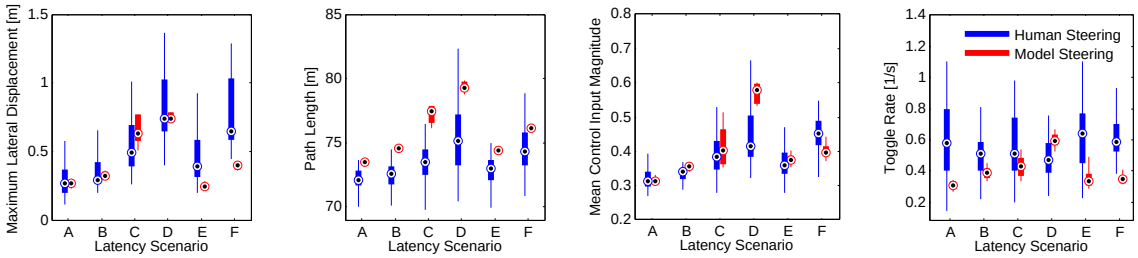


Figure 4.13: Boxplot comparison between the driver model and human users for path characteristics of maximum lateral displacement (overshoot), path length, mean control input magnitude (control effort), and gamepad toggle rate. For readability, the outliers have been removed from the boxplots.

setup with the human operator. Each of latency scenarios A-F were run five times on five different test tracks with a robot speed of 1 m/s. For the variable latency cases, the equivalent constant latency found in Section 4.6.2 for each case was used as an estimated constant latency, and gain values K_p and K_d were determined by linearly interpolating between the values in Table 4.4. Additionally, five trials were run at each equivalent latency in a constant delay scenario to verify that the linear interpolation generates acceptable gain values.

Figure 4.11 shows two example datasets generated by the steering model, which appear similar to the datasets produced by the human drivers shown in Fig. 4.8. Both the saturated input behavior and the overall lateral displacement profiles are qualitatively captured by the steering model.

As shown in Fig. 4.12, the steering model is able to emulate the median path following scores of the of the operators in the user study. Additionally, the simulations at intermediate constant latency values not explicitly measured in the user trials follow the trend of the measured data, indicating that the gains determined by interpolation are acceptable. Finally, the path-following scores of the variable latency cases agree with the scores of the intermediate constant latency tests, meaning that the latency equivalence experienced by the users has been captured in this steering model.

Figure 4.13 shows that the model, as tuned, accurately reproduces the overshoot (maximum lateral displacement) produced by the human driver in constant latency scenarios, but not variable latency cases. Additionally, while they both show the same overall trends, the path lengths for the human operator are consistently shorter than for the steering model. This is likely because the human drivers in the trial were anticipating the turns and took the inside corner of the track more often than the model. If desired, the model could be tuned to be more anticipatory. The steering model also shows good agreement with the human drivers for average control input. Finally, the steering model generally tends to toggle the input command less frequently than the human drivers toggled the joystick. This could also be adjusted for in tuning, but increased accuracy in the toggle rate or path length measures may result in less accuracy for other measures. Additionally, this model does not take into consideration any learning experience that may be gained from repeated trials. Overall however, the driver model appears to be a reasonable representation of a human driver under the conditions tested, and could be used to simulate teleoperator steering responses for evaluation of potential robot designs and technologies.

4.8 Conclusions and Future Work

This chapter presents the results of a 31-subject user study exploring the effects of constant and variable latency on teleoperated steering tasks using a simulated mobile robot receiving input commands from a teleoperator via a computer gamepad. A model of user performance under constant latency was developed, and is shown in Fig. 4.6. The model indicates a sharp decrease in path-following performance for constant latencies above about 500ms, and demonstrates that there is an interaction effect between latency and speed. Figure 4.6 also shows that both variable latency

scenarios tested in this study can be mapped to an equivalent constant latency that is higher than the average delay. Figure 4.5 indicates that the users' sensation of delay as determined by post-trial surveys is consistent with the equivalent latencies determined by the objective scores.

Using the fundamental concepts from automotive steering models, and examining the users' input commands to the simulated robot under different latency conditions, a model of a human teleoperator for steering tasks was developed in Section 4.7.2, tuned in Section 4.7.3, and validated for these tasks in Section 4.7.4. The model, as described in Eq. 4.9, is a PD controller with feedback based on the projected lateral displacement of the robot. The tuning of the model gains for different latency scenarios reflects the real-world control strategies that users employ when adapting to system latency.

This chapter raises new questions about the relationships between system latency and operator performance. Thus, one area of future work is further exploration of the possible mapping between variable latency and equivalent constant latencies. It remains to be studied under which conditions of latency distribution, task type and difficulty, and output measures such an equivalence may exist. Additionally, more work can be performed on teleoperation driver models, including the validation of the steering model developed in this work by using more varied track configurations and latency scenarios, as well as more realistic robot simulations. While one clear extension of this work is the development of longitudinal and/or combined driver models for teleoperated mobile robots, it may also be possible to model teleoperators similarly in other contexts, such as pointing or object manipulation tasks.

CHAPTER V

Conclusions and Future Work

Teleoperated mobile robots are used every day in scenarios that are too dangerous or difficult for humans to be present, but which require human judgment and decision-making skills. However, finite network bandwidth limits the fidelity of communications between the robot and operator, leading to multiple issues affecting the operator's ability to perform tasks in a remote environment. The overall effect of these issues is that teleoperation is a slow and difficult process. Many potential design solutions exist to help reduce the impact of such issues, but each solution has associated costs as well as anticipated benefits, and it is not always clear which components and designs should be chosen to most effectively improve system performance. This dissertation introduces a systematic framework for evaluating design choices for teleoperated mobile robot systems. Because this framework relies on models of human operator behavior, several demonstrative models for use in the framework have been developed to predict human performance for representative teleoperation tasks under varying system conditions.

5.1 Contributions

This dissertation's contributions are applicable to the design and implementation of teleoperated robot systems. First, the underlying factors limiting teleoperation

performance are identified and organized by their location in the teleoperation system feedback loop. Second, an optimization framework for robot speed and performance is presented. The framework offers a systematic method of determining the cost-effectiveness of implementing potential solutions for mitigating the limiting factors. Two examples are provided for a teleoperated unmanned ground vehicle (UGV): one demonstrating the methodology of the framework itself, and a second giving an example of model development for operator detection distance.

A Master-Slave (MS) manual input device and Mixed Reality (MR) visualization software for use in teleoperated mobile manipulation were developed and tested. An evaluation of teleoperator performance with these interfaces, using task completion time and accuracy as a metric, indicated that the MS input method resulted in better performance than a traditional computer gamepad. However, the MR visualization resulted in reduced performance as compared to a purely video visual feedback. Additionally, users that were adept at the task overall received less of a performance boost from the MS interface than did the users who struggled with the task.

Another set of user trials was performed for a simulated teleoperated steering task under constant and variable latency conditions. A model of user performance was developed from these trials and it was found that under the conditions of the simulation, latency with a variable distribution could be mapped to an equivalent constant latency, the magnitude of which is greater than the mean value of the variable delay. This mapping was consistent for both objective and subjective measures as well as robot speed. Finally, a teleoperator steering model was developed, tuned, and validated with the test data. This model is capable of generating realistic human-like commands to a robot for use in development and testing of teleoperated robots without the need for real-time human testing.

5.2 Future Work

The contributions of this dissertation reveal the potential for further research that is a combination of model development and validation as well as development and standardization of performance metrics.

5.2.1 Teleoperator Driving Models

This work has introduced the first driver model specifically designed for teleoperated steering tasks. However, there is much more research that could done in this domain. First, this model was developed under a limited set of test conditions, so more tests could be performed to validate the model under different latency conditions, track difficulties, and levels of scenario realism (including teleoperation of a physical robot). Additionally, this steering model is specific to a computer gamepad, so it would be of interest to develop models using other common input devices for comparison.

Using the fundamental concepts used in automotive driver modeling, combined lateral/longitudinal driver models for teleoperated robots could be developed that would be able to emulate both steering and braking/acceleration actions of a human user. It may also be possible to try to develop operator models for non-driving tasks, such as manipulator arm pointing and positioning.

5.2.2 Teleoperator Performance Models

One of the most straightforward directions for future work resulting from this dissertation is the continued development of models predicting human operator performance in teleoperation scenarios. In this work, we have developed models relating detection distance to speed and video resolution, task completion time to type of manual and visual user interface, and path-following ability to latency. However,

for the optimization framework presented here to be used widely, more performance models must be developed for both navigation and manipulation tasks. The development of a library of human-performance models relating other relevant factors (such as relating path-following ability to video frame rate) could enable researchers and designers to more easily use the optimization framework developed in this dissertation without having to create their own underlying models.

5.2.3 Performance Metrics

Tied closely to the development of teleoperator performance models is the development and standardization of the performance metrics described by such models. Current performance measures for teleoperated tasks vary widely [55], and having a set of standard metrics is essential for researchers to effectively share their modeling efforts and compare potential interface design options. However, research must be done to effectively choose the most valuable metrics to use as standards.

Additionally, aggregate or combined performance metrics could be used. For example, Accot and Zhai's steering law [1] in Eq. 4.1 contains a difficulty index based on spatial constraints. It could be investigated whether factors such as latency, video quality and video frame rate could be incorporated into this difficulty index, or if these effects could be included in the law as corrective terms.

5.2.4 Presence

Presence in the context of teleoperation is a subjective experience defined as the sensation of being at the site of the remote device [66]. Witmer and Singer have outlined and categorized the factors that contribute to a sense of presence in a virtual environment [66], many of which overlap with the factors that affect perception and manipulation abilities in teleoperated tasks. Thus, it is compelling to try to use the

concept of presence as an aggregate metric for evaluating teleoperated robot system effectiveness. Instead of measuring how a given design decision affects individual performance metrics, the features' effects on a user's sense of presence could be measured, possibly affording a more holistic user model. With such a model, it may be able to construct a better cost-benefit analysis for improving teleoperator performance.

In the two user studies discussed in this dissertation, data on operator sense of presence was collected in the form of Likert item questionnaires and interviews. An immediate direction of future work is the analysis of this data to assess how the operators' sense of presence is correlated with task performance in both of these studies. Further user testing could be performed on other recently developed interfaces [59].

5.2.5 Variable Latency

Latency plays a key role in the teleoperation control loop, so understanding the relationship between latency and operator performance is crucial to developing better teleoperated robot systems. The results of Chapter IV raise new questions about how variable delays can be modeled and mitigated in the control loop. Future work should be performed to determine under what conditions of latency distribution, task type, and task difficulty, as well as for what performance measures variable latency can be treated as an equivalent constant latency. If widely applicable, this mapping has the potential to greatly simplify the process of characterizing human operator response to delays.

5.3 Incorporating Autonomy: Keeping Humans in the Loop

As autonomous navigation and manipulation algorithms improve, robots are getting better at the low-level controls in the teleoperation loop currently provided by

the operator. In the future, robots will be able to complete many missions completely autonomously, even when facing very complicated tasks. When this time comes, what is the role of the teleoperator, if any? Certainly some rote tasks currently performed with teleoperation that will have the operator removed. However, for missions in which the goal is to gather information for real-time human interpretation, the user interface and control loop will be just as important as they are today with direct teleoperation.

Figure 2.2 on page 10 shows that a teleoperation system has an inner loop/outer loop control architecture, with the human in the outer loop, and autonomy in the inner loop. This structure remains unchanged even as more low-level processing takes place in the "Autonomous Behaviors" block. If we consider the implementation of a particular autonomous behavior as a potential design choice, the optimization framework can be used to determine if implementing the behavior is cost-effective for a given task. Therefore, the framework developed in this dissertation is not only general enough to be applicable to a highly autonomous system, it can actually be used facilitate an efficient shift from teleoperation to more autonomous robots.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Johnny Accot and Shumin Zhai. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, CHI '97, pages 295–302, New York, NY, USA, 1997. ACM.

[2] Advanced Micro Peripherals. Embedded video product selector guide, 2012. `http://www.amp-usa.com/selector.php`.

[3] G. Ahuja, G. Kogut, E. B. Pacis, B. Sights, D. Fellars, and H. R. Everett. Layered augmented virtuality. *Proceedings of the 13th IASTED International Conference on Robotics and Applications*, pages 258–263, 2007.

[4] D. M. Anand, J. R. Moyne, and D. M. Tilbury. Performance evaluation of wireless networks for factory automation applications. In *Proceedings of the IEEE Conference on Automation Science and Engineering (CASE)*, pages 340–346, Bangalore, 2009.

[5] Dhananjay Anand, Malvika Bhatia, James Moyne, Wajita Shahid, and Dawn Tilbury. Wireless test results booklet. Technical report, University of Michigan ERC/RMS, 2010.

[6] APRIL Robotics Laboratory. APRIL Laboratory : Autonomy * Perception * Robotics * Interfaces * Learning. http://april.eecs.umich.edu, 2012.

[7] A. Bechar, Y. Edan, and J. Meyer. Optimal collaboration in Human-Robot target recognition systems. In *IEEE International Conference on Systems, Man and Cybernetics, 2006. SMC '06*, volume 5, pages 4243–4248. IEEE, October 2006.

[8] P. Ben-Tzvi, A.A. Goldenberg, and J.W. Zu. Design, simulations and optimization of a tracked mobile robot manipulator with hybrid locomotion and manipulation capabilities. In *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008*, pages 2307–2312, 2008.

[9] Johann Borenstein, Russ Miller, and Adam Borrell. Teloptrak: Heuristics-enhanced indoor location tracking for tele-operated robots. *The Journal of Navigation*, 65(02):265–279, 2012.

[10] A.P. Bowling, J.E. Renaud, J.T. Newkirk, N.M. Patel, and H. Agarwal. Reliability-based design optimization of robotic system dynamic performance. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3611–3617, 2006.

[11] L. Brito Palma, F. Vieira Coito, and P. Sousa Gil. Low order models for human controller - mouse interface. In *2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES)*, pages 515–520, 2012.

[12] Michael Brown, Aidan Kehoe, Jurek Kirakowski, and Ian Pitt. Beyond the gamepad: HCI and game controller design and evaluation. In Regina Bernhaupt, editor, *Evaluating User Experience in Games*, Human-Computer Interaction Series, pages 209–219. Springer London, January 2010.

[13] J. Carlson and R.R. Murphy. How UGVs physically fail in the field. *Robotics, IEEE Transactions on*, 21(3):423–437, 2005.

[14] J.Y.C. Chen, E.C. Haas, and M.J. Barnes. Human performance issues and user interface design for teleoperated robots. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1231–1245, 2007.

[15] J.Y.C. Chen and J.E. Thropp. Review of low frame rate effects on human performance. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 37(6):1063–1076, 2007.

[16] K. Chintamani, A. Cao, R. D Ellis, and A. K Pandya. Improved telemanipulator navigation during Display-Control misalignments using augmented reality cues. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(1):29–39, January 2010.

[17] Toby Collett, Joshua Hartnoll, and Bruce Alexander MacDonald. An augmented reality debugging system for mobile robot software engineers. *Journal of Software Engineering for Robotics*, 1(1):18–32, January 2010.

[18] J. Corde Lane, C.R. Carignan, B.R. Sullivan, D.L. Akin, T. Hunt, and R. Cohen. Effects of time delay on telerobotic control of neutral buoyancy vehicles. In *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA '02*, volume 3, pages 2874 –2879, 2002.

[19] Mike Daily, Youngkwan Cho, Kevin Martin, and Dave Payton. World embedded interfaces for Human-Robot interaction. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 5 - Volume 5*, page 125.2. IEEE Computer Society, 2003.

[20] J. Davis, C. Smyth, and K. McDowell. The effects of time lag on driving performance and a possible mitigation. *IEEE Transactions on Robotics*, 26(3):590–593, June 2010.

[21] I.I. Delice and S. Ertugrul. Intelligent modeling of human driver: A survey. In *2007 IEEE Intelligent Vehicles Symposium*, pages 648–651, 2007.

[22] Munjal Desai, Poornima Kaniarasu, Mikhail Medvedev, Aaron Steinfeld, and Holly Yanco. Impact of robot failures and feedback on real-time trust. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, HRI '13, pages 251–258, Piscataway, NJ, USA, 2013. IEEE Press.

[23] J. V Draper. Teleoperator hand controllers: A contextual human factors assessment. Technical report, Oak Ridge National Laboratory, May 1994.

[24] Michael Frigge, David C. Hoaglin, and Boris Iglewicz. Some implementations of the boxplot. *The American Statistician*, 43(1):50–54, February 1989. ArticleType: research-article / Full publication date: Feb., 1989 / Copyright 1989 American Statistical Association.

[25] Scott Green, XiaoQi Chen, Mark Billinghurst, and Geoffrey Chase. Collaborating with a Mobile Robot: An Augmented Reality Multimodal Interface. In *17th IFAC World Congress (IFAC WC2008)*, page 6, 2008.

[26] Scott. A. Green, Mark Billinghurst, XiaoQi Chen, and J. Geoffrey Chase. Human-Robot collaboration: A literature review and augmented reality approach in design. *International journal of advanced robotic systems*, 5(1):1–18, 2008.

[27] Scott A. Green, J. Geoffrey Chase, XiaoQi Chen, and Mark Billinghurst. Evaluating the augmented reality human-robot collaboration system. *Int. J. Intell. Syst. Technol. Appl.*, 8(1/2/3/4):130–143, 2010.

[28] E. Guizzo. Fukushima robot operator writes tell-all blog. http://spectrum.ieee.org/automaton/robotics/industrial-robots/fukushima-robot-operator-diaries, August 23 2011.

[29] Peter F. Hokayem and Mark W. Spong. Bilateral teleoperation: An historical survey. *Automatica*, 42(12):2035–2057, December 2006.

[30] Hokuyo. Scanning range finder, 2012. `http://www.hokuyo-aut.jp/02sensor/index.html#scanner`.

[31] Albert Huang, Edwin Olson, and David Moore. LCM: Lightweight communications and marshalling. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

[32] D.B. Kaber, Yingjie Li, M. Clamann, and Yuan-Shin Lee. Investigating human performance in a virtual reality haptic simulator as influenced by fidelity and system latency. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 42(6):1562–1566, 2012.

[33] H. K Keskinpala and J. A Adams. Objective data analysis for a PDA-based human robotic interface. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2809– 2814 vol.3. IEEE, October 2004.

[34] Donghun Lee, TaeWon Seo, and Jongwon Kim. Optimal design and workspace analysis of a mobile welding robot with a 3P3R serial manipulator. *Robotics and Autonomous Systems*, 59(10):813–826, October 2011.

[35] Jason P. Luck, Patricia L. McDermott, Laurel Allender, and Deborah C. Russell. An investigation of real world control of robotic assets under communication latency. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, HRI '06, page 202209, New York, NY, USA, 2006. ACM.

[36] Charles C. MacAdam. Understanding and modeling the human driver. *Vehicle System Dynamics*, 40(1-3):101–134, 2003.

[37] I. Scott MacKenzie and Colin Ware. Lag as a determinant of human performance in interactive systems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, page 488493, New York, NY, USA, 1993. ACM.

[38] P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems*, E77-D(12):1321–1329, December 1994.

[39] P. Milgram, A. Rastogi, and J.J. Grodski. Telerobotic control using augmented reality. *Proceedings, 4th IEEE International Workshop on Robot and Human Communication, 1995. RO-MAN'95 TOKYO*, pages 21–29, 1995.

[40] P. Milgram, S. Zhai, D. Drascic, and J. Grodski. Applications of augmented reality for human-robot communication. *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3:1467–1472, 1993.

[41] R. Murphy, J. Kravitz, S. Stover, and R. Shoureshi. Mobile robots in mine rescue and recovery. *IEEE Robotics & Automation Magazine*, 16(2):91–103, June 2009.

[42] R. R. Murphy, K. L. Dreger, S. Newsome, J. Rodocker, E. Steimle, T. Kimura, K. Makabe, F. Matsuno, S. Tadokoro, and K. Kon. Use of remotely operated marine vehicles at Minamisanriku and Rikuzentakata Japan for disaster recovery. *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 19–25, November 2011.

[43] Katta G. Murty. *Optimization for decision making: linear and quadratic models*, volume 137 of *International series in operations research & management science*. Springer, New York, 2010.

[44] Phuoc-Nguyen Nguyen-Huu, Josh Titus, Dawn Tilbury, and A. Galip Ulsoy. Reliability and failure in unmanned ground vehicle (UGV). Technical Report 2009-1, University of Michigan Ground Robotics Research Center, Ann Arbor, Michigan, February 2009.

[45] Curtis W. Nielsen. *Using augmented virtuality to improve human-robot interactions.* PhD thesis, Brigham Young University, 2006.

[46] Norman S. Nise. *Control Systems Engineering.* John Wiley & Sons, fourth edition, 2004.

[47] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[48] Andriy Pavlovych and Wolfgang Stuerzlinger. Target following performance in the presence of latency, jitter, and signal dropouts. In *Proceedings of Graphics Interface 2011*, GI '11, page 3340. Canadian Human-Computer Communications Society, 2011.

[49] J Reason. Understanding adverse events: human factors. *Quality in health care: QHC*, 4(2):80–89, June 1995. PMID: 10151618.

[50] John R. Rogers. Low-cost teleoperable robotic arm. *Mechatronics*, 19(5):774–779, August 2009.

[51] Richard Rouse, III. What's your perspective? *SIGGRAPH Comput. Graph.*, 33(3):912, August 1999.

[52] K. Ruffo and P. Milgram. Effect of stereographic + stereovideo 'tether' enhancement for a peg-in-hole task. *IEEE International Conference on Systems, Man and Cybernetics*, 2:1425–1430, 1992.

[53] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, March 1978. Mathematical Reviews number (MathSciNet): MR468014; Zentralblatt MATH identifier: 0379.62005.

[54] T. B Sheridan and W. R Ferrell. Remote manipulative control with transmission delay. *IEEE Transactions on Human Factors in Electronics*, HFE-4(1):25– 29, September 1963.

[55] Aaron Steinfeld, Michael Lewis, Terrence Fong, and Jean Scholtz. Common metrics for human-robot interaction. *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, pages 33—40, 2006.

[56] S. Suganuma, M. Ogata, K. Takita, and S. Hirose. Development of detachable tele-operation gripper for the walking robot. In *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings*, volume 4, pages 3390– 3395 vol.3. IEEE, October 2003.

[57] S. Thorpe, D. Fize, C. Marlot, et al. Speed of processing in the human visual system. *Nature*, 381(6582):520–522, 1996.

[58] D. Toffin, G. Reymond, A. Kemeny, and J. Droulez. Role of steering wheel feedback on driver performance: driving simulator and modeling analysis. *Vehicle System Dynamics*, 45(4):375–388, 2007.

[59] Peter Turpel, Bing Xia, Xinyi Ge, Shuda Mo, and Steve Vozar. Balance-arm tablet computer stand for robotic camera control. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 241–242. IEEE Press, 2013.

[60] AY Ungoren and H Peng. An adaptive lateral preview driver model. *Vehicle System Dynamics*, 43(4):245–259, 2005.

[61] D. W. F. van Krevelen and R. Poelman. A Survey of Augmented Reality Technologies, Applications and Limitations. *The International Journal of Virtual Reality*, 9(2):1–20, June 2010.

[62] Steve Vozar and Dawn M. Tilbury. Augmented reality user interface for mobile robots with manipulator arms: Development, testing, and qualitative analysis. In *Proceedings of the Computers and Information in Engineering Conference (CIE)*, August 2012.

[63] Steve Vozar and Dawn M Tilbury. Improving teleoperated robot speed using optimization techniques. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 249–250. IEEE Press, 2013.

[64] Steven Vozar and Dawn M. Tilbury. Augmented reality user interface for mobile ground robots with manipulator arms. In *Proceedings of the SPIE*, volume 7878, page 18, January 2011.

[65] Steven Vozar and Dawn M Tilbury. Improving UGV teleoperation performance using novel visualization techniques and manual interfaces. *Proceedings of SPIE*, 8387:838716, 2012.

[66] Bob G. Witmer and Michael J. Singer. Measuring presence in virtual environments: A presence questionnaire. *Presence: Teleoperators and Virtual Environments*, 7(3):225–240, 1998.

[67] He Xu, Dawei Tan, Zhenyu Zhang, Zhenguo Gao, Gaoliang Peng, and Chao Li. Trade-offs design of mobile robot based on multi-objective optimization with respect to terramechanics. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2009. AIM 2009*, pages 239–244, 2009.

[68] F. Zeiger, N. Kraemer, M. Sauer, and K. Schilling. Challenges in realizing ad-hoc networks based on wireless LAN with mobile robots. In *6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008*, pages 632–639. IEEE, April 2008.

[69] Shumin Zhai, Johnny Accot, and Rogier Woltjer. Human action laws in electronic virtual worlds: An empirical study of path steering performance in VR. *Presence: Teleoperators and Virtual Environments*, 13(2):113–127, April 2004.

[70] Lelai Zhou, Shaoping Bai, and Michael Rygaard Hansen. Integrated dimensional and drive-train design optimization of a light-weight anthropomorphic arm. *Robotics and Autonomous Systems*, 60(1):113–122, January 2012.