Modeling and Improving Teleoperation Performance of Semi-Autonomous Wheeled Robots

by

Justin Storms

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2017

Doctoral Committee:

Professor Dawn Tilbury, Chair
Professor Brent Gillespie
Associate Professor Odest Chadwicke Jenkins
Assistant Professor Ramanarayan Vasudevan

This dissertation is dedicated to my parents,

Gordon and Laurie Storms,

for their endless support.

ACKNOWLEDGEMENTS

I would like to thank the Department of Mechanical Engineering and the Automotive Research Center (ARC) at the University of Michigan, with funding from government contract DoD-DoA W56HZV-14-2-0001 through the US Army Tank Automotive Research, Development, and Engineering Center (TARDEC), for their financial support during my doctoral work. Through the ARC, I was very fortunate to have the encouraging guidance of David Daniszewski and Paul Muench of TARDEC. Additionally, Mitch Rohde and Steve Rohde of Quantum Signal were very generous in the resources and advice that they provided throughout my degree.

I was fortunate to have a great group of colleagues in the ARC and in the lab. They listened to countless practice presentations, provided critical feedback, and were always willing to beta test my robot studies. In particular, my labmate, Steve Vozar was very helpful in guiding me early on in my doctoral program. Additionally, I'd like to thank Kevin Chen and Matthew Ko for working alongside me to setup many of the user tests discussed in this dissertation.

I am deeply grateful for the guidance and mentorship from my advisor, Professor Dawn Tilbury. Dawn's advising truly made it feel like she puts her students' futures first. She consistently demonstrated how to develop strong technical skills, serve as a mentor, and be a leader.

I'm very thankful for the friends I made in Ann Arbor. The time we spent together is really what made Ann Arbor feel like home for the last few years.

Last, but not least, I owe much gratitude to my family. My parents, although they did not always understand what I was doing, have supported me in all of my academic and professional pursuits.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Modeling and Improving Teleoperation Performance of Semi-Autonomous Wheeled Robots

by

Justin Storms

Chair: Dawn Tilbury

Robotics and unmanned vehicles have allowed us to interact with environments in ways that were impossible decades ago. As perception, decision making, and control improve, it becomes possible to automate more parts of robot operation. However, humans will remain a critical part of robot control based on preference, ethical, and technical reasons. An ongoing question will be when and how to pair humans and automation to create semi-autonomous systems. The answer to this question depends on numerous factors such as the robot's task, platform, environment conditions, and the user. The work in this dissertation focuses on modeling the impact of these factors on performance and developing improved semi-autonomous control schemes, so that robot systems can be better designed. Experiments and analysis focus on wheeled robots, however the approach taken and many of the trends could be applied to a variety of platforms.

Wheeled robots are often teleoperated over wireless communication networks. While this arrangement may be convenient, it introduces many challenges including time-varying delays and poor perception of the robot's environment that can lead

to the robot colliding with objects or rolling over. With regards to semi-autonomous control, rollover prevention and obstacle avoidance behaviors are considered. In this area, two contributions are presented. The first is a rollover prevention method that uses an existing manipulator arm on-board a wheeled robot. The second is a method of approximating convex obstacle free regions for use in optimal control path planning problems.

Teleoperation conditions, including communication delays, automation, and environment layout, are considered in modeling robot operation performance. From these considerations stem three contributions. The first is a method of relating driving performance among different communication delay distributions. The second parameterizes how driving through different arrangements of obstacles relates to performance. Lastly, based on user studies, teleoperation performance is related to different conditions of communication delay, automation level, and environment arrangement. The contributions of this dissertation will assist roboticists to implement better automation and understand when to use automation.

CHAPTER I

Introduction

1.1 Motivation

Robotics has allowed us to reach places and environments that would have been otherwise inaccessible to humans. Rapidly developing technology in sensors, computing, and algorithms has allowed robots to perceive their environment, think, and execute actions more effectively than ever. These advancements have created new applications for robotics and has shifted existing robots from being controlled manually to becoming autonomous or some hybrid between manual and fully autonomous. Semi-autonomous is a term used to describe this hybrid between manual and fully autonomous control and it requires integration of inputs from a human operator and autonomous controller.

As autonomy capabilities improve, one may wonder why a human operator would be involved in the robot control at all. Reasons to keep human operators involved in robot control in the control loop include:

1. **Preference:** The human operating the robot may prefer to have some level of involvement in its control. Or human bystanders in the robot's environment may prefer to have a human involved in its control. For example, in telepresence robots, human operators want to be directly involved in the control of the robot so that they can feel more present in the robot's environment [105].

2. **Ethical or Legal:** In robot operation scenarios that require difficult ethical decisions, a human operator may be required to remain actively involved. Or operation scenarios that have laws against autonomous operation may require a human operator. For example, fault for fatalities in autonomous vehicles leaves many open questions from an ethical and legal standpoint. Companies like Tesla have semi-autonomous control systems that can adjust speed and heading to avoid collisions, but still requires regular attention and inputs from a human in the driver's seat [56].

3. **Technology Limitations:** Despite sensors, computing, and algorithms continuing to improve, certain robotic applications are still to difficult to address with full autonomy. It could be that a robot operation scenario arises that autonomy has not been developed to address and there is not enough time to develop automation to address the challenge, such as in a search and rescue mission after a disaster. Or it could be that the robot's environment or task requirement is too complicated to provide a cost effective solution.

Regardless of the reason, semi-autonomous control requires some level of input from both the human operator and automation. Figure 1.1 shows a generic system setup for teleoperation of semi-autonomous robots considered in this dissertation. Throughout this dissertation, teleoperation refers to control of a robot without direct line of sight to the robot in its environment. In other words, information exchanged between the human operator and robot is passed over a communication network.

Each component of the diagram in Figure 1.1 has an impact on the overall system performance. Furthermore, there are also interactions between the effects of each component. For example, communication delay may affect performance for some human operators more than others. Or automation performance may depend on the environment. Much of the prior research has considered the elements in Figure 1.1 individually without looking at their interactions. A more detailed summary of this

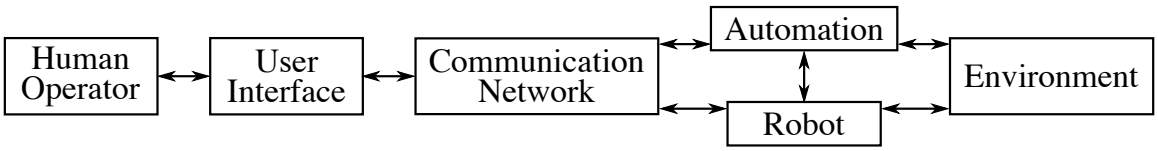

Figure 1.1: Semi-autonomous robot system diagram.

prior work and research gaps is discussed in Section 2.2.

Performance can be measured using objective measures such as average speed, path following error, number of collisions; or using subjective measures such as operator workload, sense of presence, etc. Automation is often added to robotic systems to improve these performance measures. In this dissertation, unmanned wheeled vehicle platforms are the primary focus of the analysis and experiments performed. Some of the challenges with these platforms is that they are restricted to operate at low speeds [1, 115] and are more likely to experience rollover events [106] or collide with obstacles [76].

With regards to these challenges, this dissertation makes advancements in automation methods that improve performance, and develops formalized methods to describe how the factors in Figure 1.1 interact to impact performance. The results can be used to design better semi-autonomous robot systems for a variety of applications such as military reconnaissance, search and rescue missions, infrastructure inspection, construction and mining vehicles, telepresence robots, and automotive transportation.

1.2 Contributions

The following five contributions will be presented in this dissertation:

1. **Method of Improving Handling and Preventing Rollover Using an Existing Manipulator Arm:** Mobile platforms with manipulator arms have a higher center of gravity, which causes them to be more prone to rollover and

have reduced maneuverability [106]. While the mobile base is driving around, the manipulator arm is typically kept in a stationary position. However, the manipulator arm can be used to actively move the center of gravity of the vehicle to improve driving performance. A new control law for for high-speed mobile manipulators is developed in this dissertation. The new control law is shown to not only help prevent rollover, but also improve maneuverability of the mobile robot platform. Development of the control law and results are included in Chapter III.

2. **Method of Representing Convex Obstacle Free Regions:** Optimal control based methods have become very powerful tools for calculating feasible robot paths. However, the optimal control problem must be formulated in a way that it can be solved quickly with the limited computing power available on mobile robots. Approximating the obstacle free region as a convex space can make optimization much easier to do in real-time. Chapter V describes a convex approximation of the obstacle free regions that is well suited for highly maneuverable ground vehicles. The method is integrated into a semi-autonomous controller and tested in a user study.

3. **Relationship between Communication Delay Distributions and Teleoperation Performance:** Communication delays are inevitably introduced when the human operator and robot platform are located in different environments. When communication is over wireless networks, as it typically is, the delay can be time-varying, making it more difficult for human operators to compensate for it. Previous studies on communication delay in teleoperation have shown that performance with time-varying delays is worse than constant delays with the same expected value. However, no method for relating time-varying delays of different shapes and distributions in terms of a measure, such as path-

following accuracy, has been found. This dissertation develops a method for quantitatively relating teleoperation performance among time-varying delays having different stochastic distributions. Results are presented for teleoperating driving along a path in Chapter IV.

4. **Difficulty Index Definition for Driving around Obstacles:** Although the minimum time path through a series of obstacles may be the same for a number of arrangements, the time that it takes human subjects to drive through may vary significantly. Prior work has not formalized how driving between obstacle gaps of different widths and locations can be quantitatively related in terms of performance measures such as driving time or number of collisions. Chapter VI presents a difficulty index definition for driving through a series of obstacles.

5. **Relationships between Teleoperation Conditions and Performance:** While the impacts of different conditions with individual elements described in Figure 1.1 have been extensively explored in the literature, little work has been done to understand the interaction between these elements. User studies and statistical models describing the interactions between automation, communication delay, and user interface are presented in Chapter V. Chapter VI also presents statistical models from a user study describing the interaction between automation, communication delay, and environment difficulty. These quantitative relationships describing the interaction between factors will help robot system developers to make better design decisions.

1.3 Dissertation Overview

This dissertation focuses on how performance of mobile robots with human drivers is impacted by factors including communication delay, automation, and environment difficulty. Chapter II discusses the literature relevant to this work and points out

limitations. Chapter III develops a rollover prevention method for unmanned ground vehicles with a manipulator arm. Chapter IV investigates the relationship between different time-varying communication delays and driving performance. Chapter V presents a method of approximating convex obstacle free regions for efficient path planning in semi-autonomous control. It also describes a user study investigating the interaction of communication delay and semi-autonomous control on performance in a search task. Chapter VI defines an index for describing the difficulty in driving around obstacles. A user study investigating the interaction of driving difficulty, communication delay, and semi-autonomous control is also presented. Finally, Chapter VII summarizes the contributions of this dissertation and suggests future research directions.

CHAPTER II

Background

This chapter identifies several key areas of prior robot teleoperation research in the following two sections. The first begins with a discussion of literature on the integration of automation and a human operator through shared control. Then, automation methods for obstacle avoidance and rollover prevention in vehicles and robotics are surveyed. The second section focuses on factors that impact robot teleoperation performance, including communication delay, automation, and task difficulty.

2.1 Automation in Teleoperation

Advancements in sensing/perception and available computational power have resulted in an increasing number of autonomous features on-board teleoperated robots. This section discusses prior work in the area of shared control, where the human operator and automation work together to accomplish the robot's task. Additionally, prior work related to obstacle avoidance and rollover prevention is discussed to motivate the automation methods presented in Chapters III and V.

2.1.1 Shared Control

Figure 2.1 is meant to represent a range of operation modes for robots or vehicles. On the left is pure teleoperation, where the human operator has full control over low

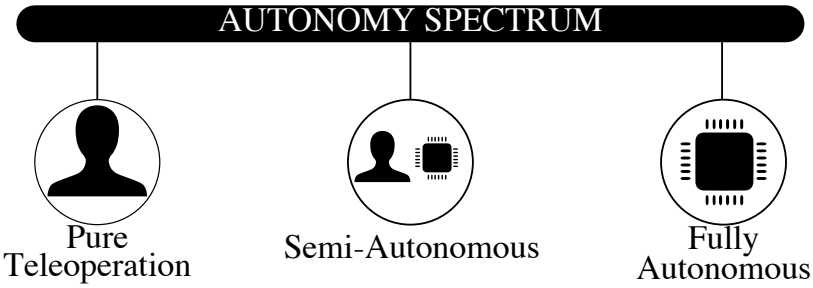

Figure 2.1: Diagram showing autonomy spectrum.

level controls for the robot, such as forward speed and turning rate. On the right side is a fully autonomous robot that can complete a mission without any human intervention. The area in between is a spectrum referred to as semi-autonomy. Many different operation modes fall in the semi-autonomous spectrum. A single robot may have multiple operating modes in the semi-autonomous spectrum that the robot operator or robot itself may select to switch between. This setup is referred to as adjustable autonomy [39]. Determining how and when to switch between different semi-autonomous modes to give the best overall teleoperation system performance is still an area of research [20].

A common approach is to only have a few operating modes and allow operators to select their preferred mode. Semi-autonomous control modes can require very different levels of input from the human operator and there are several different standards for describing the level of automation. For example, the SAE levels of driving automation [24], the NHTSA levels of vehicle automation [2], Autonomous Levels for Unmanned Systems [46], Levels of Automation [33], and Levels of Robot Automation [10]. The control method developed in this dissertation can best be described as shared control in the context of Endsley and Kaber's levels of automation [33]. Results with this shared control automation level will be compared to pure teleoperation and full autonomy, so that the full spectrum is covered.

There has been significant prior work in developing shared control methods. Design of shared control methods is a two part process: 1) an autonomous plan-

ning/control method must be selected/designed, and 2) a control arbitration method must be selected/designed. The control arbitration determines how control is divided between the human operator and autonomy. Control arbitration is often described by the very simple function

$$u = \alpha \cdot u_h + (1 - \alpha) \cdot u_a \tag{2.1}$$

where u is the input applied to the robot, u_h is the human's input, u_a is the autonomy's input, and α is a scalar between 0 and 1. The challenge is selecting a value for α that results in good system performance.

Prior work has calculated α (between 0 and 1) based on how close the human operator input was to an optimal input [34], based on the threat of the human input resulting in a collision [5], and based on a confidence level of what the human operator was trying to do [30]. Other works have in effect considered a discrete set of α values, such as $\alpha = \{0, 1\}$. For example, [11] consider control arbitration that shifts manipulator arm control to autonomy ($\alpha = 0$) when the system has high confidence in how the human operator is trying to move the arm. The semi-autonomous control method also allows the human to shift back control to themselves ($\alpha = 1$) if they do not like what the automation is doing.

Often times the control arbitration is integrated into the automation method itself. That is, the automation may calculate an input for the robot based on the human operator's input. For example, prior work has used human operator inputs to "pull" the robot in a desired direction. Macharet and Florencio consider a shared control method that utilizes artificial potential fields for path planning in a telepresence robot. In Macharet and Florencio's implementation, the human operator's input creates an area of attraction in the direction the operator wants the robot to move [69]. Janabi-Sharifi and Hassanzadeh developed a shared impedence control method for driving

mobile robots. The human operator inputs a desired velocity vector for the robot to a 2D joystick. The automation then calculates a force to apply back to the joystick the human operator is holding based on the location of obstacles around the robot. That is, the force applied to the human operator's hand from the joystick tries to move the robot in a direction away from obstacles [51].

Chipalkatty, Droge, and Egerstedt arbitrate control between the human operator and automation in robot driving by trying to minimize the error between the human and automation inputs, while driving towards a goal position [21]. Their approach formulates shared control as a model predictive control (MPC) problem. However, their formulation does not consider obstacle avoidance. The work in Chapter V describes a similar formulation to [21] and develops a computationally efficient way of representing obstacle-free areas.

2.1.2 Model Predictive Control Based Obstacle Avoidance

In the area of motion planning and obstacle avoidance, prior work has investigated using artificial potential fields [57], vector field histogram [12], dynamic window [38], and model predictive control (MPC) [5]. Artificial potential fields require little computational power, but typically do not consider vehicle kinematic and dynamic constraints. The vector field histogram and dynamic window methods are able to consider some vehicle dynamics, but are limited in the complexity of the model they can consider. MPC based obstacle avoidance methods have become most popular recently due to improvements in computing power and optimization solvers. While MPC based methods tend to be computationally more expensive than previous methods, they are able to use accurate models of the vehicle and its environment to calculate optimal paths. MPC based methods can scale well from simple vehicle models to complex nonlinear models depending on the computing resources available. The control designer has much flexibility over how to define the cost function and what

to define as optimal. Integrating a human operator into the MPC loop is currently an area of active research [93, 35, 6, 21].

Two of the main challenges with human-in-the-loop MPC are 1) estimating human input over the MPC solution horizon, and 2) formulating the MPC problem such that it can be solved fast enough. With regards to estimating the human input over the MPC horizon, some researchers simplify this challenge by assuming that the human input or a position the human is trying to move towards is known [35, 6]. This assumption is reasonable in many situations, *e.g.*, driving in structured highway environments, or when solving the MPC problem for short solution horizons. If the exact goal of the human operator is not known, but a few candidate goals are known, then [52] have suggested a method of probabilistically estimating the goal based on a history of inputs. Other researchers have suggested methods of generating a probabilistic estimate of the human input or desired trajectory using past human operator data [93, 30, 41].

With unlimited computation power, one might imagine formulating the MPC problem to contain a very detailed dynamic model of the robot being controlled and its environment. However, high fidelity dynamic models of robots are often nonlinear and representations of safe regions for robot navigation in an environment with obstacles are often non-convex. More recent advances in optimization methods have made including nonlinear dynamics (as long as the problem is still convex) possible without large sacrifices in computation time [4]. However, non-convex optimization is still challenging to do quickly and it is difficult to guarantee convergence to a globally optimal solution. As a result, some prior work has used optimization methods that focus on finding the best of a group of local minima [41]. Other researchers have focused on how to formulate the MPC problem as a convex optimization problem. Non-convex constraints often arise when mathematically representing the feasible regions for the robot to move in the environment. In general, researchers have tried

to define a single or multiple convex regions that approximate the non-convex space.

Liu *et al.* have suggested partitioning the non-convex feasible region of the robot using triangular sections constructed from each visible obstacle corner [64]. With this set of convex feasible regions, they formulate a multi-stage optimal control problem to handle the transitions between each of the feasible regions and calculate the optimal control input [64]. Similarly, Diets and Tedrake have developed a method (called IRIS) of segmenting a non-convex obstacle free space into several convex regions that approximate the non-convex space [27]. Diets and Tedrake have demonstrated that with IRIS they can formulate the path planning and obstacle avoidance problem for a UAV as a mixed-integer programming problem [28]. Both Liu and Diets have demonstrated that their optimization formulations can be solved on time-scales on the order of seconds using only laptop computing resources [28, 65]. Solve-times on this time scale work well for trajectories that can be pre-planned, but for uninterrupted operation with a human-in-the-loop, faster solve times are required.

Erlien *et al.* use an environmental envelope representation for the feasible region of the vehicle in a highway operation scenario with an obstacle in the road. The environmental envelope applies a constraint on the vehicle's lateral position in the MPC formulation by assuming the vehicle will continue moving forward at a constant speed along the road [35]. Similarly, Anderson *et al.* construct a homotopy of the safe regions that the vehicle can feasibly travel. The homotopy representation is then converted into constraints on the vehicle's lateral position in the MPC problem, based on assumptions of the vehicle's path forward. Both convex approximations by Erlien *et al.* and Anderson *et al.* allow for rapid solving of the MPC problem (multiple times per second) and are well suited for highway type scenarios [6]. However, these methods are not well suited for operation in less structured driving scenarios, *e.g.*, a mobile robot that is not following a road and could be rapidly turning around to head in the opposite direction.

The aerospace industry has similar challenges in path planning and obstacle avoidance that occur in environments without obvious roads or paths to follow. This problem has been approached by using hyperplanes to segment off areas free of obstacles. However, selecting static hyperplane constraints can result in a very conservative representation of the safe region for the vehicle. By allowing the hyperplane constraints to vary over the MPC solution horizon, a less conservative representation of the vehicle's feasible region can be created while still allowing the MPC problem to be solved rapidly in real-time. One such method of varying the hyperplane constraints is by allowing the constraints to rotate around the edges of obstacles they are bounding [85]. More detail about using rotating hyperplane to construct convex feasible regions will be discussed in Section 5.2.2.

2.1.3 Rollover Prevention and Modeling

Prior work has investigated dynamically moving a manipulator arm on-board a mobile base to increase rollover stability. One approach has been to develop strategies to control the position of the Zero Moment Point (ZMP). The ZMP is "defined at the point on the ground about which the sum of all the moments of active force is equal to zero" [47]. As long as the ZMP is inside the polygon formed by the mobile base's contact points with the ground, the mobile manipulator is stable. Huang *et al.* initially developed a motion planner for the manipulator using a potential field that drives the ZMP to the center of the stable region as the mobile base drives around [47]. In later work they developed an improved motion planner that, in addition to maintaining stability, aimed to maintain high manipulability and minimize the manipulator's path acceleration [48].

Kim and Chung investigated a dynamic weight-shifting system that combines the mobile base and manipulator arm subsystems allowing them to maintain rollover stability for both mobile base locomotion and manipulator-oriented tasks [58]. Lee et

al. use invariance control and recursive analytic gradients with the ZMP to increase robustness and computation speed in their dynamic weight-shifting control law [62]. These prior works [47, 48, 58, 62] consider simple mobile bases (without suspension) that are limited to relatively low operation speeds (simulations results were at speeds of approximately 2 m/s). In this dissertation, a high-speed mobile base with a suspension and Ackermann steering is considered (results in simulation up to 15 m/s and in hardware up to 5 m/s).

Patel and Braee have considered rollover prevention of a high-speed mobile base with Ackermann steering. They proposed to add a "tail" to the mobile base to increase maneuverability [82, 83]. The "tail" provides a reaction moment due to a change in angular momentum as it moves to help stabilize the vehicle similar to the function of the arm in [47, 48, 58, 62]. However, the stabilizing reaction moments can only be applied for short durations due to stroke limit. With the method proposed in this dissertation, the control strategy for the manipulator arm is to keep the center of gravity (CG) low and provide a stabilizing moment due to gravity's effect on the arm.

A rollover model is required to conduct the analysis. Modeling rollover of Ackermann steer vehicles is well-researched. Models consider two types of rollover: tripped and untripped. According to the National Highway Traffic Safety Administration, in tripped rollover the vehicle's tires dig into soft soil or strike an object that causes the vehicle to overturn. In untripped, the vehicle does not strike any objects; rollover is induced by a severe maneuver [45].

Untripped rollover models range from very simple one degree of freedom (DOF) to very complex models with 14 DOF or more [14, 22, 63, 86, 94]. Many simple rollover models decouple handling and roll dynamics. Rajamani derived a simple 1 DOF roll model that includes effects of a roll center offset from the vehicle CG [86, Ch.15]. This roll model can be coupled with the 2 DOF handling model also derived by Rajamani

[86, Ch.2]. Cameron and Brennan describe how a 3 DOF model (2 DOF handling model and 1 DOF roll model) can give a good prediction of performance for an actual vehicle [14]. Chen and Peng discuss the accuracy of several simple rollover models including a decoupled 2 DOF roll model with sprung and unsprung masses [17].

A higher fidelity 14 DOF vehicle model developed by Shim and Ghike was shown to give very similar outputs to commercial vehicle simulation packages [94]. Many researchers have used commercial vehicle simulation packages such as CarSim or TruckSim to test controllers for preventing rollover [16, 19]. However, adding in custom dynamics (e.g. effects of a manipulator arm) can be challenging in commercial software packages.

General multibody dynamics simulation packages such as Adams have also been used to develop rollover prevention methods [63]. Chiu developed a vehicle rollover model in Simulink SimMechanics [22] and evaluated a differential braking controller to prevent rollover [23]. SimMechanics is also an effective tool for modeling manipulator arm dynamics [112]. The model selected for analysis of the dynamic weight-shifting method in this dissertation is based on Chiu's model [22] in SimMechanics and will be discussed in Section 3.2.

To understand the effect of the dynamic weight-shifting method, it will be compared using rollover stability metrics both with and without the weight-shifting method. One common measure of rollover stability is related to wheel load transfer. Wheel load transfer metrics, such as those developed by Odenthal *et al.* [79], are simple and intuitive. Other rollover stability metrics include the force-angle stability measure for low speed mobile manipulators [81]. Peters and Iagnemma defined a stability moment measure for mobile robots operating at high speeds [84]. Moosavian and Alipour developed a moment-height stability measure [77]. Chen and Peng developed a time-to-rollover metric [17]. Lastly, energy based rollover stability metrics have been developed based on vehicle roll kinetic energy and tipover potential en-

ergy [78]. This dissertation will primarily consider wheel load transfer metrics when discussing rollover stability due to their effectiveness and intuitiveness.

2.2 Factors Impacting Teleoperation Performance

Factors including communication delay, automation and task difficulty each have been shown to have a significant impact on teleoperation performance. This section discusses prior research that has explored the impact of these factors and identifies gaps in the literature to help put the work presented in Chapters IV-VI in context.

2.2.1 Communication Delay

It is well-established that communication delay has a detrimental impact on teleoperation performance, and time delay is known to be one of the most significant factors affecting remote perception [18]. Sources of delay in a teleoperated robot system include network delays, sensing delays, and processing delays [108].

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 [92]. Since Sheridan's early work [92], many researchers have focused on methods of reducing the impact of communication delay on teleoperation performance. Strategies include using predictors [96], using augmented reality [113], adapting control gains [91], automating subtasks [39], and using different input modalities (*e.g.*, hands-free operation using gestures [72] or voice [111]).

Many of these methods have been shown to be effective at improving teleoperation performance. However, it may be difficult for designers of robot teleoperation systems to decide when it is appropriate to include such methods. That is, when does the improvement in teleoperation performance justify the added cost to include such teleoperation assistance features? To answer that question, an understanding of the relationship between performance and delay is required.

Sheridan's early studies on time-delayed space telerobotics provided a theoretical basis for predicting performance as measured by task completion time under constant delay, assuming operators performed tasks as a series of discrete open-loop movements [92]. Since then, significant research has been done to describe the relationship between constant delay and mobile robot teleoperation performance under conditions ranging from 2D driving [67, 26] to 3D underwater navigation tasks [25]. The directionality of the delay (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 delay was in the robot-to-user direction, but no objective difference in performance was observed [67].

Much of the work investigating the impact of communication delay on wheeled mobile robot teleoperation performance has focused on designing stable haptic control devices [61], incorporating techniques such as asymptotic tracking of position and force [40]. Studies have investigated how communication delay combined with human operator training [51] or additional sensor feedback or assistance [88] impacts mobile robot teleoperation performance. However, these studies only consider constant communication delay.

Real-world communication delay is often time-varying. For example, Ford demonstrated a "remote repositioning" system capable of cross-country vehicle teleoperation. The cellular networks used for communications had variable delays and bandwidth restrictions [74]. Research that has investigated the impact of time-varying delay on performance metrics other than stability (*e.g.*, time to complete a task, number of collisions with obstacles) has only compared time-varying delay with constant delays [67, 26]. Prior work has not suggested how features of a time-varying delay distribution (*e.g.*, mean and variance) could be quantitatively related to other time-varying or constant delays in terms of teleoperation performance metrics beyond stability.

2.2.2 Automation

Prior studies have explored how automation impacts robot operation performance in applications ranging from small commercial telepresence robots [105] to large military tactical vehicles [76]. Both [105] and [76] used steerable waypoint interfaces with obstacle avoidance where operators would control the location of a waypoint that the robot would drive towards. Neither study observed faster driving task completion times with the steerable waypoint feature. Takayama *et al.* did observe fewer collisions between the robot and objects in the environment when using the automation feature.

A more popular approach than the steerable waypoint for shared control allows subjects to directly input velocity or steering commands for the robot. As mentioned in Section 2.1, model predictive control (MPC) is a popular method used for this type of shared control. Several studies have shown that MPC based shared control methods for obstacle avoidance result in faster course completion times when driving robots [6, 21, 55]. Furthermore, [6] has shown that use of haptic feedback improves performance. Video stabilization was also shown to improve course completion time [55].

Macharet and Florencio considered shared control of a telepresence robot with different levels of autonomy. Their shared control method was capable of adjusting low level velocity inputs from the human operator to avoid collisions or suggesting longer paths for the robot to take in the environment. Both modes are available for the human to use during operation. Results from a user study with their control method demonstrated both faster drive times and fewer collisions with the automation feature in comparison to pure teleoperation [69].

Finzi and Orlandini consider a semi-autonomous control method that uses supervisory control. That is, the human operator can give higher level directions or commands to the robot and the automation calculates the lower level commands.

The supervised control method is compared to a pure teleoperation test case and autonomous mode in a user study. For a search task, results show that the supervisory control allowed users to find more items and cover more area than with full autonomy. While number of items found and area coverage were close when comparing the pure teleoperation and supervised control, the number of collisions was much lower with supervisory control [36].

Bruemmer *et al.* explored the impact of *who* is primarily in control. That is, in one operating mode the human operator is primarily in control and the automation intervenes when the human is about to collide with an obstacle. In the second operating mode, the automation is primarily in control and the human operator intervenes when they disagree with the automation's action. Results indicated subjects were able to find more objects in a search task when they were primarily in control and the automation only intervened to avoid collisions [13].

Each of the studies mentioned in this section have considered the impact of semi-autonomous control independent of communication delay and environments of varying difficulties. There are few studies that have investigated the interaction between communication delay and semi-autonomous control. One such study that has investigated this interaction was by [67]. Luck *et al.* found that for a mobile robot driving task with no time delay, course completion time was hardly impacted by the level of autonomy on the robot. This result motivates the questions: under what delay conditions does semi-autonomous control improve performance? With which semi-autonomous control methods/interfaces does time delay degrade performance most?

2.2.3 Task Difficulty

The environment itself can have a large impact on how teleoperated vehicle performance is affected by factors like human operator ability, automation features and communication delay. Several prior works have suggested methods of representing

environment complexity in this context. For example, the Autonomy Levels for Unmanned Systems (ALFUS) framework assigns an overall difficulty to a teleoperation task based on ratings across three categories: 1) Mission Complexity, 2) Environment Difficulty, and 3) Human Independence [75]. The ratings for each category are assigned a number on a scale from 1 to 10 based on a subjective rating.

Durst *et. al* presented a method for predicting the Mission Performance Potential (MPP) for unmanned systems, which can be used to estimate the best possible performance for a given mission at a given autonomy level [31]. The method uses fuzzy inference and logic with data about the unmanned vehicle to calculate an MPP. However, it is difficult to gain much physical intuition about how these factors impact performance.

Lampe and Chatila suggest an entropy based method for evaluating autonomous mobile robot performance in different environments [60]. The robot's environment is broken down into an occupancy grid and the obstacle density in each area of the grid is used to calculate the entropy of the environment. The entropy measurement was shown to be positively correlated with the time taken to drive through an obstacle filled environment [60]. That is, the more randomly distributed obstacles there are in an environment, the longer it takes to pass through it.

The three previous works discussed [31, 60, 75] all consider the robot's environment in a coarse sense. To consider environment difficulty at a finer scale, one can look at how human movement has been modeled. Fitts' Law is an empirical law for describing the time that it takes a person to move their finger (or another object attached to their hand) from one location to another [37]. It says that movement time to a goal position can be described as a linear function of the movement's difficulty index (ID), where the ID is a log function of distance to the goal and width of the goal [37].

Fitts' Law has become very popular in the human factors field, due to its simplicity and utility in predicting movement times. Many researchers have since explored

other conditions and adaptations that result in similar empirical movement time laws. MacKenzie and Ware conducted tests where subjects had to move a mouse cursor to different goal positions under conditions of delay ranging from 8-225 ms. They found that with an additional term for the delay in the ID definition, a linear relationship for movement time versus ID could be produced [70]. Similarly, other researchers have shown that Fitts' Law can be modified to include terms for moving around obstacles in 2D [53] and 3D [107]. Liu and van Liere demonstrated how movement speed varies for subjects tracing out 3D paths of varying curvatures [66]. However, all of these previous works [53, 66, 70, 107] consider humans moving their hand or extensions of their hand without significant dynamics.

Beyond human movement, some researchers have explored developing empirical laws for human control of dynamical systems. For example, Zhai, Accot, and Woljer developed a steering law that predicts the driving speed for a vehicle moving along paths of different shapes and widths [116]. Test subjects were explicitly shown the path that they should follow; much like driving on a road with an Ackermann steer vehicle. Helton also conducted a set of user tests for teleoperation of unmanned ground vehicles around 90^o corners [42]. The results suggest a cornering law that relates the time to navigate around a corner to a function of the width of the vehicle and the width between the walls on the corner.

Of the methods for describing difficulty discussed in this section, many of the ones related directly to unmanned system operation provide too coarse of a difficulty measurement to predict shorter movement times [31, 60, 75]. The methods based on Fitts' Law show promise, but only a few have actually been applied to robot operation [42, 116]. These methods have been applied to following paths of a given width indicated to the user, much like driving on a highway road. Aside from this dissertation work, there are no methods for representing environment difficulty for teleoperation movement in environments with obstacles and no defined path.

CHAPTER III

Dynamic Weight Shifting for Rollover Prevention

3.1 Introduction

Mobile manipulators are typically restricted to slow operating speeds and tame maneuvers. The manipulator arm contributes to a high center of gravity, making the vehicles more prone to rolling or tipping over. Typically, manipulator arms are kept in static positions while the mobile base drives. Previous works discussed in Section 2.1.3 suggest using reaction torques from a manipulator arm's inertia to stabilize roll motion. This strategy is useful in high lateral acceleration turns for very short periods of time. However, for long duration turns at high lateral acceleration, the actuators will neither have enough torque nor large enough stroke to stabilize roll motion. This chapter proposes a new dynamic weight-shifting control method for an existing manipulator arm that aids with long duration, high lateral acceleration turns. The new method keeps the manipulator arm's CG low. The reaction moments from its inertia are small in comparison to the reaction moments due to gravity.

This chapter is based on publications [103, 101]. The remainder of this chapter is organized as follows. Section 3.2 describes the dynamic weight-shifting method and how it is modeled in both a Linear and Nonlinear Model. Section 3.3 compares simulation results with the Linear and Nonlinear Models. Section 3.4 presents a sensitivity analysis of manipulator arm parameters, then describes how dynamic

weight-shifting reduces vehicle roll motion and improves maneuverability with both nonlinear simulation and hardware experiments. Lastly, Section 3.5 summarizes the conclusions.

3.2 Vehicle and Manipulator Model Description

In this work, a high-speed UGV with a manipulator arm attached to it is considered. This type of UGV could be used for a variety of tasks including scouting or retrieving objects. Typically, manipulator arms are kept in a static position during driving tasks, so the dynamic models for driving and manipulation can be developed independently. However, the dynamic weight-shifting method in this dissertation will need to capture the interaction of moving the manipulator arm while driving.

The UGV base will be modeled as a vehicle with front-wheel Ackermann steering and rear-wheel drive - a typical setup for high-speed UGVs. The manipulator arm will modeled as a two-link arm with revolute joints and an end effector - an arm representative of those used to retrieve objects.

Two different models will be compared and used to carry out the analysis. The first is a simple 3 DOF linearized model (referred to as the Linear Model) that decouples the handling (2 DOF) and roll dynamics (1 DOF). The Linear Model is used to gain physical intuition of the effects of the manipulator arm and could be used in model-based control methods.

The second is an 11 DOF Nonlinear Model developed in SimMechanics from a vehicle model by Chiu [22]. Chiu's model was chosen because of its high fidelity and flexibility to add manipulator arm dynamics.

3.2.1 Assumptions and Definition for Rollover

In this work terrain roughness is neglected and it is assumed that the mobile manipulator is operating on a flat smooth surface similar to that of a paved road.

The inputs given to each model are a steering angle δ and the forward velocity of the vehicle V_x (assumed to be kept constant in the Linear Model). It is assumed that the vehicle roll angle ϕ can be estimated on-board (this estimate of ϕ will be used to control manipulator arm joint angles).

Additionally, since terrain roughness is ignored, only untripped rollover is considered and the *critical rollover condition* is defined to be when one of the wheels lifts off the ground (when the normal force F_z on one of the tires becomes zero). The models used are only valid when all wheels are on the ground.

3.2.2 Linear Model

A linear model was first developed to gain physical intuition of the effects of automatically moving a manipulator arm on a vehicle during turning. A simple 2 DOF handling model was chosen to determine the lateral acceleration output for a given steering input. The vehicle handling model assumes a constant forward velocity V_x and linear tire model. The input is front wheel steering angle δ and output is lateral acceleration a_y.

The lateral acceleration calculated from the handling model is input into a simple 1 DOF model used to describe the roll dynamics [17]. The manipulator arm is treated as an element that provides reaction forces and a moment to the vehicle at its point of contact with the vehicle. The 1 DOF roll model has an input of lateral acceleration a_y and an output of roll angle ϕ. Both the handling and roll dynamics will be presented in state-space form $\dot{x} = Ax + Bu$, $y = Cx$ with state x, input u, output y, state matrix A, input matrix B, and output matrix C defined in the next two subsections.

3.2.2.1 Linear Handling Model

Figure 3.1 shows the free body diagram for the handling model. Coordinate frame x_0, y_0, z_0 represents the world fixed frame and frame x_1, y_1, z_1 is attached to the vehicle

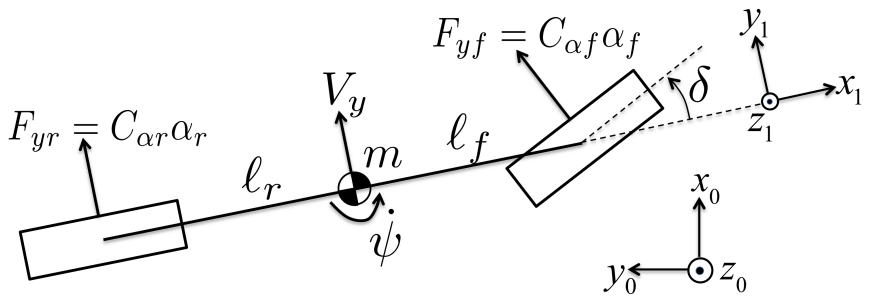

Figure 3.1: Handling dynamics 2 DOF bicycle model.

body. The vehicle has yaw inertia I_z, mass m with the CG located a distance ℓ_f from the front wheel and ℓ_r from the rear wheel. Thus, the wheelbase is $\ell = \ell_f + \ell_r$. Energy stored in this system is in the form of kinetic energy, so states of lateral velocity V_y and yaw rate $\dot{\psi}$ are the only two needed, $x_{hand} = \begin{bmatrix} V_y & \dot{\psi} \end{bmatrix}^T$. The input to the system is the front steering angle δ. The dynamic equations of motion for this planar model result from summing up the moments about the body center of gravity (CG) and adding up the forces in the lateral direction of the vehicle fixed frame.

$$\begin{aligned} m\left(\dot{V}_y + \dot{\psi}V_x\right) &= F_{yf}\cos\delta + F_{yr} \\ I_z\ddot{\psi} &= \ell_f F_{yf}\cos\delta - \ell_r F_{yr} \end{aligned} \tag{3.1}$$

In the Linear Model, forward velocity V_x is assumed to be constant. A linear tire model is used, such that the front and rear lateral forces can be described as $F_{yf} = C_{\alpha f}\alpha_f$ and $F_{yr} = C_{\alpha r}\alpha_r$, respectively. Note that $C_{\alpha f}$ and $C_{\alpha r}$ are the front and rear cornering stiffness. These values are difficult to measure, so they will be tuned to match up the Linear Model handling behavior with the Nonlinear Model in Section 3.3. The values of α_f and α_r are the front and rear tire slip angles defined as

$$\alpha_f = \delta - \tan^{-1}\frac{V_y + \ell_f\dot{\psi}}{V_x}\ ,\ \alpha_r = -\tan^{-1}\frac{V_y - \ell_r\dot{\psi}}{V_x} \tag{3.2}$$

Substituting (3.2) into (3.1) using the linear tire model results in,

$$\begin{aligned} m\left(\dot{V}_y + \dot{\psi}V_x\right) &= C_{\alpha f}\delta\cos\delta - C_{\alpha f}\tan^{-1}\frac{V_y + \ell_f\dot{\psi}}{V_x}\cos\delta - C_{\alpha r}\tan^{-1}\frac{V_y - \ell_r\dot{\psi}}{V_x} \\ I_z\ddot{\psi} &= \ell_f C_{\alpha f}\delta\cos\delta - \ell_f C_{\alpha f}\tan^{-1}\frac{V_y + \ell_f\dot{\psi}}{V_x}\cos\delta + \ell_r C_{\alpha r}\tan^{-1}\frac{V_y - \ell_r\dot{\psi}}{V_x} \end{aligned} \tag{3.3}$$

The above equations were linearized about states $\tilde{V}_y = 0\ m/s$, $\tilde{\dot{\psi}} = 0\ rad/s$ and an input of $\tilde{\delta} = 0\ rad$. Next, the linearized equations were rearranged into state space form to have an output of lateral acceleration a_y with the state matrix A_{hand}, input matrix B_{hand}, and output matrix C_{hand} shown below (Note: these derivations agree with Rajamani's bicycle model derivation [86, Sec. 2.3])

$$\begin{aligned} A_{hand} &= \begin{bmatrix} -\frac{C_{\alpha f}+C_{\alpha r}}{mV_x} & \frac{\ell_r C_{\alpha r}-\ell_f C_{\alpha f}}{mV_x} - V_x \\ \frac{\ell_r C_{\alpha r}-\ell_f C_{\alpha f}}{I_z V_x} & -\frac{{\ell_f}^2 C_{\alpha f}+{\ell_r}^2 C_{\alpha r}}{I_z V_x} \end{bmatrix} \\ B_{hand} &= \begin{bmatrix} \frac{C_{\alpha f}}{m} \\ \frac{\ell_f C_{\alpha f}}{I_z} \end{bmatrix} \\ C_{hand} &= \begin{bmatrix} -\frac{C_{\alpha f}+C_{\alpha r}}{mV_x} & \frac{\ell_r C_{\alpha r}-\ell_f C_{\alpha f}}{mV_x} \end{bmatrix} \end{aligned} \tag{3.4}$$

The transfer function relating the steering angle input to lateral acceleration output can be found from $G_{hand}(s) = \frac{A_y(s)}{\Delta(s)} = C_{hand}(sI - A_{hand})^{-1}B_{hand}$. The steady state gain is found by evaluating $G_{hand,SS} = G_{hand}(i\omega)\big|_{\omega=0}$ and is:

$$G_{hand,SS} = \frac{C_{\alpha f}C_{\alpha r}\ell {V_x}^2}{(\ell_r C_{\alpha r} - \ell_f C_{\alpha f})m{V_x}^2 + C_{\alpha f}C_{\alpha r}\ell^2} \tag{3.5}$$

3.2.2.2 Linear Roll Model

The linear roll model was derived based on Figure 3.2. Again, coordinate frame x_0, y_0, z_0 represents the world fixed frame and frame x_1, y_1, z_1 is attached to the main vehicle body. Frame x_2, y_2, z_2 is attached to the manipulator arm link directly con-

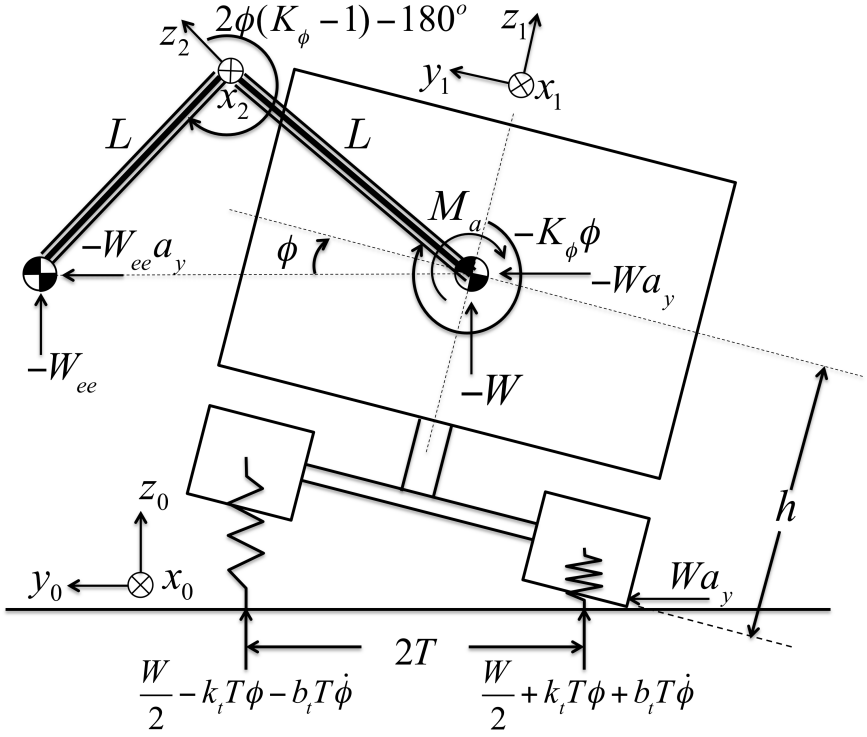

Figure 3.2: Diagram of 1 DOF model used for roll dynamics.

nected to the main vehicle body. All arrows specifying forces and moments are drawn in the positive direction. Figure 3.2 is drawn for a vehicle performing a left turn with positive roll angle and positive lateral acceleration. The physical parameters are described in Table A.1. The vehicle is assumed to have a width of $2T$ and make contact with the ground on its left and right side. Each side of the vehicle roll model has a stiffness k_t and damping b_t that contribute to the vertical loads.

A two-link manipulator arm connected to the vehicle at its CG is considered. The arm is assumed to consist of massless links with length L and an end effector with mass m_{ee}. Thus, the arm has weight $W_{ee} = m_{ee}g$ and increases the total weight of the vehicle to $W = (m_v + m_{ee})g$. The manipulator arm can move independently of the vehicle body. Thus, it is important to model the reaction moments and forces at the point where the arm is connected to the vehicle body to understand how the arm and vehicle body interact. The manipulator arm joints will be controlled such that

the end effector mass is kept at the same height as the vehicle center of gravity h. Controlling the end effector position in this way will cause the moment arm in the z_0 direction to be zero, thus the lateral force $W_{ee}a_y$ will not contribute to M_a.

In order to keep the end effector position at the same height as the vehicle CG, the joint angles of the arm are selected to be in positions proportional to that of the roll angle ϕ of the vehicle. In this case Joint 1 attached to the vehicle is proportional to ϕ by constant $-K_\phi$, where joint angle $-K_\phi\phi$ is measured with respect to the z-axis of the vehicle fixed frame. Joint 2 is attached to the end of Link 1 and is positioned at an angle of $2\phi(K_\phi - 1) - 180^o$ measured with respect to the z-axis of the Link 1 fixed frame.

With the manipulator arm end effector controlled in this way, the reaction moment due to the arm will be the weight of the end effector multiplied by its distance from the vehicle CG: $M_a = -2W_{ee}L\sin(\phi(K_\phi - 1))$. Note the following about this control law:

- Since K_ϕ appears inside the sin term and small angle approximations will be used in the Linear Model, the roll model will be more accurate for small K_ϕ and small roll angles ϕ.

- For $K_\phi = 1$ the arm mass will always be located at the CG of the vehicle and contribute no moment. This case will be referred to as the "arm stationary" case (Arm Stat.).

- For $K_\phi > 1$ the arm mass will apply a stabilizing moment to the vehicle (as it rolls). This case will be referred to as the "arm moving" case (Arm Mov.).

- For $K_\phi < 1$ the arm mass will apply a destabilizing moment to the vehicle as it rolls. Therefore only $K_\phi \geq 1$ is considered.

The energy stored in this system is in the form of kinetic and potential energy, so the states will be roll angle ϕ and roll rate $\dot{\phi}$, $x_{roll} = \begin{bmatrix}\phi & \dot{\phi}\end{bmatrix}^T$. The input to the system

is the lateral acceleration a_y. The dynamic equations of motion for this planar model result from summing up the moments about the body CG and substitution in the expression for M_a.

$$I_x\ddot{\phi} = -2k_tT^2\phi - 2LW_{ee}\sin\left(\phi\left(K_\phi - 1\right)\right) - 2b_tT^2\dot{\phi} + Wha_y \tag{3.6}$$

After applying small angle approximations to (3.6), the result is state matrix A_{roll}, input matrix B_{roll}, and output matrix C_{roll}.

$$\begin{aligned} A_{roll} &= \begin{bmatrix} 0 & 1 \\ \frac{-2k_tT^2 - 2LW_{ee}(K_\phi - 1)}{I_x} & \frac{-2b_tT^2}{I_x} \end{bmatrix} \\ B_{roll} &= \begin{bmatrix} 0 \\ \frac{Wh}{I_x} \end{bmatrix} \\ C_{roll} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \end{aligned} \tag{3.7}$$

The transfer function relating the lateral acceleration input to roll angle output can be found from $G_{roll}(s) = \frac{\Phi(s)}{A_y(s)} = C_{roll}(sI - A_{roll})^{-1}B_{roll}$. The steady state gain is found by evaluating $G_{roll,SS} = G_{roll}(i\omega)\big|_{\omega=0}$ and is displayed in Eqn. (3.8). If the arm mass is considered to be stationary at the vehicle CG, then the arm will provide no moment M_a to the vehicle. This case (Arm Stat.) can be treated as $K_\phi = 1$ causing the $2LW_{ee}(K_\phi - 1)$ term in the denominator of Eqn (3.8) to become zero. The case where the arm is actively moving (Arm Mov.) to provide a stabilizing moment (*e.g.* $K_\phi > 1$) will cause the $2LW_{ee}(K_\phi - 1)$ term in the denominator to be positive.

$$G_{roll,SS} = \frac{Wh}{2k_tT^2 + 2LW_{ee}(K_\phi - 1)} \tag{3.8}$$

3.2.3 Nonlinear Model

The Nonlinear Model description is broken down into the vehicle model adapted from prior work [22] and a custom made manipulator arm model. The vehicle and manipulator arm models were coupled together using Simulink's SimMechanics and SimElectronics tools.

3.2.3.1 Vehicle Model

The multi-body vehicle rollover model used in this simulation was developed and verified by Chiu [22]. The model will be briefly described here starting with the main vehicle body shown in Figure 3.3. Attached to the vehicle body CG is the standard ISO coordinate system with the x-axis forward in the longitudinal direction, the y-axis to the left in the lateral direction and the z-axis upward in the vertical direction [90]. Connected to the vehicle body are the front and rear axles. The front and rear axle bodies are connected by 1 DOF revolute joints each having their own roll stiffness and damping. Connected to each the front and rear axles are two wheel bodies. The two front wheels are each given a yaw DOF in the z-axis direction to accept a steering input angle δ. Each wheel body has a vertical stiffness and damping and interacts with the road through a 6 DOF joint.

The longitudinal and lateral tire-ground contact patch forces are calculated using Pacejka's Magic Formula tire model [80, Ch. 4]. Parameters for the tire model were obtained from Table 4.1 of [80]. The Magic Formula used takes into account the effects of the tire camber angle γ when the vehicle begins to roll. The normal or vertical force on each tire is calculated at each step of the simulation based on its vertical stiffness and damping interaction with the ground. If the position of the bottom of the tire is calculated to be above the ground, this condition corresponds to the tire lifting off. Traction force is applied to the rear wheels and longitudinal speed of the vehicle is controlled using a PID controller.

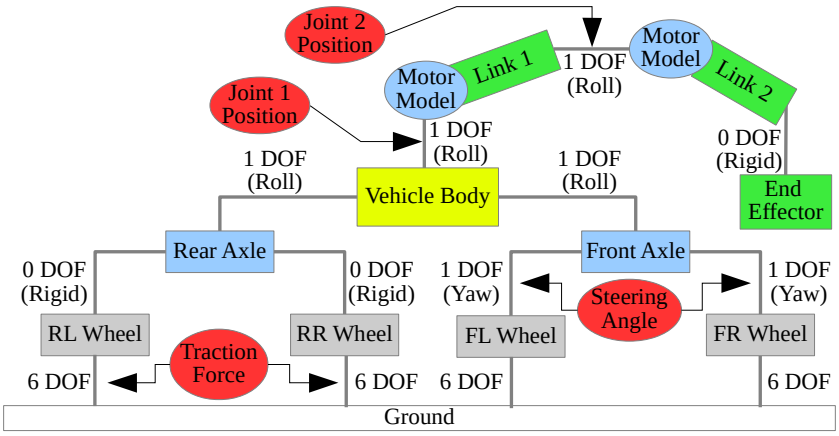

Figure 3.3: Functional diagram of nonlinear vehicle and manipulator arm model (adapted from Chiu [22]).

This model was originally developed with parameters for a full-size commercial van (approximately 2800 kg) in [22]. In order to make this simulation more relevant to smaller UGVs, the vehicle parameters were selected to be similar to that of the scale RC car in Figure 3.4. The RC car weighs approximately 3 kg and is 1:10 scale. Measurements of the RC car that could be easily obtained with standard laboratory equipment were used for the Nonlinear Model parameters (*e.g.* wheelbase, track width, vehicle mass, etc.). Parameters that were difficult to accurately measure were approximated by scaling the commercial van parameters. Based on the mass (1:1000) and length (1:10) scales, the commercial van mass parameters were multiplied by 10^{-3}, length parameters by 10^{-1}, inertia values by 10^{-5} (mass $\times$ length^2), stiffness and damping values by 10^{-4} (mass $\times$ length), etc. Specific values for Nonlinear Model parameters can be found in Table A.1.

3.2.3.2 Manipulator Arm Model

A simple two-link arm will be simulated to represent a manipulator arm on a UGV. Both DOF are revolute joints aligned with the vehicle's roll axis. The arm is representative of a simple manipulator that could be used for picking objects up

Figure 3.4: Modified 1:10 scale RC car with a two-link manipulator arm used for experimental validation.

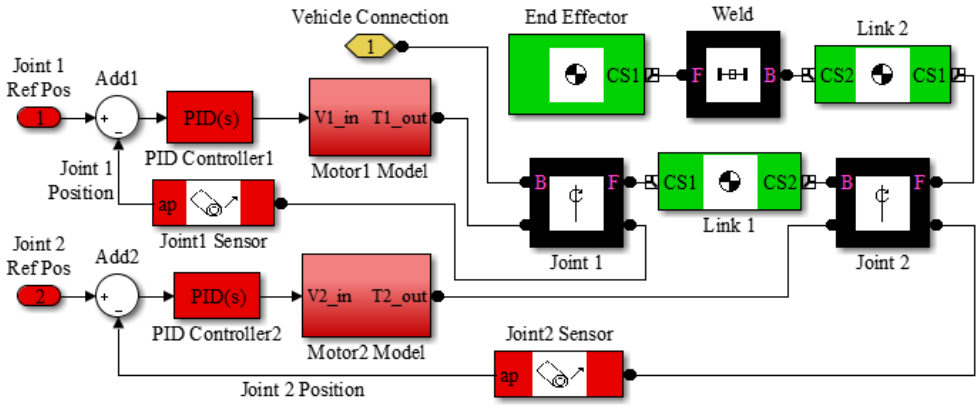

Figure 3.5: SimMechanics model of the manipulator arm.

around the UGV's base. The manipulator arm parameters were selected to match the setup in Figure 3.4.

The manipulator arm was modeled in Simulink using SimMechanics and SimElectronics as shown in Figure 3.5. The arm consists of two links, each connected by a revolute joint, and an end effector mass. The arm links are approximated as rods with mass m_L and inertia $(m_L L^2/12)\cdot\text{diag}([1\ 1\ 0])$. The end effector mass is approximated as a cuboid with mass m_{ee} and inertia $(m_{ee}d_{ee}^2/6)\cdot\text{diag}([1\ 1\ 1])$. Each joint has 1 DOF in the x-axis direction of the vehicle-body-fixed ISO coordinate system. Values for the manipulator arm parameters are listed in Table A.1.

The joints are actuated using closed loop PID position control. Included in the control loop is a DC motor model from SimElectronics with torque saturation τ_m representative of an appropriately sized motor. Each motor tracks a joint position based on K_ϕ and the current vehicle body roll angle ϕ. The motor on the base joint of Link 1 is attached to the vehicle body CG. Thus, reaction forces and torques due to the arm's dynamics are transferred back to the vehicle body.

3.3 Model Comparison

The Linear Model was used to analyze the effects of using a manipulator arm for dynamic weight-shifting during turning maneuvers. The maneuver selected for this analysis was a steering step-like input of magnitude δ with the vehicle moving at constant forward velocity V_x. The steering input profile used for nonlinear simulations in this work has a value of 0 for 1 s, then increases linearly to the final steering angle δ over a duration of 0.2 s and remains constant.

A comparison of the Linear and Nonlinear Model was performed in steady state at various steering inputs and forward velocities. Steering inputs of $\delta = 1^o \rightarrow 5^o$ in 1^o increments were given to the Nonlinear Model at forward speeds $V_x = 2, 4, 8\ m/s$. First the steady state behavior of the handling models was compared, *i.e.* the resulting steady state lateral acceleration for a constant steering input. The result is displayed in Figure 3.6 (a) and (b). As expected the Linear Model yields a linear relationship of a_y vs. δ with slope equal to $G_{hand,SS}$, which increases with increasing V_x.

The Linear Model and Nonlinear Model steady state a_y values line up well at $V_x = 4\ m/s$. At $V_x = 2\ m/s$ the Linear Model predicts slightly higher steady state a_y values. As forward speed increases to $V_x = 8\ m/s$, the Nonlinear Model becomes more nonlinear for steady state a_y values vs. δ. Since the handling and roll models are decoupled in the Linear Model, the steady state a_y values are unaffected by whether

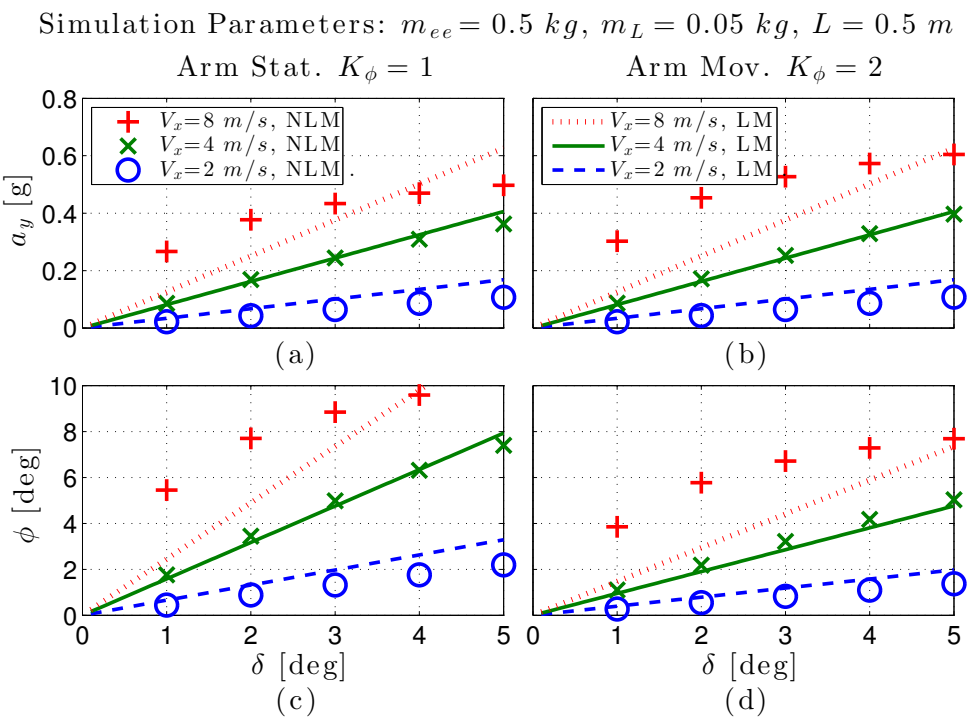

Figure 3.6: Linear Model (LM) and Nonlinear Model (NLM) steady state comparison for various steering inputs and forward velocities. The handling model comparison is shown for Arm Stat. in (a) and Arm Mov. in (b). The roll model comparison for Arm Stat. is in (c) and Arm Mov. is in (d).

Arm Stat. or Arm Mov. The Nonlinear Model shows that steady state a_y values are higher for Arm Mov. vs. Arm Stat., especially at higher V_x.

Next the steady state behavior of the roll models was compared. The result is displayed in Figure 3.6 (c) and (d). Again, the Linear Model and Nonlinear Model steady state ϕ values line up well at $V_x = 4\ m/s$. At $V_x = 2\ m/s$ the Linear Model predicts slightly higher steady state ϕ values than the Nonlinear Model. As forward speed increases to $V_x = 8\ m/s$, the Nonlinear Model deviates more from the Linear Model steady state ϕ values.

The transient behavior of the Linear and Nonlinear Model was compared for a forward speed of $V_x = 8\ m/s$ and steering input of $\delta = 3^o$ in Figure 3.7. Subplots (a) and (b) compare the lateral acceleration responses for Arm Stat. and Arm Mov., respectively. The transient responses line up well for Arm Stat., as one could have

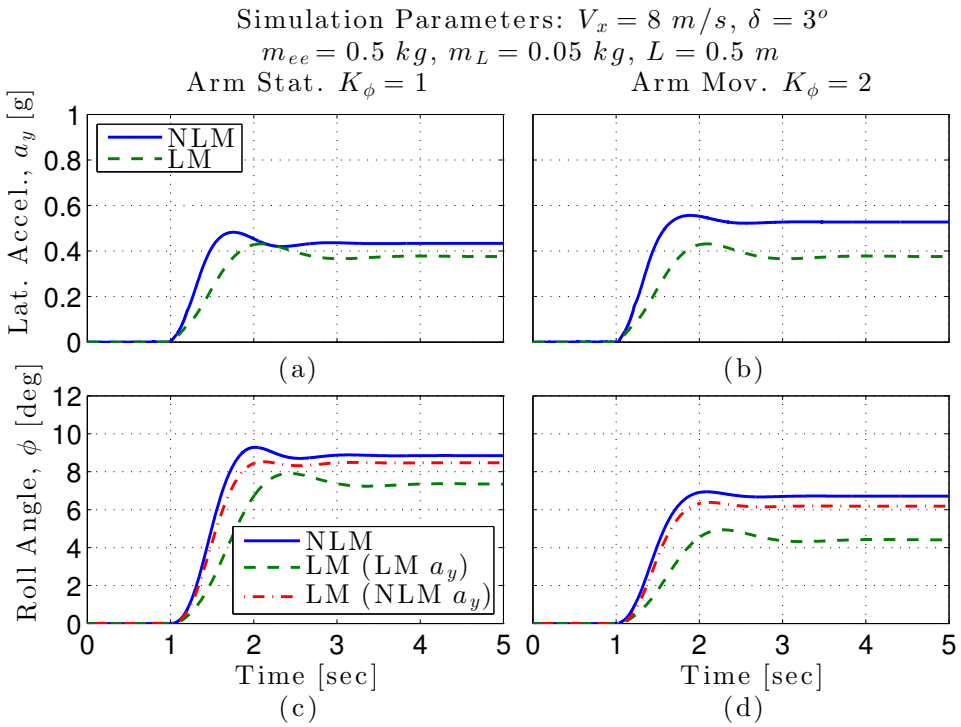

Figure 3.7: Transient responses of the Linear (LM) and Nonlinear Model (NLM) with Arm Stat. and Arm Mov. Simulations were run with forward velocity $V_x = 8\ m/s$ and step-like steering input $\delta = 3^o$. Solid blue lines represent simulation results from the NLM. Dashed green lines represents simulation results from the Linear Handling and Linear Roll Models. Dash-dot red lines represents simulation results using the NLM a_y values with the Linear Roll Model.

predicted from Figure 3.6 (a) (since the NLM '+' for $V_x = 8\ m/s$, $\delta = 3^o$ was close to the LM trendline). For Arm Mov. the Linear Model outputs a lower a_y, as one could have predicted from Figure 3.6 (b).

Subplots (c) and (d) in Figure 3.7 compare the roll angle responses for Arm Stat. and Arm Mov., respectively. The solid blue lines represent simulation results entirely using the Nonlinear Model. The dashed green lines represent simulation results using the Linear Handling and Roll Models. The dash-dot red lines input the values from the Nonlinear Model a_y transient into the Linear Roll Model. When an accurate estimate of a_y is known, the Linear Roll Model lines up closely with the Nonlinear Model, *i.e.* the solid blue and dash-dot red lines are close. However, when estimates of a_y are bad (*e.g.* dashed green line in Figure 3.7 (b)), then the Linear Roll Model

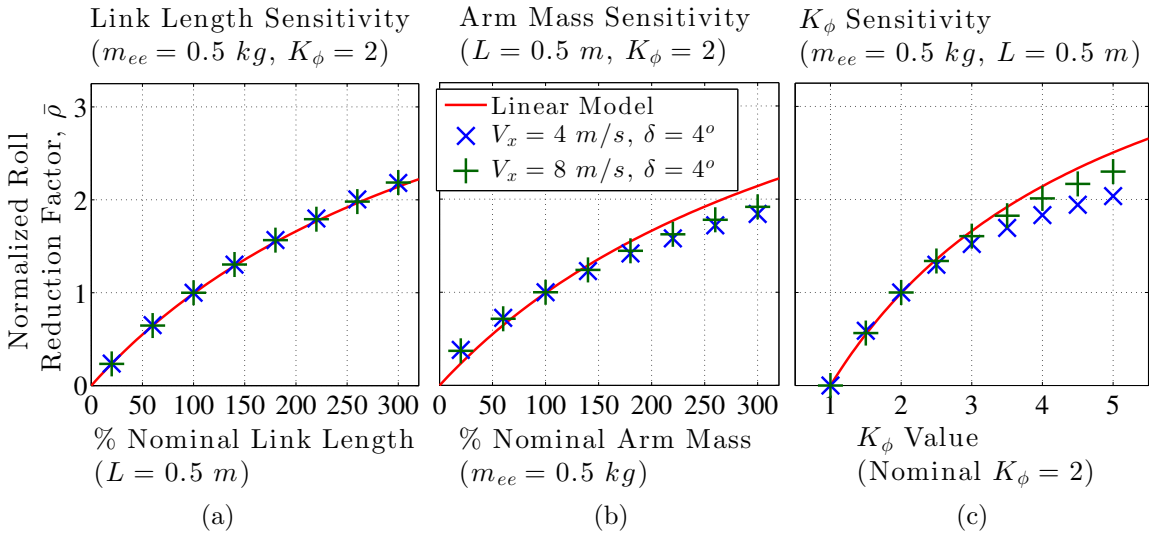

Figure 3.8: Sensitivity of normalized roll reduction factor $\bar{\rho}$ with respect to (a) link length L, (b) end effector mass m_{ee}, and (c) control gain K_ϕ for the Linear and Nonlinear Models.

will not predict transient behavior well (dashed green line does not line up with solid blue line in Figure 3.7 (d)).

While the accuracy of the full Linear (Handling and Roll) Model suffers at higher forward speeds ($V_x \geq 8\ m/s$), it does capture a similar trend to that of the Nonlinear Model. The Linear Model can serve as a starting point for selecting parameters that will improve roll stability. For instance, the Linear Model can help predict how adjusting the manipulator arm link lengths L, arm end effector mass m_{ee}, or control gain K_ϕ can improve roll stability (see Section 3.4.1). Since the full Linear Model is not as well suited for predicting transient behavior at higher forward speeds, the Linear Roll Model could be combined with a more accurate estimate of lateral acceleration. For example, the lateral acceleration may be able to be estimated from a desired vehicle trajectory. Then, the resulting lateral acceleration estimate and Linear Roll Model could be used in a model-based control method. Future work could also consider a higher DOF Linear Model that captures the dynamic interaction between roll behavior and handling.

3.4 Results

3.4.1 Manipulator Arm Parameter Sensitivity Analysis

As discussed in Section 2.1.3 many rollover stability metrics exist. Since the *critical rollover condition* was defined to occur when a wheel lifts off the ground, the brief survey of rollover stability metrics indicates that the normal load transfer metric R defined in [79] is well suited for this analysis because of its simplicity and relation to the *critical rollover condition*. This load transfer metric is shown in Eqn. (3.9), where FL refers to front left and FR to front right tire when sitting facing forward in the vehicle.

$$R = \left| \frac{F_{z,FL} - F_{z,FR}}{F_{z,FL} + F_{z,FR}} \right| \tag{3.9}$$

The load transfer metric R can be used to calculate the roll reduction factor ρ as defined in Eqn. (3.10), where $R_{mov.}$ is load transfer metric for the Arm Mov. case and $R_{stat.}$ is the load transfer metric for the Arm Stat. case.

$$\rho = \frac{R_{stat.} - R_{mov.}}{R_{stat.}} \tag{3.10}$$

In steady state with the Linear Model, the relationship between R and ϕ is found by subtracting the right from the left normal force in Figure 3.2 and dividing by the total weight of the system. Additionally, in steady state the roll rate $\dot{\phi}$ is zero. The result is Eqn. (3.11), where ϕ_{ss} is the steady state roll angle.

$$\begin{aligned} R_{lin} &= \left| \frac{{}^{W}/_{2} + k_t T \phi_{ss} - {}^{W}/_{2} + k_t T \phi_{ss}}{W} \right| \\ &= \frac{2k_t T}{W} \left| \phi_{ss} \right| \end{aligned} \tag{3.11}$$

Thus, the load transfer metric is proportional to ϕ by a constant that is independent of whether Arm Stat. or Arm Mov. in the Linear Model case. Therefore, the roll

reduction factor can be expressed as a function of only steady state roll angles for the Linear Model case as shown in Eqn. (3.12).

$$\rho_{lin} = \left| \frac{\phi_{ss,stat.} - \phi_{ss,mov.}}{\phi_{ss,stat.}} \right| \tag{3.12}$$

Since the handling and roll dynamics of the Linear Model are decoupled, the steady state gain contribution from the handling model in ρ_{lin} will cancel out. Thus, ρ_{lin} is independent of V_x and δ and can be expressed as a function of only the linear roll model parameters.

$$\rho_{lin} = 1 - \frac{k_t T^2}{k_t T^2 + L W_{ee}(K_\phi - 1)} \tag{3.13}$$

For very small L and/or W_{ee}, ρ_{lin} will approach zero. For very large L and/or W_{ee}, ρ_{lin} will approach one.

With the roll reduction factor developed above, an arm parameter sensitivity analysis was performed. Figure 3.8 (a) displays an analysis of the sensitivity of the manipulator arm link length L. Along the horizontal axis, L is varied between 20% and 300% of its nominal value of $L = 0.5\ m$. The vertical axis displays the normalized roll reduction factor $\bar{\rho}$, which is defined as the calculated roll reduction factor divided by the roll reduction factor at the nominal case of $L = 0.5\ m$.

$$\bar{\rho} = \frac{\rho}{\rho_{nom}} \tag{3.14}$$

Figure 3.8 (b) displays an analysis of the sensitivity of the manipulator arm end effector mass m_{ee}. Along the horizontal axis, m_{ee} is also varied between 20% and 300% of its nominal value of $m_{ee} = 0.5\ kg$ and the vertical axis displays the normalized roll reduction factor $\bar{\rho}$ shown in Eqn. (3.14).

Lastly, Figure 3.8 (c) displays an analysis of the sensitivity of the manipulator

arm joint angle constant K_ϕ, which is varied between $K_\phi = 1$ (Arm Stat.) to $K_\phi = 5$. The nominal value used in calculating $\bar{\rho}$ is $K_\phi = 2$.

The Nonlinear Model was run at test cases of $V_x = 4$ and 8 m/s with $\delta = 4^o$. Figure 3.8 shows that the Linear Model predicts a similar trend as the Nonlinear Model in the area around the nominal parameter value. As the parameter is increased further, it appears that the Linear Model does begin to slightly over-predict the normalized roll reduction factor of the Nonlinear Model.

From the Linear Model analysis in Eqn. (3.13) it is evident that L and m_{ee} have similar sensitivities. Both appear in the same term in the denominator and have a power of one. For this example with the nominal case of $L = 0.5\ m$ and $m_{ee} = 0.5\ kg$, the lines for the Linear Model sensitivities are very close. However, one can see that if the end effector mass were an order of magnitude larger than the nominal case and the link lengths were smaller or the same as the nominal case discussed, then m_{ee} would have a higher sensitivity than L.

This sensitivity analysis also shows that K_ϕ has a similar sensitivity to that of L and m_{ee}. This insight is valuable because improvements in roll reduction factor similar to that obtained by increasing L and m_{ee} can be achieved by only adjusting K_ϕ. Adjusting K_ϕ changes the control strategy of the manipulator arm, but it does not require changing the arm physical parameters (which may not be feasible in certain situations).

This analysis indicates that the Linear Model gives a good prediction of the expected improvement in reducing roll and load transfer. The Linear Model could be a very useful tool to robot designers in assessing how much of an improvement they could achieve in roll stability for different design scenarios. For example, the robot designer could use the sensitivity plots in Figure 3.8 to select a gain K_ϕ for their robot. In selecting K_ϕ the designer would want to consider how much torque the arm joint motors can provide. To get an estimate of the torque required, the designer could

calculate the steady state reaction moment due to the manipulator arm M_a for the extreme case when a tire lifts off and multiply by a safety factor, e.g. $\tau_m \geq \text{SF} \cdot M_a$. After using the Linear Model as a starting point for their design, it would then be necessary to test in a higher fidelity simulation similar to the Nonlinear Model used here.

3.4.2 Rollover Stability Analysis

While the Linear Model has been limited to only predicting steady state analysis, the Nonlinear Model can be used to observe the transient behavior of the system. Figure 3.9 shows the transient response for a test case of $V_x = 8\ m/s$ and step-like steering input of $\delta = 4^o$.

Notice how the radius of the turn is smaller and lateral acceleration is larger for the Arm Mov. case as shown in Figure 3.9 (a) and (b) respectively. This observation indicates that moving the manipulator arm during maneuvers does affect the handling dynamics. This trend was not captured by the Linear Model since the handling and roll dynamics were decoupled, which helps explain deviation between the Linear Model and Nonlinear Model in Figure 3.6.

The cause for this change in handling dynamics could be due to the difference in normal load distribution. For Arm Mov., R will decrease indicating a more even distribution of load between the left and right side of the vehicle. Under this condition, the tires are able to generate lateral force more effectively and the vehicle performs a tighter radius turn for a given steering input.

Additionally, the roll angle of the tire influences the tire behavior [80]. As the tire experiences larger roll angles, its ability to generate lateral force decreases. This idea agrees with the trend shown in Figure 3.6, that since the mobile manipulator with Arm Mov. experiences a smaller roll angle ϕ, it is able to more effectively generate lateral acceleration a_y and perform a tighter radius turn with the same forward speed

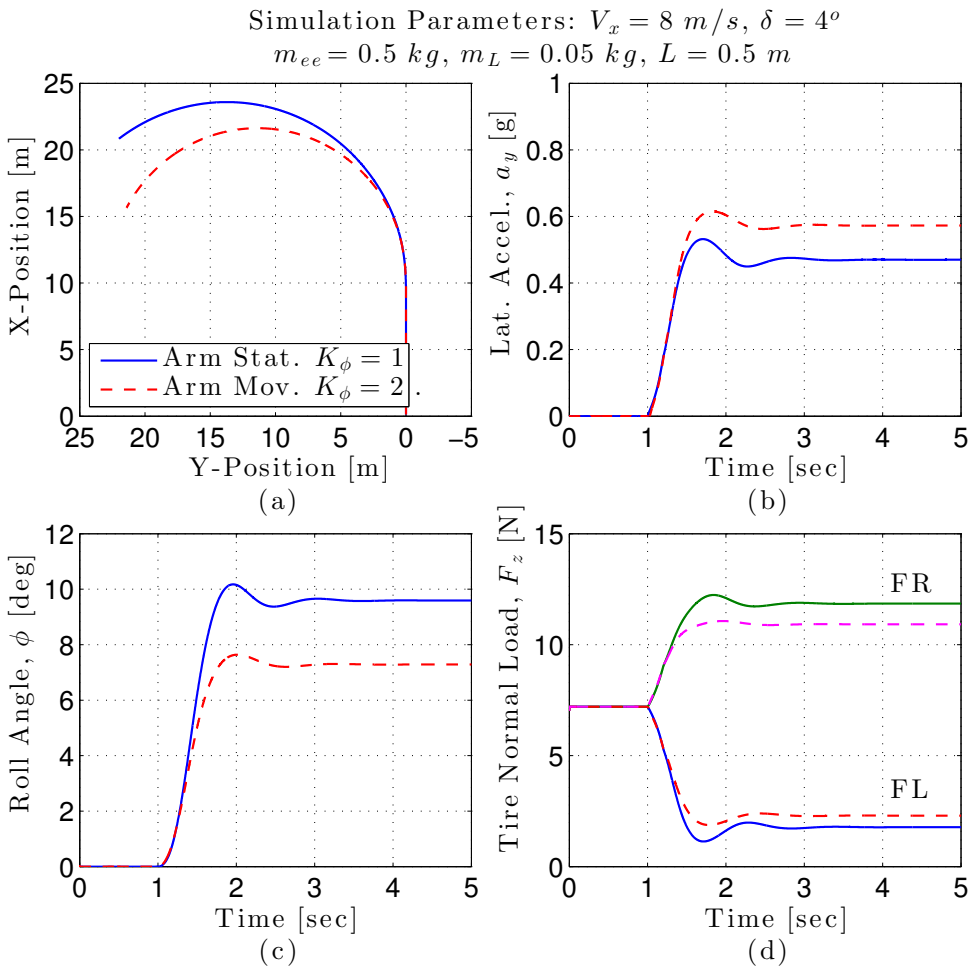

Figure 3.9: Transient responses of the Nonlinear Model with constant forward velocity $V_x = 8\ m/s$ and step-like steering input $\delta = 4^o$. Plot (a) shows the X-Y path traveled by the vehicle , (b) shows the resulting lateral acceleration transient, (c) shows the roll angle response, and (d) shows the tire normal load transients during the maneuver.

and steering input as the mobile manipulator with Arm Stat.

Further exploration of this effect of Arm Mov. on handling dynamics will be discussed in Section 3.4.3. Here, a set of steering inputs that result in turns with the same radius as shown in Figure 3.10 is considered. From Figure 3.10 (a) and (b) it can be seen that the $X - Y$ paths for Arm Mov. and Arm Stat. mobile manipulators are very close, as are the lateral accelerations. Both mobile manipulators had forward speeds of $V_x = 8\ m/s$, however the mobile manipulator with Arm Mov. only needed a steering input of $\delta = 3^o$, while a steering input of $\delta = 6^o$ was needed with Arm Stat.

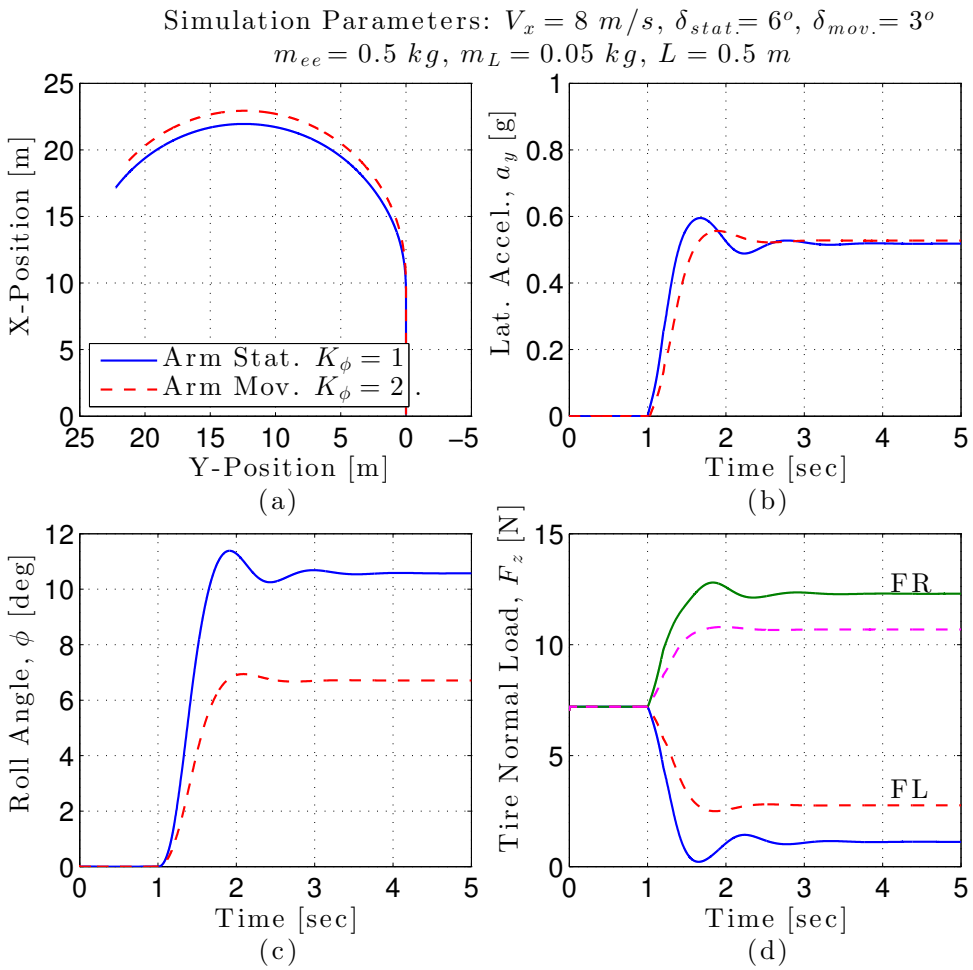

Figure 3.10: Transient responses of the Nonlinear Model with constant forward velocity $V_x = 8\ m/s$ and step-like steering input $\delta = 6^o$ for Arm Stat. and $\delta = 3^o$ for Arm Mov. Plot (a) is the X-Y path traveled by the vehicle , (b) is the lateral acceleration transient, (c) is the roll angle response, and (d) is the tire normal load transients during the maneuver.

From Figure 3.9 and 3.10, one can see the rollover stability improvement from Arm Mov. will appear to be larger when analyzed in terms of forward speed V_x and path radius, rather than in terms of forward speed V_x and steering angle δ. In the analysis for Figure 3.9 where both systems are compared in terms of the same forward speed ($V_x = 8\ m/s$) and steering input ($\delta = 4^o$), the roll reduction factor is $\rho_\delta = 0.12$. In the analysis for Figure 3.10 where both systems are compared in terms of the same forward speed ($V_x = 8\ m/s$) and $X - Y$ path radius ($\delta_{moving} = 3^o$, $\delta_{stat.} = 6^o$),

the roll reduction factor is $\rho_{radius} = 0.29$. A larger roll reduction is found when comparing the Arm Mov. versus Arm Stat. cases for mobile manipulators traveling similar trajectories instead of receiving the same inputs.

3.4.3 Handling Dynamics Analysis

In order to gain some insight into how weight-shifting affects the overall stability of the system, a batch of simulations was run with the Nonlinear Model. A range of forward velocities of $V_x = 1$ to 15 m/s was run with steering inputs of $\delta = 1$ to 15^o for both cases of Arm Stat. and Arm Mov.

Tire lift-off is defined to be when any one of the tire normal forces reaches zero. Tire saturation is defined to be when the magnitude of the lateral and longitudinal forces generated by the tire reach the friction limit. During simulations of the Nonlinear Model the following three conditions occurred:

1. NTLO,NTS: No tire lift-off and no tire saturation
2. NTLO,TS: No tire lift-off and tire saturation
3. TLO,NTS: Tire lift-off and no tire saturation

Data from the simulation runs was checked to see which one of three conditions it exhibited during the transient response. If tire lift-off or tire saturation was detected at any point during the transient, then that combination of forward velocity and steering input was labeled appropriately. Note: once the tires began to saturate, the vehicle was not able generate enough lateral force to cause a tire-lift off event for the turning maneuver tested. Thus, a tire lift-off and tire saturation (TLO,TS) condition did not occur during any of the simulation runs.

Figure 3.11 shows the results of this batch of simulations for Arm Stat. (left) and Arm Mov. (right). These graphs show that by moving the manipulator arm as previously described, the stability region of allowable forward speeds and steering

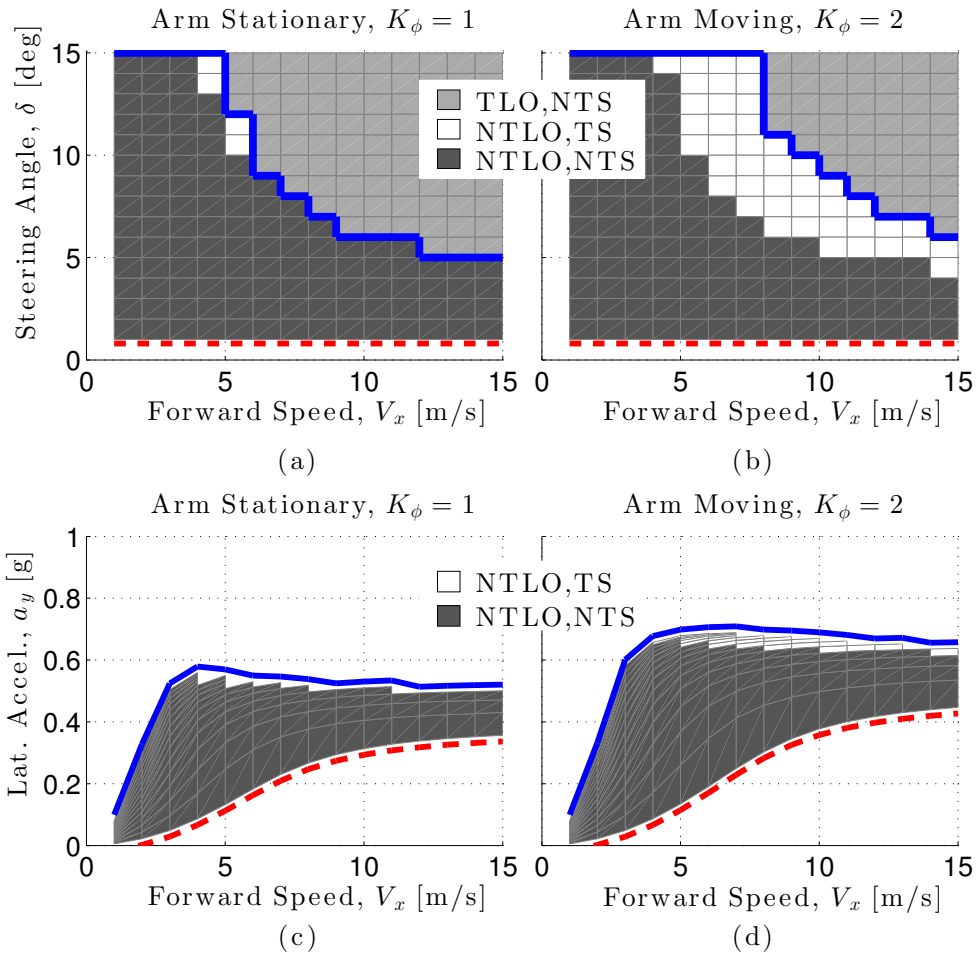

Figure 3.11: Graph of the rollover stability region over a range of forward speeds and steering inputs for Arm Stat. (a) and Arm Mov. (b) cases. The rollover stability regions in plots (a) and (b) for the NTLO conditions are shown in terms of lateral accelerations for a given forward speed in plots (c) and (d).

inputs increases. Note that in the Arm Mov. case the TLO region becomes smaller, however the NTLO,TS region becomes larger. When the tires saturate the operator is limited in their ability to generate additional lateral force and control the vehicle. While this condition is also undesirable, it can be easier to recover from than the TLO scenario.

As Figure 3.10 shows, a smaller steering input is required to achieve the same radius turn/lateral acceleration for the Arm Mov. vs. Arm Stat. case. Thus, viewing the rollover stability region in terms of lateral acceleration vs. forward speed

is also informative to understand how weight-shifting can improve maneuverability. Figure 3.11 (c) and (d) depict the same rollover stability region in Figure 3.11 (a) and (b) in terms of lateral acceleration vs. forward speed.

The dashed red and solid blue lines in Figure 3.11 are present to aid in visualizing how the rollover stability regions in plots (a) and (b) map to the regions in plots (c) and (d), respectively. That is, the simulations used to define the rollover stability regions above the dashed red line and below the solid blue line in Figure 3.11 (a) and (b) produced the lateral accelerations above the dashed red and below the solid blue line in Figure 3.11 (c) and (d). Note that for low speeds (less than 4 m/s) steering angles of 15^o did not result in tire lift-off or tire saturation. Thus, the vehicle can likely still achieve higher lateral accelerations at these low forward speeds by using larger steering angles. Additionally, the smaller lateral accelerations below the dashed red lines at high forward speeds in Figure 3.11 (c) and (d) are presumably achievable with steering angles smaller than 1^o. Overall, it is evident from Figure 3.11 that the region of achievable lateral accelerations that result in NTLO is larger for the Arm Mov. case ((a),(c)) in comparison to the Arm Stat. case ((b),(d)).

The impact of weight-shifting on the handling dynamics can also be seen in Figure 3.12. For each of the forward speed - steering angle input combinations that resulted in NTLO in Figure 3.11, the percent increase in lateral acceleration for the Arm Mov. vs. Arm Stat. case is shown in Figure 3.12. At low speeds and low steering angles, weight-shifting has a smaller effect on handling. That is, for a constant steering input at constant speed, the resulting lateral acceleration will only be slightly higher for the Arm Mov. case. However, at higher speeds and steering angles, the steering will become more sensitive for the Arm Mov. case. Overall, Figure 3.11 illustrates that weight-shifting increases the maximum lateral acceleration (allowing for smaller radius turns at high speeds), while Figure 3.12 shows that weight-shifting also causes the sensitivity of the steering angle to increase.

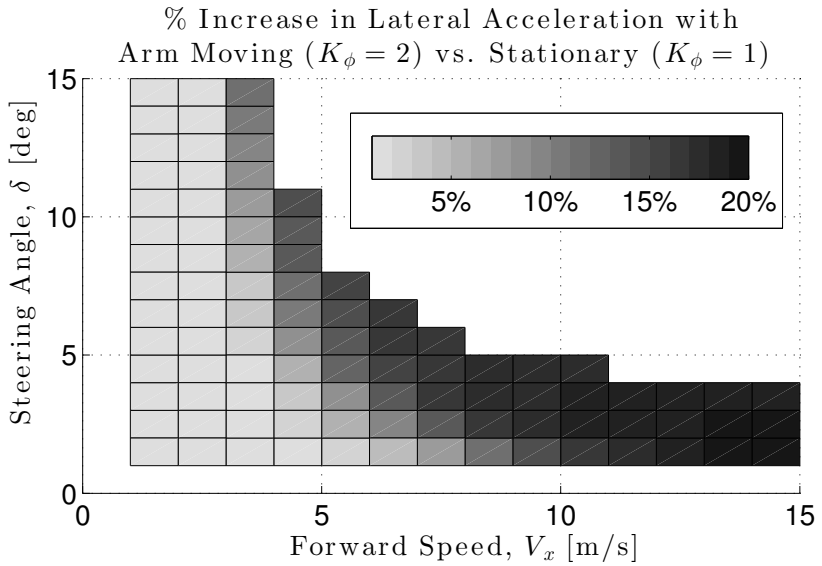

Figure 3.12: Graph showing the effect of weight-shifting on handling dynamics. The percent increase in lateral acceleration is shown at each forward speed - steering input combination that resulted in NTLO conditions for both the Arm Stat. and Arm Mov. cases.

3.4.4 Experimental Validation

To support the analysis with the Linear Model and Nonlinear Model, the dynamic weight-shifting control method was implemented on the hardware shown in Figure 3.4.

3.4.4.1 Experimental Setup

The vehicle platform used was a 1:10 scale RC car chassis from Team Associated. A custom two-link manipulator arm was fabricated primarily by a senior design team at the University of Michigan [49]. The manipulator arm joints were actuated by a Dynamixel MX-64 servo (base-Link 1 Joint) and a Dynamixel MX-28 servo (Link 1-Link 2 Joint). The Dynamixel servos have built in PID controllers allowing for easy position control.

On-board the experimental platform was a Raspberry Pi Model B microcomputer. The microcomputer collected sensor data from an inertial measurement unit (IMU) and commanded the manipulator arm joints to the desired angular positions. A six

DOF IMU was used, consisting of a three-axis accelerometer (ADXL345) and three-axis rate gyro (ITG3200), to estimate vehicle states. Data was collected from the IMU at 100 Hz. Roll angle was estimated in real time on the microcomputer using a moving average filter consisting of 10 points. Based on the estimated roll angle, joint angle position commands were sent to the arm at 10 Hz.

The dimensions and weight of the experimental platform were very close to those used for the Linear Model and Nonlinear Model. One difference was that the manipulator links on the experimental platform were slightly shorter, Link 1 was approximately 0.3 m and Link 2 was 0.2 m.

The experimental platform was operated outdoors in a flat, open parking lot. A camera was placed at a high vantage point overlooking the parking lot and captured the movement of the vehicle at 30 Hz. The camera was calibrated and the homography transformation matrix relating the pixel data to world frame data was calculated. A red marker was placed on the back of the vehicle so that it could be easily tracked using simple color threshold detection in the video recorded for each maneuver. This visual tracking allowed us to estimate the $x - y$ position and speed of the vehicle during each maneuver.

3.4.4.2 Experimental Results

Two comparisons will be made between the experimental results and analysis with the models. The first supports the effect of the arm on preventing rollover and the second supports analysis of the arm's effect on handling dynamics.

The same type of maneuver performed with the Nonlinear Model in Figure 3.9 and 3.10 was performed with the experimental platform. Given the size of the parking lot used for testing and the capabilities of the hardware, a step-like steering input maneuver at a speed of just over 5 m/s was performed. The magnitude of the steering input was 20^o, which was larger than the magnitude of the steering inputs applied

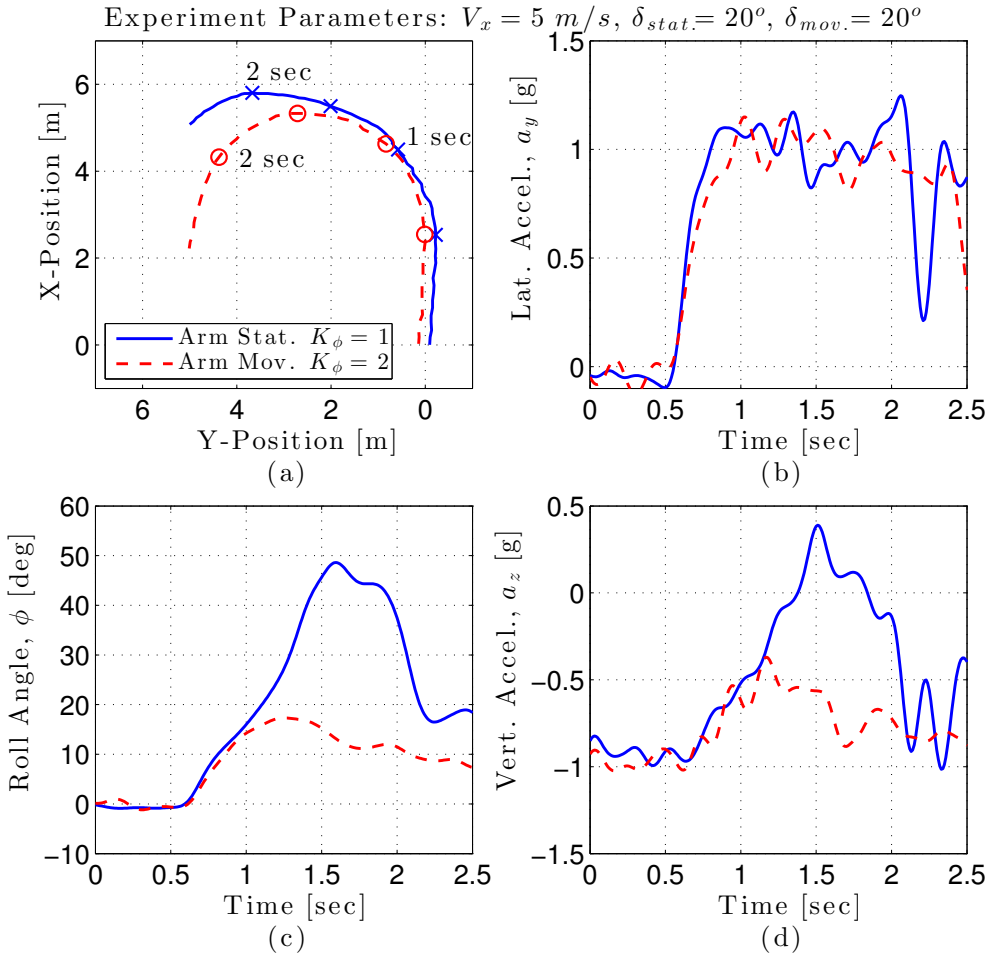

Figure 3.13: Transient responses of the experimental platform with forward velocity $V_x = 5\ m/s$ and step-like steering input $\delta = 20^o$. Plot (a) shows the X-Y path traveled by the vehicle with markers at 0.5, 1, 1.5 and 2 seconds, (b) shows the resulting lateral acceleration transients, (c) shows the roll angle response, and (d) shows the vertical acceleration transients during the maneuver.

to the Nonlinear Model. A larger magnitude steering input was selected to keep the radius of the maneuver small while still experiencing large lateral accelerations that occur in a rollover event. The manipulator arm joint angle constant was set to $K_\phi = 2$ for the Arm Mov. case.

The results of the turning maneuver with the experimental platform are shown in Figure 3.13. The $x - y$ position data collected using computer vision is shown in subplot (a). Markers are included at each 0.5 second interval to compare with the

timeseries data in subplots (b)-(d). All IMU signals were filtered using a fourth order Butterworth filter with cutoff frequency 5 Hz and Matlab's zero-phase distortion filtering function `filtfilt`. The roll angle in subplot (c) was estimated using x-IO Technologies' implementation[1] of Mahony's AHRS algorithm [71]. During the maneuver recorded in Figure 3.13 the vehicle's wheels actually lifted off the ground and the vehicle appeared to roll just past the point at which the outriggers first hit the ground for the Arm Stat. case. The outriggers (see Figure 3.4) touch the ground when the vehicle is rolled approximately 42^o in the static case. Thus, the AHRS algorithm's gain was tuned such that the maximum roll angle of the Arm Stat. case was just under 50^o. The lateral and vertical accelerations were then transformed from the vehicle frame to the world frame via multiplication by a rotation matrix about the vehicle's roll axis.

From Figure 3.13 (c), one can see that moving the manipulator arm with weight shifting had a dramatic effect on reducing the vehicle's roll angle. In this case, when the vehicle performed the maneuver with the Arm Stat., the tires lifted off and the outriggers hit the ground. This event occurred around 1.5 seconds. One can see in subplot (c) that 1.5 s is where the roll angle peaks and in subplot (d) the vertical acceleration goes positive (in the static case gravity is -1 g in the z-direction).

The second comparison made with the experimental results supports the handling dynamics analysis. The vehicle was given a constant steering input of 15^o and a constant throttle input of 45% max throttle, until it reached steady state driving in a circle. The resulting maneuver with the experimental platform is shown in Figure 3.14.

The vehicle's forward speed was estimated using video data. In particular, the radius of the circle driven by the vehicle was estimated and the time the vehicle took to complete a lap was calculated. In steady state, the forward speed of the vehicle

[1] http://www.x-io.co.uk/open-source-ahrs-with-x-imu/

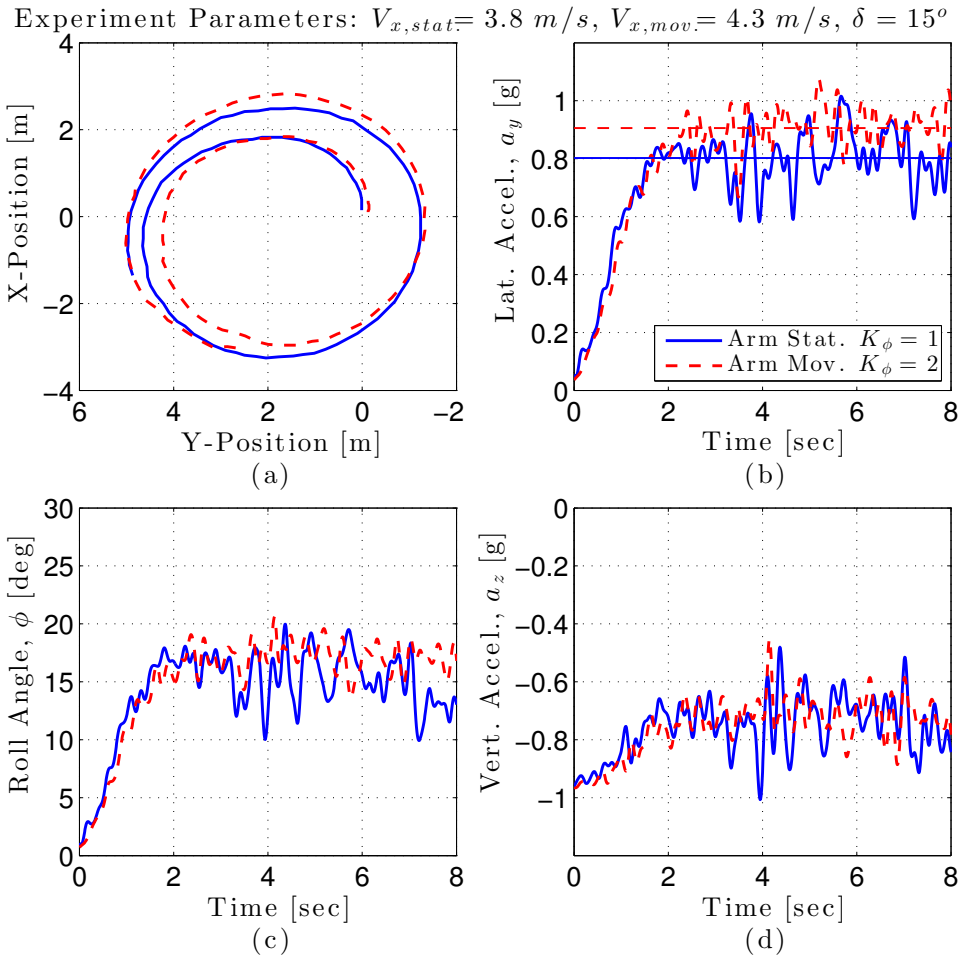

Figure 3.14: Transient responses of the experimental platform with constant throttle and steering input $\delta = 15^o$. Plot (a) shows the X-Y path traveled by the vehicle, (b) shows the resulting lateral acceleration transients steady-state values indicated, (c) shows the roll angle response, and (d) shows the vertical acceleration transients during the maneuver.

for the Arm Stat. case was approximately 3.8 m/s and the resulting average lateral acceleration for time 2 to 8 s was 0.802 g. The forward speed for the Arm Mov. case was 4.3 m/s with an average lateral acceleration of 0.906 g. Comparing these two lateral accelerations gives a 12.9% increase in lateral acceleration with the Arm Mov. vs. Arm Stat. case. This result is similar to the value of 17.6% found with the Nonlinear Model at forward speed 4 m/s and steering angle 15^o (Figure 3.12). Overall, the experimental results appear to line up well with results from the nonlinear

simulation. Both effects of reducing roll angle and increasing lateral acceleration gain were reproduced on the experimental platform.

3.5 Conclusions

This chapter presented a control method that uses an existing manipulator arm to improve rollover stability and increase maneuverability in high speed maneuvers. The Linear Model was shown to be a useful tool for analyzing the sensitivity of design parameters including link length L, mass m_{ee}, and joint angle constant K_ϕ. The Nonlinear Model was used to describe the impact of dynamic weight-shifting on both roll dynamics and maneuverability. Experimental results from a hardware implementation of dynamic weight-shifting supported both relationships for roll dynamics and maneuverability developed with the Nonlinear Model. Unlike other dynamic stability control methods that require adding additional hardware, decreasing vehicle speed or increasing turn radius, the results in this dissertation demonstrated that by dynamically shifting the weight of the manipulator arm, turns can be taken at a higher speed and smaller radius compared to keeping the manipulator arm static.

CHAPTER IV

Teleoperation with Time-Varying Communication Delay

4.1 Introduction

Communication delay has been studied and shown to have a negative impact on robot teleoperation performance as evidenced by the numerous studies discussed in Section 2.2.1. Wireless communication often introduces time-varying delay. That is, the delay of each information packet exchanged between the operator and robot is not constant. The delay applied to each packet is often described by a stochastic distribution. Previous studies have shown that for delay distributions with the same first statistical moment (mean), a larger second statistical moment (variance) results in worse tracking performance or longer task completion time [26]. However, they have not suggested a way of quantitatively relating teleoperation performance for delays having different statistical moments. This dissertation proposes a method of relating teleoperation performance among time-varying delays having stochastic distributions with different statistical moments.

The user study results and analysis presented in this chapter are based on publication [100]. The remainder of this chapter is organized as follows. Section 4.2 discusses the user study design for teleoperated driving under time-varying delays. Section 4.3

summarizes the key results of the user study, including comparison of human subject data with driver model data. Section 4.4 provides insights and discussion on the results. Lastly, Section 4.5 gives a synopsis of the findings.

4.2 Methods

4.2.1 Teleoperation System

In this study, we simulate a teleoperated mobile robot system designed to follow a path displayed on the road. Figure 4.1 summarizes the teleoperation system components.

4.2.1.1 Main Controller

Two types of main controllers are explored in this study. Both main controllers give outputs of desired robot turn rate. The first type of main controller is an actual human operator. Human operators gave control inputs to the system via a Logitech F710 gamepad (a typical type of input device for teleoperated robots like the iRobot PackBot [50]) based on visualization feedback displayed to them. A uniform distribution of noise between -0.1 and 0.1 times the maximum possible input was added to the main controller output signal. This noise was added to simulate noise present in a physical robot system.

The second type of main controller used in this study is the steering model developed by Vozar [109]. The steering model is a PD controller on the projected lateral displacement of the robot from its desired path. The projected lateral displacement is calculated by finding the distance between the robot's projected state and the closest point on the desired path. The projected state is calculated assuming the robot keeps moving in its current heading direction at a constant speed during the lookahead time T_p. Two other parameters in the steering model are the physical actuation time

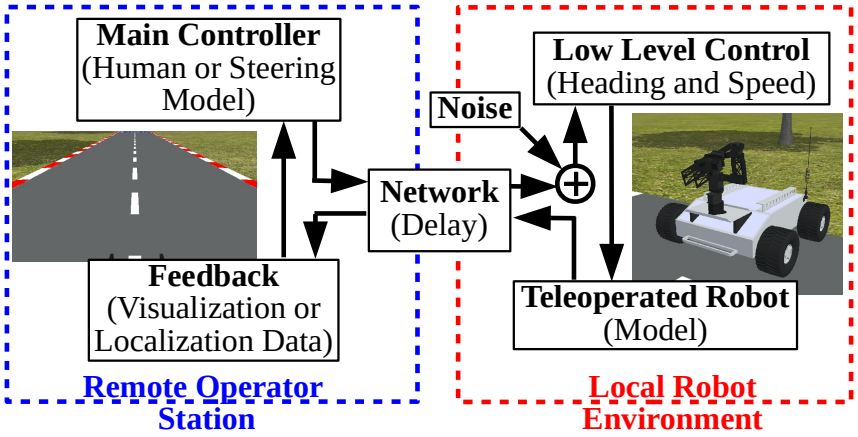

Figure 4.1: Overview of the teleoperation system's main components.

δ_h and input control threshold μ. The δ_h parameter delays the controller's input to represent the time it takes for a human driver to physically give the system an input. The μ parameter conditions the output of the controller to mimic the toggling behavior observed in users. If the magnitude of the input $u(t)$ from the steering model is less than μ, no input is given. If $|u(t)| > \mu$, then $\mu = \text{signum}(u(t))$.

4.2.1.2 Feedback

This study only considers one form of feedback to the main controller: a visualization for the human operator and localization data for the steering model. Humans receive a view from the camera mounted on the manipulator arm of the simulated robot they are controlling (shown on the left side of Figure 4.1). This visualization represents a first-person point-of-view.

Localization data fed-back to the steering model consists of robot state measurements. In this study, the steering model is assumed to know the desired path of the robot, thus, it is not necessary to feedback additional environment data.

4.2.1.3 Network Model

We simulated the network by delaying signals passed between the remote operator station and local robot environment. That is, commands passed from the main controller to low level controllers on the robot were delayed. Constant and time-varying delays were considered.

Time-varying delays were generated in one of two ways - either from a normal distribution or from the folded normal distribution with an offset (FNDO). The normal distribution will be referred to with notation $N(\mu, \sigma)$, where μ is the mean value in milliseconds (ms) and σ is the standard deviation in ms. The FNDO will be referred to with notation $\delta(\delta_{min}, \sigma)$, where δ_{min} is the delay minimum in ms and σ is the standard deviation parameter in ms. It is defined as:

$$\delta(\delta_{min}, \sigma) = \delta_{min} + |N(0, \sigma)| \tag{4.1}$$

The FNDO was used because it resembles the shape of IEEE 802.11-style wireless network packet intervals measured in [3] and it allows for comparison to prior work in [109]. Figure 4.2 shows a side-by-side comparison of the two distributions. Using the definitions of expected value and variance [29], the expected value and variance of a random variable following the FNDO $X_F \sim \delta(\delta_{min}, \sigma)$ are:

$$E(X_F) = \delta_{min} + \sigma\sqrt{\frac{2}{\pi}} \tag{4.2}$$

$$V(X_F) = \sigma^2(1 - \frac{2}{\pi}) \tag{4.3}$$

The command delays were applied in the following manner. At each timestep of the simulation, timestamped commands from the main controller were queued. A time delay of δ ms was generated according to the specified distribution. The newest main controller command that was at least δ ms old was then passed to the low level

controllers and older commands were discarded. If none of the queued commands were older than δ ms, then the previous input was passed until the oldest command was older than δ ms. This process repeated every timestep - approximately every 25 ms.

4.2.1.4 Teleoperated Robot

The teleoperated robot was a small differential drive robot (approximately 0.75m wide) with a manipulator arm modeled in the Autonomous Navigation Virtual Environment Laboratory (ANVEL) [31]. A screenshot of the robot in ANVEL is shown on the right side of Figure 4.1. The robot model considers the dynamics of the robot chassis and its wheels, as well as dynamics of the drive motors. The robot had a maximum turn rate of 1 rad/s to give it a turning radius of 1.5 m at top speed.

4.2.1.5 Low Level Control

The low level control is simulated to be onboard the teleoperated robot and receives inputs from the main controller (through the network) of desired forward speed and turn rate. The low level control uses PID controllers in ANVEL to calculate the necessary inputs to the drive motors on each side of the differential drive robot. These low level PID controllers were hand-tuned to achieve good settling times (<500 ms) and small overshoot (<10%).

4.2.2 Experiment Design

4.2.2.1 Task Description

The task for the main controller was to follow a dashed white line in the center of the track as closely as possible. Each trial of the path following task was completed on one of 18 tracks. Each track had a 10 m unscored warm-up zone followed by one of each of the following turn elements (separated by a 5 m straight section): left turn,

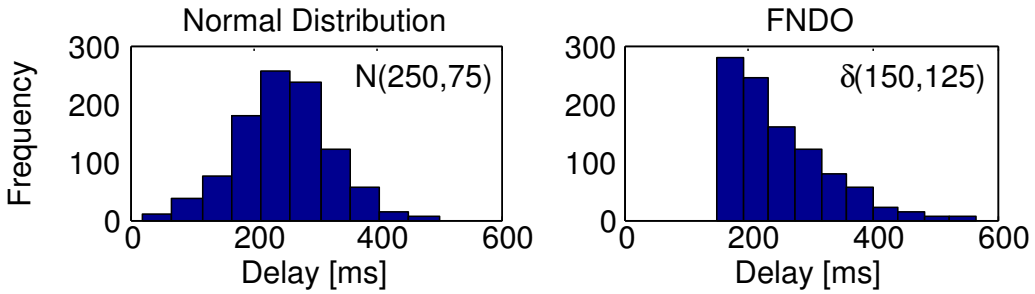

Figure 4.2: Histograms of 1000 randomly generated delays according to the normal distribution (left) and FNDO (right). Both distributions have an expected value of 250 ms and standard deviation of 75 ms.

right turn, left U-turn, right U-turn, left-right S-turn, and right-left S-turn. The 18 tracks differed in the order that the turn elements appeared. The radius of curvature for the center of the track in each turn was 2 m and the track width was 2 m. The turn radius is the same used in the ASTM E2829-11 test procedure for evaluating robot maneuverability at sustained speeds [32].

During each trial, the robot moved forward at a constant speed and the main controller's only input was desired turning rate. If the robot managed to drive off the 2 m wide track, the robot's speed was decreased by 50% to make it easier to get back on the track.

4.2.2.2 Independent Variables

The independent variables manipulated in this study were the (a) network delay and (b) robot forward speed. Ten network delay distributions were tested:

- 4 constant delays - 0, 380, 660, 750 ms
- 2 FNDO time-varying delays - $\delta(150, 125)$, $\delta(300, 250)$ with expected values 250, 500 ms, respectively, and standard deviations 75, 150 ms, respectively
- 4 normal distribution time-varying delays - $N(175, 118)$, $N(250, 75)$, $N(400, 244)$, $N(500, 150)$

These delays were selected to be within the range of prior work [109]. Specific justification for selecting these delays is discussed in Section 4.2.2.5. Robot forward speeds of 1 and 1.5 m/s were tested to compare results with prior work in [109] and because they are within the range of speeds used by ASTM E2829-11 [32].

4.2.2.3 Dependent Variables

Users' inputs and the robot's state were recorded at each timestep of the simulation. The dependent variable discussed in this work is path following score. It was calculated by averaging the normalized scores of the robot at each timestep. Once the robot crossed the start line, normalized scores at each timestep were calculated as:

$$S_i = \max\left(0, \frac{w/2 - |y_i|}{w/2}\right) \tag{4.4}$$

where w is the track width (in this case 2 m) and y_i is the lateral displacement of the robot from the center of the track at step i. The total score for each trial is determined by averaging the scores at each timestep. Therefore, a path following score of 1 indicates the robot followed the center line on the track perfectly and a score of 0 indicates the robot was off the track for the entire path. This metric was chosen because of its simplicity to explain to test subjects unfamiliar with root mean square error and the saturation effects prevent large score variations from a single mistake.

4.2.2.4 Experiment

User tests were conducted with 36 volunteers using the setup in Figure 4.3. One user withdrew part way through the study, leaving data for 35 users. There were 26 male and 9 female test subjects with an average age of 23.7 years and standard deviation (sd) 2.3 years. On a scale of 1 (low) to 7 (high), subjects reported an average video game experience of 4.7 (sd=1.7) and an average familiarity with robotics of

Figure 4.3: Photo of a volunteer with the experimental test setup.

3.4 (sd=1.4). Subjects were paid $10 for participating in the study. To incentivize subjects to do their best, $10 bonuses (with a $30 bonus cap per subject) were offered to the top performer for 10 of the scenarios that all subjects performed. Each user test took approximately 50 minutes. These tests were approved by the University of Michigan Health Sciences and Behavioral Sciences Institutional Review Board (UM IRB #HUM00044265).

This study used a repeated-measures test design with the two independent variables of delay type and robot speed. Each user experienced 9 of the 10 different delay types. Users performed two consecutive trials of the same delay type, but each was at different robot speeds of 1 or 1.5 m/s (with the order randomized). The breakdown for the number of users that tested at each delay type is shown in Table 4.1. Results are omitted for two trials that did not log properly (one for 380 ms and one for $\delta(150, 125)$).

An overview of the test procedure is as follows. First, subjects were consented to participate in the study and filled out a demographic form. Next, the experimenter explained the details of the path following task and method of scoring. Users began by completing two trials, each at different speeds, with their first delay type. Following that, users filled out a brief survey about the delay type they experienced. After the

Table 4.1: Number of users participating in each delay type. Delay values are given in ms.

Constant Delay				**FNDO**		**Normal Distribution**			
0	380	660	750	$\delta(150, 125)$	$\delta(300, 250)$	$N(175, 118)$	$N(250, 75)$	$N(400, 244)$	$N(500, 150)$
35	34	28	28	34	35	35	28	28	28

survey, users repeated the process.

4.2.2.5 Hypotheses

Prior work in [109] found that user path following performance of $\delta(150, 125)$ and $\delta(300, 250)$ lined up with the trendline of path following performance at constant delays of 380 and 660 ms, respectively, for both robot speeds. Therefore, we expected user path following scores at these time-varying and constant delay pairs would be similar. This idea motivated testing constant delay values 380 and 660 ms.

We conjectured that distribution mean and standard deviation would impact path following performance more than distribution shape. Therefore, time-varying delay distributions $N(250, 75)$ and $N(500, 150)$ were tested, which have the same mean and standard deviation as $\delta(150, 125)$ and $\delta(300, 250)$, respectively, to see if there would be a difference in path following scores.

Additionally, we predicted that a group of time-varying delay distributions could all result in similar path following performance. We hypothesized that this group of time-varying delay distributions could be described by equating a linear combination of the distribution's expected value and standard deviation to an equivalent constant delay, *i.e.*

$$\alpha \cdot E[X] + \beta \cdot SD[X] = \delta_{equiv} \tag{4.5}$$

We predicted the constant 380 ms and time-varying $N(250, 75)$ delays would both

result in path following performance similar to an equivalent delay of 380 ms. Substituting this relation into Eqn. (4.5) gives $\alpha_1 = 1$, $\beta_1 = 1.73$. Similarly, we predicted 660 ms and $N(500, 150)$ would have equivalent path following scores. Substituting these values into Eqn. (4.5) gives $\alpha_2 = 1$, $\beta_2 = 1.07$. Based on α_1, β_1 we selected $N(175, 118)$ anticipating it would result in similar performance to $N(250, 75)$. Similarly, with α_2, β_2 we selected $N(400, 244)$ to see whether path following performance would be similar to $N(500, 150)$.

In summary, the following two hypotheses will be tested with an ANOVA test at each robot speed (1 and 1.5 m/s):

(a) There is no difference in human driver path following performance for delays of 380 ms, $\delta(150, 125)$, $N(250, 75)$, and $N(175, 118)$.

(b) There is no difference in human driver path following performance for delays of 660 ms, $\delta(300, 250)$, $N(500, 150)$, and $N(400, 244)$.

The following will be discussed qualitatively due to ill-condition of the measurements for traditional statistical tests:

(c) The trend in path following performance versus delay with the kinematic robot plant model holds for a dynamic robot plant model.

(d) The steering model developed in [109] can be used to predict user performance in our study.

4.3 Results

4.3.1 Delay Distributions with Similar Performance

4.3.1.1 Distributions Similar to 380 ms - Hypothesis (a)

Human driver data for the four delay distributions in hypothesis (a) are compared side-by-side with boxplots in Figure 4.4 for robot speeds 1 m/s (top) and 1.5 m/s (bot-

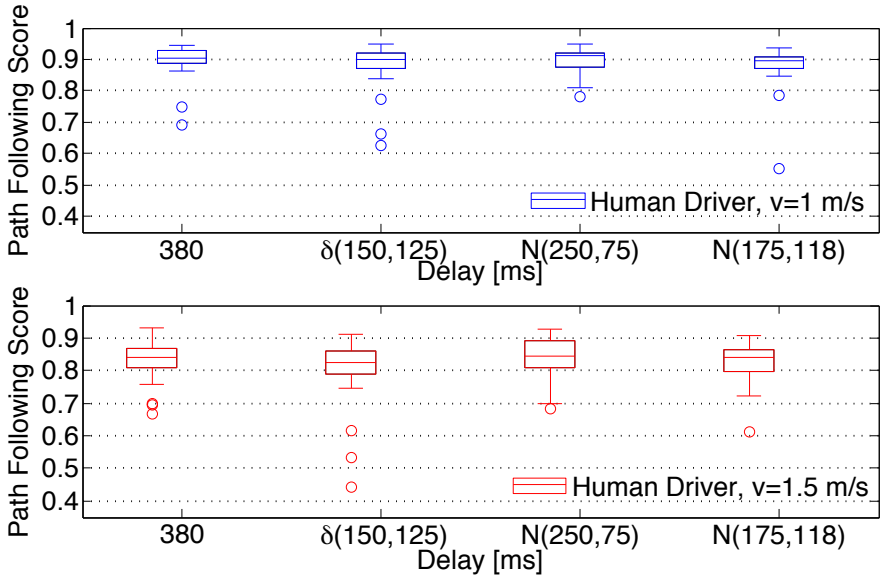

Figure 4.4: Comparison of path following performances for delay distributions we hypothesized would be equivalent to 380 ms.

tom). For all boxplots, the whiskers extend to the most extreme data points within 1.5 IQR (interquartile range) of the 25th and 75th percentiles. Data points outside of this range are marked with 'o' markers. Qualitatively, the path following scores for these four distributions appear to align very well. A single factor ANOVA test was performed on the path following scores for these four delays. Due to the nature of null hypothesis tests, we cannot conclude that the performance is the same among these four distributions. However, we can conclude that no significant difference was found among these four delay distributions for both the 1 m/s robot speed (p-value=0.47, F(3,134)=0.85) and the 1.5 m/s robot speed (p-value=0.22, F(3,134)=1.48).

4.3.1.2 Distributions Similar to 660 ms - Hypothesis (b)

Human driver data for the four delay distributions in hypothesis (b) are compared side-by-side in Figure 4.5 for robot speeds 1 m/s (top) and 1.5 m/s (bottom). A single factor ANOVA test was performed on the path following scores for these four delays. No significant difference was found among these four delay distributions for

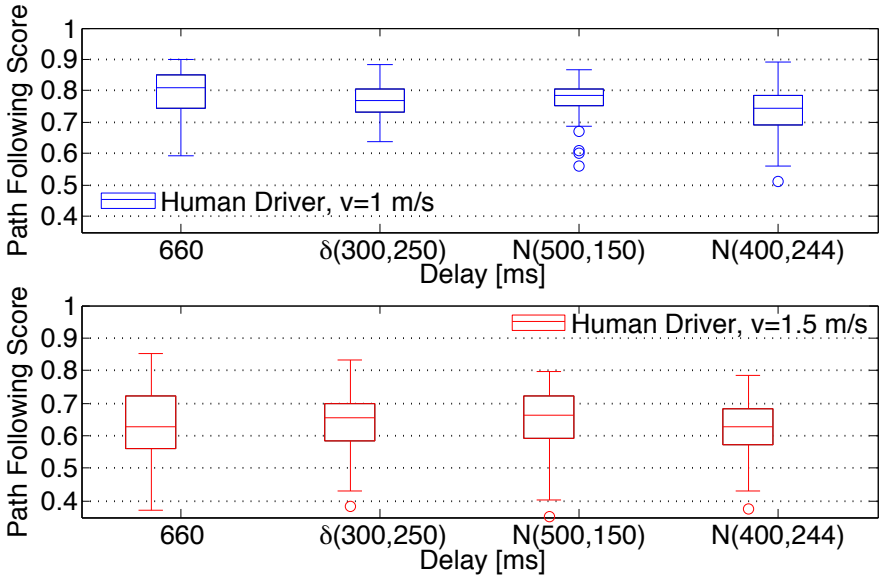

Figure 4.5: Comparison of path following performances for delay distributions hypothesized to be equivalent to 660 ms.

the 1.5 m/s robot speed (p-value=0.73, F(3,108)=0.43). However, a significant difference was found at the 1 m/s robot speed (p-value=0.015, F(3,108)=3.66). Tukey's HSD test [29] was applied to see which were significantly different. The results showed that path following scores for $N(400, 244)$ were significantly different than those at 660 ms for the 1 m/s speed (significance level $\alpha = 0.05$). Looking at the path following scores in Figure 4.5 (top) it does appear that $N(400, 244)$ at robot speed 1 m/s is significantly lower than the other distributions. Further discussion is provided in Section 4.4.

4.3.2 Comparison of Kinematic and Dynamic Robot Models

4.3.2.1 User Data - Hypothesis (c)

Figure 4.6 displays boxplots of path following scores for the user data in this study. Each boxplot is placed along the horizontal axis at its corresponding constant delay. The solid trendlines are polynomials fit to the median user at the constant delays tested. The dashed trendlines represent the polynomial fit to the median user at each

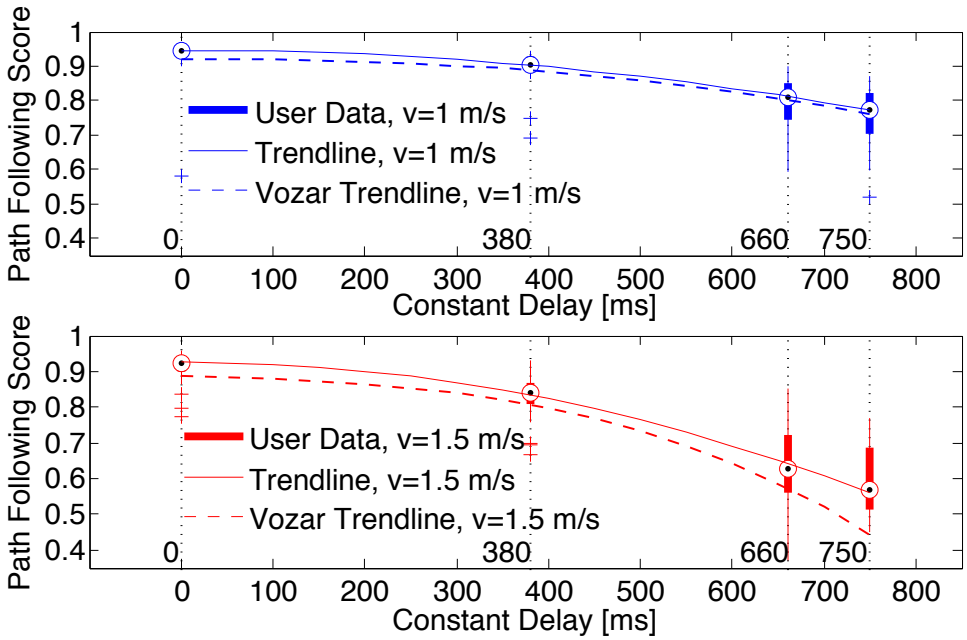

Figure 4.6: Comparison of path following performance of users from the current study (boxplots and solid trendline) with performance of the median user in Vozar's study (dashed trendline) [109]. The top plot is data for robot speed 1 m/s and the bottom is for 1.5 m/s.

constant delays tested in Vozar's study [109].

Looking at Figure 4.6 (top), the median path following scores at constant delays in this study and Vozar's study are very similar for the 1 m/s robot speed. Given the spread of the scores at higher constant delays of 660 and 750 ms, the difference between the median user in this study and Vozar's is hardly distinguishable at 1 m/s. However, in Figure 4.6 (bottom) the median path following scores at constant delays in this study seem to be slightly higher compared to those in Vozar's study for the 1.5 m/s robot speed. In general at 1.5 m/s, the median user with Vozar's simulated kinematic robot had a score that was close to the 25th percentile of users who drove the simulated dynamic robot. Insights on this difference are included in Section 4.4.

Table 4.2: Steering model controller gains for dynamic robot model in this study and kinematic robot model in Vozar's study [109].

		Constant Delay [ms]					
		0	250	380	500	660	750
Dynamic Model	K_p	1.9	-	1.4	-	0.9	0.8
v=1 m/s	K_d	0	-	0.5	-	0.6	0.7
Kinematic Model	K_p	1.7	1.6	-	1.3	-	1.0
v=1 m/s	K_d	0	0.3	-	0.7	-	1.0

4.3.2.2 Steering Model - Hypothesis (d)

To explore whether the steering model in [109] could be used to simulate path following performance of users in this study, the steering model was implemented as described in Section 4.2.1.1 and Figure 4.1. We used the same lookahead time ($T_p = 1250$ ms), physical actuation time ($\delta_h = 200$ ms) and control input threshold ($\mu = 0.5$) as [109]. The only modification was the selection of control gains K_p and K_d.

To select the control gains, a grid of K_p and K_d values were simulated with the controller at each of the constant delays tested by users. For each constant delay, the gains were selected to minimize two criteria between the steering model and median human user: the difference in path following score and difference in effort. Effort was defined to be the average magnitude of the input given by the main controller over the duration of a trial [109]. The tuned gains for the dynamic robot model are displayed in the top of Table 4.2 along with gains used in [109] with the kinematic robot model and tested delays.

The human drivers and steering model are compared across all delays tested for robot speed 1 m/s in Figure 4.7. Each steering model boxplot in Figure 4.7 is comprised of 30 simulations. Note that for each time-varying delay, the steering model gains were selected to be the same as the hypothesized constant delay with simi-

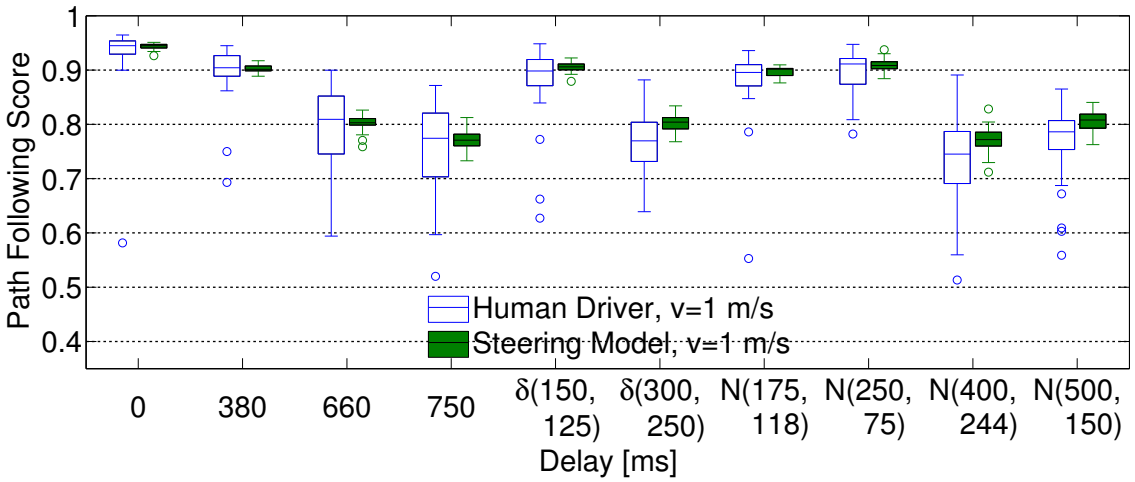

Figure 4.7: Side-by-side comparison of path following performance for human drivers in this study to steering model for robot speed 1 m/s. The steering model gains used for $\delta(150, 125)$, $N(175, 118)$, $N(250, 75)$ were the same as for 380 ms - $K_p = 1.4$, $K_d = 0.5$. The steering model gains used for $\delta(300, 250)$, $N(400, 244)$, $N(500, 150)$ were the same as for 660 ms - $K_p = 0.9$, $K_d = 0.6$.

lar path following performance. That is, steering model gains used for $\delta(150, 125)$, $N(175, 118)$, $N(250, 75)$ were the same as for 380 ms - $K_p = 1.4$, $K_d = 0.5$. The steering model gains used for $\delta(300, 250)$, $N(400, 244)$, $N(500, 150)$ were the same as for 660 ms - $K_p = 0.9$, $K_d = 0.6$.

Notice that the IQRs of the steering model in Figure 4.7 are much smaller than those of the human drivers. This observation is not surprising, however it makes quantitative comparison in the form of null hypothesis tests ill-conditioned. Thus, we will compare human driver and steering model data qualitatively. Figure 4.7 shows that the steering model appears to be a good predictor of the median user's performance - nearly all of the IQRs of the steering model fall within the IQR of the human driver data. Therefore, Figure 4.7 supports our hypothesis that the steering model developed in [109] can predict user performance with our dynamic robot simulation.

4.4 Discussion

Our hypothesis that path following performance would be the same among time-varying delays with the same expected value and standard deviation (distributions $\delta(150, 125)$, $N(250, 75)$ in hypothesis (a) and $\delta(300, 250)$, $N(500, 150)$ in hypothesis (b)) was well supported by user test results. Despite the difference in distribution shape, there was no statistically significant difference in path following scores. Thus, path following performance under delay is most affected by the expected value and standard deviation of the delay distribution, rather than shape.

However, our hypothesis that path following scores would be the same among time-varying delays related by Eqn. (4.5) had mixed results (distribution $N(175, 118)$ in hypothesis (a) and $N(400, 244)$ in hypothesis (b)). Three of the four ANOVA tests showed no significant difference, but results with $N(400, 244)$ at 1 m/s did show a significant difference. Selecting $N(400, 244)$ was an extrapolation from our two data points of $N(500, 150)$ and 660 ms. Perhaps interpolating between our two data points would have resulted in closer performance among the different distributions. Further exploration of this equivalence among time-varying delays is required. For example, the relationship between expected value and standard deviation could be nonlinear or the relationship in Eqn. (4.5) could actually depend on variance instead of standard deviation. Additionally, it would be more succinct if α and β in Eqn. (4.5) did not change for different equivalent constant delays. Perhaps a least squares estimate of α and β could be used across multiple equivalent constant delays.

Our user tests supported the time-varying to constant delay equivalence predicted by [109] (distributions $\delta(150, 125)$, constant 380 ms in hypothesis (a) and $\delta(300, 250)$, constant 660 ms in hypothesis (b)) . This equivalence in performance among constant and time-varying delay distributions of different shapes could be valuable for designers of teleoperation systems. For example, it can reduce the number of delay distributions tested with humans, saving time and money.

Path following performance of the median user at constant delays in this study were compared to those in [109]. A qualitatively similar trend to [109] was observed in this study, however performance was slightly better. We believe this difference is due to the uniform distribution noise added to the main controller inputs. The dynamic robot model acts like a low pass filter, attenuating the effects of noise added to the inputs. Overall, the robot models in both studies were effective at showing delay's impact on path following performance. Quantitatively, path following scores will change with different robots and different tracks, but the overall trend should be the same.

The steering model needed minor retuning to fit our dynamic robot model. Lookahead time T_p, physical actuation time δ_h, and control input threshold μ were all unchanged. Gains K_p and K_d were adjusted, but differ by less than 30% from the values in [109]. The gains also follow the same trend as in [110] work - K_p decreases and K_d increases as delay increases. Driver performance could also likely be predicted using the steering model for delays not tested explicitly, but within the range of those that the model was fit to. For example, future work could consider predicting human performance at 700 ms by running the steering model with K_p and K_d gains interpolated between 660 and 750 ms.

The results in this chapter promise to reduce the testing burden for future robot designers. (1) Trends in path following performance can be predicted using simple kinematic simulation models; the development of detailed dynamic models can be delayed until first-round design design decisions have been made. (2) Even when time-varying delays are expected in the feedback loop, testing can be done with a set of constant delays, and the performance under time-varying delays can be inferred based on equivalent delays. (3) Simple user models consisting of a PD controller with lookahead can be used to predict the performance of human drivers.

4.5 Conclusions

This chapter presented results from simulations and a user study exploring the impact of delay on path following performance for a teleoperated mobile robot. Simulations with the steering model and user test results support the proposed equivalence in path following performance among different time-varying delay distributions with the same expected value and standard deviation. A general rule for equating path following performance among time-varying delay distributions with different expected values and standard deviations was proposed. The extension of Vozar's steering model [108] to a dynamic robot simulation was demonstrated. Additionally, user test results with both the simulated kinematic and dynamic mobile robots demonstrated the same effects of delay on path following performance.

CHAPTER V

Obstacle Avoidance and Its Interaction with Communication Delay

5.1 Introduction

Many mobile robot operation scenarios will continue to require some human knowledge or expertise. For example, Cosenzo and Barnes claim most military robotic systems will require active human control or at least supervision with the ability to take-over if necessary [89, Ch. 2]. This chapter considers a shared control method used in mobile robot navigation for a basic search task (such as in reconaissance or search and rescue). A new technique for representing obstacle free regions that works better in unstructured environments for highly maneuverable mobile robots is implemented. A set of two user studies are presented that investigate the effects of communication delay and shared control on performance.

This chapter is based on publications [98, 99]. The remainder of this chapter is organized as follows. Section 5.2 describes the obstacle representation technique developed and the shared control method it was integrated into. Section 5.3 describes the setup for two human subject studies we conducted. Sections 5.4 and 5.5 present results and discussion about the relationships we developed. Lastly, Section 5.6 gives a summary of the chapter.

5.2 Shared Control Method

The automation available on the robot is assumed to have some computational resources and capabilities to sense obstacles, however it does not have the ability to complete the search task alone (*e.g.* it cannot detect certain objects of interest). The shared control method presented in this chapter uses model predictive control (MPC) to handle both obstacle avoidance and control arbitration.

5.2.1 Model Predictive Control Formulation

The shared control method is formulated using MPC in the following way,

$$\min_{\boldsymbol{u_a}} \sum_{i=k}^{k+p} J \tag{5.1}$$

$$\text{subject to } \boldsymbol{x_{i+1}} = A\boldsymbol{x_i} + B\boldsymbol{u_{a,i}} \tag{5.2}$$

$$F_i\boldsymbol{x_i} \leq G_i \tag{5.3}$$

$$\boldsymbol{u_{\min}} \leq \boldsymbol{u_{a,i}} \leq \boldsymbol{u_{\max}} \tag{5.4}$$

where $\boldsymbol{u_a}$ is the set of inputs to the system calculated to minimize the cost function J (described in Section 5.2.4) over p prediction steps. The set of inputs $\boldsymbol{u_a}$ consists of forward speed and turn rate.

Variable $\boldsymbol{x}$ represents the robot states. The first constraint in Eqn. (5.2) describes the evolution of the robot states using a set of discretized linear state space equations (see Section 5.2.3). The second constraint in Eqn. (5.3) describes the restrictions on the robot states. This constraint is used to restrict the position of the robot such that it will not collide with obstacles in the environment. The third constraint in Eqn. (5.4) limits the range of inputs that the MPC can select to those that are feasible for the robot.

Note that in the formulation of Eqn. (5.1)-(5.4), all constraints are linear. Ad-

ditionally, we select the cost function J to be quadratic. Thus, our shared control method is represented as a convex quadratic programming problem and can be efficiently solved. We use the CVXGEN tool to generate a solver that can easily find a solution to our problem in real-time [73].

5.2.2 Obstacle Constraint Representation

When obstacles are placed in a robot's environment, they create a "hole" in the obstacle-free space. The representation of this obstacle-free space becomes non-convex. Solving optimization problems with non-convex constraints is challenging to do rapidly and many researchers have proposed convex approximations for obstacle-free regions as described in Section 2.1.2. This section will describe the convex approximation method we developed for the MPC problem and why we believe it is well suited for highly maneuverable robots in unstructured environments.

5.2.2.1 Convex Approximations

To describe a convex region free of obstacles, one could cleverly place a set of hyperplane inequalities to exclude all obstacles. In most cases the resulting region would miss much of the obstacle free region. To overcome this shortcoming, prior work has suggested defining many convex regions to approximate a non-convex space. Recall that the MPC problem consists of solving Eqns. (5.1)-(5.4) over the prediction horizon. A challenge that arises when considering multiple convex regions is determining which convex region to consider at each step of the prediction horizon. Two approaches have been used to address this challenge. The first is to make the selection of the convex region part of the solution by formulating a multi-stage optimal control problem [64] or a mixed-integer programming problem [28]. The second is to estimate which convex region should be used at each step of the prediction horizon and specify that estimate to the MPC problem [5, 35].

The first approach creates an optimal control problem that is much easier to solve than the original problem with non-convex constraints. The optimal control problems can be solved on the order of seconds and works well for robots with full autonomy or supervisory control that can pre-plan large portions of movement without human operator input [28]. However, for shared control with the human operator providing regular inputs, the MPC problem must be solved more rapidly - several times per second. By estimating which convex region should be used at each step of the prediction horizon, the resulting MPC problem can be solved multiple times per second.

The methods developed by [6] and [35] consider Ackermann steer vehicles moving at higher speeds. They construct corridors (referred to as an environmental envelope in [35] and a homotopy in [6]) consisting of multiple convex regions to describe the area that the vehicle will travel through to reach its end goal. Based on the corridors and an assumed forward velocity for the vehicle over the prediction horizon, they apply constraints on the lateral position of the vehicle to avoid colliding with obstacles. This method has been shown to work well with full size Ackermann steer vehicles.

However, the corridor method is not well suited for skid-steer robots operating at lower speeds in environments without paths or roads. Under these operating conditions, the robot may not drive down a corridor straight ahead in between obstacles. Instead it may turn sharply to the right or completely around to head in the opposite direction and the lateral constraints from the corridor method could be meaningless for the obstacles in the new direction the robot is heading.

For these conditions with a highly maneuverable robot in an unstructured environment, we propose a new method for representing convex, obstacle free regions. Using a set of hyperplane inequalities, we define a convex, obstacle free region that contains the robot's initial position. Over the course of the MPC prediction horizon, this convex, obstacle free region changes in shape to include more area in the direction that the robot is predicted to move, while still making sure the robot's initial position

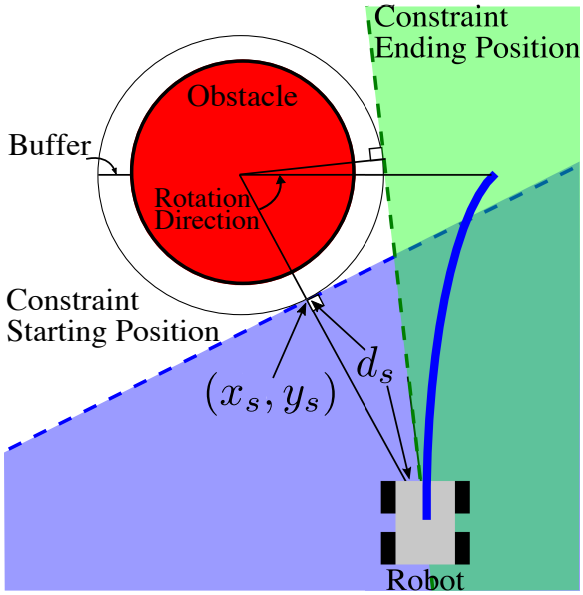

Figure 5.1: Diagram describing obstacle representation as linear constraints in MPC problem formulation.

is contained in the final region.

For simplicity, we will describe the method used to construct the convex, obstacle free regions for the 2D case, but the method could be extended to the 3D case (*e.g.* for quadcopters or underwater robots). In the 2D case, the hyperplane inequalities are simply straight line, linear inequalities. We consider one linear inequality for each of the n closest obstacles. In our implementation, we considered $n = 5$, but one could consider a higher n if the obstacles were more densely distributed or it was desirable to consider a larger area around the robot. Drawing inspiration from the spacecraft motion planning field, we will rotate the linear constraint around the obstacle [85] to represent the convex, obstacle free region.

5.2.2.2 Algorithm Description

Our method is described in Algorithm 1 with descriptions for each of the sub-routines in the text that follows. Refer to Figure 5.1 for additional description. In general, Algorithm 1 can be applied to environments with obstacles that are convex

```
Data: Robot State, Obstacle Positions, Human Inputs
Result: Convex Obstacle Free Regions for Each Step of Prediction Horizon
GeneratePredictedRobotPath
for closest obstacle to farthest obstacle do
    CalculateStartingLineEquation
    DetermineRotationDirection
    CalculateRotationRate
    CheckFeasibleRegion
    if FeasibleRegion=∅ then
        RotationRate ← 0
    end
end
for entire prediction horizon do
    for all obstacles within range do
        CalculateLineEquation
    end
end
```

Algorithm 1: Pseudocode to Generate Convex Obstacle Free Regions

or can be bounded by ellipsoids.

GeneratePredictedRobotPath: First, a predicted path for the robot over the prediction horizon is generated. The predicted path is calculated based on a predicted set of inputs applied to a model of the robot. For prediction horizons on the order of one second, prior work has shown that a zero-order hold of the human's current input can be a reasonable estimate [21]. That is, the human's current input is assumed to remain constant over the prediction horizon. For longer prediction horizons, one could consider using learning based prediction methods, *e.g.* [30]. In our implementation, we will use a simple zero-order hold model to predict human operator inputs. We apply the predicted human inputs to the robot model used in the MPC formulation (Eqn. (5.2)) and obtain the robot's predicted path.

CalculateLineStartingEquation: Next, for each obstacle, the starting position of the constraint (at the first time step of the prediction horizon) is calculated to be tangent to an ellipse bounding the obstacle at the point on the ellipse closest to the center of the robot. If the ellipse is a circle (the case considered in our user study),

then the distance d_s to the point (x_s, y_s) closest to the robot on the circle is,

$$d_s = \frac{\left(\sqrt{(x_o - x_r)^2 + (y_o - y_r)^2} - r_o\right)}{\sqrt{(x_o - x_r)^2 + (y_o - y_r)^2}} \tag{5.5}$$

$$x_s = d_s (x_o - x_r) + x_r \tag{5.6}$$

$$y_s = d_s (y_o - y_r) + y_r \tag{5.7}$$

where (x_r, y_r) is the robot position, (x_o, y_o) is the position of the center of the obstacle, and r_o is the obstacle radius. The equation of the line tangent to the circle and passing through the point (x_s, y_s) is,

$$(x_o - x_r) x + (y_o - y_r) y = (y_o - y_r) y_s + (x_o - x_r) x_s \tag{5.8}$$

The inequality sign is then selected such that any point inside the obstacle does *not* satisfy the inequality.

DetermineRotationDirection: The linear inequality will then rotate in one direction around the surface of the ellipse. The rotation direction is determined based on the robot's predicted path. Consider a line connecting the center of the robot to the center of the ellipse bounding the obstacle. Consider a second line connecting the end point of the robot's predicted path to the center of the ellipse bounding the obstacle. The direction that forms an angle less than 180^o between these two lines will be the rotation direction of the constraint.

The rotation rate for each obstacle linear inequality will depend on the end position of the constraint and the length of the prediction horizon. The end position of the constraint is defined to be tangent to the ellipse bounding the obstacle and pass through the center of the robot. Note that two lines will satisfy these criteria. The rotation direction determines which line to use, *i.e.* the line that is encountered first

when rotating the starting constraint in the rotation direction and keeping it tangent to the ellipse.

CalculateRotationRate: The rotation rate will be calculated as a percentage of the angle between the constraint starting position and ending position divided by the prediction horizon. Using a percentage of the rotation between the constraint start and end positions produces an obstacle free region that offers a wider feasible region near the robot's initial position. A higher percentage will give a wider feasible region further away from the robot. In our experiments, we used 90% of the rotation between the constraint start and end position.

CheckFeasibleRegion: If the calculated rotation directions cause the linear inequalities to create an empty obstacle free region, then the constraint for the obstacle farthest away (by Euclidean distance) is not rotated. Having to set the rotation rate of the linear constraints to zero does not occur often. In fact, in our user study experiments (described in Section 5.3) we did not have to remove the rotation on any of the linear inequalities due to generating an infeasible region.

CalculateLineEquation: Lastly, for each step in the prediction horizon, the obstacle-free convex regions are defined by constructing a set of linear inequalities. The linear inequalities are calculated by rotating each obstacle constraint from its starting position by its rotation rate around the edge of the ellipse. The output of the convex obstacle free region algorithm is a set of linear inequalities fed into Eqn. (5.3). The general form of $F_i \in \mathbb{R}^{n\times 5}$ and $G_i \in \mathbb{R}^n$ is,

$$F_i = \begin{bmatrix} a_{j,i} & b_{j,i} & 0 & 0 & 0 \end{bmatrix},\ G_i = \begin{bmatrix} c_{j,i} \end{bmatrix} \tag{5.9}$$

where $a_{j,i}$, $b_{j,i}$, and $c_{j,i}$ are the coefficients for the linear inequalities. Subscript $i = k, \ldots, k+p$ refers to the step of the prediction horizon and $j = 1, \ldots, n$ refers to obstacle index.

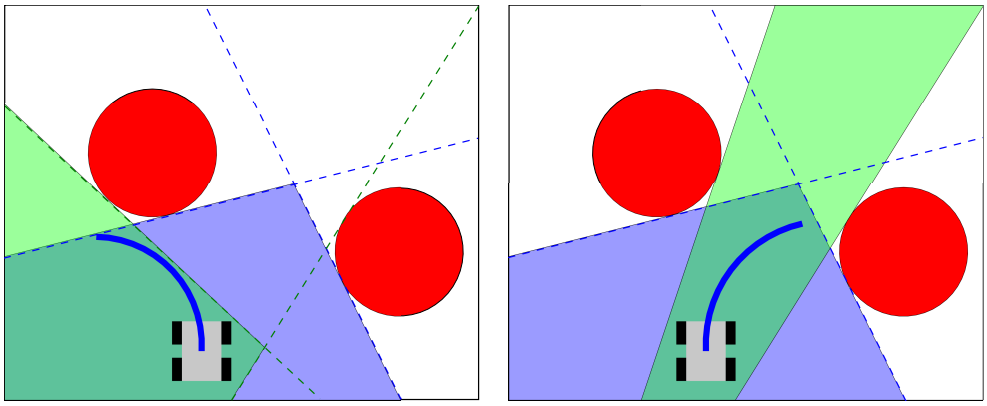

Figure 5.2: Obstacle constraints with multiple obstacles for two different predicted robot paths. The initial feasible region is represented by the light blue shaded area. The dashed blue lines rotate about the center of the obstacles they are tangent to over the prediction horizon until they reach the positions of the dashed green lines. The light green shaded area represents the feasible region at the end of the horizon.

5.2.2.3 Algorithm Discussion

A visual representation of generating the obstacle free convex regions is shown in Figure 5.2. The red circles represent obstacles. The left sub-figure considers a projected robot path (solid blue line) for a left turn and the right sub-figure considers a right turn. In each sub-figure, the obstacle free region at the start of the prediction horizon is represented by the lightly shaded blue region enclosed by the dashed blue lines. As the prediction horizon goes on, the linear inequalities tangent to the obstacle will rotate around until it reaches the dashed green line at the final step of the prediction horizon. The obstacle free region at the end of the prediction horizon is represented by the lightly shaded green region enclosed by the dashed green lines. One can see that as the robot's predicted position moves towards the obstacles, the constraints will adjust to allow for a wider range of motion while still keeping the robot's initial position in the feasible region.

One note on implementation issues: the robot may end up near the edge of the feasible region and small errors may push the robot into the infeasible region. To accommodate this issue, the ellipse enclosing the obstacle can include an added buffer

zone, so that it is slightly larger than the obstacle it is enclosing (see Figure 5.1). In our implementation, we used a buffer of 0.15 m. When the robot is near the edge of the feasible region, the overall radius of the obstacle is reduced by 0.05 m to move the robot out of the infeasible region.

Similar to most other convex approximation methods, our algorithm is a conservative estimate of the true non-convex space. The algorithm assumes the robot is not initially inside an obstacle and can be applied when obstacles can be bounded with a strictly convex curve, such as a circle or ellipse, that does not intersect with other curves bounding obstacles. The method could be extended to consider curves that are not strictly convex (*i.e.* curves that contain straight line segments) and curves that intersect. To consider curves containing line segments, one would need to develop a rule for determining the slope of the linear inequality at points where two line segments meet (*i.e.* where the slope may be undefined). If two or more convex curves bounding obstacles intersect, then one would likely want to impose a rule that the linear inequalities should *not* rotate along the curve past the intersection point.

Based on the algorithm we defined and conditions described above, we can guarantee the constraints generated will be linear and convex. This guarantee holds because the linear inequality calculated for each obstacle can be more generally described as a half-space, which is a convex set. The intersection of an arbitrary collection of convex sets is convex [87, Ch. 2].

Our algorithm can generate constraints very rapidly, making it well suited for optimization problems that need to be solved on the order of milliseconds. It is best suited for shorter prediction horizons. As the prediction horizon increases, the algorithm may feel more limiting because the robot's initial position is kept in the feasible region. The algorithm can apply constraints to robot positions in 2D and 3D space, rather than the 1D constraints on lateral position with the environmental envelope method [35] that are often used with Ackermann steer vehicles moving forward at

Figure 5.3: Exocentric view of the simulated robot that was teleoperated in the human subject studies. Same as Figure 6.2.

constant speeds. Thus, our obstacle constraint representation is better suited for skid-steer and omni-directional robots.

5.2.3 Robot Model

This analysis considers the simple skid-steer robot shown in Figure 5.3. A linearized model based on a dynamic unicycle robot model [15] is used in the MPC formulation. This type of model is commonly used to describe the behavior of skid-steer and differential drive robots.

$$\begin{cases} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \omega \\ \dot{v} &= \frac{K_1}{m}(v_{\text{des}} - v) \\ \dot{\omega} &= \frac{K_2}{J}(\omega_{\text{des}} - \omega) \end{cases} \tag{5.10}$$

In the equations above x, y represent the robot's planar position, θ is heading angle, v is forward velocity, and ω is angular turn rate. Physical robot parameters are represented by m for mass and J for rotational inertia. Inputs are desired forward

velocity v_{des} and desired angular turn rate ω_{des}. Constants K_1 and K_2 determine how well the system tracks the desired inputs.

In order to make Eqn. (5.10) more efficient for use in the MPC formulation, the equations are first linearized by applying small angle approximations. This approximation changes the first two equations of Eqn. (5.10) to $\dot{x} = v$ and $\dot{y} = v_0\theta$, where v_0 is the initial robot forward velocity. The linearized set of equations can easily be put into state space form with states $\boldsymbol{x} = [x\ y\ \theta\ v\ \omega]^T$ and inputs $\boldsymbol{u} = [v_{\text{des}}\ \omega_{\text{des}}]^T$. The state equations are expressed as $\dot{\boldsymbol{x}} = A_c\boldsymbol{x} + B_c\boldsymbol{u}$, where

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & v_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{-K_1}{m} & 0 \\ 0 & 0 & 0 & 0 & \frac{-K_2}{J} \end{bmatrix} \tag{5.11}$$

$$B_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{K_1}{m} & 0 \\ 0 & \frac{K_2}{J} \end{bmatrix} \tag{5.12}$$

Values for K_1 and K_2 can be selected by solving the linear quadratic regulator (LQR) problem to approximate the behavior of the speed controller onboard the actual robot. Lastly, Eqns. (5.11) - (5.12) must be evaluated at an initial forward velocity and discretized to be used in the MPC formulation of Eqn. (5.2). The discretization was performed using a zero-order hold on input $\boldsymbol{u}$ for timestep T_s.

5.2.4 Cost Function

Many different cost functions could be selected for this shared control method. To enable solving in real-time, we selected a quadratic cost function,

$$J\left(\boldsymbol{u}_{a,i}, \boldsymbol{u}_{h,i}, x_i\right) = \left(\boldsymbol{u}_{a,i} - \boldsymbol{u}_{h,i}\right)^T R_i \left(\boldsymbol{u}_{a,i} - \boldsymbol{u}_{h,i}\right) \tag{5.13}$$

Notice that the cost function selected only depends on the shared control input $\boldsymbol{u_a}$ and estimated human input $\boldsymbol{u_h}$. The current form does not depend on the robot state $\boldsymbol{x}$, but easily could if desired. Since, $\boldsymbol{u}_{a,i} \in \mathbb{R}^2$ and $\boldsymbol{u}_{h,i} \in \mathbb{R}^2$, $R_i \in \mathbb{R}^{2\times 2}$. We will select R_i to be diagonal and positive semidefinite. Thus, the first element of R_i will correspond to the weight on robot forward speed and the second element will weight the robot turn rate.

Hauser suggests decreasing the weightings on the cost at later times in the prediction horizon [41]. In our implementation of the shared control method we will have the weightings in R_i decrease linearly to 10% of their initial values as timestep i increases to p. Doing so effectively puts less weight (or cost) on matching the estimated human inputs at timesteps further into the future, which naturally accounts for increased uncertainty in the estimates of the human inputs at later prediction times.

The cost function we selected is similar to that proposed by [21]. The main differences are the ability to weight the cost on forward speed and turn rate in different proportions and the decreasing weights over the prediction horizon.

With the cost function selected in Eqn. (5.13), the solution to Eqn. (5.1)-(5.4) will match the estimated human input as closely as possible without violating constraints on the robot dynamics or colliding with obstacles. If the estimated human input is not predicted to cause any collisions, then the solution $\boldsymbol{u}_a$ will match $\boldsymbol{u}_h$ exactly.

Alternatively, the cost function can be easily modified to instead follow a desired

path defined by a set of robot states,

$$J\left(\boldsymbol{x}_{a,i}, \boldsymbol{x}_{h,i}\right) = \left(\boldsymbol{x}_{a,i} - \boldsymbol{x}_{h,i}\right)^T Q_i \left(\boldsymbol{x}_{a,i} - \boldsymbol{x}_{h,i}\right) \tag{5.14}$$

This variation of the cost function depends on the shared control method state $\boldsymbol{x_a}$ and human's desired state $\boldsymbol{x_h}$. As a result, the shared control will attempt to match its own robot state with the human's desired robot state as closely as possible. Although this cost function allows the user to provide a set of robot states over the prediction horizon, we will only be estimating the user's desired position. Thus, $Q_i \in \mathbb{R}^{5\times5}$ will be a positive semi-definite and diagonal matrix as before but with zero weights associated with the heading angle, forward velocity and angular turn rate.

The modified cost function in Eqn. (5.14) can be used for shared control modes that are able to estimate a desired path to follow or a desired waypoint.

5.2.5 Human Model

The shared control method presented in this chapter does not assume knowledge of an end goal position for the robot. Instead, the MPC method tries to match an estimated human input or goal position. Recent work has proposed a method of predicting user intention for manipulator arm control via learning from training data [30]. Similarly, Shia *et al.* have a method of creating a probabilistic driver model that can be fit to training data [93]. Bohren *et al.* do not require training data to predict manipulator arm movement in teleoperation [11]. Instead they propose a method of predicting movement intent from a graph of possible actions.

While these methods have been shown to offer improvements in performance, prior work has also shown that even simple prediction methods (such as a zero-order hold) of the human's input can be effective for short horizons. For example, Chipalkatty, Droge and Egerstedt have shown that using a zero-order hold to estimate the human

operator inputs performs as well as a least squares system identification method [21]. Since the MPC problem formulation in our human subject study is solved at a rapid rate and the prediction horizon is relatively short (0.5-1.5 s), we will consider a zero-order hold of the human operator's current input. That is, each time the MPC problem is solved, the values of $u_{h,i}$ are set to whatever the human operator's current forward speed and turn rate inputs are. The zero-order hold input predictor does not require any training data or description of possible robot actions.

5.3 User Study Description

The shared control method developed was used in two sets of human subject studies investigating the effectiveness of the method and the impact of several critical teleoperation factors discussed in Section 2.2.

Subjects drove a mobile robot around an environment filled with obstacles to search for objects of interest (OOIs) scattered throughout the space, similar to search and exploration mission. Subjects were told the robot could sense obstacles and help avoid them in shared control mode, but the robot was not capable of sensing the OOIs. Due to time limits we imposed, it was not possible for the robot to explore the entire area. Subjects had to prioritize which areas they searched.

As a motivating example, a robot could be enlisted to search for survivors in a disaster area, but not have the proper sensor that can distinguish humans from animals or other objects. There may not be time to add the sensors that could make such a distinction in recognizing survivors. Additionally, if time is critical, it may not be ideal for the robot to exhaustively search the whole area. A person experienced in disaster response may have expertise in prioritizing search areas that are more likely to have survivors. That person would want to have more control over where the robot navigates and their decision on where to explore next could be triggered by subtle observations.

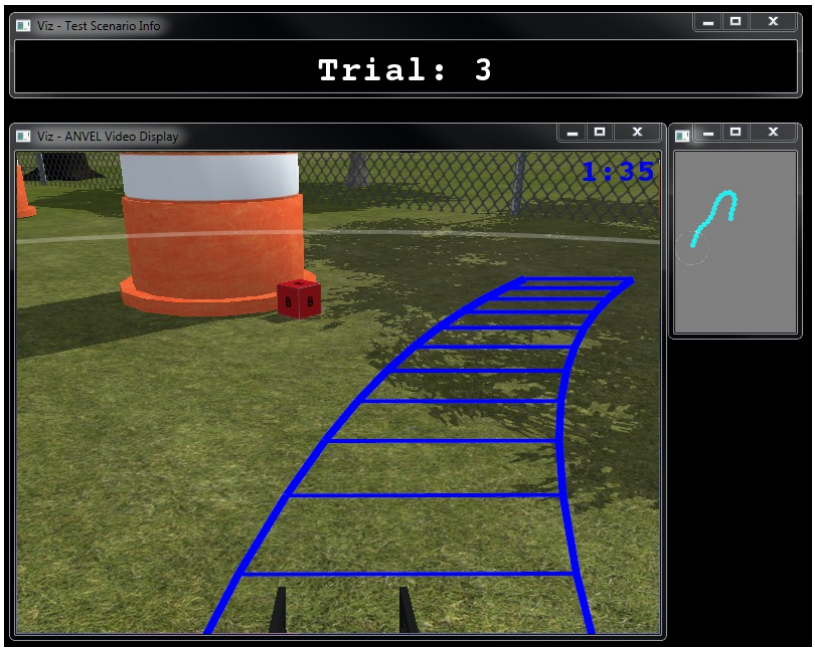

Figure 5.4: Visual display shown to participants for Interfaces A, B, and C in the human subject studies.

5.3.1 Robot Environment

In both human subject studies, participants performed a timed search task. Subjects operated a virtual skid-steer robot in an environment simulated with ANVEL [31]. ANVEL is free robot and vehicle simulation tool developed by Quantum Signal, LLC. It is built on popular open source libraries including the open dynamics engine (ODE) for simulating nonlinear dynamics/physics and the object-oriented graphics rendering engine (OGRE) for generating high fidelity, realistic graphics.

Each environment consisted of a rectangle enclosed by concrete barriers or chain-link fence. The dimensions of the environment, number of obstacles, and number of OOIs was the same across trials in each user study. However, the placement of the obstacles and OOIs were randomized in each trial. The OOIs were represented in the environment using small colored boxes with a letter printed on each side of the box face. A sample OOI can be seen next to the barrel in Figure 5.4.

The robot was assumed to have sensors (*e.g.* LIDAR, stereo camera, sonar) that

could sense obstacles locally. Only obstacles within the robot's sensing field of view ($\pm 90^o$) and distance (4 m) were considered when solving the MPC problem.

5.3.2 User Interface

A total of five different user interfaces were used. Details on which user interfaces were used in each study will be made clear in the test condition description (Section 5.3.4). The interfaces were:

- Interface A - human operator manually controls robot velocity commands.
- Interface B - shared control with autonomy trying to match human operator velocity commands. Autonomy is located on-board the robot.
- Interface C - shared control with autonomy trying to match human operator velocity commands. Autonomy is located at the operator control unit.
- Interface D - shared control with autonomy following Voronoi map paths based on human operator commands. Autonomy is located on-board the robot.
- Interface E - shared control with autonomy tracking towards a waypoint. Human operator actively controls the waypoint location. Autonomy is located on-board the robot.

As many features as possible were attempted to be kept the same across interfaces. The basic visual display shown to users in the human subject study is shown in Figure 5.4. The largest display shown in Figure 5.4 is a simulated video feed from a camera attached to the robot's manipulator arm. The video was displayed at 25 fps with a resolution of 640x480 pixels. The video has a few annotations to better help users complete each trial. In the top right corner of the video display is a clock that counts down time remaining in the current trial. The translucent white arc extending from the left to right side of the video screen indicates the robot's maximum sensing

distance of 4 m. Obstacles within this arc can be detected/avoided and OOIs can be identified.

The last annotation feature on the video display are the blue lines, which represent the projected path of the left and right side wheels of the robot over the prediction horizon. For Interface A, the blue lines show the robot's predicted path using the zero-order-hold human model and ignoring potential collisions with obstacles. For Interfaces B-E, the blue lines show the robot's predicted path from the model predictive control (MPC) problem solution.

Since much of the environment looks similar, an overhead miniature map is provided on the top right side of the video display. As subjects drive the robot around, teal dots are displayed on the map every two seconds to indicate areas the robot has been. This feature was included to help subjects travel to areas they had not explored, but did not provide enough information to navigate without the video display.

Test subjects controlled the simulated robot using a wireless XBox controller. For Interfaces A-D, driving controls were similar to those of racing games, where the right trigger was used to control forward speed and the left joystick controlled turn rate. For Interface E, the right trigger also controlled robot forward speed, however the left joystick adjusted the position of the steerable waypoint on the screen. Subjects could turn in place using the right joystick in all Interfaces.

In addition to using the XBox controller for driving, subjects used the X-Y-A-B buttons to identify OOIs. The color-letter combinations of the OOIs were the same as the those on the XBox controller. In order for an OOI to be identified, it had to be within sensing range and the subject had to double tap the corresponding button. Once identified, a ding sounded and the OOI turned translucent.

The following is a more detailed description of the differences among interfaces.

5.3.2.1 Interface A

Human operators had no assistance from the autonomy. Robot velocity commands from the gamepad were passed to the robot, regardless of whether they would cause a collision.

5.3.2.2 Interface B

The MPC problem is solved on-board the robot. As a result, the commands the MPC problem receives from the human operator will be delayed and the projected path from the MPC problem displayed on the human operator's camera view will be delayed. However, the information about robot state and obstacle location used in the MPC problem will be undelayed. Interface B uses the cost function in Eqn. (5.13).

5.3.2.3 Interface C

The MPC problem is solved at the operator control unit. As a result, the commands from the human operator and information displayed on the camera view will be undelayed. However, information about robot state and obstacle location used in the MPC problem will be delayed. Model-based predictors are used to obtain estimates of the robot and its environment's undelayed state information. Interface C uses the cost function in Eqn. (5.13).

Since the control commands will experience delay before reaching the robot, the commands calculated for the entire prediction horizon are sent from the operator side to the robot with a timestamp. Then, based on the difference between the current time and the timestamp of the delayed command, the robot selects the command intended for the current time. For example, if a packet of control commands is 300 ms old, then the robot will begin applying control commands starting 300 ms into the prediction horizon.

Given the success of predictive based control methods (*e.g.* MPC) and model

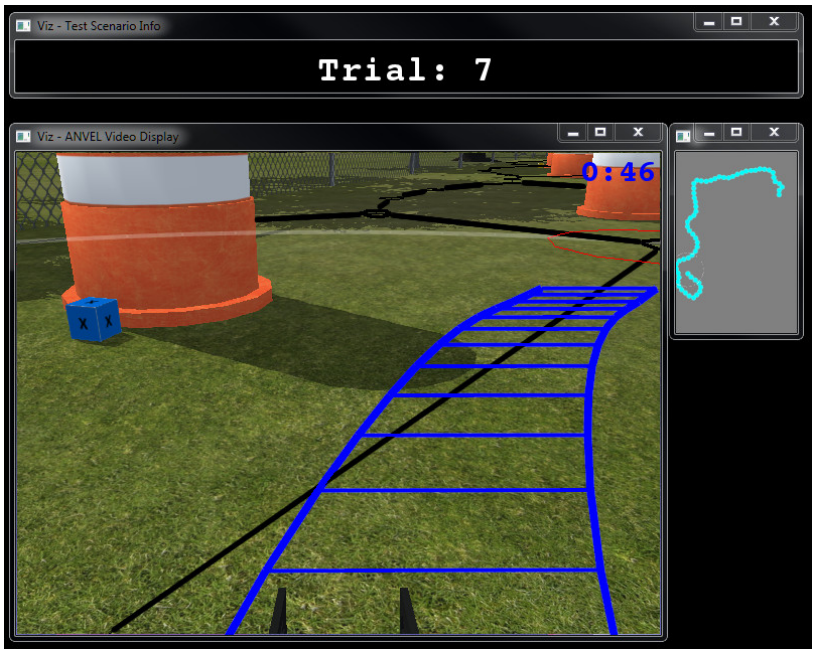

Figure 5.5: Visual display shown to participants for Interface D in the human subject study.

based predictors, we hypothesized that these predictive methods could better compensate for the delay than the human operator could. The robot's projected path will be more responsive to user inputs, but errors in the estimates of robot and environment state will likely result in more collisions.

5.3.2.4 Interface D

The MPC problem is solved on-board the robot. From the human operator model, a projected robot path is generated. The MPC problem will then be solved to have the robot move towards the point on the Voronoi map closest to the end-point of the robot's projected path. The Voronoi map is displayed using black-lines on the ground in the human operator's camera view (see Figure 5.5). The point on the Voronoi map that the MPC problem is currently trying to have the robot move towards is shown to the human using a red circle. Interface D uses the cost function in Eqn. (5.14).

The Voronoi map was generated beforehand for each environment. However, it would be feasible to generate the Voronoi map locally in real-time based on local

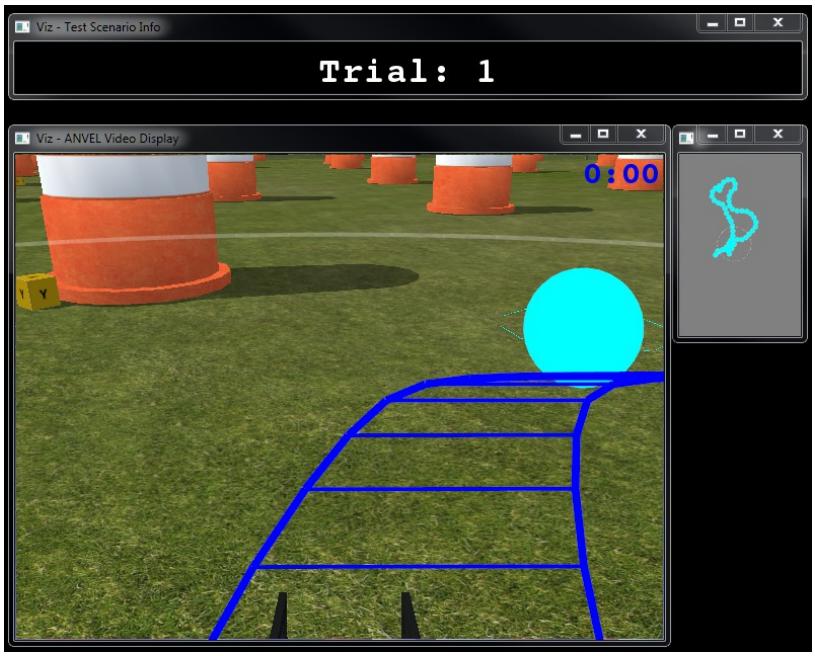

Figure 5.6: Visual display shown to participants for Interface E in the human subject study.

environment information.

5.3.2.5 Interface E

The MPC problem is solved on-board the robot. The human operator controls the (x, y) ground position of the steerable waypoint (teal circle) using the left joystick on the XBox gamepad (see Figure 5.6). The steerable waypoint is controlled in the robot's local camera frame. That is, as the robot drives towards the waypoint, the waypoint will remain in the same location in the camera view. The MPC problem will then be solved to have the robot move towards the location of the steerable waypoint. The robot's projected path (from the MPC problem solution) is displayed to the user via the blue lines. Interface E uses the cost function in Eqn. (5.14).

5.3.3 Test Procedure

Each study began with subjects filling out an informed consent form and answering basic background questions. Next, subjects underwent a training session to make sure

they had achieved a level of robot operation competency that would allow them to perform the search task. The training portion took approximately 10-15 minutes and consisted of the following parts.

1. Subjects were verbally instructed how to drive forwards, backwards and turn in an empty environment.
2. The robot's projected path represented by the blue lines was explained to subjects as they continued to practice driving in the empty environment.
3. Subjects practiced identifying a series of OOIs that were placed in a straight line ahead of them.
4. For each interface tested, subjects were placed in an environment with four obstacles and told to practice driving around them.
5. Subjects had the opportunity to complete a practice trial with each interface that was setup identically to the scored trials. In the practice trials, subjects experienced communication delay both in their commands sent to the robot and in the video sent back to them.

After the training and practice trials were completed, subjects moved on to the scored trials. Each scored trial was two minutes long. Participants were instructed to explore as much area as they could, identify as many OOIs as possible, and avoid collisions with obstacles. To incentivize participants to try their best in each of these three tasks, a bonus compensation was offered to participants with the highest scores. A combined score metric was explained to subjects and they were told the top score for each trial would be paid bonus compensation of $10 in addition to the $10 they received just for participating. The combined score metric was calculated as follows,

$$\text{score} = \frac{\text{\# OOI found}}{\text{total \# OOI}} + \frac{\text{\% area covered}}{100\%} - \frac{\text{\# collisions}}{\text{total \# obstacles}} \tag{5.15}$$

Table 5.1: Test conditions for each human subject study.

	Factor	Levels
Study 1	Interface	{A, B}
	Comm. Delay [ms]	{400, 800}
	Pred. Horizon [s]	{0.5, 1.0, 1.5}
Study 2	Interface	{B, C, D, E}
	Comm. Delay [ms]	{400, 800}
	Pred. Horizon [s]	{1.5}

Following each scored trial, subjects filled out a subjective survey to gauge their sense of presence and delay in the robot environment. After the scored trials and surveys, subjects were thanked for their participation and dismissed. Each user test took approximately one hour.

5.3.4 Test Conditions

Twenty different subjects were recruited for each of the studies. There were a total of twelve scored trials in the first study and eight scored trials in the second study. The first human subject study used a three-way repeated measures study design to evaluate human subject performance under communication delay, shared control, and different prediction horizons. The second study used a two-way repeated measures design to evaluate factors of communication delay and interface (see Table 5.1). The order of the trials was randomized among subjects.

The communication delay was introduced in two places - from human to robot (H2R) and from robot to human (R2H), as shown in Figure 5.7. The delay from R2H was selected to be larger than from H2R because we assumed additional delay would be introduced as a result of video processing and higher bandwidth requirements to send information from robot to human. Although wireless network communication delays are often time-varying, constant (time-invariant) communication delays were tested in the human subject studies. Constant delays were selected: 1) to reduce the number of factors in our user study design, and 2) because our prior work suggests that

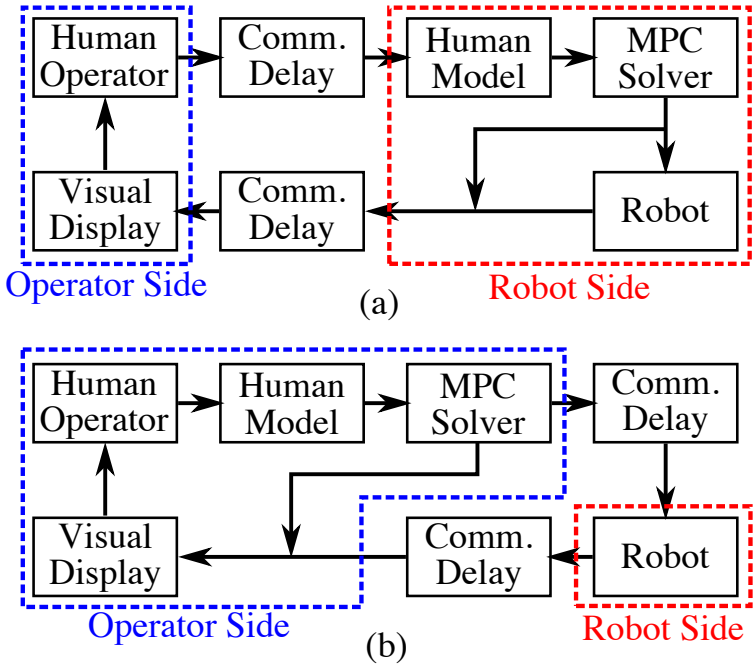

Figure 5.7: Functional diagram of information exchanged among components of the teleoperation system. Diagram (a) describes the arrangement for Interfaces A, B, D, E and (b) describes the arrangement for Interface C.

teleoperation performance with time-varying delays can be related to performance at constant delays [100]. The two communication delays tested were 400 ms (100 ms H2R, 300 ms R2H) and 800 ms (300 ms H2R, 500 ms R2H). These values are in line with those reported in studies on communication delay in video chat [114].

In user study 1, each arena had dimensions of 24×36 m and a total of 15 OOIs. We noticed that a couple of the top performers in user study 1 identified all 15 OOIs. Thus, for user study 2, the arenas were increased to have dimensions of 30×42 m and a total of 20 OOIs to prevent subjects from saturating the number of OOIs identified. A summary of the robot parameters used for the MPC are given in Table 5.2.

5.3.5 Performance Measures

As shown in Eqn. (5.15), three objective performance metrics were explained to human subjects to create the composite overall score. A brief description of how each measure was calculated and some comments on the maximum possible values of the

Table 5.2: UGV parameters used in human subject study.

Parameter	Description	Value
$v_{\max}$	max robot forward speed	2 m/s
$\omega_{\max}$	max robot turn rate	1.15 rad/s
m	robot mass	10 kg
J	robot rotational inertia	0.1 $kg \cdot m^2$
K_1	forward speed control gain	100.005 kg/s
K_2	turn rate control gain	1.005 $kg \cdot m^2/s$
T_s	discrete timestep	0.1 s

score are discussed.

5.3.5.1 Number of OOI Found

An OOI was considered to be found if it was in sensing range (within 4 m of the robot), in camera view, and the appropriate button was double-tapped by the human subject. A couple subjects identified all OOIs in user study 1, however even the best performance in user study 2 still missed 1 OOI. If an autonomous controller knew the location of each OOI before hand, using the traveling salesman problem we calculated that it was possible (given limits on the robot's velocities) to identify all OOIs.

5.3.5.2 Portion of Area Covered

The portion of area covered was calculated by adding up the total area that was seen by the robot's camera and was within the sensing range of the robot, then dividing it by the total area of the arena. Note that areas seen multiple times were only counted once and areas occluded by obstacles were not counted. In order to get an upper limit estimate of the maximum possible area coverage, we assumed a best possible scenario where the robot was trying to explore an arena with no obstacles in it. If the robot follows a path that spirals inward towards the center, does not have any overlapping area coverage, and moves at maximum velocity, then it would take 123.7 s to cover the entire area in user study 1 and 184.7 s user study 2. Following this same spiraling path inwards at maximum velocity, after 2 minutes the robot would

have covered 0.94 of the area in user study 1 and 0.67 of the area in user study 2.

5.3.5.3 Number of Collisions

A collision was counted as any time part of the robot made contact with an obstacle (*i.e.* construction barrel or wall). Collisions had to be at least one second apart to be counted as multiple collisions. The theoretical minimum number of collisions in each 2-minute trial is zero.

5.4 Results

User study 1 consisted of 14 male and 6 female test subjects with an average age of 21.9 years and standard deviation (sd) 4.1 years. On a scale of 1 (low) to 7 (high), subjects reported an average video game experience of 4.7 (sd=1.9) and an average familiarity with robotics of 3.6 (sd=1.3).

User study 2 consisted of 14 male and 5 female test subjects (one subject did not indicate their gender) with an average age of 22.0 years and standard deviation (sd) 3.4 years. On a scale of 1 (low) to 7 (high), subjects reported an average video game experience of 4.6 (sd=1.5) and an average familiarity with robotics of 4.1 (sd=1.7).

These tests were approved by the University of Michigan Health Sciences and Behavioral Sciences Institutional Review Board (UM IRB #HUM00044265).

Study participants were evaluated using the following metrics: portion of area covered, number of collisions, and number of OOIs identified. The error bars shown in all subsequent barplots represent standard error. In order to evaluate the effect of each of the manipulated variables on performance, mixed-effects models were fit to the data. The mixed-effects models were constructed with the lme4 package in R [8]. Confidence intervals for estimated parameters were constructed from profile deviance objects [9, Sec. 1.5] and the lmerTest package in R [59] was used to determine which effects were significant.

Results with the overall metric, number of OOIs identified, and portion of area covered exhibited similar trends. For conciseness, results and discussion will focus on area coverage.

5.4.1 Study 1 - Impact of Shared Control, Prediction Horizon, and Delay

From user study 1, two relationships will be highlighted. The first describes the impact of communication delay, shared control, and their interaction. The second explores the impact of prediction horizon in shared control.

To describe these relationships in a quantitative sense, a mixed-effects model was constructed for the portion of area covered metric. Main fixed effects for delay, interface, and prediction horizon were used. An interaction term was included between delay and interface. Other interaction terms were explored, but all had large p-values and thus were not ultimately included in the model presented in Table 5.3. Finally, each user was treated as a random effect on the intercept to help account for differences in participant skill. The general form of the model was,

$$\text{area} \sim c_0 + c_1 * \text{Interf.} + c_2 * \text{Delay} + c_3 * \text{Horiz.} + c_4 * \text{Interf.} \times \text{Delay} \tag{5.16}$$

where area is the portion of area covered. Interf. has values 0 representing Interface A and 1 representing Interface B. Delay has values 0 representing 400 ms and 1 representing 800 ms. Horiz. has values 0 representing 0.5 s, 1 representing 1.0 s, and 2 representing 1.5 s.

The average area covered and number of collisions for each interface $\times$ delay combination is shown in Figure 5.8. The mixed effects model coefficients, their confidence intervals, and statistical significance are displayed in Table 5.3. Based on Figure 5.8 and Table 5.3, it is evident that delay had the largest impact overall on the portion of area explored. The magnitude of the effect coefficient for delay was four times that

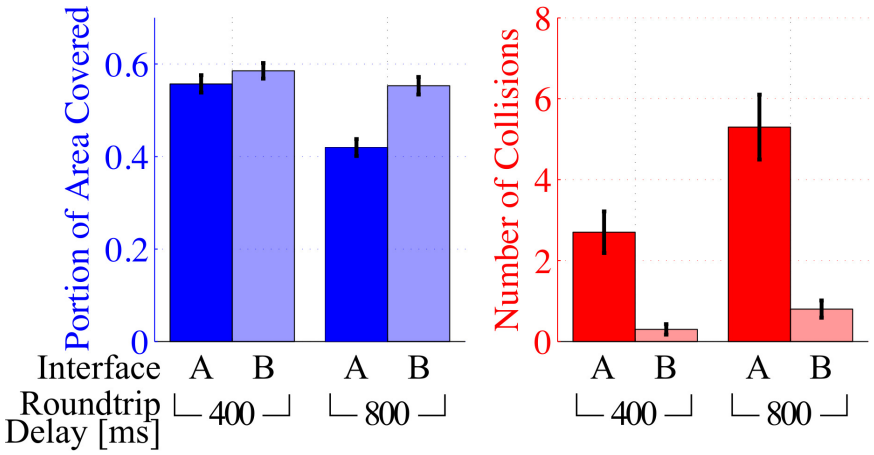

Figure 5.8: Shared control improved both performance metrics for each interface × delay combination tested in Study 1 but has a more dramatic improvement at the higher delay. The data shown is for a 1.0 s prediction horizon.

Table 5.3: Mixed-Effects Model: Portion of Area Covered (A|B). Delay has the largest impact on portion of area covered. The small mixed effects model coefficient for prediction horizon suggests that there is not a very strong relationship between prediction horizon and portion of area covered.

Effect	Coeff	95% CI	t-value	p-value
(Intercept)	*0.541*	*(0.505, 0.576)*	*30.38*	*<0.001*
Delay	-0.125	(-0.142, -0.107)	-13.63	<0.001
Interf. A|B× Delay	0.076	(0.050, 0.101)	5.86	<0.001
Interf. A|B	0.031	(0.013, 0.049)	3.39	<0.001
Prediction Horizon	0.012	(0.005, 0.020)	3.13	0.002

of interface. At the low delay, the shared control mode (Interface B) had a small positive effect on area explored, however the effect is much more pronounced at the higher delay of 800 ms.

From Figure 5.8 it is evident that delay and shared control (interface) both had large impacts on the number of collisions. Shared control decreased the number of collisions while higher delay resulted in more collisions. The difference in the number of collisions for each interface at the high delay is larger than the difference at low delay.

One may note that some collisions did still occur even with the shared control

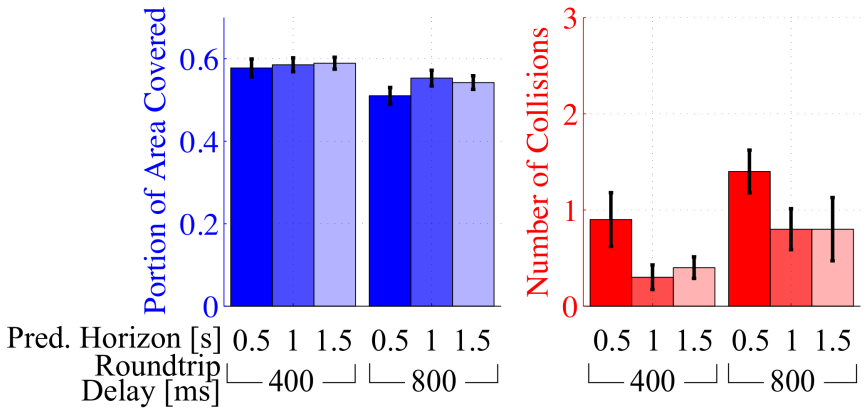

Figure 5.9: Varying lengths of prediction horizon did not have a large impact on the area covered or the number of collisions at each prediction horizon × delay combination tested in Study 1. However, increasing the prediction horizon from 0.5 s to 1.0 s resulted in fewer collisions. The data shown is for Interface B.

(Interface B). These collisions are the result of differences in the model used to solve the model predictive control (MPC) problem and the actual dynamics of the robot. The model used in the MPC problem is a linearized version of the unicycle model with dynamics. The robot operated by users in ANVEL has nonlinear dynamics that include tire friction models, drive motor models, etc. With a better model of the actual robot there would likely have been fewer collisions. However, the linearized robot model used in the MPC is computationally much more efficient. The differences between the two models give a more accurate representation of inaccuracies that would occur when using a physical robot.

Overall, results from study 1 show that shared control improves safety (fewer collisions) at both low and high communication delays. Performance (in terms of area covered) has a large improvement at high communication delay with shared control, but there is little improvement at low delay when adding shared control.

The average area covered and number of collisions for each prediction horizon × delay combination with Interface B is shown in Figure 5.9. The mixed-effects model in Table 5.3 shows that while the effect of prediction horizon on area covered is positive, it is small. In particular the difference between the 0.5 s and 1.5 s prediction horizons

was only found to result in a change of about 2.4% more area explored. This trend was similar for the number of OOIs identified.

However, the number of collisions does depend on the length of prediction horizon. The number of collisions is higher with the shortest prediction horizon of 0.5 s in comparison to the 1 and 1.5 s horizons. However, the difference in the number of collisions between the 1 and 1.5 s prediction horizons is very small. The length of the prediction horizon beyond which safety is not impacted likely depends on the dynamics of the robot. That is, a vehicle moving at higher speeds and with lower maximum deceleration capabilities would require a longer prediction horizon.

5.4.2 Study 2 - Impact of Interface and Delay

Based on the findings of study 1, we designed and conducted study 2 to investigate Interfaces C-E. From user study 2, three relationships will be highlighted. The first compares interfaces where the user inputs robot velocities versus controlling the position of a steerable waypoint. The second evaluates a navigation interface that has the robot move along Voronoi map paths. The third explores how the location of the shared control method (located on-board the robot versus operator control unit) impacts performance in the presence of communication delay.

Similar to user study 1, mixed-effects models were constructed for user study 2 as,

$$\text{area} \sim c_0 + c_1 * \text{Interf.} + c_2 * \text{Delay} + c_3 * \text{Interf.} \times \text{Delay} \tag{5.17}$$

where area is the portion of area covered. Interf. has values 0 representing Interface B and 1 representing the relevant Interface C, D, or E. Delay has values 0 representing 400 ms and 1 representing 800 ms.

A steerable waypoint interface similar to that used in the military [76] and with telepresence robots [105] was tested in study 2. Performance metrics of area and collisions are summarized in Figure 5.10 and Table 5.4. Performance, both in terms of

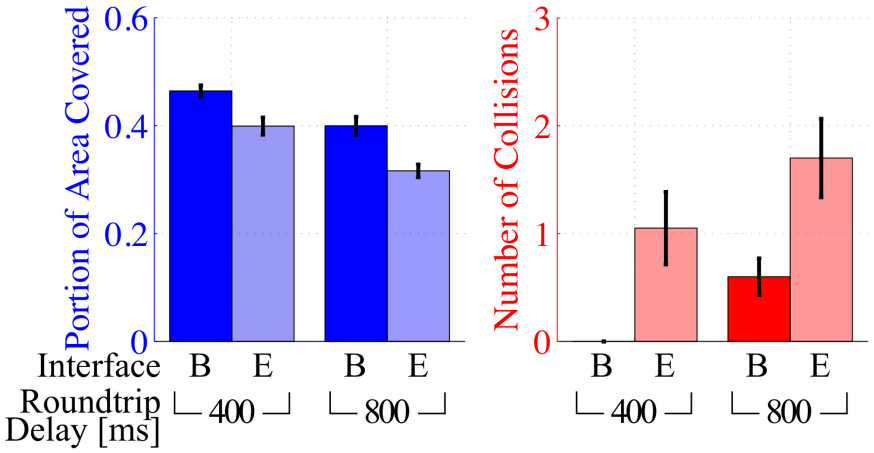

Figure 5.10: Users performed worse, in terms of area coverage and collisions, at each delay when using the steerable waypoint (Interface E) compared to robot velocity input (Interface B) in Study 2.

Table 5.4: Mixed-Effects Model: Portion of Area Covered (B|E). Higher delay and using the steerable waypoint (Interface E) both resulted in 6.5% less area covered.

Effect	Coeff	95% CI	t-value	p-value
(Intercept)	*0.465*	*(0.437, 0.492)*	*33.46*	*<0.001*
Delay	-0.065	(-0.089, -0.040)	-5.06	<0.001
Interf. B|E	-0.065	(-0.090, -0.041)	-5.12	<0.001
Interf. B|E × Delay	-0.018	(-0.0536, 0.017)	-1.02	0.312

area and collisions, was worse with the steerable waypoint (Interface E) in comparison to subjects controlling the robot's velocities (Interface B). From the mixed-effects model, one can see that the size of the effects for interface and delay are approximately the same size - 6.5% less area is covered due to the higher delay or due to using the steerable waypoint. The interface × delay coefficient has a high p-value indicating that there was not a significant interaction between communication delay and interface. Decrease in area covered is consistent with the increased task completion times observed in [76] with the steerable waypoint.

In study 1, it was observed that subjects often tried to maximize their distance from the closest obstacles. When they passed in between obstacles, they often moved along a path that was approximately halfway between the two obstacles. At high

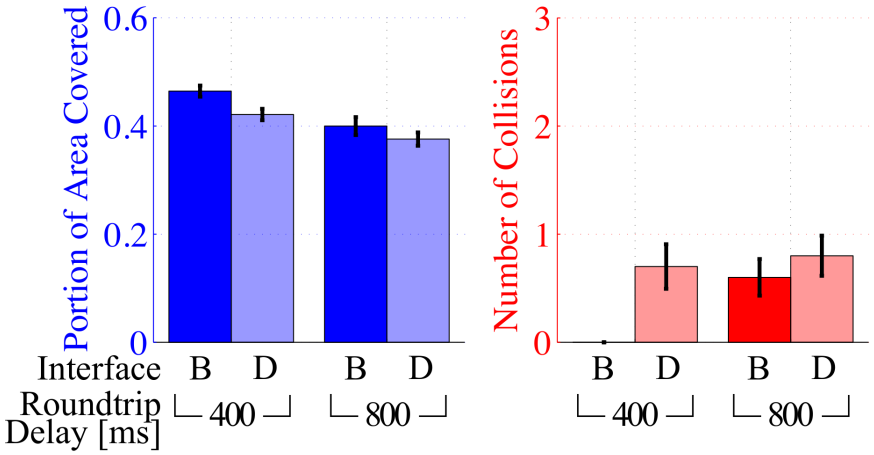

Figure 5.11: Users performed worse, in terms of area coverage and collisions, at each delay when using the Voronoi map based shared control (Interface D) compared to robot velocity input (Interface B) in Study 2.

Table 5.5: Mixed-Effects Model: Portion of Area Covered (B|D). The size of the effect for interface was not as large as that for delay (6.5% vs. 4.3%).

Effect	Coeff	95% CI	t-value	p-value
(Intercept)	*0.465*	*(0.440, 0.489)*	*36.56*	*<0.001*
Delay	-0.065	(-0.093, -0.036)	-4.43	<0.001
Interf. B|D	-0.043	-0.072, -0.015)	-2.97	0.004
Interf. B|D × Delay	-0.019	(-0.021, 0.059)	0.931	0.356

communication delays, increasing the distance from obstacles would allow subjects a larger margin for error in navigating without having collisions. Thus, we anticipated subjects would prefer and perform better moving along a Voronoi map. In Interface D, the robot tried to navigate along a Voronoi map around obstacles. Results from study 2 comparing Interfaces B and D are included in Figure 5.11 and Table 5.5.

Performance, both in terms of area and collisions, was worse with the Voronoi map based shared control (Interface D) in comparison to subjects controlling the robot's velocities (Interface B). From the mixed-effects model, one can see that the size of the effect for interface was not as large as that for delay (6.5% vs. 4.3%). The interface × delay coefficient has a high p-value indicating that there was not a significant interaction between between communication delay and interface. Additional

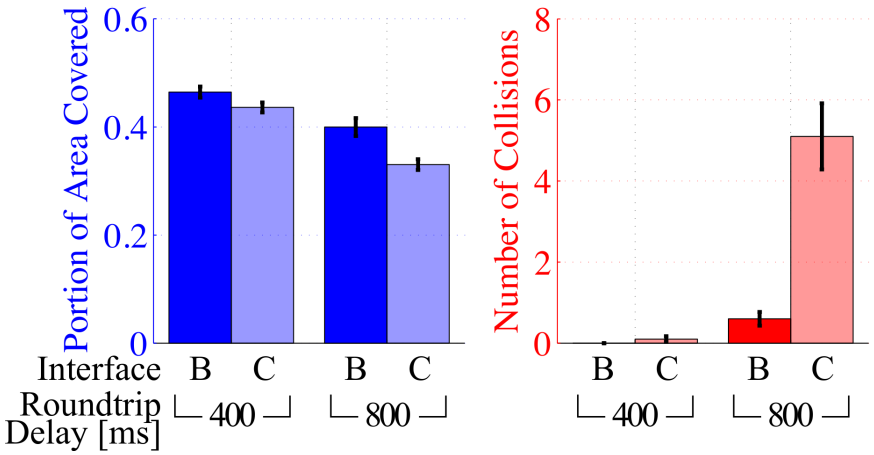

Figure 5.12: Users performed worse, in terms of area coverage and collisions, at each delay when using Interface C compared to Interface B in Study 2.

Table 5.6: Mixed-Effects Model: Portion of Area Covered (B|C). The size of the effect for interface was not as large as that for delay (6.5% vs. 2.8%). The size of the effect for Interface C (2.8%) was smaller than the effect size for Interface D (4.3%).

Effect	Coeff	95% CI	t-value	p-value
(Intercept)	*0.465*	*(0.441, 0.488)*	*39.02*	*<0.001*
Delay	-0.065	(-0.088, -0.041)	-5.27	<0.001
Interf. B|C × Delay	-0.041	(-0.075, -0.007)	-2.37	0.021
Interf. B|C	-0.028	(-0.052, -0.005)	-2.32	0.024

discussion about the Voronoi map navigation method and possible improvements are in Section 5.5.

In Interface C, the shared control method calculated robot inputs at the operator control station and thus could receive inputs from the human operator without communication delay. However, as described in Section 5.3.2, the robot state information and control commands sent to the robot were delayed. The resulting interface was more responsive to inputs from subjects (*i.e.* the robot's projected path on the screen would update instantly), however actual movement of the robot still felt delayed. Results comparing Interfaces B and C are in Figure 5.12 and Table 5.6.

Performance, both in terms of area and collisions, was worse with Interface C than Interface B. From the mixed-effects model, one can see that the size of the

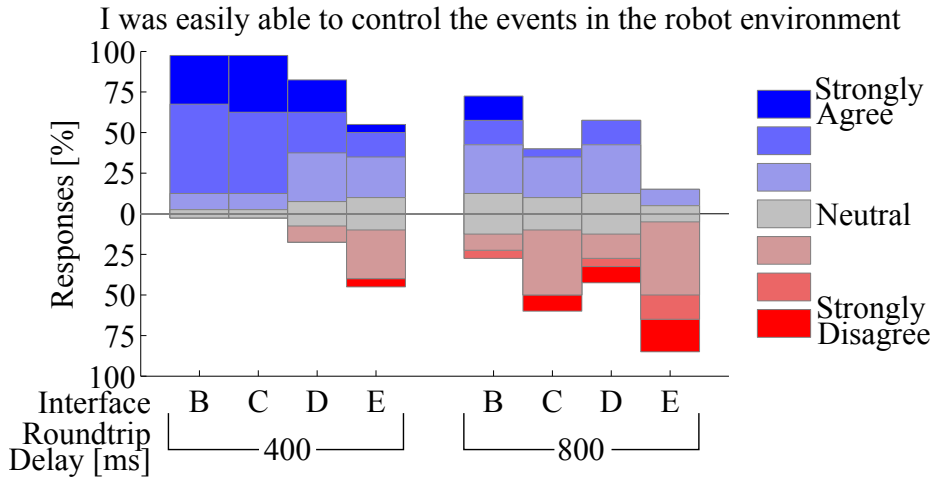

Figure 5.13: Users felt they were better able to control events in the robot environment best with autonomy located on the operator side (Interface C) at low time delay. However, at higher delay, they felt they could better control events with autonomy located on-board the robot and Voronoi map based control (Interfaces B and D, respectively) over autonomy on the operator side (Interface C).

effect for interface was not as large as that for delay and was also smaller than the effect size comparing Interface B to D. The interface $\times$ delay coefficient has a high p-value indicating that there was not a significant interaction between between communication delay and interface. Additional discussion is in Section 5.5.

Subjective survey results were collected after each test condition. The subjective measures conveyed the same messages, so results for only one of the questions are included in Figure 5.13. Subjects responded to the question (shown in the figure title) on a 7-point Likert scale with the ratings described in the legend. Some interesting results can be observed from Figure 5.13. First, at low delay the subjective ratings were spread evenly around neutral for Interface E. At the high delay, not a single participant gave a rating of 6 or 7 for Interface E. This disapproval agrees with the objective results that show low area coverage and a higher number of collisions.

Second, the subjective ratings comparing Interfaces B and D line up well with the objective measures discussed of area coverage and collisions. One can see that number of individuals giving ratings 2-6 on the 7-point scale are nearly identical

at the 800 ms delay. The difference in the subjective ratings appears to be at the extremes - more people felt strongly that they could control events with Interface B, while a similar number of people felt strongly that they could not easily control events with Interface D. Third, despite better area coverage with Interface B than C at the 400 ms delay, study participants did not rate Interface B as allowing them to more easily control events in the robot environment. More participants rated "Strongly Agree" for Interface C than B.

5.5 Discussion

Results from the two human subject studies demonstrated the effectiveness of the shared control method we developed and identified several key teleoperation relationships. In the discussion that follows, we provide some additional comments/ recommendations to consider when developing teleoperation systems.

Interaction of Shared Control and Delay

As discussed in Section 2.2, Luck *et al.* conducted a study investigating the role of different automation levels and communication delay on performance [67]. In our study, Interface A is most comparable to the teleoperation mode and Interface B is most comparable to the guarded teleoperation mode in [67]. Results from their study show that as delay increases, so do the number of drive errors, and drive errors are lower with guarded teleoperation than regular teleoperation. These results agree with our measurements for number of collisions.

With regards to time required to complete the course, results from [67] show that as delay increases, so does course completion time and guarded teleoperation is about the same as regular teleoperation. However, we found that the portion of area covered (which compares best to completion time) improves when adding shared control. In fact, we found that at higher delay, the improvement in area coverage is even more pronounced with shared control than at lower delay.

Overall, at low and high time delays our study agrees that adding shared control will decrease drive errors/collisions. Unlike [67], we also demonstrated that adding shared control provides a larger increase in performance at high delays compared to low delays. We believe the improvement in performance with shared control found in our study is due to the improved formulation of the shared control method.

Impact of Prediction Horizon

The prediction horizon used in the shared control method consists of two parts - the timestep size T_s and number of timesteps p. A smaller timestep size can result in more accurate estimates of robot state over the prediction horizon. However, decreasing the timestep results in a shorter overall prediction time, if the number of timesteps is kept constant. In our analysis, the size of the timestep was fixed to $T_s = 0.1\ s$. This value resulted in small errors between the continuous time and discretized models when simulations were run for prediction horizons ranging from 0.5 to 1.5 s. The number of timesteps was varied among trials in user study 1.

The results in Figure 5.9 and Table 5.3 suggest that prediction horizon had a small, positive effect on the portion of area covered. With regards to collisions, increasing the prediction horizon from 0.5 to 1 s resulted in fewer collisions. However, increasing the prediction horizon from 1 to 1.5 s had little impact on the number of collisions. Once the prediction horizon is long enough (in our case 1 s), then increasing the prediction horizon has little impact on safety. When selecting length of the prediction horizon for a robotic platform, one should consider the dynamics of the platform. For example, a robot that drives at higher speeds and decelerates slower would require a longer prediction horizon. Additional work is needed to explore how to find the prediction horizon that best balances robot safety and computational requirements.

Control Using Steerable Waypoint

Results in Figure 5.10 and Table 5.4 indicate that the steerable waypoint method performed worse in terms of performance and safety compared to the velocity-input

based shared control method in Interface B. Subjective results in Figure 5.13 also convey participants' opinions that control was difficult with Interface E. Many users commented that controlling the steerable waypoint with the gamepad joystick was difficult with the communication delay. The location of the steerable waypoint on the human operator's screen was responsive; however, the robot's projected path and movement was still delayed. Perhaps it would have been more intuititive to control the steerable waypoint by using a mouse cursor or using a touch interface (like a tablet computer). We wanted to minimize the number of factors that changed in each comparison (*i.e.* if performance was better with the steerable waypoint on a tablet in comparison to Interface B, it would have been difficult to determine whether that difference was due to using the steerable waypoint or using the tablet itself). Overall, we do not recommend using a steerable waypoint controlled by a gamepad, joystick or computer keyboard.

Control Moving Along Voronoi Map Paths

Voronoi diagrams are very popular in a variety of domains and many efficient methods for constructing them have been developed [7]. Voronoi maps have been used in motion planning for mobile robots to construct paths that reach an end goal while avoiding collisions with obstacles [104]. However, they often have many sharp turns that could create a jagged, unnatural feeling motion for a human operator observing a robot moving along the map. To address this concern, one could move along smoother Bezier curves generated from a Voronoi map [44]. To the best of our knowledge, no prior work has investigated shared control with Voronoi map based navigation.

The results with Voronoi map based shared control (Interface D) in user study 2 show promise, but performance with the robot velocity based input method (Interface B) is better. Subjects did make anecdotal comments that they liked the Voronoi map lines. They said that it was nice to see the possible paths that the robot could follow.

However, sometimes the exact direction a subject wanted the robot to move in was not contained on the Voronoi map, or the path to go in that direction was extra long. Subjects also commented that in areas that seemed to be a little bit more densely populated with obstacles, they preferred following the Voronoi map paths.

A number of adjustments could help improve the method. First, using an alternative input device could improve performance. For example, using a mouse or tablet interface to select Voronoi map lines to move along may be a better user interface design and could potentially allow subjects to control multiple robots at once. Second, movement along Voronoi map lines is likely more helpful in densely populated obstacle areas and/or when communication delay is high. Thus, in areas that have few obstacles and low delay, it may be better to have the robot move along a different path, *e.g.* a minimum time path to another node.

Third, the sharp corners of the Voronoi map around nodes may make the robot movement feel unnatural to human operators. Smoothing the path (especially around nodes) could make the movement feel more natural. Lastly, a better explanation of the path map that the robot is trying to follow could help, given that many operators may not understand Voronoi maps.

Shared Control on Operator Station

We had anticipated that placing the shared control calculations at the operator station would result in an interface with a more responsive feel. However, errors accumulated in the state predictors and control commands were large enough to offset this more responsive feel and cause decreased performance. Study participants' dissatisfaction with the decreased performance (area coverage) and safety (number of collisions) was also evident in their low subjective ratings of the system in Figure 5.13. These results did not agree with our hypothesis that the MPC and model based state predictors could better compensate for the delay than the human operator. In general, we recommend placing autonomous controllers on-board the robot itself, as the user

study 2 results suggest that human operators can better compensate for the delay in their inputs, than the robot can for delay in its inputs. Further exploration could be done to look at different predictors.

5.6 Conclusions

In this chapter, a shared control method to aid human operators in robot navigation tasks and avoiding collisions with obstacles was presented. The shared control method builds off previous model predictive control (MPC) formulations and a new method of representing obstacle free regions in the MPC problem is presented. The obstacle free region representation makes the method well suited for maneuvering in less structured environments (*i.e.* environments without roads or paths to follow) and allows the MPC problem to be solved very rapidly due to its convex form.

The shared control method was implemented in a realistic robot simulation engine and evaluated with two human subject studies. Results from the user studies showed that performance and safety had small improvements at low communication delay and much larger improvements at high delay with the shared control method. In addition, the user studies explored the impact of control interface and its interaction with communication delay. Delay had a larger impact on performance than interface and prediction horizon. Overall, if robot designers know a teleoperation system requires a human in the control loop and is going to experience delays on the order of hundreds of ms to 1 s, then it is recommended that they implement obstacle avoidance onboard the robot to keep system performance and safety near the performance with no delay.

CHAPTER VI

Environment Difficulty and Its Interaction with Communication Delay

6.1 Introduction

Different environment setups combined with different operating conditions will impact mobile robot teleoperation performance. Some prior work, discussed in Section 2.2, has investigated factors of environment difficulty, communication delay, and automation. However, these effects have often been considered independently, neglecting important interactions between them. In this chapter, a user study and analysis is presented that investigates how teleoperated driving performance depends on environment difficulty, level of robot automation, and communication delay in the system.

The work in this chapter is based on publications [97, 102]. The remainder of the chapter is organized as follows. Section 6.2 describes the user study design used to test factors impacting teleoperation driving performance. Section 6.3 presents the results and analysis of data collected in the user study. Section 6.4 discusses lessons learned that can be applied to the design and selection of future robot teleoperation systems.

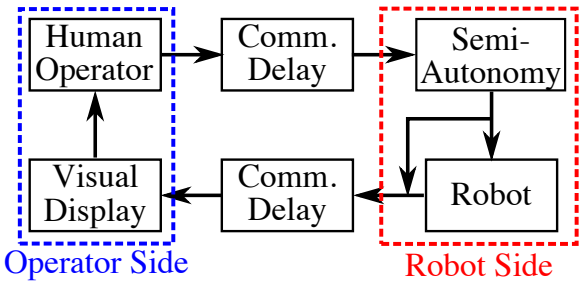

Figure 6.1: Functional diagram of information exchanged among components of the simulated teleoperation system and human subjects.

6.2 Methods

To explore how teleoperated robot driving can be described in different environment arrangements and operating conditions (*e.g.* with semi-autonomous control or communication delay), a user study was designed and carried out. In the study, subjects were first trained on how to operate a virtual robot, then they performed a series of driving tasks towards a goal location under different conditions.

6.2.1 Teleoperation Setup

In the user study, subjects teleoperated a virtual robot with the general configuration in Figure 6.1. This setup is representative of controlling a robot that is in a location different from the human operator. The subsections that follow will describe each component of the configuration.

6.2.1.1 Robot Environment

The robot and its environment were simulated using the Autonomous Navigation Virtual Environment Laboratory (ANVEL) [31]. ANVEL provides high fidelity nonlinear robot dynamics, sensor models, and realistic graphics all running in real time on a desktop computer. The robot that subjects controlled was modeled after a physical robot in our lab - one that is easily transportable and can be used indoors or outdoors for inspection, retrieving objects, etc.

Figure 6.2: Exocentric view of the simulated robot that was teleoperated in the human subject studies. Same as Figure 5.3.

To give a sense of the robot's size and speed, it is approximately 10 kg and has dimensions of 0.5 m long by 0.58 m wide. The maximum linear speed is 2 m/s and maximum turning speed is 1.15 rad/s. All of these values were selected to match that of the physical robot in our lab and are in]line with those of typical unmanned ground vehicles [1]. Figure 6.2 shows an overhead view of the robot in its environment.

6.2.1.2 Human Operator

The human operator controlled the robot using an XBox gamepad controller - a device commonly used for robot driving. The right trigger position mapped linearly to a desired forward speed, while the left joystick position mapped linearly to a desired angular velocity. More detail about subjects operating the system will be discussed in Section 6.3.

6.2.1.3 Semi-Autonomy

The semi-autonomous control method was located on-board the robot and assisted subjects to avoid obstacles. The control method is described in detail in Chap-

ter V, so only a brief description of the method is provided in this chapter. The semi-autonomous method uses model predictive control (MPC) to try and match the human driver's inputs as closely as possible while avoiding obstacles and obeying dynamic constraints of the robot.

To make the MPC problem easy to solve in real-time (approximately 10 times per second), some approximations were made to formulate a quadratic cost and convex linear constraints. Robot state estimates used in the MPC problem do not experience added delay since the semi-autonomy is located on the robot, however the input commands from the human driver do experience communication delay. The robot commands generated from the MPC problem give an optimal projected path for the robot based on safety and trying to follow the human inputs.

6.2.1.4 Communication Delay

As discussed earlier, communication delay is often introduced when the human operator and robot are located in different environments. With the study described in this chapter, simulated communication delay was introduced both in the commands sent from the human operator to the robot (forward speed and angular velocity commands) and from the robot to the human operator (camera images and projected path data). This work considers constant communication delays in each direction. The specific values tested are discussed in Section 6.2.2 and Table 6.1.

6.2.1.5 Visual Display

Subjects received information from the robot primarily through a first-person view from a camera mounted on the robot's manipulator arm as shown in Figure 6.3. The video was displayed at 25 fps with a resolution of 640×480 pixels. Above the camera view was a Trial number indicator, so subjects could track how many trials they had completed.

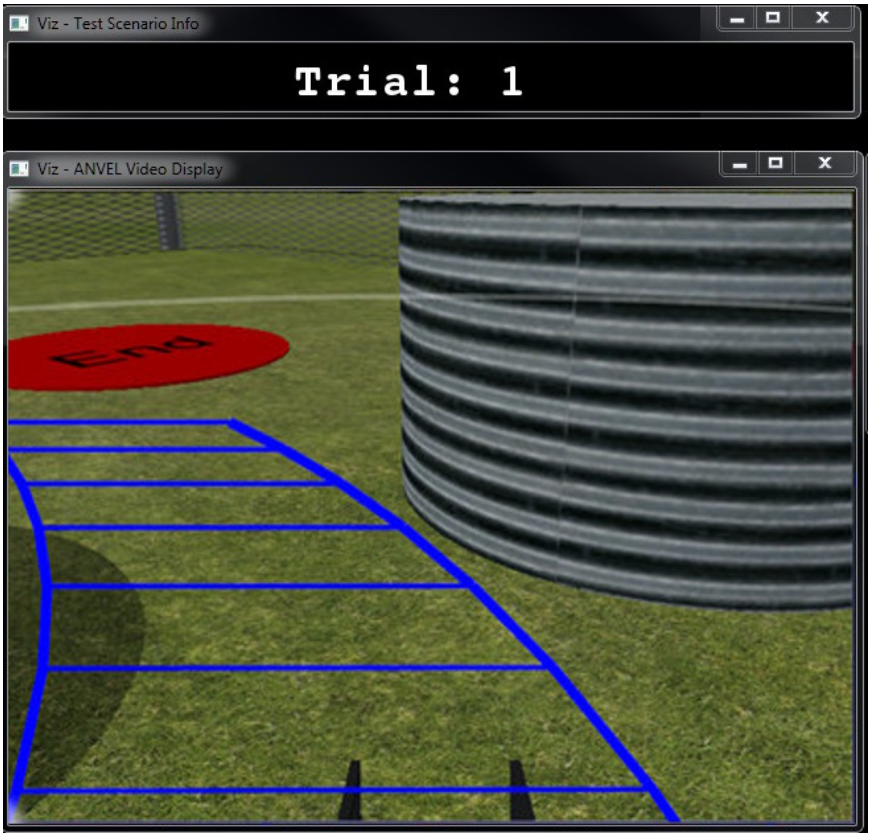

Figure 6.3: Visual display of the robot shown to test subjects.

Notice in Figure 6.3 that there are blue lines projecting in front of the robot. These lines indicate to the user the projected path of the outside of the robot's left and right side wheels for a period of 1 second into the future. This predicted path was calculated in one of two ways depending on the operating mode. 1) In manual control mode, the path is generated by assuming the subject's current inputs are kept constant for a period of 1 second and applying them to the dynamic unicycle model of the robot shown in Eqn. (6.1) [15]. 2) In semi-autonomous control mode, the path is generated by the semi-autonomous controller (described in Section 6.2.1.3).

$$\begin{cases} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \omega \\ \dot{v} &= \frac{K_1}{m}(v_{\text{des}} - v) \\ \dot{\omega} &= \frac{K_2}{J}(\omega_{\text{des}} - \omega) \end{cases} \tag{6.1}$$

In Eqn. (6.1), states x and y are the robot's longitudinal and lateral positions, θ is the robot's heading angle, v is forward speed, and ω is angular turn rate. Inputs v_{des} and ω_{des} are the desired desired forward speed and angular turn rate, respectively, from the joystick. Variables m, J, K_1, and K_2 are constants set to reflect the dynamic behavior of the robot as described in Section 5.2.3.

6.2.2 Experiment Design

In order to test how human teleoperators drive through different environments, two arrangements of obstacles were designed. The first environment had users drive in between a single obstacle gap to reach the goal position (Figure 6.4). The second required users to pass through two obstacle gaps to reach the goal position (Figure 6.5). The obstacle positions in the single obstacle gap environment were adjusted to test different gap lateral offsets relative to the robot's initial position and different gap widths. In the two obstacle gap environment, the obstacle positions were only adjusted to change the gap widths.

Each environment was tested in manual and semi-autonomous control modes. In addition, each environment $\times$ control mode combination was tested with no communication delay added and 400 ms round trip communication delay. The 400 ms of delay included 100 ms in the direction from human to robot and 300 ms from robot to human (longer delays are associated with video transmission). These values are typical delays of what we could achieve over a WiFi network in our lab and are in

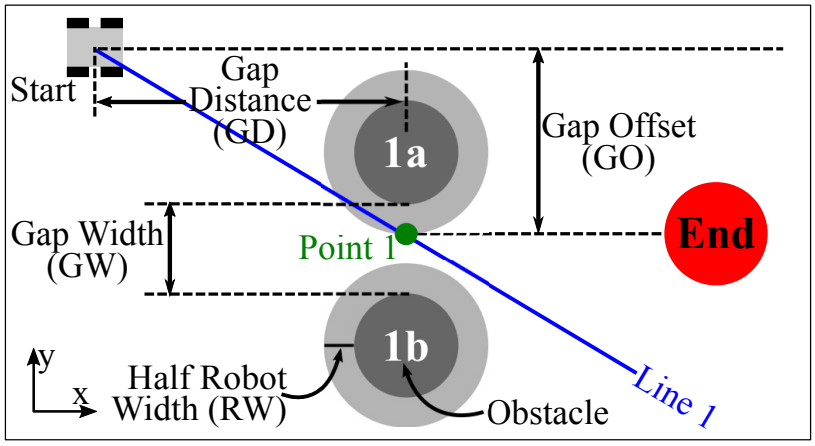

Figure 6.4: Overhead view of the single obstacle gap environment.

line with those reported in prior work investigating delay associated with video chat [114]. Thus, there were a total of 8 test blocks: 2 environments × 2 control modes × 2 communication delays.

Subjects always performed the four single gap test blocks first, followed by the four double gap test blocks. In each environment, the first two blocks either both had communication delay or did not have added delay. The order of manual and semi-autonomous control modes within each block was alternated among users to help reduce ordering effects.

Within each test block for each user, a balanced Latin Square [95] was used to order the test conditions. For the single gap test blocks there were 6 test conditions. The double gap test blocks each had 4 test conditions. Thus, each user performed 36 scored trials for each of the single gap test blocks and 16 scored trials for each of the double gap test blocks. In total, each user completed 208 score trials (36 trials × 4 single gap blocks + 16 trials × 4 double gap blocks). Values for each environment configuration are listed in Table 6.1.

The single gap environment was laid out as shown in Figure 6.4. All obstacles and the goal position had a radius of 0.5 m and all coordinates refer to the location of the center of each object unless otherwise noted. In all environments the robot started at

an initial position of (0,0) m. A few more details about the single gap environment setup:

- Obstacles 1a and 1b had an x-coordinate of 5 robot widths = 2.9 m.
- Obstacle 1a's y-coordinate was determined by placing its edge plus half the robot's width at the Gap Offset distance. The Gap Width is the smallest distance between the edges of Obstacles 1a and 1b.
  - For example, the y-coordinate of Obstacle 1a for a Gap Offset of 3 was -0.95 m and the y-coordinate of Obstacle 1b with a Gap Offset of 3 and Gap Width of 1.75 was -2.97 m.
- The y-coordinate of the goal position was the same as the Gap Offset. The x-coordinate was selected such that the minimum time path for the robot was fixed at 2.82 s. This setup corresponds to x-coordinates of 5.31, 5.49, 5.77, and 5.8 m for Gap Offsets of 3, 2.57, 1.64, and 1.5 robot widths, respectively.
- Line 1 is drawn from the robot's start position to Point 1, which is between Obstacles 1a and 1b and is one-half robot width from Obstacle 1a.

The Gap Offsets of 1.5 and 3 robot widths form angles of approximately 15 and 30 degrees between the robot's initial heading and Line 1. The rationale for testing Gap Offsets of 1.64 and 2.57 robot widths will be better explained when discussing the definition of environment difficulty index in Section 6.3.1. The Gap Widths selected, as a multiple of robot width, are in the range of those used by previous studies [42, 54].

The double gap environment was laid out as shown in Figure 6.5. A few more details about the double gap environment setup:

- Line 1 is drawn from the robot's start position to Point 1, which is between Obstacles 2b and 2c and is one-half robot width from Obstacle 2b. Line 2 is perpendicular to Line 1 and passes through Obstacle 2f's center.

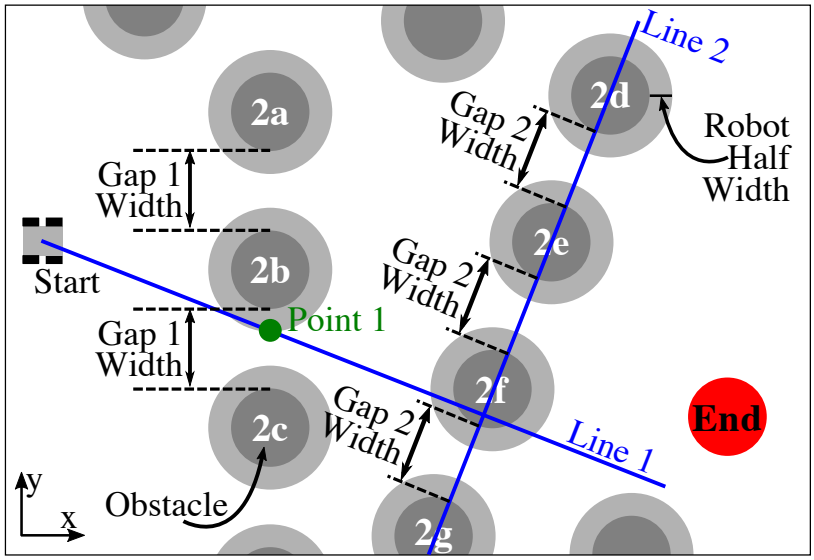

Figure 6.5: Overhead view of the double obstacle gap environment.

- Obstacle 2b was fixed at a position of (2.9, -0.37) m, which was the same as Obstacle 1a in the single gap environment with a Gap Offset of 2 robot widths.
- Obstacles 2a and 2c were placed to have an x-coordinate of 2.9 m and the y-coordinate was determined from Gap Width 1.
- Obstacle 2f was fixed at a position of (5.73, -1.89) m. This position corresponds to the setup in the single gap environment with Gap Offset 2 robot widths if the robot were oriented in the direction of Line 1 and located at the end point closest to Obstacle 2b.
- Obstacles 2d, 2e, and 2g were placed to lie along Line 2 and create the specified Gap Width 2.
- The goal position was fixed at (8.72, -2.24) m.

The double gap environment offers several options of which gaps to pass between to reach the end goal. It was designed to see if results developed with the single gap environment could be extended to more complex environments.

Table 6.1: Test conditions for each test subject.

Single Obstacle Factors	Levels
Control Mode	manual, semi-auto
Comm. Delay [ms]	0, 400
{Gap Offset, Gap Width} [robot widths]	{1.5, 1.75}, {1.5, 1.375}, {1.64, 1.5}, {2.57, 1.5}, {3, 1.75}, {3, 1.375}
Double Obstacle Factors	**Levels**
Control Mode	manual, semi-auto
Comm. Delay [ms]	0, 400
Gap {1, 2} Widths [robot widths]	{1.375, 1.375}, {1.75, 1.375}, {1.75, 1.75}, {1.375, 1.75}

6.2.3 Test Procedure

Test subjects were recruited through email lists and announcements at the University of Michigan. They were paid $10 for participating in the study and were incentivized to try their best through a bonus compensation. The top performer (subject with the minimum score) in each test block was paid a $10 bonus. Scores were explained to subjects and calculated as follows,

$$\text{score} = \sum_{\text{test block}} \text{trial time} \cdot (1 + \text{collisions}) \tag{6.2}$$

where 'trial time' is the time to move from the start to end position and 'collisions' is the number of collisions with obstacles.

At the start of each user test subjects filled out an informed consent form and answered some basic background questions. Before beginning the scored trials, there was a guided training session. The training was used to make sure subjects understood the visual interface, were capable of driving around obstacles, and could successfully reach the goal position. Training took 10-15 minutes and consisted of the following:

1. Subjects were verbally instructed how to drive forwards, backwards and turn in an empty environment.

2. The robot's projected path represented by the blue lines was explained to subjects as they continued to practice driving in the empty environment.

3. Subjects practiced driving to an end goal circle.

4. Subjects were placed in an environment with three obstacles placed in a line such that two gaps of 1.375 and 1.75 times the robot width were formed. Subjects practiced driving around the obstacles and through the gaps. This practice was done first with manual control mode, then with semi-autonomous control.

5. Step 4 was repeated with the added communication delay (400 ms roundtrip).

6. Immediately before each block of scored trials, subjects were given the opportunity to practice each environment configuration once.

In the scored trial blocks, subjects drove the robot towards the specified goal position. Once the robot reached the goal position, the robot was automatically stopped. To start the next scored trial, the user pressed the start button on the gamepad to reset the robot and environment.

After all trials were completed subjects were thanked for their participation and dismissed. Each user test took approximately one hour.

6.3 Results

In this section the results from the user study are presented. The study consisted of 15 male and 5 female test subjects with an average age of 22.9 years and standard deviation (sd) 2.4 years. On a scale of 1 (low) to 7 (high), subjects self-rated their average experience playing video games as 4.2 (sd=1.4) and their familiarity with robotics as 3.8 (sd=1.5). These tests were approved by the University of Michigan Health Sciences and Behavioral Sciences Institutional Review Board (UM IRB #HUM00044265).

6.3.1 Driving Time

The first performance metric that will be discussed is the time that it took for subjects to drive from the vehicle's start location, past the obstacles, and to the end target. This time is referred to as the movement time T in the figures and equations throughout this section. The movement time starts running once a subject begins driving forward, backwards, or turning and ends as soon as the center of the vehicle is over any part of the end circle.

Recall in Section 6.2.2 the test conditions for the single obstacle gap environment were selected such that the mininimum time path from the start to end location was the same for each {gap offset, gap width} combination tested. Thus, a perfect autonomous controller could drive from start to end in 2.82 s in each of the 6 {gap offset, gap width} combinations tested. Figure 6.6 displays boxplots of the movement times under the manual control condition with no delay added. A note on outliers: any point more than 1.5 IQR (interquartile range) above the 75th percentile is indicated as a dot above the top whisker. One can see from Figure 6.6 that at each test condition, the lower whisker of each boxplot is very close (within a couple hundredths of a second) to the minimum time of 2.82 s. This observation indicates that in manual control mode, some subjects were able to complete the task in close to the minimum time.

However, one can see that at the different test conditions in Figure 6.6, the spread of the movement times varies between different test conditions. That is, the IQRs and whiskers are not the same across test conditions. As the width of the gap that the robot must pass through decreases, the spread of the movement times increases. Likewise, as the offset of the gap from the robot's initial location increases, so does the spread of movement times. This observation indicates that although subjects can still get very close to the minimum time in some trials, their consistency gets worse under different environment configurations.

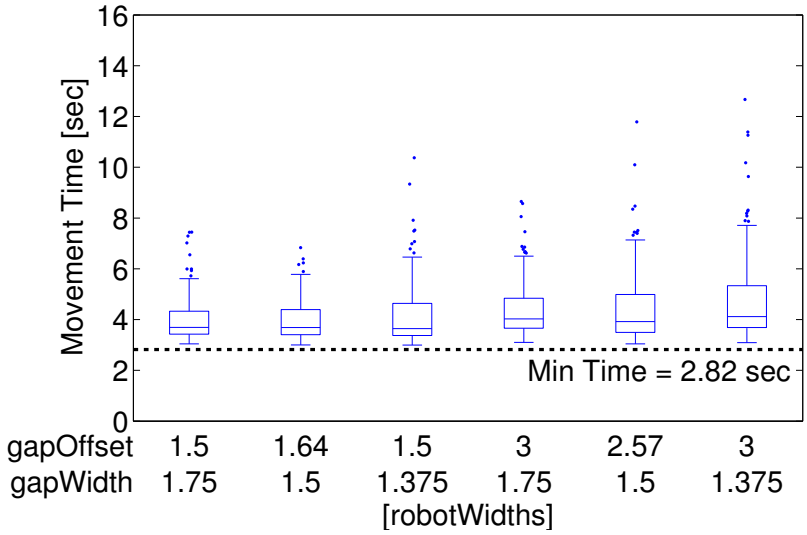

Figure 6.6: Movement times for users operating the robot with manual control mode and no added communication delay in the single obstacle gap environment.

Similarly, in the double obstacle gap environment, subjects tested 4 different combinations of gap width pairings. For all 4 combinations, the fastest time that an autonomous controller could drive from start to end was 4.46 s. Figure 6.7 shows boxplots of the movement times under the manual control condition with no delay added. One can see that the lower whiskers of the movement times for each condition are very close to the minimum time of 4.46 s. Again, in the best trials, subjects were able to come within hundredths of a second of the best possible time. However, at the smaller gap width test conditions, the size of the IQR and whiskers grows. While subjects can drive nearly as well as a perfect autonomous controller in all of the test conditions explored, the consistency of their performance is worse than autonomy. Furthermore, inconsistency in performance is made worse in environment arrangements with smaller widths between obstacles and larger offsets in the gap location relative to the robot's position.

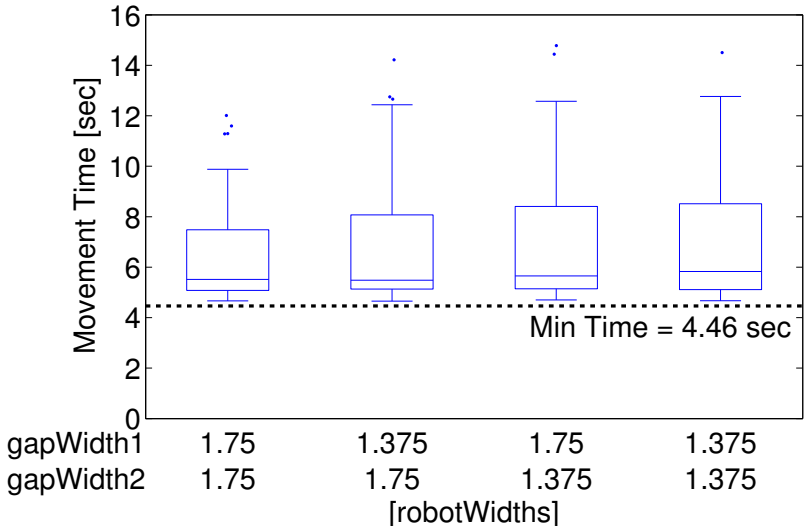

Figure 6.7: Movement times for users operating the robot with manual control mode and no added communication delay in the double obstacle gap environment.

6.3.2 Index of Difficulty

Drawing inspiration from the human movement and robot movement literature discussed in Section 2.2.3, an index of difficulty for robot driving around obstacles will be developed. Figure 6.6 illustrates that movement time does not depend on gap width or gap offset alone. There is some interaction between the two. At smaller gap offsets, movement time appears to be less sensitive to the width of the gap. To predict what the movement time for a given obstacle configuration would be, the index of difficulty definition should capture this interaction between of gap offset and width.

Vaughan *et al.* found that human movement around an obstacle could be modeled using ID $= \log_2(2[D+2O]/W)$, where D is the Euclidean distance between start and end positions, O is the size of the obstacle's intrusion into the straight line path from start to end positions, and W is the end target diameter [107]. For robot driving, Helton *et al.* found that the difficulty of a 90 degree robot turn could be modeled using ID $= \log_2 \frac{R}{(W-R)+1}$, where R is the robot width, and W is the width of the passageway around the turn [42].

The ID initially proposed for the robot driving task described in this chapter

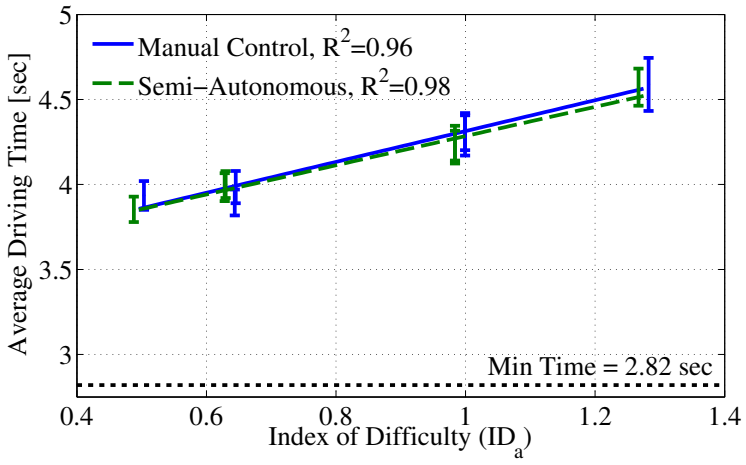

Figure 6.8: Average movement times for subjects under the no communication delay condition with manual (solid blue line) and semi-autonomous control modes (dashed green line). The index of difficulty is defined according to Eqn. (6.3). Errorbars represent ±one standard error.

draws inspiration from [107] and [42].

$$\mathrm{ID}_a = \frac{\mathrm{D}_{min}}{v_{max}} \cdot \frac{\mathrm{RW}}{\mathrm{GW}} \cdot \frac{\mathrm{GO}}{\mathrm{GD}} \tag{6.3}$$

where D_{min} is minimum time path length, v_{max} is robot max speed, RW is robot width, GW is gap width, GO is gap offset, and GD is longitudinal distance to the gap. Refer to Figure 6.4 for graphical descriptions.

Linear regressions $T = a + b \cdot \mathrm{ID}_a$ were fit to the manual control and semi-autonomous data without communication delay using ID_a and the result is shown in Figure 6.8. The R^2 values for each trendline are close to unity, indicating that the ID_a definition is a good fit for the data.

We originally defined ID_a in publication [97]. However, after looking at the ID_a definition more, we decided to simplify it in three ways:

1. We removed the RW and v_{max} terms because these terms may be misleading to individuals using the definition. That is, normalizing the ID by RW and v_{max} may lead one to think that IDs for robot platforms with different sizes and

speeds can be directly compared after dividing by their width and multiplying by their maximum speed. The ID describes the difficulty of an obstacle arrangement and should be interpreted separately for different vehicle platforms.

2. The term $\frac{\text{GO}}{\text{GD}}$ is replaced with the total change in robot heading angle required. This ratio of GO to GD gives a value that is approximately equal to the angle between the robot's initial heading and the gap, in radians. What this ratio fails to capture is how the gap between the obstacles is oriented relative to the robot, which will be captured in the modified ID definition.

3. The length D_{min} for the minimum time path length is replaced with D - the linear distance from the robot's start position to end position. Replacing D_{min} with D makes it easier to calculate the index of difficulty.

If one looks at the rearranged form of the ID_a definition,

$$\text{ID}_a = \frac{\text{GO}}{\text{GD}} \cdot \frac{\text{D}_{min}}{\text{GW}} \cdot \frac{\text{RW}}{v_{max}} \tag{6.4}$$

then making the three simplifications results in the following ID definition:

$$\text{ID} = (\alpha_1 + \alpha_2) \cdot \frac{\text{D}}{\text{GW}} \tag{6.5}$$

where D is linear distance from the robot's start to end position, GW is the width of the gap or target, and α_1 & α_2 are angles defined in Figure 6.9. The angle α_1 is between the robot's initial heading and Line 1 connecting the robot's initial location to the closest point (Point 1) between the gap that the robot is driving towards. The angle α_2 is between Line 1 and Line 3. Note that Line 2 passes through the center of the two obstacles. Line 3 is perpendicular to Line 2 and passes through Point 1.

With this definition of the environment index of difficulty, the movement time is predicted as $T = a + b \cdot \text{ID}$, where a and b are empirically estimated intercept

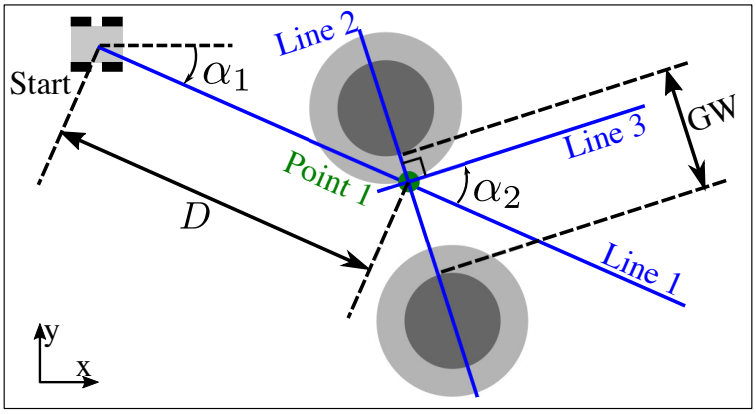

Figure 6.9: Parameters used to define difficulty index.

and slope coefficients. Figure 6.10 displays the linear fits for the movement times under the no delay condition with manual control mode (solid blue lines) and semi-autonomous control mode (dashed green lines). The R^2 values with the ID definition in Eqn. (6.5) are very close to the R^2 values with the ID_a definition in Eqn. (6.3). The simplifications made to the ID definition did not negatively impact how well the ID predicts movement time. Thus, the ID definition in Eqn. (6.5) will be considered for the remainder of this dissertation and is recommended for use in future work.

Figure 6.11 shows the linear fits for movement times under the delay condition with manual control (solid blue lines) and semi-autonomous control (dashed green lines). Given the high R^2 values, there is also a strong linear relationship between average movement times and the ID definition with communication delay in the system.

Note that for Figure 6.10 and 6.11 the {gap offset, gap width} combinations of {1.5, 1.375} and {1.64, 1.5} were selected to both have ID=1.28. Similarly, {gap offset, gap width} combinations of {3, 1.75} and {2.57, 1.5} were selected to both have ID=2.07. One can see that the different gap offset $\times$ gap width combinations lined up well under the different delay and semi-autonomous mode conditions at their respective difficulty indexes of ID=1.28 and ID=2.07.

In order to get a better estimate of the size of the effects of delay, ID, and semi-autonomous mode, a mixed-effects model was fit to the movement times for the single

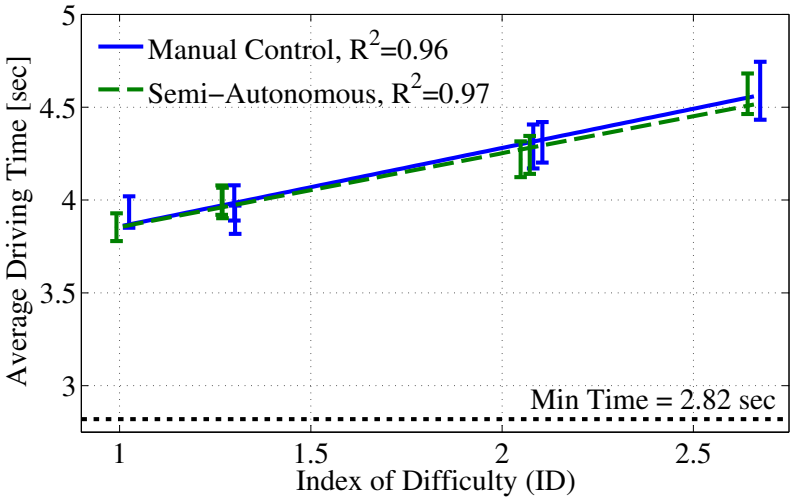

Figure 6.10: Average movement times for subjects under the no communication delay condition with manual (solid blue line) and semi-autonomous control modes (dashed green line). The index of difficulty is defined according to Eqn. (6.5). Errorbars represent ±one standard error.

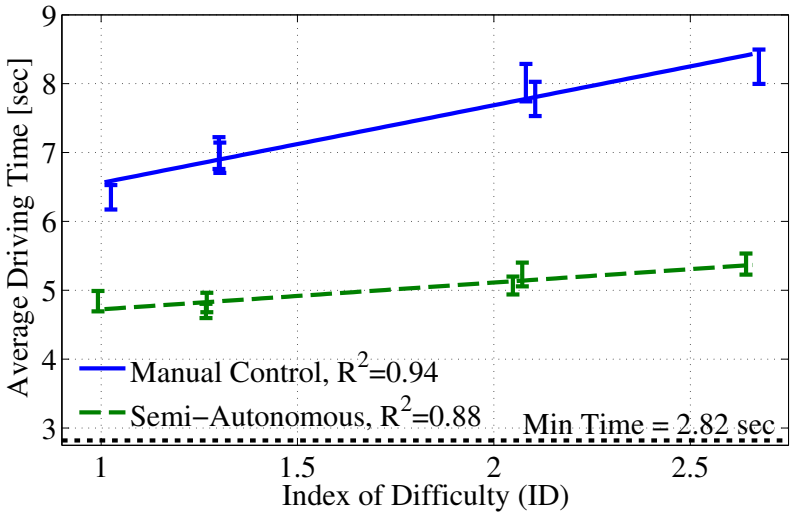

Figure 6.11: Average movement times for subjects under the 400 ms communication delay condition with manual (solid blue line) and semi-autonomous control modes (dashed green line). The index of difficulty is defined according to Eqn. (6.5). Errorbars represent ±one standard error.

obstacle gap environment. The mixed-effects model was constructed with the lme4 package in R [8]. Confidence intervals for estimated parameters were constructed from profile deviance objects [9, Sec. 1.5] and the lmerTest package in R [59] was used to determine which effects were significant. The estimates of the mixed-effects

Table 6.2: Mixed-effects model for movement time in the single obstacle gap environment.

Effect	Coeff	Coeff Value	95% CI	t-value	p-value
(Intercept)	c_0	*3.44*	*(2.89, 3.99)*	*12.34*	*<0.001*
Delay	c_2	2.00	(1.59, 2.41)	9.51	<0.001
Semi-Auto× Delay	c_4	-1.12	(-1.70, -0.54)	-3.77	<0.001
ID×Delay	c_5	0.71	(0.48, 0.93)	6.14	<0.001
ID×Delay× Semi-Auto	c_7	-0.71	(-1.03, -0.40)	-4.40	<0.001
ID	c_1	0.42	(0.26, 0.58)	5.20	<0.001
ID×Semi-Auto	c_6	-0.02	(-0.25, 0.20)	-0.21	0.83
Semi-Auto	c_3	0.02	(-0.40, 0.43)	0.10	0.92

model coefficients are listed in Table 6.2 with the general form,

$$\begin{aligned} T \ &= c_0 + c_1 * \text{ID} + c_2 * \text{Delay} + c_3 * \text{Semi-Auto} + c_4 * \text{Delay} \times \text{Semi-Auto} \\ &+ c_5 * \text{ID} \times \text{Delay} + c_6 * \text{ID} \times \text{Semi-Auto} + c_7 * \text{ID} \times \text{Delay} \times \text{Semi-Auto} \end{aligned} \tag{6.6}$$

where ID is the value defined in Eqn. (6.5), Delay has values 0 representing 0 ms and 1 representing 400 ms. Semi-Auto has values 0 representing manual control and 1 representing semi-autonomous control.

From the model coefficients in Table 6.2, one can see that communication delay (0 ms vs. 400 ms) had the largest overall effect. The next largest effects were communication delay's interaction with semi-autonomous mode and ID. The semi-autonomous mode did not have a significant main effect or interaction with ID.

Using the mixed-effects model coefficients, one can construct equations for a number of relationships between driving time and the conditions tested. For example, to construct the equation for the line $T = a + b \cdot \text{ID}$ fit to the manual control data in Figure 6.11, the intercept $a = c_0 + c_2 = 5.44$ and the slope $b = c_1 + c_5 = 1.13$. With the mixed-effects model, one can look at the impact of different conditions on driving time sensitivity. One particularly interesting trend is that the slope of the linear trendline with manual control mode changed from 0.42 s/ID to 1.13 s/ID when

comparing the no delay to delay conditions. With semi-autonomous control the slope remained at 0.40 s/ID with the no delay and delay conditions. There was almost a 170% increase in sensitivity to ID in manual control mode when delay was present in the system compared to the sensitivity remaining unchanged in semi-autonomous mode.

Despite all of these paths having the same minimum drive time, movement times with subjects were significantly different across the different gap offset and gap width arrangements. The formulation of the difficulty index in Eqn. (6.5) was found to be a good parameterization of the environment setup that can be used to predict the average movement time of the robot for paths with the same minimum drive time, but different arrangements or difficulties.

6.3.3 Driving Safety - Collisions

In addition to being told that their objective was to minimize drive time to the end goal, subjects were told to try to avoid collisions as well. A collision was counted as any time part of the robot made contact with an obstacle. Figure 6.12 and 6.13 show the average number of collisions per trial with manual control in the single obstacle gap and double obstacle gap environments, respectively. Some collisions did still occur with the semi-autonomous mode. These collisions primarily resulted from small errors between the linearized/discretized model used in the MPC problem and the nonlinear behavior of the robot plant model. In general, the number of collisions with semi-autonomous control mode was an order of magnitude lower than with manual control mode (0.035 collisions/trial with semi-autonomous control vs. 0.38 collisions/trial with manual control mode). Since the number of collisions with semi-autonomous control was so low, only collision results from manual control mode are presented.

One can see from Figure 6.12 and 6.13 that delay increased the average number

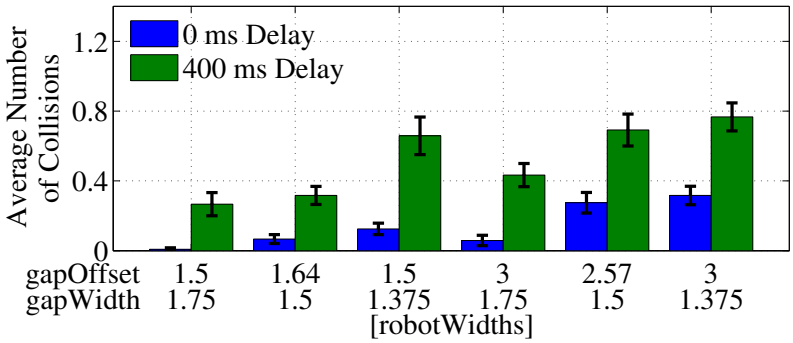

Figure 6.12: Average number of collisions for users operating the robot with manual control mode in the single obstacle gap environment.

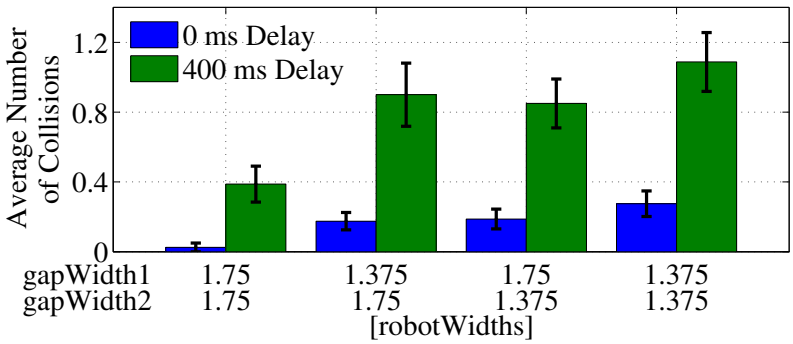

Figure 6.13: Average number of collisions for users operating the robot with manual control mode in the double obstacle gap environment.

of collisions in all environment arrangements. The driving conditions containing the smallest gap width of 1.375 robot widths resulted in the highest number of average collisions. The only exception is that the gap width of 1.5 and gap offset of 2.57 robot widths also had a high number of collisions indicating that there is some interaction between gap width and gap offset.

One can see that the ID would not produce the same type of linear relationship as it did with movement times. In particular {gap offset, gap width} conditions {1.5, 1.375} and {1.64, 1.5} both have ID=0.56, but have quite different collision numbers. The number of collisions is more sensitive to the gap width than the gap offset. Thus, if one were to define an ID to describe difficulty with regards to collisions, the definition would likely place more weight on gap width. For example, the gap width

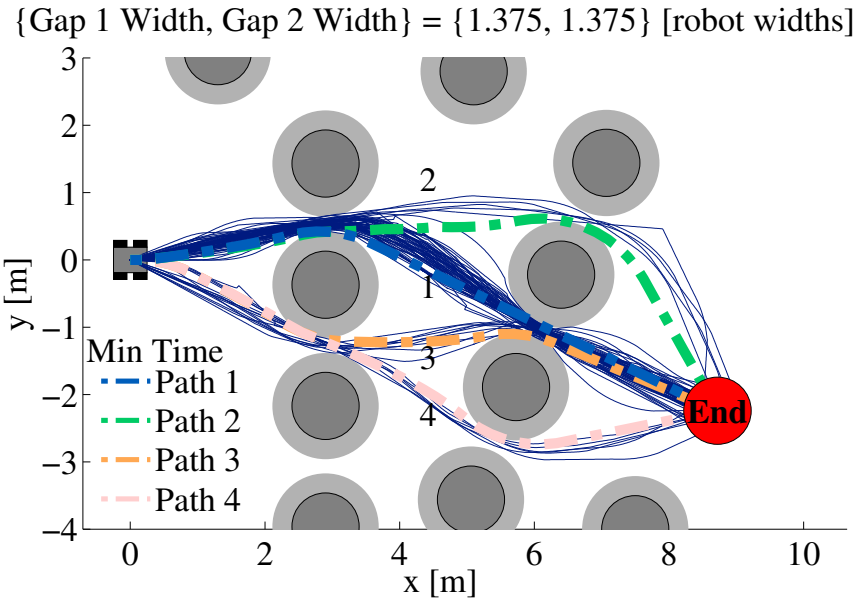

Figure 6.14: Diagram displaying paths taken by human subjects in manual control mode with no added delay for the double gap environment. Paths passing through different sets of obstacles are numbered 1-4.

in the denominator might be multiplied by a constant less than 1.

6.3.4 Driving Path Selection

In the double obstacle gap environment, test subjects were able to select which path they wanted to take to reach the end goal location. Since the arrangement of obstacles was only adjusted to change the gap widths between trials, subjects often narrowed down their path selection to one or two paths that they could drive quickly and safely. Figure 6.14 shows all of the paths taken by subjects under test conditions with manual control mode and no added time delay. Additionally, the minimum time paths are displayed for reference using dashed lines. From Figure 6.14 one can see that Path 1 was the most popular path among the paths taken. Looking at how many individuals selected to follow Path 1 to the end goal, we wondered how subjects selected that path.

To explore how subjects select a path among several options, we looked at which paths were selected during the first four practice runs and then over the course of

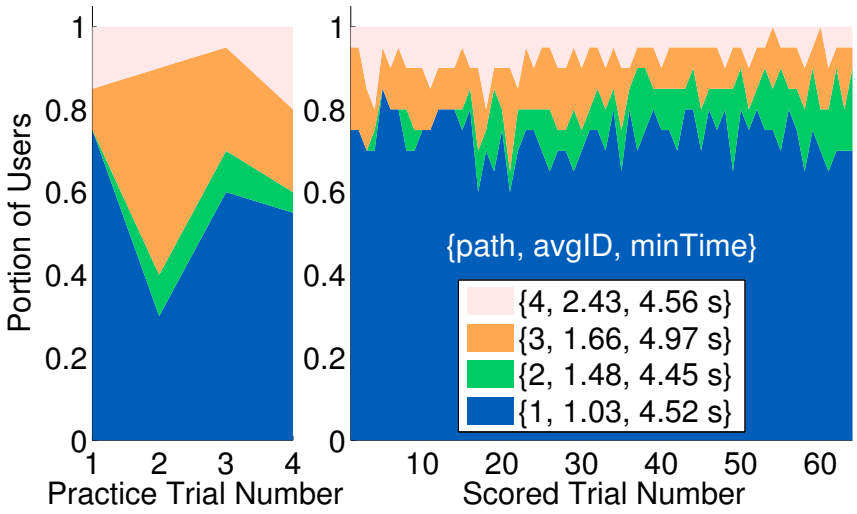

Figure 6.15: Portion of users selecting each path in practice (left) and during the scored trials (right). The legend displays the average ID for each path (averaged across the 4 gap width combinations) and the minimum time required to drive the path.

all the scored trials. Figure 6.15 displays the portion of users that took paths 1-4 at each of the first 4 practice trials and 64 scored trials. Looking at the breakdown of paths selected in the scored trials, it agrees with the paths observed in Figure 6.14: a majority of people moved along Path 1. Interestingly, Path 1 does not have the smallest minimum time; Path 3 does. This observation raises the question: why did such a large number of subjects decide to take the Path 1 route when other paths were available with similar or even smaller minimum drive times?

In the practice trials of Figure 6.15, one can see that a large number of subjects tried driving Path 3, which had the shortest minimum path time. However, many of those subjects ended up deciding to select Path 1 to reach the goal in the safest and quickest manner. The legend in Figure 6.15 shows that the average difficulty index of each path varies between 1.03 and 2.43. Path 1 has the lowest ID value and was the most traveled path. This observation suggests that the difficulty index could be used in conjunction with the minimum time path to estimate which path a human driver would take. It is likely that subjects try to select the path that appears the shortest.

Table 6.3: Number of scored trials driven along a path (columns), given the path driven in the previous trial (rows).

		Current Path Driven			
		1	2	3	4
Previous Trial Path	1	**860**	30	32	3
	2	29	**84**	6	19
	3	26	10	**59**	5
	4	9	11	7	**70**

However, if there are several paths with similar minimum drive times, then subjects are more likely to select the easier (lower ID) path to drive.

Table 6.3 shows how consistent subjects were in their path selections. Notice that the bold numbers on the diagonal are more than double nearly all of the off-diagonal numbers. This observation indicates that subjects most often stayed with the path that they had driven in the previous trial.

Furthermore, we wondered if subject's path selection depended on the test conditions (*i.e.* level of communication delay and control mode). Figure 6.16 displays the portion of trials that subjects drove each path for each test condition. The combined portions for Paths 1 and 2 are higher with the 400 ms communication delay method. Additionally, the portion of Path 2 is highest for the case with 400 ms of delay and semi-autonomous mode activated.

From Figure 6.14, one can see that Paths 1 and 2 both require the human operator to go to the left of the first obstacle. More subjects likely decided to go in this direction when the 400 ms of communication delay was present because the lower difficulty index Paths 1 and 2 were in that direction. With the semi-autonomy and communication delay, subjects sometimes could not adjust the heading of the robot quickly enough and although they intended to follow Path 1, they ended up going along Path 2 to the end goal. Overall, subjects traveled along the lower ID Paths 1 and 2 than higher ID Paths 3 and 4 under the more difficult operating condition with 400 ms of delay.

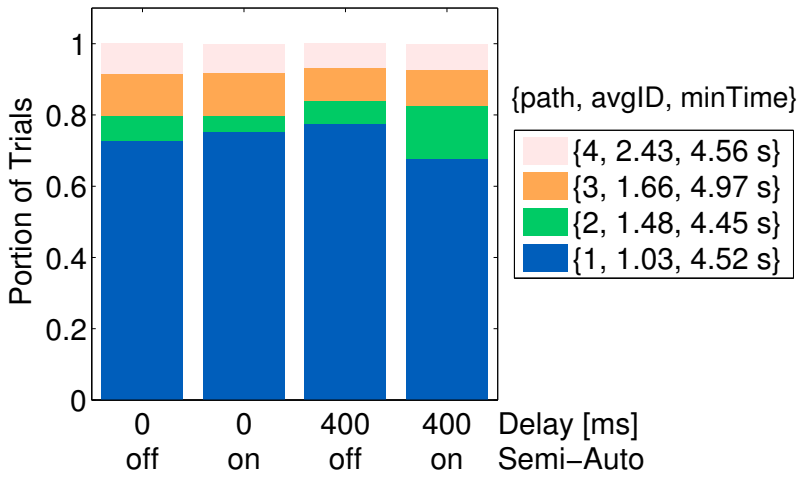

Figure 6.16: Portion of trials completed for each path at each delay and control mode test condition.

6.4 Conclusions

System performance for unmanned vehicles depends on a variety of factors including the vehicle, environment and human operator. This chapter presented results and analysis from a user study exploring the interaction of environment setup, semi-autonomous control, and communication delay in teleoperated driving performance. The key findings were the following,

- A new environment difficulty index (ID) was defined with Eqn. (6.5). The ID definition can be used to describe:

    1. the average movement time for paths with the same minimum travel time, but different obstacle configurations.

    2. the preferred path by users when choosing among paths with different minimum travel times.

- Human teleoperation drivers can perform as well as automation in environments of varying difficulty. However, variability of driving time and collisions increases as difficulty (ID) increases.

- When there is high communication delay and consistent performance is needed, some level of automation should be added to teleoperated vehicles.

These results will assist roboticists to make better design decisions for mobile robot systems and improve overall performance.

CHAPTER VII

Conclusions and Future Directions

As robots become more autonomous, human operators are alleviated of many low level tasks. However, the human's role in completing many robot missions is still critical. To this end, semi-autonomous control methods that effectively distribute control between human and automation will continue to be important. The type of automation and interface needed often depends on a number of factors including, but not limited to: the robot platform, the environment, the task itself, and the human operator. This dissertation developed methods of improving semi-autonomous control and understanding how different factors, such as communication delay and environment setup, impact mission performance. These efforts are enumerated below.

7.1 Contributions

1. **Method of Improving Handling and Preventing Rollover Using an Existing Manipulator Arm:** Manipulator arms on-board mobile robots often raise the center of gravity, making the robot more prone to rollover or tip-over. By moving the manipulator arm with the technique proposed in this dissertation, it was demonstrated both in simulation and experimentally that the robot can make smaller radius turns at higher speed without rolling over.

2. **Method of Representing Convex Obstacle Free Regions:** Obstacle free regions in a robot's environment are typically non-convex spaces, causing their mathematical representation to create difficult optimization problems in path planning. The method developed in this work provides an approximate representation of the obstacle free region that is convex, allowing it to be used in real-time optimal control problems. The method is well suited for highly maneuverable vehicles, such as skid-steer and omnidirectional robots.

3. **Relationship between Communication Delay Distributions and Teleoperation Performance:** Time-varying communication delay is typical of wireless communication networks and is known to have a negative impact on robot teleoperation in general. This dissertation proposed a method of quantitatively relating teleoperated driving performance among stochastic delay distributions with different statistical moments.

4. **Difficulty Index Definition for Driving around Obstacles:** Despite having the same minimum time path through a series of obstacles, different obstacle arrangements can be more or less difficult for users to drive through. This work defined an index of difficulty for describing different obstacle arrangements that can be used to predict the average movement time for subjects driving through an environment and determine the path that subjects are most likely to take.

5. **Relationships between Teleoperation Conditions and Performance:** Prior user studies have considered how teleoperation performance is impacted by conditions, such as automation and communication delay, independently. Results from the user studies conducted in Chapters V and VI describe the interaction of teleoperation conditions including automation, communication delay, and environment difficulty on task performance. One of the most interesting results was that the sensitivity of robot driving time to environment

difficulty did not change when communication delay was present and obstacle avoidance was used. However, without obstacle avoidance the sensitivity of robot driving time to environment difficulty increased by almost 170% when communication delay was present in the teleoperation system.

7.2 Future Work

While this dissertation has made significant contributions in the areas discussed, many questions and challenges in the field of semi-autonomous robot operation still exist. The following subsections describe several areas of future work.

7.2.1 Dynamic Weight Shifting for Stabilization

Contribution 1 demonstrated how an existing manipulator arm can help prevent rollover and improve handling. Recent work has also looked at adding tails or using robot arms to help prevent legged robots from tipping over [68, 43]. However, there is still opportunity to use robot manipulator arms or tails to stabilize the yaw motion in aggressive driving maneuvers. For example, the robot's manipulator arm could be moved along the vehicle's longitudinal axis to correct for oversteer or understeer. Or if the vehicle starts experiencing a large yaw rate (indicating that it is close to spinning out), then the manipulator arm could be moved to provide a reaction moment to counter the large yaw rate.

The technique presented in Chapter III was a purely reactionary method. However, using predictive methods to estimate the vehicle's roll and lateral dynamics for a short period of time into the future could yield even better improvements in rollover and handling stability. Another improvement could be to integrate the dynamic weight shifting method into a path planner that could plan paths that leveraged the weight shifting behavior. Finally, an extension to consider tripped rollover events would make the method more robust in real-world operation.

7.2.2 Time-Varying Delay Relationships with Driver Model

The relationship for relating driving performance among different stochastic time-varying delay distributions in Chapter IV was empirically derived from user study data. The relationship was also supported by simulations with the proportional derivative (PD) controller based driver model in Chapter IV. Future work could conduct analysis to develop an analytical relationship between path following performance with the PD driver model and stochastic communication delay distributions.

The user study and analysis in Chapter IV considered steering a skid-steer robot using a gamepad. Future work could investigate other important features of teleoperated driving, such as different road curvatures, underlying vehicle dynamics, and control input devices (*e.g.* steering wheel), from an analytical and experimental standpoint. Results could likely extend to the broader field of time delay systems.

7.2.3 Defining Task Difficulty for General Environments

Chapter VI and Section 2.2.3 developed definitions to describe task difficulty. However, the definitions apply to a relatively limited set of scenarios, such as path following, driving around corners, or driving around static obstacles. Future work could add considerations for describing the impact of type of automation, environment conditions (*e.g.* lighting, weather), user interface (*e.g.* haptic steering wheel, tablet interface), and moving obstacles on task difficulty. The impact of many of these factors on robot operation performance is not yet well understood.

The sensors and hardware on the robot also have a large impact on how difficult a task is to complete and warrant consideration. For example, features of the camera view including field of view, resolution, and viewpoint may impact task difficulty. These considerations would facilitate analysis of task difficulty for a more complete set of scenarios. The analysis could be used in the design of robot systems or when selecting robots for tasks.

APPENDIX

APPENDIX A

Rollover Model Parameter Description

A brief description of parameters and values used in the Linear and Nonlinear Model are listed in Table A.1. There are differences in tire stiffness k_t and damping b_t values for the Linear and Nonlinear Model because the Nonlinear Model contains roll stiffness and damping both in the tires and about its roll center axis. The Linear Model contains all of its roll stiffness and damping in the tires. The Nonlinear Model tires are stiff compared to the roll stiffness. Therefore, k_{roll} and b_{roll} decrease the overall effective stiffness and damping. Smaller values of k_t and b_t were selected for the Linear Model in order to match the steady state response of the Nonlinear Model in Figure 3.6 for $u = 4\ m/s$.

The Nonlinear Model inertia parameters of the body, front axle, rear axle, front wheel, and rear wheel were the following (with units $[10^{-4} kg \cdot m^2]$):

$$\begin{aligned} I_{body} &= \text{diag}\left([227.5,\ 1340,\ 1609]\right) \\ I_{ax,f} &= \text{diag}\left([5.647,\ 0.1613,\ 5.638]\right) \\ I_{ax,r} &= \text{diag}\left([5.748,\ 0.1642,\ 5.739]\right) \\ I_{wh,f} &= \text{diag}\left([0.1777,\ 0.3080,\ 0.1777]\right) \\ I_{wh,r} &= \text{diag}\left([0.1470,\ 0.2547,\ 0.1470]\right) \end{aligned} \tag{A.1}$$

Table A.1: Description and nominal values of robot parameters.

Parameter	**Description**	**Nonlinear Model**	**Linear Model**
α_f $[rad]$	Front tire slip angle	varies	varies
α_r $[rad]$	Rear tire slip angle	varies	varies
b_{roll} $[N{\cdot}m{\cdot}s]$	Total axle roll damping (damp.)	2	n/a
b_t $[N{\cdot}s/m]$	Vert. damping of each tire	1000	90
$C_{\alpha f}$ $[N/rad]$	Linear Model front tire cornering stiffness (stiff.)	n/a	21
$C_{\alpha r}$ $[N/rad]$	Linear Model rear tire cornering stiffness	n/a	50
d_{ee} $[m]$	Manip. arm end effector width	0.05	n/a
F_{roll} $[\cdot]$	Front roll stiff. & damp. dist.	0.55	n/a
h $[m]$	Ground to CG vert. dist.	0.15	0.15
h_{roll} $[m]$	Ground to roll center vert. dist.	0.05	n/a
I_x $[kg{\cdot}m^2]$	Linear Model roll inertia	n/a	0.025
I_z $[kg{\cdot}m^2]$	Linear Model yaw inertia	n/a	0.2
K_D	Arm motor derivative control gain	-70	n/a
K_I	Arm motor integral control gain	-1700	n/a
K_P	Arm motor proportional control gain	-1000	n/a
K_ϕ $[\cdot]$	Manip. arm joint angle const.	varies	varies
k_{roll} $[N{\cdot}m]$	Total axle roll stiffness (front + rear axles)	13	n/a
k_t $[N{\cdot}m]$	Vert. stiffness of each tire	5000	450
L $[m]$	Manip. arm link length	0.5	0.5
ℓ_f $[m]$	CG to front axle long. dist.	0.2	0.2
ℓ_r $[m]$	CG to rear axle long. dist.	0.13	0.13
ℓ $[m]$	Rear to front axle long. dist.	0.33	0.33
m $[kg]$	Total mass (manip. arm + veh.)	3.6	3.5
M_a $[N{\cdot}m]$	Moment on veh. from manip. arm	varies	varies
m_{axle} $[kg]$	Vehicle (veh.) axle mass	0.15	n/a
m_{body} $[kg]$	Vehicle sprung mass	2.5	n/a
m_{ee} $[kg]$	Manip. arm end effector mass	0.5	0.5
m_L $[kg]$	Manip. arm link mass	0.05	n/a
m_v $[kg]$	Total mass of vehicle	3.0	3.0
m_{wh} $[kg]$	Vehicle wheel mass	0.05	n/a
r_{tire} $[m]$	Vehicle tire radius	0.05	n/a
τ_m $[N{\cdot}m]$	Manip. arm motor sat. torques	8	n/a
T $[m]$	CG to tire lat. dist.	0.15	0.15

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Unmanned ground vehicles: Core capabilities & market background. Tech. rep., The Association for Unmanned Vehicle Systems International (AUVSI) (2013). URL `https://goo.gl/EZxVnU`

[2] Administration, N.H.T.S., et al.: Preliminary statement of policy concerning automated vehicles. Washington, DC (2013)

[3] Anand, D., Bhatia, M., Moyne, J., Shahid, W., Tilbury, D.: Wireless test results booklet. Technical report, University of Michigan ERC/RMS (2010)

[4] Andersen, M., Dahl, J., Vandenberghe, L.: Cvxopt: Python software for convex optimization, version 1.1 (2015)

[5] Anderson, S.J., Peters, S.C., Pilutti, T.E., Iagnemma, K.: An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. International Journal of Vehicle Autonomous Systems **8**(2-4), 190–216 (2010)

[6] Anderson, S.J., Walker, J.M., Iagnemma, K.: Experimental performance analysis of a homotopy-based shared autonomy framework. Human-Machine Systems, IEEE Transactions on **44**(2), 190–199 (2014)

[7] Aurenhammer, F.: Voronoi diagramsa survey of a fundamental geometric data structure. ACM Computing Surveys (CSUR) **23**(3), 345–405 (1991)

[8] Bates, D., Maechler, M., Bolker, B., Walker, S.: lme4: Linear mixed-effects models using Eigen and S4. arxiv e-print:1406.5823 (2014). URL `http://arxiv.org/pdf/1406.5823.pdf`

[9] Bates, D.M.: lme4: Mixed-effects modeling with R. URL http://lme4.r-forge.r-project.org/lMMwR/lrgprt.pdf (2010)

[10] Beer, J., Fisk, A.D., Rogers, W.A.: Toward a framework for levels of robot autonomy in human-robot interaction. Journal of Human-Robot Interaction **3**(2), 74 (2014)

[11] Bohren, J., Paxton, C., Howarth, R., Hager, G.D., Whitcomb, L.L.: Semi-autonomous telerobotic assembly over high-latency networks. In: The Eleventh ACM/IEEE International Conference on Human Robot Interation, pp. 149–156. IEEE Press (2016)

[12] Borenstein, J., Koren, Y.: The vector field histogram-fast obstacle avoidance for mobile robots. Robotics and Automation, IEEE Transactions on **7**(3), 278–288 (1991)

[13] Bruemmer, D.J., Few, D.A., Boring, R.L., Marble, J.L., Walton, M.C., Nielsen, C.W.: Shared understanding for collaborative control. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans **35**(4), 494–504 (2005)

[14] Cameron, J.T., Brennan, S.: A comparative, experimental study of model suitability to describe vehicle rollover dynamics for control design. Proceedings of ASME IMECE, Dynamic Systems and Control Division pp. 405–414 (2005)

[15] Carona, R., Aguiar, A.P., Gaspar, J.: Control of unicycle type robots tracking, path following and point stabilization. In: Proceedings of IV Jornadas de Engenharia Electrónica e Telecomunicações e de Computadores, pp. 180–185. Citeseer (2008)

[16] Chen, B.C., Peng, H.: Rollover prevention for sports utility vehicles with human-in-the-loop evaluations. In: 5th Int'l Symp on Advanced Vehicle Control (2000)

[17] Chen, B.C., Peng, H.: Rollover warning for articulated heavy vehicles based on a time-to-rollover metric. Transactions-ASME Journal of Dynamic Systems Measurement and Control **127**(3), 406 (2005)

[18] Chen, J.Y., Haas, E.C., Barnes, M.J.: 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)

[19] Cherian, V., Shenoy, R., Stothert, A., Shriver, J., Ghidella, J., Gillespie, T.D.: Model-based design of a SUV anti-rollover control system. SAE Paper p. 0579 (2008)

[20] Chiou, M., Hawes, N., Stolkin, R., Shapiro, K.L., Kerlin, J.R., Clouter, A.: Towards the principled study of variable autonomy in mobile robots. In: Systems, Man, and Cybernetics (SMC), IEEE International Conference on, pp. 1053–1059. IEEE (2015)

[21] Chipalkatty, R., Droge, G., Egerstedt, M.B.: Less is more: Mixed-initiative model-predictive control with human inputs. Robotics, IEEE Transactions on **29**(3), 695–703 (2013)

[22] Chiu, J.: A simulink model for vehicle rollover prediction and prevention. Master's thesis, Purdue University (2008)

[23] Chiu, J., Solmaz, S., Corless, M., Shorten, R.: A methodology for the design of robust rollover prevention controllers for automotive vehicles using differential braking. Int'l Journal of Vehicle Autonomous Systems **8**(2), 146–170 (2010)

[24] Committee, S.O.R.A.V.S., et al.: Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems (2014)

[25] Corde Lane, J., Carignan, C., Sullivan, B., Akin, D., Hunt, T., Cohen, R.: Effects of time delay on telerobotic control of neutral buoyancy vehicles. In: IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA '02, vol. 3, pp. 2874 –2879 (2002). DOI 10.1109/ROBOT.2002.1013668

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

[27] Deits, R., Tedrake, R.: Computing large convex regions of obstacle-free space through semidefinite programming. In: Algorithmic Foundations of Robotics XI, pp. 109–124. Springer (2015)

[28] Deits, R., Tedrake, R.: Efficient mixed-integer planning for uavs in cluttered environments. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 42–49. IEEE (2015)

[29] Devore, J.: Probability and Statistics for Engineering and the Sciences. Cengage Learning (2011)

[30] Dragan, A.D., Srinivasa, S.S.: A policy-blending formalism for shared control. The International Journal of Robotics Research **32**(7), 790–805 (2013)

[31] Durst, P.J., Goodin, C., Cummins, C., Gates, B., Mckinley, B., George, T., Rohde, M.M., Toschlog, M.A., Crawford, J.: A real-time, interactive simulation environment for unmanned ground vehicles: The autonomous navigation virtual environment laboratory (ANVEL). In: Information and Computing Science (ICIC), Fifth International Conference on, pp. 7–10. IEEE (2012)

[32] E54 Committee: Test method for evaluating emergency response robot capabilities: Mobility: Maneuvering tasks: Sustained speed. Tech. rep., ASTM International (2011). URL `http://enterprise.astm.org/filtrexx40.cgi?+REDLINE_PAGES/E2829.htm`

[33] Endsley, M.R.: Level of automation effects on performance, situation awareness and workload in a dynamic control task. Ergonomics **42**(3), 462–492 (1999)

[34] Enes, A., Book, W.: Blended shared control of zermelo's navigation problem. In: American Control Conference (ACC), 2010, pp. 4307–4312. IEEE (2010)

[35] Erlien, S.M., Funke, J., Gerdes, J.C.: Incorporating non-linear tire dynamics into a convex approach to shared steering control. In: American Control Conference (ACC), 2014, pp. 3468–3473. IEEE (2014)

[36] Finzi, A., Orlandini, A.: Human-robot interaction through mixed-initiative planning for rescue and search rovers. In: Congress of the Italian Association for Artificial Intelligence, pp. 483–494. Springer (2005)

[37] Fitts, P.M.: The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology **47**(6), 381 (1954)

[38] Fox, D., Burgard, W., Thrun, S., et al.: The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine **4**(1), 23–33 (1997)

[39] Goodrich, M.A., Olsen, D.R., Crandall, J., Palmer, T.J.: Experiments in adjustable autonomy. In: Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents, pp. 1624–1629 (2001)

[40] Hashemzadeh, F., Tavakoli, M.: Position and force tracking in nonlinear teleoperation systems under varying delays. Robotica **33**(04), 1003–1016 (2015)

[41] Hauser, K.: Recognition, prediction, and planning for assisted teleoperation of freeform tasks. Autonomous Robots **35**(4), 241–254 (2013)

[42] Helton, W.S., Head, J., Blaschke, B.A.: Cornering law the difficulty of negotiating corners with an unmanned ground vehicle. Human Factors: The Journal of the Human Factors and Ergonomics Society **56**(2), 392–402 (2014)

[43] Hill, J., Fahimi, F.: Active disturbance rejection for walking bipedal robots using the acceleration of the upper limbs. Robotica **33**(02), 264–281 (2015)

[44] Ho, Y.J., Liu, J.S.: Collision-free curvature-bounded smooth path planning using composite bezier curve based on Voronoi diagram. In: Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on, pp. 463–468. IEEE (2009)

[45] Howe, J.G., Garrott, W.R., Forkenbrock, G., Heydinger, G.J., Lloyd, J.: An experimental examination of selected maneuvers that may induce on-road, untripped light vehicle rollover - phase I-A of NHTSA's 1997-1998 vehicle rollover research program. Tech. rep., National Highway Traffic Safety Administration (2001)

[46] Huang, H.M.: Autonomy levels for unmanned systems (alfus) framework: safety and application issues. In: Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems, pp. 48–53. ACM (2007)

[47] Huang, Q., Sugano, S., Kato, I.: Stability control for a mobile manipulator using a potential method. In: Intelligent Robots and Systems. Advanced Robotic Systems and the Real World. Proceedings of the IEEE/RSJ/GI Int'l Conference on, vol. 2, pp. 839–846. IEEE (1994)

[48] Huang, Q., Tanie, K., Sugano, S.: Coordinated motion planning for a mobile manipulator considering stability and manipulation. Int'l Journal of Robotics Research **19**(8), 732–742 (2000)

[49] Hwang, S., Khan, S., Kuo, A., Olsen, T.: Stability control of a teleoperated robot during high speed maneuvers. Technical Report. Video: `http://youtu.be/m_p1tyGe1hs` (2012)

[50] iRobot Corporation: iRobot 510 PackBot - specifications. URL `http://www.irobot.com/us/learn/defense/packbot/Specifications.aspx`

[51] Janabi-Sharifi, F., Hassanzadeh, I.: Experimental analysis of mobile-robot teleoperation via shared impedance control. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **41**(2), 591–606 (2011)

[52] Javdani, S., Srinivasa, S.S., Bagnell, J.A.: Shared autonomy via hindsight optimization. arXiv:1503.07619 (2015)

[53] Jax, S.A., Rosenbaum, D.A., Vaughan, J.: Extending Fitts Law to manual obstacle avoidance. Experimental Brain Research **180**(4), 775–779 (2007)

[54] Jones, K.S., Johnson, B.R., Schmidlin, E.A.: Teleoperation through apertures passability versus driveability. Journal of Cognitive Engineering and Decision Making **5**(1), 10–28 (2011)

[55] Kelly, A., Chan, N., Herman, H., Warner, R.: Experimental validation of operator aids for high speed vehicle teleoperation. In: Experimental Robotics, pp. 951–962. Springer (2013)

[56] Kessler, A.M.: Elon Musk says self-driving Tesla cars will be in the US by summer. The New York Times p. B1 (2015)

[57] Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. The international journal of robotics research **5**(1), 90–98 (1986)

[58] Kim, J., Chung, W.K.: Real-time zero moment point compensation method using null motion for mobile manipulators. Advanced Robotics **20**(5), 581–593 (2006)

[59] Kusnetsova, A., Brockhoff, P.B., Christensen, R.H.B.: lmertest: Tests for random and fixed effects for linear mixed effects models (lmer objects of lme4 package). R package version pp. 2–0 (2013)

[60] Lampe, A., Chatila, R.: Performance measure for the evaluation of mobile robot autonomy. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4057–4062. IEEE (2006)

[61] Lee, D., Martinez-Palafox, O., Spong, M.W.: Bilateral teleoperation of a wheeled mobile robot over delayed communication network. In: Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pp. 3298–3303. IEEE (2006)

[62] Lee, S., Leibold, M., Buss, M., Park, F.C.: Rollover prevention of mobile manipulators using invariance control and recursive analytic ZMP gradients. Advanced Robotics **26**(11-12), 1317–1341 (2012)

[63] Linstromberg, M., Scholpp, G., Scherf, O.: Test and simulation tools in a rollover protection development process. In: Siemens restraint Systems GmbH, ESV Conference, Washington, USA (2005)

[64] Liu, J., Jayakumar, P., Stein, J.L., Ersal, T.: A multi-stage optimization formulation for MPC-based obstacle avoidance in autonomous vehicles using a LIDAR sensor. In: ASME 2014 Dynamic Systems and Control Conference, pp. V002T30A006–V002T30A006. American Society of Mechanical Engineers (2014)

[65] Liu, J., Jayakumar, P., Stein, J.L., Ersal, T.: An MPC algorithm with combined speed and steering control for obstacle avoidance in autonomous ground vehicles. In: ASME 2015 Dynamic Systems and Control Conference, pp. V003T44A003–V003T44A003. American Society of Mechanical Engineers (2015)

[66] Liu, L., van Liere, R.: The effect of varying path properties in path steering tasks. In: Proceedings of the 16th Eurographics conference on Virtual Environments & Second Joint Virtual Reality, pp. 9–16 (2010)

[67] Luck, J.P., McDermott, P.L., Allender, L., Russell, D.C.: 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, pp. 202–209. ACM (2006)

[68] Machairas, K., Papadopoulos, E.: On quadruped attitude dynamics and control using reaction wheels and tails. In: Control Conference (ECC), 2015 European, pp. 753–758. IEEE (2015)

[69] Macharet, D.G., Florencio, D., et al.: A collaborative control system for telepresence robots. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 5105–5111. IEEE (2012)

[70] MacKenzie, I.S., Ware, C.: Lag as a determinant of human performance in interactive systems. In: Proceedings of the INTERACT and CHI Conference on Human Factors in Computing Systems, pp. 488–493. ACM (1993)

[71] Mahony, R., Hamel, T., Pflimlin, J.M.: Nonlinear complementary filters on the special orthogonal group. Automatic Control, IEEE Transactions on **53**(5), 1203–1218 (2008)

[72] Marge, M., Powers, A., Brookshire, J., Jay, T., Jenkins, O.C., Geyer, C.: Comparing heads-up, hands-free operation of ground robots to teleoperation. Robotics: Science and Systems VII p. 193 (2012)

[73] Mattingley, J., Boyd, S.: Cvxgen: a code generator for embedded convex optimization. Optimization and Engineering **13**(1), 1–27 (2012)

[74] McCracken, H.: I Drove Ford's Golf Cart In Atlanta (Note: I Was In Silicon Valley At The Time). Fast Company (2015). URL `http://www.fastcompany.com/3041297/i-drove-fords-golf-cart-in-atlanta-note-i-was-in-silicon-valley-at-the-time`

[75] McWilliams, G.T., Brown, M.A., Lamm, R.D., Guerra, C.J., Avery, P.A., Kozak, K.C., Surampudi, B.: Evaluation of autonomy in recent ground vehicles using the autonomy levels for unmanned systems (ALFUS) framework. In: Proceedings of the Workshop on Performance Metrics for Intelligent Systems, pp. 54–61. ACM (2007)

[76] Metcalfe, J.S., Alban, J., Cosenzo, K., Johnson, T., Capstick, E.: Field testing of tele-operation versus shared and traded control for military assets: an evaluation involving real-time embedded simulation and soldier assessment. In: SPIE Defense, Security, and Sensing, pp. 769,206–769,206. International Society for Optics and Photonics (2010)

[77] Moosavian, S.A.A., Alipour, K.: On the dynamic tip-over stability of wheeled mobile manipulators. Int'l Journal of Robotics and Automation **22**(4), 322–328 (2007)

[78] Nalecz, A., Bindemann, A.: Sensitivity analysis of vehicle design attributes that affect vehicle response in critical accident situations - part 1: user's manual. final report (1987)

[79] Odenthal, D., Bünte, T., Ackermann, J.: Nonlinear steering and braking control for vehicle rollover avoidance. In: Proceedings of European Control Conference (1999)

[80] Pacejka, H.: Tyre and vehicle dynamics. Butterworth-Heinemann (2005)

[81] Papadopoulos, E., Rey, D.A.: The force-angle measure of tipover stability margin for mobile manipulators. Vehicle System Dynamics **33**(1), 29–48 (2000)

[82] Patel, A., Braae, M.: Rapid turning at high-speed: inspirations from the cheetah's tail. In: IROS, pp. 5506–5511. IEEE (2013)

[83] Patel, A., Braae, M.: Rapid acceleration and braking: Inspirations from the cheetah's tail. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 793–799. IEEE (2014)

[84] Peters, S.C., Iagnemma, K.: Stability measurement of high-speed vehicles. Vehicle System Dynamics **47**(6), 701–720 (2009)

[85] Petersen, C., Jaunzemis, A., Baldwin, M., Holzinger, M., Kolmanovsky, I.: Model predictive control and extended command governor for improving robustness of relative motion guidance and control. In: Proc. AAS/AIAA Space Flight Mechanics Meeting (2014)

[86] Rajamani, R.: Vehicle dynamics and control. Springer (2012)

[87] Rockafellar, R.T.: Convex analysis. Princeton university press (2015)

[88] Sanders, D.: Analysis of the effects of time delays on the teleoperation of a mobile robot in various modes of operation. Industrial Robot: An International Journal **36**(6), 570–584 (2009)

[89] Savage-Knepshield, P.: Designing soldier systems: Current issues in human factors. Ashgate Publishing Company (2012)

[90] Sayers, M.: Standard terminology for vehicle dynamics simulations. The University of Michigan Transportation Research Institute (UMTRI), Tech. Rep (1996)

[91] Sheik-Nainar, M.A., Kaber, D.B., Chow, M.Y.: Control gain adaptation in virtual reality mediated human–telerobot interaction. Human Factors and Ergonomics in Manufacturing & Service Industries **15**(3), 259–274 (2005)

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

[93] Shia, V., Gao, Y., Vasudevan, R., Campbell, K.D., Lin, T., Borrelli, F., Bajcsy, R., et al.: Semiautonomous vehicular control using driver modeling. Intelligent Transportation Systems, IEEE Transactions on **15**(6), 2696–2709 (2014)

[94] Shim, T., Ghike, C.: Understanding the limitations of different vehicle models for roll dynamics studies. Vehicle system dynamics **45**(3), 191–216 (2007)

[95] Shuttleworth, M.: Counterbalanced measures design. Explorable.com (2009). URL `http://explorable.com/counterbalanced-measures-design`

[96] Slawiñski, E., Mut, V.: Control scheme including prediction and augmented reality for teleoperation of mobile robots. Robotica **28**(01), 11–22 (2010)

[97] Storms, J., Chen, K., Tilbury, D.: Modeling teleoperated robot driving performance as a function of environment difficulty. In: IFAC Conference on Cyber-Physical & Human-Systems (2016)

[98] Storms, J., Chen, K., Tilbury, D.: A semi-autonomous control method to improve performance of small unmanned ground vehicles with communication latency. In: ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (2016)

[99] Storms, J., Chen, K., Tilbury, D.: A shared control method for obstacle avoidance with mobile robots and its interaction with communication delay. International Journal of Robotics Research **Conditionally Accepted**, 1–18 (2016)

[100] Storms, J., Tilbury, D.: Equating user performance among communication latency distributions and simulation fidelities for a teleoperated mobile robot. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4440–4445. IEEE (2015)

[101] Storms, J., Tilbury, D.: Dynamic weight-shifting for improved maneuverability and rollover prevention in high-speed mobile manipulators. Journal of Dynamic Systems, Measurement, and Control (2016)

[102] Storms, J., Tilbury, D.: A new difficulty index for teleoperated robot driving through obstacles. Journal of Intelligent and Robotic Systems **Submitted**, 1–21 (2016)

[103] Storms, J.G., Tilbury, D.M.: Dynamic weight-shifting to reduce rollover risk in high speed mobile manipulators. In: ASME 2014 Dynamic Systems and Control Conference, pp. V003T48A006–V003T48A006. American Society of Mechanical Engineers (2014)

[104] Takahashi, O., Schilling, R.J.: Motion planning in a plane using generalized Voronoi diagrams. Robotics and Automation, IEEE Transactions on **5**(2), 143–150 (1989)

[105] Takayama, L., Marder-Eppstein, E., Harris, H., Beer, J.M.: Assisted driving of a mobile remote presence system: System design and controlled user evaluation. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1883–1889. IEEE (2011)

[106] Talke, K., Kelley, L., Longhini, P., Catron, G.: Tip-over prevention through heuristic reactive behaviors for unmanned ground vehicles. In: SPIE Defense+ Security, pp. 90,840L–90,840L. International Society for Optics and Photonics (2014)

[107] Vaughan, J., Barany, D.A., Sali, A.W., Jax, S.A., Rosenbaum, D.A.: Extending Fitts Law to three-dimensional obstacle-avoidance movements: support for the posture-based motion planning model. Experimental Brain Research **207**(1-2), 133–138 (2010)

[108] Vozar, S.: A framework for improving the speed and performance of teleoperated mobile manipulators. Ph.D. dissertation, University of Michigan, Ann Arbor (2013)

[109] Vozar, S., Storms, J., Tilbury, D.: Development and analysis of an operator steering model for teleoperated mobile robots under constant and variable latencies. Robotica pp. 1–20 (2016)

[110] Vozar, S., Tilbury, D.: Driver modeling for teleoperation with time delay. In: Proceedings of the 19th IFAC World Congress, pp. 3551–3556 (2014)

[111] Wang, B., Li, Z., Ding, N.: Speech control of a teleoperated mobile humanoid robot. In: IEEE International Conference on Automation and Logistics (ICAL), pp. 339–344 (2011). DOI 10.1109/ICAL.2011.6024739

[112] Wood, G.D., Kennedy, D.C.: Simulating mechanical systems in Simulink with SimMechanics. The Mathworks Report (2003)

[113] Xiong, Y., Li, S., Xie, M.: Predictive display and interaction of telerobots based on augmented reality. Robotica **24**(04), 447–453 (2006)

[114] Xu, Y., Yu, C., Li, J., Liu, Y.: Video telephony for end-consumers: measurement study of Google+, iChat, and Skype. In: Proceedings of the ACM conference on Internet Measurement Conference, pp. 371–384. ACM (2012)

[115] Yamauchi, B.: Driver assist behaviors for high-speed small UGVs. In: SPIE Defense, Security, and Sensing, vol. 8045 (2011)

[116] Zhai, S., Accot, J., Woltjer, R.: 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 (2004)